Nonlinear Dyn (2020) 101:875–891
https://doi.org/10.1007/s11071-020-05847-5

ORIGINAL PAPER

Damage modeling and detection for a tree network using fractional-order calculus

Xiangyu Ni · **Bill Goodwine**

Received: 4 April 2020 / Accepted: 25 July 2020 / Published online: 5 August 2020

Abstract Large networks are increasingly common in engineered systems, and therefore, monitoring their operating conditions is increasingly important. This paper proposes a model-based frequency-domain damage detection method for an infinitely large self-similar network. The first aim is to exactly model the overall frequency response for any specific damage case of that network, which we show has an explicit multiplicative relation to the undamaged transfer function. Then, leveraging that knowledge from modeling, this paper also proposes an algorithm to identify damaged components within that network as well as quantifies their respective damage amounts given a noisy measurement for that network's overall frequency response.

Keywords Mathematical modeling · Damage identification · Network systems · Fractional calculus

The support of the US National Science Foundation under Grant No. CMMI 1826079 is gratefully acknowledged.

X. Ni (✉) · B. Goodwine
Aerospace and Mechanical Engineering, University of Notre Dame, 365 Fitzpatrick Hall, Notre Dame, IN 46556, USA
e-mail: xni@nd.edu

B. Goodwine
e-mail: billgoodwine@nd.edu

1 Introduction

Network systems exist everywhere in real life, for instance, ventilating systems, plumbing networks, natural and robotic swarms, and power grids. Controlling and monitoring the operational health of such large networks are research topics which have a long history. For examples of controlling networks, see the survey paper [27], the book [29] and the paper [31] for multi-vehicle cooperative control, and the papers [2,7,11,30] for formation control. The convergence speed of a consensus behavior within scale-free networks is studied in [35]. Health monitoring of large networks and structures is another crucial consideration in modern industry. Some related research can be seen in the survey paper [39]. Different types of health monitoring methods have been proposed in [20,32,34]. One class of approaches uses system identification to monitor a system's health as illustrated in [6,8,18,28]. The damage detection method proposed in this paper belongs to that class.

In addition to those real networks, many researches are also conducted on hypothetical networks which are infinitely large and perfectly self-similar. Those researches take advantage of the fact that those idealized networks have less computational burden to model as opposed to real large-scale yet finite networks [15]. Moreover, as stated in [21,26], infinitely large networks' behavior is good approximations for them. In light of those considerations, we also choose those ide-

alized networks as the starting point of our work. Other real networks which are modeled as infinite networks include bio-systems in human body. Fractal models of human blood vessels using tree-like structures are proposed in [14]. Fractal analysis for the vascular tree in human retina is studied in [25].

The dynamics of an infinite network naturally leads to fractional-order differential equations. As a result, along with infinite networks mentioned above, this research also has a potential impact on the systems governed by fractional-order differential equations. Some examples of using infinite networks to model fractional-order systems can be seen in the following literature. The paper [16] proposes that the network shown in Fig. 1, which we call the tree network, can be a rheological model of viscoelastic behavior. A ladder network is used to model the lung impedance in [17]. The fact that a fractional-order viscoelastic model like Fig. 1 can represent 1D relaxation of the aortic valve is shown in [13].

Other researches regarding fractional-order systems mainly leverage some intrinsic properties of fractional-order derivatives to achieve a better model for complicated systems. First of all, fractional-order derivatives are non-local. The corresponding applications include modeling epidemics as shown in [1]. A physically based approach to non-local elasticity theory is introduced in [12]. The fact that fractional-order dynamics exist in non-local heat transfer and mechanics is shown in [4] and [10]. Moreover, the time-domain response for a linear fractional-order system can follow a power law decay rate, which also leads to many applications. Chapter 1 in [23] shows a modeling example of the firing rate for premotor neurons in the visual system while an eyeball is scanning words. A fractal network model to describe a power law behavior in soft tissue is proposed in [19].

This paper studies the infinite network as shown in Fig. 1, motivated by a viscoelastic model from [16]. The goal is twofold. On the one hand, we want to exactly model its frequency response when it is damaged. On the other hand, we aim at detecting the damaged components and estimating their damage amounts given a noisy measurement of its frequency response.

In the damage modeling part, we propose a recursive damage modeling algorithm which returns the closed-form expression of that damaged network's fractional-order transfer function. More importantly, we discover that any damaged transfer function can be explicitly written down as a multiplicative disturbance imposed upon the undamaged transfer function. That multiplicative model is classical in the robust control research area, and thus, the damage modeling result of this paper also has potential impacts on that.

In the damage identification part, we propose a model-based diagnostic procedure, which is an application of our novel modeling method mentioned above. That diagnostic procedure is non-trivial as it is applied to such a large network characterized by a fractional-order dynamics. Moreover, the fact that the diagnostic procedure is model-based brings two advantages. First, it is able to detect the damaged components and estimate their damage amounts, which is more challenging and meaningful for the health monitoring purpose as opposed to merely detecting the damage occurrence [33]. Second, its identification result is more robust to the measurement noise, thus having potential to be employed in real damage detection scenarios.

Some of the second author's previous work, such as [22], set the similar goal. However, there are three noticeable improvements of this paper. First, this work obtains the explicit expression for the damaged transfer function, which can be related to the undamaged transfer function in the closed form. That explicit relation has never been revealed before. Second, the work in [22] only attempted to identify damage by looking into the changes of the system's order. In contrast, this paper uses exact modeling of damage cases in the damage detection algorithm which is a superior approach in the sense that it can now locate the damaged components. Third, the damage detection method proposed in this paper is tested against noisy frequency response measurements, whereas previous work always assumed a perfect measurement.

Although this paper only focuses on one specific network shown in Fig. 1, our initial analysis shows that it has potential to be generalized to a broader class of networks, which is discussed in Sect. 6. For that class of networks, their damaged transfer functions can also be modeled in a manner similar to the damaged tree model [Eq. (5)]. Then, the damage identification proposed here can also be applied to that class of networks.

The rest of this paper is organized as follows. Section 2 defines the tree model, reviews existing literature about its recurrence formula, and about how to compute its undamaged transfer function. Extending from the undamaged version, Sect. 3 presents the first main result of this paper which is about exactly mod-

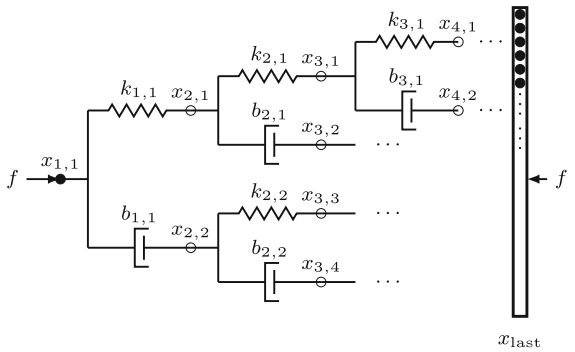

Fig. 1 Mechanical diagram of the tree model

eling damaged infinite trees. Then, Sect. 4 shows the other main result which proposes a damage identification algorithm for the infinite tree by leveraging the knowledge from the damage modeling introduced in Sect. 3. In addition, it also shows the test results for the proposed identification algorithm. Section 5 discusses the effects on both the proposed damage modeling and the damage identification algorithms brought by deep damages inside the tree model. Finally, Sect. 6 concludes this paper and discusses future research plans in regard to this topic.

2 Background and literature review

2.1 The tree model

Figure 1 shows the tree model, which has an infinite number of generations. Proceeding to the right, at each generation, the number of nodes is doubled compared to the one on its left. For each pair of two nodes, the upper one is connected to its left node through a linear spring, while the lower one is connected to its left node through a linear damper. As shown in Fig. 1, $x_{g,n}$ denotes the displacement of the node (g, n) which is the n-th node on the g-th generation. All nodes on the last generation are locked together, whose displacement is represented by x_{last}. In addition, $k_{g,n}$ denotes the spring to the right of the node (g, n), and $b_{g,n}$ denotes the damper to the right of the node (g, n).

The input of interest for the tree model is the force applied to its ends, that is f in Fig. 1. The output is the relative displacement between both ends, that is $x_{1,1} - x_{\text{last}}$. Therefore, unless stated otherwise, all transfer functions for the tree model in this paper refer to that system, *i.e.*,

$$G(s) = \frac{X_{1,1}(s) - X_{\text{last}}(s)}{F(s)}.$$

The tree model is called undamaged when all springs have the same constant k, and all dampers have the same constant b. Otherwise, we say the tree model is damaged.

It is worth noting that the components used in this work are all idealized models. More complicated networks can be obtained by using more sophisticated models, such as the models which can accurately simulate hysteretic phenomena of materials exhibiting either kinetic hardening or softening [5,36–38]. However, that is beyond the scope of our current work because we are more interested in the interaction of the components within large-scale networks.

2.2 Self-similarity for the tree model

Existing literature states that the transfer function of the entire tree model can be derived by focusing on its first generation and by taking advantage of the tree model's self-similarity (the definition of self-similarity is discussed in [24].) Suppose that $G_U(s)$ and $G_L(s)$ are two transfer functions representing the two sub-trees between the second generation and the last generation, that is,

$$G_U(s) = \frac{X_{2,1}(s) - X_{\text{last}}(s)}{F_{2,1}(s)},$$
$$G_L(s) = \frac{X_{2,2}(s) - X_{\text{last}}(s)}{F_{2,2}(s)},$$
$$F_{2,1}(s) + F_{2,2}(s) = F(s).$$

Then, the entire tree model can be illustrated by Fig. 2, from which, using series and parallel relations for mechanical components, we can conclude that the transfer function for the entire tree model can be written as

$$G(s) = \frac{1}{\dfrac{1}{\dfrac{1}{k_{1,1}} + G_U(s)} + \dfrac{1}{\dfrac{1}{b_{1,1}s} + G_L(s)}}. \tag{1}$$

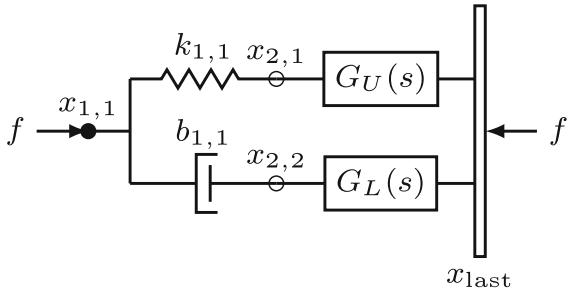

Fig. 2 An illustration about the tree model where the sub-networks after the first generation are represented by their transfer functions $G_U(s)$ and $G_L(s)$

Equation (1) is called the *recurrence formula* for the tree model, as it transforms two transfer functions of two sub-trees, $G_U(s)$ and $G_L(s)$, to the transfer function of the entire tree, $G(s)$. Note that the recurrence formula (1) works for all trees no matter damaged or not. Only the explicit expressions for $G(s)$, $G_U(s)$, $G_L(s)$, $k_{1,1}$, and $b_{1,1}$ depend on the actual state.

Due to self-similarity, both $G_U(s)$ and $G_L(s)$ should have formulations similar to Eq. (1). Therefore, when integer-order calculus can be used only, the transfer function for the entire tree model inevitable becomes a complicated infinite continued fraction, which consists of unlimited copies of the recurrence formula (1):

$$G(s) = \left[\cfrac{1}{\cfrac{1}{k_{1,1}} + \cfrac{1}{\cfrac{1}{\cfrac{1}{k_{2,1}} + \ddots} + \cfrac{1}{\cfrac{1}{b_{2,1}s} + \ddots}}} + \cfrac{1}{\cfrac{1}{b_{1,1}s} + \cfrac{1}{\cfrac{1}{\cfrac{1}{k_{2,2}} + \ddots} + \cfrac{1}{\cfrac{1}{b_{2,2}s} + \ddots}}}} \right]^{-1}. \tag{2}$$

Existing literature shows that when fractional-order calculus is also allowed, the transfer function for the undamaged tree model actually has a very concise representation, which is discussed next. Furthermore, the work in this paper shows that even when the tree model is damaged, its transfer function is still well structured and much simpler than the infinite continued fraction (2).

2.3 Transfer function for the undamaged tree

As defined in Sect. 2.1, for the undamaged tree, all springs (dampers) have the same constant k (b). Therefore, due to self-similarity, it is obvious that those two sub-networks $G_U(s)$ and $G_L(s)$ are both undamaged, and thus are exactly same as the entire undamaged tree model. That is, if $G_\infty(s)$ denotes the transfer function for the entire undamaged tree model, then

$$G_U(s) = G_L(s) = G_\infty(s).$$

In addition, we also know that both the spring $k_{1,1}$ and the damper $b_{1,1}$ are undamaged. Therefore, $k_{1,1} = k$ and $b_{1,1} = b$. Then, for the undamaged tree, the recurrence formula (1) becomes

$$G_\infty(s) = \frac{1}{\cfrac{1}{\cfrac{1}{k} + G_\infty(s)} + \cfrac{1}{\cfrac{1}{bs} + G_\infty(s)}}, \tag{3}$$

in which $G_\infty(s)$ is the only unknown. Solving Eq. (3) for $G_\infty(s)$, we can see that the undamaged tree model is exactly half-order:

$$G_\infty(s) = \frac{1}{\sqrt{kbs}}. \tag{4}$$

By comparing Eq. (4) with (2), we see that fractional-order calculus allows us to write down a much simpler transfer function for such a large network.

The correctness of Eq. (4) is assessed by the convergence of the undamaged tree's response as the number of generations increases to infinity. That convergence was showed both in the frequency domain and in the time domain by the second author's previous work [15].

3 Modeling damaged trees

In this section, we show that fractional-order calculus still allows us to write down a more concise transfer function even when the tree model is damaged. The problem we solve in this section is a forward problem,

where the damage information for the entire tree model is assumed to be known, and we show how to obtain its transfer function given that damage information. First, in Sect. 3.1, we prove the structure of that transfer function which is non-integer order and is much more concise than Eq. (2). The proof then lays the foundation for a recursive algorithm to compute that transfer function for a specific damage case, which is proposed in Sect. 3.2.

3.1 The structure of a damaged tree's transfer function

In this section, we show that a damaged tree model has a particularly simple multiplicative form, namely

$$G(s) = G_\infty(s)\Delta(s). \tag{5}$$

As defined in Sect. 2.1, for a damaged tree model, some springs and/or dampers have their constants different from the undamaged values, *i.e.*, k and b. In addition, we assume that all damages enter each component in a multiplicative way. That is, a damaged spring (damper) has a damage amount ε when its constant is $k\varepsilon$ ($b\varepsilon$) instead of k (b). Note that, by doing so, a damage case for the tree model can be completely described by a pair of two lists, $(\boldsymbol{l}, \boldsymbol{\epsilon})$, where the list $\boldsymbol{l}$ includes all damaged components and the list $\boldsymbol{\epsilon}$ includes their corresponding damage amounts. For example,

$$(\boldsymbol{l}, \boldsymbol{\epsilon}) = ([k_{1,1}, b_{2,2}, k_{3,3}], [0.1, 0.2, 0.3])$$

represents the damage case where $k_{1,1} = 0.1k$, $b_{2,2} = 0.2b$, $k_{3,3} = 0.3k$, while all the other springs' (dampers') constants stay at k (b).

Theorem 1 *The transfer function for any damaged trees has the following structure:*

$$\begin{aligned} G(s) &= G_\infty(s)\frac{s^n + c_{N,1}s^{n-\frac{1}{2}} + \cdots + c_{N,2n}}{s^n + c_{D,1}s^{n-\frac{1}{2}} + \cdots + c_{D,2n}} \\ &= G_\infty(s)\frac{\boldsymbol{c_N}^\top \boldsymbol{\mu}(s;n)}{\boldsymbol{c_D}^\top \boldsymbol{\mu}(s;n)}, \end{aligned} \tag{6}$$

where $G_\infty(s)$ is the undamaged tree model's transfer function as defined by Eq. (4). Moreover, $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ are two coefficient vectors in $\mathbb{R}^{2n+1}$,

$$\boldsymbol{c_N} = \left[1\ c_{N,1}\ \ldots\ c_{N,2n}\right]^\top,$$
$$\boldsymbol{c_D} = \left[1\ c_{D,1}\ \ldots\ c_{D,2n}\right]^\top,$$

and $\boldsymbol{\mu}(s;n)$ is a vector of univariate monomial basis of s whose order decreases from n to 0 with a decrement of $1/2$, i.e., $\boldsymbol{\mu}(s;n) = \left[s^n\ s^{n-\frac{1}{2}}\ \ldots\ s^{\frac{1}{2}}\ 1\right]^\top$.

There are three non-trivial properties in Eq. (6) which we are going to show in the proof later.

Property 1 The monomial basis $\boldsymbol{\mu}(s;n)$ of the numerator is always same as that of the denominator.

Property 2 The coefficient of the highest order is always 1 for both the numerator and the denominator, that is the first element of both $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ is always 1.

Property 3 Both coefficients of the lowest order are always same for the numerator and the denominator, that is $c_{N,2n} = c_{D,2n}$.

Note that the structure of Eq. (6) holds for all trees, and it is independent of the specific damage case for a tree. Only the values for $\boldsymbol{c_N}$, $\boldsymbol{c_D}$ and n depend on the specific damage case. In fact, it even holds for the undamaged tree where $\boldsymbol{c_N} = \boldsymbol{c_D} = 1$ and $n = 0$. A similar idea is also true for the recurrence formula (1), which holds no matter whether a tree is damaged or not and no matter how much damage that tree is bearing. Only the expressions for $G(s), G_U(s), G_L(s), k_{1,1}$, and $b_{1,1}$ in the recurrence formula (1) rely on a specific state. For example, as we saw in Eq. (3), when undamaged, $G(s) = G_U(s) = G_L(s) = G_\infty(s)$, $k_{1,1} = k$, and $b_{1,1} = b$.

To determine the $n, \boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in Eq. (6) for a specific damage case, we need the corresponding $G_U(s)$ and $G_L(s)$ for that damage case, which themselves are also trees due to self-similarity. Therefore, we can always assume both of them have structures same as Eq. (6), that is

$$G_U(s) = G_\infty(s)\frac{\boldsymbol{c_{N_U}}^\top \boldsymbol{\mu}(s;n_U)}{\boldsymbol{c_{D_U}}^\top \boldsymbol{\mu}(s;n_U)},$$
$$G_L(s) = G_\infty(s)\frac{\boldsymbol{c_{N_L}}^\top \boldsymbol{\mu}(s;n_L)}{\boldsymbol{c_{D_L}}^\top \boldsymbol{\mu}(s;n_L)}.$$

Moreover, we also know the damage amounts for $k_{1,1}$ and $b_{1,1}$ since the damage information for the entire tree is assumed to be known in the modeling problem. Therefore, we can suppose that the damage amount for $k_{1,1}$ is ε_k, and that for $b_{1,1}$ is ε_b ($\varepsilon_b \neq 0$), that is,

$$k_{1,1} = k\varepsilon_k,$$
$$b_{1,1} = b\varepsilon_b.$$

Then, we have the following results for the n, $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in Eq. (6):

Theorem 2 *The n, $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in Eq. (6) can be computed through the following three equations given the expressions of $G_U(s)$ and $G_L(s)$.*

$$n = n_U + n_L + 1, \tag{7}$$

$$\begin{aligned}\boldsymbol{c_N} = &\left[\boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}}\ 0\ 0\right]^\top \\ &+ \left[0\ \sqrt{\tfrac{k}{b}}\left(\varepsilon_k \boldsymbol{c_{N_U}} * \boldsymbol{c_{N_L}} + \tfrac{1}{\varepsilon_b}\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}\right) 0\right]^\top \\ &+ \left[0\ 0\ \tfrac{k\varepsilon_k}{b\varepsilon_b}\boldsymbol{c_{N_U}} * \boldsymbol{c_{D_L}}\right]^\top,\end{aligned} \tag{8}$$

$$\begin{aligned}\boldsymbol{c_D} = &\left[\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}\ 0\ 0\right]^\top \\ &+ \left[0\ \sqrt{\tfrac{k}{b}}\varepsilon_k(\boldsymbol{c_{N_U}} * \boldsymbol{c_{D_L}} + \boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}})\ 0\right]^\top \\ &+ \left[0\ 0\ \tfrac{k\varepsilon_k}{b\varepsilon_b}\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}\right]^\top,\end{aligned} \tag{9}$$

where the operator $*$ denotes the convolution of two vectors.

As we will see later, Theorem 2 forms the core of the induction step in the following proof and also the core of our damage modeling algorithm to compute Eq. (6) for a specific damage case, which is proposed in Sect. 3.2.

Note that the conclusion addressed here cannot be applied to the case where the damage amount for some damper is 0 as inferred by the fact that ε_b appears on the denominator in Eqs. (8) and (9). In such a case, the tree model is completely disconnected at some internal nodes.

The proof here follows mathematical induction.

Proof Base Case: The base case, although trivial, is the undamaged tree. For that case, since $G(s) = G_\infty(s)$, Eq. (6) trivially holds with $\boldsymbol{c_N} = \boldsymbol{c_D} = 1$, and $n = 0$, *i.e.*, the monomial basis is $\boldsymbol{\mu}(s;0) = s^0 = 1$. Note that Properties 1 to 3 for Eq. (6) also hold trivially for this base case.

Induction Step: Suppose that Eq. (6) holds for all damage cases where damages do not happen deeper than the generation g, and Properties 1 to 3 are also satisfied. We are going to show that Eq. (6) also holds for any damage case where damages do not happen deeper than the generation $g+1$, and meanwhile Properties 1 to 3 stay invariant. Note that the base case where the tree is undamaged can be viewed as $g = 0$.

For a damage case where damages happen only in first $g+1$ generations, due to self-similarity, those two sub-networks $G_U(s)$ and $G_L(s)$ have to be two damaged tree models whose damages can only happen in first g generations. Then, according to the assumption for the induction step, Eq. (6) holds for both $G_U(s)$ and $G_L(s)$. Therefore, we can assume that

$$G_U(s) = G_\infty(s)\frac{N_U(s)}{D_U(s)} = G_\infty(s)\frac{\boldsymbol{c_{N_U}}^\top \boldsymbol{\mu}(s;n_U)}{\boldsymbol{c_{D_U}}^\top \boldsymbol{\mu}(s;n_U)},$$
$$G_L(s) = G_\infty(s)\frac{N_L(s)}{D_L(s)} = G_\infty(s)\frac{\boldsymbol{c_{N_L}}^\top \boldsymbol{\mu}(s;n_L)}{\boldsymbol{c_{D_L}}^\top \boldsymbol{\mu}(s;n_L)},$$

where

$$\begin{aligned}\boldsymbol{c_{N_U}} &= \left[1\ c_{N_U,1}\ \ldots\ c_{N_U,2n_U}\right]^\top, \\ \boldsymbol{c_{D_U}} &= \left[1\ c_{D_U,1}\ \ldots\ c_{D_U,2n_U}\right]^\top, \\ \boldsymbol{\mu}(s;n_U) &= \left[s^{n_U}\ s^{n_U-\frac{1}{2}}\ \ldots\ 1\right]^\top, \\ \boldsymbol{c_{N_L}} &= \left[1\ c_{N_L,1}\ \ldots\ c_{N_L,2n_L}\right]^\top, \\ \boldsymbol{c_{D_L}} &= \left[1\ c_{D_L,1}\ \ldots\ c_{D_L,2n_L}\right]^\top, \\ \boldsymbol{\mu}(s;n_L) &= \left[s^{n_L}\ s^{n_L-\frac{1}{2}}\ \ldots\ 1\right]^\top.\end{aligned}$$

Note that both $G_U(s)$ and $G_L(s)$ are supposed to satisfy Properties 1 to 3 for Eq. (6) here according to the assumption for this induction step. In addition, we assume that the damages happening on the first generation are such that $k_{1,1} = k\varepsilon_k$ and $b_{1,1} = b\varepsilon_b$ ($\varepsilon_b \neq 0$). Note that this assumption includes those cases where $k_{1,1}$ and/or $b_{1,1}$ are undamaged because ε_k and/or ε_b can be 1.

Substituting those elements into Eq. (1), we have

$$G(s) = \frac{1}{\dfrac{1}{\dfrac{1}{k\varepsilon_k} + G_\infty(s)\dfrac{N_U(s)}{D_U(s)}} + \dfrac{1}{\dfrac{1}{b\varepsilon_b s} + G_\infty(s)\dfrac{N_L(s)}{D_L(s)}}}.$$

After simplification, the above equation leads to

$$G(s) = G_\infty(s)\frac{N(s)}{D(s)}, \tag{10}$$

where

$$\begin{aligned} N(s) &= N_L(s)D_U(s)s \\ &\quad + \sqrt{\frac{k}{b}}\left(\varepsilon_k N_U(s)N_L(s) + \frac{1}{\varepsilon_b}D_U(s)D_L(s)\right)s^{\frac{1}{2}} \\ &\quad + \frac{k\varepsilon_k}{b\varepsilon_b}N_U(s)D_L(s), \\ D(s) &= D_U(s)D_L(s)s \\ &\quad + \sqrt{\frac{k}{b}}\varepsilon_k\left(N_U(s)D_L(s) + N_L(s)D_U(s)\right)s^{\frac{1}{2}} \\ &\quad + \frac{k\varepsilon_k}{b\varepsilon_b}D_U(s)D_L(s). \end{aligned}$$

Then, we can obtain Eqs. (8) and (9). For example,

$$\begin{aligned} N_L(s) &= \boldsymbol{c_{N_L}}^\top \boldsymbol{\mu}(s; n_L), \\ D_U(s) &= \boldsymbol{c_{D_U}}^\top \boldsymbol{\mu}(s; n_U), \\ s &= \begin{bmatrix}1 & 0 & 0\end{bmatrix}\boldsymbol{\mu}(s; 1), \end{aligned}$$

then, the $N_L(s)D_U(s)s$ in the numerator $N(s)$ of Eq. (10) becomes

$$\begin{aligned} &N_L(s)D_U(s)s \\ &= (\boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}} * \begin{bmatrix}1 & 0 & 0\end{bmatrix}^\top) \cdot \boldsymbol{\mu}(s; n_U + n_L + 1) \\ &= \begin{bmatrix}\boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}} & 0 & 0\end{bmatrix}\boldsymbol{\mu}(s; n_U + n_L + 1), \end{aligned}$$

which corresponds to the first part of $\boldsymbol{c_N}$ in Eq. (8).

Note that those Properties 1 to 3 of Eq. (6) still hold here in Eq. (10), which is equivalent to Eqs. (8) and (9):

1. Those vector convolutions in both Eq. (8) and (9) always take one coefficient vector from $G_U(s)$ and the other one from $G_L(s)$. In our assumption for this induction step, we see that both coefficient vectors $\boldsymbol{c_{N_U}}$ and $\boldsymbol{c_{D_U}}$ from $G_U(s)$ are of length $2n_U+1$, and both coefficient vectors $\boldsymbol{c_{N_L}}$ and $\boldsymbol{c_{D_L}}$ from $G_L(s)$ are of length $2n_L + 1$. Therefore, all vector convolutions in Eqs. (8) and (9) have the same length $2n_U + 2n_L + 1$, which leads to the fact that both $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ are of length $2n_U + 2n_L + 3$. As a result, the coefficient vectors $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ for the numerator and denominator of $G(s)$ correspond to the same monomial basis $\boldsymbol{\mu}(s; n_U + n_L + 1)$, which also explains Eq. (7).
2. Equations (8) and (9) show that the first element in $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ is determined only by the first element in $\boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}}$ and $\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}$, respectively. Based on our assumption for this induction step, we know that the first element for $\boldsymbol{c_{D_U}}$, $\boldsymbol{c_{N_L}}$ and $\boldsymbol{c_{D_L}}$ is all 1. Therefore, the first element of both $\boldsymbol{c_{N_L}} * \boldsymbol{c_{D_U}}$ and $\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}$ has to be 1, so as that of both $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$.
3. Similar to the above argument, Eqs. (8) and (9) show that the last element in $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ is determined only by the last element in $\frac{k\varepsilon_k}{b\varepsilon_b}\boldsymbol{c_{N_U}} * \boldsymbol{c_{D_L}}$ and $\frac{k\varepsilon_k}{b\varepsilon_b}\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}$, respectively. Based on our assumption for this induction step, we know that the last element of $\boldsymbol{c_{N_U}}$ is same as that of $\boldsymbol{c_{D_U}}$. Therefore, the last element of both $\boldsymbol{c_{N_U}} * \boldsymbol{c_{D_L}}$ is same as that of $\boldsymbol{c_{D_U}} * \boldsymbol{c_{D_L}}$, which means that the last element of $\boldsymbol{c_N}$ is same as that of $\boldsymbol{c_D}$.

□

The above proof treats the undamaged case as the starting point to prove all damaged cases which may seem suspicious. However, it can be shown that the cases where damages only occur at the first generation are actually one induction step from the undamaged case. In fact, following the same argument of the induction step in the above proof, when damages only happen on the first generation, those two sub-networks represented by $G_U(s)$ and $G_L(s)$ in Fig. 2 are both undamaged, that is $g = 0$ in the language of the above proof. As a result,

$$G_U(s) = G_L(s) = G_\infty(s).$$

Then, when damages only happen at the first generation, Eq. (1) becomes

$$G(s) = \frac{1}{\dfrac{1}{\dfrac{1}{k\varepsilon_k} + G_\infty(s)} + \dfrac{1}{\dfrac{1}{b\varepsilon_b s} + G_\infty(s)}}.$$

After simplification, this leads to

$$G(s) = G_\infty(s)\frac{s + \sqrt{\frac{k}{b}}\left(\varepsilon_k + \frac{1}{\varepsilon_b}\right)s^{\frac{1}{2}} + \frac{k\varepsilon_k}{b\varepsilon_b}}{s + 2\varepsilon_k\sqrt{\frac{k}{b}}s^{\frac{1}{2}} + \frac{k\varepsilon_k}{b\varepsilon_b}}.$$

Hence, Eq. (6) holds in this case with

$$n = 1,$$
$$\boldsymbol{c_N} = \left[1 \ \sqrt{\tfrac{k}{b}}\left(\varepsilon_k + \tfrac{1}{\varepsilon_b}\right) \ \tfrac{k\varepsilon_k}{b\varepsilon_b}\right]^\top,$$
$$\boldsymbol{c_D} = \left[1 \ 2\varepsilon_k\sqrt{\tfrac{k}{b}} \ \tfrac{k\varepsilon_k}{b\varepsilon_b}\right]^\top.$$

The above result satisfies Eqs. (7) to (9) in Theorem 2 with $n_U = n_L = 0$ and $c_{N_U} = c_{D_U} = c_{N_L} = c_{D_L} = 1$. Therefore, this case is indeed one induction step from the undamaged case. Note that Properties 1 to 3 for Eq. (6) also hold for this case.

3.2 A recursive algorithm to model damaged trees

In Sect. 3.1, we show the structure for the damaged tree's transfer function $G(s)$, which holds for all trees. In this section, we propose an algorithm to compute the expression of that transfer function for a specific damage case $(\boldsymbol{l}, \boldsymbol{\epsilon})$ which is denoted by $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$. Namely, we want to determine the coefficient vectors $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ given a damage case $(\boldsymbol{l}, \boldsymbol{\epsilon})$. Note that n and the completed expression of $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ can be determined immediately from those two coefficient vectors.

The core of our algorithm is Eqs. (8) and (9), which show how to compute $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$. Note that Eqs. (8) and (9) rely on two things. First, they require damage information at the first generation, that is ε_k and ε_b in Eqs. (8) and (9). Second, they need knowledge about $G_U(s)$ and $G_L(s)$ whose coefficient vectors $\boldsymbol{c_{N_U}}, \boldsymbol{c_{D_U}}, \boldsymbol{c_{N_L}}$, and $\boldsymbol{c_{D_L}}$ are in the formula. As a result, the following is the basic idea of our modeling algorithm. Upon receiving the damage case $(\boldsymbol{l}, \boldsymbol{\epsilon})$, the algorithm first divides it into three parts, where $(\boldsymbol{l_1}, \boldsymbol{\epsilon_1})$ is the damage information at the first generation, $(\boldsymbol{l_U}, \boldsymbol{\epsilon_U})$ is the damage information for the upper sub-tree represented by $G_U(s)$, and $(\boldsymbol{l_L}, \boldsymbol{\epsilon_L})$ is the damage information for the lower sub-tree represented by $G_L(s)$. Then, $(\boldsymbol{l_1}, \boldsymbol{\epsilon_1})$ is processed to obtain ε_k and ε_b which are passed directly to Eqs. (8) and (9). Next, both $(\boldsymbol{l_U}, \boldsymbol{\epsilon_U})$ and $(\boldsymbol{l_L}, \boldsymbol{\epsilon_L})$ are passed to our modeling algorithm twice recursively to compute their respective coefficient vectors $\boldsymbol{c_{N_U}}, \boldsymbol{c_{D_U}}, \boldsymbol{c_{N_L}}$, and $\boldsymbol{c_{D_L}}$. After that, Eqs. (8) and (9) have everything they need, so the value of the coefficient vectors $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ can be returned.

Table 1 Pseudocode for our damage modeling algorithm

Compute the coefficient vectors $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ given a specific damage case (`l`,`e`) and the undamaged constants `k`, `b` for the tree model.

```
[cN,cD] = modelAlg(l,e,k,b)
if isEmpty(l)
  cN=1;
  cD=1;
else
  [l1,e1,lU,eU,lL,eL] = partition(l,e);
  [eK,eB] = determineEKEB(l1,e1);
  [cNU,cDU] = modelAlg(lU,eU,k,b);
  [cNL,cDL] = modelAlg(lL,eL,k,b);
  [cN,cD] = merge(k,b,eK,eB,cNU,cDU,cNL,cDL);
end
```

The pseudocode for our damage modeling algorithm is formally listed in Table 1. Since our modeling algorithm has a recursive nature, it requires a base case to terminate, which is determined by the `if` condition. As discussed in the proof in Sect. 3.1, the base case is the undamaged tree, which is thus characterized by an empty list of damaged components $\boldsymbol{l}$. Therefore, for this base case, our algorithm directly returns $c_N = c_D = 1$.

As a concrete example, suppose we want to compute the transfer function, $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$, for the damage case

$$(\boldsymbol{l}, \boldsymbol{\epsilon}) = ([k_{1,1}, b_{2,1}, k_{3,2}, b_{3,3}], [0.1, 0.2, 0.3, 0.4]), \tag{11}$$

when the undamaged constants are $k = 2N/m$ and $b = 1N \cdot s/m$. Then, we should call our modeling algorithm with the following arguments

$$\mathtt{l} = [k_{1,1}, b_{2,1}, k_{3,2}, b_{3,3}],$$
$$\mathtt{e} = [0.1, 0.2, 0.3, 0.4],$$
$$\mathtt{k} = 2,$$
$$\mathtt{b} = 1.$$

First, the `partition` function returns the damage information at the first generation:

$$\texttt{l1} = [k_{1,1}],$$
$$\texttt{e1} = [0.1],$$

the damage information with respect to the upper sub-network, $G_U(s)$:

$$\texttt{lU} = [b_{1,1}, k_{2,2}],$$
$$\texttt{eU} = [0.2, 0.3],$$

and the damage information with respect to the lower sub-network, $G_L(s)$:

$$\texttt{lL} = [b_{2,1}],$$
$$\texttt{eL} = [0.4].$$

Note that the subscripts in `lU` and `lL` are for their respective sub-networks, so they are different from those in `l`. For example, $k_{3,2}$ for the entire tree is equivalent to $k_{2,2}$ in the upper sub-tree, so $k_{3,2}$ in `l` is converted to $k_{2,2}$ in `lU` by the `partition` function.

Next, based on `l1` and `e1`, the `determineEKEB` function computes the damage amounts at the first generation, that is ε_k for $k_{1,1}$, and ε_b for $b_{1,1}$. In this example, the `determineEKEB` function returns

$$\texttt{eK} = 0.1,$$
$$\texttt{eB} = 1,$$

since $b_{1,1}$ is undamaged. After that, our modeling algorithm recursively calls itself twice to compute the coefficient vectors of two sub-trees, `cNU`, `cDU`, `cNL`, and `cDL`, given their respective damage information `lU`, `eU`, `lL`, and `eL`. Finally, our algorithm has everything it needs, which are then fed to the `merge` function in order to determine `cN` and `cD` based on Eqs. (8) and (9) for this specific damage case (`l`,`e`).

For the example (11), our algorithm returns

$$c_N = [1\ \ 10.9\ \ 68.9\ \ 273.8\ \ 738.5\ \ 1336.5\ \ 1507\ \ 1128.5\ \ 525.4\ \ 144.2\ \ 12]^\top,$$
$$c_D = [1\ \ 9.6\ \ 57\ \ 200.5\ \ 488.3\ \ 725.1\ \ 727.3\ \ 479.1\ \ 221.5\ \ 61.1\ \ 12]^\top.$$

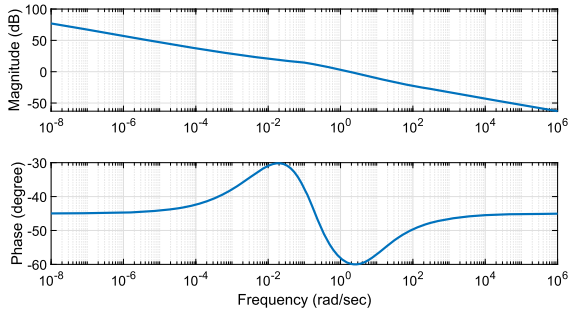

Fig. 3 Bode plot for the damaged tree's transfer function $G_{(l,\epsilon)}(s)$ whose damage case (l, ϵ) is described by Eq. (11)

Note that both c_N and c_D have length 11, which immediately tells us that $n = 5$ in this case. Consequently, we know that the transfer function for the damage case (11) is

$$G_{(\boldsymbol{l},\boldsymbol{\epsilon})} = G_\infty(s) \left(\frac{s^5 + 10.9s^{4.5} + \cdots + 144.2s^{0.5} + 12}{s^5 + 9.6s^{4.5} + \cdots + 61.1s^{0.5} + 12} \right), \quad (12)$$

whose Bode plot is shown in Fig. 3.

Similar to the undamaged tree, the correctness of the damaged transfer function Eq. (12) can also be shown by the convergence of the damaged tree's response as the number of generations increases to infinity. Figure 4 shows that convergence in the frequency domain, and Fig. 5 shows that in the time domain. The unit-step response of the finite trees in Fig. 5 are obtained by integrating the system of differential equations describing the motion using the `ode45()` function in `MATLAB`. However, the unit-step response of the infinite tree in Fig. 5 is computed by using the `step()` function in the FOTF `MATLAB` toolbox [9] given the analytical transfer function Eq. (12).

4 Identifying damages in the tree model

In this section, we propose a method to solve the inverse problem using the knowledge of the forward problem as discussed in Sect. 3. Specifically, we hope to identify the damaged components and quantify their damage amounts within the tree model given its (perhaps noisy) frequency response measurement. We test our identification algorithm on all damage cases where at most two components are damaged in the first three generations.

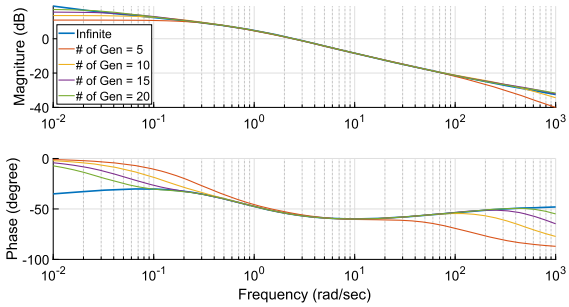

Fig. 4 As the number of generations increases, the frequency response of the damaged tree whose damage case is Eq. (11) converges to the infinite case whose analytical transfer function is Eq. (12)

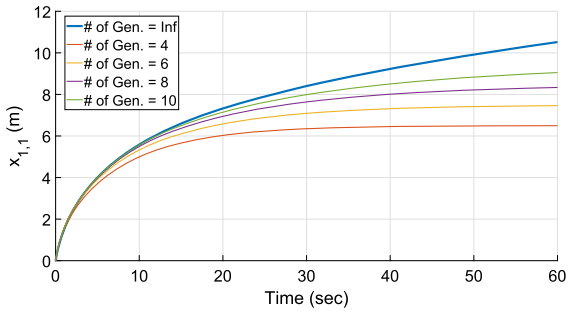

Fig. 5 As the number of generations increases, the unit-step response of the damaged tree whose damage case is Eq. (11) converges to the infinite case whose analytical transfer function is Eq. (12)

4.1 Damage identification algorithm for the tree model

From Eq. (6), we can observe that the transfer function for all trees contain $G_\infty(s)$. Therefore, for the identification, we only focus on the other part which varies for different damage cases. That is, we only focus on $\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ in the transfer function:

$$G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s) = G_\infty(s)\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s) = G_\infty(s)\frac{\boldsymbol{c_N}^\top \boldsymbol{\mu}(s;n)}{\boldsymbol{c_D}^\top \boldsymbol{\mu}(s;n)}.$$

A crucial advantage brought by solving the damage modeling problem is that it maps a damage case $(\boldsymbol{l},\boldsymbol{\epsilon})$ to its frequency response $G_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$. Without that knowledge, those coefficients, namely $\boldsymbol{c_N}$ and $\boldsymbol{c_D}$ in Eq. (6), surely can be found through, for example, the least squares regression. However, that does not provide any damage information which is our goal of identification. Moreover, without solving forward problems, the proposed identification method would not work well when there is noise in the frequency response measurements.

As a result of that mapping, we are able to formulate the damage identification problem as an optimization problem where the damage case $(\boldsymbol{l},\boldsymbol{\epsilon})$ is the decision variable directly. Inspired by [22], the optimization problem for damage identification can be formulated as

$$\min_{(\boldsymbol{l},\boldsymbol{\epsilon})} J(\boldsymbol{l},\boldsymbol{\epsilon}) = \sum_s \frac{\|\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s) - \overline{\Delta}(s)\|}{\|\overline{\Delta}(s)\|}, \tag{13}$$

such that

$$\begin{aligned}
&\boldsymbol{l} \in L,\\
&\text{length}(\boldsymbol{\epsilon}) = \text{length}(\boldsymbol{l}),\\
&0 < \varepsilon < 1 \text{ for all } \varepsilon \in \boldsymbol{\epsilon},\\
&\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})} = \frac{\boldsymbol{c_N}^\top \boldsymbol{\mu}(s;n)}{\boldsymbol{c_D}^\top \boldsymbol{\mu}(s;n)}.
\end{aligned}$$

The noisy frequency response measurement is $\overline{\Delta}(s)$, and $\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ is computed by our modeling algorithm listed in Table 1 providing a damage case $(\boldsymbol{l},\boldsymbol{\epsilon})$. The operator $\|\cdot\|$ means the 2-norm of a complex number. The relative difference is summed over all $s = i\omega$ where ω is the frequency at which the measured frequency response $\overline{\Delta}(s)$ is sampled. The set of all possible lists of damaged components, L, is defined in advance, so that the variable $\boldsymbol{l}$ is not left unconstrained.

However, because the list of damaged components $\boldsymbol{l}$ is a discrete variable, and the list of damage amounts $\boldsymbol{\epsilon}$ is a continuous variable, the optimization problem (13) is in fact a mixed-integer programming problem which is usually hard to solve [3]. Therefore, we separate $\boldsymbol{l}$ from $\boldsymbol{\epsilon}$, and thus convert a mixed-integer programming problem to many continuous nonlinear programming problems. Specifically, our identification algorithm iterates inside the set of all possible lists of damaged components L. For each possible list of damaged components, $\boldsymbol{l} \in L$, it solves a nonlinear programming problem for the best $\boldsymbol{\epsilon}$ at that $\boldsymbol{l}$ locally:

$$\min_{\boldsymbol{\epsilon}} J(\boldsymbol{l},\boldsymbol{\epsilon}) = \sum_s \frac{\|\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s) - \overline{\Delta}(s)\|}{\|\overline{\Delta}(s)\|}, \tag{14}$$

Table 2 Pseudocode of our algorithm to identify the damage within the tree model given its frequency response measurements

Identify the damage case $(\boldsymbol{l}^*,\boldsymbol{\epsilon}^*)$ given the frequency response measurement for a damaged tree.

```
JMin← +∞;
for l ∈ L do
 for ϵ0 ∈ E0 do
   Find ϵ and J where ϵ solves the nonlinear
   programming problem (14) at l, and J is the
   corresponding value for the objective function.
  if J < JMin
   JMin←J;
   l* ← l;
   ϵ* ← ϵ;
  end
 end
end
```

such that

$$\text{length}(\boldsymbol{\epsilon}) = \text{length}(\boldsymbol{l}),$$
$$0 < \varepsilon < 1 \text{ for all } \varepsilon \in \boldsymbol{\epsilon},$$
$$\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})} = \frac{\boldsymbol{c_N}^\top \boldsymbol{\mu}(s;n)}{\boldsymbol{c_D}^\top \boldsymbol{\mu}(s;n)}.$$

Finally, among all locally best $(\boldsymbol{l}, \boldsymbol{\epsilon})$, the globally best one $(\boldsymbol{l}^*, \boldsymbol{\epsilon}^*)$ is returned as the final identification result. Note that $\boldsymbol{\epsilon}$ is the only decision variable in the optimization problem (14), which renders it continuous.

The pseudocode of our damage identification algorithm is formally listed in Table 2. Note that at each possible list of damaged components $\boldsymbol{l}$, our algorithm starts at different initial guesses in the $\boldsymbol{\epsilon}$-space as indicated by the inner `for` loop. This is mainly because the objective function (14) is pretty "flat" in the $\boldsymbol{\epsilon}$-space away from the optimal point. Therefore, by starting at different initial guesses for the same $\boldsymbol{l}$, the possibility where the optimization solver stalls within that "flat" region is reduced. As an example, Fig. 6 plots the value of the objective function $J(\boldsymbol{l}, \boldsymbol{\epsilon})$ *versus* $\boldsymbol{\epsilon}$, where $\overline{\Delta}(s)$ is a perfect frequency response measurement for the damage case $([b_{1,1}, k_{2,1}], [0.05, 0.15])$, and $\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$ is computed when $\boldsymbol{l}$ is fixed at $[b_{1,1}, k_{2,1}]$ while both elements in its $\boldsymbol{\epsilon}$ vary from 0 to 1. From Fig. 6, we can observe that the region where both elements in $\boldsymbol{\epsilon}$ are close to 1 is quite "flat."

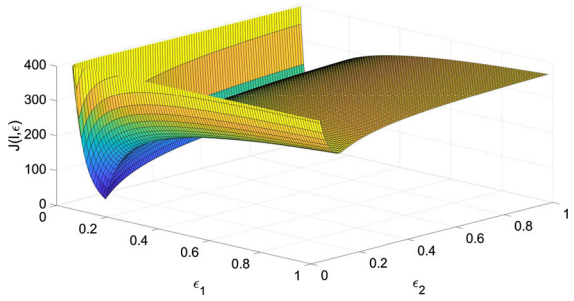

Fig. 6 The objective function $J(\boldsymbol{l}, \boldsymbol{\epsilon})$ may have a "flat" region in the space of the optimization variable $\boldsymbol{\epsilon}$

As a concrete example, Fig. 7 plots a noisy frequency response measurement $\overline{\Delta}(s)$ for the damage case

$$(\boldsymbol{l}, \boldsymbol{\epsilon}) = ([b_{1,1}, k_{3,1}], [0.45, 0.65]),$$

which needs identification. Suppose that we know there exist two damaged components within the first four generations in advance. Then, we can define the set of all possible lists of damaged components L as

$$L = \{[k_{1,1}, b_{1,1}], [k_{1,1}, k_{2,1}], \cdots, [k_{4,8}, b_{4,8}]\}.$$

As a result, the identification algorithm would iterate through all $\boldsymbol{l} \in L$. At the first iteration, $\boldsymbol{l} = [k_{1,1}, b_{1,1}]$, after solving the optimization problem (14), the identification algorithm finds that $\boldsymbol{\epsilon} = [0.814, 0.517]$ is the local best at that $\boldsymbol{l}$ with the corresponding objective function value equal to 26.2. Then, $\boldsymbol{l}$ goes to the next element in L, which is $[k_{1,1}, k_{2,1}]$ and the identification algorithm repeats the same procedure. At some point, $\boldsymbol{l} = [b_{1,1}, k_{3,1}] \in L$ at which solving optimization problem (14) returns $\boldsymbol{\epsilon} = [0.814, 0.517]$ with the corresponding objective function value equal to 23.8. Note that the identification process would not terminate until $\boldsymbol{l}$ reaches $[k_{4,8}, b_{4,8}]$, which is the last element in L. After that, the identification algorithm determines that 23.8 is the globally smallest objective function value. Therefore, it returns the corresponding $(\boldsymbol{l}^*, \boldsymbol{\epsilon}^*) = ([b_{1,1}, k_{3,1}], [0.437, 0.665])$ as the identification result. The identified frequency response $\Delta_{(\boldsymbol{l}^*,\boldsymbol{\epsilon}^*)}(s)$ is plotted in Fig. 8 along with the same noisy measurement from Fig. 7. Note that in this case, we say the algorithm correctly identifies the damage because the identified damaged components are correct, and the identified damage amounts are close to their actual values.

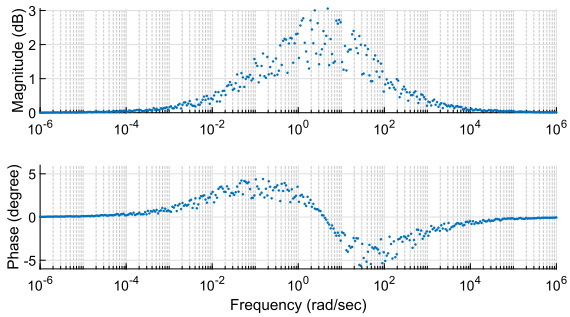

Fig. 7 A noisy frequency response measurement $\overline{\Delta}(s)$ for a damaged tree whose damage case $(\boldsymbol{l}, \boldsymbol{\epsilon}) = ([b_{1,1}, k_{3,1}], [0.45, 0.65])$. (50% noise added)

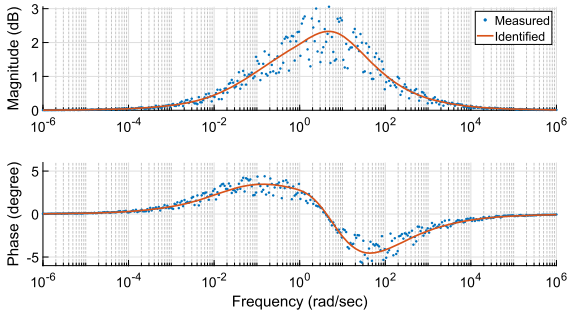

Fig. 8 The noisy measurement from Fig. 7 is plotted along with its identified $\Delta_{(l^*,\epsilon^*)}(s)$

4.2 Test scheme for the damage identification algorithm

The proposed damage identification algorithm is tested for all damage cases where at most two components are damaged within the first three generations. We use `fmincon()` from `MATLAB` to solve the optimization problem (14). For each damaged component, 10 different damage amounts starting from 0.05 to 0.95 with an increment of 0.1 are tested. From Fig. 1, we can conclude that there are $2 + 2^2 + 2^3 = 14$ components in the first three generations for the tree model. Therefore, the number of different damage cases is

$$\binom{14}{1} \times 10 + \binom{14}{2} \times 10^2 = 9,240.$$

To imitate real measurements, we also add 21 different levels of noise to the frequency response measurements $\overline{\Delta}(s)$. Therefore, our identification algorithm

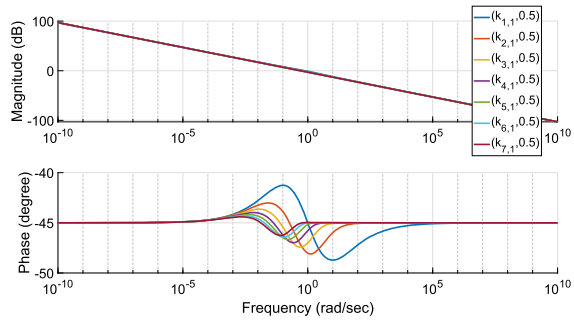

Fig. 9 Damaged transfer functions $G_{(l,\epsilon)}(s)$ for 7 different damage cases where the damaged component is $k_{1,1}$ through $k_{7,1}$, and the damage amount stays at 0.5

goes through $21 \times 9,240 = 194,040$ tests in total. Such a large number of different cases is the main reason why we limit ourselves to only two damaged components at most in the first three generations. For example, suppose we extend that limitation to three damaged components at most, then there are 7, 838, 040 damage cases in total, which is increased by a factor of 40.

The reason why we limit the test to only first three generations is because the frequency response for a damaged tree becomes less identifiable as the damage goes deeper. For example, the frequency responses $G_{(l,\epsilon)}(s)$ for 7 damage cases from $(k_{1,1}, 0.5)$ to $(k_{7,1}, 0.5)$ are shown in Fig. 9, from which we can observe that the discrepancy between two frequency responses becomes smaller as the damage goes deeper. This also reveals one intrinsic challenge of damage identification for a large network given its overall response which is less sensitive to deeper damages.

However, those limitations set for the identification test are only due to above two practical considerations. Theoretically, the proposed damage identification algorithm is able to handle any damage cases for the tree model.

Since the damages are limited to the first three generations, during the test, the outer `for` loop in Table 2 only iterates within the first four generations, so as to cover all neighboring generations surrounding possible damages while reduce the test's total running time. Suppose we also know there exist at most two damaged components. As a result,

$$L = \{k_{1,1}, b_{1,1}, \cdots, b_{4,8}, [k_{1,1}, b_{1,1}], \cdots, [k_{4,8}, b_{4,8}]\}.$$

In addition, for this test, the initial guesses are chosen to form a uniform grid from 0.1 to 0.9 with an increment of 0.2. That is,

$$E_0 = \{[0.1, 0.1], [0.1, 0.3], \cdots, [0.1, 0.9], \cdots, [0.9, 0.9]\}.$$

As said above, different levels of noise are added to the frequency response measurement $\overline{\Delta}(s)$. The meaning of adding $n_{\max}\%$ noise to $\overline{\Delta}(s)$ is that if the analytical value of $\overline{\Delta}(s) = A + iB$ at some angular frequency ω ($s = i\omega$), what the identification procedure can see is its corresponding noisy value of $\overline{\Delta}(s)$ where

$$\|\overline{\Delta}(s)\| = 10^{(1+n\%)\log_{10}(\sqrt{A^2+B^2})},$$
$$\angle\overline{\Delta}(s) = (1+n\%)\text{atan2}(B, A),$$

and n is a uniformly distributed random variable between $-n_{\max}$ and $n_{\max}$. For example, Fig. 7 shows the Bode plot when 50% measurement noise is added to $\overline{\Delta}(s)$ for the damage case $([b_{1,1}, k_{3,1}], [0.45, 0.65])$. During the test, 21 different levels of noise are added to the frequency response measurement $\overline{\Delta}(s)$ from 0 to 100% with a increment of 5%.

4.3 Test results for the damage identification algorithm

When no noise presenting in the frequency response measurement $\overline{\Delta}(s)$, our algorithm does very well, as expected. Out of 9240 damage cases, our algorithm only misidentifies the following two cases:

1. It misidentifies $([k_{3,2}, k_{3,3}], [0.95, 0.85])$ as $([b_{3,2}, k_{3,3}], [0.9622, 0.8395])$;
2. It misidentifies $([b_{3,2}, b_{3,3}], [0.85, 0.95])$ as $([b_{3,2}, k_{3,3}], [0.8395, 0.9622])$.

The above two misidentified cases reveal the nature of how difficult it is to identify damages within large networks. Ideally, when the measurements are perfect, our algorithm should identify the damage exactly, since it uses the knowledge from solving the modeling problem. However, that is not the case as indicated by the existence of the above two misidentified cases, even when we limit ourselves to just two damaged components within the only first three generations. The main reason is that the mapping from a damage case to its frequency response is not completely one-to-one. Therefore, not all inverse problems can be solved exactly. As an example, Fig. 10 shows the Bode plot for those two different damage cases in the first misidentified case above, from which we can confirm that their frequency responses $G_{(l,\epsilon)}(s)$ are the same.

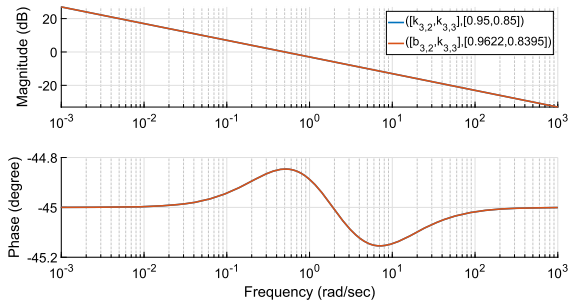

Fig. 10 The transfer functions $G_{(l,\epsilon)}(s)$ of two different damage cases overlap each other, which reveals the fact that the mapping from a damage case to its frequency response is not completely one-to-one

Except for the above two misidentified cases, all the other damage cases are correctly identified, with the maximum absolute error for the list of damage amounts ϵ being 1.2637×10^{-4}, which happens at the damage case $([b_{2,2}, k_{3,4}], [0.05, 0.45])$.

When noise presents in the frequency response measurements, the number of misidentified cases increases with the level of noise added to the frequency response measurements. However, the performance of our identification method is still very good. Figure 11 plots the percentage of misidentified cases out of total 9240 damage cases *versus* the level of noise added to the frequency response measurements. From Fig. 11, we can observe that, for example, even when 50% noise is added to the frequency response measurements, which should be as noisy as that shown in Fig. 7, only 35% of total 9240 damage cases are misidentified.

Based on the test results, we also have the following two observations when measurement noise presents in the test for our identification algorithm:

1. The possibility of misidentification increases as the damage goes deeper.
2. On the same generation, damages happening on the inner components are more inclined to be misidentified comparing to those on the outer components.

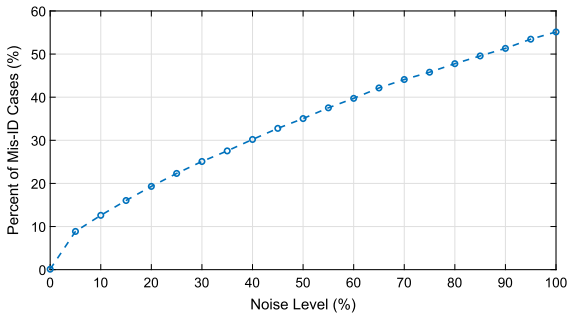

Fig. 11 Percentage of misidentified cases during the test *versus* the level of noise added to the measurement

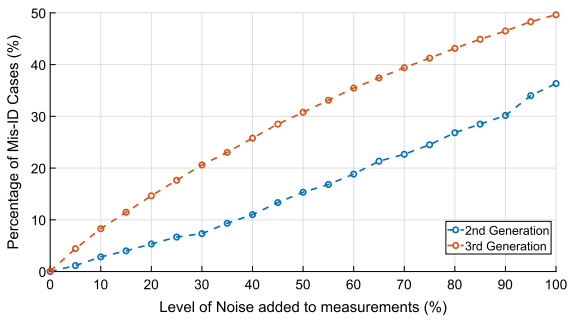

Fig. 12 Percentage of misidentified cases *versus* level of noise added to the measurements for those damage cases which are purely on the second and the third generation

Both of the above observations are intuitive. Observation 1 can be seen from Fig. 12, which compares the percentage of misidentified cases for those damage cases happening purely on the second and the third generation. From Fig. 12, we can observe that throughout all levels of added noise, the damages purely happening on the second generation always have less possibility of misidentification comparing to those on the third generation. The reason for such trend can be seen in Fig. 9 again, which shows that the discrepancy between two frequency responses becomes smaller when damages go deeper in the network.

For the second observation, we focus on the third generation, where we call $k_{3,1}$, $b_{3,1}$, $k_{3,4}$, $b_{3,4}$ outer components, and call $k_{3,2}$, $b_{3,2}$, $k_{3,3}$, $b_{3,3}$ inner components. During our test, among all damage cases within the above four outer components, 19% are misidentified. However, among all damage cases within the above four inner components, 42% are misidentified, which is more than twice larger than the previous one.

4.4 Computation time

We use `MATLAB R2019b` on a single CPU of Intel Core i7-4510U. On average, solving the optimization problem (14) for one initial guess, that is the part inside the double `for` loop of our damage identification algorithm listed in Table 2, takes 0.01 seconds. This time mainly depends on the nature of damage: the number of damaged components and the deepest generation where damages happen. The reason is that the nature of damage impacts the time required by the modeling algorithm to compute $\Delta_{(l,\epsilon)}(s)$ during each optimization iteration. Built upon that time, the total running time of our identification procedure is further affected by how many optimization iterations the solver takes to find the solution starting from one initial guess, and also by how that double `for` loop in the identification algorithm is implemented. For example, the total running time can be largely reduced by parallelizing the identification algorithm, and it can also be reduced by taking less initial guesses at the second `for` loop. A detailed analysis of running time can be found next in Sect. 5.

5 Effects brought by deeper damages

In this section, we discuss the effects on both damage modeling and damage identification brought by damaged components which locate deep inside the tree model.

5.1 Effects on damage modeling

There are two effects on damage modeling brought by deeper damages. A deeper damage case requires more time to compute its transfer function, and its coefficients are more inclined to numerical overflow.

To see how a deeper damage case affects the running time to compute its transfer function, we need to revisit the modeling algorithm listed in Table 1. The algorithm takes a divide-and-conquer approach, which divides the tree into two sub-trees, recursively computes the transfer functions for those two parts, and finally merges those two transfer functions into one describing the entire tree's dynamics. For the damage case where both components at the first generation are damaged, those two sub-trees are undamaged. As

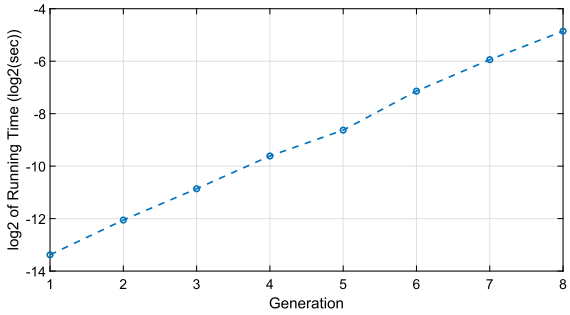

Fig. 13 $log_2()$ of the total running time for the modeling algorithm *versus* the generation g where all components in the first g generations are damaged with damage amount at 0.5. The undamaged constants' values are $k = 2N/m$ and $b = 1Ns/m$

a result, those two recursive calls would immediately return since their `if` condition are both satisfied. After that, the `merge()` function is called once to compute the final result. For the damage case where all components in the first two generations are damaged, those two sub-trees have damages only at their first generations. As a result, each of them calls the `merge()` function once. After that, there is an additional call of the `merge()` function for the entire tree. Therefore, the `merge()` function is called three times in total for this case. Continuing the similar reasoning, we can prove that in the worst case, where all components are damaged in the first g generations, the `merge()` function would be called $2^g - 1$ times. Because the `merge()` function is the only time-consuming part of the modeling algorithm, the total running time for the modeling algorithm is expected to double when a damaged component goes one generation deeper. Such trend is confirmed by Fig. 13 where the slope of $log_2()$ of the total running time is about one per generation. The computation environment is same as that in Sect. 4.3.

Another effect brought by a deeper damage case is that the coefficients in its transfer function are more inclined to numerical overflow. Although, ideally, our modeling algorithm can compute transfer functions for any damage cases, its practical ability is limited by the computational machine in use. The reason for numerical overflow is that there exist vector convolutions in Eqs. (8) and (9). As damages go deeper, more convolutions are used, which causes the middle elements of those coefficient vectors c_N and c_D to grow larger compared to the boundary elements of those two coefficient vectors. As damages continue going deeper, at some point, those middle elements would break the numerical limit of the machine in use. For example, in 64-bit `MatlabR2019b`, the coefficients start overflow when all components in the first nine generations are damaged with damage amount 0.5, and the undamaged constants' values are $k = 2N/m$ and $b = 1Ns/m$. Once the numerical limit of the computational machine in use is violated, the result returned by our modeling algorithm is no longer useful. As a result, it is very important to monitor the numerical overflow status when our modeling algorithm is implemented on some machine which does not do that by itself.

5.2 Effects on damage identification

There are two effects on damage identification brought by deeper damages. First, a deeper damage case is more difficult to be identified through the overall frequency response of a network. Second, to identify a deeper damage case requires more time using our damage identification algorithm.

The first effect means that a deeper damage case is more likely to be misidentified, which has already been discussed in Sect. 4.2 and is shown in Fig. 9. The reason is that a deeper damage has less effect on a network's overall behavior which follows the intuition. This is an intrinsic challenge of identifying deep damages inside a large network given its overall behavior only.

There are two factors which increase the running time of our identification algorithm for a deep damage case. First, note that our identification algorithm computes the damaged transfer functions online. That is, at each optimization iteration, the optimization solver proposes a candidate list of damage amounts, $\boldsymbol{\epsilon}$, which is combined with the current list of damaged components, $\boldsymbol{l}$, to create a candidate damage case, $(\boldsymbol{l},\boldsymbol{\epsilon})$. Then, that $(\boldsymbol{l},\boldsymbol{\epsilon})$ is passed to our damage modeling algorithm to compute $\Delta_{(\boldsymbol{l},\boldsymbol{\epsilon})}(s)$. Therefore, due to the reason we discuss in Sect. 5.1, the time required at each optimization iteration would double when a damage case goes one generation deeper. The second factor is that a deep damage case increases the size of the set L which includes all possible lists of damaged components. As a result, that increment in size leads to more iterations taken

by the outer `for` loop in our identification procedure. For example, as we see in Sect. 4.2, if the set L is constructed by the premise that at most two damaged components exist in the first three generations, the size of L is

$$\binom{14}{1} + \binom{14}{2} = 105.$$

Now, if that assumption becomes at most two damaged components exist in the first four generations, the size of L would increase to

$$\binom{30}{1} + \binom{30}{2} = 465.$$

6 Concluding remarks

In this paper, we show that the structure of the transfer function for a damaged tree always follows Eq. (6). In addition, we propose a recursive algorithm to compute the expression of that transfer function for a damage case. Then, using the knowledge from damage modeling, we also propose a damage identification algorithm trying to identify the damaged components and quantify their damage amounts inside the tree model through its overall frequency response.

The main assumption for the proposed damage identification method to work is that we know the mapping from a damaged network to its overall frequency response. However, one limitation of this work at its current state is that we only know that mapping for a very specific network, the tree model. Therefore, the urgent future work is to generalize this idea to other networks. Our initial analysis shows that promising. Recently, we believe that for a self-similar infinite network, if its recurrence formula is similar to Eq. (1), and that recurrence formula also leads to an analytical solvable undamaged transfer function, similar to Eq. (4), then its damaged transfer function always has the multiplicative structure which is similar to Eq. (5). Thus, we are working to extend these results to any infinitely large self-similar network, not only the specific tree network. The problem left is then how to determine the coefficient vectors for that damaged network as we did for the tree model in Sect. 3. Note that the transfer function for a damaged tree is commensurable as shown in Eq. (6), which means that all orders of s have a common factor $1/2$. Our initial analysis shows that even for those networks whose transfer function has irrational parts, for example $\sqrt{s+1}$, their damaged transfer function can still be modeled by Eq. (5). An additional advantage brought by Eq. (5) is that it is a classical model in the area of robust control. Therefore, that multiplicative nature may help us to control those damaged networks, which is another goal of our future work. On top of the above-mentioned two, we also plan to characterize the difference between the response of a finite network with respect to its infinite version. By doing that, we hope to make the damage detection method proposed in paper more applicable to the real networks' health monitoring setting.

Compliance with ethical standards

Conflicts of interest Xiangyu Ni has worked in MathWorks for 3 months. The authors have no other conflicts of interest in addition to that.

References

1. Ahmed, E., Elgazzar, A.: On fractional order differential equations model for nonlocal epidemics. Physica A **379**(2), 607–614 (2007)
2. Balch, T., Arkin, R.C.: Behavior-based formation control for multirobot teams. IEEE Trans. Robot. Autom. **14**(6), 926–939 (1998)
3. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: Mixed-integer programming: a progress report. In: The Sharpest Cut: The Impact of Manfred Padberg and His Work, pp. 309–325. SIAM (2004)
4. Borino, G., Di Paola, M., Zingales, M.: A non-local model of fractional heat conduction in rigid bodies. Eur. Phys. J. Spec. Top. **193**(1), 173–184 (2011)
5. Bouc, R.: A mathematical model for hysteresis. Acta Acust. United Acust. **24**(1), 16–25 (1971)
6. Brincker, R., Zhang, L., Andersen, P.: Modal identification of output-only systems using frequency domain decomposition. Smart Mater. Struct. **10**(3), 441 (2001)
7. Cao, Y., Ren, W.: Distributed formation control for fractional-order systems: dynamic interaction and absolute/relative damping. Syst. Control Lett. **59**(3–4), 233–240 (2010)
8. Chatzi, E.N., Smyth, A.W.: The unscented Kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. Struct. Control Health Monit.: Off. J. Int. Assoc. Struct. Control Monit. Eur. Assoc. Control Struct. **16**(1), 99–123 (2009)
9. Chen, Y., Petras, I., Xue, D.: Fractional order control—a tutorial. In: 2009 American Control Conference, pp. 1397–1411 (2009). https://doi.org/10.1109/ACC.2009.5160719
10. Cottone, G., Di Paola, M., Zingales, M.: Fractional mechanical model for the dynamics of non-local continuum. In: Advances in Numerical Methods, pp. 389–423. Springer, Berlin (2009)

11. Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J., Taylor, C.J.: A vision-based formation control framework. IEEE Trans. Robot. Autom. **18**(5), 813–825 (2002)
12. Di Paola, M., Failla, G., Zingales, M.: Physically-based approach to the mechanics of strong non-local linear elasticity theory. J. Elast. **97**(2), 103–130 (2009)
13. Doehring, T.C., Freed, A.D., Carew, E.O., Vesely, I.: Fractional order viscoelasticity of the aortic valve cusp: an alternative to quasilinear viscoelasticity. J. Biomech. Eng. (2005)
14. Gabryś, E., Rybaczuk, M., Kędzia, A.: Fractal models of circulatory system: symmetrical and asymmetrical approach comparison. Chaos Solitons Fract. **24**(3), 707–715 (2005)
15. Goodwine, B.: Modeling a multi-robot system with fractional-order differential equations. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1763–1768. IEEE (2014)
16. Heymans, N., Bauwens, J.C.: Fractal rheological models and fractional differential equations for viscoelastic behavior. Rheol. Acta **33**(3), 210–219 (1994)
17. Ionescu, C.M., Machado, J.T., De Keyser, R.: Modeling of the lung impedance using a fractional-order ladder network with constant phase elements. IEEE Trans. Biomed. Circuits Syst. **5**(1), 83–89 (2010)
18. Juang, J.N., Pappa, R.S.: An eigensystem realization algorithm for modal parameter identification and model reduction. J. Guid. Control Dyn. **8**(5), 620–627 (1985)
19. Kelly, J.F., McGough, R.J.: Fractal ladder models and power law wave equations. J. Acoust. Soc. Am. **126**(4), 2072–2081 (2009)
20. Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., Turon, M.: Health monitoring of civil infrastructures using wireless sensor networks. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, pp. 254–263 (2007)
21. Leyden, K., Goodwine, B.: Using fractional-order differential equations for health monitoring of a system of cooperating robots. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 366–371. IEEE (2016)
22. Leyden, K., Goodwine, B.: Fractional-order system identification for health monitoring. Nonlinear Dyn. **92**(3), 1317–1334 (2018)
23. Magin, R.L.: Fractional Calculus in Bioengineering. Begell House Redding (2006)
24. Mandelbrot, B.B.: The Fractal Geometry of Nature, vol. 173. W.H. Freeman, New York (1983)
25. Masters, B.R.: Fractal analysis of the vascular tree in the human retina. Annu. Rev. Biomed. Eng. **6**, 427–452 (2004)
26. Mayes, J., Sen, M.: Approximation of potential-driven flow dynamics in large-scale self-similar tree networks. Proc. R. Soc. A: Math. Phys. Eng. Sci. **467**(2134), 2810–2824 (2011)
27. Murray, R.M.: Recent research in cooperative control of multivehicle systems. J. Dyn. Syst. Meas. Control **129**(5), 571–583 (2007)
28. Peeters, B., De Roeck, G.: Stochastic system identification for operational modal analysis: a review. J. Dyn. Syst. Meas. Control **123**(4), 659–667 (2001)
29. Ren, W., Beard, R.W.: Distributed Consensus in Multivehicle Cooperative Control. Springer, Berlin (2008)
30. Ren, W., Beard, R.W., Atkins, E.M.: Information consensus in multivehicle cooperative control. IEEE Control Syst. Mag. **27**(2), 71–82 (2007)
31. Ren, W., Moore, K.L., Chen, Y.: High-order and model reference consensus algorithms in cooperative control of multivehicle systems. J. Dyn. Syst. Meas. Control (2007)
32. Roemer, M.J., Kacprzynski, G.J.: Advanced diagnostics and prognostics for gas turbine engine risk assessment. In: 2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484), vol. 6, pp. 345–353. IEEE (2000)
33. Rytter, A.: Vibrational based inspection of civil engineering structures. Ph.D. thesis, Aalborg University (1993)
34. Sikorska, J., Hodkiewicz, M., Ma, L.: Prognostic modelling options for remaining useful life estimation by industry. Mech. Syst. Signal Process. **25**(5), 1803–1836 (2011)
35. Sun, W., Li, Y., Li, C., Chen, Y.: Convergence speed of a fractional order consensus algorithm over undirected scale-free networks. Asian J. Control **13**(6), 936–946 (2011)
36. Vaiana, N., Sessa, S., Marmo, F., Rosati, L.: A class of uniaxial phenomenological models for simulating hysteretic phenomena in rate-independent mechanical systems and materials. Nonlinear Dyn. **93**(3), 1647–1669 (2018). https://doi.org/10.1007/s11071-018-4282-2
37. Vaiana, N., Sessa, S., Marmo, F., Rosati, L.: Nonlinear dynamic analysis of hysteretic mechanical systems by combining a novel rate-independent model and an explicit time integration method. Nonlinear Dyn. **98**(4), 2879–2901 (2019). https://doi.org/10.1007/s11071-019-05022-5
38. Wen, Y.K.: Method for random vibration of hysteretic systems. J. Eng. Mech. Div. **102**(2), 249–263 (1976)
39. Worden, K., Farrar, C.R., Haywood, J., Todd, M.: A review of nonlinear dynamics applications to structural health monitoring. Struct. Control Health Monit.: Off. J. Int. Assoc. Struct. Control Monit. Eur. Assoc. Control Struct. **15**(4), 540–567 (2008)