

# Damage Identification for The Tree-like Network through Frequency-domain Modeling<sup>★</sup>

Xiangyu Ni<sup>\*</sup> Bill Goodwine<sup>\*\*</sup>

<sup>\*</sup> *Department of Aerospace and Mechanical Engineering, University of Notre Dame, IN 46556 USA (e-mail: xni@nd.edu).*

<sup>\*\*</sup> *Department of Aerospace and Mechanical Engineering, University of Notre Dame, IN 46556 USA (e-mail: bill@controls.ame.nd.edu)*

**Abstract:** In this paper, we propose a method to identify the damaged component and quantify its damage amount in a large network given its overall frequency response. The identification procedure takes advantage of our previous work which exactly models the frequency response of that large network when it is damaged. As a result, the test shows that our method works well when some noise present in the frequency response measurement. In addition, the effects brought by a damaged component which is located deep inside that large network are also discussed.

Copyright © 2020 The Authors. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0>)

Keywords: Fractional-order dynamics, Multi-agent systems, Frequency-domain identification, Fault detection and diagnosis.

## 1. INTRODUCTION

This paper proposes a damage identification method for a tree model given a noisy measurement of its overall frequency response when there exists one damaged component. The tree model, which is shown in Fig. 1, has many applications. For example, see its application to viscoelastic behavior in Heymans and Bauwens (1994), blood vessel in Gabryś et al. (2005), the vascular tree in human retina in Masters (2004), and 1D relaxation of the aortic valve in Doehring et al. (2005). The main reason for us to choose that model as the starting point of our work is because its transfer function is rational with real orders and commensurable (See Valério and Da Costa (2013) for the definition of commensurable transfer function), which is the simplest case among all non-integer-order systems. Note that although the tree model consists of linear springs and dampers, it can also be converted to an electrical, a fluid or a thermal system since springs and dampers have their corresponding equivalent components for those systems.

Fault detection is indispensable to modern industry in real life. Consequently, different types of fault detection methods have been proposed, for example Roemer and Kacprzyński (2000) and Sikorska et al. (2011). One type of methods uses system identification to monitor a system's health, for example Juang and Pappa (1985), Brincker et al. (2001) and Peeters and De Roeck (2001). The damage identification method proposed in this paper belongs to that type. In addition to using system identification, the method proposed in this paper also leverages the result from our previous work, which exactly models the frequency response of a tree with one damaged component. That knowledge from modeling brings two advantages.

First, it makes our identification procedure "know" the answer when the measurement of a damaged tree's frequency response is perfect. As a result, that modeling knowledge helps our identification method work well with a very noisy measurement. Second, that modeling knowledge also helps us to cast a damage identification problem as an optimization problem where the damage case is the decision variable directly. Therefore, we can not only identify the transfer function, but, importantly, also identify the damage inside that tree directly. As we shall see later, the overall frequency responses for large networks are non-integer order naturally. Hence, our proposed method is derived from a fractional-order system identification method proposed by Oustaloup (1995). Other fractional-order system identification methods can be seen in Hartley and Lorenzo (2003), Liu et al. (2013), and Zhou et al. (2013).

There are at least two limitations for this work at its current state. First, the number of damaged components is limited to one in this paper. Second, it only applies to one specific network, the tree model. Therefore, we are working on generalizing this idea to multiple damaged components in a class of networks similar to the tree. The result of our initial analysis is promising.

A closely related literature, Leyden and Goodwine (2018), from our group sets a goal similar to this paper. However, they are not exactly the same. Leyden and Goodwine (2018) uses the order variation in a large network's transfer function to monitor its health which is different from the method proposed in this paper. Furthermore, that work cannot identify the exact damage case inside a network. In contrast, this paper aims at locating the damaged component and quantifying its damage amount.

The rest of this paper is organized as follows. Section 2 formally defines the tree model and computes its undam-

<sup>★</sup> The partial support of NSF Award 1826079 is gratefully acknowledged.

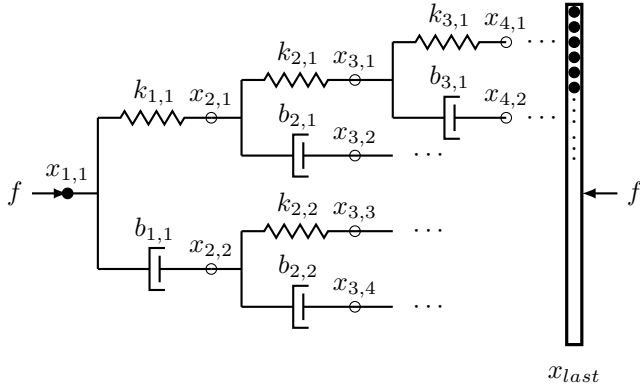


Fig. 1. The tree model.

aged transfer function. Most importantly, it also recaps our previous work showing how to model its damaged transfer function. Then, Section 3 uses the knowledge from that damage modeling to propose a damage identification algorithm. Section 4 shows the test results for that algorithm and lists observations about the misidentified cases. Section 5 talks about some rationales behind that identification procedure and some effects brought by a damaged component which is located at a deep generation inside the tree. Finally, Section 6 concludes this paper.

## 2. THE TREE MODEL

The tree model, as shown in Fig. 1, has an infinite number of generations. 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 and the lower one is connected through a linear damper. At the last generation, all nodes are locked together. The input of interest here is the force,  $f$ , applied to both ends of the tree, and the output is the relative displacement,  $x_{1,1} - x_{last}$ , between both ends. Therefore, throughout this paper, without explicit exception, transfer functions always represent  $(X_{1,1}(s) - X_{last}(s))/F(s)$ .

Due to self-similarity, it is easy to see those two sub-networks, from  $x_{2,1}$  and  $x_{2,2}$  to  $x_{last}$ , are also trees. Let us use  $G_U(s)$  and  $G_L(s)$  to represent their transfer functions, that is

$$\begin{aligned} G_U(s) &= \frac{X_{2,1}(s) - X_{last}(s)}{F_1(s)}, \\ G_L(s) &= \frac{X_{2,2}(s) - X_{last}(s)}{F_2(s)}, \\ F(s) &= F_1(s) + F_2(s). \end{aligned}$$

Then, the tree model can be illustratively drawn as Fig. 2, from which, using series and parallel connection rules for mechanical components, we can obtain that the transfer function for the entire tree can be computed by the following formula given the expressions of  $G_U(s)$  and  $G_L(s)$ ,

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

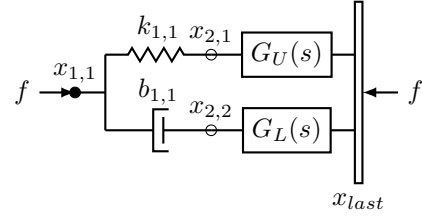


Fig. 2. An illustration about the tree model taking advantage of its self-similarity, which is equivalent to Fig. 1.

which we call the transformation formula for the tree model.

For the undamaged tree, all of its springs have the same constant  $k$  and all of its dampers have the same constant  $b$ . In addition, we use  $G_\infty(s)$  to specifically denote the transfer function for the undamaged tree model. Chapter 3 in Mayes (2012) shows that by converting the transformation formula (1) to its undamaged version, we can obtain that the undamaged transfer function for the tree model is exactly half-order:

$$G_\infty(s) = \frac{1}{\sqrt{kbs}}. \quad (2)$$

When the tree model is damaged, some of its springs' (dampers') constants are different from their undamaged value  $k$  ( $b$ ). Throughout this paper, we assume that only one component, either one spring or one damper, is damaged, which is denoted by  $l$ . In addition, we also assume the constant of that damaged component  $l$  changes from the undamaged value  $k$  ( $b$ ) to  $k \cdot \epsilon$  ( $b \cdot \epsilon$ ), where  $\epsilon$  is called its damage amount and  $0 < \epsilon < 1$ . Note that, in the rest of this paper, we use a pair  $(l, \epsilon)$  to refer to a certain damage case.

Our previous work shows that the transfer function of a damage case  $(l, \epsilon)$  for the tree model with one damaged component can be expressed as

$$G_{(l, \epsilon)}(s) = G_\infty(s) \cdot \Delta_{(l, \epsilon)}(s),$$

where

$$\Delta_{(l, \epsilon)}(s) = \frac{N_{(l, \epsilon)}(s)}{D_{(l, \epsilon)}(s)} = \frac{\prod_{j=1}^{2g} (s^{\frac{1}{2}} + z_j(l, \epsilon))}{\prod_{j=1}^{2g} (s^{\frac{1}{2}} + p_j(l, \epsilon))}. \quad (3)$$

Here,  $-z_j$  and  $-p_j$  are called half-order zeros and poles. Moreover,  $g$  denotes the generation where the damaged component  $l$  is located. For example, when the damper  $b_{3,1}$  is damaged,  $l = b_{3,1}$  and thus  $g = 3$ . Therefore, all damage cases  $(b_{3,1}, \epsilon)$  have 6 pairs of half-order zeros and poles. Note that the damaged transfer function (3) is rational and commensurable.

To compute  $z_j$  and  $p_j$  for a specific damage case  $(l, \epsilon)$  numerically, we have to start at the first generation and go through each generation until the damaged component  $l$  is reached. Such computation takes advantage of the fact that the tree model is self-similar.

As a concrete example, let us show how to use the transformation formula (1) repeatedly to obtain  $z_j$  and  $p_j$  for the damage case  $(k_{3,1}, \epsilon)$ . Due to self-similarity, from Fig. 1, we observe that the  $k_{3,1}$  component of the entire tree model is equivalent to the  $k_{2,1}$  component within the sub-network from  $x_{2,1}$  to  $x_{last}$ , and the  $k_{2,1}$

component of the entire tree model is equivalent to the  $k_{1,1}$  component within the sub-network from  $x_{2,1}$  to  $x_{\text{last}}$ . Therefore, using such self-similarity, we see that we need to first compute the transfer function for the damage case  $(k_{1,1}, \epsilon)$ , namely  $G_{(k_{1,1}, \epsilon)}(s)$ . Then, using that result, we can compute  $G_{(k_{2,1}, \epsilon)}(s)$ , which is then used to finally compute  $G_{(k_{3,1}, \epsilon)}(s)$ .

For the damage case  $(k_{1,1}, \epsilon)$ , that is when  $k_{1,1}$ 's spring constant becomes  $k \cdot \epsilon$  and all the other springs' (dampers') constants stay at  $k$  ( $b$ ), the transfer function for the sub-network from  $x_{2,1}$  to  $x_{\text{last}}$  in this damage case is same as the undamaged transfer function, *i.e.*  $G_U(s) = G_\infty(s)$ . For the same reason,  $G_L(s) = G_\infty(s)$ , too. Hence, for this damage case, we can replace  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  with  $(G_{(k_{1,1}, \epsilon)}(s), G_\infty(s), G_\infty(s), k \cdot \epsilon, b)$  in Eq. (1) which leads to

$$G_{(k_{1,1}, \epsilon)}(s) = \frac{1}{\frac{1}{\frac{1}{k \cdot \epsilon} + G_\infty(s)} + \frac{1}{\frac{1}{bs} + G_\infty(s)}}.$$

After simplification, this gives

$$\begin{aligned} G_{(k_{1,1}, \epsilon)}(s) &= G_\infty(s) \cdot \Delta_{(k_{1,1}, \epsilon)}(s) \\ &= G_\infty(s) \cdot \frac{N_{(k_{1,1}, \epsilon)}(s)}{D_{(k_{1,1}, \epsilon)}(s)} \\ &= G_\infty(s) \cdot \frac{\left(s^{\frac{1}{2}} + \sqrt{\frac{k}{b}}\right) \left(s^{\frac{1}{2}} + \epsilon \sqrt{\frac{k}{b}}\right)}{\left(s^{\frac{1}{2}} + \epsilon \sqrt{\frac{k}{b}} + \sqrt{\frac{\epsilon(\epsilon-1)k}{b}}\right) \left(s^{\frac{1}{2}} + \epsilon \sqrt{\frac{k}{b}} - \sqrt{\frac{\epsilon(\epsilon-1)k}{b}}\right)}. \end{aligned} \quad (4)$$

Using the result in Eq. (4), we can now compute the transfer function for the damage case  $(k_{2,1}, \epsilon)$ , that is  $G_{(k_{2,1}, \epsilon)}(s)$ . For the similar reason explained above,  $G_{(k_{2,1}, \epsilon)}(s)$  can be obtained by replacing  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  with  $(G_{(k_{2,1}, \epsilon)}(s), G_{(k_{1,1}, \epsilon)}(s), G_\infty(s), k, b)$  in Eq. (1), which gives

$$G_{(k_{2,1}, \epsilon)}(s) = \frac{1}{\frac{1}{\frac{1}{k} + G_{(k_{1,1}, \epsilon)}(s)} + \frac{1}{\frac{1}{bs} + G_\infty(s)}}.$$

After simplification, this gives

$$G_{(k_{2,1}, \epsilon)}(s) = G_\infty(s) \cdot \frac{sD + s^{\frac{1}{2}} \sqrt{\frac{k}{b}}(N + D) + \frac{k}{b}N}{sD + s^{\frac{1}{2}} \sqrt{\frac{k}{b}}(N + D) + \frac{k}{b}D}. \quad (5)$$

Note that here  $D = D_{(k_{1,1}, \epsilon)}(s)$  and  $N = N_{(k_{1,1}, \epsilon)}(s)$  are known from Eq. (4). Hence, by using a numerical equation solver, we can find the values for  $s^{\frac{1}{2}}$  which make either the numerator or the denominator in Eq. (5) equal to 0, and those values are the half-order zeros and poles for damage case  $(k_{2,1}, \epsilon)$ . Then, from the numerical value of those half-order zeros and poles, we can construct that

$$G_{(k_{2,1}, \epsilon)}(s) = G_\infty(s) \cdot \frac{\prod_{j=1}^4 \left(s^{\frac{1}{2}} + z_j(k_{2,1}, \epsilon)\right)}{\prod_{j=1}^4 \left(s^{\frac{1}{2}} + p_j(k_{2,1}, \epsilon)\right)}. \quad (6)$$

Table 1. Correspondence among all springs up to the 4<sup>th</sup> generation.

1 <sup>st</sup> Gen.	2 <sup>nd</sup> Gen.	3 <sup>rd</sup> Gen.	4 <sup>th</sup> Gen.
$k_{1,1}$	$k_{2,1}$	$k_{3,1}$	$k_{4,1}$
			$k_{4,5}$
		$k_{3,3}$	$k_{4,3}$
			$k_{4,7}$
	$k_{2,2}$	$k_{3,2}$	$k_{4,2}$
			$k_{4,6}$
		$k_{3,4}$	$k_{4,4}$
			$k_{4,8}$

Table 2. Elements substituted into  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  in Eq. (1) enable us to move between corresponding components at two consecutive generations.

Elements substituted into Eq. (1)	Corresponding components
$(G_\infty(s), G_\infty(s), G_\infty(s), k, b)$	Undamaged
$(G_{(k_{1,1}, \epsilon)}(s), G_\infty(s), G_\infty(s), k \cdot \epsilon, b)$	Undamaged $\rightarrow k_{1,1}$
$(G_{(b_{1,1}, \epsilon)}(s), G_\infty(s), G_\infty(s), k, b \cdot \epsilon)$	Undamaged $\rightarrow b_{1,1}$
$(G_{(k_{2,1}, \epsilon)}(s), G_{(k_{1,1}, \epsilon)}(s), G_\infty(s), k, b)$	$k_{1,1} \rightarrow k_{2,1}$
$(G_{(k_{2,2}, \epsilon)}(s), G_\infty(s), G_{(k_{1,1}, \epsilon)}(s), k, b)$	$k_{1,1} \rightarrow k_{2,2}$
$(G_{(k_{3,1}, \epsilon)}(s), G_{(k_{2,1}, \epsilon)}(s), G_\infty(s), k, b)$	$k_{2,1} \rightarrow k_{3,1}$
$(G_{(k_{3,2}, \epsilon)}(s), G_{(k_{2,2}, \epsilon)}(s), G_\infty(s), k, b)$	$k_{2,2} \rightarrow k_{3,2}$
$(G_{(k_{3,3}, \epsilon)}(s), G_\infty(s), G_{(k_{2,1}, \epsilon)}(s), k, b)$	$k_{2,1} \rightarrow k_{3,3}$
$(G_{(k_{3,4}, \epsilon)}(s), G_\infty(s), G_{(k_{2,2}, \epsilon)}(s), k, b)$	$k_{2,2} \rightarrow k_{3,4}$

Finally, we can compute half-order zeros and poles for the damage case  $(k_{3,1}, \epsilon)$  by using that  $G_{(k_{2,1}, \epsilon)}(s)$  in Eq. (6). For the similar reason explained above, we can obtain  $G_{(k_{3,1}, \epsilon)}(s)$  by replacing  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  with  $(G_{(k_{3,1}, \epsilon)}(s), G_{(k_{2,1}, \epsilon)}(s), G_\infty(s), k, b)$  in Eq. (1), which gives an expression similar to Eq. (5). Then, again, using a numerical equation solver, we are able to find those half-order zeros and poles for  $G_{(k_{3,1}, \epsilon)}(s)$ .

In summary, to compute  $z_j$  and  $p_j$  for a damage case  $(l, \epsilon)$ , we need to compute its corresponding damage case at each generation from the first one to the one where the damaged component  $l$  is located. Table 1 shows such correspondence among all springs up to the fourth generation. For example, if we want to compute  $z_j$  and  $p_j$  for the damage case  $(k_{4,7}, \epsilon)$ , using Table 1, we know that we first need to compute  $G_{(k_{1,1}, \epsilon)}(s)$ . Next, we need to use that result to compute  $G_{(k_{2,1}, \epsilon)}(s)$ , which enables us to further obtain  $G_{(k_{3,3}, \epsilon)}(s)$ . Then, we finally reach at  $G_{(k_{4,7}, \epsilon)}(s)$ . Note that the correspondence among all dampers works exactly the same.

Additionally, as described in the example above, moving between corresponding components at two consecutive generations requires to substitute correct elements into  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  in Eq. (1). Such substitution is listed in Table 2. For example, if we want to compute  $G_{(k_{3,3}, \epsilon)}(s)$  based on  $G_{(k_{2,1}, \epsilon)}(s)$ , from Table 2, we know that we can replace  $(G(s), G_U(s), G_L(s), k_{1,1}, b_{1,1})$  with  $(G_{(k_{3,3}, \epsilon)}(s), G_\infty(s), G_{(k_{2,1}, \epsilon)}(s), k, b)$  in Eq. (1). Again, note that the substitution for all dampers are exactly the same except for  $b_{1,1}$  which has also been listed in Table 2.

### 3. DAMAGE IDENTIFICATION ALGORITHM

In this section, we formally describe our damage identification procedure. The goal is that, for a tree with one damaged component, given a noisy measurement of its frequency response, we want to identify the damaged component and quantify its damage amount. The knowledge brought by the exact modeling of damaged trees allows us to formulate a damage identification problem as an optimization problem where the damage case  $(l, \epsilon)$  is the decision variable directly. That is, we are able to come up with a metric which directly maps a damage case  $(l, \epsilon)$  to a quantification of the difference between its computed frequency response and the measured one. Inspired by Chapter 4 in Leyden (2018), we use the following metric to quantify that identification error

$$J(l, \epsilon) = \sum_s \frac{\|\Delta_{(l, \epsilon)}(s) - \bar{\Delta}(s)\|}{\|\bar{\Delta}(s)\|}, \quad (7)$$

where  $\Delta_{(l, \epsilon)}(s)$  is the computed frequency response for a damage case  $(l, \epsilon)$  and  $\bar{\Delta}(s)$  is a noisy measurement waiting for identification. In addition, the summation is over  $s = i \cdot \omega$  where  $\omega$  are the angular frequencies at which  $\bar{\Delta}(s)$  is sampled.

Here is the main idea of our identification procedure. Before identification, we need a prior knowledge about where that damage is possibly located. For example, we assume that the damaged component is within the first four generations. Then, we can define a finite set for possible damaged components, that is

$$L = [k_{1,1}, b_{1,1}, k_{2,1}, b_{2,1}, k_{2,2}, b_{2,2}, \dots, k_{4,8}, b_{4,8}]. \quad (8)$$

For each component  $l \in L$ , we find the locally best  $\epsilon$  which minimizes the identification error  $J(l, \epsilon)$ . Then, among all those locally best pairs of  $(l, \epsilon)$ , we pick the one which gives the globally smallest identification error to be the final identification result  $(l^*, \epsilon^*)$ .

At every  $l \in L$ , we solve the nonlinear programming problem

$$\min_{\epsilon} J(l, \epsilon) = \sum_s \frac{\|\Delta_{(l, \epsilon)}(s) - \bar{\Delta}(s)\|}{\|\bar{\Delta}(s)\|}, \quad (9)$$

subject to

$$0 < \epsilon < 1;$$

$$\Delta_{(l, \epsilon)}(s) = \frac{\prod_{j=1}^{2g} \left( s^{\frac{1}{2}} + z_j(l, \epsilon) \right)}{\prod_{j=1}^{2g} \left( s^{\frac{1}{2}} + p_j(l, \epsilon) \right)}.$$

Algorithm 1 shows the pseudocode for our identification procedure. Note that  $J_{\min}$  is initialized to  $+\infty$  which means the largest real number determined by the machine in use.

Those half-order zeros and poles  $-z_j$  and  $-p_j$  are computed offline. Before the identification procedure shown in Algorithm 1, we sample  $-z_j$  and  $-p_j$  at different  $\epsilon_a$  and store them to a database. Then, during the identification, we use the piecewise linear interpolation to obtain  $-z_j$  and  $-p_j$  from those stored values for any  $0 < \epsilon < 1$ . Specifically, at each component  $l \in L$ , we pick 500  $\epsilon_a$ 's between 0 and 1. Then, for each pair of  $(l, \epsilon_a)$ , we use the method described in Section 2 to compute the corresponding  $-z_j(l, \epsilon_a)$  and  $-p_j(l, \epsilon_a)$ , and store them to a database.

**Result:** Identify the damaged component  $l^*$  and quantify its damage amount  $\epsilon^*$  given the measured  $\bar{\Delta}(s)$ .

```

 $J_{\min} \leftarrow +\infty;$ 
for  $l \in L$  do
  for The initial guess  $\epsilon_0 \in [0.1, 0.2, \dots, 0.9]$  do
    Find  $(\epsilon, J)$  such that  $\epsilon$  solves the optimization
    problem (9) at the component  $l$ , and  $J$  is the
    corresponding optimized identification error;
    if  $J < J_{\min}$  then
       $J_{\min} \leftarrow J;$ 
       $l^* \leftarrow l;$ 
       $\epsilon^* \leftarrow \epsilon;$ 
    end
  end
end

```

**Algorithm 1:** Pseudocode for our identification procedure. Note that the double **for** loop here can be easily implemented in parallel.

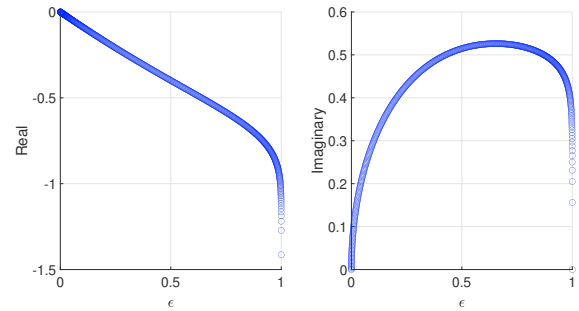


Fig. 3. Sampling of the real and imaginary part for the half-order zero  $-z_7(k_{4,7}, \epsilon_a)$  where  $a = 1, 2, \dots, 500$  and  $\epsilon_a$  is determined by Eq. (10).

For example, Fig. 3 shows the real and imaginary part of the half-order zero  $-z_7(k_{4,7}, \epsilon_a)$  for  $a = 1, 2, \dots, 500$ . Note that, for the damaged tree model,  $-z_j$  and  $-p_j$  are usually very sensitive as  $\epsilon \rightarrow 0$  and  $\epsilon \rightarrow 1$ . Therefore, we pick discrete  $\epsilon_a$  based on the Chebyshev nodes, that is

$$\epsilon_a = \frac{1}{2} \left[ \cos \left( \frac{2a-1}{1000} \pi \right) + 1 \right], \quad a = 1, 2, \dots, 500. \quad (10)$$

Then, during the identification, we use piecewise linear interpolation to obtain half-order zeros and poles,  $-z_j$  and  $-p_j$ , for any  $0 < \epsilon < 1$ . That is,

$$\begin{aligned} -z_j(l, \epsilon) &= -z_j(l, \epsilon_a) \cdot \left( 1 - \frac{\epsilon - \epsilon_a}{\epsilon_{a+1} - \epsilon_a} \right) \\ &\quad - z_j(l, \epsilon_{a+1}) \cdot \frac{\epsilon - \epsilon_a}{\epsilon_{a+1} - \epsilon_a}, \end{aligned} \quad (11)$$

$$\begin{aligned} -p_j(l, \epsilon) &= -p_j(l, \epsilon_a) \cdot \left( 1 - \frac{\epsilon - \epsilon_a}{\epsilon_{a+1} - \epsilon_a} \right) \\ &\quad - p_j(l, \epsilon_{a+1}) \cdot \frac{\epsilon - \epsilon_a}{\epsilon_{a+1} - \epsilon_a}, \end{aligned} \quad (12)$$

where  $\epsilon_a \leq \epsilon < \epsilon_{a+1}$ . For example, Fig. 4 shows the locus of half-order zeros and poles when the damaged component  $l = k_{2,1}$  after interpolation. Those arrows in Fig. 4 indicate the direction along which  $-z_j(k_{2,1}, \epsilon)$  and  $-p_j(k_{2,1}, \epsilon)$  move when  $\epsilon$  varies from 1 (undamaged) to 0 (completely damaged).

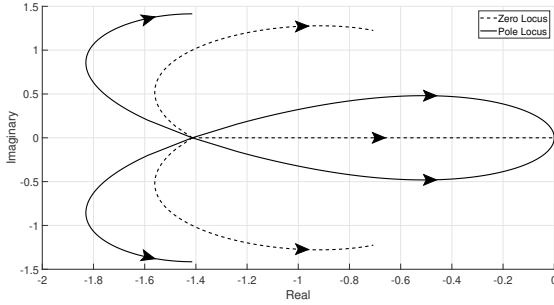


Fig. 4. Locus for half-order zeros and poles,  $-z_j(k_{2,1}, \epsilon)$  and  $-p_j(k_{2,1}, \epsilon)$  for  $j = 1, 2, 3, 4$  and  $\epsilon$  varying from 1 (undamaged) to 0 (completely damaged).  $-z_1(k_{2,1}, \epsilon)$  stays at  $-\sqrt{\frac{k}{b}}$  for all  $0 < \epsilon < 1$ .

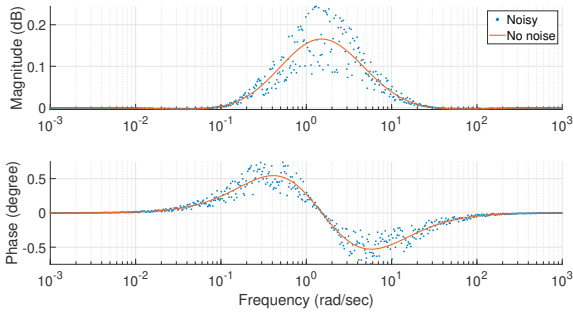


Fig. 5. Bode plot of  $\bar{\Delta}(s)$  to which 50% noise is added when the damage case  $(l, \epsilon) = (k_{3,2}, 0.5)$ .

#### 4. IDENTIFICATION TEST RESULTS

We test our identification procedure on all damage cases where the damaged component is located in the first three generations and each damaged component has ten different damage amounts. That is, each damage case  $(l, \epsilon)$  during the test is an element of the following Cartesian product

$$\{k_{1,1}, b_{1,1}, \dots, k_{3,4}, b_{3,4}\} \times \{0.05, 0.15, \dots, 0.95\}.$$

Therefore, 140 different damage cases are tested in total. During the test, we use `fmincon()` from MATLAB to solve the nonlinear programming problem (9).

To imitate real measurements, we add noise to the analytical value of  $\bar{\Delta}(s)$ . Here is what we mean by adding  $n_{\max}\%$  noise to  $\bar{\Delta}(s)$ : If the analytical value of  $\bar{\Delta}(s) = A + i \cdot B$  at some angular frequency  $\omega$ , what the identification procedure can see is its corresponding noisy value of  $\bar{\Delta}(s)$  where

$$|\bar{\Delta}(s)| = 10^{(1+n\%) \cdot \log_{10}(\sqrt{A^2+B^2})},$$

$$\angle \bar{\Delta}(s) = (1+n\%) \cdot \text{atan2}(B, A),$$

and  $n$  is a uniformly distributed random variable between  $-n_{\max}$  and  $n_{\max}$ . For example, Fig. 5 shows the Bode plot of  $\bar{\Delta}(s)$  to which 50% noise is added when the damage case  $(l, \epsilon) = (k_{3,2}, 0.5)$ . Note that, by doing so, the identification result depends on the value of the random variable  $n$  chosen by MATLAB. As a result, to accommodate that dependence, we test our identification algorithm ten times at each level of added noise.

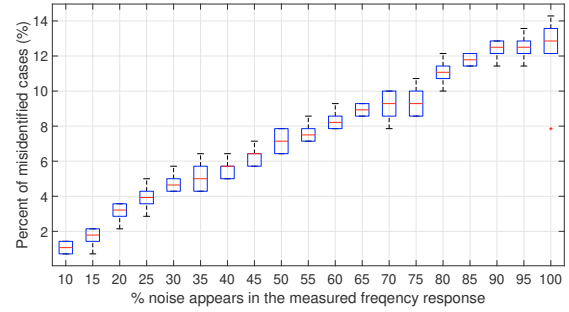


Fig. 6. Box plot for the percentage of misidentified cases out of total 140 damage cases versus the percentage of noise added to  $\bar{\Delta}(s)$ . Ten runs for each level of added noise.

Table 3. Average percentage of misidentified cases for components on the third generation.

Damaged components	Average % of misidentified cases
$k_{3,1}, b_{3,1}$	0.5%
$k_{3,2}, b_{3,2}, k_{3,3}, b_{3,3}$	20.75%
$k_{3,4}, b_{3,4}$	2%

When no noise presents in the measured  $\bar{\Delta}(s)$ , that is when the measured  $\bar{\Delta}(s)$  is same as its theoretical value, our method correctly identifies all 140 damage cases. The maximum absolute difference between the actual damage amount  $\epsilon$  and the identified  $\epsilon^*$  is  $1.89 \times 10^{-5}$  which happens at the damage case  $(l, \epsilon) = (b_{3,1}, 0.45)$ .

When measurement noise exists in  $\bar{\Delta}(s)$ , some misidentified cases appear. However, our identification method still works well. For example, when we add 100% noise to  $\bar{\Delta}(s)$ , only about 13% of total 140 cases are misidentified on average. In the following, we list four observations.

- (1) When more noise is added to  $\bar{\Delta}(s)$ , more misidentified cases happen. We can observe such trend from Fig. 6.
- (2) For the same level of noise, a damage case which happens at a deeper generation is more likely to be misidentified.
- (3) For the same level of noise, and for the damage cases which happen at the same generation, those cases occurring at inner components are more inclined to misidentification compared to those occurring at outer components.
- (4) For the same level of noise, and for the same damaged component, misidentification happens more frequently when the damage amount  $\epsilon$  is close to 1 (undamaged).

The above observations from (2) to (4) are shown next based on ten different runs where 50% noise is added to  $\bar{\Delta}(s)$ . Fig. 7 shows the percentage of the misidentified cases at each generation, from which we can confirm the observation (2). For the observation (3), we focus on the third generation where we call  $k_{3,1}, b_{3,1}, k_{3,4}, b_{3,4}$  outer components, and  $k_{3,2}, b_{3,2}, k_{3,3}, b_{3,3}$  inner components. Table 3 shows average percentage of misidentified cases for components on the third generation, from which we can confirm the observation (3). For the observation (4), we focus on those inner components at the third generation which have most misidentified cases. Fig. 8 plots the



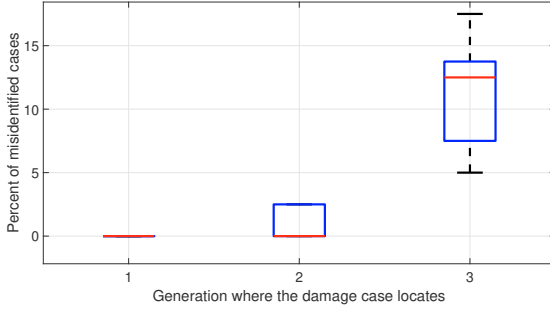


Fig. 7. Box plot for the percentage of the misidentified cases at each generation. Ten runs when 50% noise is added.

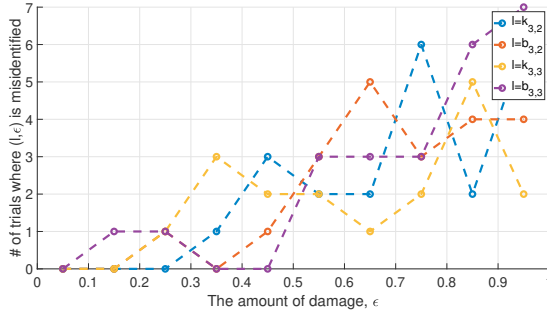


Fig. 8. Number of trials out of 10 total trials where each damage case  $(l, \epsilon)$  is misidentified *versus* the amount of damage  $\epsilon$ .

number of trials out of total 10 runs where each damage case  $(l, \epsilon)$  is misidentified *versus* the amount of damage  $\epsilon$ , from which we can confirm the observation (4).

Finally, to summarize the observations from (2) to (4), we plot the second best identification error against both the index of the damaged component  $l$  and the amount of damage  $\epsilon$  in Fig. 9. Note that Fig. 9 is based on only one of those ten trials where 50% noise is added to  $\bar{\Delta}(s)$ , but plots for all the other nine trials are qualitatively similar. Recall that in our identification procedure, we solve an optimization problem (9) to find the minimum identification error at each component. Then, among all those locally best identification errors, the globally smallest one gives the final identification result. Therefore, for a damage case  $(l, \epsilon)$ , how small its globally second best identification error is indicates how easily that damage case can be misidentified. Here, in order to make plotting convenient, we index the components up to the third generation by integers from 1 to 14, that is,  $k_{1,1} \rightarrow 1$ ,  $b_{1,1} \rightarrow 2$ , ...,  $b_{3,4} \rightarrow 14$ . Hence, indices  $\{1, 2\}$  represent components on the first generation, indices  $\{3, 4, 5, 6\}$  represent components on the second generation, and indices  $\{7, 8, \dots, 14\}$  represent components on the third generation. Moreover,  $\{9, 10, 11, 12\}$  represent the inner components on the third generation. Therefore, from Fig. 9, we can get an overall idea for the observations (2) to (4).

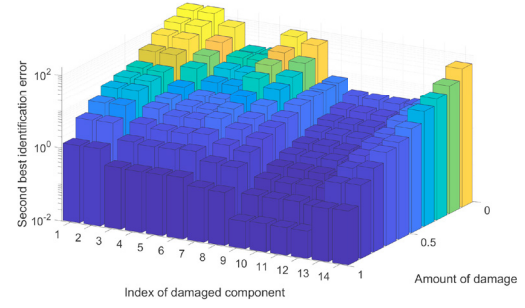


Fig. 9. Second best identification error *versus* the index of damaged component  $l$  and the amount of damage  $\epsilon$ .

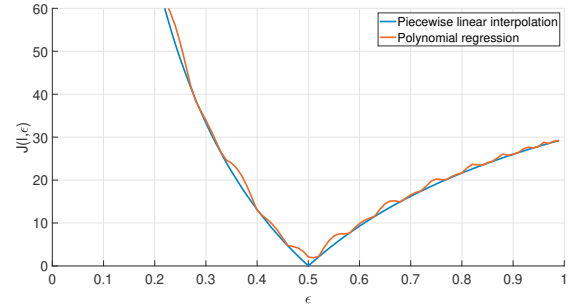


Fig. 10. Comparison between two  $J(k_{2,1}, \epsilon)$  where the measured  $\bar{\Delta}(s)$  is for  $(l, \epsilon) = (k_{2,1}, 0.5)$ . The blue one is obtained by using piecewise linear interpolation to construct a continuous mapping from  $\epsilon$  to  $z_j$  and  $p_j$  based on those sampled values  $z_j(k_{2,1}, \epsilon_a)$  and  $p_j(k_{2,1}, \epsilon_a)$ . The red one uses the polynomial regression to construct that continuous mapping.

## 5. DISCUSSION

### 5.1 Justification for using piecewise linear interpolation to make the mapping from $\epsilon$ to $z_j$ and $p_j$ continuous

As described in Section 3, before the damage identification, we sample  $z_j$  and  $p_j$  at discrete  $\epsilon_a$  which is picked as stated by Eq. (10). Then, that mapping is made continuous using piecewise linear interpolation so that  $z_j$  and  $p_j$  can be computed for all  $0 < \epsilon < 1$  according to Eqs. (11) and (12).

The most important consideration of picking a suitable interpolation method in this case is that it does not create extra saddle points and local minima in the identification error  $J(l, \epsilon)$ , since those would prevent the decision variable from converging to the actual minimizer. Fig. 10 compares two different  $J(k_{2,1}, \epsilon)$  *versus*  $\epsilon$  where the  $\bar{\Delta}(s)$  is for the damage case  $(l, \epsilon) = (k_{2,1}, 0.5)$ . One  $J(k_{2,1}, \epsilon)$  is obtained by using piecewise linear interpolation to construct a continuous mapping from  $\epsilon$  to  $z_j$  and  $p_j$  based on those sampled values  $z_j(k_{2,1}, \epsilon_a)$  and  $p_j(k_{2,1}, \epsilon_a)$ . The other  $J(k_{2,1}, \epsilon)$  is obtained by using polynomial regression to construct that continuous mapping. From Fig. 10, we see that the  $J(k_{2,1}, \epsilon)$  obtained by polynomial regression has a lot of oscillations. Such behavior is due to the oscillatory nature of polynomial regression known as Runge's phenomenon, and it is undesirable. Should another interpolation method lead to a smooth  $J(l, \epsilon)$  similar to the blue curve in Fig. 10, it can also be used.

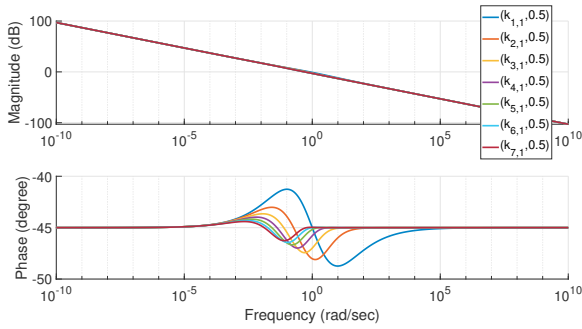


Fig. 11. The tree's overall frequency responses for 7 different damage cases where the damaged component is  $k_{1,1}$  through  $k_{7,1}$ , and the damage amount stays at 0.5.

### 5.2 Effects brought by a deeper damage

There are three aspects of consequences when a damage goes deeper inside the tree. First, a deep damaged component is naturally difficult to be identified as it has little effect on the overall frequency response of the tree. Fig. 11 plots the frequency response of the same damage amount when that damage goes from the first generation to the seventh, from which we can see that the discrepancy between two curves is less obvious for a deeper damage.

Second, a deeper damage requires a larger database to store discrete samples of  $z_j$  and  $p_j$ , which thus also requires more time to construct that database. Note that from the first to the  $n$ -th generation, there are  $2^{n+1} - 2$  components in total. Therefore, the size of that database doubles for each generation deeper. As for the time required to construct that database, we note that the computation of a damage case is always based on other damage cases which have already been stored in that database. Therefore, the computation time of a new damage case would not be significantly affected by its generation. As a result, the total time consumption to construct that database is proportional to its size, and thus it would also double for each generation deeper.

Third, a deeper damage requires more iterations during our identification procedure, because the size of  $L$  increases. As a result, the outer for loop in Algorithm 1 iterates more. For the same reason explained in the above paragraph, the size of  $L$  doubles for each generation deeper, so the total running time would also double. Note that the time consumption inside the outer for loop is independent of how deep a damage case is.

## 6. CONCLUSION

In this paper, we propose a method to identify the damaged component  $l$  and quantify its damage amount  $\epsilon$  in a damaged tree given its overall frequency response. Our identification procedure iterates through all possible components and solve a nonlinear programming problem (9) at each component. Formulation of that optimization problem takes advantage of our previous work about modeling the damaged tree model as  $G_\infty(s) \cdot \Delta_{(l,\epsilon)}(s)$  where  $\Delta_{(l,\epsilon)}(s)$  is completely determined by its corresponding damage case  $(l,\epsilon)$  as shown in Eq. (3). In addition, the performance of that identification algorithm and the effects brought

by a damaged component at a deep generation are also discussed in this paper.

## REFERENCES

- Rune Brincker, Lingmi Zhang, and Palle Andersen. Modal identification of output-only systems using frequency domain decomposition. *Smart materials and structures*, 10(3):441–445, 2001.
- Todd C Doehring, Alan D Freed, Evelyn O Carew, and Ivan Vesely. Fractional order viscoelasticity of the aortic valve cusp: an alternative to quasilinear viscoelasticity. *Journal of Biomechanical Engineering*, 127(4):700–708, 2005.
- Elżbieta Gabryś, Marek Rybaczuk, and Alicja Kędzia. Fractal models of circulatory system. symmetrical and asymmetrical approach comparison. *Chaos, Solitons & Fractals*, 24(3):707–715, 2005.
- Tom T Hartley and Carl F Lorenzo. Fractional-order system identification based on continuous order-distributions. *Signal processing*, 83(11):2287–2300, 2003.
- Nicole Heymans and J-C Bauwens. Fractal rheological models and fractional differential equations for viscoelastic behavior. *Rheologica acta*, 33(3):210–219, 1994.
- Jer-Nan Juang and Richard S Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of guidance, control, and dynamics*, 8(5):620–627, 1985.
- Kevin Leyden. *System Monitoring by Tracking Fractional Order*. PhD thesis, University of Notre Dame, April 2018.
- Kevin Leyden and Bill Goodwine. Fractional-order system identification for health monitoring. *Nonlinear Dynamics*, 92(3):1317–1334, 2018.
- Da-Yan Liu, Taous-Meriem Laleg-Kirati, Olivier Gibaru, and Wilfrid Perruquetti. Identification of fractional order systems using modulating functions method. In *2013 American Control Conference*, pages 1679–1684. IEEE, 2013.
- Barry R Masters. Fractal analysis of the vascular tree in the human retina. *Annu. Rev. Biomed. Eng.*, 6:427–452, 2004.
- Jason Mayes. *Reduction and approximation in large and infinite potential-driven flow networks*. PhD thesis, University of Notre Dame, April 2012.
- Alain Oustaloup. *La dérivation non entière*. Hermes, Paris, 1995.
- Bart Peeters and Guido De Roeck. Stochastic system identification for operational modal analysis: a review. *J. Dyn. Sys., Meas., Control*, 123(4):659–667, 2001.
- Michael J Roemer and Gregory J Kacprzynski. Advanced diagnostics and prognostics for gas turbine engine risk assessment. In *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*, volume 6, pages 345–353. IEEE, 2000.
- JZ Sikorska, Melinda Hodkiewicz, and Lin Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical systems and signal processing*, 25(5):1803–1836, 2011.
- Duarte Valério and José Sá Da Costa. *An introduction to fractional control*, volume 91. IET, 2013.
- Shengxi Zhou, Junyi Cao, and Yangquan Chen. Genetic algorithm-based identification of fractional-order systems. *Entropy*, 15(5):1624–1642, 2013.