Structural Engineering and Mechanics, *Vol. 63, No. 4 (2017) 457-469*
DOI: https://doi.org/10.12989/sem.2017.63.4.457

Parallel damage detection through finite frequency changes on multicore processors

Arcangelo Messina[a] and Massimo Cafaro[*]

Dipartimento di Ingegneria dell'Innovazione, Università del Salento, Lecce, Italy

(Received January 5, 2017, Revised February 15, 2017, Accepted May 14, 2017)

Abstract. This manuscript deals with a novel approach aimed at identifying multiple damaged sites in structural components through finite frequency changes. Natural frequencies, meant as a privileged set of modal data, are adopted along with a numerical model of the system. The adoption of finite changes efficiently allows challenging characteristic problems encountered in damage detection techniques such as unexpected comparison of possible shifted modes and the significance of modal data changes very often affected by experimental/environmental noise. The new procedure extends MDLAC and exploits parallel computing on modern multicore processors. Smart filters, aimed at reducing the potential damaged sites, are implemented in order to reduce the computational effort. Several use cases are presented in order to illustrate the potentiality of the new damage detection procedure.

Keywords: damage detection; natural frequency changes; MDLAC; parallel computing; inverse methods

1. Introduction

Damage identification methods, based on changes of modal data or, more in general, vibration-based data, have attracted the attention of researchers in the past thirty years. Hundreds of papers have been written all over the world, including reviews or technical reports, trying to recap previous work in the last twenty years; in particular, the list of references (Dimarogonas 1996, Salawu 1997, Doebling *et al*. 1998, Staszewski 1998, Farrar *et al*. 2001, Chang *et al*. 2003, Sohn *et al*. 2004, Montalvao *et al*. 2006, Taha *et al*. 2006, Su *et al.* 2006, Friswell 2007, Worden *et al.* 2007, Yan *et al.* 2007, Fan *et al.* 2011, Thatoi *et al.* 2012, Sinou 2013, Hakim *et al.* 2014) regards reviews with a rate of about one paper per year. Within such a vast scientific overview the vibration-based structural health monitoring as mean of diagnosis of unhealthy structure (existence of damage) is generally also intended as an activity aimed at identifying location of damage. The extent of the damage and of the residual life of the structure are also recognized of engineering interest (Rytter 1993), but both existence and location are still playing a significant role in all areas interested by structural health monitoring (*i.e.* aeronautical, civil and mechanical engineering).

The problem of vibration-based structural health monitoring is essentially inverse with non-unique solution and a few data available, polluted both by noise and ambient influences. The appeal of this problem is related to the expectation of being able to extract diagnosing information without knowing in advance where the possible damage is placed. First traces of such a research activity can be found around the 70' (Adams *et al.* 1978). The development of modern digital systems and the reduced costs of devices have also contributed to additionally increase ideas, thus resulting in a remarkable increasing amount of literature. In broad terms, we could try to recap the state of the art of the proposed methodologies by looking at the existence of two approaches: (*i*) vibration-based structural health monitoring by only using the actual system and (*ii*) vibration-based structural health monitoring by including a comparison of the actual system to its original healthy state. Along with such proposed methodologies, the nature of adopted data also plays a fundamental role in distinguishing different approaches.

In this latter regard since Adams *et al.* (Adams *et al.* 1978) launched the idea of using natural frequencies, several researchers based their investigations on these particular set of data simply because such a set is privileged for several reasons: (*i*) its measurement is cheap if compared to the measurement process of other modal data (*i.e.* damping or mode shapes), (*ii*) its measurement is quite stable, accurate and relatively less polluted by noise (e.g., (Messina *et al.* 1996)) and, finally, (*iii*) each natural frequency which is a potential messenger of the health state of the system can, in principle, be measured without necessarily knowing a priori the potential damaged location. Of course, such a set of modal data is not only bearer of good news; we cannot indeed forget that N natural frequencies are just a list of N numbers and, even though their N variations symptomatically can represent that something in the system under test is changing, we have not any direct correlation between such varied numbers and the same system. Therefore, even though natural frequency changes could serve as a warning against insurgent damages

∗Corresponding author, Professor
E-mail: massimo.cafaro@unisalento.it
[a]Professor
E-mail: arcangelo.messina@unisalento.it

ISSN: 1225-4568 (Print), 1598-6217 (Online)

(detection), key information correlating the frequencies changes (between health ad damaged structures) with the geometry of the system under test is needed to single out damaged places (location); to this end the numerical model is the key information providing the correlation between natural frequency changes (differential or finite) and the system.

The attraction of using such procedures (based on frequency changes), for an analyst, becomes therefore quite clear especially when we realize that the literature is recently re-evaluating frequency shifts; indeed, ref. (Sinou 2013) recognizes that even if a lot of robust and new tools based on linear measurements have been developed, using frequency shifts to detect damage appears to be more practical in engineering applications. On the other hand, however, the analyst should also be willing to implement a related numerical model whenever his/her analysis is related with detection and placement of damage. Based on the above premise in 1998 (Messina *et al.* 1998) a damage identification procedure driven by the optimization of a specific objective function (MDLAC: Multiple Damage Location Assurance Criterion) was introduced and is here taken into account as a starting base. MDLAC was later used by other researchers in order to extend or generalize the method and/or using itself as a base for comparisons versus other proposals. The so called MDLAC-strategy in ref. (Messina *et al.* 1998) was based on the advantage of using a privileged class of experimental data (natural frequencies) along with the possibility to implement the method for any type of structure (beam, plates, truss, frames etc.). As extensively discussed in ref. (Messina *et al*. 2012), MDLAC was initially introduced (Messina *et al.* 1998) in conjunction with the sensitivity frequencies of the system around its healthy condition and thus basing the approach on a first order perturbation of the natural frequencies around the healthy condition; in this regard the MDLAC approach, based on the classification of ref. (Friswell 2007), would belong to the class of sensitivity methods.

The damage at its early stage is well represented through a differential formulation, although the frequency shifts could more easily be masked by shifts due to influences which are not properly related with occurring damage (i.e., ambient and/or noise influences); therefore, the present work is aimed at extending the MDLAC approach from a differential formulation to a finite formulation by simultaneously keeping the adoption of a privileged class of experimental data: natural frequencies. The criterion based on MDLAC is also kept, because MDLAC, being based on a statistical comparing criterion, showed even the ability to slightly go over a differential formulation, although an increasing number of damaged sites along with remarkable finite damage could create uncertain identifications. The extension of the classical MDLAC approach (Messina *et al.* 1998) to use finite frequency changes is herein carried out exploiting parallel computing on multicore processors. The challenge related to the computational effort required is overcome through the selection of a reduced set of possible scenario-solutions; in this latter regard, two different criterions are introduced along with recommendations to efficiently implement them in parallel. The analysis, the criterions and the implementations are carried out within the frame of large complex structures and also by accounting for the reality, uncommonly taken into account in the literature, i.e., distinguishing between finite elements with damaged places.

2. Theoretical formulation of the damage detection method

Let us call $\Delta\mathbf{f}_e$ the column vector containing the relative measured frequency changes which are experimentally measurable (N) and occurring between the healthy and damaged states of the system, *i.e.* $\Delta\mathbf{f}_e = \left(\left(f_1^{healthy} - f_1^{damaged}\right)/f_1^{healthy}, \ldots, \left(f_N^{healthy} - f_N^{damaged}\right)/f_N^{healthy}\right)^T$.

If a numerical model of the system is available, a correlation between the finite measured frequency changes ($\Delta\mathbf{f}_e$) and a generic set of relative numerical frequency changes ($\Delta\mathbf{f}_n$) can be evaluated through the following equation.

$$\cos(\Delta\mathbf{f}_e, \Delta\mathbf{f}_n)^2 = \frac{\left(\Delta\mathbf{f}_e^{\mathrm{T}}\Delta\mathbf{f}_n\right)^2}{\left(\Delta\mathbf{f}_e^{\mathrm{T}}\Delta\mathbf{f}_e\right)\left(\Delta\mathbf{f}_n^{\mathrm{T}}\Delta\mathbf{f}_n\right)} \tag{1}$$

Eq. (1) corresponds to an extended version of MDLAC (Messina *et al.* 1998) which was originally implemented between finite frequency changes compared to differential approximations. Eq. (1) also stresses how the MDLAC criterion is established through the squared cosine of two vectors (e.g., Horn *et al.* 1985) straightforwardly clarifying the reason why the result belongs to [0, 1], with 0 indicating no correlation and 1 indicating an exact match between the patterns of finite frequency changes. Any other value of the squared cosine in the middle between (0, 1) is assumed as measuring a correlation between pairs of vectors. An alternative to criterion (1), aimed at comparing vectors, could be, for example, the Euclidean distance between vectors. However, this proved to be experimentally not effective as the criterion established by the squared cosine (1). The numerical model we assume to have available consists essentially of two matrices (mass and stiffness matrix) related through Eq. (2) to the relevant modal parameters of the system (natural frequencies in Hz: $f_k = \omega_k/2\pi = \sqrt{\lambda_k}/2\pi$)

$$(\mathbf{K} - \lambda_k\mathbf{M})\boldsymbol{\phi}_k = 0 \tag{2}$$

In this work, we simulate an occurring damage as a uniform elemental reduction of the stiffness in N_e finite elements through their respective elemental stiffnesses. Such a uniform elemental reduction is realized by associating a relevant stiffness reduction coefficient D_j multiplying the structural element matrix ($\mathbf{K}_j$) as in (3) (Messina *et al.* 1998).

$$\mathbf{K} = \sum_{j=1}^{N_e} D_j\mathbf{K}_j \tag{3}$$

where, D_j=1 for no damage and D_j=0 for complete loss of the jth element (in this latter case the damage corresponds

to 100%).

2.1 Parallel implementation on multicore computers: the combinatorial method

The algorithm proposed in this work basically founds its rationale on Eqs. (1)-(3). Namely, we are looking for a testing damaged scenario $\mathbf{D}_t=(D_1, D_2, ..., D_{Ne})^T$ which applied to (3) is able to furnish a global stiffness matrix whose eigenvalues analysis through Eq. (2) provides finite natural frequency shifts (i.e., $\Delta\mathbf{f}_n$) perfectly correlated to the experimental eigenvalues shifts (i.e., $\cos(\Delta\mathbf{f}_e, \Delta\mathbf{f}_n)^2=1$ through Eq. (1)). In spite of the simplicity of this rationale, we could clearly realize that such a simple algorithm is unpractical or even impossible to carry out without any further consideration. Indeed, owing to the fact that the damage is unknown (placement and extents), the number of possible combinations of testing damaged scenarios is infinite; moreover, the required number of eigenvalue analyses (2) would be infinite as well. However, if we assume that our damaged system contains both a maximum number of damaged locations (N_d) and a maximum number of damage levels (N_l) in an established scale (for example we assume testing N_l=9 levels from 0% up to $L_{\max}$=80% by 10% steps), then the number of eigenvalue (N_a) analyses and the relevant comparisons would correspond to the following equation

$$N_a(N_e, N_d, N_l) = \binom{N_e}{N_d} N_l^{N_d} = \frac{N_e!}{N_d!\,(N_e - N_d)!} N_l^{N_d} \qquad (4)$$

Eq. (4) should be read through two essential terms, i.e., n_1 and n_2. In particular, n_1 refers to the binomial coefficient N_e choose N_d and multiplies $n_2=N_l^{Nd}$. In other terms, n_1 corresponds, all over the elements of the system, to the number of all possible combinations of N_d damaged elements. If we were certain about the existence of a single damage level, we would need to test, by comparing ($\Delta\mathbf{f}_e$, $\Delta\mathbf{f}_n$), only n_1 cases. Since we are assuming that each element can be subject to N_l possible damage levels (or close to each one of these levels), then each single combination of N_d elements in the population of n_1 cases can be subject to n_2 damaged scenarios. In total, we have to compare, as stated by equation (4), $N_a=n_1n_2$ pairs of frequency shifts ($\Delta\mathbf{f}_e$, $\Delta\mathbf{f}_n$) and aiming at choosing the respective scenarios providing the highest squared cosine (1). Therefore, on the basis of Eq. (4), the most critical computational aspects of the problem we are dealing with are the following two main points: (*i*) the need to store matrix $\mathbf{C}_{n1xNd}$ keeping all possible combinations of potentially damaged elements along with matrix $\mathbf{V}_{n1xNd}$ which keeps all possible damaging levels of all elements listed row by row in **C**; (*ii*) the need to solve the eigenproblem (2) $N_a = n_1n_2$ times.

Both matrices **C** and **V** play a significant role in our approach because they can become extremely large depending on the investigated scenario; in this regard, the analyst could save these matrices on disk or keep them into Random Access Memory (RAM). The practice of keeping (**C**, **V**) into the RAM is faster and should be always preferred as it has been carried out in this work; however, the required amount of RAM may be high. For example, if an analyst sets up four possible damaged elements (N_d=4) through 10 levels of possible damage (N_l=10) in a finite elements model constituted by N_e=250 elements, the required memory allocation in double precision for **C** and **V** would be 4.7 GB along with N_a=1,588,827,500,000 analyses (2). As a second example, a numerical FE-model made up of 930 elements (e.g., the model in ref. (Farrar *et al.* 2000)) would even require about 923 GB to store **C** and **V** whilst requiring N_a=309,681,406,800,000 analyses. The above mentioned computational efforts could seem unfeasible but they should be read within the context of the most modern workstations and computers available today. Indeed, current workstations can be easily configured with 1024 GB of RAM and several multicore processors; even high-end laptops can be equipped with 64 GB of RAM along with quad core processors. The configuration and performances of computers dramatically increase when high-performance computing systems (HPC) are taken into account. With regard to the TOP500 List of June 2016 (TOP500.org 2017), the number one HPC system Sunway TaihuLight is equipped with 10,649,600 cores and 1,310,720 GB of memory. Therefore, if the proposed damage detection procedure would be solely implemented through Eqs. (1)-(3) on the most intensive above mentioned case (N_e=930, N_l=10, N_d=4) its computational effort on the Sunway TaihuLight, would be abated to at most N_a=29,079,159 analyses per core.

It is finally stressed that each analysis among the population of N_a analyses is absolutely independent and can be carried out without requiring any intermediate results extraneous to the specific and single analysis; from a computational point of view such a situation requires implementing a straightforward embarrassingly parallel application.

2.2 Filters excluding improbable damaged scenarios

In spite of the possibility offered by the most modern computers equipped with multicore processors, still there exist the need to reduce the number of eigenvalue analyses (2) (i.e., $N_a=n_1n_2$). In this regard two distinct filters (or procedures) are herein presented in order to exclude a number of elements as being potentially undamaged and thus reducing the number of eigenvalue analyses. Both the procedures are based on a linear approximation and are herein tested along with the combinatorial method described in Section 2.1. We shall also test one of the procedures by proving that it is able by itself, *i.e.* independently from the combinatorial method, to coarsely single out large regions where the damage could be placed.

2.2.1 Filter 1: Large-scale damage detection method

This method is essentially designed in order to coarsely locate large areas where the damage can be potentially placed. For this reason, such a method is implemented to directly detect damaged large areas rather than aimed at complementarily excluding potentially undamaged areas. Of course, if the method were able to properly single out such large areas where the damage is potentially placed it would be able to provide an indication regarding the

elements which are undamaged and should be excluded from any more accurate subsequent investigation. The method is initially funded on the following equation which is based on a linear approximation

$$\delta f_k = f_{k,D_1}\delta D_1 + \ldots + f_{k,D_{Ne}}\delta D_{Ne} \qquad k = 1,\ldots, N \tag{5}$$

between a general damaged scenario (δD_1, δD_2,..., δD_{Ne}) and its consistent variation of measurable natural frequencies (δf_1, δf_2,..., δf_N) where $f_{k,Dj}$ indicates $\partial f_k/\partial D_j$ with j=1,...,N_e. Eq. (5) states an evident differential relationship among the above mentioned quantities; such a relationship is however, based on the choice of looking at any element (for j=1,..., N_e) as bearer of a potential damage. This choice is strategically the simplest one and it is usually adopted in all the algorithms proposed in the literature. However, we cannot ignore that elements are usually part of a mesh which is conditioned by convergence criterions, technological needs etc. Above all, any valid reason aimed at believing that a damage must occur in single elements does not exist at all. Conversely, when an FE-model has been built up along with its mesh, we should expect a damaged zone generally different by a damaged element. In other words, we should introduce the concept of zones that can contain damage and, more specifically, (*i*) zones which are arranged by the analyst depending on the investigated model within its FE-scheme, (*ii*) a zone can fall in an element or in a group of elements, (*iii*) such a zone can even contain both damaged and undamaged elements. This latter eventuality is more in agreement both with macro-damage we are dealing with FE-schemes which are usually characterized by many elements for, at least, convergence criterions.

In this perspective, we could implement an algorithm by associating a damaging coefficient (Z_i with i=1,...,N_z) at each single zone rather than associating a damaging coefficient (D_j with j=1,…, N_e) at each single element. Based on this choice to group several elements in each zone, the number of zones will be less than or equal to the number of elements ($N_z \leq N_e$). Such a choice would transform Eq. (5) in Eq. (6)

$$\delta f_k = f_{k,Z_1}\delta Z_1 + \ldots + f_{k,Z_{Nz}}\delta Z_{Nz} \qquad k = 1,\ldots, N \tag{6}$$

where $f_{k,Zi}$ indicates $\partial f_k/\partial Z_i$ with i=1,...,N_z, and a uniform condition of damage per zone has been attributed. At this stage, by taking into account the uniqueness of a damaged zone and comparing Eqs. (5) and (6), for example with regard to zone 1 made of N_1 elements, the following equation holds.

$$f_{k,D_1}\delta D_1 + \ldots + f_{k,D_{N1}}\delta D_{N1} = f_{k,Z_1}\delta Z_1 \tag{7}$$

From Eq. (7), recalling the physical meaning of the sensitivity terms (i.e., property 2 in appendix of ref. (Contursi *et al.* 1998)) it is possible to realize that the formal position aimed at classifying N_1 elements in an established group essentially consists of evaluating an extent of damage for a zone which is a mean weighted through the frequency sensitivities (8).

$$\delta Z_1 = \frac{f_{k,D_1}\delta D_1 + \ldots + f_{k,D_{N1}}\delta D_{N1}}{f_{k,D_1} + \ldots + f_{k,D_{N1}}} \tag{8}$$

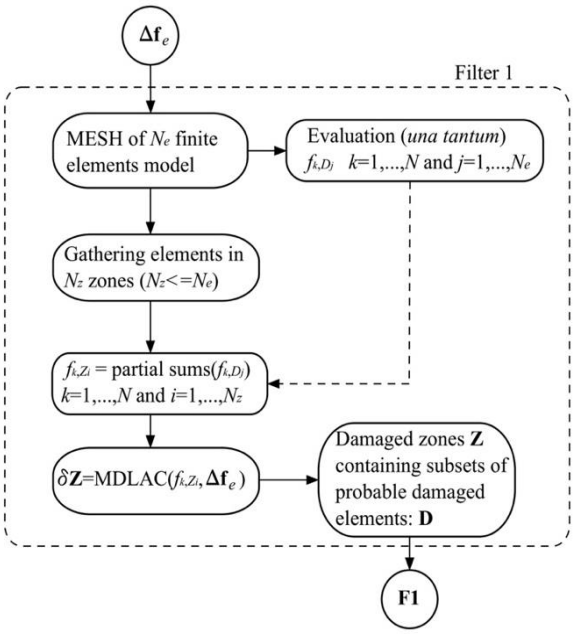


Fig. 1 Filter 1: flowchart

Eq. (8) through its simplicity proves some intriguing and non-negligible aspects. For example, if a high value of damage, even finite, is present only in one element or in a few ones of the group, we should expect an evaluation of damage spread over the entire zone and thus leading to a δZ less than the true localized damage; such a minor extent of damage also goes in the direction of improving the differential relationship between damage and the consistent elementary variation of frequencies. Moreover, the problem of the real availability of experimental data (usually consisting of a few lower natural frequencies) is also mitigated: indeed, a method working with zones goes in the direction of reducing the gap between unknowns (damaged zones) and data (natural frequency changes). Therefore, an approach dealing with zones is generally expected to provide benefits.

A question remains open regarding the criterion of making up groups of elements in N_z zones where each zone is characterized by a uniform damage (δZ_i with i=1,...,N_z). In this work, we shall create zones by classifying elements having centroids placed at the lowest distance from a reference point. Once zones are defined, the algorithm herein used to identify damage shall be based on the classical MDLAC (Messina *et al.* 1998) implemented *per zones* rather than *per elements*. In both cases the function *fmincon*() built-in within the Matlab Optimization toolbox (MathWorks 2017) is used. MDLAC shall provide the damaged zones or equivalently shall implement a large-scale damage detection method, as well as allowing a subsequent focused application of the combinatorial method on a drastically reduced number of potentially damaged elements or subzones. Fig. 1 depicts the flowchart of this filter and clarifies that its output, as implemented in the present context, regards a set of elements probably damaged because belonging to the zones which have been detected as containing damage.

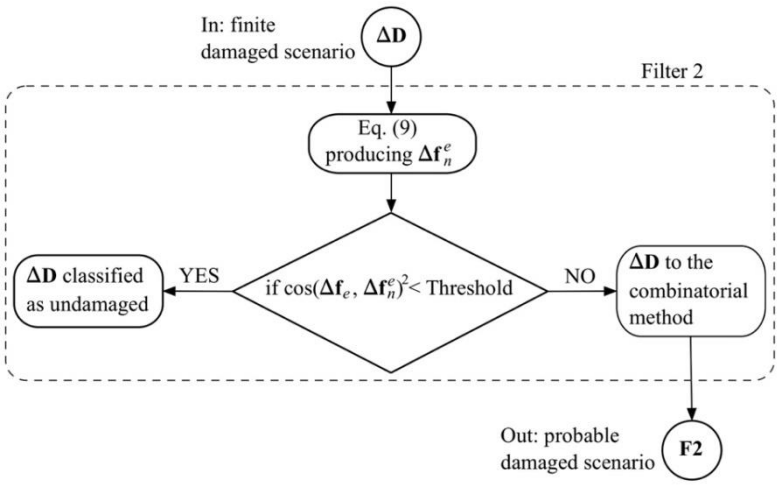


Fig. 2 Filter 2: flowchart

2.2.2 Filter 2: Sensitivity damage exclusion method

This filter is strictly designed in order to work for the combinatorial method. We have experimentally confirmed in all the implementations carried out in this work that the eigenvalue analysis (2) represents the most demanding part of the combinatorial method, requiring N_a (4) analyses. Therefore, this filter was designed in order to exclude certain N_d combinations reasonably undamaged. For example, N_a analyses contain combinations certainly and trivially undamaged such as all the n_1 rows in matrix **C** respect to the undamaged condition, which are strongly uncorrelated with any damaged scenario because in ideal conditions (uncontaminated measurements by noise and/or ambient influences) the relevant patterns of frequency changes $\Delta\mathbf{f}_n$ would be coincident with the nil vector. This latter statement can be generalized introducing the following filter: any simulated damaged scenario providing an uncorrelated pattern of frequency changes, which means providing a $\cos(\Delta\mathbf{f}_e, \Delta\mathbf{f}_n)^2$ lower than a threshold value, is excluded from the N_a analyses.

By looking at the combinatorial method, such a filter would seem to be unfeasible or resulting in a vicious circle; namely, in order to estimate if a simulated damaged scenario provides an uncorrelated pattern of frequency changes an eigenvalue analysis (2) is needed to produce the pattern of frequency changes. It would be, indeed, a vicious circle if we pretend to accurately compute the pattern of frequency changes due to the simulated damaged scenario; differently such a filter is applied by estimating (rather than accurately computing) the pattern of frequency changes due to the simulated damaged scenario. Such an estimation is herein carried out through the linear approximation of Eq. (9).

From Eq. (9), it is clear that the estimation of relative frequency changes is carried out by a linear approximation through a sensitivity matrix evaluated once for all at any beginning analysis. This implies that before carrying out an eigenvalue analysis a correlation test is carried out as reported in Fig. 2, which shows that operationally the present filter consists of making a matrix product (9) for subsequently carrying out the evaluation of a $\cos()^2$ (Eq. (1)), operations which are computationally much less demanding than an eigenvalue analysis (Eq. (2)). The latter is carried out only if the simulating damage vector $\Delta\mathbf{D}$ passes through the filter acquiring the qualification of probable damaged scenario.

$$\Delta\mathbf{f}_n^e = \begin{pmatrix} \Delta f_{n_1}^e / f_1^{healthy} \\ \vdots \\ \Delta f_{n_N}^e / f_N^{healthy} \end{pmatrix} = \begin{bmatrix} f_{1,D_1}/f_1^{healthy} & \cdots & f_{1,D_{Ne}}/f_1^{healthy} \\ \vdots & \ddots & \vdots \\ f_{N,D_1}/f_N^{healthy} & \cdots & f_{N,D_{Ne}}/f_N^{healthy} \end{bmatrix} \cdot \begin{pmatrix} \Delta D_1 \\ \vdots \\ \Delta D_{Ne} \end{pmatrix} \quad (9)$$

In order to implement the filter in Fig. 2, the analyst must set a threshold which can be set in the range [0, 1] where 0 means the filter 2 is completely bypassed whilst a threshold equal to 1 means a filter probably closed with the possibility to provide an empty set of damaged scenarios.

3. Implementation of the combinatorial method and Filters 1 & 2

In this Section the implementation regarding the combinatorial method along with the filters by themselves or in a mutual co-operation are illustrated by keeping in mind multicore processors used to design serial and parallel algorithms. The relevant numerical implementations have been based on the parallel computing toolbox (Ver. 6.6) implemented in Matlab (R2015a), running on a computer equipped with an Intel I7 2.5 GHz quad-core processor, and 16 GB of RAM.

As reported in Fig. 3, the starting point is based on a pattern of experimental finite frequency changes ($\Delta\mathbf{f}_e$) which alarms the analyst with regard to significant changes occurring on the system. Such a condition starts the operations aimed at searching the damaged scenario. Filter 1 is initially applied in order to single out probably damaged zones; these zones are constituted by a number R_e of elements (located in damaged zones) generally less than N_e. The searching method is then focused among these R_e elements once a maximum number of damaged elements (N_d) has been set through an established number of damaged levels (N_l). Therefore, all the possible

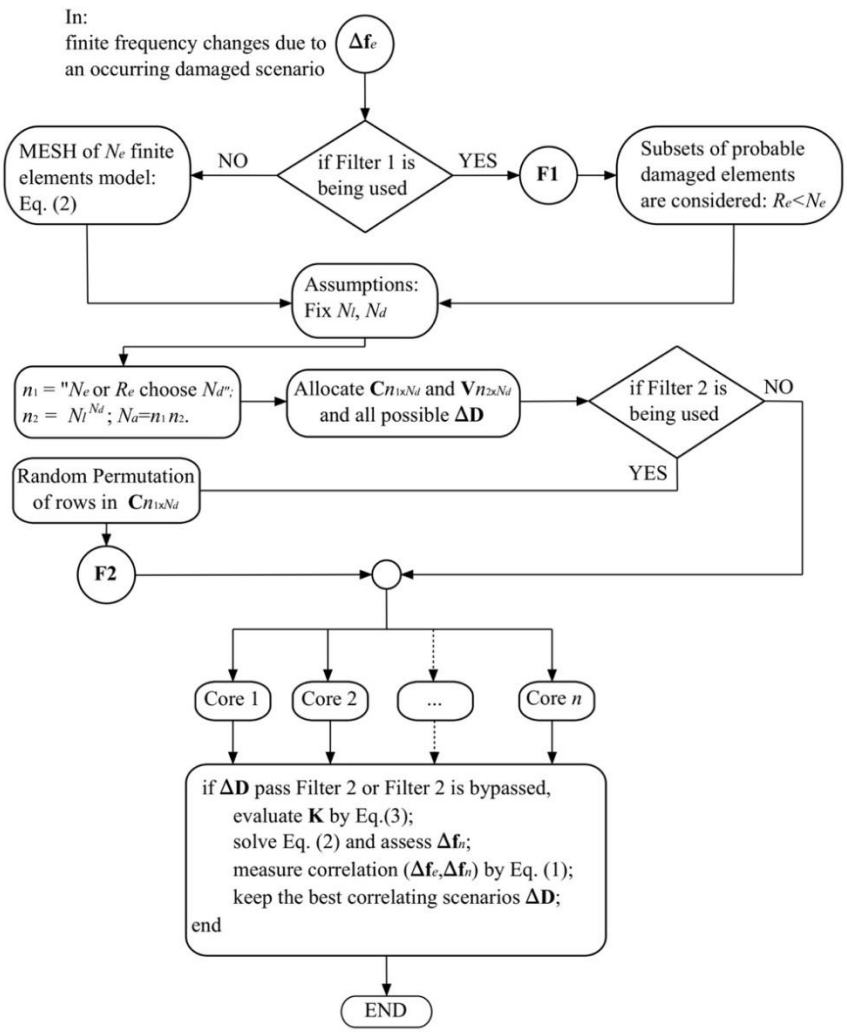


Fig. 3 The full procedure: flowchart

combinations of N_d elements are allocated in $\mathbf{C}_{n1xNd}$ and for each of its rows n_2 damaged scenarios ($\mathbf{V}_{n2xNd}$) should be tested by running Eqs. (1)-(3). At this stage filter 2 can save significant computational time; indeed, filter 2 dynamically allows comparing pairs ($\Delta\mathbf{f}_e$, $\Delta\mathbf{f}_n$) by running Eqs. (1)-(3) only when δf_n passes through the same filter. In this way, the eigenvalue analysis (Eq. (2)) is carried out only when it is certainly required, thus saving a significant amount of computational time.

Before closing this paragraph, a note is worth mentioning when the perspective is to carry out the comparisons among the residual damaged scenarios through an embarrassingly parallel application: the rows of $\mathbf{C}_{n1xNd}$ should be randomly permuted; if this recommendation were not followed the whole computational workload could be carried out by only one of the cores and the advantages of parallel computing could be entirely lost.

4. Numerical examples

The combinatorial method is herein tested versus two examples: (*i*) a clamped 2D Euler-Bernoulli beam meshed through 10 and 60 elements and (*ii*) a truss structure realized through 250 classical rod elements. These two examples are structurally different and were chosen in order to prove the capability of the combinatorial method to identify damaged locations even though only a few natural frequencies are available and independently from the structural case under test. In both cases filters 1 and 2 (Figs. 1, 2) are used along with the combinatorial method; however, the ability of filter 1 of identifying macro damaged areas is also illustrated independently of using or not the combinatorial method. The beam (Fig. 4) is characterized by the following material and geometric characteristics: length 1 *m*, density 7860 kg/m^3, Young's module 210 *GPa*, a square transversal section 1 *cm*×1 cm.

The second example (Fig. 5) consists of a truss tower (Sonnenhof Holdings 2013) 20 *m* high and made of 10 identical double frames (1 *m*×1 *m*×2 *m*) connected end to end. All joints are free to rotate in any direction. The 4 nodes at the base are constrained against displacement in all directions. All beams have the same material (200 *GPa* and density is 7800 kg/m^3) and whilst the chords at the corners have 0.01 m^2 cross sectional area, the bracing members have 0.001 m^2 cross sections.

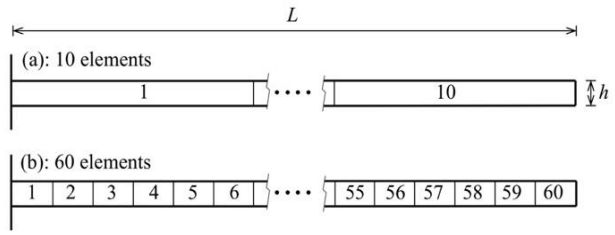


Fig. 4 Clamped 2D Euler-Bernoulli beam meshed through 10 and 60 elements; nomenclature

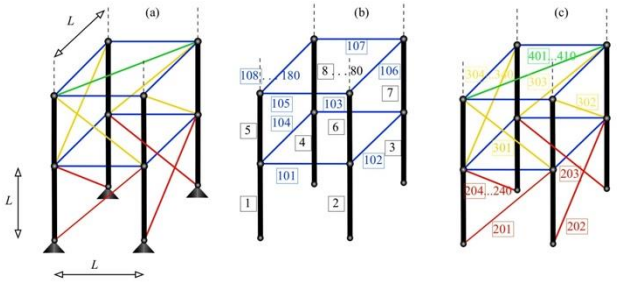


Fig. 5 Truss structure realised through 250 elements; basic group (double frame) and nomenclature (ref. (Sonnenhof Holdings 2013))

4.1 The case of a clamped beam

Based on Fig. 4 two meshes have been taken into account: 10 and 60 elements. With regard to the clamped beam the numeration of the elements follows the natural numbers through a unitary step. This is a particular case where the number of an element is representative of its location; in general, this is not true as Fig. 5 clearly shows. The natural frequencies slightly change when two meshes are used but this has not been considered significantly relevant with the strategy we are dealing with; $f^{healthy}$[Hz]MESH10=(8.349856, 52.32934, 146.5561, 287.3920, 475.8238, 712.8381, 1,000.146, 1,339.921, 1,732.5001, 2,153.748); $f^{healthy}$[Hz]MESH60= (8.349849, 52.32761, 146.5189, 287.1187, 474.6283, 709.0144, 990.2808, 1,318.431, 1,693.469, 2,115.403).

Fig. 6 illustrates in one graph three location charts of the damage with regard to the mesh of 10 elements. The white bars, in this and in all of the next figures, correspond to the simulated true damaged scenario whilst the red bars represent the identified damage based on the first ten natural frequency changes.

The red vertical bars illustrate the identified damage through the classical MDLAC (Messina *et al.* 1998), element by element, whilst the red horizontal bars illustrate the identified damage through the classical MDLAC based on zones rather than on elements. In this latter case Filter 1 has been adopted to identify the areas where the damage can be potentially located; in this regard the reference point was the clamped node whilst the zones containing the remainder elements (due to the fact that N_e/N_z is not usually an integer number) were randomly distributed all over the zones. Each horizontal bar has a length proportional to the number of elements grouped in its respective zone; i.e., counting the zones from left to right in Fig. 6 as (1, 2, 3, 4) they contain elements [(1, 2), (3, 4, 5), (6, 7, 8), (9, 10)]

Table 1 Damage detection results by using MDLAC based on only Filter 1 (ref. Fig. 4(a)); N_z=4, N=6 frequency changes; true damaged scenario: ΔD_1=30%, ΔD_6=60%

Run	Zone 1: dam%	Zone 2: dam%	Zone 3: dam%	Zone 4: dam%
1	1,2,3: 6.4	4,5,6: 47	7,8: no damage	9,10: no damage
2	1,2,3: 6.4	4,5: 59	6,7,8: 7.7	9,10: no damage
3	1,2: 26	3,4,5: no damage	6,7,8: 30	9,10: no damage
4	1,2: 23	3,4,5: no damage	6,7: 49	8,9,10: no damage

Table 2 Damage detection results by using MDLAC based on only Filter 1 (ref. Fig. 4(a)); N_z=3, N=6 frequency changes; true damaged scenario: ΔD_1=30%, ΔD_6=60%

Run	Zone 1: dam%	Zone 2: dam%	Zone 3: dam%
1	1,2,3,4: 10	5,6,7: 39	8,9,10: no damage
2	1,2,3: 6.4	4,5,6: 47	7,8,9,10: no damage

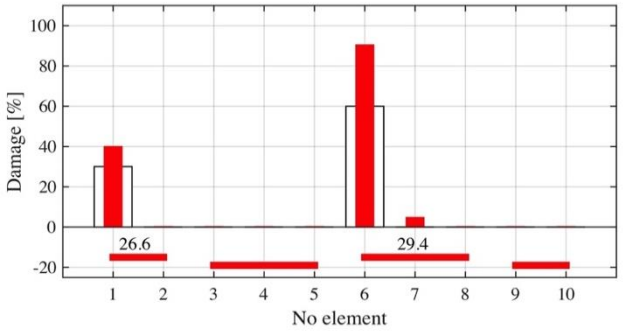
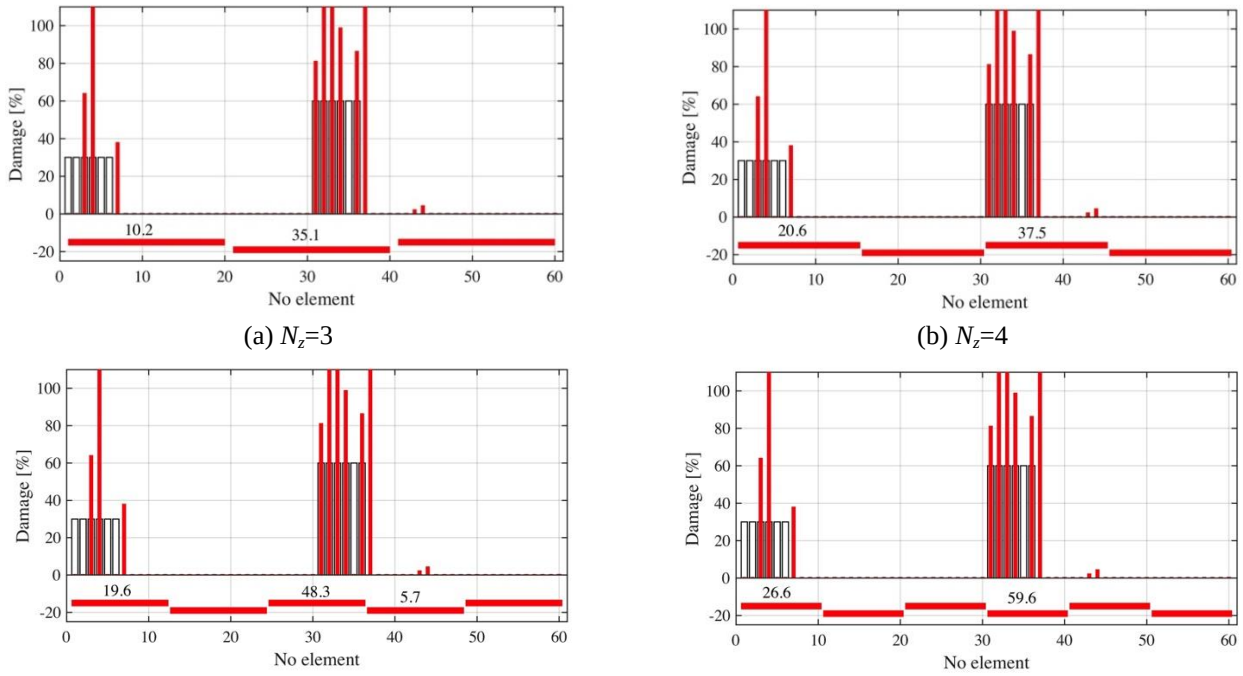


Fig. 6 Location chart through MDLAC for elements and for zones (MESH:10). The absolute % frequency changes (100xΔf_e) correspond to 8.63, 15.7, 5.55, 8.48, 6.32, 6.05, 6.98, 6.95, 5.80, 8.02

respectively. The percent damage is written close to each single horizontal bar if damaged through a significant percent.

Fig. 6 was based on the use of the first 10 natural frequency changes and clearly shows the capability of Filter 1 to correctly identify macro areas containing damaged places. A similar performance was obtained by using the first six natural frequencies as well as by meshing the whole beam in three distinct areas; in this regard Tables 1 and 2 illustrate the capability of Filter 1 to single out areas containing damaged elements. Tables 1 and 2 contain more lines with each respective simulation in order to show that the random grouping of elements in zones does not significantly affect the result. Finally, with regard to Fig. 6, it could be of a certain interest to know that MDLAC run on 10 elements took about 1 second whilst it took about 0.2 seconds when run on four zones. In practice, MDLAC run on zones was about 5 times faster.

Fig. 7 is the counterpart of Fig. 6 when the clamped beam is meshed on the base of 60 elements. In this regard, four sub cases were analyzed with (3, 4, 5, 6) zones respectively (corresponding to Fig. 7(a), 7(b), 7(c) and 7(d)). Fig. 7 clearly shows the capability of Filter 1 to accurately single out the macro areas where the damaged elements are placed. In this case the classical MDLAC ran



(a) N_z=3 (b) N_z=4

(c) N_z=5 (d) N_z=6

Fig. 7 Location chart through MDLAC for elements (MESH: 60) and for zones. The absolute % frequency changes (100xΔf_e) correspond to 8.63, 15.7, 5.55, 8.48, 6.33, 6.05, 7.09, 6.79, 6.49, 7.92

on 60 elements taking about 9 seconds whilst the same algorithm, when run per zones, took about 0.2 seconds almost independently of the number of zones (in this case, MDLAC run on zones was about 45 times faster); this latter result is evidently due to the marginal time required in all these cases where a few zones were adopted. However, in addition to all the above considerations we should recognize as impressive the capability of Filter 1 to single out macro areas containing damage or, complementarily, its capability to exclude others macro areas which could be considered healthy.

On the basis of the performance shown by Filter 1, it comes quite natural and advantageous the application of the combinatorial method (Fig. 3) aimed at investigating a reduced number of elements when a finite damage is expected to be localized in macro areas. Therefore, in order to show the capability of the combinatorial method, based on both serial and parallel computing, the specific damaged scenario involving elements number 3 and 33 (over a mesh of 60 elements) with a percent damage of 60% and 90% respectively, is carried out. Ten frequency changes are still taken into account and the combinatorial method depicted in Fig. 3 is applied on the basis of four zones grouping fifteen elements per zone, by assuming N_l=10, L_{max}=95%, *Threshold*=0 or 0.5 (Filter 2) and N_d=2 or 3. Here it is added the strategy suggested in ref. (Messina *et al.* 2012) which is aimed at applying the combinatorial method in two subsequent phases. The first phase investigates the whole structure or a portion of it (depending on the application or the exclusion of Filter 1 respectively) and tries to reduce the computational effort by assuming a reduced number of damaging levels (the mentioned N_l=10) whilst the second phase applies again the combinatorial method on the best solution achieved after running the first phase on N_d elements; in the second phase the number of damaging levels is increased to N_l=100 because the combinatorial method shall be applied on N_d elements only, thus allowing a significantly reduced computational effort.

Fig. 8 illustrates the capability of MDLAC based on elements (classical MDLAC) and on zones (MDLAC identifying damaged macro areas). Fig. 9 illustrates the capability to correctly identify the damaged sites through the combinatorial method when all the possible damaged elements are tested.

The damage detection illustrated in Fig. 9 has been achieved by carrying out the computation serially with N_d=2, by activating both Filter 1 (N_z=4) and Filter 2 (*Threshold*=0.5), by using the first ten natural frequency

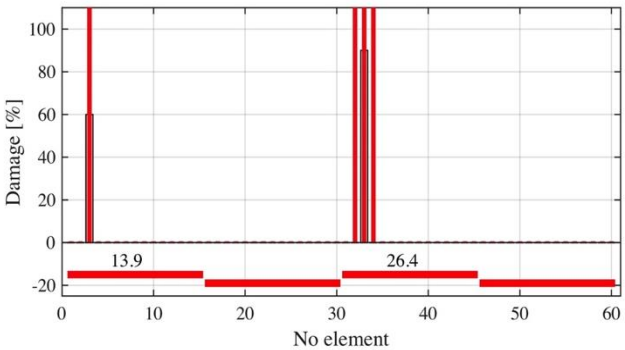


Fig. 8 Location chart through MDLAC for elements and for zones (MESH:60). The absolute % frequency changes (100xΔf_e) correspond to 6.35, 14.7, 3.56, 8.87, 3.74, 5.16, 4.51, 2.52, 5.28, 1.11

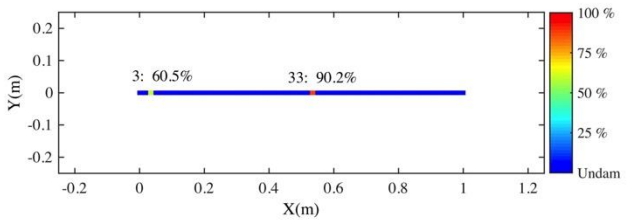


Fig. 9 Location chart through MDLAC and combinatorial method (ref. Figs. 3, 8) (MESH:60). Mode: serial computation

changes and leaving N_l=10 for the first and N_l=100 for the second phase respectively.

The damage detected in Fig. 9 produces a pattern of natural frequency changes which are correlated with the simulated counterpart through a squared cosine (equation (1)) (let us call it a MAC as in (Allemang *et al.* 1982)) equal to MAC_{Iphase}=0.9983 and $MAC_{IIphase}$=0.9999935. Such a Fig. 9 was obtained after running a simulation taking about 26 seconds. An identical picture was achieved when the routines were run by increasing N_d from 2 to 3; in this latter case the location 14 was also slightly detected as damaged with 1% of damage which does not disrupt the excellent capability of the combinatorial method to detect the true damaged scenario.

The damage detection, based on the combinatorial method, was then run by changing the involved parameters in order to appreciate the computational effort involved; in all the simulations Fig. 9 is representative of the capability of the method to detect the damage as above described. What, instead, needs to be appreciated is the computational time required, which, for the combinatorial strategy we are dealing with, is the most representative parameter describing the successful or unsuccessful application of the method; in this regard Tables 3 and 4 recap the performances of the method. Namely Table 3 and Table 4 refer to assuming N_d=2 and N_d=3 respectively. Both Tables 3 and 4 show how the computational time taken is well correlated with the number of analyses required. In particular, profiling the code during runs clarified that the required computational time was mostly due to the routines extracting eigenvalues. Therefore, in general efficient routines extracting eigenvalues would result in great computational savings.

The first four rows in both tables regard the performance of the method when it is run on the basis of classical serial computation. Instead, the last four rows refer to the combinatorial method performed by using parallel computing through four cores. For each core the respective number of analyses is reported in the second column. In particular, in order to partition the analyses to be carried out among the available cores, we use a 1D block-based domain decomposition technique, commonly used in parallel computing, i.e., the N_a analyses to be performed are distributed to the p available cores so that each one is responsible for either $\lfloor N_a/p \rfloor$ or $\lceil N_a/p \rceil$ elements; letting *left* and *right* be respectively the indices of the first and last analysis performed by the core with rank *id* (ranks are numbered from 0 to $p-1$), then the core whose rank is *id* performs all the analyses in the range [*left*, *right*], with *left* $=\lfloor (id-1)N_a/p \rfloor$ and *right*$=\lfloor id\ N_a/p \rfloor -1$. It follows that the core whose rank is *id* performs a total of $e_k=\lfloor (id+1)\ N_a/p \rfloor - \lfloor id\ N_a/p \rfloor$ analyses. The third and fourth columns report using or not Filters 1 (N_z=4) and 2 (*Threshold*=0.5). The fifth column lists the computational time required by the combinatorial method whilst the last two columns reports the required number of analysis (N_a) and the analyses eventually excluded by the Filters 1 and/or 2.

Table 3 Performance table for the combinatorial method (N_z=4, Threshold=0.5) serial and parallel runs, N=10, N_d=2; true damaged scenario: ΔD_3=60%, ΔD_{33}=90%

Mode	Procs: Analyses	Filter 1	Filter 2	Comp. Time [s]	First phase no analyses (N_a)	First phase analyses excluded (%)
Serial	1: 177,000	no	no	180	177,000	0 (0)
Serial	1: 81,916	no	yes	80	81,916	95,084 (54)
Serial	1: 43,500	yes	no	41	43,500	133,500 (75)
Serial	1: 28,311	yes	yes	26	28,311	148,689 (84)
Parallel	1: 44,300 2: 44,200 3: 44,300 4: 44,200	no	no	74	177,000	0 (0)
Parallel	1: 20,605 2: 18,933 3: 21,694 4: 20,684	no	yes	44	81,916	95,084 (54)
Parallel	1: 10,900 2: 10,900 3: 10,800 4: 10,900	yes	no	32	43,500	133,500 (75)
Parallel	1: 7,197 2: 7,169 3: 7,052 4: 6,893	yes	yes	29	28,311	148,689 (84)

In particular, with regard to Table 3 (N_d=2), the combinatorial method requires N_a=177,000 analyses. Such analyses are totally carried out unless Filter 1 and 2 are used; this happens independently of the mode (serial or parallel). For example, when both filters are activated, only 28,311 analyses are carried out whilst the remaining 148,689 (corresponding to 84% of the whole computational work) are excluded. Looking at such a simple example, the importance of designing efficient filters should be appreciated.

With regard to specific comparisons between serial and parallel computations, by looking at Table 3 it would seem that certain circumstances can let the serial computation to be less time consuming than the corresponding parallel computation. Indeed, when both filters are activated the serial computation takes about 26 seconds to complete the analysis whilst its counterpart, based on parallel computing, takes 3 seconds more. In all the remaining cases the parallel computation is faster. This can be explained taking into account that the parallel computation incurs a computational overhead (due to the time required for the activation of the Matlab Parallel Toolbox on the available cores, concurrent access to shared memory etc.) which can make it not convenient when the number of required analyses falls down a machine specific threshold; this is the

Table 4 Performance table for the combinatorial method (N_z=4, Threshold=0.5) serial and parallel runs, N=10, N_d=3; true damaged scenario: ΔD_3=60%, ΔD_{33}=90%

Mode	Procs: Analyses	Filter 1	Filter 2	Comp. Time [s]	First phase no analyses (N_a)	First phase analyses excluded (%)
Serial	1: 34,220,000	no	no	31,731	34,220,000	0 (0)
Serial	1: 12,608,078	no	yes	20,761	21,611,922	12,608,078 (37)
Serial	1: 4,060,000	yes	no	3,789	4,060,000	30,160,000 (88)
Serial	1: 3,449,011	yes	yes	3,502	3,449,011	30,770,989 (90)
Parallel	1: 8,500,000 2: 8,500,000 3: 8,500,000 4: 8,500,000	no	no	10,721	34,220,000	0 (0)
Parallel	1: 5,391,059 2: 5,433,963 3: 5,385,715 4: 5,401,185	no	yes	7,053	21,611,922	12,608,078 (37)
Parallel	1: 1,015,000 2: 1,015,001 3: 1,015,002 4: 1,015,003	yes	no	1,327	4,060,000	30,160,000 (88)
Parallel	1: 863,722 2: 866,468 3: 863,634 4: 855,187	yes	yes	1,100	3,449,011	30,770,989 (90)

case in one of our experiments when both filters are activated and a significantly low number of analyses is taken into account (28,311 with regard to the initial 177,000 cases). Such an interpretation is confirmed by also looking at Table 4 where a particularly high number of cases is taken into account (from about 34 to 3.4 millions); here independently of the activation or not of the filters the parallel approach is always more convenient (less time consuming) than its serial counterpart. However, if the parallel computation shows interesting performance along with the proposed combinatorial method, certainly the designed filters play an important role when used in combination with it. Indeed, Table 4 makes evident that when the classical serial mode is adopted (first row) the combinatorial method takes 31,731 seconds (about 9 hours; moreover, the second phase, which runs serially, requires about 900 seconds); the computational effort reduces to about 1/3 when the parallel computation is used (10,721 s) and when the parallel computation is used along with the activation of both filters the same analysis even runs in about 33 minutes letting the computational time reduce to about 1/18 (1,100 s).

We now discuss the performances obtained by the parallel implementation. Let $T(n,1)$ be the running time of the application performing n analyses when run serially on 1 core and $T(n, p)$ be the running time when run in parallel on p cores. The parallel speedup obtained by the parallel application is defined as $S(n, p)=T(n, 1)/T(n, p)$; the speedup measures how much an application gains when running in parallel on multiple cores. The theoretical maximum speedup achievable is p, obtained when the parallel overhead is equal to zero. Indeed, denoting the parallel overhead by $T_o(n, p)$, since the total time spent by all of the cores is $pT(n, p)$, it follows that $pT(n, p)=T(n, 1)+T_o(n, p)$, so that $T(n, p)=(T(n,1)+T_o(n, p))/p$ and $T_o(n, p)=pT(n, p)-T(n, 1)$. Therefore, we can rewrite the speedup as follows: $S(n, p)=T(n, 1)/T(n, p)=pT(n, 1)/(T(n, 1)+T_o(n, p))$ from which it follows immediately that the $S(n, p)=p$ when $T_o(n, p)=0$. From a theoretical perspective, embarrassingly parallel applications are characterized by null parallel overhead. However, in practice the parallel overhead is never null.

An important metric, which measures the utilization of the available cores is the parallel *efficiency*, which is defined as $E(n, p)=T(n,1)/(p\ T(n, p))=S(n, p)/p$. The theoretical maximum efficiency achievable is 1 (or 100%). Again, $E(n, p)=1$ when $T_o(n, p)=0$. Indeed, it is enough to rewrite the efficiency as $E(n, p)=S(n, p)/p=T(n, 1)/(T(n,1)+T_o(n, p))$. A common rule of thumb used by parallel application designers and developers is to try to achieve a parallel efficiency of at least 0.7 (or 70%). Below this threshold, the application is not making a good use of the parallel resources available. Tables 5 and 6 report the speedup and the efficiency achieved by the parallel implementation on 4 cores, respectively for the simulations related to Tables 3 and 4. It is immediate verifying that, as already stated, the simulations discussed in Table 3 contain too few analyses to benefit from parallelization in terms of speedup and efficiency. In particular, when both Filter 1 and 2 are activated, the resulting number of analyses to be actually carried out is so small that the corresponding speedup is less than 1, *i.e.*, in this case parallel computing is worse than its serial counterpart (as already noted). The parallel performances depicted in Table 6 are much better, owing to the increased number of analyses to be carried out despite the powerful action of the filters. Indeed, in all of the cases speedup and efficiency values are high. In particular, the speedup is around 3 in the first three cases when at most one of the filters is activated, and is 3.18 when both filters are activated (the interested reader should

Table 5 Parallel Performance table for the combinatorial method (N_z=4, *Threshold*=0.5) parallel runs on 4 cores, N=10, N_d=2; true damaged scenario: ΔD_3=60%, ΔD_{33}=90%

Filter 1	Filter 2	Comp. Time [s]	Speedup	Efficiency
no	no	74	2.43	0.60
no	yes	44	1.81	0.45
yes	no	32	1.28	0.32
yes	yes	29	0.89	0.22

Table 6 Parallel Performance table for the combinatorial method (N_z=4, *Threshold*=0.5) parallel runs on 4 cores, N=10, N_d=3; true damaged scenario: ΔD_3=60%, ΔD_{33}=90%

Filter 1	Filter 2	Comp. Time [s]	Speedup	Efficiency
no	no	10,721	2.95	0.73
no	yes	7,053	2.94	0.73
yes	no	1,327	2.85	0.71
yes	yes	1,100	3.18	0.79

take into account that 4 is the maximum theoretical speedup when using 4 cores). The efficiency is greater than 70% in all of the cases, and even reaches 79% when both filters are activated. Finally, it is worth noting here that the performances reported depend heavily on the particular technology employed, which is Matlab and its Parallel Toolbox. Future improvements to the internals of the Parallel Toolbox could lead to increased parallel performances without the need to modify the Matlab source code.

Before closing the discussion on Tables 3 and 4 a final perusal regarding the effectiveness of Filter 1 is certainly worth of further consideration. When N_d=2 (Table 3) is taken into account along with the activation of only Filter 1, 133,500 analyses are excluded by the combinatorial strategy, i.e., only 25% of the initial cases needs to be tested by leaving the remaining 75% out. When N_d=3 (Table 4) is taken into account the percentage of excluded analyses increases from 75% to 88% letting us infer that once a number of zones (N_z) is established, the relative gain increases when the number of potential damaged sites (N_d) increases. This can be generalized through the following proof. Let G be the relative gain defined as in Eq. (10), in which a new term has been introduced (N_e^z) consisting of the total number of elements belonging to the N_z zones; in this regard $N_e > N_e^z$ is an inequality which is always true for obvious reasons.

$$G = \frac{\binom{N_e}{N_d} N_l^{N_d} - \binom{N_e^z}{N_d} N_l^{N_d}}{\binom{N_e}{N_d} N_l^{N_d}} \tag{10}$$

It is clear that the relative gain G has been intended as only involving one of the essential terms of Eq. (4) and specifically the combinations all over the N_e elements (i.e., n_1). Accordingly, the second essential term of Eq. (4) ($N_l^{N_d}$) is simplified. G is clearly less than 1 and corresponds to the percentage value reported in Tables 3 and 4 as % in the respective last columns. Based on the binomial coefficient definition, Eq. (10) can be simplified as in Eq. (11), through which it is possible to prove that G is monotonically increasing in N_d.

$$G=1-\frac{(N_e-N_d)!}{(N_e^z-N_d)!} \cdot \frac{N_e^z!}{N_e!} \tag{11}$$

Indeed, taking into account Eq. (11) and letting $G(N_d+1) > G(N_d)$, Eq. (12) follows

$$1-\frac{(N_e-N_d-1)!}{(N_e^z-N_d-1)!} \cdot \frac{N_e^z!}{N_e!} > 1-\frac{(N_e-N_d)!}{(N_e^z-N_d)!} \cdot \frac{N_e^z!}{N_e!} \Rightarrow \frac{(N_e-N_d-1)!}{(N_e^z-N_d-1)!} < \frac{(N_e-N_d)!}{(N_e^z-N_d)!} \tag{12}$$

which can be transformed into Eq. (13)

$$\frac{(N_e-N_d-1)!}{(N_e^z-N_d-1)!} < \frac{(N_e-N_d)(N_e-N_d-1)!}{(N_e^z-N_d)(N_e^z-N_d-1)!} \tag{13}$$

for finally obtaining Eq. (14) which is always verified because $N_e > N_e^z$.

$$\frac{(N_e-N_d)}{(N_e^z-N_d)}>1 \tag{14}$$

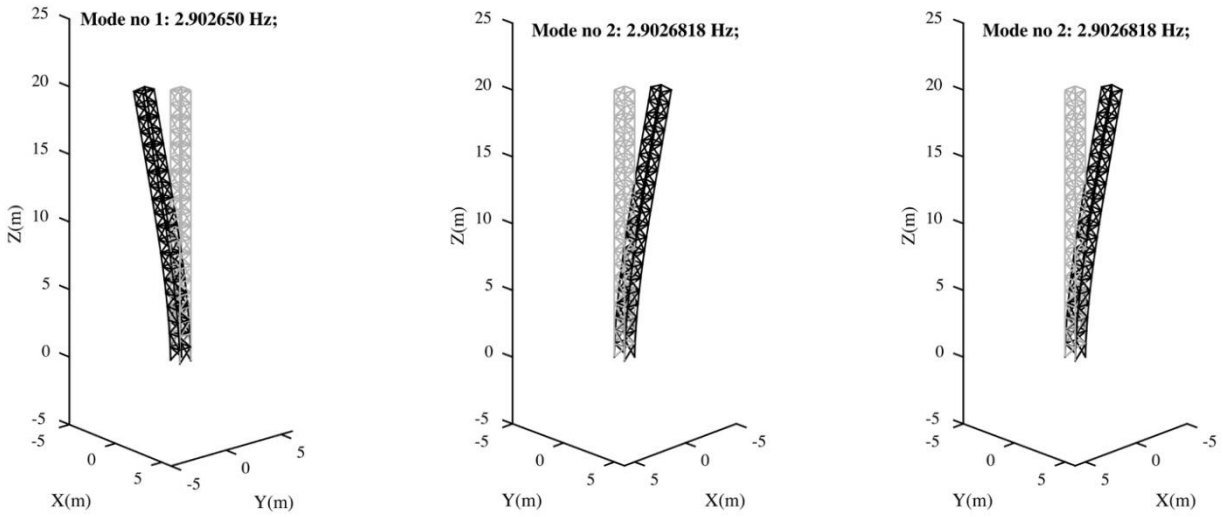
Therefore, as proved by Eqs. (11)-(14), Filter 1, once a number of zones (N_z) has been established, always provides an increasing percent of computational saving when an increasing number of elements potentially damaged (N_d) is chosen.

4.2 The case of a three-dimensional 250-bar truss tower

This case takes into account the three-dimensional vibrating truss tower shown in ref. (Sonnenhof Holdings 2013) among samples and verification. This structure presents symmetries and can make particularly difficult the detection of damaged scenario through natural frequency changes only. This structure is recalled in Fig. 5 along with its first eleven natural frequencies here assessed through a built-in Matlab code (MathWorks 2017) as follows $f^{healthy}$[Hz]: (2.902650, 2.9026818, 7.643402, 13.38692, 13.38783, 22.92009, 29.10572, 29.11273, 38.16575, 45.07039, 45.09624) and from which weak modal symmetries can clearly be expected from the bending modes of the tower as, for example, depicted in Fig. 10 (mode 1, 2, 3 depicted respectively in Fig. 10(a), 10(b) and 10(c)). In the following simulations the first ten natural frequencies have been used in a simulated damaged scenario involving only two elements (i.e., ΔD_1=75%, ΔD_5=50%). Firstly, a damage detection procedure has been carried out through only Filter 1. In this regard the reference point was assumed to be the fixed node of element n. 1 (Fig. 5) whilst the zones containing the remainder elements (due to the fact that N_e/N_z is not usually an integer number) were randomly distributed all over the closer zones.

Fig. 11 depicts two location charts with abscissae reporting the number of elements which are not relevant with their own numeration of Fig. 5; such a number of elements corresponds, instead, to an ordered number of elements grouped in zones and ordered by the distance from the reference point. Fig. 11 still illustrates the capability of Filter 1 to identify correctly the macroscopic zone where the damage is placed; this is achieved independently of the number of zones assumed, N_z=3 and N_z=5 for Fig. 11(a) and 11(b) respectively. It is also interesting to note the improvement provided by Filter 1 to the original MDLAC method (Messina *et al.* 1998) (per elements). Indeed, Filter 1 has not any doubt on the macro area containing damage; it is also interesting to note that the larger the area the lower the damage is, exactly as theoretically predicted by Eqs. (6)-(8). Finally, with regard to Fig. 11, it could be of a certain interest to know that MDLAC runs on 250 elements taking about 110 second whilst it took about 0.2 seconds when run on three or five zones. In practice, MDLAC run on zones was about 550 times faster.

Therefore, the results of this second example corroborate the capability of Filter 1 to aim at improving the performances of MDLAC (Messina *et al.* 1998) when it is carried out per zones and is thus able to insulate reduced areas where the damage can be potentially located. Once a specific zone (shown in Fig. 11) has been detected as potentially containing the damage, the combinatorial



(a) (b) (c)

Fig. 10 First three mode shapes of the three-dimensional truss tower (ref. (Sonnenhof Holdings 2013))

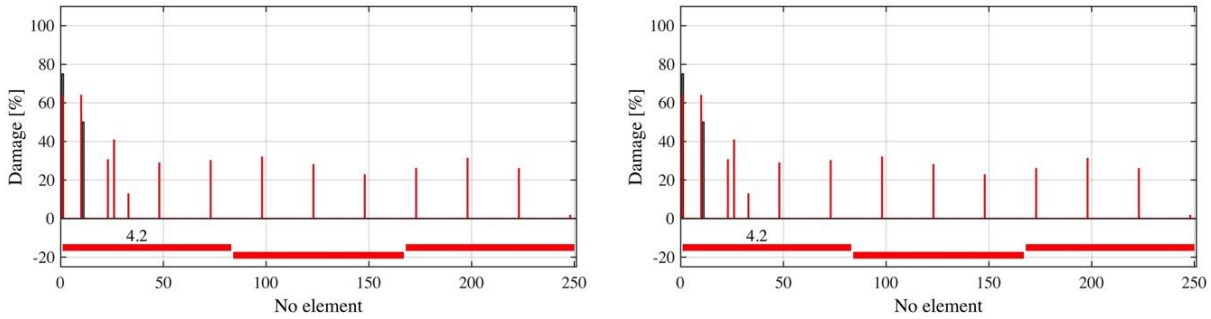


Fig. 11 Location chart through MDLAC for elements and for zones (N_z=3, N_z=5)

method can be carried out. In this latter regard, on the basis of both filters activated (N_z=5, *Threshold*=0.5) and N_d=2 (N_l=10 first phase, N_l=100 second phase, L_{max}=95%), the combinatorial method provided the location chart depicted in Fig. 12 taking about 120 s in the first phase and about 44 s in the second phase. Such an evaluation only involved 26,236 analyses in the first phase, thus excluding 99% of the total analyses, which were purely based on existing combinations (i.e., 3,112,500). The adoption of parallel computing (always based on four cores) allowed reducing the computational time of the first phase from 120 to 57 seconds.

A full and pure combinatorial analysis of the truss tower, based on N_d=3, would require 2,573,000,000 analyses, reducing to 19,600,000 when only Filter 1 is activated. In particular, by setting N_d=3, N_l=10 (first phase), N_l=100 (second phase) along with the activation of both filters 1 and 2 (N_z=5, *Threshold*=0.5) the serial mode evaluation took about 20,172 s (5.6 h) along with 3,947 s (1.1 h) for the second phase to carry out the relevant simulations consisting of 4,807,410 analyses (99.8% of the analyses excluded). For the same experiment, carried out exploiting parallel computing, the first phase, with four cores, took about 2.7 h offering a significant computational saving with regard to the serial computation. For N_d=3 the geometrically symmetric solution (ΔD_3=74.8485%, ΔD_7=48.8990%) was detected (Fig. 5) instead of the true damaged solution (ΔD_1=75%, ΔD_5=50%) as shown in Fig. 12; this result is expected because the detection comes through the second phase and the latter consists of the refinement based on the best result associated to the first phase. The first two highest MAC values kept as solution of the first phase were numerically equivalent (0.9999942596057161, 0.9999942596056380) and contained both the geometrically symmetric solutions (i.e., (ΔD_1, ΔD_5) and (ΔD_3, ΔD_7)), which, therefore must be considered equivalent. Before closing the discussion, a question is worth additional discussion: why the relevant solutions are not detected with a perfect correlation (MAC=1.0) in all the numerical analyses herein carried out? This happened because all levels used (neither N_l=10 in the first phase nor N_l=100 in the second phase) did not contain the true damaged scenario.

5. Conclusions

This manuscript has introduced novel approaches aimed at identifying multiple damaged sites in structural components through finite frequency changes. The approaches essentially regard finite damage along with finite frequency changes. The natural frequencies have been taken into account because the

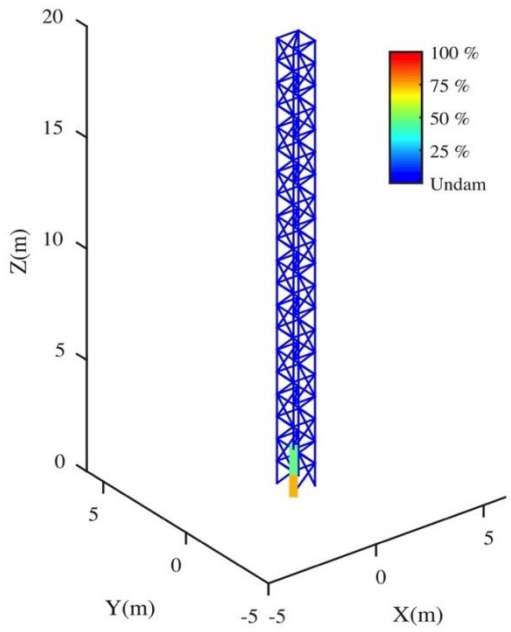


Fig. 12 Location chart through MDLAC and combinatorial method. Detected damage: ΔD_1=74.8485%, ΔD_5=48.8990%

scientific literature still considers nowadays the frequencies as a privileged set of modal data. The methods herein dealt with need a reliable numerical model of the system in order to correlate the natural frequency changes for identifying the damaged locations. This allows dealing with any structure independently of its geometry and/or material characteristics. The approach, based on a combinatorial method, has been carried out through multicore computers implementing parallel computation along with smart strategies aimed at reducing the relevant computational efforts. The parallel computation has clearly shown successful performance independently of the structures under investigation.

However, because the combinatorial method alone can, still today, become cumbersome for several structures and even when a high number of cores are available, the present work has introduced two smart filters (or criterions) aimed at reducing the computational work. The relative gain of computational saving associated to one of the filters has also been theoretically proved. The efficiency of these smart filters has been tested in relevant simulations and one (Filter 1) can even be used alone to single out macro areas within which damage can be potentially insulated; this is achieved by using very few natural frequency changes. The efficiency of such a Filter 1 should be considered highly worthy of mention for two reasons: (*i*) the analyst can focus the attention on a specific area, (*ii*) few elements need to be investigated for damage detection, thus making the combinatorial approach feasible for many structures and even through few measurements (i.e., natural frequencies).

References

Adams, R.D., Cawley, P., Pye, C.J. and Stone, B.J. (1978), "A vibration technique for non-destructively assessing the integrity of structures", *J. Mech. Eng. Sci.*, 20(2), 93-100.

Allemang, R.J. and Brown, D.L. (1982), "A correlation coefficient for modal vector analysis", *Proceedings of the 1st International Modal Analysis Conference*, Orlando, Florida, November.

Chang, P.C., Flatau, A. and Liu, S.C. (2003), "Review paper: Health monitoring of civil infrastructure", *Struct. Hlth. Monit.*, **2**(3), 257-267.

Contursi, T., Messina, A. and Williams, E. (1998), "A multiple-damage location assurance criterion based on natural frequency changes", *J. Vib. Control*, **4**(5), 619-633.

Dimarogonas, A.D. (1996), "Vibration of cracked structures: A state of the art review", *Eng. Fract. Mech.*, **55**(5), 831-857.

Doebling, S.W., Farrar, C.R. and Prime, M.B. (1998), "A summary review of vibration-based damage identification methods, identification methods", *Shock Vib. Digest*, **30**, 91-105.

Fan, W. and Qiao, P. (2011), "Vibration-based damage identification methods: A review and comparative study", *Struct. Hlth. Monit.*, **10**(1), 83-111.

Farrar, C.R., Cornwell, P.J., Doebling, S.W. and Prime, M.B. (2000), "Structural Health Monitoring Studies of the Alamosa Canyon and I-40 Bridges", Los Alamos National Laboratory.

Farrar, C.R., Doebling, S.W. and Nix, D.A. (2001), "Vibration-based structural damage identification", *Phil. Tran. Royal Soc. London A: Math. Phys. Eng. Sci.*, **359**(1778), 131-149.

Friswell, M.I. (2007), "Damage identification using inverse methods", *Phil. Tran. Royal Soc. London A: Math. Phys. Eng. Sci.*, **365**(1851), 393-410.

Hakim, S. and Razak, H.A. (2014), "Modal parameters based structural damage detection using artificial neural networks - a review", *Smart Struct. Syst.*, **14**(2), 159-189.

Horn, R. and Johnson, C. (1985), *Matrix Analysis*, Cambridge University Press.

MathWorks (2017), Matlab© getting started guide.

Messina, A. and Mantriota, G. (2012), "Introspective of damage detection methods based on natural frequencies changes and sensitivities", *Proceedings of the 15th International Conference on Experimental Mechanics, New Trends and Perspectives*. Porto, July.

Messina, A. and Williams, E.J. (1996), "Damage detection and localisation using natural frequency changes", *Proceedings of the 1st International Conference on Identification in Engineering Systems*, Wales, Swansea.

Messina, A., Williams, E. and Contursi, T. (1998), "Structural damage detection by a sensitivity and statistical-based method", *J. Sound Vib.*, **216**(5), 791 - 808.

Montalvao, D., Maia, N. and Ribeiro, A. (2006), "A review of vibration-based structural health monitoring with special emphasis on composite materials", *Shock Vib. Digest*, **38**(4), 295-324.

Rytter, A. (1993), "Vibrational based inspection of civil engineering structures", Ph.D. Dissertation, University of Aalborg, Denmark.

Salawu, O. (1997), "Detection of structural damage through changes in frequency: a review", *Eng. Struct.*, **19**(9), 718-723.

Sinou, J. (2013), "A review of damage detection and health monitoring of mechanical systems from changes in the measurement of linear and non-linear vibrations", Hal archives-ouvertes.fr

Sohn, H., Farrar, C.R., Hemez, F.M., Shunk, D.D., Stinemates, D. W., Nadler B. R. and Czarnecki J. J. (2004), "A review of structural health monitoring literature", 1996-2001, LA (Series) (Los Alamos, N.M.).

Sonnenhof Holdings (2013), Lisa finite element analysis software version 8.0.0 (2013), Tutorials and Reference Guide. http://www.lisafea.com.

PL