POLITECNICO DI TORINO

Master of Science in Civil Engineering

Master's degree Thesis

Calibration of a new metric for computing structural robustness

Supervisor:

Prof. Valerio De Biagi

Candidate:

Davide Barroero

December 2025

Dedicato alla mia famiglia, ai miei amici
e a tutte le persone che mi sono state vicino
durante questo percorso.

Table of Contents

ABSTRACT

Structural robustness has emerged as a fundamental property in the design of structures in modern days, reflecting the need for system-level design approach against unforeseen and accidental actions. In fact, traditional design approaches based on member-level checks, mainly for strength and stability, have proven to be not sufficient in some occurrences, in particular in case of local damages. Despite its recognized importance, structural engineers and the research world around it are still struggling in assessing robustness in a quantitative way. This thesis aims thus to try filling that gap, by calibrating and developing a new quantitative metric based on connectivity for assessing robustness of frame structures, starting from the Resilience Index proposed by Chiaia et Al. (2019). The metric evaluates the response of a frame structure to a local damage, by analysing its kinematic matrixes for every possible damage scenario, defined by the removal of elements and degree of constraints.

The work so begins with a wide theoretical framework. First the evolution of the concept of robustness is analysed, along with its history and definitions, showing also its integration in some of the major international standards (Eurocode, U.S. and Canadian codes). Subsequently, the mathematical formulation of the metric is discussed, together with its original MATLAB implementation. Such program automatically constructs kinematic matrixes from structural data (only topology and constraints setup) for the frame itself and for every analysed damaged condition, computing finally a scalar robustness index which quantifies the system's ability to avoid disproportionate collapse after a local damage.

It follows an accurate calibration phase, in which are defined the constants that govern the metric, ensuring physical meaningfulness and that the contributions of different damage typologies are equally balanced. The metric is then tested against a series of operative properties, each reflecting a precise expectation for a general robustness metric, as to assess its consistency with intuitive structural behaviour.

Finally, future possible developments are highlighted, with special focus on the introduction of a genetic algorithm, to identify efficiently critical scenarios and avoid calculating every possible one, preventing computational burden, which otherwise arises as a main limitation across the whole thesis work. The concluding chapter then summarises obtained results, discussing the potential of the proposed approach as a way how to give a scalar value to such a complex structural aspect.

1 STRUCTURAL ROBUSTNESS

1.1 Definition

Structural robustness represents a very important concept when talking about safety and reliability of modern structures. Despite how's common to discuss about structural robustness as part of research and engineering practice, it can have different meanings depending on the context. Generally, structural robustness is intended to be the ability of a system to keep working in case of an unexpected event occurring, without important loss in terms of performance. In the structural engineering field, that means the capability of a building (or an infrastructure) to resist unforeseen actions – for instance accidental loads, local damages, or even design mistake… - without suffering disproportionate or progressive collapse (Starossek & Haberland, 2011).

The idea of a structure to be able to withstand a certain level of damage and continue working isn't certainly something new, in fact Frangopol and Curley discussed about that in their article of 1987 (Frangopol & Curley, 1987), regarding the concepts of redundancy and reliability. Basically, a robust structure is supposed to be still able to perform its main job, even in the case of failure of one or more of its components (Ellingwood & Dusenberry, 2005). This line of thought radically changes structural design philosophy and focus, since in this way is not sufficient anymore to check every structure's component independently, but engineers should now work with an overall look at the behaviour of structures as a whole (JCSS, 2001). This system-level approach is gaining more and more importance nowadays, because the goal isn't anymore to avoid collapse in standard conditions, but also to guarantee a safe behaviour of the structure in case of unusual situations.

1.1.1 Origins and Historical Background

Structural robustness started gaining interest following some famous collapses, which revealed limits of traditional design rules. A well known example is the Ronan Point apartment building's case in London (Figure 1), where in 1968 a gas explosion in a flat generated the collapse of the entire corner of the building. At the time, the initial damage was relatively small, but due to the lack of redundancy and continuity in the structure, the indirect damage was way larger leading to a spread of the collapse to other parts. This type of chain reaction is nowadays usually identified as "progressive collapse" (Pearson & Delatte, 2005).

Another illustrious example, universally known, is represented by the World Trade Center collapse (Figure 2), in New York, in that famous day of 2001. The damage caused by plane impacts, followed by fire, led to global instability of the structure, resulting in the total collapse of the two towers (Bazant & Zhou, 2002). In the event more than 2800 people lost their lives.

Figure 1 – The Ronan Point apartment building collapse, London 1968

Figure 2 – World Trade Center collapse, New York City 2001

Those examples are significative, since they show how a local damage can grow to the point of generating a total collapse, especially in case of absence or insufficient presence of alternative load paths. In response to events of this type, some design codes started including structural robustness as an explicit requirement to be assessed in the design stage; particularly regarding accidental loads, such as fire, explosions and vehicular impacts. Anyway, despite clear and critical implications in terms of safety and reliability, robustness requirements are still not explicitly defined in many standards around the world.

1.1.2 Formal Definitions and Conceptual Framework

Robustness has been defined in different ways depending on authors and organizations. Eurocode EN 1990 (CEN, 2002) defines it as" the ability of a structure to withstand events like fire, explosion, impact or the consequences of human error, without being damaged to an extent disproportionate to the original cause". This definition focuses on the concept of disproportionate collapse, which is intended to be characterized by an extent of failure not commensurate to the initiating event.

Starossek and Haberland provided a more mechanical definition (Starossek & Haberland, 2011). In their formulation, robustness means the insensibility to local damage, which implies that a robust structure should be characterized by a minimum variation of residual strength after an initial damage.

There's another interpretation which deserves to be reported, the one provided by the Joint Committee for Structural Safety (JCSS, 2001), which links robustness to the concept of redundancy at the system level, stating that a robust structure should remain stable and functional even in case of unforeseen load scenarios. In this respect, robustness is connected not only to strength and stiffness, but also to connectivity and topology.

1.1.3 Robustness vs. Strength and Stability

It's useful to distinguish robustness from other structural performance parameters, such as strength and stability. Strength is the capability of resisting to a load without reach material failure stage, while stability represents the ability to maintain geometric equilibrium under load, especially against buckling. Robustness regards instead what happens to the structure as a whole following a local damage (Starossek & Haberland, 2011). Based on these definitions, the following important statements are derived. A structure can be quite strong and resistant against buckling, but not robust (for instance in case it collapses after the failure of a single element). On the other end, a non-particularly strong structure can be robust, if it has various alternative load paths and it is so able to redistribute forces (Ellingwood & Dusenberry, 2005). For this reason, it's clear how it's important to consider robustness as a design criteria distinct from common strength and serviceability checks.

1.1.4 System-Level Perspective and Structural Typologies

Robustness is intrinsically a system-level parameter, since it can't be guarantee just through single element checks. What matters for the purpose of robustness is how elements are working together in case some of them get damaged (JCSS, 2001). So in this regard, it's possible to notice how continuous frames or redundant grids are usually more robust than pinned or statically determinate systems, having more reserve capacity and alternative loads paths (Frangopol & Curley, 1987). This view shows the relationship between robustness and graph theory and network analysis. Modelling a structure as an interconnected set of arches and nodes, representing respectively structural elements and relative connections, engineers can evaluate the capacity of the structure to redistribute forces in case of removal of some elements (De Biagi, 2014).

1.2 Strategies for Structural Robustness

Having defined robustness as the capacity of a structure to withstand a local damage without generating a disproportionate or progressive collapse, it is now good to focus on how to guarantee this property in the design stage. Difficulties in this regard lie in the fact that robustness isn't a material parameter or a well defined performance index. It is rather the combination of a series of structural features, such as configuration, connectivity, redundancy, ductility and detailing, among others (Starossek & Haberland, 2011). Unlike usual limit state design, which takes into account specific load combinations, robustness-oriented design aims to guarantee structures reliability in unpredictable conditions, such as element failures or sudden disturbances. Since there is therefore no way how to quantify those events, strategies used to confer robustness provide usually a larger reliability in an undirect way, through design choices that improve system capability to tolerating and limiting the damage (Ellingwood & Dusenberry, 2005).

The following section presents thus the most common strategies used in engineering practice and discussed in literature to improve robustness, accompanied by their features and limitations.

1.2.1 REDUNDANCY

To provide redundancy means to provide several independent paths, with which loads can get to external supports. A redundant structure can thus lose one or more members, without determining an unacceptable stress increase in the other elements (Melchers, 1999). Redundancy is considered as one of the most important techniques to get robustness, since it makes the system more adaptable to unforeseen situations. It can be achieved using more elements than what strictly necessary, or generally defining topology and geometry in a way that guarantees the presence of alternative load paths (Frangopol & Curley, 1987). Reliability theory shows clearly how system reliability increases with the number of alternative paths, provided that these paths are sufficiently strong and stiff (JCSS, 2001).

However, to guarantee effectiveness of redundancy, it's necessary a sufficient level of ductility, in order to allow load redistribution following a local damage. Ductility, which is defined as the capability of an element to deform significantly without lose strength, allows to develop plastic hinges and membrane actions to transfer loads. On the contrary, in case of brittle failure, for instance shear failure in columns, those alternative paths can't be activated, negating redundancy benefits (Ellingwood & Dusenberry, 2005).

It's therefore natural that necessity of measure or at least check structures redundancy arises. To do that the Alternate Load Path Method (ALPM) is the most used tool in practice. ALPM consists in the remotion of one or more critical elements (for instance columns, beams or walls) and then the analysis of structure residual capacity. Analysis can be conducted by linear static, non linear static (pushdown), or dynamic simulations (CEN, 2006) (NIST, 2007). ALPM allows to have a measurable approach in redundancy design, allowing for case-specific assessment of load redistribution.

1.2.2 Structural Continuity and Tie Systems

Continuity describes how elements are connected to enable force redistribution across the system. Continuous frames, beams, and slabs tend to be significatively more robust than segmented or pinned systems (Kwasniewski, 2010).

Tie-force systems are a structural engineering approach to create continuity. Many codes (including EN 1991-1-7 and NIST IR 7396) require longitudinal, transverse, and vertical ties to eventually keep the structure connected after local damage, redistributing loads away from it. Ties enable force redistribution directly connecting structural elements. These connections usually work together with redundancy. Codes often specify minimum tying forces to be provided by the structural system to maintain integrity.

Horizontal ties basically have two purposes (BCSA, n.d.), that are to ensure developing of catenary actions and to hold columns in place in case of internal explosion or similar damage, as illustrated in figures below. For what regards vertical ties (BCSA, n.d.), the principal aim is to allow loads to be redistributed through alternative load paths, as presented in the figure below. Additionally, vertical ties can also limit the risk of the upper floor to be blown upwards in case of explosion.

Horizontal and vertical ties functions are well illustrated in Figure 3, Figure 4 and Figure 5.

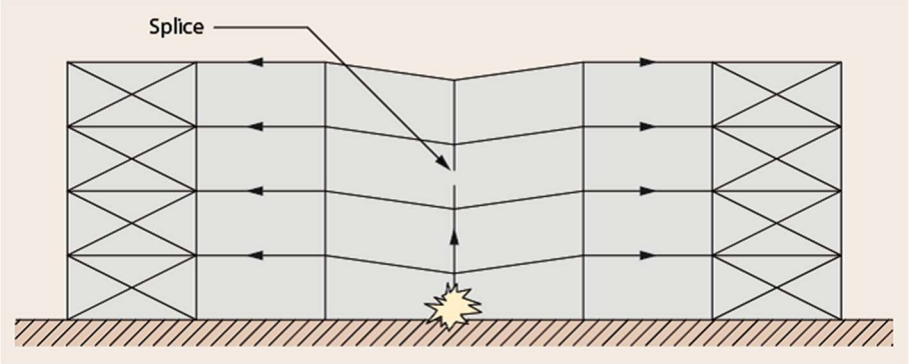


Figure 3 – horizontal ties enabling catenary actions to develop

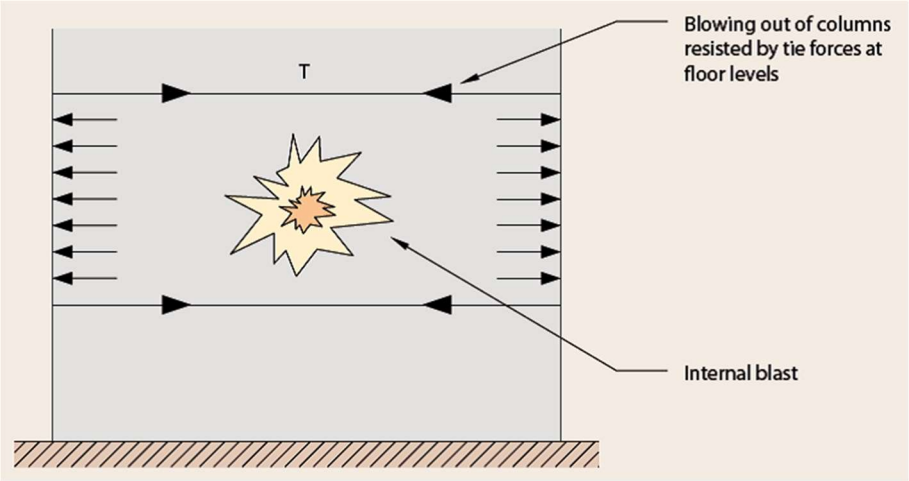


Figure 4 – horizontal ties holding columns in place

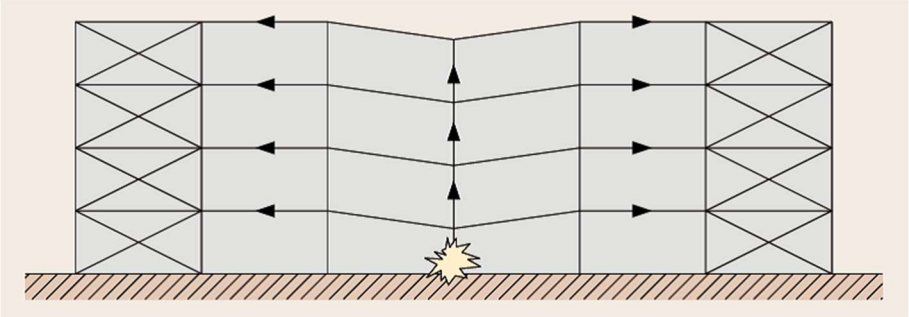


Figure 5 – vertical tying allowing loads to find alternative load paths

1.2.3 COMPARTMENTALISATION

Compartmentalisation consists in dividing a structure into separate units, or "compartments", to prevent damage from spreading. This can be achieved with separation joints or anyway introducing other discontinuities in load paths, limiting the affected area. Compartmentalisation is commonly used in bridges, tunnels, and large open-plan buildings where fire or blast containment is required. While it can reduce the extent of collapse, it may also reduce redundancy, so it must be balanced with the need for continuity to reduce the overall risk of disproportionate collapse.

1.2.4 KEY ELEMENT DESIGN

Some codes allow, or even require, the identification of "key elements" as a strategy to increase robustness. Those elements are components whose failure would lead to disproportionate collapse and they should be designed to resist specific accidental loads, such as blasts, impacts, internal explosions… (CEN, 2006) (Department of Defense - United States of America, 2009). This approach is used in NIST guidelines, where it's assumed that it's not always feasible to make the whole system robust, and so critical elements are designed to be strong enough to prevent initiating failure chains (Ellingwood & Dusenberry, 2005). This method works well especially when redistribution is limited and hazards are known. However, identifying which elements are supposed to be qualified as "key" requires engineering judgment, making the process subjective in some ways.

1.2.5 EVENT CONTROL

While the vast majority of robustness strategies aim to ensure that a structure resist and contain damage, event control strategies are aimed on reducing damage probability first, and so local failure. Rather than increasing structure strength, those strategies limit accidental nature actions exposure, such as vehicle impact, explosion, fire, or human error (De Biagi, 2014). In this way each solution is basically tailored for the specific risk. Common examples include: physical barriers preventing vehicles from striking key elements, blast standoff distances to reduce explosion effects, fire prevention and suppression systems to lower the probability of fire-related structural damage, impact absorption systems in bridge piers and tunnel portals to absorb energy from vehicle collisions, controlled access and security screening to reduce intentional threats, and so on and so forth...

Figure 6 – Anti-collision devices for bridge piers

1.2.6 SUMMARY AND INTEGRATION

Strategies reported above are interdependent and form part of a broader framework. This interdependency can be easily grasped, looking at how redundancy requires ductility and continuity to be effective, compartmentalisation can limit damage but may interfere with alternative load paths, key element design addresses specific threats, while event control reduces the likelihood of such threats occurring. Robustness, therefore, results from the interaction of multiple strategies, guided by a design process that considers potential hazards, failure modes, and acceptable risk levels (JCSS, 2001).

1.3 Structural Robustness Evaluation Metrics

As previously described, robustness is a critical parameter for reliability, but it's usually just a qualitative concept. Robustness meaning is clear and well described by several definitions, but engineers in practice need for quantitative measures to evaluate it, comparing structures, assessing design choices, or verifying compliance with performance objectives. In this respect, a set of measures, often referred to as robustness indices, were proposed. Their goal is to express a structure's post-damage behaviour in a single numerical value (Baker, et al., 2008) (Starossek & Haberland, 2011).

Over the past decades, multiple approaches (metrics) have been proposed. Some of them are based on reliability theory, using probabilistic measures of system and component performance (Fu & Frangopol, 1990). Other metrics are based on risk analysis, and they combine event probability with its consequences (Baker, et al., 2008). There are then other measures focusing on structural performance after damage, referring thus to the ability to contain failure propagation or to dissipate energy under dynamic loading (Starossek & Haberland, 2011).

It is important to note that these indices are not interchangeable:

- A reliability-based index may rate a system as robust because it has many alternative load paths, even if it loses significant functionality for certain damage scenarios;
- A consequence-based index might focus on whether the damage stays localized, regardless of the probability of the initiating event;
- An energy-based index could consider a structure robust if it can absorb large amounts of energy before collapse, which is particularly relevant for impact or blast situations, without caring about damage spreading.

Therefore, there's not a single metric that is able to describe every aspect of structural robustness. Instead, every index report informations about a specific perspective. Usefulness of each metric depends thus on the context, on structure typology and risks that are taken into account.

The following section aims to present five robustness metrics, well defined in the scientific and engineering research field, just as defined in their original sources. For each of them, formulas and definitions of each term are reported exactly as in the original publication.

1.3.1 Probabilistic Redundancy Index - Fu & Frangopol (1990)

Fu and Frangopol have defined the probabilistic redundancy index RI (Fu & Frangopol, 1990) as:

$$RI = \frac{P_f(dmg) - P_f(sys)}{P_f(sys)} \tag{1.1}$$

Where:

- $P_f(sys)$ is the system failure probability, defined as:

$$P_f(sys) = P[any\ g_i \leq 0, i = 1, 2, \dots, m]\ ; \tag{1.2}$$

- $P_f(dmg)$ is the system damage probability (failure of at least one component), defined as:

$$P_f(dmg) = P[any\ y_j \leq 0, j = 1, 2, \dots, n]\ . \tag{1.3}$$

With:

- g_i = performance function associated with system failure mode i ;
- y_j = performance function associated with failure of component j;
- m = number of system failure modes ;
- n = number of system components.

By definition, RI is always non-negative, because $P_f(dmg)$ is an upper bound for $P_f(sys)$. An index value $RI = 0$ corresponds to a non-redundant system, in fact it means that the failure of a single component leads directly to system collapse. Higher RI values indicate greater capacity to sustain local failures without collapsing.

1.3.2 INDEX OF ROBUSTNESS - BAKER, SCHUBERT & FABER (2008)

Baker, Schubert and Faber have defined the index of robustness I_{rob} (Baker, et al., 2008) as:

$$I_{rob} = \frac{R_{dir}}{R_{dir} + R_{ind}} \tag{1.4}$$

Where:

- R_{dir} is the direct risk, representing the expected consequences directly from the initiating damage;
- R_{ind} is the indirect risk, representing the expected consequences from subsequent events triggered by the initial damage.

Risks are calculated as follows:

$$R_{dir} = \int_x \int_y C_{dir}\ f_{D|E_{XB}}(y|x) f_{E_{XB}}(x) dy dx \tag{1.5}$$

$$R_{ind} = \int_x \int_y C_{ind}\ P(F|D = y)\ f_{D|E_{XB}}(y|x)\ f_{E_{XB}}(x) dy dx \tag{1.6}$$

Where:

- C_{dir} = magnitude of direct consequences;
- C_{ind} = magnitude of indirect consequences;
- $f_{E_{XB}}(x)$ = probability density function of exposure variable x

- $f_{D|E_{XB}}(y|x)$ = conditional probability de nsity function of damage state y given exposure x;
- $P(F|D = y)$ = probability of system failure given damage state y.

The index ranges from 0 to 1 (non-robust to robust structure), following that:

- $I_{rob} = 1$ if all system risk comes from direct consequences (no indirect effects);
- $I_{rob} = 0$ if all system risk comes from indirect consequences.

1.3.3 Stiffness-based Measure of Robustness - Starossek & Haberland (2011)

Starossek and Haberland have proposed a simple measure of robustness based on the static system stiffness (Starossek & Haberland, 2011), whose index is defined as follows:

$$R_s = \min_j \frac{\det K_j}{\det K_0} \tag{1.7}$$

Where:

- R_s = stiffness-based robustness index;
- K_0 = active system stiffness matrix of the intact structure;
- K_j = active system stiffness matrix after removing a structural element or connection j;

This measure aims to capture the smallest relative stiffness that results after the removal of any single element or connection. Its simplicity and low computational cost make it appealing for preliminary assessments, even if the authors noted that R_s tends to measure system connectivity more than true robustness, leading sometimes to non representative results.

1.3.4 Damage-based Measure of Robustness - Starossek & Haberland (2011)

A damage-based approach quantifies the progression of damage after an initial failure (Starossek & Haberland, 2011). One formulation is:

$$R_d = 1 - \frac{p}{p_{lim}} \tag{1.8}$$

Where:

- p = maximum total damage from the assumed initial damage i_{lim} ;
- p_{lim} = acceptable total damage according to design objectives.

A more comprehensive version integrates the damage progression over all possible initial damage sizes:

$$R_{d,int} = 1 - 2\int_0^1 [d(i) - i]\, di \tag{1.9}$$

And, when restricted to local failures:

$$R_{d,int,lim} = 1 - \frac{2}{i_{lim}(2 - i_{lim})}\int_0^{i_{lim}} [d(i) - i]\, di \tag{1.10}$$

1.3.5 Energy-based Measure of Robustness - Starossek & Haberland (2011)

Starossek and Haberland also propose an energy-based measure (Starossek & Haberland, 2011):

$$R_e = 1 - \max_j \frac{E_{r,j}}{E_{f,k}} \tag{1.11}$$

Where:

- $E_{r,j}$ = energy released during the failure of element j that is transferred to a subsequent element k;
- $E_{r,j}$ = energy required to cause failure of element k.

Values near 1 indicate very robust systems (low risk of propagation), while negative values mean the released energy is more than enough to cause further failures.

1.4 Design Standards Background

Nowadays, in structural engineering, robustness has become a specific design goal, often recognized by design codes and guidelines, all over the world. Some famous failures in the past, such as the Ronan Point in London (1968) and the World Trade Center in New York (2001), showed that certain collapse mechanism can't be explained just analysing structures looking at single member's level (Ellingwood, 2006). Those collapses showed thus the necessity for robustness specifications in codes, triggering the introduction in main design regulations, shifting so the design process from pure single element checks to consider also the system level performance (JCSS, 2001).

Today's codes incorporate robustness through different approaches, the three main ones are reported below:

- Prescriptive: in which specific detailing rules are fixed to ensure minimum integrity (kind of indirect method);
- Performance-based: where direct verifications of system capacity after damage are assessed;
- Risk-based: for which design targets are set according to acceptable collapse probabilities (Vrouwenvelder, 2002).

The approach depends on regional engineering culture, regulatory philosophy, and the balance between prescriptive clarity and performance flexibility (Gulvanessian, et al., 2002). The following subsections compare how Eurocode, U.S., and Canadian standards include robustness in their frameworks.

1.4.1 EUROCODE

In Europe, robustness is addressed first in EN 1990 (Eurocode - Basis of structural design), which impose that structures should not suffer damage disproportionate to the cause (CEN, 2002). EN 1991-1-7 (Eurocode 1 - Accidental actions) provides guidance for accidental actions, suggesting measures to improve performance under undefined extreme events (CEN, 2006).

A central element of European approach is the Consequence Class (CC) system (Gulvanessian, et al., 2002). This system categorises buildings with respect to the societal consequences of their failure:

- CC1: Low consequence (e.g., agricultural storage buildings);
- CC2a / CC2b: Medium consequence (e.g., standard offices, residential buildings);
- CC3: High consequence (e.g., stadiums, large public venues).

The higher is the class, the more demanding is the robustness measures prescribed. These measures can range from basic continuity detailing to explicit alternate load path analysis (JCSS, 2001).

Moreover, three design routes are defined to reach robustness requirements (CEN, 2006):

- Avoid accidental actions - for instance, by moving critical elements away from risk zones or to place physical protection (bollards protecting pillars…);
- Design key elements to resist specified abnormal loads (specific blast resistance for key members);

- Provide alternative load paths to allow loads to be redistributed after member loss.

Higher-consequence buildings (CC2b, CC3) must incorporate "tie-force rules", with accurate prescriptions for both horizontal and vertical ties. These requirements assume that the structure has sufficient ductility to allow load redistribution, through the development of plastic hinges and/or membrane actions. Without this deformation capacity, the intended tie forces may not develop entirely, reducing the measure benefits (CEN, 2006).

Alternatively, the Eurocode allows for using the direct method, where selected structural elements are notionally removed in an analytical model and the residual stability of the structure is verified. This verification is often carried out using nonlinear static or dynamic analysis, in order to highlight redistribution effects and potential dynamic amplification (Izzudin, et al., 2008). However, Eurocode doesn't prescribe specific quantitative performance thresholds, such as a minimum residual capacity or acceptable displacement limits. In this way it leaves these decisions to engineer's judgement and requirements of the project brief (Ellingwood & Dusenberry, 2005) (Vrouwenvelder, 2002).

1.4.2 U.S. STANDARDS

In United States of America, robustness is discussed in designs in regulations mainly involving performance-based approaches. ASCE 7-22 (Minimum Design Loads and Associated Criteria for Buildings and Other Structures), that represent the national adopted standards for general loading in structure designs, contains integrity provisions as well as risk categories. Risk Categories are correlated to European consequence classes purposes, although they are related mainly to load factors, rather than robustness measures by themselves (Ellingwood, 2006). However, specific robustness verifications in US are generally given only to special occupancies (ASCE, 2022).

A main reference for robustness design in US is NIST IR 7396 (Best Practices for Reducing the Potential for Progressive Collapse in Buildings) (NIST, 2007). It provides to engineers best practices to reduce progressive collapse in buildings under abnormal loadings. Three approaches are specified: an indirect approach, based on continuity, redundancy and improved detailing; a key-design approach, by special local resistance for key members, in order to make them resistant to abnormal loads; finally, the alternate path approach, that's handling checking for load redistribution after members failure. NIST also provides as a Robustness Index, defined as the ratio between post-damage residual capacity and service gravity load. This is a strength-based indicator, intended to quantify somehow the capacity of the structure to remain supporting service loads after suffering local damage.

1.4.3 CANADIAN CODE

In Canada, robustness is considered in the NBCC (NRC, 2020) and CSA standards, usually indirectly. Also in this case, importance Categories (Low, Normal, High, Post-disaster) are assigned to structures, setting design requirements (NRC, 2020). But, unlike Eurocode, there are no prescriptions about tie-force rules or removal simulations. However, it's possible to find some prescriptions related to robustness in specific codes. For instance, CSA S850 (explosions) and CSA S16 (steel structures) recommend: compartmentalisation to limit failure spread; ductility enhancement via detailing and capacity design; redundancy to maintain load paths after damage. This attention for ductility reflects the philosophy, emphasised by several authors and guidelines, that ductility is key to preventing

progressive collapse. NBCC also applies a risk-based philosophy, where allowable collapse probability decreases with higher importance category (NRC, 2020).

1.4.4 Comparative Discussion and Observations

The three frameworks show different balances between prescriptive rules and performance freedom:

- Eurocode offers a clear classification and mandatory detailing for higher-consequence structures.
- U.S. guidelines prioritise flexibility and advanced analysis methods, but rely heavily on engineering judgment.
- Canada focuses on redundancy and ductility without formal enforcement of specific robustness checks.

2 THEORETICAL BACKGROUNDS

The following chapter aims to provide theoretical bases for mathematical and engineering tools that are widely utilized and developed later in the thesis. In particular, a theoretical background will be provided about graph theory and about the original formulation of the connectivity-based robustness index, theorized by Chiaia et Al. That index will be further developed in next chapters, trying to find some possible extensions to make it more significant and accurate in describing structure robustness. Basic concepts of graph theory are reported because structures' kinematic description, that will be used later on, is based on it.

2.1 Graph Theory and graph models in structural engineering

As discussed so far, traditional design is focused on member checks about strength, stability, deformability and so on, but robustness needs for a system-level analysis. In fact, it's necessary to check for how forces can be redistributed after a local failure, in order to avoid configurations that lack in alternative load paths. It's possible to do that by breaking away from classic structural engineering paradigms, leaving strength and stiffness on a side to focus on how elements are connected through the structure. To this purpose, graph theory can be used to describe structural topology and configuration, providing a more abstract description of structures, which capture connectivity and redundancy, rather than material behaviour.

Graph theory, representing a branch of mathematics, studies the relations between objects through networks of nodes and edges. It can find applications in a big variety of fields, such as computer science, medicine, biology, social science, economics, physics and much more. When applied to structural engineering, it can be used to describe a structure considering members, such as columns and beams, as edges and relative connections as nodes. In this way a building, a bridge or whatever type of structure can be represented as a graph, and its robustness studied looking at the properties of that graph. In this way, it's possible to see analytically, with a simple tool, what happens to the system if one element is removed, to check if there are enough alternative paths for loads and still which elements are critical for connectivity.

2.1.1 BASIC CONCEPTS IN GRAPH THEORY

A graph is an ordered pair, defined as follows:

$$G = (V, E) \tag{2.1}$$

where:

- $V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes (or vertices);
- $E = \{e_1, e_2, \dots, e_m\} \subseteq V \times V$ is the set of edges (or links).

The elements that make up the set E are 2-element subset of V. It means that an edge is described by the two vertices on its sides, e.g. $e_m = (v_i, v_j)$, those vertices are called "ends" of the edge. An edge $\{x, y\}$ can be written as xy.

Two vertices x, y belonging to G, are defined adjacents if $xy \in E(G)$.

The number of nodes is called "order of the graph", indicated with the letter "n", $n = |V|$.

The number of edges is called "size of the graph", indicated with the letter "m", $m = |E|$.

The degree of a node v_i, that can be written as $d(v_i)$, represents the number of edges incident to it.

Considering a structural system, the order represents thus how many connections are in it, the size represents how many members, while finally the degree of a node represents the number of members converging in the related connection.

Edges can be characterized by a direction, in this case they're called arrows, and the resulting graph is defined as "directed graph". Eventually, it is a pair $G = (V, A)$ of vertices V and ordered pairs of vertices A.

A path is a sequence of distinct nodes $(v_1, v_2, \dots, v_k)$, with an edge connecting every couple of consecutive nodes. In case the first and the last node coincide, the path forms what is called cycle.

A subgraph of a graph G, is itself a graph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$. It can be used in a structural application to represent a portion of the structure.

A graph is defined as "connected" if there is at least one path between any couple of nodes. Otherwise, it is disconnected, composed of different connected components. A tree is a connected graph (or subgraph) without cycles. It follows that a tree with n vertices has exactly $n - 1$ edges. A disjoint collection of trees is called a "forest", so each connected components of a forest is itself a tree.

Finally, a last concept about graph theory that deserve to be reported is the "cut set". It's a set of nodes or edges whose removal disconnect the graph. Imagining describing a structure through a graph, identifying the cut set means to find those critical members or joints whose failure could cause disproportionate collapse.

2.1.2 Algebraic Representations of Graphs

Beyond visual diagrams, graph can be represented through matrixes. This algebraic representation turns very useful, allowing for an analytical description of the graph. In this way engineers can study properties such as connectivity and robustness of graphs using linear algebra.

The most important graph matrixes are reported below.

- Adjacency matrix

For a graph with n nodes, the adjacency matrix A is an $n \times$ n matrix defined as:

$$a_{ij} = \begin{cases} 1 \; if \; there \; is \; an \; edge \; between \; v_i \; and \; v_j \\ 0 \; otherwise \end{cases} \tag{2.2}$$

In undirected graphs, the adjacency matrix A is symmetric. In case of directed graph, the $ij - entry$ of the matrix $A(G)$ is equal to 1 if there's an arrow from i to j, and equal to 0 otherwise. In this sense, the adjacency matrix of the non directed graph can be seen as a special case of the directed ones, in which an arrow in both directions is considered to be present for every edge. In some cases, it's possible to consider how many links are there between two nodes, allowing for entries different from 0 and 1.

- Degree matrix

For a graph with n nodes, the degree matrix D is a diagonal $n \times$ n matrix defined as:

$$D = diag\big(d(v_1), d(v_2), \dots, d(v_n)\big) \tag{2.3}$$

So, each entry corresponds to the degree of the respective node. This matrix can be useful to evaluate the redundancy of a graph.

- Incidence matrix

For a directed graph with n nodes and m edges, the Incidence matrix B is a $n \times m$ matrix, where entries b_{ij} are defined as:

$$b_{ij} = \begin{cases} +1 \; if \; node \; v_i \; is \; the \; initial \; endpoint \; of \; edge \; e_j \\ -1 \; if \; node \; v_i \; is \; the \; terminal \; endpoint \; of \; edge \; e_j \\ 0 \; otherwise \end{cases} \tag{2.4}$$

Rows and columns are indexed respectively by the vertices and edges of the graph.

In case of undirected graph, the orientation is arbitrary and the Incidence matrix is just a 01-matrix.

The incidence matrix is central in structural mechanics, since it's used for equilibrium and kinematic equations.

- Laplacian matrix

One of the most useful matrices in graph theory is the graph Laplacian matrix L. For a directed graph, it is defined as:

$$L = D - A = BB^T \tag{2.5}$$

Being a square symmetric matrix, the Laplacian matrix L has real eigenvalues. Those eigenvalues are then all nonnegative, since L is positive semidefinite. By studying its eigenvalues, we can get important features of the graph, as reported below.

The number of zero eigenvalues of L equals the number of connected components in the graph.

The smallest non zero eigenvalue, also called the algebraic connectivity, gives a measure of how well connected the graph is.

2.2 The connectivity-based resilience index (Chiaia et al., 2019)

This section reports the original formulation by Chiaia, Barchiesi, De Biagi and Placidi of the resilience index proposed in their article published in 2019 "A novel structural resilience index: definition and application to frame structure" (Chiaia, et al., 2019). As anticipated, in later chapters, this thesis will extend the approach in such a way that stiffness will also be taken into account, represented through the mass parameters. Here the original formulation is reported, following the paper's notation exactly.

This approach is based on the connectivity of the structures, which is described through kinematics relations. The idea is to analyse structure kinematic matrices before and after a damaging event, looking for kinematic mechanisms that take place due to the event. It is formulated for in plane analysis. To quantify the robustness, the index considers the amount of initial and indirect damage, through a parametrization that is exposed just below. The authors, and this chapter too, divide the approach in three stages, helping the comprehension of the method.

2.2.1 PRE-EVENT PHASE

Consider a structure with N bodies, that leads to $n = 3N$ degrees of freedom (considering in-plane kinematics). The vector q contains its Lagrangian parameters. Consider than m degrees of constraint, assuming no prescribed displacements. Assuming the matrix J as the kinematic matrix of the mechanical system, it follows that:

$$J\,q = 0, \qquad J \in M(m \times n) \tag{2.6}$$

Where M is the set of $m \times n$ matrices with real entries.

In the pre-event phase structures are assumed to be statically indeterminate (as they usually are), that in mathematical terms can be expressed as: $m \geq n = rank(J)$. Hence, the structure is statically indeterminate of order $m - n$.

2.2.2 DURING-EVENT PHASE (ABRUPT DAMAGE)

At this point, it occurs to define the abrupt damage and to quantify it. The damage is considered to consist in the removal of h bodies and k degrees of constraint.

The removal of some bodies involves the removal of some degrees of constraint that were connected to those bodies. For instance, removing those h bodies imply the removal of $\tilde{k}$ degrees of constraint too. Hence, those k degrees of constraint considered consist just in those removed constraint that were not in touch with any of the removed bodies. It is the case of the formation of a plastic hinge, for example.

To define the so called "Deteriorating Agent", that means to quantify the damage, a weight is given to the bodies introducing the parameter m_h, which represents the mass fraction of those h bodies that has been removed. While constraints are assumed to be massless. Anyway, for what concern constraints, a distinction is done between rotational and translational ones. This because it's way less common to have a failure of a displacement type constraint. Removed constraints are so counted as follows:

$$k = ck_R + dk_T \tag{2.7}$$

The authors developed a calibration of those parameters, which led to the values of $c = 1$ and $d = 3$ (the entire process is not reported, but it's available in the paper) (Chiaia, et al., 2019).

The deteriorating agent is then defined as

$$D = am_h + bk \tag{2.8}$$

Again, the authors developed the calibration of parameters a and b, defined them as: $a = 1$ and $b = {}^1/_5$. As before, the calibration isn't reported here, but available in the paper.

2.2.3 POST-EVENT PHASE

After the abrupt damage the structure has $\overline{N} = N - h$ bodies and $\bar{n} = 3\overline{N}$ degrees of freedom.

Just like before, the vector $\bar{q}$ contains the Lagrangian parameters of the $\overline{N}$ bodies.

The remaining degrees of constraint are defined as: $\bar{m} = m - \tilde{k} - k_R - k_T$.

The kinematic matrix of the structure in this phase is derived by the original one, just removing $\tilde{k} + k_R + k_T$ rows and $3h$ columns. In algebraic terms:

$$\bar{J} \in M(\bar{m} \times \bar{n}) \tag{2.9}$$

In this stage, it's no more guaranteed that $\bar{m} \geq \bar{n} = rank(\bar{J})$, since it's possible to have a partial or total collapse of the structure at this point.

Evaluating the kernel of the kinematic matrix $\bar{J}$, it's possible to enumerate those Lagrangian parameters (in addition to the null trivial solution) that fill up the following equation:

$$\bar{J}\,\bar{q} = 0 \tag{2.10}$$

Finally, it's possible to evaluate the mass fraction of the collapsing bodies (considering the bodies belonging to the structure after the damage), identified as m_a. It follows that $m_h + m_a \leq 1$

2.2.4 DEFINITION OF THE STRUCTURAL RESILIENCE INDEX

For each admissible damage event i, the deteriorating agent D^i and the mass fraction ${m_a}^i$ are evaluated.

Thus, the non minimized resilience R^i is defined for each damage event as follows:

$$R^i = \frac{D^i}{a\,{m_a}^i} = \frac{a\,{m_h}^i + b(c\,{k_R}^i + d\,{k_T}^i)}{a\,{m_a}^i} \tag{2.11}$$

The structural resilience index R is defined as the minimum value assumed of the non minimized resilience, considering all the possible k damaging events. The structural resilience index is referred thus to the worst-case scenario.

$$R = \min(R^1, R^2, \dots, R^k) = \min\left(\frac{D^1}{a\ {m_a}^1}, \frac{D^2}{a\ {m_a}^2}, \dots, \frac{D^k}{a\ {m_a}^k}\right) \tag{2.12}$$

The index has a lower bound equal to 0, due to its definition, and an upper bound equal to 1, that comes from the limit case in which the removal of all bodies is considered. It is so defined in the range [0, 1].

3 MATLAB IMPLEMENTATION OF THE ROBUSTNESS INDEX

This chapter of the thesis aims to present and analyse the original MATLAB implementation of the Connectivity Based Resilience Index. The original MATLAB code was developed by Prof. Valerio De Biagi, advisor of this thesis, and it's reported in its original version at 3.1. This code is included, despite it's not authored by the candidate, since it constitutes the computational framework utilized in numerical analyses and from which all the developments present in this thesis work come from.

This code is concepted to be a proof-of-concept tool, translating the theoretical framework previously discussed into a computational environment. It is therefore not a version ready to be used in real context, due to computational limits that will be exposed later on and for the need to test accuracy and validity of results. This last task will also be developed in the next chapters. Anyway, what the code is able to do is:

- To read structure description by Excel files, which schematize structures through graph theory, providing information on topology and on how elements are connected to each other.
- To assembly the kinematic matrix, which encodes compatibility relationships within structural elements. That matrix is then used to identify the eventual presence of kinematic mechanisms resulting from the damage. Elements involved in the kinematic mechanism will be considered as collapsed.
- To consider every possible damage scenario and iterate on them. In this initial version only constraint removals are considered, in all of its possible combination. In the next developments, it's intended to include element removals.
- To compute the robustness index, condensing thus just information related to topology of the structure into a scalar value representing robustness. This is done through the iteration discussed in the previous point, identifying the critical scenario as the one giving the lowest value of the index.

The workflow of this implementation is summarized in Figure 7.

This "version 0" of the code is conceptually valid and solid, but its intentional exhaustiveness makes it computationally expensive. In fact, systematically evaluating all the possible constraints removal combinations, it rapidly reaches combinatorial explosion already for modest structural schemes. This limits its applicability to small benchmark structures. In any case, at present, the role of this code in the current research is to constitute a starting platform, useful to test ideas, validate concepts and eventually identify strengths and weaknesses of this approach.

In the continuation of the thesis, starting from this code, the goal will be to work on the following development fronts:

- Extension of damage combination to elements removal, making simulations more general and realistic;
- Mitigation of the excessive computational demand;

- Calibration of constants a, b, c, d, investigating how to set the weighting factors present in the damage magnitude calculation, eventually calibrating them depending on the structure, in order to ensure consistency with robustness principles;
- Verification of consistency with fundamental properties, assessing whether the method satisfies basic requirements of a robustness metric.

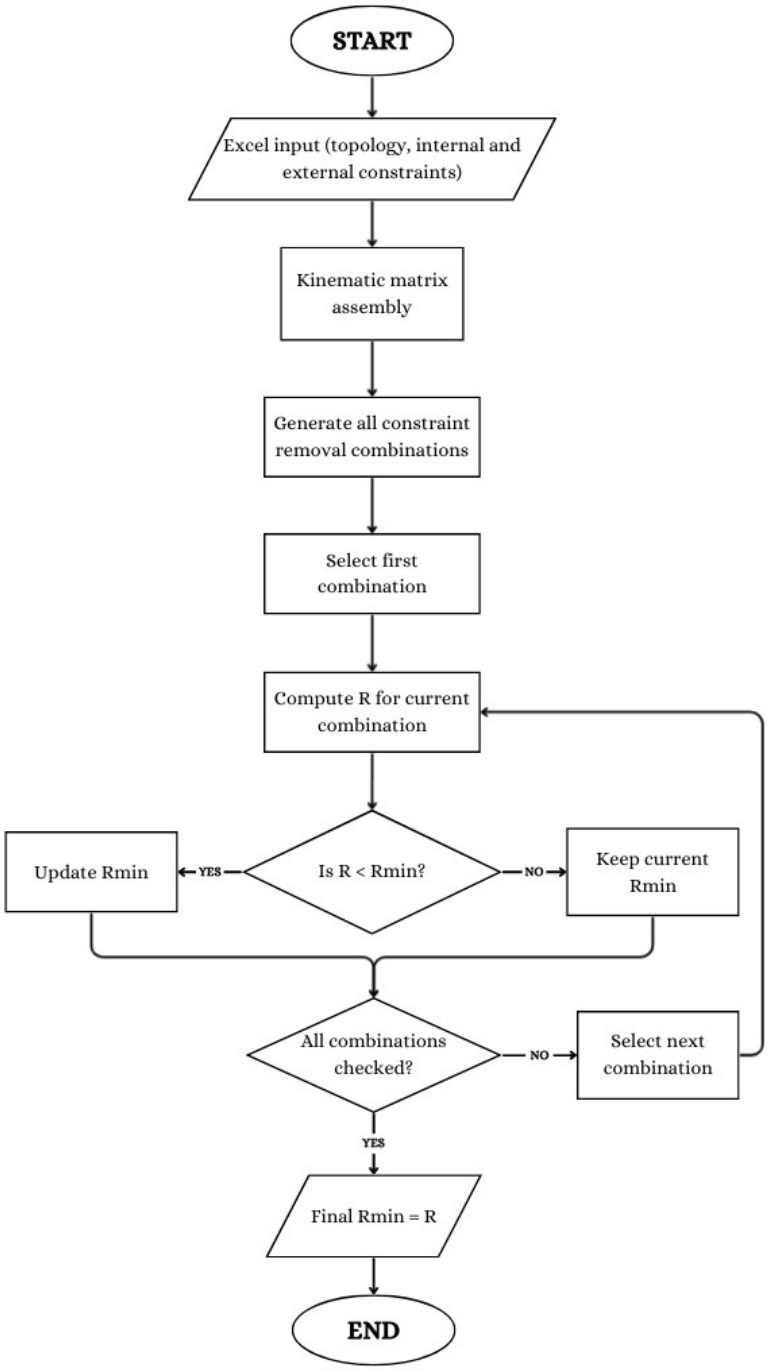


Figure 7 - Flowchart of the original MATLAB implementation

3.1 Original MATLAB Listing

```
clear
clc

constants.a=1;
constants.b=0.2;
constants.c=1;
constants.d=3;

FileName='Str 02';

[kinematic matrix,nodesId,massMatrix]=kinematic matrixFunction([FileName
'.xls']);

noConstr=size(kinematic matrix,1);
ConstraintTypes=reshape(nodesId,2,length(nodesId)/2)';
kinematic matrix(abs(kinematic matrix)<1e-7)=0;

RStore=Inf;
k=1;
for i=1:size(kinematic matrix,1)
  Cases=nchoosek(1:noConstr,i);
  for j=1:size(Cases,1)
    removedConstraint=Cases(j,:);

[Ri(k)]=ResilienceIndex(kinematic matrix,massMatrix,ConstraintTypes,constants,re
movedConstraint);
    % It updates progressively the minimum value of R, tracking the most severe
    % configuration
    if Ri(k)<RStore
      minConstraint=removedConstraint;
    end
    [RStore]=min(Ri(k),RStore);
    k=k+1;
  end
end


function [Ri]=ResilienceIndex(kinematic matrix,massMatrix,ConstraintTypes,
constants,removedConstraint)

a=constants.a;
b=constants.b;
c=constants.c;
d=constants.d;


removedElement=[];
removedConstraintVector=1:size(kinematic matrix,1);
removedConstraintVector(removedConstraint)=[];
% removedConstraintVector contains indexes of rows to be taken from kinematic
% matrix
% modKinMatrix is the kinematic matrix without removed constraints
modKinMatrix=kinematic matrix(removedConstraintVector,:);

removedElementsVector=1:size(kinematic_matrix,2);
```

```
removedElementsVector([(removedElement-1)*3+1 (removedElement-1)*3+2
(removedElement-1)*3+3])=[];

modKinMatrix=modKinMatrix(:,removedElementsVector);

NU=null(modKinMatrix);

if size(NU,2)>1
  NUmod=max(abs(NU'))';
else
  NUmod=NU;
end

NUmod=reshape(NUmod,3,length(NUmod)/3);
A=max(abs(NUmod));
M=massMatrix(A>.001,2);

mh=sum(massMatrix(removedElement,2))/sum(massMatrix(:,2));

D=a*mh+b*(c*sum(ConstraintTypes(removedConstraint,2)==0)+d*sum(ConstraintTypes(r
emovedConstraint,2)==1));

ma=sum(M)/sum(massMatrix(:,2));

Ri=D/(a*ma);

end


function [kinematic matrix,nodesId,massMatrix]=kinematic matrixFunction
(nomefile)

[n,b,top,constrain,internal]=File2Str KIN(nomefile);

kinematic matrix=zeros(1,b*3);
nodesId=[];
connection matrix=zeros(n,b);
for i=1:b
  connection matrix(top(i,2),i)=-1;
  connection matrix(top(i,3),i)=1;
end

node degree=sum(abs(connection matrix),2);

%internal constraints projected
internal projected=internal;
for i=1:b
  if abs(top(i,4))==pi/2
    internal projected(i,3)=internal(i,4);
    internal projected(i,4)=internal(i,3);
    internal projected(i,6)=internal(i,7);
    internal projected(i,7)=internal(i,6);
  end
end

KK=1;

for i=1:n
```

```
if node degree(i)==1
  % nodi con grado 1
  beam=find(or(connection matrix(i,:)==-1, connection matrix(i,:)==1));
  if connection matrix(i,beam)==-1
    target int contraints=internal projected(beam,2:4);
  elseif connection matrix(i,beam)==1
    target int contraints=internal projected(beam,5:7);
  end
  target node=constrain(i,2:4);

  for k=1:3
    if and(target int contraints(k)==1,target node(k)==1)
      switch k
        case 1
          kinematic matrix(KK,(beam-1)*3+1)=1;
          nodesId=[nodesId KK 0];
        case 2
          if connection matrix(i,beam)==-1
            kinematic matrix(KK,(beam-1)*3+1)=+top(beam,5)/2*sin(top(beam,4));
            kinematic matrix(KK,(beam-1)*3+2)=1;
          elseif connection matrix(i,beam)==+1
            kinematic matrix(KK,(beam-1)*3+1)=-top(beam,5)/2*sin(top(beam,4));
            kinematic matrix(KK,(beam-1)*3+2)=1;
          end
          nodesId=[nodesId KK 1];
        case 3
          if connection matrix(i,beam)==-1
            kinematic matrix(KK,(beam-1)*3+1)=-top(beam,5)/2*cos(top(beam,4));
            kinematic matrix(KK,(beam-1)*3+3)=1;
          elseif connection matrix(i,beam)==+1
            kinematic matrix(KK,(beam-1)*3+1)=+top(beam,5)/2*cos(top(beam,4));
            kinematic matrix(KK,(beam-1)*3+3)=1;
          end
          nodesId=[nodesId KK 1];

      end
      KK=KK+1;
    end
  end

else
  % nodi con grado >1
  beam=find(or(connection matrix(i,:)==-1, connection matrix(i,:)==1));
  beam cases=combnk(beam,2);

  for c=1:size(beam cases,1)
    beamA=beam cases(c,1);
    beamB=beam cases(c,2);

    if connection matrix(i,beamA)==-1
      int contraints A=internal projected(beamA,2:4);
    elseif connection matrix(i,beamA)==+1
      int contraints A=internal projected(beamA,5:7);
    end

    if connection matrix(i,beamB)==-1
      int contraints B=internal projected(beamB,2:4);
    elseif connection_matrix(i,beamB)==+1
```

```
        int contraints B=internal projected(beamB,5:7);
      end

      for k=1:3
        if and(int contraints A(k)==1,int contraints B(k)==1)
          switch k
            case 1
              kinematic matrix(KK,(beamA-1)*3+1)=1;
              kinematic matrix(KK,(beamB-1)*3+1)=-1;
              nodesId=[nodesId KK 0];
            case 2
              if connection matrix(i,beamA)==-1
                kinematic matrix(KK,(beamA-
1)*3+1)=+top(beamA,5)/2*sin(top(beamA,4));
                kinematic matrix(KK,(beamA-1)*3+2)=1;
              elseif connection matrix(i,beamA)==+1
                kinematic matrix(KK,(beamA-1)*3+1)=-
top(beamA,5)/2*sin(top(beamA,4));
                kinematic matrix(KK,(beamA-1)*3+2)=1;
              end

              if connection matrix(i,beamB)==-1
                kinematic matrix(KK,(beamB-1)*3+1)=-
top(beamB,5)/2*sin(top(beamB,4));
                kinematic matrix(KK,(beamB-1)*3+2)=-1;
              elseif connection matrix(i,beamB)==+1
                kinematic matrix(KK,(beamB-
1)*3+1)=+top(beamB,5)/2*sin(top(beamB,4));
                kinematic matrix(KK,(beamB-1)*3+2)=-1;
              end
              nodesId=[nodesId KK 1];

            case 3
              if connection matrix(i,beamA)==-1
                kinematic matrix(KK,(beamA-1)*3+1)=-
top(beamA,5)/2*cos(top(beamA,4));
                kinematic matrix(KK,(beamA-1)*3+3)=1;
              elseif connection matrix(i,beamA)==+1
                kinematic matrix(KK,(beamA-
1)*3+1)=+top(beamA,5)/2*cos(top(beamA,4));
                kinematic matrix(KK,(beamA-1)*3+3)=1;
              end

              if connection matrix(i,beamB)==-1
                kinematic matrix(KK,(beamB-
1)*3+1)=+top(beamB,5)/2*cos(top(beamB,4));
                kinematic matrix(KK,(beamB-1)*3+3)=-1;
              elseif connection matrix(i,beamB)==+1
                kinematic matrix(KK,(beamB-1)*3+1)=-
top(beamB,5)/2*cos(top(beamB,4));
                kinematic matrix(KK,(beamB-1)*3+3)=-1;
              end
              nodesId=[nodesId KK 1];
          end
          KK=KK+1;
        end
      end
    end
```

```
  end
end

massMatrix=top(:,[1 end]);

end


function [n,b,top,constrain,internal]=File2Str KIN(StrFile)
%
%-GENERAL PROPERTIES OF THE FRAME----------------------------------------
%
%   n:          number of nodes                                  (1 x 1)
%   b:          number of beams                                  (1 x 1)
%
%-TOPOLOGICAL AND MECHANICAL PROPERTIES OF THE FRAME----------------------
%
%   top:        matrix of frame topology                         (b x 4)
%       [Beam no    Init node(i)    Final node(j)  Angle]
%       Note: the angle is anti-clockwise positive defined and represents
%             the angle between the horizontal axis and beam axis at the
%             initial node.
%
%   constrain:  matrix of the external constrains                (n x 4)
%       [Node no    Rotation    H-trans     V-trans]
%              0 for unconstrained displacement
%              1 for constrained displacement
%
%   beam:       matrix of beam properties                        (b x 5)
%       [Beam no    Length  X-sec area   X-sec inertia   Young's modulus]
%       Note: the lenght is expressed in m, the cross section area is
%             expressed in m^2, the inertia in m^4 and Young's modulus is
%             expressed in N/m^2.
%
%   internal:   matrix of internal constrains                    (b x 7)
%       [Beam no     rot i   v i     w i     rot j   v j     w j]
%              0 for unconstrained displacement
%              1 for constrained displacement
%
internal = xlsread(StrFile,'int constr');
constrain = xlsread(StrFile,'ext constr');
top = xlsread(StrFile,'topology');

n = size(constrain,1); %No of nodes
b = size(top,1); %No of beams
end
```

(Note: in the following sections references to specific blocks will be done by function name rather than repeating the listing.)

3.2 Kinematic Matrix

3.2.1 STRUCTURE DESCRIPTION

The structure is described by three Excel sheets read by the function *File2Str_KIN*. Those Excel sheets are illustrated below:

- Topology (b×6):

BeamNumber	InitialNode	FinalNode	Angle	Length	Mass

For the first three columns, just beams' and nodes' IDs are reported, as numbers. Angles are measured counter-clockwise from the global x axis to the member axis at the initial node. Lengths are expressed in meters. The mass is originally a scalar value assigned by the designer, to ideally account for members importance when computing damage and collapse magnitudes. A possible development could include the expression of the mass as a function of stiffness.

- External constraints (n×4):

NodeID	Rotation	H-translation	V-translation

This table is filled by 0/1 entries, representing respectively free or fixed displacement, in this way it's possible to mark which nodal DOFs are externally restrained.

- Internal constraint (b×7):

Beam Number	Initial node			Final node		
	Rotation	Axial	Lateral	Rotation	Axial	Lateral

This table is also filled by 0/1 entries, representing respectively free or fixed end release, in this way it's possible to describe internal constraints configuration and so how members are connected to each other's. Expressing these relations in terms of axial and lateral translations, means to work in a local reference system. This point will be considered later.

The script, taking the information just reported, constructs then a node-member incidence matrix, which is named *connection_matrix* (size n×b, entries ±1 at node-member connections, distinguishing by the sign initial and final node), and the node degree vector, as the row-sum of absolute values of the incidence matrix. Here there's a clear reference to Graph Theory: structural nodes are graph vertices, beams are graph edges and node degrees are the representation of vertex valence. In next steps, the kinematic equations that will compose the kinematic matrix depend on whether a node has degree equal to one or more, representing respectively what is called "leaf" in graph theory or internal joint. Finally, a small but important preprocessing point is the swap for members with angle equal to $\pm\pi/2$ (*internal_projected* in the listing), that consist in interchanging directions so that the degree of freedom labelling remain consistent regardless of the orientation.

3.2.2 MATRIX CALCULATION

After got the preprocessing stages just showed, the function named *kinematic_matrixFunction* builds the global kinematic matrix row by row. Each row encodes a kinematic equation, that can be referred to an internal or external constraint. In case of internal joint with more than two members converging in it, an equation is obtained for each couple of connected elements. Columns come in triplets per member, that means a total of *3b* columns overall, ordered as follows (for the member "m"):

- column 3m−2: end rotation-like term,
- column 3m−1: end translation along the local axis parallel to the member,
- column 3m: end translation along the local axis orthogonal to the member,

with appropriate geometric coupling through sin(angle)/cos(angle) and lever arms (±L/2) to map nodal DOFs to member-end generalized DOFs. Columns organization can be seen in Table 4.

As previously anticipated, when building the kinematic matrix, the code distinguishes between nodes with degree equal to one ore larger than one:

- Nodes with degree = 1 (leaves in graph theory): these nodes represents supports or free ends, in any case nodes in which only one member is attached. Rows generated for this kind of nodes describe the interaction between the degrees of freedom of the member attached and external constraint, representing so the anchoring of the structure with the surrounding environment.

- Nodes with degree > 1 (joints): these nodes represent internal joints. For them, the algorithm considers all pairs of incident members, and for each one, if both ends are internally constrained for the same DOF type (rotation, horizontal, vertical), a row is added to the matrix to impose compatibility between those members at the node.

The script then tags every emitted row in *nodesId*, with *typeFlag* = 0 for rotation-type rows and *typeFlag* = 1 for translation-type rows. Later the function *reshape* is used to better store this information, that is finally used to count for each damage how many translational and rotational constraints are removed. In this way the two types of constraint will account for damage in a different proportion, according to constant *d*.

Finally, tiny numerical noises in *kinematic_matrix* are eliminated, zeroing entrances lower than 1e-7.

3.2.3 MINIMAL FRAME EXAMPLE

An example regarding a simple portal frame is provided to clarify the procedure. The structure consists of just three members: a couple of vertical elements of length 3, linked by an horizontal member of length 2. At the bottom left node, the vertical element n1 is fully fixed externally, while at the bottom right there's a roller (vertically restrained). Internal joints are rigid. The configuration is shown in Figure 8, for better clarity.

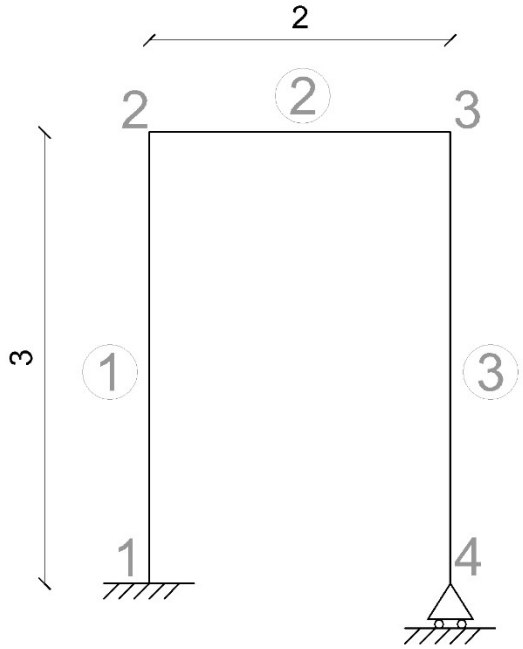


Figure 8 – Structural representation of the example

The structure is described through the three Excel sheets introduced in 3.2.1, representing topology, external constraints and internal end-releases. Table 1, Table 2 and

Table 3 report these three inputs.

Table 1 - Table describing the topology of the structure.

Beam number	Initial node	Final node	Angle	Length	Mass
1	1	2	π/2	3	2
2	2	3	0	2	2
3	4	3	π/2	3	2

Table 2 - Table describing the external constraints of the structure.

Node Number	Nodal displacements		
	Rotation	Horizontal	Vertical
1	1	1	1
2	0	0	0
3	0	0	0
4	0	0	1

Table 3 - Table describing the internal constraints of the structure.

Beam number	Initial node			Final node		
	Rotation	Axial	Lateral	Rotation	Axial	Lateral
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1

Starting from these inputs, the function *kinematic_matrixFunction* creates the kinematic matrix, reported below in Table 4. As previously described, columns are grouped in triplets (three columns per each member), while rows correspond to compatibility equations. Each non-zero entry indicates the participation of a specific member degree of freedom in the equation corresponding to the row in which it is found.

Table 4 - Kinematic matrix obtained from the structural description

Beam N1			Beam N2			Beam N3		
φ_1	u_1	v_1	φ_2	u_2	v_2	φ_3	u_3	v_3
1	0	0	0	0	0	0	0	0
1.5	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
1	0	0	-1	0	0	0	0	0
-1.5	0	0	-1	-1	0	0	0	0
0	0	1	0	0	-1	0	0	0
0	0	0	1	0	0	-1	0	0
0	0	0	-1	1	0	-1	0	0
0	0	0	0	0	1	-1.5	0	-1
0	0	0	0	0	0	-1.5	0	1

This simple example shows how the structural description, provided by the illustrated Excel sheets, is systematically converted into algebraic form.

3.3 Robustness calculation and iteration on damage combinations

After having generated the kinematic matrix, the MATLAB code goes on with the robustness evaluation. As widely discussed, the idea at this point is to test how the structure behaves when subsets of constraints are progressively removed. Each combination of removed constraints corresponds to a damage scenario and for each of them the algorithm computes a scalar value representing the robustness. Finally, the critical scenario is identified as the one giving the lowest value for the index *R*.

The procedure used to get the index is implemented into two nested loops. First, the algorithm selects the number of constraints to be removed (starting from 1, up to *n.* Being *n* the total number of rows

in the kinematic matrix). Then, for each case, it computes all the possible combinations by using the function *nchoosek*, saving them in the matrix *Cases*. Then, for each combination, the corresponding rows are deleted from the kinematic matrix and the so obtained matrix is analysed by computing the **null space** to detect the presence of admissible kinematic mechanisms. In case some mechanisms are presents, the resilience index R is calculated, based on the active mass and the type and number of removed constraints. If the obtained value of R is smaller than the current minimum R_{min}, it replaces it as the new critical case and then the process continues until all combinations have been exhausted.

The algorithm is conceptually straightforward, including all possible damage scenarios, but exactly this feature can be seen as a weakness. In fact the use of the function *nchoosek* guarantees completeness, but on the other hand, it results definitely memory-intensive, generating entire arrays of combinations. For even moderate values of n, the number of combinations quickly becomes unmanageable. For this reason, the code tends to become extremely slow or crash, just passing few dozens of constraints. Surely, the calculation of the null space for each combination also contributes to computational burden. So, the key drawback of this brute-force approach is the combinatorial explosion: the number of cases is measured in $\sum_{i=1}^{n}\binom{n}{i} = 2^n - 1$ cases. Where $\binom{n}{i}$ is the number of ways to remove i constraints from the total of n. Even for relatively small structures, the computational demand quickly becomes prohibitive. For example, with 20 constraints the algorithm must process more than one million cases, each requiring matrix manipulation and a null-space calculation. To illustrate this growth, Table 5 and Figure 9 report the number of combinations as a function of the total number of constraints, both in tabular and graphical form.

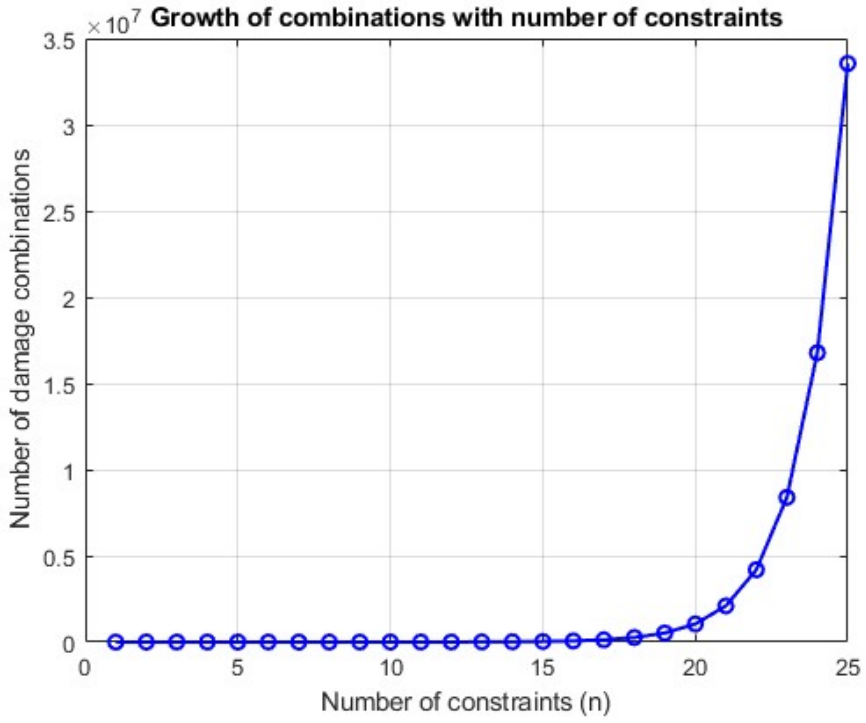


Figure 9 - Growth of damage combinations with the total number of constraints

Table 5 - Number of damage combinations as a function of the total number of constraints

Number of constraints	Number of combinations
1	1
2	3
3	7
4	15
5	31
10	1023
15	32767
20	1048575
25	33554431

This issue is intrinsic to the original version of the code and represents one of the principal directions in which developments should go, also in the context of this thesis. Next chapters will thus explore strategies to avoid, or at least mitigate, the computational burden.

3.4 Extension of damage combination to elements removal

An updated version of the code has been proposed by Prof. De Biagi, in order to extend the original implementation by considering also element removals withing damage scenarios. In fact, in the first version reported in 3.1 only constraint removals were considered (both internal and external), limiting damage scenarios to local constraints failures. The addition of this possibility was planned from the beginning, to follow what was theorized, but the first listing was developed like that for simplicity, to provide a beta version to start testing the metric. Anyway, element removals were somehow already considered by the removal of all the constraints linked to them. In any case, the new formulation now allows to consider every combination of constraints and elements removal, producing more general damage scenarios.

3.4.1 General structure of the new implementation

The updated script maintains the same global workflow - starting from constants definition, importing structural description from Excel file, generating kinematic matrixes and finally iterating on damages to calculate R for each of them – but it introduces a couple of new sections to account for element removals. Moreover, it introduces also a new constant, *e*, that is ideated as a normalization factor in the computation of robustness index. So having the possibility to act on the portion of damage coming from element removals by constant a, and on the portion coming from constraint removals by constant b, constant e can be used as a global factor acting on both portions.

To clearly show the modifications and briefly discuss about them, the updated parts of code are reported below, starting with the main iterative portion, now consisting in three stages.

```
constants.a=1;
constants.b=0.1;
constants.c=1;
constants.d=3;
constants.e=1;
```

```
...

% Only constraint removals
removedElement=[];
for i=1:noConstr
  ConstrCases=nchoosek(1:noConstr,i);
  for j=1:size(ConstrCases,1)
    removedConstraint=ConstrCases(j,:);
    [Ri(k)]=ResilienceIndex(kinematic matrix,massMatrix,ConstraintTypes,
constants,removedConstraint,removedElement);
    ...
  end
end

% Constraint and element removals
for ii=1:noEle
  EleCases=nchoosek(1:noEle,ii);
  for jj=1:size(EleCases,1)
    removedElement=EleCases(jj,:);
    KK matrix ele=[kinematic matrix (1:noConstr)'];

    for ee=1:length(removedElement)
      L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+1))>0);
      KK matrix ele(L,:)=zeros(length(L),size(kinematic matrix,2)+1);
      L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+2))>0);
      KK matrix ele(L,:)=zeros(length(L),size(kinematic matrix,2)+1);
      L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+3))>0);
      KK matrix ele(L,:)=zeros(length(L),size(kinematic matrix,2)+1);
    end

    EffConstr=find(KK matrix ele(:,end)>0);

    for i=1:length(EffConstr)
      ConstrCases=nchoosek(EffConstr,i);
      for j=1:size(ConstrCases,1)
        removedConstraint=ConstrCases(j,:);
        [Ri(k)]=ResilienceIndex(kinematic matrix,massMatrix,ConstraintTypes,
constants,removedConstraint,removedElement);
        ...
      end
    end
  end
end

% Only element removals
removedConstraint=[];
for ii=1:noEle
  EleCases=nchoosek(1:noEle,ii);
  for jj=1:size(EleCases,1)
    removedElement=EleCases(jj,:);
    [Ri(k)]=ResilienceIndex(kinematic matrix,massMatrix,ConstraintTypes,
constants,removedConstraint,removedElement);
    ...
  end
end
```

As seen, the main script is now organized into three separated loops:

- The first, considering constraint removals only, reproducing the original behaviour;
- The second, considering combined element and constraint removals. In this stage, the code creates an auxiliary matrix (KK_matrix_ele) for each combination of removed elements and zeros all the rows regarding constraints related with deleted elements. Then it extract the effective constraints that result still connected with remaining members (EffConstr), to ensure that subsequent combinations can include only meaningful rows of the kinematic matrix;
- Finally the third, considering element removal only, to isolate the influence of member loss.

3.4.2 MODIFICATIONS IN *RESILIENCEINDEX* FUNCTION

Also for the function *ResilienceIndex* some variations were provided. The update consists in a new set of instructions that are supposed to properly handle element deletions and to clean up the kinematic matrix accordingly. In particular, when an element is removed, constraint rows linked to it are zeroed. At the end of this operation, rows that become entirely null are excluded from subsequent calculations.

The relevant modified portion of the function is reported below:

```
for ee=1:length(removedElement)
  L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+1))>0);
  kinematic matrix(L,:)=zeros(length(L),size(kinematic matrix,2));
  L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+2))>0);
  kinematic matrix(L,:)=zeros(length(L),size(kinematic matrix,2));
  L=find(abs(kinematic matrix(:,3*(removedElement(ee)-1)+3))>0);
  kinematic matrix(L,:)=zeros(length(L),size(kinematic matrix,2));
end

noRemovedConstraint=length(removedConstraint);
for rc=1:noRemovedConstraint
  if max(abs(kinematic matrix(removedConstraint(noRemovedConstraint-
rc+1),:)))==0
    removedConstraint(noRemovedConstraint-rc+1)=[];
  end
end
```

This set of instructions ensure that removed elements and corresponding constraints are excluded from the reduced system considered in the null-space computation. At this point, the mechanism looks then for *NU=null(modKinMatrix)* and the calculation of m_a and m_h complete the ingredients needed to express Ri.

The resilience index is then computed as:

$$R^i = \frac{D^i}{e\ {m_a}^i} = \frac{a\ {m_h}^i + b(c\ {k_R}^i + d\ {k_T}^i)}{e\ {m_a}^i} \tag{3.1}$$

3.4.3 COMPUTATIONAL IMPLICATIONS

As it's easy to imagine, introducing element removals increases significantly the number of possible combinations. In fact the total number of damage cases is now the sum of constraint, element and mixed removals. The enumeration process, that is still based on the function nchoosek, becomes thus even more computationally demanding. Although last implementations guarantee completeness, they reinforce even more the need for future development aimed at reducing the combinatorial burden. Approaches for this aim could consider bounding strategies or, more likely, selective sampling of critical scenarios.

In any case, this extension of the code makes the tool capable of reproducing a wider range of failure mechanism, getting to a more realistic and complete analysis.

4 CONSTANTS CALIBRATION

The robustness index (R) introduced in previous chapters, considering developments introduced in 3.4, depends on five constants (a, b, c, d, e), which weight the contribution of different damage actors. A proper calibration of these constants is crucial to obtain results that are physically meaningful and consistent with engineering intuition. This study is founded on the calibration proposed in the original formulation of the method (Chiaia, et al., 2019), where the constants were set equal to a = 1, b = 1/5, c = 1, d = 3. Values adopted for *a* and *c* were defined for sake of simplicity. Constant *b* was defined by comparing the effect of element and constraints removals for a specific structural scheme, needing thus for generalization. Constant *d* was set equal to 3, to reflect the assumed relative importance of different types of constraint losses, in this sense translational constraints losses have a higher weight in damage magnitude. This because it's assumed that to lose a rotational constraint is much more likely to happen, so a translational constraint loss has to be part of a more severe damage. Finally, constant *e* wasn't defined yet.

First, the calibration of *b, c,* and *d* has been carried out by using a set of six reference structures, illustrated in Figure 10. These structures are characterized by having the same geometry, and elements that composed them have all the same mass, this to avoid any influence of mass on results and isolate the effects of constraints differences on robustness (since constants *b, c* and *d* are related to constraints configuration). Moreover, having similar structures, which differs just for external constraints configuration, allows to establish an expected hierarchy of *R*, which in particular should decrease progressively from Structure 1 to Structure 6 (more to less robust). This first part has been carried out by using the original model of the code, which doesn't account for element removals. This because the calibration of this first stage isn't related to element removal, the goal is just to find an equilibrium between translational and rotational constraint removal weight and eventually normalize their contribution to the damage value. After this first part, constant *a* will be calibrated in order to guarantee an equilibrium in contribution coming from element and constraint removals and finally constant *e* will be studied, in case a normalization will be considered necessary.

Principal features of the set of structures used for calibrating constants b, c and d are reported in Table 6.

Table 6 – Basic geometry and mass features of reference structures

Beam	1	2	3	4
Length	4	4	3	2
Mass	2	2	2	2

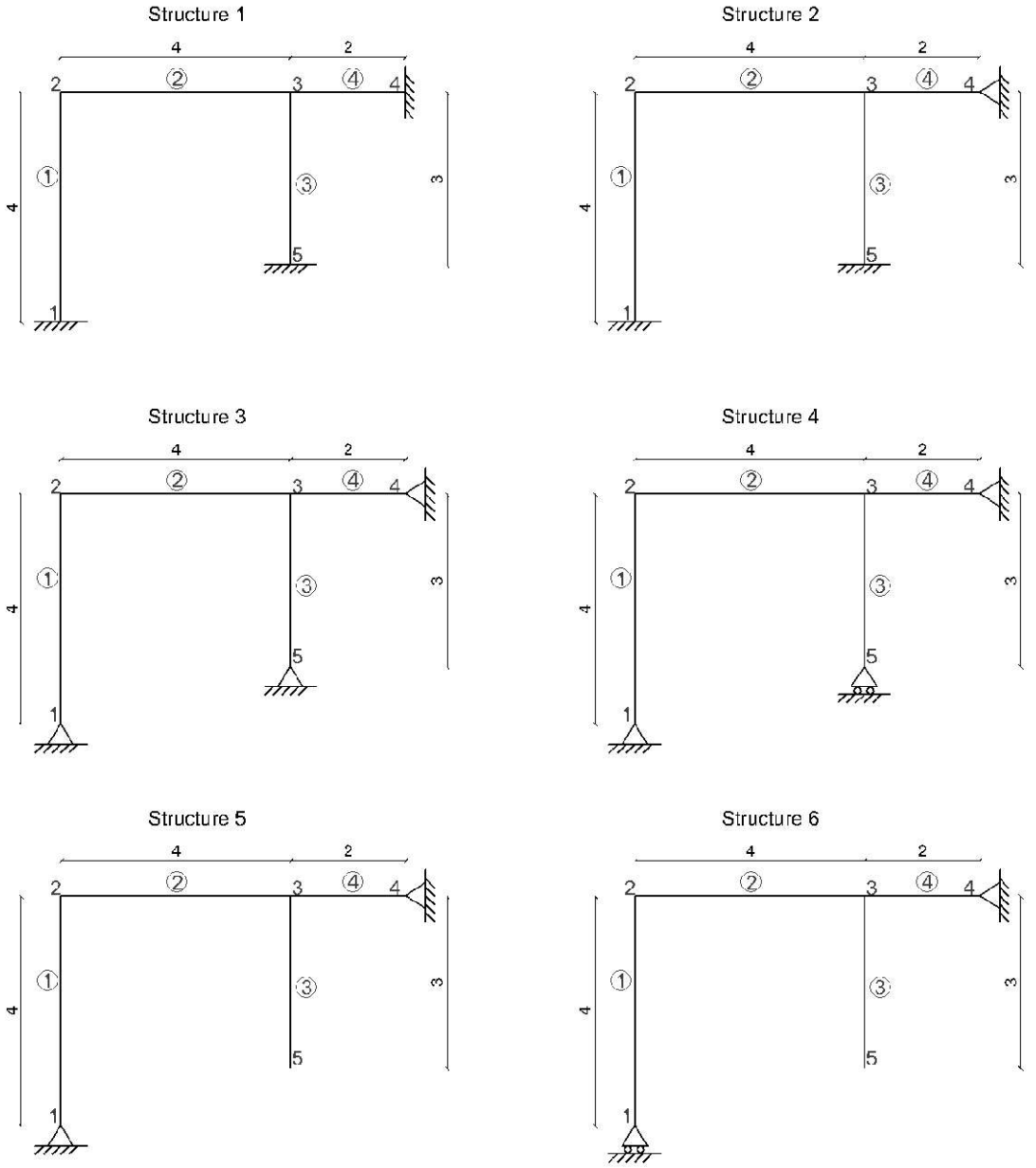


Figure 10 – Reference set of six structural configurations adopted for constants calibration

4.1 Calibration of Constant *b*

Constant b governs the contribution of constraints redundancy, weighting the penalty associated with the removal of constraints in general. In the original formulation, a = 1, c = 1, and d = 3, while b was set equal to 0.2. That value of b was obtained equating two scenarios that lead to complete collapse, one involving just constraint removals and the other just element removal, for a specific structural scheme (Chiaia, et al., 2019). Being fair for that specific configuration, that calibration was lacking in generality, since the removal of a certain number of degrees of freedom can determine a huge or an insignificant damage, depending on dimensions and complexity of the structure. Furthermore, looking for an index that is supposed to have a value between 0 and 1, the original value set for constant b isn't good because it's likely to bring the index r out from that range.

To check these properties, a simple analysis has been conducted, calculating the robustness index R for the set of structures introduced in the previous paragraph. It has been done keeping a, c and d as they were in the original formulation and varying the constant b, which was assuming the following values: [0.05; 0.1; 0.2; 0.5; 1].

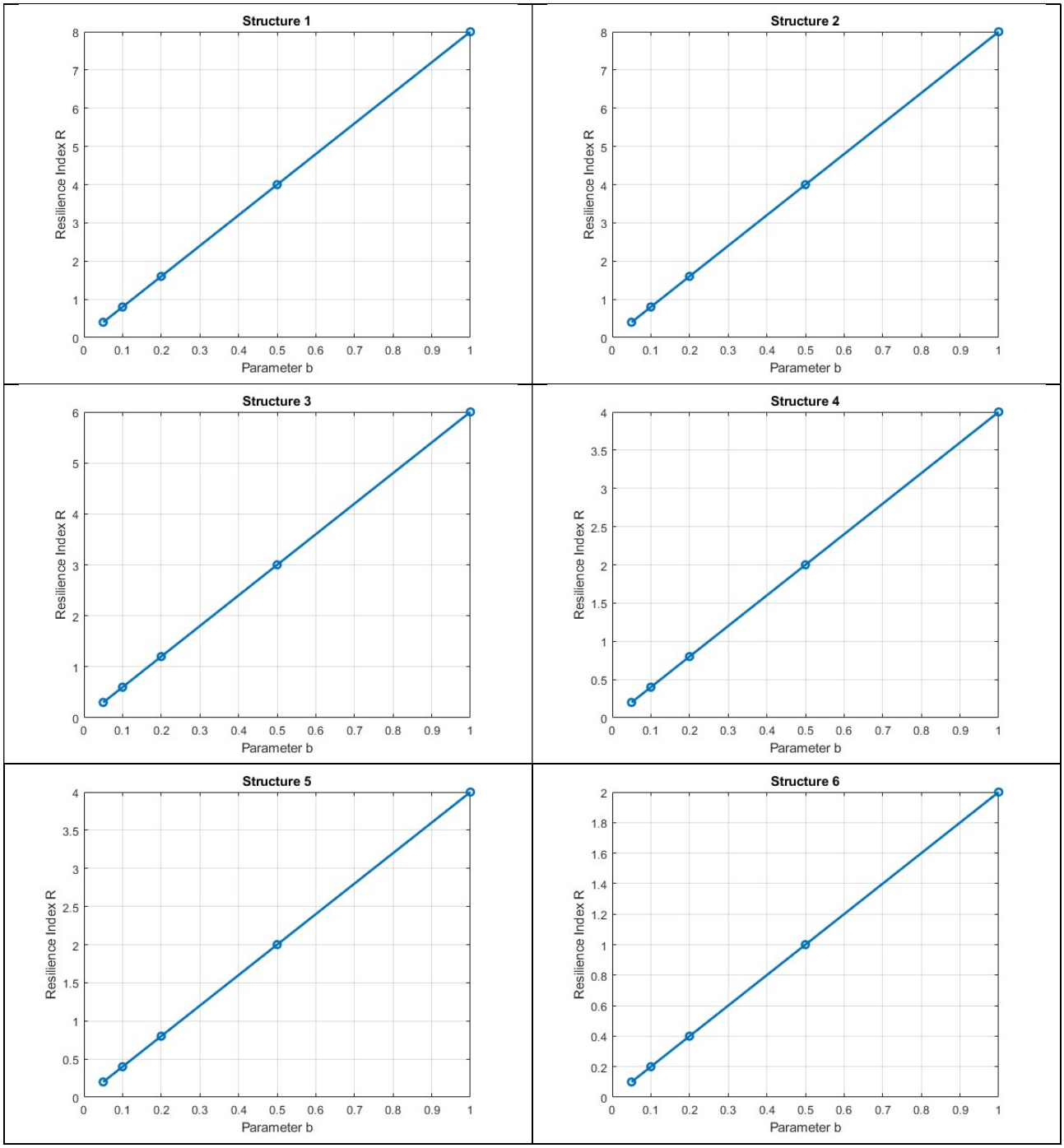


Figure 11 – Variation of robustness index R with constant b, across the six reference structures

The results show that increasing b scales the index proportionally, with R values growing linearly as expected. Results show how, setting the constant b equal to a fixed value, the robustness index R is likely to go out of the range wanted for it, especially if structures get more complex. Anyway, as

expected, varying the value of the constants, the critical combination of removed constraints remains the same.

4.1.1 NORMALISATION OF *B*

A normalisation of constant b is proposed, so that the damage contribution deriving from support removal remains bounded in [0,1]:

$$b * (c * k_R + d * k_T) \in [0,1] \tag{4.1}$$

where k_R and k_T represent respectively the amount of removed rotational and translational restraints. To enforce this bound for any admissible removal pattern, b is defined as a function of the total number of constraints present in the structure, taking into account weighting constants, as showed in the formula below:

$$b = \frac{1}{c * N_R + d * N_T} \tag{4.2}$$

where N_R and N_T represent the total amount of rotational and translational constraints. This choice ensures that the maximum possible removal never yields a damage contribution above 1, in particular the removal of all constraint would lead to a damage value equal to 1 (excluding element removal).

The motivation behind this approach is that the original constant (b = 1/5) could yield non-representative damage values. That was clear comparing systems of very different redundancy: in fact, with the original calibration, removing one translational restraint in a statically determinate portal (leading to collapse) and removing one translational restraint in a highly redundant and complex structure would produce the same damage term, despite the drastically different scenario from an intuitive point of view. Hence, this size-aware normalisation has been proposed. Tests were performed on the six reference structures, using the just formulated value of b and keeping a = 1, c = 1, and d = 3 as in the original formulation.

Table 7 – Results of robustness index using normalized constant b

Structure	Normalized value of b	Rmin	Critical constraints combination
Str1	0.020408	0.163265	[1 4 17 20]
Str2	0.020833	0.166667	[1 4 16 19]
Str3	0.021739	0.130435	[3 9 12 15]
Str4	0.023256	0.093023	[3 15]
Str5	0.025000	0.100000	[2 3]
Str6	0.027027	0.054054	[5 8]

The normalisation generally delivered the expected hierarchy, with the robustness index *R* decreasing from Structure 1 to Structure 6. A notable exception arises between Structures 4 and 5: in fact, Structure 5 shows a slightly larger value of *R*, which is not what is supposed to be, since Structure 5 has less degrees of constraint and so it's supposed to be less robust. In fact, for a couple of very similar structures, that define *R* for the same combination of removed constraints, it results that the

structure with the lower number of total constraints has a higher value of *R*, since the damage is considered larger. This edge case suggests a potential refinement, to be developed in case if it proves significant in broader testing. The potential refinement consists in defining the normalisation with respect to the number of structural elements rather than restraints.

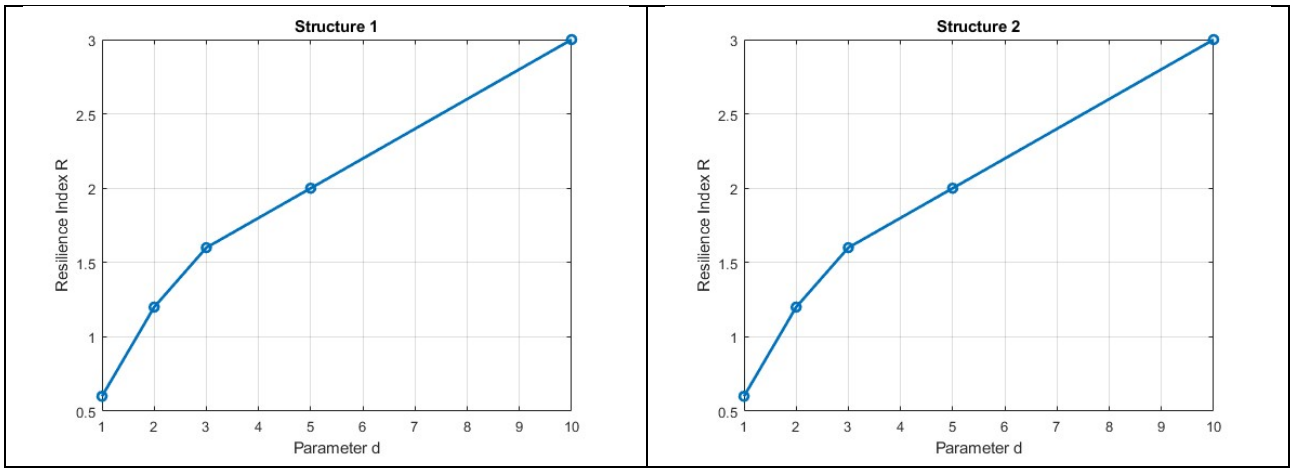
4.2 Calibration of Constants c and d

As anticipated, in the original formulation, constants c and d were set respectively equal to 1 and 3. This choice was done to reflect the greater probability of losing a rotational restraint (through plastic hinge formation), compared to a translational one. Consequently, translational removals were weighted more heavily (d = 3), in order to recognize the loss of a translational constraint as a more severe damage.

For sake of research, some tests have been conducted to see the variation of the robustness index with those constants. Since the index depends only on the ratio *c:d*, calibration can be performed by varying just one of the two constants. In this case was chosen to vary d while keeping c = 1. Values of d = {1, 2, 3, 5, 10} were tested, with a = 1 and b = 0.2. Results are illustrated in Figure 12.

The results show that higher values of *d* amplify the damage contribution, increasing R, as expected. However, beyond certain thresholds and in particular for simple structures, the governing critical scenarios exclude translational removals, since these become excessively penalised. This leads to a plateau effect, in which R does not increase further despite higher value of *d,* since translational constraint removals are no longer taken in consideration.

Finally, considering the tests conducted, it follows that the ratio *c:d* must be carefully balanced. Too high a value of *d* overemphasises rotational removals, while too low a value fails to capture translational removals relative severity.



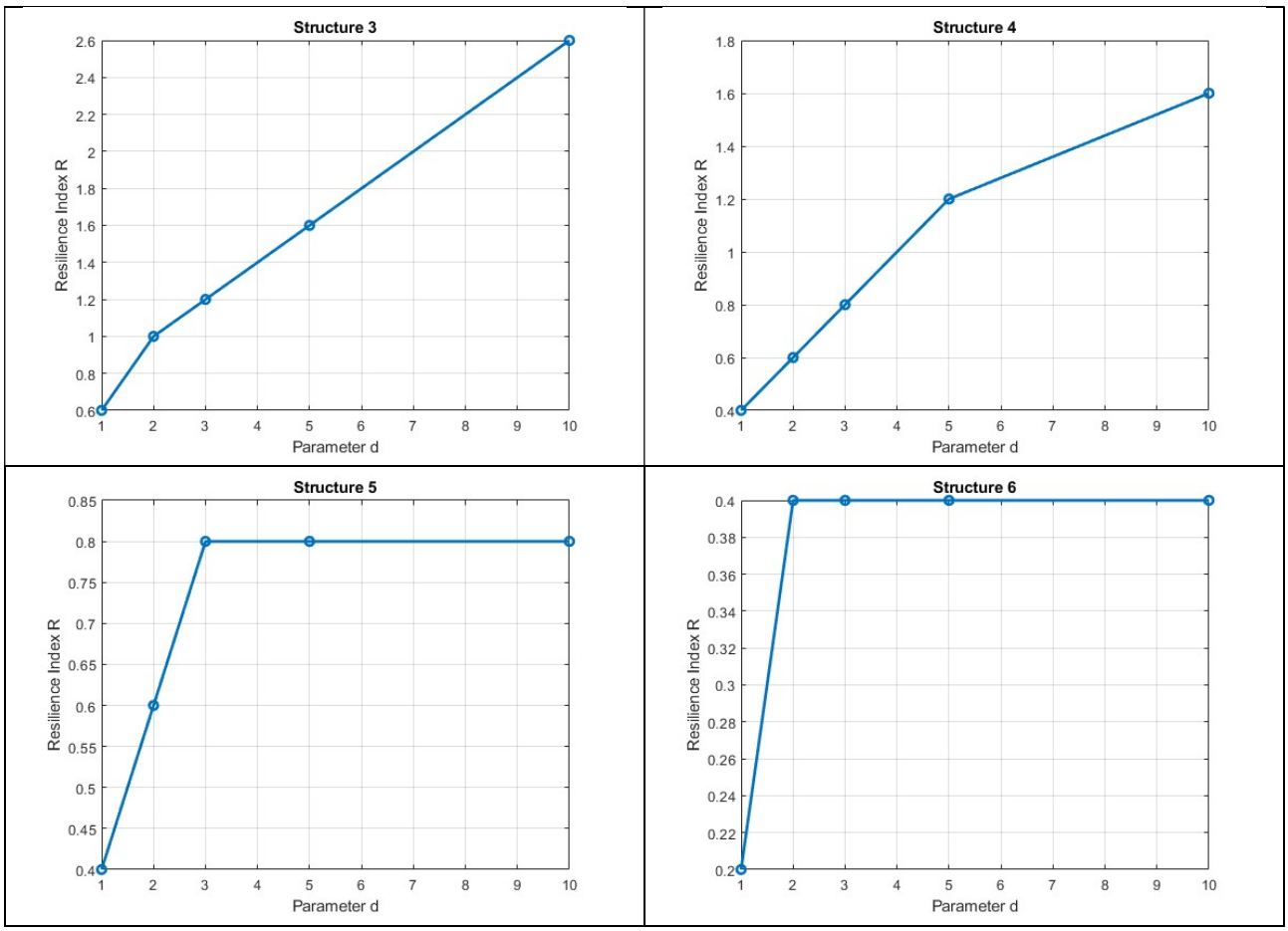


Figure 12 - Variation of robustness index R with constant d, across the six reference structures

The original definition of constants c and d (c = 1, d = 3), selected to reflect the less likelihood of translational constraint losses compared to rotational ones, started to show a defect during the calibration of constant a (see next section). In particular, it was observed that maintaining the original ratio, was often leading to unrealistic critical combinations. In fact, in several cases, the governing configuration corresponded to the removal of all translational constraints at the external supports, either horizontal or vertical, resulting thus in a rigid-body motion of the entire structure. Although that scenario was giving the mathematical minimum value of the robustness index (having mass fraction m_a equal 1 and initial damage relatively low for systems with few external supports), it was representing a physically meaningless mechanism. Such a situation, for civil structures, would correspond to the complete detachment of foundations or the formation of mechanisms like trolley or even double pendulum at the base of the structure. It's evident that this kind of conditions are basically impossible, considering that foundations are properly designed.

To avoid considering that unrealistic outcome, ensuring so that the governing mechanism corresponds to a credible collapse pattern, the value of the constant d was finally set equal to 10. With this adjustment, the influence of those trivial modes is significantly reduced, leading to more meaningful and realistic configuration for the critical damage combination.

Since what matter is the ratio between constants c and d, rather than their absolute values, constant c is maintained equal to 1.

4.3 Calibration of Constant a

After having developed the calibration of constants b, c and d, the following step is to determine a value for constant a. This parameter multiplies the term m_h, corresponding to the mass fraction of elements removed in the initial damage. This constant was originally fixed equal to 1, for simplicity. However, due to changes that have been just described about the other constants, it's possible that some refinements will be necessary. In particular, a normalized value for constant b has been defined. This won't be the case for constant a, since m_h is defined as a fraction, representing the ratio between removed and total mass, and therefore lies within the range [0, 1]. The damage magnitude coming from element removals is therefore already normalized. The goal in this stage is so to ensure a balance between element and constraints removal in the determination of the critical scenario, and so the robustness index, guaranteeing to consider every scenario without prioritizing anyone. To this end, several tests were carried out on structures of different typologies, geometries, and levels of redundancy.

But before to perform a set of analysis with the full index evaluation, a preliminary theoretical comparison is conducted to check the scale compatibility between element and constraint removal contribution. For a set of various structures, differing on dimensions, number of elements, structural typologies, ecc… the damage resulting from the removal of one or more elements has been compared to the damage obtained by removing all the constraints connected to those elements. All this with the purpose to compare those values and check for the balance among them. Ideally, the damage coming from the two different ways should be of the same order of magnitude, with a slightly lower value for element removals, because otherwise element removals would never emerge as part of the governing combinations. With this purpose a set of structures has been defined and for each of them different damage scenarios, as illustrated in Figure 13.

The comparison has been performed using the following values for constants:

- $a = 1$;
- $b = \frac{1}{c * N_R + d * N_T}$ (normalized as defined in 4.1.1);
- $c = 1$;
- $d = 10$;
- $e = 1$.

The same mass has been assigned to each element, to avoid any bias. Results are summarized in Table 8.

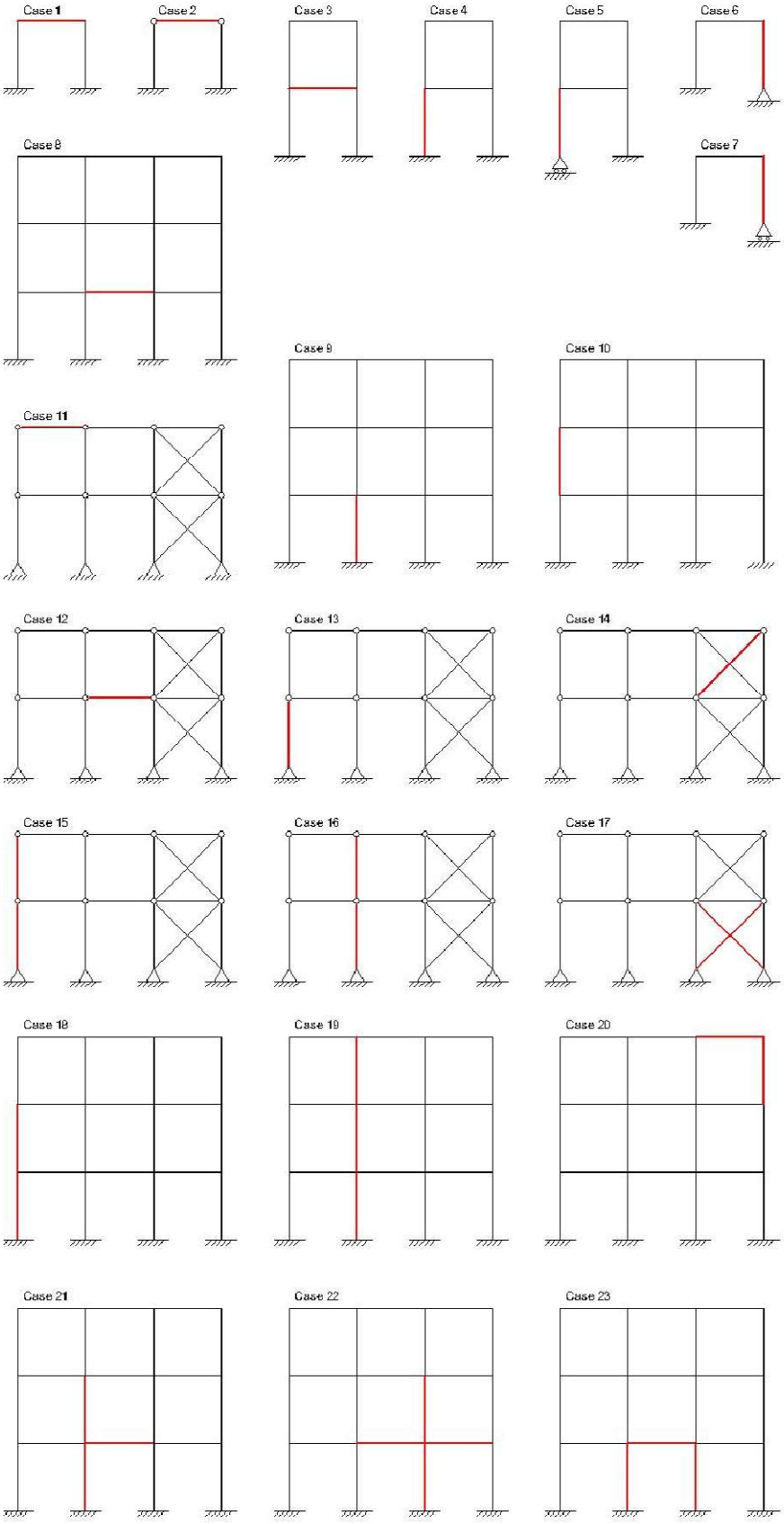


Figure 13 - Structural schemes used for the preliminary comparison between element- and constraint-based damage.

Table 8 – Results of comparison between element-based and constraint-based damage values for the tested structures.

Case n.	Removed elements	Removed constraints		Damage from elements	Damage from constraints
		k_T	K_R		
1	1	4	2	0.3333	0.5000
2	1	4	0	0.3333	0.4615
3	1	8	4	0.1667	0.4000
4	1	6	3	0.1667	0.3000
5	1	5	2	0.1667	0.2576
6	1	4	1	0.3333	0.4815
7	1	3	1	0.3333	0.4167
8	1	12	6	0.0476	0.1250
9	1	8	4	0.0476	0.0833
10	1	8	4	0.0476	0.0833
11	1	6	0	0.0556	0.0588
12	1	16	0	0.0556	0.1569
13	1	6	0	0.0556	0.0588
14	1	14	0	0.0556	0.1373
15	2	10	0	0.1111	0.0980
16	2	16	0	0.1111	0.1569
17	2	26	0	0.1111	0.2549
18	2	12	5	0.0952	0.1220
19	3	26	13	0.1429	0.2679
20	2	10	5	0.0952	0.1042
21	3	26	13	0.1429	0.2679
22	4	30	15	0.1905	0.3125
23	3	24	12	0.1429	0.2500

Results confirmed that the two quantities are consistent in scale. Almost in every case the damage due to element removal is lower than that caused by the corresponding constraint removal, but still comparable in magnitude, as expected. The same analysis results are shown in the in the scatter plot reported below in Figure 14, which shows the relation between the two contributions in relative terms. In the graph, cases of multiple and single elements removal are represented with different symbols, allowing to assess that there's not any results trend coming from this feature. The dashed line represents equality ($D_{elem} = D\_c_{onstr}$).

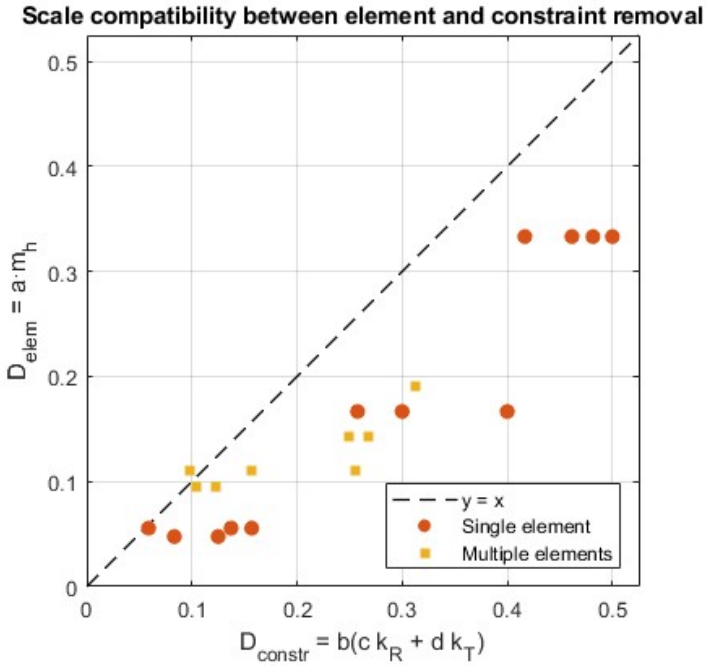


Figure 14 - Scatter plot comparing damage from element and constraint removal

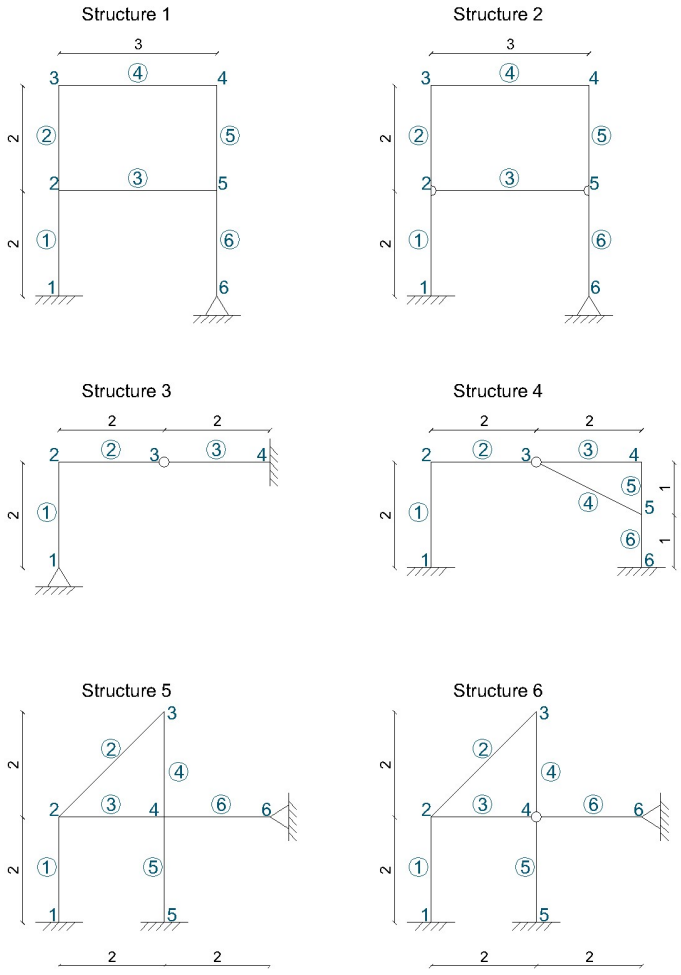


Figure 15 - Benchmark structures employed in the extended analysis for constant a calibration

After the scale compatibility study, a further numerical analysis has been carried out to effectively assess if both damage sources could appear in critical combinations. To do that a has initially been set equal to 1, to be then lowered to 0.5, 0.2 and 0.1. Other constants have been maintained as in the previous analysis. For this study, six structures have been analysed, using the complete version of the Matlab code. Those structures are reported in Figure 10, they were specifically conceived to broaden the variety of topology, boundary conditions, and redundancy levels. As usual, mass is constant throughout all the elements, to avoid any bias. Results are finally reported in tables below (Table 9, Table 10, Table 11, Table 12), where the robustness index *R* and the corresponding critical combination were identified for each case.

Table 9 – Numerical analysis for constant "a" calibration, results for a = 1

Case n.	Removed constraints		Removed elements	Critical combination	R
	kR	kT			
1	5	0	0	Only constr.	0.0239234
2	3	0	0	Only constr.	0.0146341
3	2	0	0	Only constr.	0.0365854
4	5	0	0	Only constr.	0.0241546
5	3	2	0	Only constr.	0.0842491
6	4	1	0	Only constr.	0.0524345

Table 10 - Numerical analysis for constant "a" calibration, results for a = 0.5

Case n.	Removed constraints		Removed elements	Critical combination	R
	kR	kT			
1	5	0	0	Only constr.	0.0239234
2	3	0	0	Only constr.	0.0146341
3	2	0	0	Only constr.	0.0365854
4	5	0	0	Only constr.	0.0241546
5	3	2	0	Only constr.	0.0842491
6	4	1	0	Only constr.	0.0524345

Table 11 - Numerical analysis for constant "a" calibration, results for a = 0.2

Case n.	Removed constraints		Removed elements	Critical combination	R
	kR	kT			
1	5	0	0	Only constr.	0.0239234
2	3	0	0	Only constr.	0.0146341
3	2	0	0	Only constr.	0.0365854
4	5	0	0	Only constr.	0.0241546
5	5	0	1	Mixed	0.0774725
6	4	1	0	Only constr.	0.0524345

Table 12 - Numerical analysis for constant "a" calibration, results for a = 0.1

Case n.	Removed constraints		Removed elements	Critical combination	R
	kR	kT			
1	0	0	1	Only elements	0.0200000
2	3	0	0	Only constr.	0.0146341
3	2	0	0	Only constr.	0.0365854
4	5	0	0	Only constr.	0.0241546
5	5	0	1	Mixed	0.0524725
6	0	0	2	Only elements	0.0333333

Results show of course how the variation of constant a doesn't affect at all the result until element removals aren't included. But more importantly, for a =1 and a = 0.5, element removals weren't included at all in critical scenarios, which were of type "Only constraints" for every structures. However, when a decreased to 0.2 or 0.1, the governing mechanism shifted to "Mixed" (involving both element and constraint removals) or even to "Only elements" for some structures. It's also important to highlight that those shifts don't consist in a strong variation of R, confirming the overall stability of the metric. Finally, results' trend indicates that smaller values of a enhance the balance between the two damage sources, preventing the metric from being dominated by constraint-related effects. In other words, a reduced a makes the influence of element loss more perceptible.

Considering all these observations, a reasonable compromise for constant a lies between 0.1 and 0.2. These values are leading in fact to critical scenarios including both damage sources, with results that respect engineering intuition and that are physically consistent. Nevertheless, a broader validation study, considering a larger number of structures with different geometries, degrees of redundancy and in particular scale, would allow further refinement of this constant value. Unfortunately, further analyses including bigger structures weren't possible due to the well-known computational limitations.

4.4 Calibration of Constant e

Finally, constant e is the last one to be defined within the robustness index formulation. As expressed in equation 3.1, it appears in the denominator of the index formula, as a scaling factor for the overall result (not acting specifically on a damage source like a, b, c, d). It serves to ensure that the resulting robustness values remain within a convenient and easily interpretable numerical range (ideally 0 to 1), being so a kind of global normalization coefficient. The formulation is here recalled for convenience: $R^i = \frac{D^i}{e\, {m_a}^i}$.

Looking at results of the previous section 4.3, it's evident how the computed values of R were consistently in the range between 0.01 and 0.1. Such small values, even if mathematically correct and reasonable, make the index interpretation less intuitive. Due to this reason, it turns out convenient to rescale the results by adjusting the constant e. In particular, setting e = 0.1, that means to multiply all values of R by a factor of 10, makes the results to stay in the optimal range for readability and

interpretability, without affect relative comparison between different structures. In this way, an index close to 1 would identify a robust structure, while values approaching zero would identify low level of robustness.

It would be good to validate this scaling factor through further analysis including a large number of structures and especially larger and more complex structural systems. Nevertheless, the goodness choice of the factor is expected to be confirmed, since both damage terms are normalized to structural size (to the number of total elements and constraints). Consequently, the set of $e = 0.1$ is expected to provide clarity for robustness interpretation, without altering in any way the physical and mathematical significance of the metric.

5 CORRELATION WITH BASIC PROPERTIES

The robustness metric illustrated in the previous chapters is now tested against a set of operative properties to verify its consistency from a theoretical and engineering point of view. The set of properties was originally proposed by Professor De Biagi as a collection of logical and physical expectations that every robustness metric is supposed to satisfy to be considered reliable and meaningful. Each property reflects a specific structural behaviour, providing a benchmark to test its reliability.

The aim of this chapter is therefore twofold: first, to verify whether the Matlab implementation of the metric behaves coherently with expected outcomes of the various properties, and second, to get insight about strengths and weaknesses of the metric, eventually identifying potential refinements in case of deviation from the expectations.

All the analyses were performed using the Matlab implementation introduced in the previous chapters. Unless otherwise specified, the constants were set equal to:

$$a = 0.2,\ \ b = b_{norm},\ \ c = 1,\ \ d = 10,\ \ e = 0.1;$$

While an equal mass value was assigned to all members, isolating the effects of the tested property.

Each section of this chapter addresses one operative property individually. For each of them, the expected behaviour is first discussed from a theoretical point of view, introducing also some intuitive expectations, after that the description of numerical tests is reported together with their results. In case the original formulation doesn't fully comply with the property, specific solutions or algorithmic improvements will be eventually proposed and discussed.

The seven selected properties form thus a validation framework for the proposed robustness metric, providing both a theoretical and computational assessment of its reliability.

5.1 Verification of Property 1 – Null Robustness of Statically Determinate Structures

This first section tests the metric for the first operative property, which states that a solid robustness metric should exhibit null robustness for statically determinate schemes. To assess this feature, the reference structure reported in Figure 16 has been used. Dimensions are reported in the figure, while masses are the same for each element, to avoid any bias.

Applying the Matlab implementation to the isostatic scheme, a robustness index value of $R = 0.1389$ is obtained. Coherently with the extreme low redundancy of the structure, the calculation exhibit a low value of the index, which however is not null. Some other tests were conducted on similar schemes, reporting always low values, differing depending on geometry, showing though an undesirable variability among different isostatic structures.

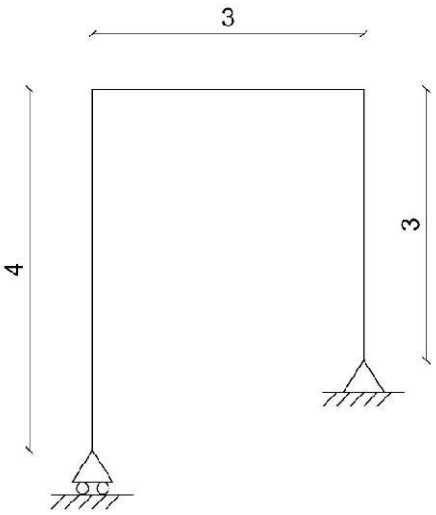


Figure 16 – Reference structure for Property n1

To align results with expectations defined by Property 1, harmonising thus outcomes across isostatic cases, an operational rule is proposed for being included in the implementation: whenever the total collapse is triggered by the removal of a single degree of constraint without any element removal, then *R* is set equal to zero and the search for critical scenarios is terminated. This rule is supposed to identify total collapses, that means situations in which kinematic motions include the whole structures by checking when *ma = 1*. With this adjustment, the metric consistently returns *R = 0* for statically determinate schemes, ensuring consistency across nominally equivalent static configurations.

5.1.1 CODE CHANGES AND RATIONALE

To get the just discussed results, a couple of localised changes were proposed:

- First, the introduction of a terminal rule inside the function *ResilienceIndex*;
- Second, the introduction of a loop-level stop flag in the function *Main*, to exit cleanly from the damage scenarios iteration as soon as the rule fires.

Both the modifications are reported below.

About the terminal rule in *ResilienceIndex*, the proposal is for the function to include a boolean output called *isTerminal* and force *R = 0*, when the following three conditions are simultaneously met:

- One constraint removed;
- Zero element removed;
- Mass fraction involved in the kinematic mechanism essentially equal to 1.

A small tolerance range is proposed *(epsTol)* to prevent numerical artefacts (e.g., *ma = 0.9999998*) from obstructing a physically obvious total-collapse case.

The just introduced code changes are reported below:

```
% ResilienceIndex(...)  → now returns [Ri, isTerminal]
function [Ri, isTerminal] = ResilienceIndex(...)
```

```
  isTerminal = false;
  epsTol     = 1e-6;

  % ... (standard computation up to 'ma') ...
  % ma = sum(M)/sum(massMatrix(:,2));

  % === Operational rule for Property 1 (isostatic schemes) ===
  if isempty(removedElement) && numel(removedConstraint)==1 && (abs(ma-1) <
epsTol)
      Ri = 0.0;
      isTerminal = true;   % signal the caller to stop iterating
      return
  end

  Ri = D/(e*ma);
end
```

A brief resume of changes is now reported, explaining the code lines and the applied functions.

- *isempty(removedElement)* ensures no elements are removed;
- *numel(removedConstraint)==1* captures the single-constraint removal typical of isostatic failure;
- *abs(ma-1) < epsTol* formalises if the mechanism involves the entire mass.

Under these conditions, the property prescribes null robustness; hence the implementation set *Ri = 0* and flag *isTerminal = true*.

At this point, it follows a brief explanation of the early-exit logic in the caller. A global flag *stopAll* is added, together with a check on *isTerminal* after each call. In case the rule triggers, *R = 0* is store as a result and the implementation breaks immediately out of the loop, to avoid unnecessary computations. The just introduced code changes are reported below:

```
stopAll = false;  % global stop flag

% Example: constraints-only loop (same pattern applied elsewhere)
for i=1:noConstr
  if stopAll, break; end
  ConstrCases = nchoosek(1:noConstr, i);
  for j=1:size(ConstrCases,1)
    if stopAll, break; end
    removedConstraint = ConstrCases(j,:);
    [Ri(k), isTerminal] = ResilienceIndex(kinematic matrix, massMatrix, ...
                                          ConstraintTypes, constants, ...
                                          removedConstraint, removedElement);
    % update RStore as usual...
    if isTerminal
       RStore = 0;                % enforce the null value
       bestRemovedConstraint = removedConstraint;    %to store critical scenario
       bestRemovedElement    = removedElement;       %to store critical scenario
       bestCase              = 'only-constraints';
       stopAll               = true; % break all loops cleanly
       break;
```

```
    end
    k = k+1;
  end
end
```

With the proposed changes, described in this section, the metric is now consistent with the first operative property, returning null robustness for statically determinate structures, while leaving all remaining analysis pathways unchanged.

5.2 Verification of Property 2 – Null Robustness of Statically Indeterminate Structures, but fixed by means of Statically Determinate Constrains

The second operative property is strictly related to the previous one, stating that a structural system which is statically indeterminate in its internal scheme, but externally fixed through determinate constraints must also exhibit null robustness. In this kind of configurations, the removal of any single external constraints immediately triggers a full collapse mechanism. That means that even if a structure has a high level of internal redundancy, it basically possesses no intrinsic robustness. This section tests thus the metric against this second property, by using the structure reported in Figure 17 as the reference configuration.

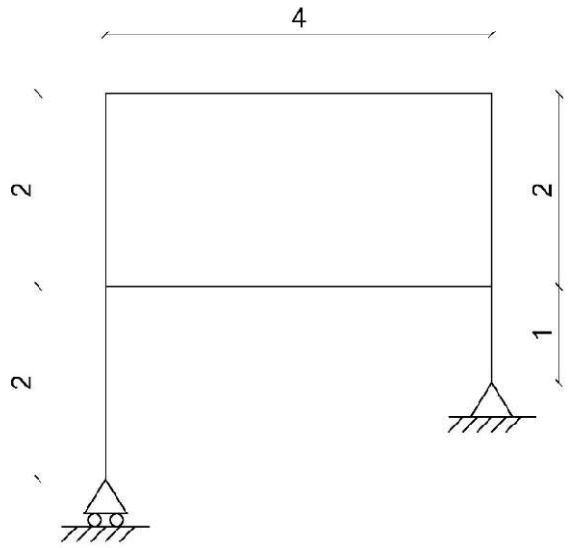


Figure 17 – Reference structure for Property n2

With the original Matlab implementation, a robustness value of $R = 0.1010$ is obtained. Again, just like for the first property, it's correctly a low value, but it isn't the null value requested by the property. Moreover, analysing test results it was discovered that the critical combination identified by the algorithm involved the removal of two rotational constraints. In particular, the removals were about the internal rotational constraints binding one of the base columns to the two connected members. This means that the external static determinacy wasn't effectively considered in determining the critical scenario. This discrepancy from expectations arose from the different weighting factors

assigned by constants calibration, which make the removal of rotational constraints lighter in term of damage.

Anyway, the adjustments introduced for Property n1 (in 0) resolve this issue as well, providing consistency among this kind of structures. In the updated version of the code developed for the previous property, the condition that enforce $R = 0$ whenever a single constraint removal leads to full body mechanism, effectively captures this critical behaviour also for structures fixed by statically determinate constraints. Therefore, the revised algorithm ensures compliance with Property n2, returning $R = 0$ for externally determinate structural schemes.

5.3 Verification of Property 3 – Insensitivity to the Addition of Elements with Negligible Stiffness

5.3.1 Baseline verification with equal masses and manual zeroing

The third operative property concerns the addition of an element with local stiffness properties tending to zero, stating that such addition must not affect the robustness of the original structure. The situation can be seen in practical terms as the inclusion in the scheme of some non-structural elements. Analytically, the property statement means that the robustness index should remain unchanged if an extremely flexible member, practically unable to transmit forces, is introduced in the system. An additional element, weak as described, should not alter thus either the critical collapse mechanism or the robustness index value.

To assess whether this property is respected by the original implementation of the metric, a preliminary test has been carried out considering the structural scheme in Figure 18. In the illustration the scheme is represented with and without the weak element, which is shown with a thinner line. Using the original version of the code, the same mass value is assigned to every element. The structural scheme has been specifically selected as to have the weak element in a position/configuration such as to be important for the critical scenario identification. Otherwise, having an additional weak element in a position in which it doesn't affect critical scenario identification, could avoid erroneous metric behaviour by being identified.

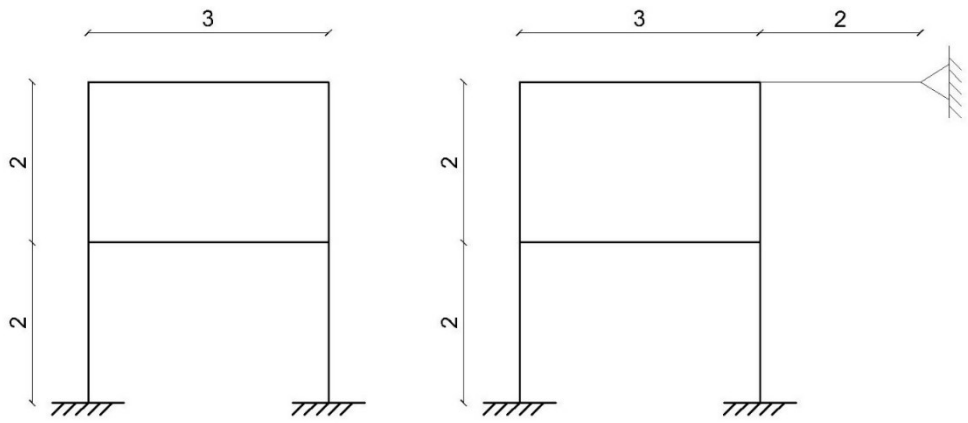


Figure 18 – Reference structure for Property n3

So, the two structural schemes were analysed, to allow a comparison of the so obtained robustness indexes. The analysis determined robustness values of $R = 0.457$ for the configuration without the weak element and $R = 0.548$ after its inclusion. The difference shows off an artificial increase in robustness that is not physically justified, for the already discussed reasons. This behaviour highlights an intrinsic limitation of metric original formulation. In fact, to be fundamentally a connectivity-based index, means that the metric evaluates robustness solely according to the topological connections between nodes and elements, without accounting in any way to mechanical properties of members. Hence, even a weak element that is structurally irrelevant still increases connectivity and so the calculated robustness, since the metric has no information about the strength of it.

To overcome this issue, the most immediate solution is to assign a mass equal to zero to all elements that are characterized by negligible stiffness. This manual method is drastic but also conceptually straightforward, in fact the removal of such an element from a structural system doesn't account in damage magnitude for the metric, just like the removal of such an element doesn't change a structural system in real terms. This simple approach has so been tested for the same structure as before (Figure 18), obtaining an index value of $R = 0.441$. This result is very close to the one obtained for the structure without the weak element, which was $R = 0.457$. Results confirm thus that manual zeroing can be a solution to restore the expected behaviour, however this method still require the user to decide *a priori* which member are supposed to be zeroed. To leave this task to the user means to make the procedure subjective and not easily generalisable. Moreover, it's also interesting to dwell on that slight difference raised between the results of the structure without the weak element and the structure in which it was modelled as having mass equal to zero. The difference, measured in 2.19%, is explained forward in section 5.3.25.3.3.

Due to the reasons that have just been exposed, a more objective and automatic procedure has been developed, and is here reported. In this new procedure, the mass of each member is made proportional to its stiffness. In this way, very flexible members acquire automatically negligible mass, without the need for user classification, fulfilling Property n3 requirement. The procedure is illustrated in the next section (5.3.2).

5.3.2 AUTOMATIC STIFFNESS-BASED MASS ASSIGNMENT

A quantitative procedure has so been developed to formalise the idea of stiffness-based automatic mass assignment. This procedure aims is to ensure that the influence of each member of the structure on the robustness index reflects its real structural contribution, and not only its topological position in the connectivity network. This means that a relatively stiff member is supposed to have a high mass value, being it relatively important to structural stability. This approach intends so to make the metric sensitive to physical characteristics of members (such as material and section properties), while preserving the computational simplicity of the original implementation. In this respect stiffness indicators are defined as follows, to be then adopted to perform the automated mass assignment.

For every element i, two scalar stiffness indicators are first defined (equations 5.1 and 5.2):

$$K_{N,i} = \frac{E_i A_i}{L_i} \tag{5.1}$$

$$K_{M,i} = \frac{12E_i I_i}{L_i^3} \tag{5.2}$$

representing, respectively, the axial and the flexural stiffness of an Euler-Bernoulli beam with fixed-fixed boundary conditions. E_i, I_i, A_i and L_i are respectively elastic modulus, second moment of area, cross sectional area and length of member i. The first of the two terms, $K_{N,i}$, quantifies the axial stiffness, while the second the translational stiffness related to bending. Although the coefficient $K_{M,i}$ doesn't describe the full coupled flexural behaviour, which includes also a couple of other terms proportional to EI/L^2 and EI/L, and moreover it doesn't account for effects of end releases, it represents a lightweight and computationally easy tool to account for material, cross sectional inertia and member's length. This rough simplification is so consciously adopted for sake of research, while a more refined procedure could later be developed to include the full element stiffness matrix with the effects of end releases.

Each stiffness component is then normalised by its global reference value ($\widetilde{K}_N$ and $\widetilde{K}_M$), which is computed as the median across all members:

$$\widehat{K}_{N,i} = \frac{K_{N,i}}{\widetilde{K}_N} \tag{5.3}$$

$$\widehat{K}_{M,i} = \frac{K_{M,i}}{\widetilde{K}_M} \tag{5.4}$$

The use of the median rather than the mean is a project choice, taken to prevents distortion from extremely short or stiff elements. The two normalised quantities are then averaged to obtain a single, dimensionless stiffness score:

$$S_i = \frac{1}{2}(\widehat{K}_{N,i} + \widehat{K}_{M,i}) \tag{5.5}$$

Finally, the mass of each member is assigned by linear rescaling so that the stiffest element corresponds to a reference mass $m_{\max} = 10$:

$$m_i = m_{\max} \frac{S_i}{\max\limits_j S_j} \tag{5.6}$$

This formulation leads to the assignment of nearly zero mass to members with negligible stiffness, through an automatic process. This mass setting aims so to ensure that the inclusion or the removal of such a member does not influence the calculated robustness. The proposed method is tested in the next section (5.3.3).

5.3.3 VALIDATION OF THE AUTOMATED PROCEDURE

The current section aims to verify the proposed automatic stiffness-based mass assignment and to do that the same structural configuration illustrated in Figure 18 is analysed once again. Structural geometry is so the identical to the one used in previous test, while mechanical properties are this time made explicit by assuming real steel profiles for each member. In particular, "rigid" members were modelled as *IPE 500* sections, whereas the weak added element as a hot rolled round steel bar *R10* (10 mm of diameter), in order to highlight the strong contrast in stiffness, consciously extremizing the difference.

Material and section properties of each member were then collected in an Excel file for each structure (with and without the weak element). Those Excel files are then used for the Matlab routine. The table constituting the Excel file is reported below (Table 13) as an example (just remove the last row representing the weak element to get the table of the original structure):

Table 13 – Material and geometric properties of element, reference structure for Property n3

Beam number	E (MPa)	Length (m)	Inertia (cm^4)	A (cm^2)
1	210000	2	48017	115
2	210000	2	48017	115
3	210000	3	48017	115
4	210000	3	48017	115
5	210000	2	48017	115
6	210000	2	48017	115
7	210000	2	0.05	0.79

Using these data, the automatic stiffness-based procedure yielded the following mass distributions, respectively for the structure with and without the weak added element:

- [10.00; 10.00; 4.815; 4.815; 10.00; 10.00; 0.034]
- [10.00; 10.00; 4.815; 4.815; 10.00; 10.00]

The weakest member, corresponding to the round bar R10, acquired a mass value close to zero, while the most rigid members reached the upper bound, with the reference value of 10 ($m_{\max} = 10$). Members having an intermediate stiffness exhibit values consistent with their relative stiffness, characterized by the same order of magnitude of stiffer ones, confirming thus that the procedure actually captures the expected mechanical hierarchy among the various members. It's important to notice, at this regard, that the member length has a significant influence on the computed mass values, this behaviour is, by the way, physically meaningful and consistent with structural mechanics laws.

The robustness index was finally recalculated for the same structural configuration both with and without the weak element, using stiffness-based mass values, by using the original Matlab implementation. Obtained results were the following:

- $R = 0.481538$ for the original structure without the weak element;
- $R = 0.433449$ for the structure including the weak element.

Results show thus a difference of about 10 %, with the presence of the weak element that no longer increases the value of robustness, but instead produces a slightly lower result. This behaviour can be explained by how constant b has been defined, with its calibration that was formulated to weight constraints removal proportionally to the total number of constrained degrees of freedom. In particular, when a new element is introduced, the total number of constraints increases, making the same critical combination of removals producing a slightly different damage value, with obvious difference on the robustness index. In this specific case, that difference is quite high due to the

presence of just few elements and so few constraints (due to the well-known computational limits). For a more complex structure (as it's supposed to be a real one), that difference is expected to be lower.

To confirm the exposed interpretation, the same test was done on the same structural schemes, this time using a fixed value for constant *b*, which was set $b = 0.0037$, that is approximately the value assumed in the original configuration. This time the following results were obtained:

- *R = 0.428343* for the original structure without the weak element;
- *R = 0.430186* for the structure including the weak element.

Results show thus a very slightly difference this time, measured in just the 0.43%, with the higher value for the structure including the additional element. This little difference is attributable to the minimal resistance offered by the weak element, which has a certain stiffness, even if very low. In fact a small, but nonzero mass was assigned to the member, resulting in 0.034 (with 10 as the maximum value). Confirming the interpretation, the mass of the weak element was then set equal to zero, obtaining so a value of *R = 0.428343*, identical to the one of the structures without the additional element.

In summary, the definition of element masses as a function of their stiffness demonstrates a good potential, even beyond the verification of Property n3 itself. In fact, up to this point the mass definition has never been discussed, and element were designed as having all the same mass., without any link to a physical criterion, and this stiffness-based approach represent a possible rational solution. Nonetheless, the compatibility between this new formulation and the definition of constant *b*, offers scope for improvement. A possible refinement, aimed to eliminate that slight difference raised due to different values of *b*, could involve defining masses automatically by the just described procedure and then eliminate all elements having mass value under a certain threshold. Additional tests and parametric studies would ne necessary to assess the implications of such an approach, setting the threshold and so on

Finally, in the next section, the Matlab implementation of the mass calculation procedure is reported.

5.3.4 MATLAB IMPLEMENTATION

The procedure has been implemented in a dedicated MATLAB routine. The script, below reported, has been developed as to read initially a single Excel file that must contain, in the worksheet beams, one row per structural element with the following columns:

- BeamNo;
- E (Young's modulus in MPa);
- A (cross sectional area in cm^2);
- I (second moment of area in cm^4);
- L (element length in m).

Where units have been chosen for simplicity, being the most intuitive and most used for each property, even if they aren't consistent among each other. By the way, any other unit could have been chosen.

In fact, the normalization process makes unit choice irrelevant. Of course, what is important is to use the same for the various elements.

Once executed, the script computes the axial and flexural stiffness indicators for each element, normalising them just after by their median values. After that, it averages the two contributions, rescaling finally the resulting score so that the maximum value equals 10. In the meanwhile, a relative tolerance parameter τ is used to turn very small mass values automatically to zero. The output, in this version just defined as written in the command window, consists of a two columns array listing each *BeamNo* and its corresponding *mass*. These values can be then manually pasted into the Excel file used by the main robustness code.

In conclusion, the MATLAB implementation for the mass calculation is reported below.

```
clear; clc;

% === Excel file to be used ===
excelFile = 'name.xls';

% === Parametri ===
useMedian = true;   % true: median (recommended); false: mean
m max     = 10.0;   % assigned max for stiffest beam
tau       = 1e-6;   % lower bound to exclude very low mass values

%% === Data reading ===
% Columns: [BeamNo, E, L, I; A]
beam = xlsread(excelFile, 'beams');
BeamNo = beams(:,1);
E      = beams(:,2);
L      = beams(:,3);
I      = beams(:,4);
A      = beams(:,5);



%% === Stiffness indicators ===
KN = E .* A ./ L;            % axial stiffness   (EA/L)
KM = 12 .* E .* I ./ (L.^3); % flexural stiffness (12EI/L^3, proxy fixed-fixed)

%% === Global scale (median or mean) ===
if useMedian
    KN ref = median(KN);
    KM ref = median(KM);
    modeTxt = 'median';
else
    KN ref = mean(KN);
    KM ref = mean(KM);
    modeTxt = 'mean';
end

%% === Dimensionless indicators and their combination ===
SN = KN / KN ref;      % axial normalized
SM = KM / KM ref;      % flexural normalized
S  = 0.5 * (SN + SM);  % mean of the two values

%% === Normalized masses ===
```

```
Smax = max(S);
mass = m max * (S / Smax);
mass(S < tau * Smax) = 0;   % optional: to zero negligible values

mass out = [BeamNo, mass];

%% === Output ===
fprintf('\n--- Mass calculation ---\n');
fprintf('File: %s | Foglio: beams | Normalizzazione: %s\n', excelFile, modeTxt);
fprintf('KN ref = %.6g   KM ref = %.6g   (unità coerenti ai dati di input)\n',
KN ref, KM ref);
fprintf('m max = %.3g   tau = %.3g\n', m max, tau);
fprintf('--------------------------------\n');
for i = 1:numel(BeamNo)
    fprintf('Beam %g : SN=%.3f   SM=%.3f   -> mass=%.3f\n', ...
        BeamNo(i), SN(i), SM(i), mass out(i,2));
end
fprintf('--------------------------------\n\n');
```

5.4 Verification of Property 4 – Effect of Element Duplication on Structural Robustness

The fourth operative property states that the duplication of the existing elements of a structure should increase the robustness, with a limit case represented by the duplication of all the elements infinite times resulting in the maximum value for the index. It means that adding identical copies for each element and thus multiply its stiffness and resistance, should lead to an increase of robustness, since the duplication of elements offers new load paths. However, considering the logic of the metric presented in this work, the property is not expected to be satisfied. Indeed, the duplication of an element effectively increases the local stiffness and resistance, but it doesn't create any new link between nodes that were previously unconnected. Consequently, the topology of the structure remains theoretically unchanged and since the metric evaluates both the initial damage and the extent of the collapse as a fraction of the total structural mass (or fraction of constraints for the damage due to constraint removals), the duplication of elements is expected not to alter these ratios and so not to improve the robustness index.

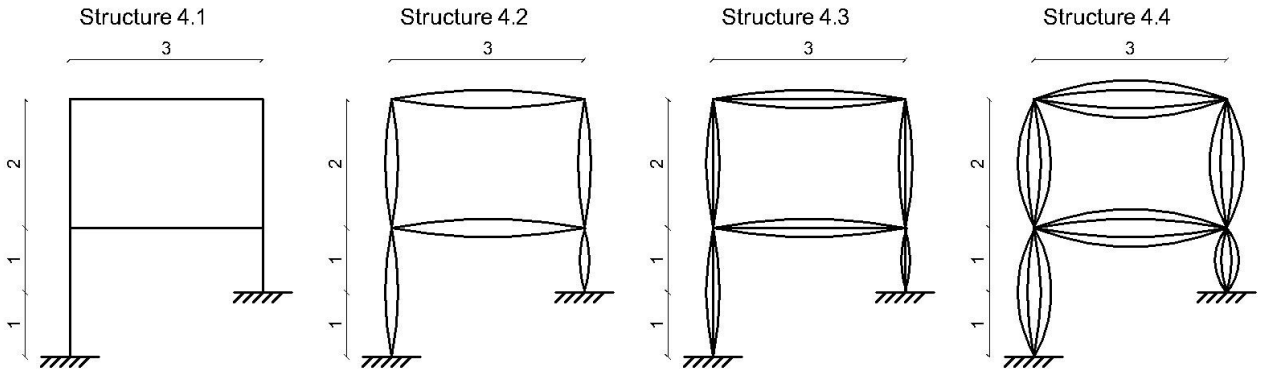


Figure 19 - Set of Reference structures for Property 4

Taking into account now the benchmark structures reported in Figure 19 - Set of Reference structures for Property 4Figure 19, in which all the elements are first duplicated, then triplicated and finally

quadruplicated, it's straightforward to consider that if in the original configuration (*Structure 4.1*), the removal of one specific member produce a certain collapse mechanism, the same mechanism would be triggered by the simultaneous removal of the elements substituting the initial one. Yet, since the damage magnitude is normalised by the total mass, the removal of the member and its duplicates would yield to an identical damage magnitude. In the same way, the scenario would be characterised by the mobilisation of the same mass fraction as before, leading to an analogue value for the inducted effects, even if the number of elements has been multiplied. Therefore, no significant variation of the robustness index is expected because of the duplication. A very similar argument can descend when considering constraint removals, although the relationship is more complex in this case. The damage associated to constraint removals is indeed weighted by constant *b*, which normalises the damage by the total number of constraints originally present (considering the type by mean of constant *c* and *d*). However, when elements are duplicated, the total number of constraints grows non-linearly, since for every duplicated element new constraint relations are written with respect of every connected element. Consequently, the total number of constraints is expected to grow faster then the number of elements, altering thus the relative weight of constraint removals. This non-linear growth can so change the critical scenarios that define the minimum robustness value at each step of duplication, making basically impossible to predict a trend.

To verify all those expectations that were collected, a set of tests has been carried out by the MATLAB implementation of the metric, with the integration of a genetic algorithm. Such algorithm has been introduced because of the computational limitation of the original version, which used to crash into computational burden already with about ten elements. So the aim of the genetic algorithm is to improve computational efficiency, identifying the critical damage scenario and so the index without exhaustively evaluating all possible damage combinations. The algorithm, still under development, will be fully described in the next chapter (6), but anyway its current version already allows for an exploration of the problem under analysis.

Results of the tests performed on the benchmark structures of Figure 19 are reported below:

- *Structure 4.1:* R = 0.4571;
- *Structure 4.1:* R = 0.4000;
- *Structure 4.1:* R = 0.8974;
- *Structure 4.1:* R = 0.8285.

The obtained results don't show a clear monotonic trend, with the robustness index initially decreasing and then increasing to later decrease again, suggesting an irregular and non-physical behaviour. These results confirm thus that the actual version of the metric don't comply with Property 4, due to its topological nature and due to the normalisation by total structural quantities as well. Nevertheless, it's relevant to report that the strange and apparently inexplicable oscillation may be partially influenced by the experimental nature of the genetic algorithm, which still presents some instability in identifying the true critical combinations, as it will be shown in the next chapter.

5.5 Verification of Property 5 – Influence of Adding a New Node and Connecting Element

The fifth operative property states that the addition of a new node and a new element connecting the new node with an existing one would reduce the robustness of the structure. In fact, trying to visualize such statement in practical terms, adding an appendage that doesn't improve structure stability nor structure redundancy or new load paths, should not result in a higher robustness index.

So, while the appendage isn't changing anything about load paths, such a new element results in the addition of a new portion of mass. Consequently, this new element is expected to influence the computation of robustness index, mainly through the quantities involved in the damage evaluation. In particular, the initial damage is affected by the increase of the total mass of the system, since it depends on the removed mass fraction m_h. For instance, the same initial damage consisting in the removal of a base column leads to a smaller damage value (and so a lower robustness) after having added an appendage, with the removed mass representing a smaller fraction of the total. At the same time, the activated mass fraction can increase as the added member can participate in the mechanism, for an equivalent kinematic motion, leading also to a lower robustness index value. Similar considerations can be done for the damage contribution coming from constraint removals. In fact, due to constant *b* normalization, the damage term is supposed to be lower for the removal of the same set of constraint when the added element is present. This because, with the presence of such a member, the number of total constraints increases, inducing the removed ones to result as a smaller fraction and so leading to a smaller damage magnitude.

In order to verify these considerations, two sets of numerical tests were performed using the usual Matlab implementation of the robustness metric. The first regards the pair of structures reported in Figure 20, which differ only by the addition of a lateral appendage in the second configuration.

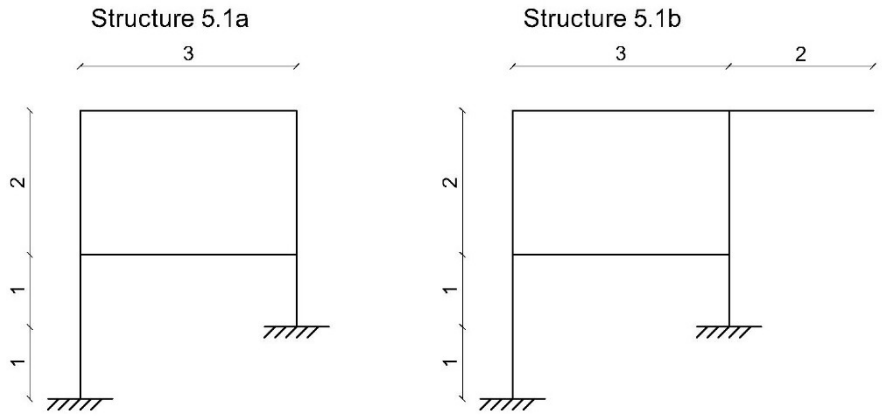


Figure 20 – 1st couple of reference structures for Property n5

From the calculations, the robustness indexes of the illustrated structures resulted in the following values:

- $R = 0.457143$ for structure 5.1a (without added element);
- $R = 0.379630$ for structure 5.1b (with the added element).

In both case the critical damage configuration was the same, including the removal of one element and one rotational constraint. As expected, the observed index difference can be attributable to the two reasons previously explained: the removed mass fraction in the initial damage term is smaller and the damage caused by constraint removal is lower because of the larger total number of constraints with the presence of the additional element.

To further examine the behaviour, and to have an additional proof that what was stated is true, a second comparison has been carried out on the pair of structures 5.2a and 5.2b, reported in Figure 21.

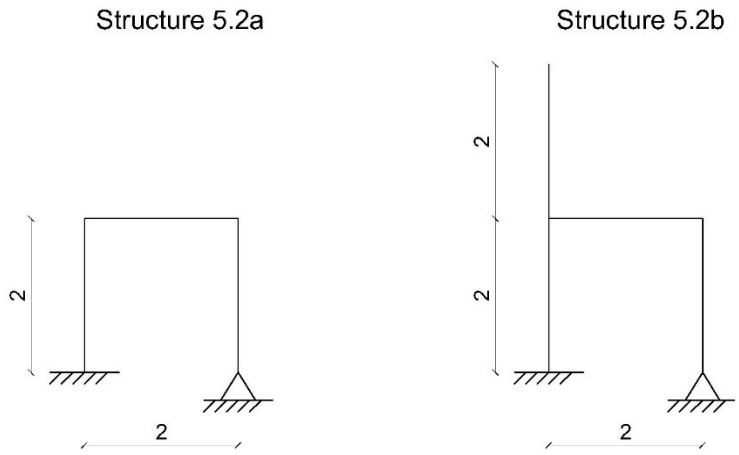


Figure 21 – 2nd couple of reference structures for Property n5

Once again, the structure with the additional appendage showed a lower robustness value, passing from R = 0.361446 for structure 5.2a to a lower value of R = 0.320000 for structure 5.2b. This last test confirms so once more the validity of the property. It's interesting and also important to highlight how the critical failure combination changed between these two configurations, moving from the removal of three rotational constraints to four of them. Analysing results deeper, looking at which constraints have been removed, it is possible to appreciate how the resulting collapse mechanism is physically identical in the two cases (both cases lead to the transformation of three built-in constraints – one external and two internal- into three hinges), but the program identifies one more constraint removal in the second structural scheme. This must be attributable to how kinematic matrix are constructed by the code: each connection between two elements introduces in fact an equation for every degree of constraint, so the additional member on the left is associated with three rotational, three horizontal, and three vertical constraint equations per each member connected with it. Consequently, one more rotational constraint has to be removed to reproduce the same mechanism as for the original structure.

To validate the explanation, an additional test has been carried out, consisting in calculate the robustness index for structures *5.2a* and *5.2b* with the constant *b* fixed to a reference value, as to eliminate the effect of normalization with respect to the total number of constraints. The constant *b* was so fixed *b=0.0120*, approximatively equal to the value assumed for the first scheme (*structure 5.2a*). Under these conditions, calculations result in *R=0.3600* and *R=0.4800*, for *structure 5.2a* and *structure 5.2b* respectively, still maintaining the same critical damage configuration as before (three

and four removed rotational constraints). These results showed off as the property isn't satisfied for a fixed value of *constant b*, confirming the soundness of its current definition. In fact when the constant is allowed to vary according to the total number of constraints and their nature, the metric correctly scales damage with respect to structure complexity, ensuring that results remain consistent through different structural types and sizes.

Overall, described tests confirm that the metric is consistent with Property n5, highlighting the importance and the goodness of the definition of constant b proposed in section 4.1.1

5.6 Verification of Property 6 – Independence of Robustness from Mesh Density

5.6.1 Effect of Mesh Density on the Robustness Metric

The sixth operative property states that the mesh density of the structural model should not influence the robustness of the structure. It means that the robustness index must depend only on structural topology, geometry, boundary conditions… and not on how the system is discretized in the numerical model. It follows that two structural models representing the same structure, one adopting a coarse mesh with one element per each member, while the other adopting a fine mesh with each member subdivided in several elements connected through rigid nodes, should exhibit the same robustness index.

However, the current Matlab implementation as well as the metric itself is not expected to fulfil this property. In fact, discretizing some elements into smaller parts opens new damage and mechanism options. Moreover, an equivalent initial damage in terms of effects, can be developed through the removal of only a piece of an element, while otherwise without the discretization the whole element would be removed, accounting for a different damage magnitude. This means that the initial damage, which is proportional to the fraction of removed mass, would be often smaller for equivalent scenarios, in case of discretized models. It's supposed to work similarly for constraints removal damages. In fact, discretization leads to an higher number of total constraints which makes a removal to be then weighted differently, resulting in a smaller damage value. Taking into account all these considerations, it appears that finer meshes are expected to artificially reduce the robustness index, even if the actual physical behaviour of the structure remains the same.

To verify such hypotheses, a set of tests has been carried out by using the Matlab implementation of the metric, testing the structural schemes reported in Figure 22. As it can be checked in the illustration, those models represent the same structure, where one or more elements have been subdivided into parts. The idea is so to study the variation of the index throughout the different setups.

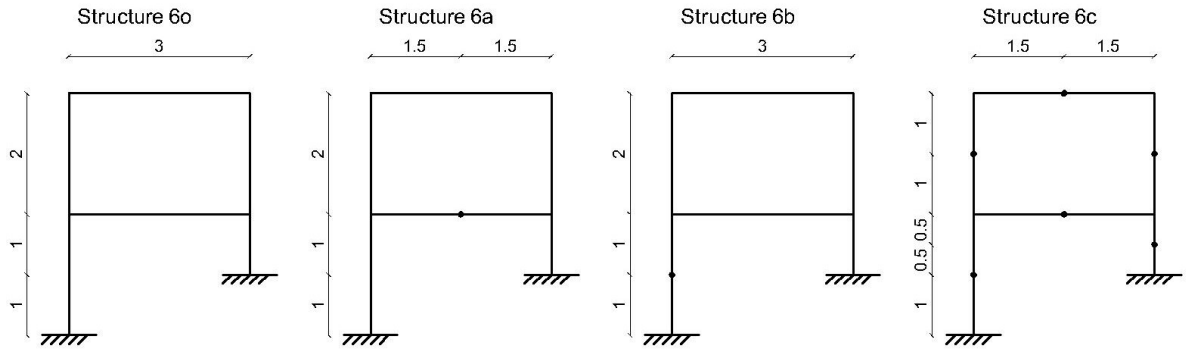


Figure 22 – Set of Reference structures for Property 6

Structure6o represents the original configuration, while *Structure6a, Structure6b* and *Structure6c* are theoretically equivalent schemes obtaining by the subdivision of one or more elements into smaller parts. The corresponding robustness indexes resulting from the Matlab implementation are reported below:

- *Structure 6o: R = 0.457143;*
- *Structure 6a: R = 0.407792;*
- *Structure 6b: R = 0.216450;*
- *Structure 6c: R = 0.119048.*

As expected, results confirm that the robustness index metric and its implementation are sensitive to mesh density, generally decreasing in robustness as the discretization level increases. In addition to the quantitative difference, some qualitative ones could also be found looking at the critical damage combination leading to the index definition. Indeed, different sets of removed elements and constraints have been found defining the minimum value of *R* among the different structures. This represents a further confirmation of how the metric perceive each differently discretized model as a structurally distinct system. This behaviour clearly contradicts Property6 statement.

To address this unwanted behaviour, a data pre-processing algorithm was developed, with the aim to automatically reconstruct the equivalent "undiscretized" structural model, before to evaluating robustness with the usual implementation. This operation can be particularly useful for structural models exported from Finite Element Method software, which typically divide each member into smaller portions, restoring the original structural scheme and ensuring compliance with Property6. This algorithm is reported and commented in the following section.

5.6.2 MATLAB IMPLEMENTATION OF THE PRE-PROCESSING ALGORITHM

As anticipated, the algorithm has been developed to automatically reconstruct the equivalent "undiscretized" structure prior to evaluating robustness. This procedure aims so to merge all consecutive collinear elements belonging to the same member, by removing internal mesh nodes connecting them and calculating merged properties of mass and length. To recognize elements supposed to be merged, parameter are the collinearity and the typology of internal connection that

has to be rigid (to not merge a couple of elements connected by an internal hinge for example). In this way internal released or any other significant joints are preserved.

Anyway, after an initial input processing and storage phase, the algorithm actually recognizes nodes to be eliminated first, by checking their degree (degree two nodes are supposed to be eliminated) and collinearity of connected elements, in addition to joint rigidity. After detection and elimination of nodes and elements, new members are created representing the merged ones, calculating overall values of properties that are supposed to be summed. It follows a renumerating phase in which elements and nodes number are redefined, along with some post-processing stages to guarantee cleanness or results. The algorithm ends exporting a new Excel file ready for the robustness analysis.

The script is reported below, with comments provided inline for clarity.

```
%% ------------------------------------------------------------
%  mesh elimination.m  (script-only, maximal chain merge)
%  Merge entire collinear chains separated by degree-2 internal nodes
%  (no supports and no end releases at the internal chain nodes).
%  Sums Length & Mass, carries end releases from chain ends,
%  renumbers nodes & beams, and writes "<input> undisc.xlsx".
%
%  Fixed sheets and columns:
%    topology   : [BeamNo, Node i, Node j, Angle, Length, Mass]
%    ext constr : [NodeNo, Rot, Htrans, Vtrans]      (1 = constrained)
%    int constr : [BeamNo, rot i, v i, w i, rot j, v j, w j] (1 = rigid)
%
%  Usage:
%    1) Set 'inputFile' below (same folder as this script)
%    2) Run the script
% ------------------------------------------------------------

clear; clc;

%% === INPUT FILE ===
inputFile  = 'struttura6a.xls';   % <- set here
outputFile = [erase(inputFile, {'.xlsx','.xls'}) ' undisc.xlsx'];

%% === Fixed settings ===
LEN COL     = 5;                    % Length column (topology)
MASS COL    = 6;                    % Mass column   (topology)
angleTolDeg = 0.5;                  % tolerance for collinearity [degrees]
ANGLE TOL   = deg2rad(angleTolDeg);

%% === Read sheets ===
Ttop = readtable(inputFile, 'Sheet','topology',   'ReadVariableNames',true);
Text = readtable(inputFile, 'Sheet','ext constr', 'ReadVariableNames',true);
Tint = readtable(inputFile, 'Sheet','int constr', 'ReadVariableNames',true);

%% === Extract columns ===
BeamNo = Ttop{:,1};
Ni     = Ttop{:,2};
Nj     = Ttop{:,3};
AngRaw = Ttop{:,4};      % may be degrees or radians
Len    = Ttop{:,LEN COL};
Mas    = Ttop{:,MASS COL};
```

```
Node = Text{:,1};
Rot  = Text{:,2};  Ht = Text{:,3};  Vt = Text{:,4};

Bin  = Tint{:,1};
R i  = Tint{:,2};  V i = Tint{:,3};  W i = Tint{:,4};
R j  = Tint{:,5};  V j = Tint{:,6};  W j = Tint{:,7};

%% === Detect angle units automatically ===
if nnz(abs(AngRaw) > 2*pi) > 0.2*numel(AngRaw)
    Ang = deg2rad(AngRaw);
    angUnit = 'deg';
else
    Ang = AngRaw;
    angUnit = 'rad';
end
fprintf('Angle unit detected: %s (tolerance = %.3f deg)\n', angUnit,
angleTolDeg);

%% === Helpers (anonymous) ===
wrapAng = @(x) atan2(sin(x),cos(x));
sameDir = @(a,b) (abs(wrapAng(a-b)) < ANGLE TOL) || (abs(abs(wrapAng(a-b))-pi) <
ANGLE TOL);
% Direction from a given node along a beam (away from that node)
dirFromNode = @(ang, beamNi, beamNj, nodeN) ang + (beamNj==nodeN)*pi;
% Rigid end check (uses Tint arrays by index)
isRigidEnd = @(idx, endTag) ...
    (endTag=='i' && (R i(idx)==1)&&(V i(idx)==1)&&(W i(idx)==1)) || ...
    (endTag=='j' && (R j(idx)==1)&&(V j(idx)==1)&&(W j(idx)==1));

%% === Identify supports ===
maxNode = max([Ni; Nj]);
isSupport = false(maxNode,1);
isSupport(Node( (Rot|Ht|Vt)~=0 )) = true;

%% === Main chain-merge loop ===
changed = true;
while changed
    changed = false;

    % Refresh local copies (topology may change)
    BeamNo = Ttop{:,1};
    Ni     = Ttop{:,2};
    Nj     = Ttop{:,3};
    Ang    = Ttop{:,4};
    Len    = Ttop{:,LEN COL};
    Mas    = Ttop{:,MASS COL};

    Bin  = Tint{:,1};
    R i  = Tint{:,2};  V i = Tint{:,3};  W i = Tint{:,4};
    R j  = Tint{:,5};  V j = Tint{:,6};  W j = Tint{:,7};

    % Build adjacency: node -> incident beam indices (rows of Ttop)
    uniqueNodes = unique([Ni; Nj]);
    incBeams = containers.Map('KeyType','double','ValueType','any');
    for r = 1:height(Ttop)
        a = Ni(r); b = Nj(r);
        if ~isKey(incBeams,a), incBeams(a) = []; end
        if ~isKey(incBeams,b), incBeams(b) = []; end
```

```
        incBeams(a) = [incBeams(a), r];
        incBeams(b) = [incBeams(b), r];
    end

    % Node degree
    deg = zeros(max(uniqueNodes),1);
    for n = uniqueNodes(:)'
        if isKey(incBeams,n)
            deg(n) = numel(incBeams(n));
        else
            deg(n) = 0;
        end
    end

    % Visit flags for beams
    visitedBeam = false(height(Ttop),1);

    % Try to form maximal chains starting from each unvisited beam
    for r0 = 1:height(Ttop)
        if visitedBeam(r0), continue; end

        % Initialize chain with beam r0
        chainBeams = r0;
        visitedBeam(r0) = true;

        % Current chain endpoints
        leftNode  = Ni(r0);
        rightNode = Nj(r0);

        % ---- Extend LEFT ----
        while true
            nodeN = leftNode;
            curr  = chainBeams(1);

            % conditions on nodeN: deg==2, not support
            if ~(nodeN>0 && nodeN<=numel(isSupport)), break; end
            if isSupport(nodeN), break; end
            if deg(nodeN) ~= 2, break; end

            % "other" beam at nodeN
            beamsAtNode = incBeams(nodeN);
            other = setdiff(beamsAtNode, curr);
            if numel(other) ~= 1, break; end
            b2 = other(1);
            if visitedBeam(b2), break; end

            % releases at nodeN for curr and b2
            endA = 'i'; if Nj(curr)==nodeN, endA='j'; end
            endB = 'i'; if Nj(b2)==nodeN, endB='j'; end
            idxA = find(Bin==Ttop{curr,1},1,'first');
            idxB = find(Bin==Ttop{b2,1},1,'first');
            rigidA = true; rigidB = true;  % default rigid if missing
            if ~isempty(idxA), rigidA = isRigidEnd(idxA,endA); end
            if ~isempty(idxB), rigidB = isRigidEnd(idxB,endB); end
            if ~(rigidA && rigidB), break; end

            % collinearity: direction from nodeN into curr vs into b2
            dirCurr = dirFromNode(Ang(curr), Ni(curr), Nj(curr), nodeN);
```

```
        dirB2   = dirFromNode(Ang(b2),   Ni(b2),   Nj(b2),   nodeN);
        if ~sameDir(dirCurr, dirB2), break; end

        % extend chain
        chainBeams = [b2, chainBeams]; %#ok<AGROW>
        visitedBeam(b2) = true;
        % new left node is the far end of b2
        leftNode = Ni(b2); if leftNode==nodeN, leftNode = Nj(b2); end
    end

    % ---- Extend RIGHT ----
    while true
        nodeN = rightNode;
        curr  = chainBeams(end);

        if ~(nodeN>0 && nodeN<=numel(isSupport)), break; end
        if isSupport(nodeN), break; end
        if deg(nodeN) ~= 2, break; end

        beamsAtNode = incBeams(nodeN);
        other = setdiff(beamsAtNode, curr);
        if numel(other) ~= 1, break; end
        b2 = other(1);
        if visitedBeam(b2), break; end

        endA = 'i'; if Nj(curr)==nodeN, endA='j'; end
        endB = 'i'; if Nj(b2)==nodeN, endB='j'; end
        idxA = find(Bin==Ttop{curr,1},1,'first');
        idxB = find(Bin==Ttop{b2,1},1,'first');
        rigidA = true; rigidB = true;
        if ~isempty(idxA), rigidA = isRigidEnd(idxA,endA); end
        if ~isempty(idxB), rigidB = isRigidEnd(idxB,endB); end
        if ~(rigidA && rigidB), break; end

        dirCurr = dirFromNode(Ang(curr), Ni(curr), Nj(curr), nodeN);
        dirB2   = dirFromNode(Ang(b2),   Ni(b2),   Nj(b2),   nodeN);
        if ~sameDir(dirCurr, dirB2), break; end

        chainBeams = [chainBeams, b2]; %#ok<AGROW>
        visitedBeam(b2) = true;
        rightNode = Ni(b2); if rightNode==nodeN, rightNode = Nj(b2); end
    end

    % If only one beam, skip
    if numel(chainBeams) < 2
        continue;
    end

    % ---- Create merged beam for the whole chain ----
    nA = leftNode; nB = rightNode;

    newLen = sum(Len(chainBeams));
    newMas = sum(Mas(chainBeams));

    % Angle coherent with nA -> nB using first beam of the chain
    bFirst = chainBeams(1);
    if Ni(bFirst) == nA
        newAng = wrapAng(Ang(bFirst));
```

```
        else
            newAng = wrapAng(Ang(bFirst) + pi);
        end

        % End releases from chain ends (default rigid if missing)
        bLast = chainBeams(end);
        ri=1; vi=1; wi=1; rj=1; vj=1; wj=1;

        idxL = find(Bin==Ttop{bFirst,1},1,'first');
        if ~isempty(idxL)
            if Ni(bFirst) == nA
                ri = R i(idxL); vi = V i(idxL); wi = W i(idxL);
            else
                ri = R j(idxL); vi = V j(idxL); wi = W j(idxL);
            end
        end

        idxR = find(Bin==Ttop{bLast,1},1,'first');
        if ~isempty(idxR)
            if Nj(bLast) == nB
                rj = R j(idxR); vj = V j(idxR); wj = W j(idxR);
            else
                rj = R i(idxR); vj = V i(idxR); wj = W i(idxR);
            end
        end

        % Append merged beam
        newBeamNo = max(Ttop{:,1}) + 1;
        newRow = Ttop(1,:); newRow{:,:} = NaN;
        newRow{1,1} = newBeamNo;
        newRow{1,2} = nA;
        newRow{1,3} = nB;
        newRow{1,4} = newAng;
        newRow{1,LEN COL}  = newLen;
        newRow{1,MASS COL} = newMas;
        Ttop = [Ttop; newRow]; %#ok<AGROW>

        % Append merged internal constraints (rigidities at ends)
        Tint = [Tint; {newBeamNo, ri,vi,wi, rj,vj,wj}]; %#ok<AGROW>

        % Compute internal nodes (from current chainBeams) BEFORE deletion
        chainNodesAll = [];
        for bb = chainBeams
            chainNodesAll = [chainNodesAll; Ni(bb); Nj(bb)]; %#ok<AGROW>
        end
        internalNodes = unique(chainNodesAll(chainNodesAll~=nA &
chainNodesAll~=nB));

        % Delete all beams in chain from tables
        beamsToRemove = Ttop{chainBeams,1};
        Ttop(ismember(Ttop{:,1}, beamsToRemove),:) = [];
        Tint(ismember(Tint{:,1}, beamsToRemove),:) = [];

        % Remove internal chain nodes from ext constr
        if ~isempty(internalNodes)
            Text(ismember(Text{:,1}, internalNodes),:) = [];
        end
```

```
        % IMPORTANT: we modified Ttop/Tint/Text while iterating over r0.
        % Restart the outer while-loop to rebuild indices and adjacency.
        changed = true;
        break;  % exit the for r0 loop; while 'changed' will iterate again
    end
end

%% === Final cleanup: keep only consistent rows ===
usedNodes = unique([Ttop{:,2}; Ttop{:,3}]);
usedBeams = unique(Ttop{:,1});
Text = Text(ismember(Text{:,1}, usedNodes), :);
Tint = Tint(ismember(Tint{:,1}, usedBeams), :);

%% === Compact renumbering ===
% Nodes
mapNode = zeros(max(usedNodes),1);
mapNode(usedNodes) = 1:numel(usedNodes);
Ttop{:,2} = arrayfun(@(x) mapNode(x), Ttop{:,2});
Ttop{:,3} = arrayfun(@(x) mapNode(x), Ttop{:,3});
Text{:,1} = arrayfun(@(x) mapNode(x), Text{:,1});

% Beams
oldBeam = Ttop{:,1};
[~,~,newIdx] = unique(oldBeam,'stable');
Ttop{:,1} = newIdx;
% Map Tint beam numbers via first-occurrence mapping
mapBeam = containers.Map('KeyType','double','ValueType','double');
for k = 1:numel(oldBeam)
    if ~isKey(mapBeam, oldBeam(k))
        mapBeam(oldBeam(k)) = newIdx(k);
    end
end
for k = 1:height(Tint)
    b = Tint{k,1};
    if isKey(mapBeam, b)
        Tint{k,1} = mapBeam(b);
    end
end

%% === Angle normalization to canonical positives (no endpoint swap) ===
tol zero = 1e-6;   % ~0
tol pi   = 1e-3;   % ~pi
tol v    = 1e-3;   % ~vertical

Ang = Ttop{:,4};
for r = 1:height(Ttop)
    a = Ang(r);
    a = atan2(sin(a), cos(a));   % wrap to [-pi, pi]
    if a < 0, a = a + pi; end    % map to [0, pi)
    if a >= pi, a = a - pi; end
    if (abs(a) < tol zero) || (abs(a - pi) < tol pi), a = 0; end   % horizontals
-> 0
    if abs(a - pi/2) < tol v, a = pi/2; end                        % verticals
-> +pi/2
    Ang(r) = a;
end
Ttop{:,4} = Ang;
```

```
%% === Report & write ===
fprintf('\n--- Mesh elimination report (chain-merge) ---\n');
fprintf('Remaining beams: %d | Remaining nodes: %d\n', height(Ttop),
numel(unique([Ttop{:,2};Ttop{:,3}])));

fprintf('\nWriting cleaned Excel: %s\n', outputFile);
writetable(Ttop, outputFile, 'Sheet','topology',   'WriteVariableNames',true);
writetable(Text, outputFile, 'Sheet','ext constr', 'WriteVariableNames',true);
writetable(Tint, outputFile, 'Sheet','int constr', 'WriteVariableNames',true);
fprintf('Done.\n');
```

5.7 Verification of Property 7 – Impact of Increasing Degrees of Freedom on Robustness

The seventh operative property states that an increase in the number of degrees of freedom of a structure should lead to a decrease in its robustness. This statement sounds absolutely logical, since eliminating some degrees of constraint means to reduce redundancy and so somehow reduce the "defences" of the structure against progressive collapse by reducing the possible alternative load paths. In other words, adding degrees of freedom makes the structure less capable to limiting the propagation of collapse mechanisms, and so the failure of a single member becomes more likely to trigger a mechanism involving a larger portion of the system. Although all this seems completely logical, it's possible that the removal of some degrees of constraint don't change the structural behaviour under specific conditions, in that case an equivalent critical combination can lead to equal or even higher values of robustness (higher if the combination include constraint removals, which would lead to a lower damage magnitude if the total number of degrees of constraint is lower). However, the general trend is expected to respect the property, with a decrease in the robustness index as a result of degrees of constraint removals. By the way, those are just logical expectations, that have to be proven by numerical analysis.

So, to verify what has been said so far, a set of numerical test were performed, using the same structural scheme analysed in the previous section as a reference structure. Starting from that original model (*Structure7o*), several configurations were generated by removing one or more degrees of freedom, either rotational or translational, obtaining the structural schemes illustrated in Figure 23.

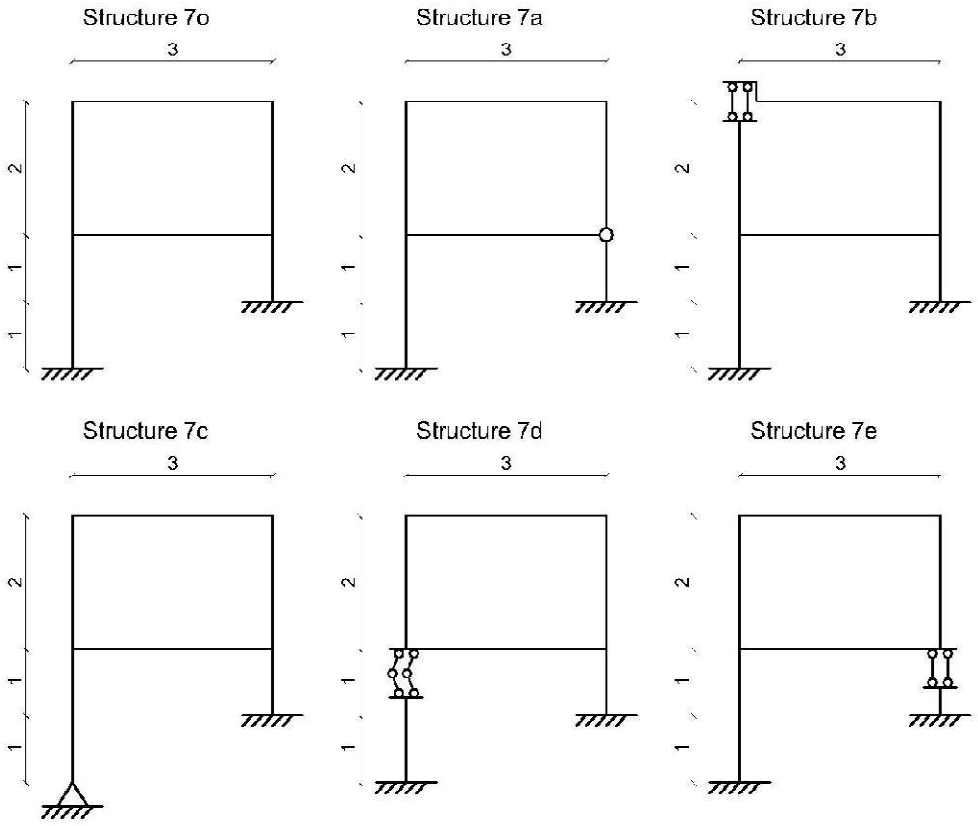


Figure 23 – Set of Reference Structures for Property 7

The illustrated structures has been tested by using the Matlab implementation of the metric, obtaining the robustness indexes reported below:

- *Structure 7o: R = 0.457143;*
- *Structure 7a: R = 0.193237;*
- *Structure 7b: R = 0.300000;*
- *Structure 7c: R = 0.239234;*
- *Structure 7d: R = 0.117647.*
- *Structure 7e: R = 0.189474*

Results confirm the expected trend, with a full compliance of results with the property. Indeed, the reduction in constraint stiffness exposes the structure to larger collapse mechanisms even for smaller initial damages, as reflected by the decrease in the robustness index. Despite the simplicity of the analysed structural schemes, the tests performed validate the soundness of both the Matlab implementation and the metric itself, being this property fundamental in the robustness metric logic. Nevertheless, the same trend is expected to hold for more complex structures, where the removal of degrees of constraint would as well degrade the system capacity to withstand local failures and prevent progressive collapse.

5.8 Summary and Discussion

The analyses reported in this chapter have tested the proposed metric against the set of operative properties defined by Professor Valerio De Biagi with the aim of proving the consistency of a robustness metric. Each of those seven properties represents thus a distinct aspect of structural behaviour that a meaningful and reliable robustness metric should reproduce in its outcomes. Therefore, assessing the correlation between expected theoretical response and numerical results allowed to check the degree of coherence of the metric and to identify the aspects requiring refinement.

About Properties 1 and 2, corresponding respectively to statically determinate and externally static determinate structures, the robustness index in its original form didn't return a null value as suggested by the Properties. The metric, because of its nature, returned very small values, but still larger than zero. In fact a statically determinate structure would certainly experience a collapse mechanism involving all of its members, but the damage magnitude would surely be larger than zero (even for a single constraint removed), yielding to positive values of the index. Therefore, to comply with the Properties, thus reproducing a null robustness for such structures, a little modification was introduced in the Matlab script, imposing the automatic assignation of $R = 0$ in case of a mechanism involving the entire structure and triggered by the removal of a single constraint. Such adjustment aligns metric outcomes with the theorized expectations of Properties 1 and 2.

About Property 3, which concerns the insensitivity to the addition of elements with negligible stiffness, the first tests showed that the original formulation of the metric violates it. In fact, it considers just topological aspects, without accounting for mechanical and material properties. A suitable solution aimed to avoid this kind of error was found introducing an automatic stiffness-based mass assignment. In this way the metric assigns very small or null mass to members technically unable to carry significant loads, ensuring thus that the removal of such an element doesn't affect damage magnitude and so the computed robustness. This correction finally restored compliance with Property 3, providing also a grounded framework for defining element masses.

Property 4, which regards an ideal increasing in robustness as results of the duplication of elements was instead not satisfied. In fact, duplicating each element increases stiffness and strength of each one, but without creating any new links or connections. So, since the index depends on damage and mechanism values which are both normalised with respect to total mass and total constraint count, the robustness value does not exhibit a consistent growing trend with element duplication. Results obtained using a genetic algorithm to speed up calculations, showed irregular variations of R, confirming that the current metric is insensitive to purely local redundancy increase, if such variation is done uniformly on the whole structure.

Property 5, which considers the effect of adding "secondary appendages" to the structure, as a new element with a new node, states that this should lead to a decrease in robustness. Tests confirm the expectation, with the inclusion of a new weakly connected element resulting systematically in a decrease of the robustness index, due to the changes in the total mass and basically the consequent lower damage magnitude for an equivalent scenario.

Property 6, which affirms a need for insensitivity of the metric from mesh density, was definitely not respected by results. Refining the mesh causes the decrease of the robustness index, changing the critical damage combination, confirming thus how the original implementation treated discretized models as topologically different systems. Basically, discretizing an element in ten pieces, make possible to reach an equivalent damage by removing one tenth of an element instead of the whole one, resulting in lower damage values and lower robustness index too. Therefor, to overcome this limitation, a pre-processing algorithm was developed to automatically merge collinear element chains into equivalent single members. The procedure so basically de-discretises structural models, eliminating the mesh and restoring the equivalence between differently meshed representation of the same structure. In this way the pre-processing step actually ensures compliance with the property.

Finally, the last one, Property 7, which aims to investigate the influence of increasing degrees of freedom, confirmed the expected trend, whereby the removal of an internal or external constraint results in a lower robustness value. Therefore, the metric reflects the loss of structural redundance in the index results, which means a higher susceptibility of the system to collapse mechanism involving larger portion of the structure.

In conclusion, the overall verifications confirms that the proposed metric, after some targeted refinements, exhibits a consistent and physically meaningful response. A part for the aspects related to element duplication, which require further theoretical development, the metric has demonstrated a robust and reliable performance in representing important structural behaviour.

The next chapter will explore the genetic algorithm introduced in 5.4, developed to enhance computational efficiency, solving the computational burden problem related to complex structural systems.

6 GENETIC ALGORITHM

6.1 Introduction

The computational framework developed in the previous chapters relies on the complete evaluation of all possible damage combinations, obtained by removing one or more item among elements and constraints from the structure. While this kind of approach ensures exhaustiveness and accuracy, its computational cost increases exponentially with the complexity of the structure (with the number of elements and constraints), becoming prohibitive very quickly and resulting thus in computational burden already for medium-scale system. Even for moderate structural systems, the enumeration of all damage scenarios requires thousands of millions of iteration, including the calculation of kinematic matrixes, the null space and so on. Therefore, the high computational demand makes the procedure inefficient and unsuitable for design applications. To overcome this limitation, an optimization solution is proposed, based on Genetic Algorithms. The method aims to preserve the accuracy of the robustness metric, while drastically reducing the computational cost. Instead of testing every single combination, the algorithm explores the solution space adaptively, searching the most critical damage scenario through an evolutionary learning process.

This chapter presents thus the implementation of a Genetic Algorithm-based optimization for the robustness metric. The chapter will be developed with the following sections:

- An overview of the main theoretical principles of genetic algorithms and their relevance for computational optimization;
- The detail of the application of such optimization in the Matlab implementation of the robustness metric;
- A comparison between results got with the original and the optimized implementation, as to validate the accuracy of the method through a correlation of results. Also efficiency will be tested, through computational times and modification/integrations will be eventually proposed.

6.2 Overview of Genetic Algorithms

Genetic Algorithms (GAs) are stochastic search and optimization techniques inspired by the mechanisms of natural evolution. As described by several well-recognized text books (Mitchell, 1996) (Haupt & Haupt, 2004), they belong to the broader class of evolutionary algorithms, which simulate the processes of selection, reproduction, and mutation of biological populations. Their basic idea is to generate initially a population of candidate solution and then evolve it towards better approximations of the optimal solution until reaching a convergence. The process governing the approximation and the search for the optimal solution mimics natural selection.

6.2.1 BASIC CONCEPTS

Genetic algorithms operate on population of individuals which represent each one a possible solution of the problem. Typically, every individual is called *chromosome* and is encoded as a sequence of *genes,* which represent the decision variables of the problem. The possible gene "settings" are called

alleles (Mitchell, 1996). In the most common formulation, chromosomes are represented by binary strings, even if they do exist some other encodings, such as real valued, integer or symbolic representations. In a bit string an allele can be either 0 or 1.

The algorithm starts creating an initial population, usually created randomly to ensure diversity. After that, every chromosome is evaluated through a fitness function, which measures the quality of the solution that it represents. Therefore, the fitness function is the quantitative criterion that drives the evolutionary process. That means that individuals with higher fitness are considered more "adapted" and are more likely to pass their traits to the next generation. After the initial evaluation, new generation are produced iteratively applying reproduction and variation to the current population. Alternative solutions are so explored, while keeping the high-quality traits. Generation by generation, selection pressure gradually shifts the population towards fitter regions of the solution space. During this stage, random variation preserves diversity and reduces the risk of premature convergence. This process continues until meeting a stopping criterion, which can typically be represented by a maximum number of generations, a target fitness level or a lack of improvement over a certain number of iterations. Meeting so a stopping criterion, the best individual found so far is returned as the algorithm solution (Haupt & Haupt, 2004).

6.2.2 Evolutionary Operators

Three genetic operators govern the evolution of the population, through their repeated application. Those operators are: selection, crossover and mutation (Mitchell, 1996). They're briefly described below:

- Selection represents the process used to chose individuals to become parents, considering their fitness features. There are several selection methods reported in the literature (Haupt & Haupt, 2004), such as random pairing, roulette-wheel selection, tournament selection, weighted random pairing. The aim of these methods is to ensure that fitter individuals have more probability to reproduce, while still guaranteeing for diversity, by allowing less fit individuals a chance to contribute.
- Crossover (or mating) is considered by Mitchell (Mitchell, 1996) as the main distinguishing feature of a genetic algorithm. It is the operator in charge to produce new individuals by exchanging genetic material between two parents chromosomes. Those new individuals are called *offsprings*. Crossover enables thus the combination of successful features from different individuals, representing the main mechanism for exploiting information gathered during the search. It can be performed at a single or at multiple points along the chromosome, depending on the chosen implementation.
- Mutation is the operator responsible to introduce random changes into the genetic material of individuals. In case of binary string, it typically operates by flipping one or more bits, while in case of real-values representations it adds small perturbations. The aim of this operator is to preserve genetic diversity within the population and to prevent the algorithm from converging prematurely to local optima missing the correct solution.

6.2.3 ALGORITHMIC FLOW

Usually, genetic algorithms follow the procedure reported below (Haupt & Haupt, 2004):

1. Initialization: the first stage consisting in the generation of the initial population of candidate solutions, which can be done according to different methods;
2. Evaluation: in the second step the algorithm computes the fitness of each individual using the fitness function;
3. Selection: at this point individuals to be reproducted are chosen according to their fitness;
4. Crossover and Mutation: those two parameters together are the ones used to create the new population of individuals in the fourth stage;
5. Convergence check: the last step is a decisional point, which lead to exit in case of meeting a stopping criterion, otherwise the algorithm restart from stage 2.

As the algorithm continues to iterate along these steps the quality of solution grows progressively. The termination conditions can consist in a maximum number of iterations, a desired fitness threshold or the absence of significant improvement over several iterations.

The algorithm is illustrated in detail in Figure 24.

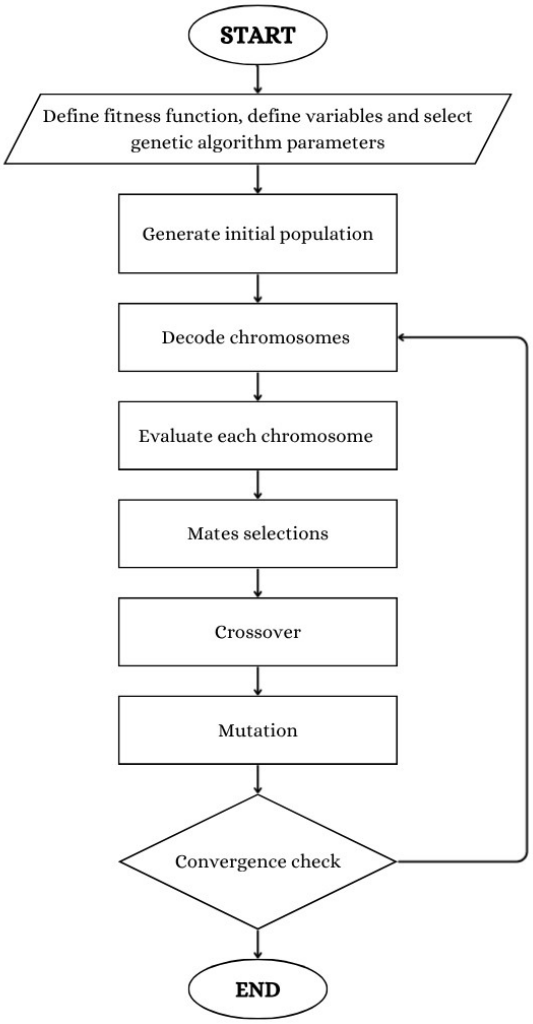


Figure 24 – Flowchart of a typical genetic algorithm

6.2.4 SUMMARY, KEY FEATURES AND ADVANTAGES

Genetic algorithms are a very powerful tool for optimization purposes, in particular for complex, non linear or discontinuous problems, where gradient-based methods struggle. As they rely on probabilistic rules, rather than explicit derivates, they can deal with noisy, multi-modal or analytically intractable fitness functions. Moreover, the strategy of starting with a population of solutions allows for the exploration of multiple areas of the search space simultaneously, increasing the likelihood of getting the actual solution (the global optima) (Mitchell, 1996). However, to use efficiently genetic algorithms, it's important to pay attention to parameter tuning. In fact, parameters such as population size, crossover and mutation probabilities, and stopping criteria influence significatively convergence speed and accuracy. Haupt and Haupt highlight in their publication (Haupt & Haupt, 2004) the importance of a correct balance between exploration and exploitation to achieve stable and efficient convergence, where the first is maintained by mutation and the latter is driven by selection and crossover.

Concluding, genetic algorithms are a very flexible and robust optimization tool. They search for optima not following a single path, but instead starting from different initial points, which are the individuals of the original population. Then applying evolutionary principles of selection, inheritance and variation, they simultaneously analyse the evolution of the whole population towards the optimal solution. Because of its stochastic nature, results can differ across different runs (if population generation, selection, crossover and mutation actually have a random nature), but a good parameter tuning allows to navigate large, complex and discontinuous search up to find the true optimal. Their flexibility and efficiency features, make them an important tool widely utilized in context such as engineering, economics, computer science and many other fields.

6.3 Robustness metric MATLAB implementation with a Genetic Algorithm

This section purpose is to present the new Matlab implementation of the robustness metric, in which a genetic algorithm replaces the exhaustive combinatorial search. The aim of this section is so to illustrate how the genetic algorithm is configured and invoked, showing as well as the existing computational kernel is still present without altering the metric computation and logic.

In the beginning of the script there's not so much to highlight, since the pre-processing phase remain identical to the original pipeline: structural data are taken from the Excel file, kinematic matrix is assembled, storing constraint and element data in the memory, and finally computing constants. The only addition in the first part of the code is a line calling *global handles*, just at the very beginning of the code as reported below.

```
global constants kinematic_matrix massMatrix ConstraintTypes noConstr noEle;
```

After the first part of data reading and constants definition, it comes the genetic algorithm setup phase, which is reported below. In this section the encoding of individuals in carried out. Each candidate is thus encoded as a bitstring of length $n = noConstr + noEle$. In this framework, the first *noConstr* bits indicate whether each constraint is kept (0) or removed (1), while the remaining *noEle* bits

indicate whether each element is kept (0) or removed (1). This encoding method places the genetic algorithm directly on the discrete combinatorial space that the original implementation used to explore by enumeration of all possible solutions, via nested calls to *nchoosek* across three blocks ("only constraints", "mixed", "only elements"). The block which substitutes though the entire exhaustive enumeration with a single evolutionary search is so reported below. After that some further details will be explored (MathWorks Inc, 2025).

```
% decision vector: [constraint bits | element bits]
n      = noConstr + noEle;
intcon = 1:n;

rng default

PopulationSize       = 10000;
MutationFcnScale     = 1;
MutationFcnShrink    = 1;
MaxStallGenerations  = 40;
CrossoverFraction    = 0.8;
FunctionTolerance    = 0.1;
Generations          = 500;

options = optimoptions('ga', ...
  'PopulationType','bitstring', ...
  'UseVectorized', false, ...
  'UseParallel',   false, ...
  'PopulationSize',PopulationSize, ...
  'PlotFcns',{@gaplotbestf,@gaplotstopping,@gaplotdistance,@gaplotrange}, ...
  'MutationFcn',{@mutationgaussian MutationFcnScale MutationFcnShrink}, ...
  'MaxStallGenerations',MaxStallGenerations, ...
  'CrossoverFraction',CrossoverFraction, ...
  'FunctionTolerance',FunctionTolerance, ...
  'Generations',Generations);
```

Still about the encoding process, the line *intcon = 1:n* sets out that every gene has to be represented by an integer, while the option *'PopulationType','bitstring'* is used to ensure binary nature of these variables, meaning that alleles must be just 0 or 1.

The variable called *PopulationSize* define the number of individuals of each generation. Of course an higher number implies deeper exploration but, on the other hand, larger computational time.

Parametric mutation variables *MutationFcnScale* and *MutationFcnShrink* refer to the mutation method *@mutationgaussian*, identified inside the *options* toolbox. *MutationFcnScale* define the initial amplitude of the random perturbation (the standard deviation of the perturbation), so larger values identify coarser mutations, while smaller ones identify finer mutations. *MutationFcnShrink* defines instead the coefficient of progressive mutation reduction, which means that a value larger than 1 leads to a more delicate mutation in time, while a value equal to one keeps the mutation amplitude constant. Anyway, as reported in 6.4, this mutation method will result not ideal for bitstring chromosomes (defined by 0 and 1 only), to which the just reported parameters don't have a real impact (MathWorks Inc, 2025). For this reason *@mutationgaussian* will be substitute with *@mutationuniform,* an uniform bitflip mutation mode.

The *CrossoverFraction* represents instead the percentage of individuals generated by combination of selected parents (MathWorks Inc, 2025). The remaining part (20% in this case) is generated by mutation or by keeping elite individuals. A larger value for *CrossoverFraction* means thus to leverage more on combinations already considered good, while a lower value leaves more space for exploration.

Finally, *MaxStallGenerations* together with *FunctionTolerance,* and *Generations*, define the two algorithm exit criteria. These are important for achieve the right balance between quality and efficiency.

The genetic algorithm is then invoked with a minimal fitness wrapper which maps the constraints and elements removal sets, delegating the metric evaluation to the existing core function. Then the script decodes the best chromosome, separating elements and constraints indices and recomputing the Robustness index for completeness of output. After having interpreted the chromosomes, the fitness wrapper calls the unchanged structural kernel. These steps are reported below.

```
[x, Fval, exitFlag, Output] = ga(@ga resilience function, n, [], [], [], [], [],
[], [], options);
ConstraintsPointer = logical(x(1:noConstr));
ElementsPointer    = logical(x(noConstr+1:end));

ConstraintsList    = 1:noConstr;
ElementsList       = 1:noEle;

removedConstraint  = ConstraintsList(ConstraintsPointer);
removedElement     = ElementsList(ElementsPointer);

Ri = ResilienceIndex(kinematic matrix, massMatrix, ConstraintTypes, ...
                     constants, removedConstraint, removedElement)
function Ri = ga resilience function(x)
  global constants kinematic matrix massMatrix ConstraintTypes noConstr noEle

  ConstraintsPointer = logical(x(1:noConstr));
  ElementsPointer    = logical(x(noConstr+1:end));

  removedConstraint  = find(ConstraintsPointer);
  removedElement     = find(ElementsPointer);

  Ri = ResilienceIndex(kinematic matrix, massMatrix, ConstraintTypes, ...
                       constants, removedConstraint, removedElement);
end
```

the computational core remains the same as it was, applying removals to kinematic matrixes, calculating null spaces to evaluate mechanisms and finally computing the index. Anyway the signatures are reported below for completeness.

```
function Ri = ResilienceIndex(kinematic matrix, massMatrix, ConstraintTypes, ...
                              constants, removedConstraint, removedElement)
  % ... (unchanged)
end

function [kinematic matrix, nodesId, massMatrix] =
kinematic_matrixFunction(nomefile)
```

```
  % ... (unchanged)
end

function [n, b, top, constrain, internal] = File2Str KIN(StrFile)
  % ... (unchanged)
end
```

In conclusion, this new approach, as mentioned, replaces the previous one, which involved enumerating each possible combination through a search that focuses the attempts on the regions of the solution space where the minimum values are best found. It is therefore expected to obtain the same values as those obtained with the original implementation, as numerical consistency is guaranteed by nature; the only condition to be centered in order to obtain this consistency is that the algorithm actually reaches convergence, without finishing before encountering the optimal solution.

In the next section, the method just described will be tested, evaluating its correlation with the previously obtained results for total enumeration. Subsequently, the efficiency of the method will also be evaluated, considering the calculation times.

6.4 Validation of the genetic algorithm: accuracy and runtime

The genetic algorithm is validated in this chapter using the same 15 reference structures adopted in the previous chapter, for which the robustness index was obtained yet by exhaustive enumeration. The goal of this test is twofold: to evaluate how closely the genetic algorithm is with respect to the enumerative reference and to quantify the computational savings it allows. If a good correlation of results won't be found, appropriate improvements will be proposed and further tests will be performed.

Table 14 - Comparison between the enumerative reference and the Genetic Algorithm results obtained with the original parameter configuration

Structure	Enumerative	GA (baseline)	Relative error (%)
Str_1	0.1389	0.1389	0.00
Str_2	0.1010	0.1010	0.00
Str_3a	0.2857	0.4571	59.99
Str_3b	0.5478	0.8000	46.04
Str_5.1a	0.2857	0.4571	59.99
Str_5.1b	0.3796	0.3796	0.00
Str_5.2a	0.3614	0.3614	0.00
Str_5.2b	0.3200	0.3200	0.00
Str_6a	0.4078	0.4519	10.81
Str_6b	0.2164	0.2290	5.82
Str_7a	0.1932	0.1932	0.00
Str_7b	0.3000	0.4600	53.33
Str_7c	0.2392	0.4000	67.22
Str_7d	0.1176	0.1176	0.00
Str_7e	0.1895	0.1895	0.00

Table 14 reports the results of the first test. A discrepancy is found in seven out of fifteen cases, with errors ranging from small to very significative. Considering all the 15 structures, the average relative error is about 20%, with a maximum relative error of 67% (*Str_7c*). Considering only the mismatched

cases, the median relative error is about 53%. Those numbers indicate that when the genetic algorithm does not detect the actual optima, it can do so by a non-negligible margin.

The results just discussed are clearly not satisfactory and for this reason following modifications are proposed in an attempt to achieve a better correlation. Initially, parameter modification strategies are sought to achieve better exploration of the solution space, reducing the risk of premature convergence. In particular, the crossover fraction is reduced, in such a way as to ensure greater diversity within the population. To the same end, the mutation rate is slightly increased, which is also reformulated as a bitflip mutation and no longer Gaussian. This modification should allow for a deeper exploration of alternative regions of the solution domain, thus avoiding missing the optimal solution as often happened in the previous test. An elitism strategy is also introduced to ensure that individuals with the best fitness characteristics are maintained from one generation to the next, thus preventing the loss of good solutions due to stochastic effects. Finally, a lower functional tolerance and a greater maximum number of generations were introduced, in order to avoid premature convergence.

Table 15 - Comparison between the enumerative reference and the Genetic Algorithm results obtained after parameter tuning (modified configuration)

Structure	Enumerative	GA (modified)	Relative error (%)
Str_1	0.1389	0.1389	0.00
Str_2	0.1010	0.1010	0.00
Str_3a	0.2857	0.4571	59.99
Str_3b	0.5478	0.8000	46.04
Str_5.1a	0.2857	0.2857	0.00
Str_5.1b	0.3796	0.3796	0.00
Str_5.2a	0.3614	0.3614	0.00
Str_5.2b	0.3200	0.3200	0.00
Str_6a	0.4078	0.4519	10.81
Str_6b	0.2164	0.2519	16.40
Str_7a	0.1932	0.4580	137.06
Str_7b	0.3000	0.4600	53.33
Str_7c	0.2392	0.4000	67.22
Str_7d	0.1176	0.1176	0.00
Str_7e	0.1895	0.1895	0.00

As reported in Table 15, these parameter changes did not yield the desired results. In particular, the new parameters corrected a previously incorrect case, but several errors remained the same as before, and in some cases, the percentage errors were even worsened. Considering all the reference structures, the average relative error reached approximately 26% while the relative error maxes out at 137%, thus showing a deterioration for both parameters. These results suggest that parameter tuning alone is not sufficient to achieve an acceptable level of convergence. Considering the stochastic nature of the algorithm, it is therefore decided to investigate the influence of the initial population generation on the final result.

Due to the stochastic nature of the method, the initial population, as well as the mutation and crossover operations, depend on a random number generator. However, as the generator defaults, the same initial population is always obtained between runs, so to evaluate the variability of the results with respect to the initial population, new seeds must be introduced. The seed is an initial value used by the pseudorandom number generator to produce a sequence of numbers that appear random, but are actually deterministic: for the same seed, the generator will always return the same sequence of

numbers (MathWorks Inc, 2025). Therefore, in this case, an attempt was made defining the seed as the set of integers from 1 to 5. This way, the code is intended to run with an iteration that uses five distinct starting populations. The idea is that if different results are obtained between the various iterations, it would be possible to identify the actual optima as the smallest of these, thus identifying the robustness index. The very first tests immediately showed good prospects. In several cases, this strategy showed correlation with the results obtained with the exhaustive enumeration, where it had not been obtained before. On the other hand, however, this strategy immediately showed a weakness: an increase in computation time. Using five seeds as initially tested, the computation times were approximately five times the initial ones. This lost, at least partially, the goal of introducing the genetic algorithm, that consists in the reduction of computational times.

Thus, in order to reconcile accuracy and efficiency, parallel execution was introduced through the Parallel Computing Toolbox. In short, this solution allows you to distribute multiple runs (with different seeds) across multiple processors. This way, the same best-of-seed selection can be exploited, without resulting in longer computational times. Matlab documentation confirms that genetic algorithms support parallel execution, thus operating several loops in parallel (MathWorks Inc, 2025). In particular, the computer used had four workers available, so it was advantageous to attempt with a multiple of four seeds. Thus, the first attempt was performed with four seeds, and the results were excellent. In fact, for the totality of the reference structures, perfect correlation was obtained with the results given by the total enumeration of combinations, as reported in Table 16. These results were obtained by keeping the calculation times in the order of a few minutes. Although some variability was shown between the various runs of each structure (between different seeds), in each of the cases, the final result corresponds to the reference value, thus demonstrating the goodness of the strategy of using parallel execution and multi-seed exploration combined.

Table 16 - Results of the parallel multi-seed Genetic Algorithm: values obtained for each seed and comparison of the best-of-seeds outcome with the enumerative reference

Structure	Enumerative	Genetic Algorithm Implementation (parallel computing)			
		Seed1	Seed2	Seed3	Seed4
Str_1	0.1389	0.1389	0.1389	0.1389	0.1389
Str_2	0.1010	0.1010	0.1010	0.1010	0.1010
Str_3a	0.2857	0.4571	0.4571	0.4571	0.2857
Str_3b	0.5478	0.8000	0.5478	0.8000	0.8000
Str_5.1a	0.2857	0.2857	0.2857	0.4571	0.2857
Str_5.1b	0.3796	0.3796	0.3796	0.3796	0.3796
Str_5.2a	0.3614	0.3614	0.3614	0.3614	0.3614
Str_5.2b	0.3200	0.3200	0.3200	0.3200	0.3200
Str_6a	0.4078	0.4078	0.4519	0.4519	0.4519
Str_6b	0.2164	0.2291	0.2291	0.2291	0.2164
Str_7a	0.1932	0.4580	0.4580	0.4580	0.1932
Str_7b	0.3000	0.3000	0.4600	0.4600	0.3000
Str_7c	0.2392	0.2392	0.4000	0.4000	0.4000
Str_7d	0.1176	0.1176	0.1176	0.1176	0.1176
Str_7e	0.1895	0.1895	0.1895	0.1895	0.1895

Regarding efficiency, measured through runtime, the parallel multi-seed strategy performed on the 15 reference structures gave the following wall-times: 84, 121, 97, 132, 106, 133, 51, 62, 112, 106, 120, 126, 135, 164, 119 s, for an average of about 111 s. To give a reference, the set of analysed

structures have an average of six elements in total. even though it is a limited sample, the parallel multi seed setup showed great potential, having re-established the efficiency sought with the introduction of the genetic algorithm, while ensuring an accuracy at the level of exhaustive enumeration.

Finally, the approach was tested on more complex structural models, for which exhaustive enumeration had proven to be completely impractical. Indeed, for these five structures tested, a result was not obtained even with computation times exceeding 12 hours, precisely because of the well-known issue of the number of combinations that grows exponentially with the number of elements and degrees of constraint. Therefore, for these structures there are no reference values of the robustness index to perform a correlation, but only the calculation time is considered. Reference structures have a total number of elements around 20 units. The results are reported in Table 17, which shows the calculation times and the number of elements for each of the cases analysed. Although the times have increased compared to previous tests, the result is still excellent, considering that without the introduction of the genetic algorithm, no results had been obtained even after 12 hours of calculation and therefore the actual times are not even known. In conclusion, the adoption of the genetic algorithm therefore allows to extend the application of the metric to structures of dimensions that were previously out of search for the exhaustive enumeration method. Furthermore, parallel execution has proven to be a good method for preserving wall-times using multiple seeds.

Table 17 - Execution time for structures of increasing size and complexity

Structure	Number of elements	Runtime
B1	22	885 s
B2	22	658 s
B3	21	592 s
B4	18	513 s
B5	18	558 s

Taking into account all what was analysed in this validation section, it turns out that considering the set of 15 analysed reference structures, the implementation of the genetic algorithm with a single-run can lack accuracy even after refining the parameter tuning. Using multiple seeds solves these result correlation problems, and parallel execution allows these results to be obtained without dilating the wall time. For more complex structures, this method has proven to work where exhaustive enumeration stalls, with total runtimes on the order of 8–14 minutes for 18–22 elements. the validation indicates that the genetic algorithm with best-of-seeds selection reports the accuracy of the exhaustive enumeration itself, but significantly reducing the computation time and allowing to deal with structures of larger size and complexity that were previously not possible to analyse.

7 CONCLUSIONS

7.1 Overview of the Study and Objectives

Quantifying structural robustness remains a challenging task, although design codes provide criteria for promoting it, they don't provide a univocal method for calculating a scalar measure of it. The definition of a scalar indicator that measures robustness is therefore still an open research side. In this context, metrics based on the topology and kinematics of structures can represent a solution. These are in fact presented as a mathematically simple tool, which takes into account the connectivity of the structure to study the possible activations of collapse mechanisms, without considering material properties and mechanical properties of the elements.

Among the robustness metrics proposed in the literature, the Connectivity-Based Resilience Index introduced by Chiaia et al. (2019) aims to be a rigorous and compact tool for measuring structural robustness by analyzing kinematic matrices of structures in a series of damage scenarios, to evaluate the onset of kinematics. The metric evaluates these possible collapse scenarios, quantifying both the initial damage and the kinematic response to calculate an indicator, defining the robustness index as the minimum value obtained among all possible damage scenarios. This thesis lays thus its foundations on the original formulation of this metric and on a first numerical implementation of it, developed by Professor De Biagi on Matlab. The role of this thesis was therefore to test this implementation, analyze its behaviour, evaluate its strengths and weaknesses, and calibrate its parameters to have a balanced index.

The objectives of this thesis work are therefore collected in the following three lines:

- Metric analysis and calibration, ensuring that the inclusion of all damage possibilities to the initial version that only included the removal of constraints results in a consistent and balanced representation of structural robustness;
- Validate the metric from a theoretical and intuitive point of view through a series of seven operational properties, designed to evaluate the behaviour of the index by verifying the agreement with engineering intuition and classical structural mechanics;
- Overcome computational burden problems arising from the enumerative nature of the metric, i.e., considering all possible combinations of constraints and removed elements, by introducing a genetic algorithm that must ensure accuracy of the results but drastically reduce computation times.

Through these developments, the thesis therefore aims to consolidate this metric as a solid tool and to ensure its applicability from a computational point of view, thus contributing to the evolution of this tool from purely conceptual to operative.

7.2 Main Scientific Contributions

The work carried out in this thesis helped to consolidate and develop the connectivity-based resilience index under several points, starting from its numerical implementation, constant calibration and computational operability. As mentioned yet, a first contribution consisted of testing the new version of the implementation, which for the first time contained all types of damage. This broader

implementation was therefore needing to be systematically tested and validated in its behavior. This was done using a large number of benchmark structures. In this way, the structural coherence and the fact that the various types of damage were taken into account in a balanced manner were verified. The strengths and weaknesses of the method were therefore highlighted, proposing refinement and improvement strategies.

The second substantial contribution of the thesis concerns the work carried out in the calibration of the constants a–b–c–d–e, which govern the quantification of both initial damage and any eventual resulting kinematics. Compared to the calibration proposed in the original formulation of the metric of 2019, the calibration proposed here is more general, as some of the constants were originally defined by reference to very precise structural schemes, thus lacking in generality. The calibration followed a very precise procedure: initially, the effect of single constants variation was evaluated, after that the effects of the interaction between the various constants were then also considered. All this was done with the aim of identifying a balanced set of values that produces realistic and easily interpretable results. In particular, the new calibration introduced a normalized definition of the constant governing the weight of constraint removal, ensuring independence of the results from the size and complexity of the analyzed structure; the value of the constant associated with element removal was then reduced; the value associated with translational constraint removal was increased with the aim of limiting the appearance of unrealistic mechanisms of rigid translation of the structure as a critical combination; and finally, a new constant was introduced, which has the task of simply scaling the final result to make it more readable.

A third scientific contribution concerns the validation of the metric through the correlation of results with a series of operational properties, defined to evaluate the consistency with basic concepts of structural mechanics and in general with trends that a robustness metric is expected to respect. Initially, only few of these properties were satisfied, but after careful modifications or refinements to the procedure that do not affect the logic of the metric, six out of seven properties are finally verified. Most significative insights are the follow up ones. Specific tests carried out to analyse Properties 1 and 2 revealed how isostatic or externally isostatic structures produced non-zero index values, unlike what was expected, even if they do produce low values. This result led to the introduction of a specific clause within the script, which provides for a zero R index if an unproportioned collapse scenario is detected causing the entire structure to collapse starting from the removal of only one degree of constraint, thus achieving compliance with the two properties that were not initially satisfied. After that, Property 3 motivated the introduction of a system for assigning the mass parameter of the elements depending on their stiffness. This prevents elements with negligible stiffness from artificially raising the value of robustness. Property 6 immediately showed through simple tests a strong dependence of the results on the mesh density, contrary to what was desired. To address this occurrence, a pre-processing stage that substantially eliminates the mesh was proposed, returning the structural model "clean" and ready for use in robustness analysis. The only property that remains unsatisfied is the number 4, due to an intrinsic limitation of the method, which does not prove in the results a local increase in redundancy if the global connectivity remains unchanged.

Finally, another important contribution achieved by this thesis work concerns the computational feasibility of the numerical implementation of the metric. In fact, the original brute-force exploration, which took into account every possible damage configuration, soon proved to be prohibitive and

inapplicable with the increase in structural elements. This problem already arose for about ten elements, effectively making the method inapplicable for real structural models. Therefore, the introduction of a genetic-algorithm-based search strategy represents a fundamental step forward, allowing to compute structures of significantly greater complexity than those that could initially be examined. Then, multi-seed palallelised implementation has been identified as the right solution to ensure the same accuracy achieved for exhaustive search, finally managing to have a large computational efficiency without losing quality in terms of results. With these measures, it's possible to perform calculations that used to have impracticable durations, on the order of days, in just a few minutes. The advances brought about by the introduction of this method have therefore transformed the metric from a purely theoretical construct to a tool that can be realistically applied in engineering practice.

7.3 Discussion: Strengths and Limitations

The developments carried out in this thesis, together with all the tests conducted, allow for a reflection on the strengths and weaknesses of the metric and the related numerical calculation method. Certainly, one of the strengths lies in mathematical simplicity. In fact, since the metric only evaluates the collapse mechanisms through kinematic matrices, there are no parameters regarding materials, stiffnesses or other nonlinear behaviours that fall within the calculation. This makes the metric a conceptually transparent and easily interpretable tool, with robustness being a function of structural topology alone, rather than of strength or deformation capabilities. Furthermore, another strength to note is the correlation that was found with conceptual expectations in most cases, with the satisfaction of six out of seven operational properties following the refinements introduced in this work. Another point in favour concerns the applicability of the method regardless of the structural typology, provided that they are plane frames. In fact, since the metric does not depend on any type of material characteristic, it can be used for steel, concrete, wood, composite structures, or any other type. Finally, the development of a pre-processing phase aimed at eliminating any discretization of the structural elements makes the metric insensitive to mesh density, thus ensuring that the evaluation of the metric does not depend on artifacts of the modelling process.

Despite these strengths, there are several weaknesses that remain intrinsic to the nature of the metric. A first limitation comes from an aspect that has also been identified as a strength of the metric, namely the topological character of the method. In fact, this characteristic means that there is a binary identification of the collapse mechanisms, therefore present or absent, without taking into account the required energy, applied loads, deformation capacity or any dynamic effects. This aspect, which at the same time represents a strength by ensuring mathematical simplicity for the instrument, on the other hand limits the refinement of the result, preventing us from capturing, for example, the severity of different collapse mechanisms. However, this aspect can be partially mitigated by the proposed introduction of a mass parameter assignment strategy based on the stiffness of the elements, developed in the context of Property 3. This solution in fact includes some mechanical characteristics in the quantification of initial damage and consequent collapse. Another limitation concerns the dependence of the scalar result of the index on the calibration of the *a-b-c-d-e* constants. Although the calibration developed in this thesis ensures consistency, generality, and easy interpretability, the fact remains that different choices regarding the values of the constants would lead to different results. This makes the metric likely to be more effective as a comparative or screening tool, rather than as

an absolute measure of robustness. Finally, another major weakness of the method lies in the fact that it is currently only applicable for two-dimensional frames. Extending the formulation to three-dimensional cases is possible, however, but would require redefining the kinematic matrices with six degrees of freedom per node and consequently significantly increasing the level of complexity throughout the entire numerical implementation, as well as posing a new challenge in terms of computational efficiency.

All in all, the proposed robustness metric still turns out to be a simple, transparent, and computationally efficient enough tool to capture the topological essence of robustness. In any case, the limitations reported in this paragraph highlight the aspects that could be further developed to obtain an even more comprehensive tool.

7.4 Future Developments

The tests conducted and the results obtained in this thesis indicate several directions towards which the Connectivity-based Resilience Index could be further developed and refined. A first development scenario concerns the mass parameter assignment strategy introduced in this work. The current formulation incorporates mechanical information into the mass parameter in order to weigh the initial damage and kinematic value generated proportionally to the stiffness of the elements in question. However, the stiffness indicators used in defining the mass parameters are intentionally simplified and do not consider some structural aspects, such as end releases. These indicators are not in fact based on the full stiffness matrix, for reasons of simplicity. Therefore, a mass assignment based on the stiffness of the elements that takes into account the stiffness matrix of the elements as a whole could more accurately represent the characteristics of the structure, leading to an even more precise quantification of the robustness index.

A second possible direction of development concerns the extension of the metric to three-dimensional structures, as anticipated in the previous paragraph. It would be necessary to generalize kinematic matrices to six degrees of freedom per node, thus making it possible to identify a wider and more realistic variety of collapse scenarios. On the other hand, this solution would introduce significant computational and modeling challenges. The dimensionality of the null space would indeed have to grow,, and much more complex algorithms would be needed in general to complete the analysis. Nevertheless, such an extension would significantly increase the spectrum of applicability of the metric, making it an even more versatile tool in real-world engineering contexts.

A third area of possible further development concerns a possible improvement of the genetic algorithm. While the current implementation has already made huge strides in reducing computation times, it would be possible to explore further optimization strategies. For example, adaptable genetic operators could be tested, which could customize the evolution of mutation and crossover parameters depending on research progress. This could accelerate the convergence process while ensuring greater exploration of the solution space. Furthermore, another possible development strategy involves the introduction of machine-learning-based guidance to support algorithm initialization, i.e., to identify the most promising regions of the solution space and generate the first population of individuals in a targeted manner. This could potentially make the algorithm even more efficient, making the next exploration phase faster.

A final future development that could be very useful is to apply the metric to real structures or known benchmarks. This would provide important information regarding the method's performance and allow comparisons to be made with other structural parameters to evaluate the metric's consistency. It would also be useful in identifying further needs for refinement or development.

Overall, these potential future development fronts show potential for improvement both theoretically and in terms of the practical use of the Connectivity-based Resilience Index.

7.5 Final Remarks

The work presented in this thesis has contributed to the theoretical and scientific advancement of the Connectivity-based Resilience Index. Through systematic analyses of the metric's numerical implementation, careful calibration of the constants, validation of the method through a set of operational properties, and the introduction of a genetic algorithm to streamline the computational model, the index has been transformed from a promising conceptual formulation to a more mature and operational tool for assessing the robustness of structures. The results confirm that such kind of approach based on topology can effectively identify disproportionate collapse mechanisms and provide useful insights for structural assessment.

At the same time, the metric is not intended to replace any traditional structural analyses. Instead, it should be interpreted as a complementary screening tool, useful for capturing the topological essence of structural robustness and highlighting critical damage scenarios requiring further investigation. The developments made in this thesis serve as the foundation for future developments that may include more detailed mechanical characteristics, the extension to three-dimensional models, and the refinement of computational strategies. Finally, the Connectivity-based Resilience Index represents a flexible framework capable of evolving through future developments, yet already endowed with mathematical rigor and capable of providing important information on structural robustness.

8 REFERENCES

ASCE, 2022. *Minimum Design Loads and Associated Criteria for Buildings and Other Structures (ASCE/SEI 7-22).* 7 22nd ed. Reston, Virginia, USA: American Society of Civil Engineers (ASCE), Structural Engineering Institute (SEI).

Baker, J. W., Shubert, M. & Faber, M. H., 2008. On the assessment of robustness. *Structural Safety,* Volume 30, pp. 253-267.

Bazant, Z. P. & Zhou, Y., 2002. Why Did the World Trade Center Collapse? Simple Analysis. *Journal of Engineering Mechanics,* 128(1), pp. 2-6.

BCSA, n.d. *SteelConstruction.info - British Constructional Steelwork Association.* [Online] Available at: https://www.steelconstruction.info/Structural_robustness [Accessed 4 September 2025].

CEN, 2002. *Eurocode - Basis of structural design,* Brussels: Eropean Committee for Standardization.

CEN, 2006. *Eurocode 1 - Actions on structures - Part 1-7: General actions - Accidental Actions.* Brussels: European Committee for Standardization.

Chiaia, B., Barchiesi, E., De Biagi, V. & Placidi, L., 2019. A novel structural resilience index: definition and application to frame structures. *Mechanics Research Communications,* Volume 99, pp. 52-57.

De Biagi, V., 2014. *Complexity and robustness of structures against extreme events.* Torino: Politecnico di Torino.

Department of Defense - United States of America, 2009. *Unified Facilities Criteria - Design of Building to Resist Progressive Collapse.* USA: Department of Defense.

Ellingwood, B. R., 2006. Mitigating Risk from Abnormal Loads and Progressive Collapse. *Journal of Performance of Constructed Facilities,* 20(4), pp. 315-323.

Ellingwood, B. R. & Dusenberry, D. O., 2005. Building Design for Abnormal Loads. *Computer-Aided Civil and Infrastructure Engineering,* Volume 20, pp. 194-205.

Frangopol, D. M. & Curley, J. P., 1987. Effects of damage and redundancy. *Journal of Structural Engineering,* 113(7), pp. 1533-1549.

Fu, G. & Frangopol, D. M., 1990. Balancing weight, system reliability and redundancy in a multiobjective optimization framework. *Structural Safety,* Volume 7, pp. 165-175.

Gulvanessian, H., Calgaro, J. A. & Holický, M., 2002. *Designer's Guide to EN 1990 Eurocode: Basis of Structural Design.* 1st ed. London: Thomas Telford Publishing.

Haupt, R. L. & Haupt, S. E., 2004. *PRACTICAL GENETIC ALGORITHMS.* II ed. Hoboken: John Wiley & Sons, Inc..

Izzudin, B. A., Vlassis, A. G., Elghazouli, A. Y. & Nethercot, D. A., 2008. Progressive collapse of multi-storey buildings due to sudden column loss - Part 1: simplified assessment framework. *Engineering Structures,* Volume 30, pp. 1308-1318.

JCSS, 2001. *PROBABILISTIC MODEL CODE Part 1 - BASIS OF DESIGN.* Zurich: Joint Committee on Structural Safety.

Kwasniewski, L., 2010. Nonlinear dynamic simulations of progressive collapse for a multistory building. *Engineering Structures ,* Volume 32, pp. 1223-1235.

MathWorks Inc, 2025. *MATLAB Help Center.* [Online] Available at: https://it.mathworks.com/help/gads/genetic-algorithm-options.html [Accessed 11 11 2025].

Melchers, R. E., 1999. *Structural Reliability Analysis and Prediction.* Newcastle: John Wiley & Sons Ltd..

Mitchell, M., 1996. *An Introduction to Genetic Algorithms.* Cambridge, Massachusetts: Massachusetts Institute of Technology.

NIST, 2007. *Best Practices for Reducing the Potential for Progressive Collapse in Buildings.* USA: National Institute for Standards and Technology.

NRC, 2020. *National Building Code of Canada 2020.* Ottawa, Canada: National Research Council of Canada (NRC).

Pearson, C. & Delatte, N., 2005. Ronan Point Apartment Tower Collapse and its Effect. *Journal of Performance of Constructed Facilities,* 19(2), pp. 172-177.

Starossek, U. & Haberland, M., 2011. Approaches to measures of structural robustness. *Structure and Infrastructure Engineering,* Volume 7, pp. 625-631.

Vrouwenvelder, A. C. W. M., 2002. Developments towards full probabilistic design codes. *Structural Safety,* Volume 24, pp. 417-432.

9 LIST OF FIGURES