Structures 27 (2020) 1777–1785

Contents lists available at ScienceDirect

Structures

journal homepage: www.elsevier.com/locate/structures

Implementation of new elements and material models in OpenSees software to account for post-earthquacke fire damage

Seyed Javad Mortazavi[a], Iman Mansouri[b,c,*], Paul O. Awoyera[d], M.Z. Naser[e]

[a] Department of Civil Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
[b] Department of Civil Engineering, Birjand University of Technology, Birjand, Iran
[c] Institute of Research and Development, Duy Tan University, Da Nang 550000, Viet Nam
[d] Department of Civil Engineering, Covenant University, Ota, Nigeria
[e] Glenn Department of Civil Engineering, Clemson University, Clemson, SC 29634, United States

ARTICLE INFO

Keywords:
Constitutive model
OpenSees
Fire
Thermal analysis
New material
New element

ABSTRACT

Post-earthquake fire (PEF) damage in buildings is a hazard, which is usually not taken into consideration during structural design in buildings. Past instances of PEF events have led to severe damage and loss of lives, often higher than those experienced during the earthquake event. PEF is a complex phenomenon that requires special treatment, one that can be pursued via *OpenSees*– an efficient structural modeling software. Despite recent advancements in OpenSees, the latest edition still lacks constitutive thermal and mechanical models for spring element, spring section, parallel material, and beam elements with hinges. For this, carrying out a global thermal analysis of structures with the aforementioned element/member characteristics has been a daunting task,. To overcome this knowldeg gap, this work presents the development of constitutive models to be used in OpenSees to enable engineers from conducting a reaslitic and practical engineering simulation of a PEF event. A further development of OpenSees, introducing a New *UserMat* function, has also been carried out. Lastly, a series of case studies were explored using numerical implementations of the proposed models and validated based on hand calculations and new OpenSees material development.

1. Introduction

In the aftermath of an earthquake event, structures are not only damaged but turn vulnerable to following load actions; most notably fire. For this, fire following an earthquake eventis often regarded as a major concern in areas highly active for seismic occurrences. This is particualry true knowing that the resulting interaction from the two events are not covered in the design process [1–7].

For a start, fire resistance of buildings is often evaluated by adhering to traditional provisions founded in ISO 834 [8] and ASTM-E119 [9]. Such provisions were developed via a primarily prescriptive approach in which individual structural members are tested under standard fire conditions. On the contrary, real (natural) fires merely follow standard requirements. In fact, in an actual structure, structural members interact together, and hence their behaviour differs from that in isolated fire tests. In other words, standard fire tests do not often consider the structural interactions of a member among all elements contained in the whole structure [10,11].

Despite the increasing urbanization in regions vulnerable to seismic activities, there is yet an increase in the potential damages associated with a fire hazard. Overall, occupants and officials expect that their structures should withstand in terms of stability and integrity, whether due to earthquake, or fire of PEF event, to allow emergency teams to evacuate affected occupants of such structures [12,13]. Limited works are available on the structural response of system-level and frames under the effect of PEF, mainly due to the limited regulatory requirements [14,15]. This has opened up the door to pursue a performance-based design perspective. Thus, this research is carried out to further ongoing research efforts such as that in full-scale fire tests at IIT Roorkee in India, and with the support of the University of Edinburgh [16], which focuses on experimentally modeling the resistance of an earthquake (simulated seismic loading) damaged structures to the fire. This motivated our work, wherein we used OpenSees (an open-source structural software) for modeling behaviour of structures under earthquake and fire loading.

Unlike OpenSees, other software with the capability of modeling

* Corresponding author at: Department of Civil Engineering, Birjand University of Technology, Birjand 97175-569, Iran.
E-mail addresses: mansouri@birjandut.ac.ir (I. Mansouri), mznaser@cemson.edu (M.Z. Naser).
URL: http://www.mznaser.com (M.Z. Naser).

https://doi.org/10.1016/j.istruc.2020.08.021
Received 23 May 2020; Received in revised form 23 July 2020; Accepted 6 August 2020
Available online 11 August 2020

structural behaviour during an earthquake or fire also exist and these fall under two categories [17]. The first, have been explicitly developed for structural analysis of buildings subjected to fire under the effort of prominent researchers such as SAFIR [18]. And the second, are commercial software thatcan be used for modeling and analyzing the behaviour of fire-damaged structures. While such software were not specifically developed to handle fire-related analysis, they still offer numerous capabilities; including pre- and post-processing performance, and intuitive interface.

Some of the popular commercial software, include ABAQUS, ANSYS, and DIANA – which have been ascertained for their suitability for thermal and mechanical analysis of structures.

These days resilience of communities during natural events has become an issue of concern in the world. For instance, in the event such as reported in this study (post-earthquake fire), there is a variety of issues relating to economic and social losses. The occurrence of a fire outbreak after an earthquake event is not rampant, however, its occurrence can be catastrophic. Forensic investigation on post-earthquake fire-affected site like that under extreme loading involves checking the intensity, location, and characteristics of primary and secondary hazards, and properties and response of structural elements. Unfortunately, the currently available design processes only are focused on prescriptive methods.

Currently, available performance-based design methods are achieved by advanced calculation models, like those in SAFIR. However, these complex models may require a very high level of technical expertise, user-input, and computational skill from designers [19]. These approaches are, therefore, not feasible in most structural cases, so they reserved for complex structures like that of China Central Television (CCTV) building in Beijing [19]. To develop a simple and efficient performance-based design approach has been overwhelming [20]. There is an intermediate level software program that is based on a slab panel method (SPM) by Clifton [21] that is suitable for analysis of slab subjected to fire. Walls [22] articulated the need for frames to be analyzed by Fire Beam Element (FBE) methods, which is a simple design approach fire-affected buildings. However, the inability of FBE to handle global analysis and structural continuity related problems requires investigation.

Hence, there is a need for development of a probabilistic approach that considers structures exposed to elevated temperatures and risk of fire ignition. For this, we need a powerful tool for structural fire analysis.

Due to the problems of unavailability of affordable software packages, the complexity of modeling realistic fire occurrence, and the transfer of heat within structures, the cases of fire occurrence and response of structures is not well explored globally. As such, there is a need for modern and practical solutions. One such solution is OpeSees.

OpenSees, which is interpreted as an Open System for Earthquake Engineering Simulation, is a program developed to handle the structural and geotechnical analysis of frames subjected to earthquakes. The software program has been initiated by the Pacific Earthquake Engineering Research (PEER) Center, to be employed to solve problems in the performance-based earthquake engineering field. Due to its open-source nature, different software developers regularly upgrade the program; which has aided the advancement of the OpenSees to solve complex problems.

The current study developed models of PEF-affected structural systems in OpenSees. our efforts are motivated by the notion that OpenSees is fast, stable, and efficient in solving large nonlinear models. Further, OpenSees is an open-source software framework, and hence provides an attractive solution to a problem that share a magnitude of PEFs. One should still that the current version of OpenSees fire does not include the thermal springs. Therefore, with the incorporation of a software code addition (written primarily in C++) for simulation applications in earthquake engineering using finite element methods, this study opens the door to the possibility of a more accurae and realistic thermal evaluation of a variaety of structural systems in which springs elements are essential. For instance, spring elements are often used in the mathematical modeling of many systems, in which structural members are modeled as springs that are assembled in parallel and/or series configurations to represent a structure's equivalent stiffness. Other applications of spring elements entail panel zone, link beam in eccentrically braced frames, gusset plate connections, and post-tensioned beam-column connections.

1.1. Theory and framework design

1.1.1. OpenSees program methodology

OpenSees, a powerful open-source structural analysis software package, is based on four main modules, which enable its performance as a structural analysis tool [23,24]. Finite element models are performed in the OpenSees *Modelbuilder* module. The procedures and specifications for the analysis are done in the analysis module. The selected quantities to be monitored during the analysis are tracked in the Domain, and meanwhile, the output results are specified by the recorder object. The modules are interrelated from the perspective of object-oriented programming standards on which OpenSees is based. As a result, there is a more versatile relationship between the objects in the program, and this makes the program to be more efficient and modifiable.

When the program is running, the Domain object exhibits the role of the central operator, and this module is also known to be responsible for object storage after the *Modelbuilder* object has created it. Also, access to the objects at different levels of the analysis for the Analysis and Recorder objects is provided by the same module. The information of the structure being analyzed is constructed and stored by the *Modelbuilder*. The module also defines the element properties and constraints adopted for the modeling. The analysis object does the analysis calculations, and subcomponents needed for the structural analysis operations are stored in the analysis module. The sub-components do the following: handling constraint, numbering DOF, model analysis, algorithm solution Algorithm, Integrator, System of Equations, and Solver. The recorder object monitors present parameters in the model, and this during the analysis, and the data are sorted in a set of user-defined files. This module documents the overall behaviour of the system during the analysis.

1.2. Programming strategy

The object-oriented programming technique, coupled with the C++ programming language, has been employed in OpenSees to form the necessary objects required for the operation of the modules. C++ language has been utilized owing to the fact it the most widely available object-oriented language. This programming language has a significant support base. In general, object-oriented programming entails object creation comprising of several modules that interact within a set framework.

Various independent object classifications, with similar performance, are found in OpenSees platform, this also contributes some procedural methods, or functions. A final product, which is capable of potential changes in source codes, is obtained during implementation of the object-oriented techniques as the framework for OpenSees is constructed.

The OpenSees programming technique during modeling requires data to be abstracted within the programmed objects. The data abstraction entails decomposition of the data, algorithms, or calculations into a simplified procedure to showcase the essential behaviour of the related data. Issues of complexity are controlled, and user-friendly environment for changing the software by the data decomposition and algorithmic processes needed for the software analysis. The independent operation of modules is achieved by good abstraction in the programming base-levels, and thus, addition or any form of

modification in the program framework becomes easy. As a result, new codes can be implemented into the program code, thus with only a few changes to other objects or modules.

2. Materials and methods

2.1. Verification with an experimental test

Since any numerical study needs a verification, a well-known exprerimental test is selected for simulation in OpenSees herein. The effect of uniformly increasing temperature profile on some set of simply supported steel I-beams (1.14 m long) made of an IPE80 (ST37.2) section has been investigated by Rubert and Schaumann [25]. The beams were designed to carry an imposed load of a load ratio of 0.2 at the mid-span section. A validation of the aforementioned beams has been conducted earlier by Jiang and Usmani [26] in OpenSees, and Jeffers and Sotelino [27] following modelling of the force-based beam elements in ABAQUS.

The current study focuses on the comparative validation of beam-column elements developed in OpenSees. This experiment entails modeling of steel beams with load ratio of 20%. This model showcased the localised nonlinearity occurrence in the beam as a result of non-uniform thermal action. Before the thermal action, a uniformly distributed load of 4.5 kN/m was applied at the left end of the beam. A coefficient of thermal expansion α of 1.2×10^{-5} m/(m· K) and elastic stiffness E_0 of 2×10^5 MPa were assigned to the members. Fig. 1 shows the trend of the generated results. As one can see, there is a strong agreement between the mid-span deflection curves for beam temperature and model reported by Jeffers and Sotelino [27]. It is worthy of note that different mesh fineness modes gave similar outputs since there is constant thermal beam action.

2.2. Steel in fire

Past experiences of fire hazards have shown that unprotected steel is vulnerable to fire, mainly due to its high thermal conductivity and thinness of the members. It should be stated that significant deformation or total failure often occurs as a result of a fire hazard in steel structure especially at high temperature exceeding 750 °C – as a result of of a phase change and drastic strength and stiffness losses in steel.

2.2.1. Thermal expansion of steel

Eurocode EN 1993-2-1 [28] outlines the provision of curves for thermal strain (ε_θ) of various typical carbon steels, and reinforcing steels. The following equations show the variation of thermal strain in steel with temperature (θ_a):

$$\varepsilon_\theta = -2.416 \times 10^{-5} + 1.2 \times 10^{-5}\theta_a + 0.4 \times 10^{-8}\theta_a^J 20^\circ C < \theta_a \leq 750^\circ C \quad (1)$$

$$\varepsilon_\theta = 11 \times 10^{-3} 750^\circ C < \theta_a \leq 860^\circ C \quad (2)$$

$$\varepsilon_\theta = -6.2 \times 10^{-3} + 2 \times 10^{-5}\theta_a 860^\circ C < \theta_a \leq 1200^\circ C \quad (3)$$

2.2.2. Mechanical properties of steel at high temperature

Steel generally possesses a specific yield strength f_y as consider in conventional designs. However, the yield strength is not considered for fire design, as its stress-strain behaviour varies with increasing temperature. The Eurocode applies a reduction factor to take care of the reduction of steel strength and stiffness properties. The Eurocode factors are temperature-dependent; they are multiplied by the original strength or stiffness at room temperature (20 °C). The factors are as follows:

$$k_{y,\theta} = f_{y,\theta}/f_y \quad (4)$$

$$k_{p,\theta} = f_{p,\theta}/f_y \quad (5)$$

$$k_{E,\theta} = E_\theta/E \quad (6)$$

where $k_{y,\theta}$ = reduction factor for effective yield strength, $k_{p,\theta}$ = reduction factor proportional limit, and $k_{E,\theta}$ = reduction factor for the slope of the linear elastic range.

High-temperature stress-strain curves for a temperature range is specified in Eurocode EN 1993-1-2 [28].

It is shown in the stress–strain curves at different temperatures, that they similar shape, there is lower stress pattern at higher temperatures. The effects of creep on yield stress are encompassed in the curves [19,20], where strain hardening is conservatively neglected. The following equations are used to represent the curves in EN 1993-1-2 [28]:

$$\sigma = \varepsilon E_\theta 0 < \varepsilon \leq \varepsilon_{p,\theta} \quad (7.1)$$

$$\sigma = f_{p,\theta} - c + \frac{b}{a}\sqrt{a^2 - (0.02 - \varepsilon)^2}\,\varepsilon_{p,\theta} < \varepsilon \leq \varepsilon_{p,\theta} = 0.02 \quad (7.2)$$

$$\sigma = f_{y,\theta}\varepsilon_{y,\theta} < \varepsilon \leq \varepsilon_{t,\theta} = 0.15 \quad (7.3)$$

$$\sigma = f_{y,\theta}\left(1 - \frac{\varepsilon - 0.15}{0.05}\right)\varepsilon_{t,\theta} < \varepsilon \leq \varepsilon_{u,\theta} = 0.2 \quad (7.4)$$

$$\sigma = 0.0\varepsilon > \varepsilon_{u,\theta} \quad (7.5)$$

The term c is calculated by:

$$c = \frac{(f_{y,\theta} - f_{p,\theta})^2}{(\varepsilon_{y,\theta} - \varepsilon_{p,\theta})E_\theta - 2(f_{y,\theta} - f_{p,\theta})} \quad (8)$$

This is then substituted to determine a and b:

$$a^2 = (0.02 - \varepsilon_{p,\theta})\left(0.02 - \varepsilon_{p,\theta} + \frac{c}{E_\theta}\right) \quad (9)$$

$$b^2 = c(0.02 - \varepsilon_{p,\theta})E_\theta + c^2 \quad (10)$$

2.3. Fiber section

During a fire incident, there is a tendency that the cross-section of a beam is exposed to a thermal gradient. Thus, the fiber element approach is applied to the cross-sections to model the non-uniform temperature distribution. As shown in Fig. 2, section height is discretized into n number of smaller rectangular fibers, with each fiber i denoting a small rectangular area A_i of the section and the second moment of inertia I_i about the neutral axis (NA). It is noteworthy that the I_i value changes according to the shifting of the NA.

The importance of applying the fiber section approach for FBE are as follows:

- There is a possibility of assigning different temperatures to each fiber, which therefore simulates a thermal gradient.
- As each fiber is assigned a material model, the fiber section, therefore, takes care of composite parts.
- Gradual yielding and plasticity distribution in a section during the fire is captured using the fiber analogy.

2.4. Strains in fire

According to EN 1992-1-2 [29], the total strain (ε) *comprises of* the sum of thermal strain (ε_θ), mechanical strain (ε_σ), creep strain (ε_{creep}), and transient strain (ε_{tr}), as shown in Eq. (11) [20]:

$$\varepsilon = \varepsilon_\theta + \varepsilon_\sigma + \varepsilon_{creep} + \varepsilon_{tr} \quad (11)$$

As shown in Fig. 3, the Eurocode, however, incorporates creep and transient strain in the material models and reduction factors. This thus simplifies the Eqs. (11) and (12).

$$\varepsilon = \varepsilon_\theta + \varepsilon_\sigma \quad (12)$$

Thus, the thermal strain is developed as there is an increase in

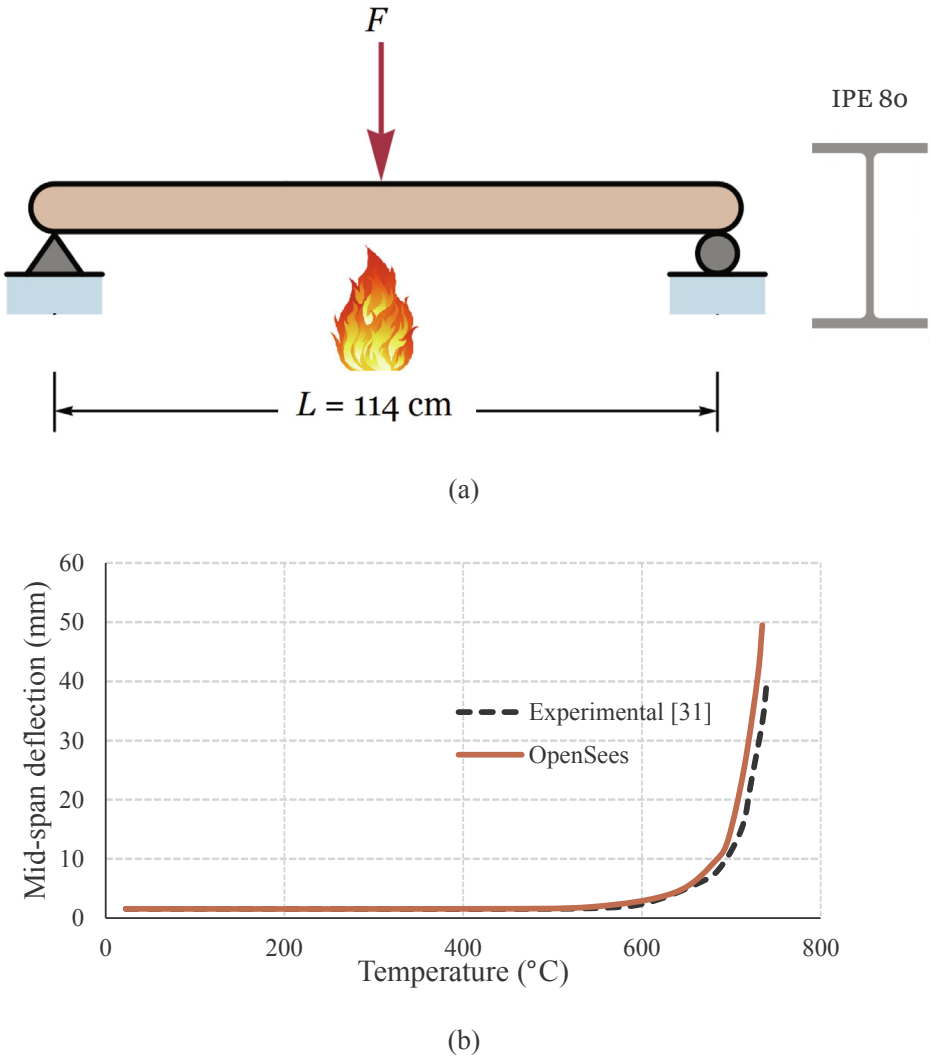

Fig. 1. Verification, (a) the test specimen, (b) comparison of results of OpenSees and test.

temperature, which can be calculated directly based on the specified material models. This study follows Eurocodes [28,29] criteria for calculation of the thermal strains of steel. It should be noted that this takes sign convention for strains and stresses as negative for compression and positive for tension.

3. Results and discussion

3.1. Developing of the thermal spring element

In this study, the new material model for the OpenSees program was developed with Visual C++ programming language. With that, programming performed with the new material was executed using the proposed formula, followed by the complication of the written code. The output of the compiled code was DLL format. Thus, the software utilizes the DLL during the OpenSees software operation and compiling code for the new material.

C++ Programming language is described as an object-oriented programming language, which requires that each material and element is defined as an object. Moreover, classes are the unit of programming using this language, from which objects are created through it. Also, functions are linked to each class. The implication of this is that a new programming class, having numerous functions, is used for the development of new material. There are two main categories of functions, which comprises of those functions associated with the OpenSees software (also called by the software). With the aid of specific input and output terms, the functions are programmed following OpenSees developers' standards.

A new strain in the material is set based on the *setTriaStrain()* method, which is called by an element. Whereas, for subsequent calls, *getTangent()* and get *stress()* are returned to their corresponding tangent and stress values, respectively. It should be noted that *setTrialStrain()* is invoked when the solution algorithm performs trials of finding solutions, thus going from a valid solution to the next on the solution path.

However, *commitState()* method is invoked after a trial solution can be seen solution path. Yet, the material is responsible for backtracking to that solution if there is invoking of a *revertToLastCOmmit()*. This kind of occurrence is observed if a solution on the solution path could not be found by the algorithm.

An element in the element constructor is responsible for the invoking of the *getCopy()* method. But a unique copy of the materials is returned to the element. As a result, different elements can utilize the same type of material and similar properties, in such a way that each element possessed its unique copy.

Several functions are also available, in which programmers are implemented based on the predicted algorithm for their desired

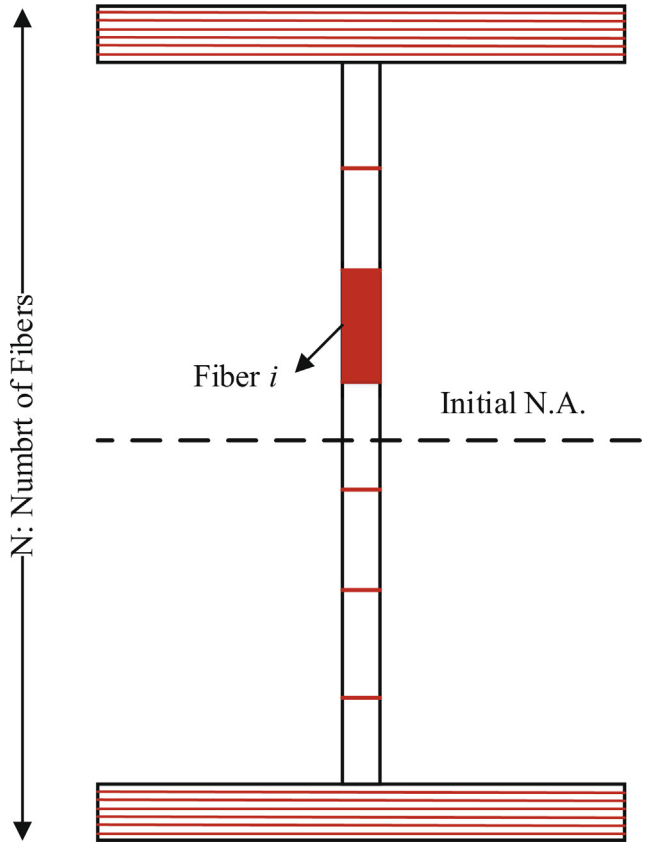

Fig. 2. A typical discretized steel profile into *n* number of fibers [20].

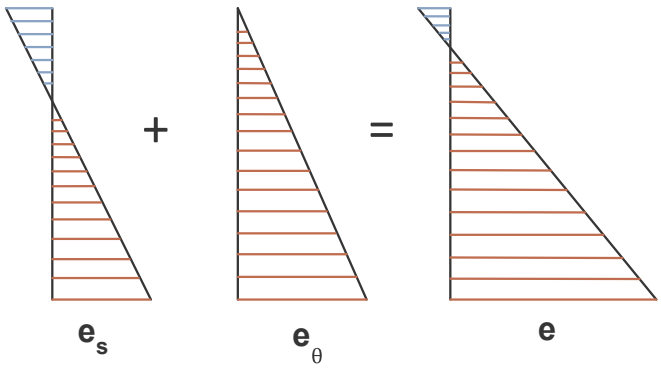

Fig. 3. Example of a) the total strain in a cross-section as the combination of b) the mechanical strain and c) the thermal strain. In this example, the thermal load has a uniform temperature gradient across the depth of the cross-section [20].

material. The developers take full charge of this, as the functions can have inputs and outputs following the developer's will, without any known limitations.

A *zeroLength* element object (having a two nodes thermal functions) is constructed using the developed command. Connection of nodes was achieved using multiple *UniaxialMaterial* objects for the representation of the element force-deformation relationship:

element zeroLengthThermal $eleTag $iNode $jNode -mat $matTag1 $matTag2 … -dir $dir1 $dir2 … < -doRayleigh $rFlag > < -orient $ × 1 $ × 2 $ × 3 $yp1 $yp2 $yp3 >

in which $eleTag = unique element object tag, $iNode $jNode = end nodes, $matTag1 $matTag2 … = tags associated with previously-defined *UniaxialMaterials*, $dir1 $dir2 … = material directions, $ × 1 $ × 2 $ × 3 = vector components in global coordinates defining local x-axis (optional), $yp1 $yp2 $yp3 = vector components in global coordinates defining vector yp which lies in the local x-y plane for the element (optional), $rFlag = optional: default = 0 where rFlag = 0 means NO RAYLEIGH DAMPING (default) and rFlag = 1 include Rayleigh damping.

Appendix A1 contains the most important code written in C++ in Microsoft Visual Studio.

In this section, validation of the *zeroLength* element implemented in OpenSees is done by the application of a problem. This entails heating a one-dimensional spring element under a uniform temperature (0–1000 °C). The uniform load is defined as a *pattern Plain Linear* in OpenSees. Therefore, the expression described in section 3 could be used to manually solve the problem. The spring's response to fire under this methodology is evaluated by comparing hand calculation results and that of OpenSees. The process of implementation matched with the theory is illustrated in Fig. 4.

Table 1 shows how the thermal loading is applied to the *zeroLength* element, which is compatible with the OpenSees output.

Also, there was the incorporation of the thermal functions to the *zeroLengthSection* element. Using this command, the construction of two nodes *zeroLength* element objects, is achieved along with the same location. Nodes connection was made using a single section object representation of the force–deformation relationship of the element.

3.2. Developing of the thermal parallel material

The command for parallel thermal material was used for the construction of an identical material object, which is made up of an arbitrary number of previously-constructed *UniaxialMaterial* objects. An example of this is presented in Fig. 5.

uniaxialMaterial ParallelThermal $matTag $tag1 $tag2 … < -factors $fact1 $fact2 … >

in which $matTag = integer tag identifying material, $tag1 $tag2 … = identification tags of materials making up the material model, $fact1 $fact2 … = factors to create a linear combination of the specified materials. Factors can be negative to subtract one material from another (optional, default = 1.0).

Appendix A2 contains the most important code written in C++ in Microsoft Visual Studio.

An example was considered to control the correct implementation of parallel material in OpenSees fire, which takes care of the concept of the parallel spring. Heat analysis was performed under a stable temperature condition. Fig. 6 shows the implementation of identical material in OpenSees fire.

3.3. Developing of the beam with hinges thermal element

With this command, it has been possible to construct a *beamWithHinges* element object. This utilizes the non-iterative (or iterative) flexibility formulation, and plasticity is considered as concentrated over specified hinge lengths of the element ends, as shown in Fig. 7. Plastic hinges are localized at the element end using the *beamWithHinges* element.

With this type of hinges, elements are divided into three parts. There are two hinges at the ends, a linear-elastic region in the middle. Thus, hinges are defined simply by assigning them to each of the previously-defined sections. The user is expected to specify the length of each hinge.

This study adopted thermal functions that were added to the element in the following format:

element beamWithHingesThermal $eleTag $iNode $jNode $secTagI $HingeLengthI $secTagJ $HingeLengthJ $secTagC $transfTag < -mass $massDens > < -iter $maxIters $tol >

$eleTag = unique element object tag, $iNode and $jNode = end nodes, $secTagI = identifier for previously-defined section object corresponding to node i, $HingeLengthI = hinge length at node i,

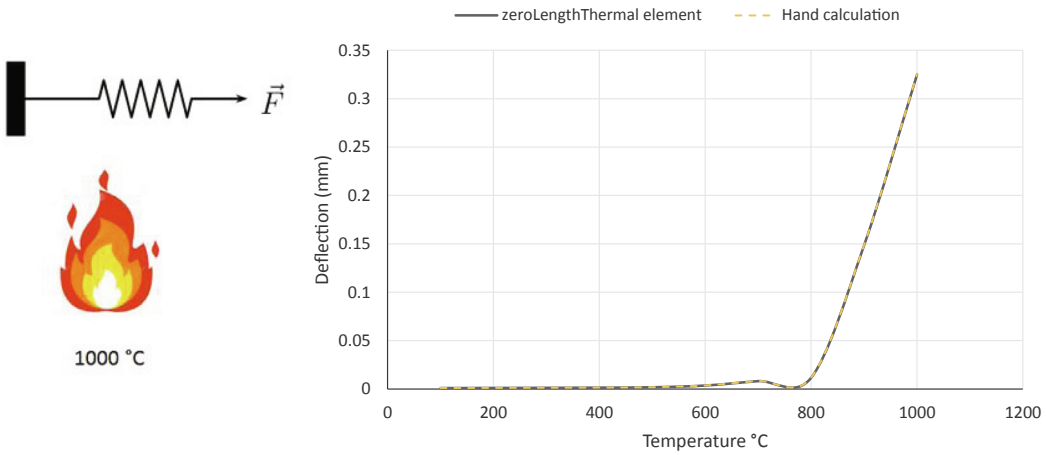

Fig. 4. Time-temperatures curves recorded for the spring element by hand calculation and OpenSees.

Table 1
Calculations of spring deflection under thermal loading.

Temperature °C	$\varepsilon_{,\theta}$	Deflection (mm)	$f_{y,\theta}$ (MPa)	E_θ (MPa)		
43.2	0.000502	0.000955	1	100,000		
89.7	0.001084	0.001008	1	99,030		
136.3	0.001686	0.001062	1	94,370		
182.8	0.002303	0.001116	1	89,720		
229.3	0.002938	0.001179	1	85,070		
275.8	0.00359	0.001247	1	80,420		
322.3	0.004259	0.001324	1	75,770		
368.8	0.004945	0.001412	1	71,120		
415.3	0.005649	0.001528	0.92234	66,470		
461.8	0.00637	0.001702	0.82004	61,820		
508.3	0.007109	0.001995	0.69227	51,793		
554.8	0.007865	0.002834	0.54812	38,308		
601.3	0.008638	0.003709	0.41888	27,166		
647.8	0.009428	0.005823	0.30728	18,796		
1	100,000	694.3	0.010236	0.007938	0.21284	12,428
1	100,000	740.8	0.011061	0.009624	0.15704	10,568
1	100,000	787.3	0.011	0.011252	0.10635	8835.75
1	100,000	833.8	0.011	0.05796	0.0831	7789.5
1	100,000	880.3	0.011406	0.121607	0.05985	6743.25
1	100,000	1000	0.0138	0.324938	0.036	4050

$secTagJ = identifier for previously-defined section object corresponding to node j, $HingeLengthJ = hinge length at node j, $secTagC = identifier for previously-defined section object corresponding to the elastic section, $transfTag = identifier for previously-defined coordinate-transformation object, $massDens = element mass density (per unit length, $maxIters = maximum number of iterations to undertake to satisfy element compatibility, and $tol = tolerance for satisfaction of element compatibility.

Appendix A3 contains the most critical code written in C++ in Microsoft Visual Studio.

A 1.4 m long beam having only the left half subjected to a normal range (0–1000 °C) is presented in Fig. 8. However, the right half of the

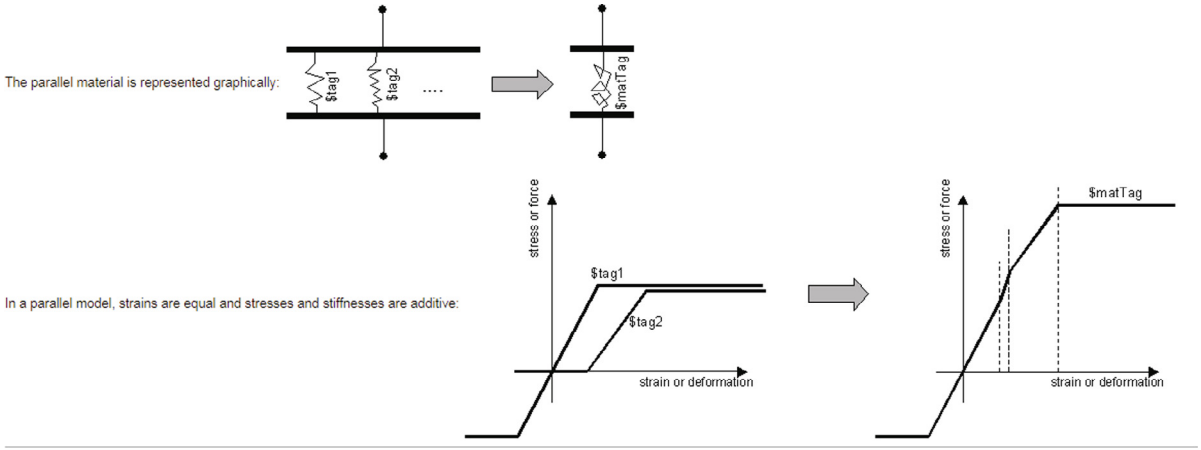

Fig. 5. Parallel material [30].

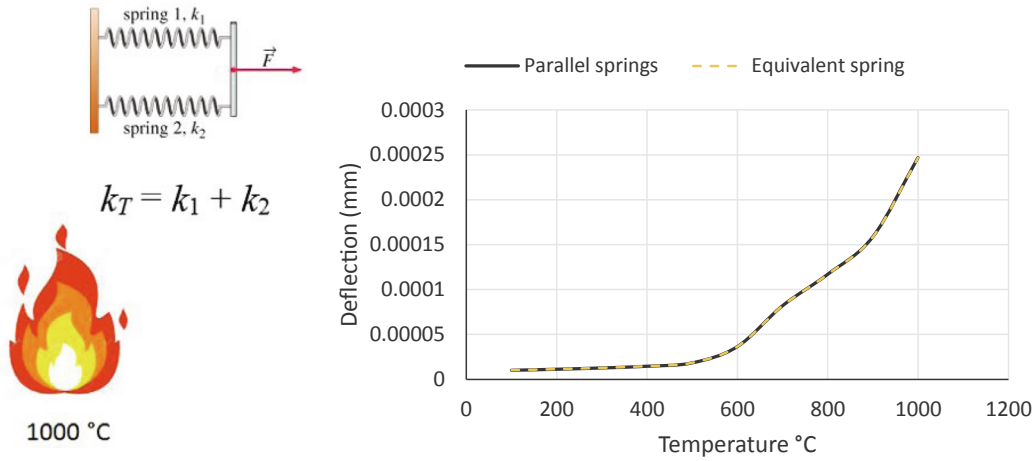

Fig. 6. Time-temperatures curves recorded for the parallel material by hand calculation and OpenSees.

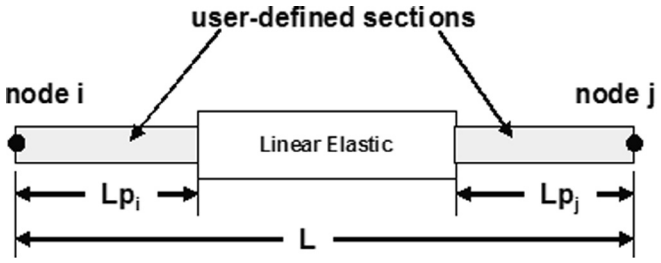

Fig. 7. The *beamWithHinges* element in OpenSees [30].

beam is at ambient temperature, and it thus acts as a translational spring for restraining the displacement of the left part. The model includes two beam elements in which material class *Steel01Thermal* and an initial elasticity modulus of 200 GPa at 0 °C was used.

It is possible that uniform heating is applied to the member until failure as a result of the loss of stability. In Fig. 8, the displacements recorded during the experimental test at the mid-span of the beam are presented. The modeling process covers the use of both *dispBeamColumnThermal* and *beamWithHingesThermal* elements. Overall, it has been demonstrated that strong agreement exists between test data and displacements (calculated using the modified OpenSees module) as evidence in the plots. This thus shows the adequacy of the implementation of the new elements in OpenSees fire.

4. Conclusions

This study focuses on the implementation of thermal spring elements, thermal spring section, thermal parallel material, and beam with hinges thermal element in OpenSees.

The available code on the OpenSees website has been verified to be compatible with the current version of C++ available in Visual Studio.Net. It is required that the version of C++ should be present on the computer with the Tcl scripting language program (applicable to the release of the code being built) after downloading the source code. Information regarding the presence and location of system files needed for a valid code compiling is available on the website. This requirement is taken care of by simply copying the correct file folders to the correct computer directories.

In this study, the development of the OpenSees for the analysis of fire-damaged structures was investigated by expanding the program to take into consideration spring and *beamWithHingesThermal* elements with other parallel material. This has been achieved by the implementation of fire beam element methodology into the OpenSees finite element software and validating results using case study examples.

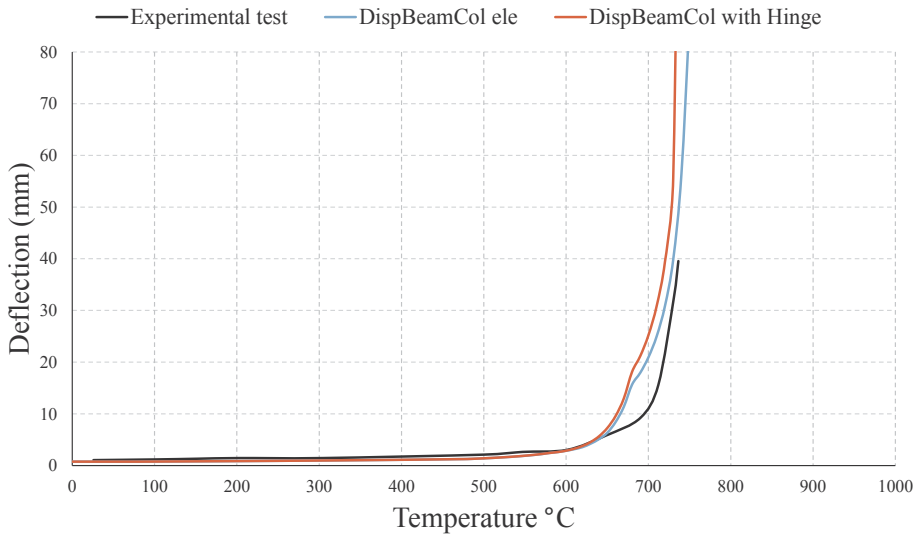

Fig. 8. Results of numerical OpenSees against the experimental test.

Finally, due to the unavailability of elements and material, the global thermal analysis was impossible for structures.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix:. Developed code in C++ for OpenSees

A.1. Function ZeroLengthThermalEle

```
#include < ElementalLoad.h >
#include < NodalThermalAction.h >
#include < ThermalActionWrapper.h >
#include < math.h >
#include < stdlib.h >
#include < string.h >
#include < ElementResponse.h >
// initialise the class wide variables
Matrix ZeroLengthThermalEle::ZeroLengthThermalEleM2(2,2);
Matrix ZeroLengthThermalEle::ZeroLengthThermalEleM4(4,4);
Matrix ZeroLengthThermalEle::ZeroLengthThermalEleM6(6,6);
Matrix ZeroLengthThermalEle::ZeroLengthThermalEleM12(12,12);
Vector ZeroLengthThermalEle::ZeroLengthThermalEleV2(2);
Vector ZeroLengthThermalEle::ZeroLengthThermalEleV4(4);
Vector ZeroLengthThermalEle::ZeroLengthThermalEleV6(6);
Vector ZeroLengthThermalEle::ZeroLengthThermalEleV12(12);
  // Matrix& tran = *t1d;
  int ret = 0;
  for (int mat = 0; mat < numMaterials1d; mat++) {
    // compute strain and rate; set as current trial for material
    strain = this->computeCurrentStrain1d(mat,diff);
  strainRate = this->computeCurrentStrain1d(mat,diffv);
    //opserr<<Temp<<"\n";
  ret += theMaterial1d[mat]->setTrialStrain(strain,Temp,strainRate);
}
  // get tangent for material
    strain = this->computeCurrentStrain1d(mat,diff);
  strainRate = this->computeCurrentStrain1d(mat,diffv);
    theMaterial1d[mat]->setTrialStrain(strain,Temp,strainRate);
  E = theMaterial1d[mat]->getTangent();
  //opserr<<"E"<<E<<" Temp"<<Temp<<"\n";
```

A.2. Function ParallelThermal

```
// $Revision: 1.12 $
// $Date: 2007-02-02 01:19:30 $
// $Source: /usr/local/cvs/OpenSees/SRC/material/uniaxial/ParallelMaterialThermal.cpp,v $
// File: ~/material/ParallelModel.C
//
// Written: fmk
// Created: 07/98
// Revision: A
//
// Description: This file contains the class definition for
// ParallelModel. ParallelModel is an aggregation
// of UniaxialMaterial objects all considered acting in parallel.
//
// What: "@(#) ParallelModel.C, revA"
#include < ParallelMaterialThermal.h >
#include < ID.h >
#include < Vector.h >
ParallelMaterialThermal::ParallelMaterialThermal(
    int tag,
    int num,
    UniaxialMaterial ** theMaterialModels)
:UniaxialMaterial(tag,MAT_TAG_ParallelMaterialThermal),
trialStrain(0.0), trialStrainRate(0.0), numMaterials(num), theModels(0)
{
// create an array (theModels) to store copies of the MaterialModels
theModels = new UniaxialMaterial *[num];
if (theModels == 0) {
  opserr << "FATAL ParallelMaterialThermal::ParallelMaterialThermal() ";
  opserr << " ran out of memory for array of size: " << num << "\n";
  exit(-1);
}
}
```

A.3. Function beamWithHingesThermal

```
#include < Domain.h >
#include < Node.h >
#include < Matrix.h >
#include < ForceBeamColumn2dThermal.h >
#include < ForceBeamColumn3dThermal.h >
#include < HingeMidpointBeamIntegration.h >
#include < HingeEndpointBeamIntegration.h >
#include < HingeRadauTwoBeamIntegration.h >
#include < HingeRadauBeamIntegration.h >
#include < ElementalLoad.h >
#include < NodalThermalAction.h >
#include < ThermalActionWrapper.h >
sections[0] = sectionI;
  sections[1] = sectionI;
  sections[2] = sectionM;
  sections[3] = sectionM;
  sections[4] = sectionJ;
  sections[5] = sectionJ;
  }
  else if (strcmp(argv[1],"beamWithHingesThermal3") == 0 ||
    strcmp(argv[1],"beamWithHingesThermal") == 0) {
  theBeamIntegr =
  new HingeRadauBeamIntegration(lenI, lenJ);
if (isShear) {
  SectionForceDeformation *sectionL = theBuilder- > getSection(shearTag);
  if (sectionL == 0) {
  opserr ≪ "WARNING section L does not exist\n";
  opserr ≪ "section: " ≪ shearTag;
  opserr ≪ "\nBeamWithHinges: " ≪ tag ≪ endln;
  return TCL_ERROR;
  }
  sections[numSections++] = sectionL;
  }
  theElement = new ForceBeamColumn2dThermal(tag, ndI, ndJ, numSections,
      sections, *theBeamIntegr,
      *theTransf,massDens,numIters,tol);
  delete theBeamIntegr;
```

References

[1] Kodur VKR, Naser MZ. Designing steel bridges for fire safety. J Constr Steel Res 2019;156:46–53. https://doi.org/10.1016/j.jcsr.2019.01.020.

[2] Naser MZ, Kodur V. Structural fire engineering. 1st ed. United States: McGraw Hill; 2020.

[3] Naser MZ. Autonomous Fire Resistance Evaluation. J Struct Eng 2020;146(6). https://doi.org/10.1061/(ASCE)ST.1943-541X.0002641.

[4] Naser M, Kodur V. Response of fire exposed composite girders under dominant flexural and shear loading. J Struct Fire Eng 2018;9(2):108–25. https://doi.org/10.1108/JSFE-01-2017-0022.

[5] Naser MZ, Kodur VKR. Comparative fire behavior of composite girders under flexural and shear loading. Thin Walled Struct 2017;116:82–90. https://doi.org/10.1016/j.tws.2017.03.003.

[6] Khorasani NE, Garlock M, Gardoni P. Probabilistic performance-based evaluation of a tall steel moment resisting frame under post-earthquake fres. J Struct Fire Eng 2016;7(3):193–216. https://doi.org/10.1108/JSFE-09-2016-014.

[7] Gerasimidis S, Khorasani NE, Garlock M, Pantidis P, Glassman J. Resilience of tall steel moment resisting frame buildings with multi-hazard post-event fire. J Constr Steel Res 2017;139:202–19. https://doi.org/10.1016/j.jcsr.2017.09.026.

[8] International Organisation for Standardisation (ISO), Part 1: Fire resistance tests-elements of building construction, ISO-834, Geneva, Switzerland, 1992.

[9] Standard test methods for fire tests of building construction and materials, ASTM-E119-07, West Conshohocken, PA, 2007.

[10] Jiang J, Jiang L, Kotsovinos P, Zhang J, Usmani A, McKenna F, et al. OpenSees software architecture for the analysis of structures in fire. J Comput Civ Eng 2015;29(1). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000305.

[11] Jeffers AE, Beata PA. Generalized shell heat transfer element for modeling the thermal response of non-uniformly heated structures. Finite Elem Anal Des 2014;83:58–67. https://doi.org/10.1016/j.finel.2014.01.003.

[12] Elhami Khorasani N, Garlock MEM, Quiel SE. Modeling steel structures in OpenSees: Enhancements for fire and multi-hazard probabilistic analyses. Comput Struct 2015;157:218–31.

[13] Hu JW, Chicchi R, Mansouri I, Mortazavi SJ, Kim JJ. Thermal performance of steel eccentrically braced frames subjected to fire conditions, 10th International Symposium on Steel Structures (ISSS). South Korea: Jeju; 2019.

[14] G. Della Corte, R. Landolfo, F.M. Mazzolani, Post-earthquake fire resistance of moment resisting steel frames, Fire Saf. J. 38(7) (2003) 593-612. https://doi.org/10.1016/S0379-7112(03)00047-X.

[15] Mousavi S, Bagchi A, Kodur VKR. Review of post-earthquake fire hazard to building structures. Can J Civ Eng 2008;35(7):689–98. https://doi.org/10.1139/L08-029.

[16] Zhang J. Developing OpenSees software framework for modelling structures in fire Ph.D. thesis United Kingdom: Heriot-Watt University; 2014.

[17] Franssen JM. SAFIR a thermal/structural program modelling structures under fire. Eng J, AISC 2005;42(3):143–58.

[18] Franssen JM, Gernay T. Modeling structures in fire with SAFIR®: Theoretical background and capabilities. J Struct Fire Eng 2017;8(3):300–23. https://doi.org/10.1108/JSFE-07-2016-0010.

[19] Wang Y, Burgess I, Wald F, Gillie M. Performance-based fire engineering of structures. New York: Spon Press; 2012.

[20] J.F. Volkmann, Implementation of the Fire Beam Element (FBE) in OpenSees for the analysis of structures in fire, M.Sc. thesis, Stellenbosch University, South Africa, 2018.

[21] Clifton GC. Design of composite steel foor systems for severe fires, Report R4–131. Manukau City: HERA; 2006.

[22] Walls RS. A beam finite element for the analysis of structures in fire Ph.D. thesis South Africa: Stellenbosch University; 2016.

[23] Lu X, Xie L, Guan H, Huang Y, Lu X. A shear wall element for nonlinear seismic analysis of super-tall buildings using OpenSees. Finite Elem Anal Des 2015;98:14–25. https://doi.org/10.1016/j.finel.2015.01.006.

[24] Shayanfar M, Abbasnia R, Khodam A. Development of a GA-based method for reliability-based optimization of structures with discrete and continuous design variables using OpenSees and Tcl. Finite Elem Anal Des 2014;90:61–73. https://doi.org/10.1016/j.finel.2014.06.010.

[25] Rubert A, Schaumann P. Structural steel and plane frame assemblies under fire action. Fire Saf J 1986;10(3):173–84. https://doi.org/10.1016/0379-7112(86)90014-7.

[26] Jiang L, Usmani A. Computational performance of beam-column elements in modelling structural members subjected to localised fire. Eng Struct 2018;156:490–502. https://doi.org/10.1016/j.engstruct.2017.11.008.

[27] Jeffers AE, Sotelino ED. Analysis of steel structures in fire with force-based frame elements. J Struct Fire Eng 2012;3(4):287–300. https://doi.org/10.1260/2040-2317.3.4.287.

[28] EN1993-1-2, Eurocode 3: Design of steel structures - Part 1-2: General rules - Structural fire design, vol. 2, Britisch Standards Institute, London, 2005.

[29] EN1992-1-2, Eurocode 2: Design of steel structures - Part 1-2: General rules - Structural fire design, vol. 2, Britisch Standards Institute, London, 2004.

[30] Open System for Earthquake Engineering Simulation (OpenSees): https://opensees.berkeley.edu/.).