OpenSees software architecture for the analysis of structures in fire

Citation for published version:

Digital Object Identifier (DOI):
10.1061/(ASCE)CP.1943-5487.0000305

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published in:
Journal of Computing in Civil Engineering, ASCE

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
OpenSees Software Architecture for the Analysis of Structures in Fire

Jian Jiang1; Liming Jiang2; Panagiotis Kotsovinos3; Jian Zhang4; Asif Usmani5; Frank McKenna6; and Guo-Qiang Li7

Abstract: Computational modeling of structures subjected to extreme static and dynamic loads (such as snow, wind, impact, and earthquake) using finite-element software are part of mainstream structural engineering curricula in universities (at least at graduate level), and many experts can be found in industry who routinely undertake such analyses. However, only a handful of institutions around the world teach structural response to fire (at any level) and only a few of the top consulting engineers in the world truly specialize in this niche area. Among the reasons for this are the lack of cheap and easily accessible software to carry out such analyses and the highly tedious nature of modeling the full (often coupled) sequence of a realistic fire scenario, heat transfer to structure and structural response (currently impossible using a single software). The authors in this paper describe how finite-element software can be extended to include the modeling of structures under fire load. The added advantage of extending existing finite-element codes, as opposed to creating fire-specific applications, is due to ability to perform multitasking type analysis, e.g., fire following earthquake. Due to its open source nature and object-oriented design, the OpenSees software framework is used for this purpose. In this work, the OpenSees framework, which was initially designed for the earthquake analysis of structures, is extended by the addition of new concrete classes for thermal loads, temperature distributions across element cross sections, and material laws based on Eurocodes. Through class and sequence diagrams, this paper shows the interaction of these classes with the existing classes in the OpenSees framework. The performance of this development is tested using benchmark solutions of a single beam with infinite stiffness boundary conditions and a steel frame test. The results from OpenSees agree well with analytical solutions for the benchmark problem chosen and provide reasonable agreement with the test. The experience with OpenSees so far suggests that it has excellent potential to be the basis of a unified software framework for enabling computational modeling of realistic fires, and further work is continuing towards the achievement of this goal. The extensions made to OpenSees described in this work, in keeping with the open source ideals of the framework, have been included in the current OpenSees code and are available for researchers and practicing engineers to test, develop, and use for their own purposes. DOI: 10.1061/(ASCE)CP.1943-5487.0000305. © 2014 American Society of Civil Engineers.

Author keywords: Computational modeling; OpenSees; Thermomechanical analysis; Software architecture; Class diagram; Sequence diagram.

Introduction

The traditional approach of evaluating the fire resistance of structures (based on prescriptive building codes) is by testing individual structural members under a standard fire [such as ISO-834 (ISO 1992) and ASTM-E119 (ASTM 2007)], where member capacity is associated with a limiting temperature. This approach does not consider natural fire scenarios and the enormous associated uncertainties, and furthermore the behavior of structural members in isolation entirely ignores the structural interactions a member would experience as part of the whole structure. The unscientific nature of prescriptive approaches has led to gradual and accelerating adoption of so-called performance-based design or, more accurately, performance-based structural engineering (PBSE) approaches, characterized by much greater reliance on scientific understanding and numerical modeling technologies. Admireable research advances have been made towards applying these methodologies in the field of earthquake engineering, most notably the Pacific Earthquake Engineering Research Center—performance-based earthquake engineering (PEER-PBEE) methodology (Deierlein et al. 2003). However, to enable the application of an equivalent PBSE methodology for engineering structural fire resistance, considerable further development of modeling technologies is required. This is because modeling tools for simulating fire, heat transfer to structural components and structural response are typically separate and unconnected due to the significantly different...
Various methodologies and tools have been developed to study the interaction between fire, thermal, and structural models. Ghojel (1998) proposed a simple heat transfer model to simulate temperature profiles of steel structures under real fire conditions accounting for the convective and radiative properties of the main products of combustion. It did not consider the geometrical shape of the enclosure and assumed uniform temperature distribution across or along the elements. A gap radiation model was proposed by Ali et al. (2004) to simulate radiative heat transfer between the gas and the structure surface. It assumed the exposed portions of the structure were totally enveloped by the hot gas. Three-dimensional heat transfer analysis and subsequent 2D structural analyses were performed using the ABAQUS software. Prasad and Baum (2005) proposed an FDS-interface-ANSYS analysis procedure. The interface employed a zone model to manage the data generated by FDS. The zone model divided the compartment into a hot upper and cool lower layer. The properties of the two layers are taken from suitably chosen temporal and spatial average of output generated by FDS. A concept of adiabatic surface temperature (AST) (Wickstrom and McGrattan 2007) was introduced as an efficient interface between the fire model and the structural model. The AST was calculated from the heat flux and gas temperature obtained from the fire mode and then translated back to a net heat flux in the structural model. The advantage was that only one quantity (AST) was transferred, instead of heat flux and gas temperature from the fire model, which was computationally convenient and cost-effective. The method was tested by and FDS-AST-ANSYS simulation of series of compartment fire experiments.

Liew et al. (1998) performed a transient heat transfer analysis using FAHTS (1995) which forms a link between the fire simulation model KFX and the structural analysis program USFOS. The gas temperature at each time step was prescribed in space grids that envelop the structures. The temperature distribution across the element section was calculated by subdividing line beam-column element into a four-node quadrilateral element that can be retrieved by the structural analysis program. Shi et al. (2008) developed an integrated simulation system, BFireSAS, to simulate the overall fire safety performance of large buildings supported by several software tools such as AutoCAD, FDS, and ANSYS. Additional model transformers were created to transfer the gas temperature from FDS to structural analysis solved by ANSYS. AutoCAD was used to construct the geometry of the structural model, FDS to simulate a fire field, and ANSYS for structural analysis. A core database was developed to support the data store and exchange of integrated system and bridge the connection between the different modules. Duthinh et al. (2008) presented two interfaces in fire-thermal-structural analysis. A macroscopic finite-element model was developed by Kodur et al. (2009) for predicting the entire fire response of reinforced concrete structures from fire analysis to structural collapse analysis. Lee et al. (2011) proposed an FDS-interface-ABAQUS analysis. A Matlab subroutine was created as a tool for transferring the FDS temperatures to ABAQUS input. The transfer was conducted by tracing the same coordinate of heat transfer model with the FDS model. A novel fiber-element approach was developed by Jeffers and Sotello (2012) to evaluate the thermosturctural response of nonuniformly heated structural frames. The same fiber discretization in the structural model was used as in the heat transfer model.

Previous studies focused on the development of interfaces between specialist software or commercial packages. Although specialist programs are cost-effective to purchase and easy to use, they lack generality and versatility. In addition, more tellingly continuous development, quality, robustness, and long-term sustainability of such research group–based software must remain in perpetual doubt because of a relatively small number of users.
and developers. The commercial packages have a large library of finite elements and excellent graphical user interfaces (GUIs) to enable efficient and detailed modeling of structural responses to fire and also to allow user subroutines for modeling special features of behavior. Despite obvious advantages, commercial packages require substantial recurring investment for purchase and maintenance that often make them unaffordable for researchers and deter new entrants to the field. Furthermore, the development of commercial codes is not in the hands of the user and users have little control over the direction the development takes. This is usually dictated by the needs of the largest commercial subscribers and rarely addresses the needs of discounted subscription-paying researchers.

An alternative to commercial software is open-source software, where the source code for the software is made available for anyone to download, modify, and use (mostly for free). In successful open-source projects, many outside developers contribute new developments and bug fixes back to the project to further its capabilities. Examples of successful open-source projects include Mozilla Firefox, GNU Linux, and the Apache HTTP server software. In the structural engineering field, OpenSees (McKenna 1997) is an open-source object-oriented software framework developed at the University of California–Berkeley, and supported by PEER and Nees. OpenSees has so far been focused on providing an advanced finite-element computational tool for analysing the nonlinear response of structural and geotechnical systems subjected to seismic excitations. In contrast to algorithm-based programs, object-oriented programs are composed of objects, each with a number of attributes and methods, and can be viewed as the interaction between objects by the sending of messages due to the support of abstraction, encapsulation, modularity, and inheritance (Booch 1994). These features of object-oriented programs make OpenSees computationally efficient, flexible, extensible, and portable (McKenna 1997; Scott et al. 2008). This means a developer can combine and reuse the existing classes in OpenSees to create an application to solve one’s own specific problem. Given that OpenSees is open-source and has been available for best part of this decade, it has spawned a rapidly growing community of users as well as developers who have added to its capabilities over this period. For the analysis of structural and geotechnical systems, it now has capabilities developed by researchers that have yet to appear in commercial software. OpenSees offers the potential of a common community owned research program with large and growing modeling capability in many areas of structural engineering. It will enable researchers to collaborate freely across geographical boundaries with a much greater potential longevity of research and development efforts.

The research team at the University of Edinburgh has been working to add a structures in fire modeling capability in OpenSees. Eventually, this capability will involve a heat transfer model, a structural model and an interface between them to map the temperature data automatically from the heat transfer analysis to the structural analysis, without losing the spatial and temporal resolution of the temperatures when applied to the structural elements. Further work is planned to link OpenSees to the open-source CFD model OpenFOAM (capable of modeling compartment fires) leading to a fully automated software framework for modeling fire, heat transfer, and structural response (Fig. 1).

This paper presents the extensions to OpenSees to enable 2D thermomechanical analysis. This involved creating a new thermal load pattern, modifying existing material classes to include temperature-dependent properties and modifying methods in element and section classes in OpenSees. The algorithm used for thermomechanical analysis of structures is given first, followed by class diagrams describing the hierarchy and architecture of the development in OpenSees. Based on the algorithm and class hierarchy, sequence diagrams are presented to illustrate the interaction between thermal load classes and the material, section, and element classes. The sequence diagrams provide an overview of important aspects of how to apply thermal load and obtain element forces.

**Thermomechanical Algorithm**

In an incremental-iterative nonlinear analysis, three phases can be identified: predictor, corrector, and convergence check (Yang and Kuo 1994). The predictor needs to predict an initial out-of-balance force and calculate the displacement increment due to this unbalanced force, given the stiffness matrix at the previous step. For thermomechanical analysis, in addition to the general external load increment, the unbalanced force should include the equivalent fixed-end force due to thermal load and material softening.
The corrector is concerned with the recovery of element force increment from the displacement increment obtained in the predictor phase. Equilibrium of the structure is checked at the end of each iteration to ensure that convergence is achieved in the new deformed configuration.

**Predictor**

The unbalanced force resulting from thermal load and material softening should be calculated in the predictor phase. The thermal load can be considered as elemental load derived from the temperature distribution along the section. In the finite-element analysis, the elemental load should be transformed into equivalent nodal load.

![Fig. 2. A general section divided into n fibers](Image)

Fig. 2. A general section divided into n fibers

Another source of unbalanced force is the material softening or material degradation due to the increment of the temperature. The imbalance between the applied external load and reduced resisting force leads to further deformation of the structure. Therefore, at the beginning of each thermal load step, the temperature-dependent material properties should be updated, given current temperature, and then the resisting force should be calculated again, given the converged deformation at the previous step using the updated material properties.

The out of balance force $F_u^{1}$ at the beginning of each load step is determined by

$$F_u^{1} = F_{ex} + F_{th} - F_{re} \tag{4}$$

where $F_{ex}$ = external load; $F_{th}$ = elemental thermal force by integration of section force $F_{sec}$ along the element; and $F_{re}$ = updated resisting force due to material softening.

**Corrector**

Once the initial displacement increment is obtained due to the updated out-of-balance force $F_u^{1}$, iterations are needed to determine the converged displacements for the nonlinear problem. In this case, when forming the out of balance force, there is no need to consider thermal force $F_{th}$, i.e.,

$$F_u = F_{ex} - F_{re} \tag{5}$$

where $F_u$ = out-of-balance force calculated for the iterations after the first iteration [i.e., $F_u^{1}$ in Eq. (4)] where the temperature-induced elemental resisting force $F_{th}$ is not considered. Remember that the stress state depends only on the mechanical strain.
where $\varepsilon_{\text{mechanical}}$, $\varepsilon_{\text{total}}$, and $\varepsilon_{\text{thermal}}$ are the mechanical strain, total strain, and thermal strain, respectively. The total strain can be obtained from the strain-displacement relation and the thermal strain can be calculated as $\varepsilon_{\text{thermal}} = \alpha \Delta T$.

With these two modifications, the corrector phase of thermomechanical analysis can follow the general procedure of mechanical analysis of structures (Spacone and Filippou 1992). Fig. 3 shows the flowchart of element state determination for thermomechanical analysis.

### Class Diagrams for Thermomechanical Analysis in OpenSees

In order to implement the aforementioned solution algorithm in OpenSees, new subclasses were implemented and new methods were developed that derive behavior from existing components in OpenSees. These involved creating a new thermal load pattern class, and modifying existing material classes to include temperature dependent properties. Fig. 4 shows the class hierarchy of new classes added in OpenSees using the graphical unified modeling language notation (Booch et al. 1998). The class ThermalLoadPattern was created to store the temperature distribution in the structure and can be used as an interface. It paralleled other load patterns such as earthquake. The temperature distribution stored can be either retrieved from the output of the heat transfer analysis or directly input by the user according to standard codes and experimental data. The data transfer between heat transfer and structural model was designed to account for the disparity in spatial and temporal scales and different element types. One of the functions of ThermalLoadPattern was to call the class Beam2dThermalAction to pass the temperature distribution across the section. It can then be retrieved by the element class such as DispBeamColumn2dThermal and be passed to material classes (e.g., Steel01Thermal) through section classes such as FiberSection2dThermal. The material properties at elevated temperature will be updated corresponding to the temperature input. Beam2dThermalAction can also be used independently to define simple temperature profiles, such as uniform and linearly distributed temperature distribution. The temperature distribution in the structural element can be considered as elemental load. Therefore, the class Beam2dThermalAction defining the temperature distribution in the element was created as a subclass of ElementalLoad. The detailed attributes and implementation of these classes will be presented in the following sections.

### Thermal Load Class

Fig. 5 shows the class diagram of thermal load classes created in OpenSees, and their implementations are shown in Fig. 6.

Thermal load class Beam2dThermalAction was defined as a subclass of ElementalLoad ranked with point load and uniform load. Beam2dThermalAction was created to store the temperature distribution through the depth of the beam section defined by coordinate (LocY) and corresponding temperature ($T$). The temperature of each fiber located along the depth of beam section will be determined by interpolating the temperature at the nearest coordinate point according to its location. At this stage, three kinds of constructors were defined in Beam2dThermalAction to deal with the input of two, five, and nine temperature points through the height of beam section, respectively. Uniform and linearly distributed temperatures can be defined using two temperature points defined at the top and bottom of the section, respectively.

Thermal load pattern ThermalLoadPattern was created to define detailed and highly varying time-dependent temperature distributions in structural members. It can be used as an interface to transfer the temperature distribution from the heat transfer model to the structural model where the structural responses will be predicted. So far, the thermal analysis and structural analysis has been uncoupled in OpenSees, which means that temperature distribution along the element should be provided as input before the structural analysis. Parallel work is under progressing on automatically generating time-varying structural temperature data from a heat transfer analysis within OpenSees (Usmani et al. 2012), however, direct inputs will always be required for modeling of experiments. A series of parameters containing time points and corresponding temperature for the nine temperature points along the height of the section respectively are defined as the input of ThermalLoadPattern. The maximum temperature at each temperature point through the whole fire duration will be defined first and the temperature can then be defined as a ratio of its absolute value to the corresponding maximum temperature. This scheme can accommodate both heating and cooling scenarios.

![Fig. 4. Class diagram for thermomechanical analysis in OpenSees](image-url)
Modified Material Class

There are many types of material models available in OpenSees for steel and concrete, defining their mechanical constitutive relationships, however, some of these needed to be modified to include temperature dependent properties. New temperature-dependent material classes Steel01Thermal (for steel) and Concrete02Thermal (for concrete) were created by modifying the existing material class steel01 and Concrete02. While these new classes share the same stress-strain relations in absence of thermal effects (thus our choice of names), they are not derived from the existing classes, as no reuse of any of the existing class methods was possible. The temperature-dependence added in these two material classes were based on Eurocode stipulations [ENV 1992-1 -2 (Eurocode 2 2004); ENV 1993-1 -2 (Eurocode 3 2004)]. Fig. 7 shows the class

Fig. 5. Class diagram of thermal load classes in OpenSees

Fig. 6. Implementation of functions defined in thermal load classes in OpenSees
Sequence Diagram for Thermomechanical Analysis in OpenSees

The previous section presented a static view of the new classes contributed to OpenSees. To describe how these objects interoperate to conduct thermomechanical analysis, this section presents sequence diagrams showing how to apply thermal load and obtain element resisting force.

Figs. 9 and 10 show the sequence diagrams for applying thermal load to beam element through thermal load classes. The thermal load is applied by invoking method applyLoad() in class ThermalLoadPattern. This method is primarily responsible for two operations. The first responsibility is to retrieve temperature ratio of each temperature point according to current time point from object LinearSeries by calling method getFactor(). The second step is to invoke the method addLoad() in the associated DispBeamColumn2dThermal object to add thermal load to the beam element as shown in Fig. 9. The temperatures and their distributions (dataMix) at current time are calculated and then passed to the section class by invoking the method getTemperatureStress(), which in turn will invoke the method getElongTangent() in the materials. The method getElongTangent() has two operations. One is to update the material properties according to the current temperature. The other function is to send back the temperature-dependent elastic modulus (tangent) and thermal elongation (ThermalElongation) of each fiber material to the section class. These temperature-dependent properties are then used to calculate the force of each fiber through which the section thermal force ($sT$) can be calculated by integration. The thermally induced resisting force of the element can be calculated by integration through sections.

Fig. 11 shows the procedure for obtaining elemental resisting force. As mentioned in Section 2, the out-of-balance force of an element at the beginning of each load step comes from three sources including mechanical load, thermal load and reduced resisting force due to material degradation. In the method getResistingForce(), a parameter (counterTemperature) is set to determine whether it is the first iteration of each load step. If counterTemperature = 0 (means first iteration), the method update() is invoked to update the element state due to the material degradation and the thermally induced resisting force is considered to calculate the total out-of-balance force ($s + sT$). For the next iteration, only mechanically induced out-of-balance force is considered.

Validation

In order to test the performance of the thermomechanical analysis capability in OpenSees, two benchmark cases were carried out, including bending of a single beam with finite-boundary conditions and a steel frame test.

Single Beam Benchmark

Fig. 12 shows the schematic of the single-beam model which is extracted from a framed structure. The beam in the middle...
Fig. 8. Implementation of functions defined in temperature-dependent material classes in OpenSees

```cpp
def ThermalElongation = theMat->getThermalElongation();
strain = strain - ThermalElongation;
```

```cpp
def FiberSection2dThermal
- numFibers
- matData
- kData, sData
- sTData
- TemperatureTangent
+ setTrialSectionDeformation ()
+ getTangentStiff ()
+ getResistingForce ()
+ ...
```

```cpp
for (int i = 0; i < numfibers; ++i) {
    //...
    if (fiberLocs[i] <= dataTemp[3]) {
    }
    //...
    theMat->getTangent(thermals, tangent, ThermoeAdaptive, FiberTempMax);
    double sfTemp = tangent*A*ThermalElongation;
    sData[0] += sfTemp;
    sData[1] += sfTemp * -y;
}
```

```cpp
def Steel01Thermal
- fy: E0: b
- Temp
- ThermalElongation
+ getStrain ()
+ getStress ()
+ getTangent ()
+ setTrialStrainTemperature ()
+ getThermalElongation ()
+ getTangentTangent ()
+ ...
```

```cpp
const Vector &data = theLoad->getData(type, tempFactors(0));
if (type == LOAD_TAG_Beam2dUniformLoad) {
    //...
} else if (type == LOAD_TAG_Beam2dPointLoad) {
    //...
} else if (type == LOAD_TAG_Beam2dThermalAction) {
    dataMix[i] = data[i] * tempFactors[i];
    for (int i = 0; i < numSections; i++) {
        Vector dataMixV(dataMix, 27);
        &st = theSections[i]->getTemperatureStress(dataMixV);
        q0Temperature = st;
    }
```

Fig. 9. Sequence diagram for applying thermal load in thermal load classes

```
class ThermalLoadPattern
    applyLoad (time)
       getFactor (time) = cFactor * time : double
       applyLoad(tempFactors)
```

```
class LinearSeries
    theElement->addLoad (this, tempFactors)
```

```
class ElementalLoad
```

```
class DispBeamColumn2dThermal
```

```
//determine temperature dependent properties
...
else if (TempT <= 180) {
    //...
}
```

```
//Calculate thermal elongation...
else if (TempT <= 730) {
    //...
```

```
return ThermalElongation;
```
of the frame is subjected to uniformly distributed load (UDL) and is restrained by one beam and two columns at both ends. The restraining capability offered by these surrounding elements can be represented by equivalent rotational and translational springs. Therefore, the framed structure can be transformed to an equivalent single-beam with finite end restraints. The dimensions of beams and columns in the Cardington restrained beam test (Kirby 1997) are used (i.e., $305 \times 165 \times 40$ UB for beam and $254 \times 254 \times 89$ UC for column). The corresponding second moment of area of column ($I_c$) and beam ($I_b$) cross section can be calculated as...
The equivalent stiffness of rotational spring \( K_r \) and translational spring \( K_t \) can be calculated as 
\[
K_r = \frac{8E I_c}{l + 4E I_b/l} \quad \text{and} \quad K_t = \frac{2EA_o/l + 48E I_c/l^3}{l^2} \text{with values of } K_r = 4.8 \times 10^5 \text{ kN} \cdot \text{m/rad} \quad \text{and} \quad K_t = 3.8 \times 10^5 \text{ kN/m}. 
\]

Based on fundamental structural mechanics, the analytical solution of the response of the beam subjected to UDL and thermal gradient \( T \) can be given as 
\[
\delta = \frac{5q l^4}{384EI} \left(1 - 0.8\varphi_r\right) + \frac{\alpha T l^2}{8} \left(1 - \varphi_r\right) 
\]
\[
\theta = \left(\frac{q l^2}{12} + E I o T \right) \frac{1}{K_r + 2E I / l} 
\]
\[
u = \alpha \Delta T l (1 - \varphi_r) 
\]
where \( \delta \), \( \theta \) and \( u \) are the mid-span deflection, end rotation and horizontal displacement of the beam respectively; \( \varphi_r = 1/(1 + 2E I/K_r l) \) and \( \varphi_t = 1/\left[1 + (EA_o/l)/K_t\right] \) is a factor due to the rotational and translational end restraint, respectively.

OpenSees was used to analyse the response of a 6-m beam \((l = 3 \text{ m})\) with finite end restraints \((K_r = 4.8 \times 10^5 \text{ kN} \cdot \text{m/rad} \quad \text{and} \quad K_t = 3.8 \times 10^5 \text{ kN/m})\) subjected to UDL and thermal gradient. The UDL is assumed to be 30 kN/m; the temperature at top of the beam was assumed to be 0°C and it varied linearly over the depth of the beam from temperatures of 100°C to 1,000°C. Steel01Thermal was used to model the steel material. A finite large value was assigned to yield stress in order to make the material behavior largely elastic. The elastic modulus at ambient temperature is 200 GPa, and a constant coefficient of thermal elongation \( \alpha = 1.2 \times 10^{-5}/\text{°C} \) was assumed. Three different analyses were conducted, including materially and geometrically linear analysis and materially nonlinear but geometrically linear analysis, as well as both materially and geometrically nonlinear analysis. Material nonlinearity is limited to elastic modulus being dependent on temperature. Fig. 13 shows good agreement

\[\text{Fig. 12. Beam with translational and rotational springs at the ends}\]

\[\text{Fig. 13. Responses of the beam from OpenSees compared with analytical solutions (δ, u:mm; θ:10^{-3} \text{ rad})}\]

\[\text{Fig. 14. Midspan deflection of the beam against temperature}\]

\[\text{Fig. 15. End rotation of the beam against temperature}\]
between OpenSees and analytical solutions using Eqs. (7)–(9) for the linear case.

The responses of the beam subjected to UDL and thermal load for the three analyses in OpenSees are shown in Figs. 14–16. The symbol MatL represents materially linear analysis and MatNL for materially nonlinear analysis. Similarly, GeoL represents geometrically linear analysis and GeoNL for geometrically nonlinear analysis. The material nonlinearity has an obvious effect on the end rotation and horizontal movement of the beam end. In contrast, the effect of the geometrical nonlinearity is obvious on the mid-span deflection but negligible on the rotation and horizontal displacement of the beam. The midspan deflection of the beam continued to increase and experienced a larger slope after about 600°C for nonlinear analysis. This is because the beam deflection, as temperature increases, is dominated by the thermal bowing effect and this downward bending is accelerated by material degradation at high temperature. As shown in Figs. 15 and 16, the rotation and horizontal displacement of the beam increased first driven by the thermal elongation (and thermal gradient) until 500°C and then began to decrease as the decreasing of modulus of elasticity of the beam is unable to resist the stored strain energy and elastic rebound of the unheated rotational and translational spring respectively.

Three-dimensional frame models can also be transformed into an equivalent single-beam model with finite end restraints as long as the stiffness of rotational spring includes the torsional stiffness from the out-of-plane beams connected at the ends.

**Steel Frame Test**

A series of tests on plane steel frames at elevated temperatures were performed in Germany (Rubert and Schaumann 1986). A schematic diagram of two steel frames EHR3 and ZSR1 are shown.

![Fig. 16. Horizontal displacement of movable end of the beam against temperature](image)

**Steel Frame Test**

A series of tests on plane steel frames at elevated temperatures were performed in Germany (Rubert and Schaumann 1986). A schematic diagram of two steel frames EHR3 and ZSR1 are shown.

![Fig. 17. Schematic of the tested steel frames (mm): (a) frame EHR3; (b) frame ZSR1](image)

**Fig. 18. Comparison between predicted and test deflection results: (a) frame EHR3; (b) frame ZSR1**

in Fig. 17. The braced two-bar frame (HER3) was subjected to a uniform temperature rise and only one bay of the two-portal frames (ZSR1) was uniformly heated. All structural elements were made of IPE80 I-shaped steel. The yield stresses and modulus of elasticity are 382 N/mm² and 210 N/mm² at ambient temperature for EHR3 and 355 N/mm² and 210 N/mm² for ZSR1, respectively. Steel01 material class was used to model the properties of the Steel 37 material and nonlinear static analysis was conducted in OpenSees. Comparisons between the predicted deflections and the test results illustrated in Fig. 18 show satisfactory agreement. For this steel frame example, a bilinear material was used to model the realistic steel material in the experiment and reasonable qualitative agreement was achieved. The author consider this to be adequate validation considering that actual test conditions (such as restraint, temperature distributions, and material behavior at elevated temperature) in large-scale thermal testing (as this was) cannot really be fully or accurately represented in models.

Conclusions

The open-source, object-oriented, finite-element-based structural engineering framework OpenSees was extended to perform thermomechanical analysis. The class and sequence diagrams presented provide a logical overview of the hierarchy and relationship between the newly created and modified classes in OpenSees. The thermomechanical analysis capability was tested using two cases including bending of a beam and a steel frame test. Good agreements were achieved between OpenSees and analytical solutions of the beam benchmark test. Reasonable agreement was found against test data. The verification of the extended OpenSees framework is currently limited to two-dimensional cases. The classes described in this work have been added to the OpenSees framework and are available for others to review, download, and use. Further work is being done to extend OpenSees for large deflection of 3D frames including plate and shell elements in fire.

Acknowledgments

Jian Jiang and Liming Jiang were supported by China Scholarship Council and the University of Edinburgh. Jian Zhang is supported by Heriot-Watt University. The work was finalized by Jian Jiang under supervision of Professor Guo-Qiang Li with the financial support of National Natural Science Foundation of China under Grant No. 51120185001.

References


