Develop your own OpenSees Architecture & Components

*SiF Workshop: OpenSees for fire, 3 Dec 2020*

*Presented by Dr Liming Jiang*
Assistant professor
Research Center for Fire Safety Engineering
Department of Building Services Engineering
The Hong Kong Polytechnic University

Special thanks to:
Frank McKenna (University of California, Berkeley)
## Our Roadmap

### Fire Model & Heat Transfer
- By 2020
  - OpenFIRE (OpenSees-FDS Middleware)
  - Heat Transfer (HT) module
    - Tcl/Python Script
    - Idealised uniform fire models (standard, parametric)
    - Idealised non-uniform fire models (localised, travelling fires)

- 2020-2021
  - OpenFIRE (OpenSees-FireFoam Middleware)
  - Travelling fire models
    - Considering travelling mechanism

- After 2021
  - HT sections for composite column
  - HT material for timber sections

### Thermo-mechanical Analysis
- Frame members in fire
  - TM BeamColumn elements (Disp&Force based)
  - Fibre based TM sections
  - Uniaxial materials (concrete & steel)

- Slabs in fire
  - TM Shell elements (ShellMITC4Thermal & ShellNLDKGTQThermal)
  - Layered shell section
  - TM multiaxial material
  - Plate/RebarThermal
  - ConcreteDamagePlasticity

- Beam-Column Joint in fire
  - 3D thermo-mechanical solid elements
  - Continuum elements

### Struc in Fire Application
- SIFBuilder (Integrated Structure in fire simulation tool)
- GiD-OpenSees interface for SIF analyses

- Hybrid Simulation Testing for Structures in Fire

### Algorithm & Solution
- Static analysis
  - Time step, fire duration

- Auto removal solution for failed elements
  - Arc-length solution

- Static-dynamic solution

---

OpenSees Workshop
Openseesforfire.github.io
1. OpenSees for fire website
2. OpenSees Framework
3. Build OpenSees
4. How OpenSees works
5. Interface of material classes
6. Manage a project
7. A summary of OpenSees for Fire
Part 1: OpenSees for Fire Website
Github Website

For Users

User Documentation at Main OpenSees Site

It is important for all users who intend to use the thermal version of OpenSees to first read the user manuals on OpenSees Berkeley Site. A pdf version of the step-by-step instructions on downloading, installing and using OpenSees can be downloaded here.


User Documentation for OpenSees Fire

The following documents are specially developed for OpenSees for fire.

Getting Started with OpenSeesFire

This documentation provides a basic understanding on how to evaluate the structural response in fire using OpenSeesThermal. Relevant Tcl Commands are illustrated by several examples.

Command Manual for OpenSeesThermal

This manual provides a list of all new commands along with its attributes.

OpenSees for fire Examples

Examples are presented to demonstrate the thermomechanical analyses using the thermal version of OpenSees Framework. These are in addition to the simple example discussed in Getting Started with OpenSees Thermal.

Projects of OpenSees for Fire

Heat Transfer in OpenSees

OpenSees now has been extended to heat transfer analysis, which is based on a simple mesh tool with particular emphasis on heat transfer to structural members from fire.

SIFBuilder Project

SIFBuilder is developed as an computational tool to perform integrated analysis of structures in fire

Download

Command Manual

Examples
Download Executable for Windows

The latest version of OpenSees for fire executable file can be downloaded here (Base version 3.0). To use it, please refer to the latest user manual.

Download OpenSeesForFire.exe (Updated on 7 July 2020)

Download Tcl/Tk

OpenSees uses the Tcl interpreter which has been extended with model builder commands. Before using OpenSees, the Tcl library (8.6 version) should be installed and the installation file can be found from this site.

Activate Tcl Website

It should be noticed that the Tcl/Tk is recommended to be installed into C:/Program files/Tcl (NOTE: The current version only supports the 64bit version of Tcl. 32bit version has to be compiled and set up by the user himself)

Download Python

OpenSees has been extended for Python interpreter and can be run as a python application. Due to the inclusion of Python, the original package (Python 3 version) should be installed and the installation file can be found from this site.

Official Python Website

It should be noticed that the Python is recommended to be installed into C:/Program files/Python (NOTE: The current version only supports the 64bit version)

Browse Source Code

Source code on GitHub

To browse our version of source code, you could simply go to our Github page or click the “view on github” link here (Remember to change the branch to OpenSees_SRC)

Download source code package

You can download the source code package through the link shown as below, which is based on the latest version(main release) 3.0, and the build environment is suggested to be Visual Studio 2019 64bit version, Tcl 8.6 64bit, and Python 3.7.

Download source code package: OpenSees3_for_fire.zip

Development Environment
## Source Codes

This branch is 27 commits ahead, 462 commits behind master.

### LimingXLimiting update for timber

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
<th>Date</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER</td>
<td>update</td>
<td>00fa852 on Aug 17</td>
<td>11 months ago</td>
</tr>
<tr>
<td>SRC</td>
<td>update for timber</td>
<td>00fa852 on Aug 17</td>
<td>4 months ago</td>
</tr>
<tr>
<td>Win32</td>
<td>change directories</td>
<td>00fa852 on Aug 17</td>
<td>2 years ago</td>
</tr>
<tr>
<td>Win64</td>
<td>updateHT</td>
<td>00fa852 on Aug 17</td>
<td>5 months ago</td>
</tr>
<tr>
<td>.gitattributes</td>
<td>update</td>
<td>00fa852 on Aug 17</td>
<td>11 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Update .gitignore</td>
<td>00fa852 on Aug 17</td>
<td>5 months ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
<td>00fa852 on Aug 17</td>
<td>11 months ago</td>
</tr>
</tbody>
</table>

README.md
Part 2: OpenSees Framework
A framework is **NOT an executable**;

- It is a set of cooperating software components for building applications in a specific domain;
- It is a collection of **abstract and derived** classes;
- **Loose-coupling** of components within the framework is essential for extensibility and re-usability of the applications.
OpenSees for Fire

- Started at Edinburgh University since 2009;
- Based on a group of PhD students’ work;
- Developed for modelling ‘Structures in Fire’;

**SIFBuilder**

- User-friendly interface for creating (regular) structural models and enable consideration of realistic fire action

**Fire**

- Models of fire action (only *idealised* fires), i.e., Standard fire, Parametric fire, EC1 Localised fire, Travelling fire

**Heat Transfer**

- Heat transfer to the structural members due to fire action

**Thermo-mechanical**

- Structural response to the elevated temperatures
31 Projects in OpenSees

- actor
- cblas
- convergence
- cssparse
- damage
- database
- graph
- handler
- matrix
- modelbuilder
- OpenSees
- OpenSeesTk
- Optimization
- quickMain
- reliability
- renderer
- string
- superLU
- system
- tagged
- utility
- analysis
- domain
- element
- material
- tcl
- recorder
- HeatTransfer
- SIFBuilder
- HTMain
- fire
Part 3: Build OpenSees
Build OpenSees

Step 1: Compilation

project

object1.h
object1.cpp
object2.h
object2.cpp
...

Compiler

project

object1.obj
object2.obj
...

Build OpenSees

**Step 2: Assembling**

- `project`
  - `object1.obj`
  - `object2.obj`
  - ...

- `Assembler`
  - `Assembling`
  - `Project.lib`
  - Static library
  - `...`
  - `Project_1.lib`
  - `Project_2.lib`
  - `...`

**Step 3: Link**

- `Application`
  - `OpenSees`
Build OpenSees

OpenSees Source Code Package

DEVELOPER

EXAMPLES

OTHER

MAKES

WIN32

SRC

PACKAGES

SCRIPTS

Simplified developer
test tool

QuickMain

3rd party Solver

Make tool definition

Project files
for VS

Sourcecode!
.h & .cpp
Makefile

bin

lib

obj

proj

OpenSees.exe

Assembled libraries

Debug/Release compiled objects

Project files

opensees.sln

Solution for Visual studio

Openseesforfire.github.io
Build OpenSees

If you want to build it in Linux or MacOS?

GCC & GNU Make

Makefile.def + SRC + OTHER

make
GNU Make

GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files.

Make gets its knowledge of how to build your program from a file called the *makefile*, which lists each of the non-source files and how to compute it from other files. When you write a program, you should write a makefile for it, so that it is possible to use Make to build and install the program.

Makefile.def

- Program directory
- Paths (definition of SRC and OTHER directories)
- Libraries (definition of library location)
- Compilers (Compiler location & compiler and linker tags)
- Compilation behaviour
- Other supporting libraries
- Include files
Build OpenSees

Using Windows PC

This is what Visual Studio looks like!
Give it a try to build your own OpenSees…
Part 4: How OpenSees Works
# Example 1. portal frame in 2D
# static pushover analysis of Portal Frame, with gravity.
# all units are in kip, inch, second
# elasticBeamColumn ELEMENT
# Silvia Mazzoni & Frank McKenna, 2006
#
# SET UP ---------------------------------------------------------------
wipe;
model basic -ndm 2 -ndf 3;  # 2 dimensions, 3 dof per node
file mkdir data;  # create data directory

# define GEOMETRY ------------------------------------------------------
# nodal coordinates:
node 1 0 0;  # node#, X Y
node 2 0 0 0
node 3 0 432
node 4 504 432

# Single point constraints -- Boundary Conditions
fix 1 1 1 1;  # node DX DY DZ
fix 2 1 1 1;  # node DX DY DZ
fix 3 0 0 0
fix 4 0 0 0

# nodal masses:
mass 1 5.18 0 0.;  # node#, Mx My Mz, Mass=Weight/g.
mass 4 5.18 0 0.

# define ELEMENTS -----------------------------------------------------
# define geometric transformation: performs a linear geometric transformation of beam stiffness and resisting
# force from the basic system to the global-coordinate system
geomTransf Linear 1;  # associate a tag to transformation

# connectivity: (make A very large, 10e6 times its actual value)
$%elasticBeamColumn 1 1 3 36000000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iNode $jNode $A
$%elasticBeamColumn 2 2 4 36000000000 4227 1080000 1

# Define RECORDERS ---------------------------------------------------
recorder Node -file Data/DFree.out -time -node 1 2 3 disp;  # displacements of free nodes
recorder Node -file Data/DBase.out -time -node 1 2 3 disp;  # displacements of support nodes
recorder Node -file Data/DBase.out -time -node 1 2 3 reaction/  # support reaction
recorder Drift -file Data/Drift.out -time -node 1 2 -jNode 3 4 -dof 1 -perpDir 2;  # lateral drift
corder Element -file Data/Force.out -time -ele 1 2 globalForce;  # element forces -- column
recorder Element -file Data/Force.out -time -ele 3 globalForce;  # element forces -- beam
Tcl Script

Node 1 0 0;
Node 2 0 1;

Python Script

Node(1, 0, 0)
Node(2, 0, 1)

C++ function

```c
intTclCommand_addNode(ClientData clientData, Tcl_Interp *interp, int argc, TCL_Char **argv) {
  if (theTclBuilder == 0) {
    opserr << "Want: node nodeTag? \[ndmcoordinates?\] <‐mass \[ndfvalues?\]>
    return TCL_ERROR;
  } 
  Node 1 0 0;
  Node 2 0 1;
  int ndm = theTclBuilder->getNDM();
  int ndf = theTclBuilder->getNDF();
  if (argc < 2+ndm) {
    opserr << "WARNING insufficient arguments\n"
    return TCL_ERROR;
  }
  opserr << "Want: node nodeTag? \[ndm coordinates?\] <‐mass \[ndf values?\]\n"
  return TCL_ERROR;
}
```
# Example 1. portal frame in 2D
# static pushover analysis of Portal Frame, with gravity.
# all units are in kip, inch, second
# elasticBeamColumn ELEMENT
# Silvia Mazzoni & Frank McKenna, 2006

# ---------------------------

# SET UP

wipe;
model basic -ndm 2 -ndf 3;
file mkdir data;

# clear openses model
# 2 dimensions, 3 dof per node
# create data directory

# define GEOMETRY

# nodal coordinates:
node 1 0 0;
node 2 504 0
node 3 0 432
node 4 504 432

# Single point constraints -- Boundary Conditions
fix 1 1 1 1;  # node DX DY RZ
fix 2 1 1 1;  # node DX DY RZ
fix 3 0 0 0
fix 4 0 0 0

...
Inside OpenSees
Model Builder

A simple model
```cpp
theNode = new Node(nodeId, ndf, xLoc, yLoc);
...
if (theTclDomain->addNode(theNode) == false) {
    opserr << "WARNING failed to add node to the domain\n";
    opserr << "node: " << nodeId << endln;
    delete theNode; // otherwise memory leak
    return TCL_ERROR;
}
```
Analysis Commands

constraints Plain;
numberer Plain;
system BandGeneral;
test NormUnbalance 1.0e-3 100 4;
algorithm Newton;
integrator LoadControl 0.005;
analysis Static;
analyze 200;

Analysis

• Algorithm
• Solver
• Analysis type
• Convergence
• Integrator
Analyze command

```c
TclModelBuilder.cpp
Tcl_CreateCommand(interp, "analyze", &analyzeModel,
(ClientData)NULL, (Tcl_CmdDeleteProc *)NULL);
```

result = theStaticAnalysis->analyze(numIncr);

---

Run Analysis

- **Analyze command**
- **TclModelBuilder.cpp**
  - `Tcl_CreateCommand(interp, "analyze", &analyzeModel,
    (ClientData)NULL, (Tcl_CmdDeleteProc *)NULL);`
  
  ```c
  result = theStaticAnalysis->analyze(numIncr);
  ```

---

**Stiffness matrix**

**Unbalanced force**

- Update Domain
- Solve equation
- Check convergence
State Determination

Beam element

Material model for fibre $f_i$

Element state

Fibre section $s_i$

Structure state

Structure model

Fibre $f_i(y_{fi}, z_{fi}, A_{fi})$

State Determination

Material model

($\varepsilon_m, \sigma, E_i, \varepsilon_{th}, T$)
Part 5: Interface of material classes
Material classes

- **Uniaxial materials**
  - Fibre based sections
  - Displacement based / force based

- **nD materials**
  - Shell elements
  - Multi-axial materials
  - Multi-layered plate sections

OpenSees FE model
**Uniaxial material class**

- A material model: Stress-strain relationship

![Graph showing stress-strain relationship]

- Where to find the uniaxial material classes:

  OpenSees/SRC/material/uniaxial
**Uniaxial Material**

uniaxialMaterial ElasticMaterial

User script

TclCommand_addUniaxialMaterial

TclModelBuilderUniaxialMaterialCommand

TclModelBuilder.cpp

ElasticMaterial.cpp and ElasticMaterial.h.

Standard interface of uniaxial materials?

- strain
- stress
- Tangent stiffness

Header file contains declaration of all class functions
Uniaxial Material

class UniaxialMaterial : public Material
{
    public:
        UniaxialMaterial (int tag, int classTag);
    virtual ~UniaxialMaterial();

    virtual int setTrialStrain (double strain, double strainRate =0) =0;
    virtual int setTrialStrain (double strain, double temperature, double strainRate);
    virtual int setTrial (double strain, double &stress, double &tangent, double strainRate = 0.0);
    virtual int setTrial (double strain, double temperature, double &stress, double &tangent, double &thermalElongation, double );

    virtual double getStrain (void) = 0;
    virtual double getStrainRate (void);
    virtual double getStress (void) = 0;
    virtual double getTangent (void) = 0;
    virtual double getInitialTangent (void) = 0;
    virtual double getDampTangent (void);
    virtual double getRho (void);

    virtual int commitState (void) = 0;
    virtual int revertToLastCommit (void) = 0;
    virtual int revertToStart (void) = 0;

    virtual UniaxialMaterial *getCopy (void) = 0;
    virtual UniaxialMaterial *getCopy(SectionForceDeformation *s);

    virtual Response *setResponse (const char **argv, int argc, 
        OPS_Stream &theOutputStream);
    virtual int getResponse (int responseID, Information &matInformation);

    // AddingSensitivity:BEGIN
    virtual double getStressSensitivity (int gradIndex, bool conditional);
    // AddingSensitivity:END
}
Part 6: Interface of element classes
Element classes

- ZeroLength elements
- Truss elements
- Joint elements
- Beam-Column elements
- Shell elements
- Other elements: Brick, contact, soil

OpenSees/SRC/element
**BeamColumn Element**

- **element** `dispBeamColumnThermal`
  - User script
  - `DispBeamColumn2dThermal.h & DispBeamColumn2dThermal.cpp`
  - Apply elemental load (UDL, Thermal load)
  - section deformation
  - Tangent stiffness matrix
  - Resisting force

- `TclCommand_addElement`
- `TclModelBuilderElementCommand`
- `TclModelBuilder.cpp`
- `TclElementCommands.cpp`
class DispBeamColumn2dThermal : public Element
{
public:
    DispBeamColumn2dThermal(int tag, int nd1, int nd2,
                            int numSections, SectionForceDeformation **ss,
                            BeamIntegration &bi, CrdTransf &coordTransf,
                            double rho = 0.0);
    DispBeamColumn2dThermal();
    ~DispBeamColumn2dThermal();

    const char *getClassType(void) const { return "DispBeamColumn2dThermal";};

    int getNumExternalNodes(void) const;
    const ID &getExternalNodes(void);
    Node **getNodePtrs(void);

    int getNumDOF(void);

    void setDomain(Domain *theDomain);

    // public methods to set the state of the element
    int commitState(void);
    int revertToLastCommit(void);
    int revertToStart(void);

    // public methods to obtain stiffness, mass, damping and residual information
    int update(void);
    const Matrix &getTangentStiff(void);
    const Matrix &getInitialStiff(void);
    const Matrix &getMass(void);

    void zeroLoad();
    int addLoad(ElementalLoad *theLoad, double loadFactor);
    int addLoad(ElementalLoad *theLoad, const Vector &loadFactors);

    int addInertiaLoadToUnbalance(const Vector &accel);

    const Vector &getResistingForce(void);
    const Vector &getResistingForceIncInertia(void);
BeamColumn Element

Global displacement (Global coordinate system)

Geometric Transformation

- Linear
- PDelta
- Corotational

Local displacement (Basic system)

Section response (integration points)

Element stiffness & Resisting force (Basic system)

Geometric Transformation

Global stiffness & unbalanced forces (Global system)
Part 7: How to add a project
How to Add a Project

What in a new Project?

**project**

```
object1.h
object1.cpp
object2.h
object2.cpp
...```

- In a header file (.h)
  - Inclusion of other header files
  - Declaration of variables
  - Declaration of functions

- In a source file (.cpp)
  - Inclusion of header files
  - Constructors of class
  - Destructor of class
  - Definition of functions

Prepare the files, and save them in the right folder

OpenSees/SRC/\(<your\ project>\)
How to Add a Project

• Add a new Project to OpenSees

  ❖ Create a project folder in
    OpenSees/win32/proj/<your project>

  ❖ Add this new project
    --if it is completely new, headers and sources
    have to be added;
    --if it is not, files are imported automatically as
    the structure has been defined in the proj file
How to Add a Project

- **Project property**  (right click at the project->configuration properties)

  - Project properties are defined for **debug** and **release** separately
  - Add the dependencies (additional included directories)
    
    subfolders in SRC/<project name>
  - Preprocessor tag( _SIFBUILDER, _HEATTRANSFER)
    
    #ifdef could selectively activate code block
  - output as multi-threaded debug for debugging build
  - multi-thread for release
How to Add a Project
How to Add a Project

Possible Errors

**Compiler**
- Not including right headers
- Deleted variables (destructor)
- Mismatched returned value from a function
- Mismatched constructor and usage of a class
- Incorrect project properties

**Linker**
- Not including right libraries
- Referenced function can not be found because it’s not correctly defined
- Library is not produced
- Linker property of OpenSees project
Part 7: A summary of OpenSees for Fire
OpenSees for Fire

- **SIFBuilder**
  - User-friendly interface for creating (regular) structural models and enable consideration of realistic fire action

- **Fire**
  - Models of fire action (only *idealised* fires), i.e., Standard fire, Parametric fire, EC1 Localised fire, Travelling fire

- **Heat Transfer**
  - Heat transfer to the structural members due to fire action

- **Thermo-mechanical**
  - Structural response to the elevated temperatures
Fire modelling

- Idealised Uniform fire action
  - Standard fire
  - Parametric fire
  - ...

- Idealised non-uniform fire action
  - Localised fires
  - Travelling fires
  - ...

- CFD fire simulation
  - Real fires
  - ...

OPENSEES WORKSHOP
Openseesforfire.github.io
Heat Transfer

- **Heat Conduction**
- **Heat Absorption**
- **Heat Loss**
- **Convection**
  \[ \dot{q} = h(T_s - T_a) \]
- **Radiation**
  \[ \dot{q} = \sigma(T_s^4 - T_a^4) \]

Concrete slab, Steel beam.
Let’s recall the OpenSees framework:

**Finite Element Model**
- Nodes
- Elements
- Boundary conditions
- Load

**Structural Analysis**
- Domain
- Analysis
- Recorder

**Heat Transfer Analysis**
- Domain
- HTAnalysis
- HeatTransfer
- HTCModelBuilder
- HTNodes
- HTElements
- TemperatureBC
- HeatFluxBC
- HTRecorder

**Diagram Notes**
- Let’s recall the OpenSees framework.
- Finite Element Model includes Nodes, Elements, Boundary conditions, and Load.
- Structural Analysis involves Domain, Analysis, and Recorder.
SIFBuilder Project

- **Tcl** supported
  (Tool command language)
- **SIFBuilderDomain** as main storage
- **SIFModel** created for building info
  (material, section, members)

- Various types of Imposed loads
- Various types of fire action
- Automated heat transfer analyses
- Automated implementation of thermal action
About

The OpenSees development for modelling 'structures in fire' was first started at University of Edinburgh in 2009. A couple of students and researchers worked on this long-term project with their own contributions which enable OpenSees to perform heat transfer and thermo-mechanical analyses.

Users

A number of web pages are constructed to offer the users a detailed guidance to the recently added capabilities within OpenSees.

Developers

A detailed description is always new to OpenSees structure and tools are needed for thermo-mechanical analyses in OpenSees.

Documents

Relevant publications to OpenSees for fire development can be found from the above section.

Download

The executable file for OpenSees for fire can be downloaded on this page. The version is updated frequently.

Q&A

If you encounter any problems when using OpenSees for fire, please leave your messages on the "issues" board.

This project is maintained by Liming Jiang

Hosted on GitHub Pages — Theme by orderedlist
Thanks! Any questions?