OpenSees Workshop
Brunel, May 2016

With acknowledgements to:
Jian Zhang, Yaqiang Jiang, Jian Jiang, Panagiotis Kotsovinos, Shaun Devaney, Ahmad Mejbas Al-Remal, & Praveen Kamath & the IIT Roorkee and Indian Institute of Science teams, and China Scholarship Council!

& special acknowledgement to:
Frank McKenna at University of California, Berkeley for OpenSees
1. A Brief Intro & Preparation
2. Basic knowledge about Tcl
3. Getting Started with Standard OpenSees
4. Getting Started with OpenSees for Fire
Day 1: A Brief Intro & Preparation
What is OpenSees?

- The Open System for **Earthquake Engineering** Simulation;
- Developed at **Berkeley** and based on Frank McKenna’s PhD work;
- A software framework based on **finite element method**;
- It is written primarily in the **object-oriented** programming language C++;
- It uses **Tcl Interpreter** to interpret a user-input script.
A Software 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
Where to get OpenSees?

- Berkeley Main Site: http://opensees.berkeley.edu/
  - Download
  - User manual
  - Source code
  - Examples

Standard release
Where to get OpenSees?

- Edinburgh Wiki Site: https://www.wiki.ed.ac.uk/display/opensees

OpenSees
The Open System for Earthquake Engineering Simulation, featured as an object-oriented and open source framework.

About OpenSees at UoE
The OpenSees developers group based in the School of Engineering, University of Edinburgh first started in 2009. The aim of this work is to add a "structures in fire" modelling capability in OpenSees.

Users
A number of wiki pages are provided to help users to carry out thermomechanical analyses with OpenSees using simple examples.

Developers
A detailed description of all the new or modified classes developed for enabling thermomechanical analyses in OpenSees.

Publications
Links to publications by the group are provided here.

Download
An executable version of OpenSees compiled for use in Windows can be downloaded and source codes developed can be browsed or downloaded. We'll update all the bug-fixing issues on that page.
Where to get OpenSees?

- GitHub Site: [http://openseesforfire.github.io/](http://openseesforfire.github.io/)
Before we start...
- **Installation** file can be downloaded from [ActivateState](http://www.activestate.com/) site;

- **Tcl** 8.5 should be installed into a proper directory;

  \[\text{C:}\text{\Program files\Tcl}\]
Preparation

To Use OpenSees

- Tcl installed
- OpenSees executable downloaded
- Tcl script written

- OpenSees.exe placed in the same folder with Tcl script (recommended)
- Run the exe and import the script using source command
Dev Tool

Microsoft Visual Studio

To Develop OpenSees

GCC compiler + Textmate

IDE
Integrated Development Environment
Day 1: How to use OpenSees
To Use OpenSees modelling Structures in fire

- Tcl installed
- OpenSees executable downloaded
- Tcl script written

The same Tcl (32bit)
Our site
Extended commands
To Use OpenSees
modelling
Structures in fire

Thermo-mechanical Analysis

Heat Transfer Analysis

SIFBuilder Aided Integrated Analysis

Part 1

Part 2

Part 3
To get started with OpenSees
(Berkeley standard version)
How does it work?

1. Write your own **tcl script** to build up your model

2. Then it uses **OpenSees interpreters** to read Tcl commands (Tcl version 8.5) for finite element analysis
What is Tcl?

Tcl is a dynamic programming language

- It is a **string** based command language;

- Variables and variable **substitution**;

- Expression evaluation;

- Basic **control structures** (if, while, for, foreach...);

- Procedures, file manipulation, sourcing other files.
Tcl resources

- Tcl documentation link: http://www.tcl.tk/doc/

- Tcl commands manual link:
  http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm
Tcl Syntax Rules

commandName $arg1 $arg2 $arg3 ..

- The first word is the command name
- The remaining words are the command arguments
- A Tcl Script is a sequence of Tcl Commands
- Commands in script are separated by newlines or ;
- The words of a command are separated by white spaces
Tcl Syntax Rules

- `#` code that is **skipped** by the computer, but allows you/someone else to understand what is happening in the code

- `set` a variable is set with a symbolic name used to refer to some location in memory that has a value, such as `set a 2.0`

- `$` to use the value of the variable, such as `set b $a`
Tcl Syntax Rules

- **puts**  
  result sent to screen, usually used for tcl script debugging

- **expr**  
  command is used to calculate mathematical expressions, such as `expr sqrt(($x*$x)+($y*$y))`

- **proc**  
  command is used to create procedures, first arg is your own procedure name, such as:

  ```tcl
  proc sum {a b} {return [expr $a + $b]}
  ```
The OpenSees interpreters are **tcl interpreters** which have been extended to include commands for **finite element analysis**.
# 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;  # clear openses model
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 2 504 0
node 3 0 432
node 4 504 432

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

# nodal masses:
mass 3 5.18 0.0.;  # node#, Mx My Mz, Mass=Weight/g.
mass 4 5.18 0.0.

# define ELEMENTS -- 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, 10^6 times its actual value)
element elasticBeamColumn 1 1 3 3600000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iNode $jNode $A $E $Iz $TransTag
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 442680 1

# Define RECORDERS ---------------------------------------------
recorder Node -file Data/FreeOut.out -time -node 3 4 -dof 1 2 3 disp;  # displacements of free nodes
recorder Node -file Data/FreeOut.out -time -node 1 2 -dof 1 2 3 disp;  # displacements of support nodes
recorder Node -file Data/FreeOut.out -time -node 1 2 -dof 1 2 3 reaction;  # support reaction
recorder Drift -file Data/DriftOut.out -time -iNode 1 2 -jNode 3 4 -dof 1 -perpDirn 2;  # lateral drift
recorder Element -file Data/FC01.out -time -ele 1 2 globalForce;  # element forces -- column
recorder Element -file Data/FFBeam.out -time -ele 3 globalForce;  # element forces -- beam
OpenSees Interpretors

```
# 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, 1000 times its actual value)
element elasticBeamColumn 1 1 3 36000000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $iNode $jNode $A
SE 51x StrainTransf
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 4423680 1

# Define RECODERS ------------------------------------------------
recorder Node -file Data/DFree.out -time -node 3 4 -dof 1 2 3 disp;  # displacements of free nodes
recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 disp;  # displacements of support nodes
recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 reaction;  # support reaction
recorder Drift -file Data/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1 -perpDir 2;  # lateral drift
recorder Element -file Data/FC1.out -time -ele 1 2 globalForce;  # element forces -- column
recorder Element -file Data/FBeam.out -time -ele 3 globalForce;  # element forces -- beam

# define GRAVITY -----------------------------------------------
[pattern Plain 1 Linear {  # distributed superstructure-weight on beam
    eleLoad -ele 3 -type -beamUniform -7.94;  # distributed superstructure-weight on beam
}
constrain Plain;  # how it handles boundary conditions
reorderer Plain;  # renumber dof's to minimize band-width (optimization), if you want to
system BandGeneral;  # how to store and solve the system of equations in the analysis
test NormDispVec 1.0e-6;  # determine if convergence has been achieved at the end of an
iteration step
algorithm Newton;  # use Newton's solution algorithm: updates tangent stiffness at every iteration
integrator LoadControl 0.1;  # determine the next time step for an analysis, # apply gravity in 10 steps
analysis Static;  # define type of analysis static or transient
analyze 10;  # perform gravity analysis
loadConst -time 0.0;  # hold gravity constant and restart time

# define LATERAL load -----------------------------------------------
# Lateral load pattern
[pattern Plain 2 Linear {  # node 1, F Y Z --- representative lateral load at top nodes
    load 5 2000. 0.0 0.0;
    load 4 2000. 0.0 0.0;  # place 1/2 of the weight for each node to get shear coefficient
}

# pushover: displacement controlled static analysis
integrator DisplacementControl 9;  # switch to displacement control, for node 11, dof 1, 0.1 increment
analyse 100;  # apply 100 steps of pushover analysis to a displacement of 10
puts "Done!"
```
OpenSees Interpretors

The OpenSees interpreters are **tcl interpreters** which have been extended to include commands for **finite element analysis**

- **Modeling** – create nodes, elements and constraints
- **Analysis** – specify the analysis procedure.
- **Output** – specify what it is you want to monitor during the analysis.
A simple model of an elastic portal frame. The objective of this example is to give an overview of input format.

An 2D portal frame example from Berkeley website
# 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;  # clear opensees model
model basic -ndm 2 -ndf 3;  # 2 dimensions, 3 dof per node
file mkdir data;  # create data directory
source DisplayPlane.tcl;  # procedure for displaying a plane in model
source DisplayModel2D.tcl;  # procedure for displaying 2D perspective of model
Building up the model

```plaintext
*define GEOMTRY

*nodal coordinates:
node 1 0 0;  # node#, X Y
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

*nodal masses:
mass 3 5.18 0. 0. 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
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)
element elasticBeamColumn 1 1 3 3600000000 4227 1080000 1;  # element elasticBeamColumn $eleTag $in
SA $2 $1z $transfTag
element elasticBeamColumn 2 2 4 3600000000 4227 1080000 1
element elasticBeamColumn 3 3 4 5760000000 4227 4423680 1
```
Define recorders

```plaintext
42 # Define RECORDERS
43 # displacements of free node
44 recorder Node -file Data/DFree.out -time -node 3 4 -dof 1 2 3 disp;
45 # displacements of support
46 recorder Node -file Data/DBase.out -time -node 1 2 -dof 1 2 3 disp;
47 # support reaction
48 recorder Node -file Data/RBase.out -time -node 1 2 -dof 1 2 3 reaction;
49 # lateral drift
50 recorder Drift -file Data/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1 -perpDirn 2;
51 # element forces -- column
52 recorder Element -file Data/FCol.out -time -ele 1 2 globalForce;
53 # element forces -- beam
54 recorder Element -file Data/FBeam.out -time -ele 3 4 globalForce;
55 # view the deformed shape
56 set ViewScale 5;
57 DisplayModel2D DeformedShape SViewScale ;
58 # display deformed shape, the scaling factor needs to be adjusted for each model
```
Define gravity & analysis

```plaintext
# define GRAVITY .................................................................

pattern Plain 1 Linear {
  eleLoad -ele 3 -type -beamUniform -7.94 ; # distributed superstructure-weight on beam
}

constraints Plain;
numberer Plain;
system BandGeneral;
test NormDispIncr 1.0e-6 6 ; # iteration step
algorithm Newton;
iteration
integrator LoadControl 0.1;

analysis Static
analyze 10;
loadConstr -time 0.0;
```

# how it handles boundary conditions
# renumber dofs to minimize band-width (optimization), if you:
# how to store and solve the system of equations in the analysis
# determine if convergence has been achieved at the
# use Newton's solution algorithm: updates tangent stiffness
# determine the next time step for an analysis, # apply
# define type of analysis static or transient
# perform gravity analysis
# hold gravity constant and restart time
Define lateral load & analysis

```ruby
# define LATERAL load
# Lateral load pattern
pattern Plain 2 Linear {
  load 3 2000.0 0.0 0.0;  # node#, FX FY MZ -- representative lateral load at
  load 4 2000.0 0.0 0.0;  # place 1/2 of the weight for each node to get shear
}

# pushover: displacement controlled static analysis
integrator DisplacementControl 3 1 0.1;  # switch to displacement control, for node
analyze 100;  # apply 100 steps of pushover analysis to a displacemen
puts "Done!"
```
### Analysis results can be found in:

```
.../bin/data
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

First column is the pseudo-time
To run a thermo-mechanical analysis in OpenSees...
Using OpenSees

Modules extended for Thermo-mechanical analyses

Temperature data?

User Input

Temperature distribution

2D/3D Heat transfer

FireLoadPattern (or universal pattern)

Thermal action (zLoc, yLoc, T)

Applying thermal action?

Fibre-based section (zLoc, yLoc, Area, σ, ε, E, T)

Thermo-mech materials (σ, ε, E, T)

Node i

Thermo-mech element

[DispBeamColumn2d(3d)Thermal]

Node j

BeamColumn element
Modules extended for Thermo-mechanical analyses

- **Node i**
- **Integration point**
- **Node j**

**Fiber**\((y_{Loc,0}, area, maTag)\)

**FiberSection2dThermal**

**Thermal Action**

- Temperature history

**Element**

- Implementation of thermal action

**Section**

- Temperature distribution

**Material**

- Temperature dependent properties

**Uniaxial Material** (Thermal)
Getting Started!

Files in the folder:
- DisplayModel2D.tcl
- DisplayModel3D.tcl
- DisplayPlane.tcl
- example-SteelBeam-uniform.tcl
- Wsection.tcl

A beam example:
- Download from website
- Place the Tcl script with OpenSees

Thermo-mechanical analyses:
- A steel I-section beam
- Simply supported
- UDL q=8N/mm
- Uniform temperature
wipe;
model BasicBuilder -ndm 2 -ndf 3;
source Wsection.tcl;
source DisplayPlane.tcl
source DisplayModel2D.tcl
node 1 0 0;
node 2 1000 0;
node 3 2000 0;
node 4 3000 0;
node 5 4000 0;
node 6 5000 0;
node 7 6000 0;
fix 1 1 1 0;
fix 7 0 1 0;
Using OpenSees

example-SteelBeam-uniform.tcl

uniaxialMaterial SteelECThermal 1 308 2.1e5;

set d 355; set bf 171.5;
set tf 11.5; set tw 7.4;
set nfdw 8; set nfw 1;
set nfbf 1; set nftf 4;

Wsection 1 1 $d $bf $tf $tw $nfdw $nfw $nfbf $nftf

Thermo-mechanical material

I-section geometry

Call the pre-defined I section script

section fiberSecThermal $secID {
  fiber $yLoc $zLoc $A $matTag
  patch quad $matTag $nIJ $nJK $yI $yJ $zI $zJ $yK $zK $yL $zL
}

Fibre based I-section
Using OpenSees

example-SteelBeam-uniform.tcl

recorder Node -file 1.out -time -node 1 -dof 1 2 3 disp;

pattern Plain 1 Linear {

  eleLoad -ele 1 -type -beamUniform $UDL 0
  ....
  eleLoad -ele 6 -type -beamUniform $UDL 0
}

constraints Plain;

integrator LoadControl 0.1;

analysis Static;

analyze 10;

puts "Fire";

loadConst -time 0.0

Reset Pseudo time  ts=0~1;
set HalfD [expr $d/2];

pattern Plain 1 Linear {
    eleLoad -range 1 6 -type -beamThermal 1000 -$HalfD 1000 $HalfD;
}

constraints Plain;
..
integrator LoadControl 0.01;
analysis Static;
analyze 100;
Using OpenSees

Thermal Action Definition

Beam2dThermalAction

Beam3dThermalAction

ShellThermalAction
To run a Heat Transfer analysis in OpenSees...
Let's recall the OpenSees framework.

Using OpenSees

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

Structural Analysis
- Tk interpreter
- ModelBuilder
- Analysis
- Recorder

Heat Transfer Analysis
- Tk interpreter
- HT ModelBuilder
- HTNodes
- HTElements
- TemperatureBC
- HeatFluxBC
- HTAnalysis
- HeatTransfer Domain
- HTRecorder
A typical heat transfer problem in structure
Development for Heat Transfer Analysis

Heat transfer model

Simple mesh tool

HT nodes
HT elements
Temperature BC
Heat flux BC

Fire model

Gas temperature
or
Incident heat flux

Convection
Radiation
Prescribed heat flux
Heat transfer analysis flowchart

Facilities (e.g. solver, numberer, etc.) for heat transfer analysis may need to be defined, otherwise the default ones will be assigned.
Using OpenSees

Getting Started!

Files in the folder
HT_demo.tcl

Composite section

- Download from website
- Place the Tcl script with OpenSees

Heat transfer analyses

- A steel I-section connected to a concrete slab section
- Heat loss at the slab top
- Fire beneath the composite section

0.1m

0.4m

0.2m
wipe;
HeatTransfer 2D;

HTMaterial CarbonSteelEC3 1;
HTMaterial ConcreteEC2 2 0.0;

HTMaterial $materialType $materialTag <$par_1...$par_n>

HTEntity Isection 1 0.0 0.2 0.2 0.40 0.02 0.02;
HTEntity Block 2 0.0 0.45 0.6 0.1;

HTEntity $EntityType $EntityTag $centre_x $centre_y $dim_1
<$dim_2..$dim_n>
# Using OpenSees

## HTEntity Definition

Table A.1: Commands for creating a heat transfer entity

<table>
<thead>
<tr>
<th>Type</th>
<th>HTEntity</th>
<th>Centroid</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D</td>
<td>Line</td>
<td>$x_c$</td>
<td>$l$</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>$x_c, y_c$</td>
<td>$b, d$</td>
</tr>
<tr>
<td>2D</td>
<td>Isection</td>
<td>$x_c, y_c$</td>
<td>$b_f, h_b, t_w, t_f$</td>
</tr>
<tr>
<td></td>
<td>IsectionPro</td>
<td>$x_c, y_c$</td>
<td>$b_f, h_b, t_w, t_f, c$</td>
</tr>
<tr>
<td></td>
<td>Composite</td>
<td>$x_c, y_c$</td>
<td>$b_f, h_b, t_w, t_f, b$</td>
</tr>
<tr>
<td>3D</td>
<td>Isection3D</td>
<td>$x_c, y_c, z_c$</td>
<td>$b_f, h_b, t_w, t_f$</td>
</tr>
<tr>
<td></td>
<td>Composite3D</td>
<td>$x_c, y_c, z_c$</td>
<td>$b_f, h_b, t_w, t_f$</td>
</tr>
</tbody>
</table>

![Diagram of heat transfer entity](image.png)
Using OpenSees

HT_demo.tcl

HTMesh 1 1 1 -phaseChange 0 -MeshCtrls 0.01 0.005 0.005 0.014
HTMesh 2 2 1 -phaseChange 1 -MeshCtrls 0.02 0.02

HTRefineMesh -Entity 2 -SeedTag 1 4 -space 0.02 10 0.01 9 0.005 4 0.01 9 0.02 10;

HTMeshAll;

SetInitialT 293.15;

HTNodeSet 1 -Entity 1 -face 12;
HTNodeSet 2 -Entity 2 -face 1 -locx -0.1 0.1;

HTCoupleT -NodeSet 1 2;
HTConstants 1 4.0 293.15 0.7 5.67e-8 0.7;
HTConstants 2 25.0 293.15 0.7 5.67e-8 0.7;

HTPattern AmbientBC 1 {
    HeatFluxBC -HTEntity 2 -faceTag 4 -type ConvecAndRad -HTConstants 1;
}

HTRecorder -file temp0.out -NodeSet 1;
HTRecorder -file temp1.out -NodeSet 2;

HTAnalysis HeatTransfer
HTAnalyze 20 30;

wipeHT;
```tcl
FireModel standard 1;

HTNodeSet 3 -Entity 2 -Locx -0.3 -0.1;
HTEleSet 1 -Entity 2 -NodeSet 3 -face 1;
HTNodeSet 4 -Entity 2 -Locx 0.1 0.3;
HTEleSet 2 -Entity 2 -NodeSet 4 -face 1;

HTPattern fire 2 model 1 {
    HeatFluxBC -HTEntity 1 -face 1 4 5 6 7 8 9 -type ConvecAndRad -HTConstants 2;
    HeatFluxBC -HTEleSet 1 -face 1 -type ConvecAndRad -HTConstants 2;
    HeatFluxBC -HTEleSet 2 -face 1 -type ConvecAndRad -HTConstants 2;
}
```
HTRecorder -file temp0.out -NodeSet 1;
HTRecorder -file temp1.out -NodeSet 2;

HTAnalysis HeatTransfer
HTAnalyze 20 30;

wipeHT;
To run a SIFBuilder aided analysis in OpenSees...
SIFBuilder is an **unified** tool for performing **automated** structural fire analysis for **large** structures under **realistic** fire.
SIFBuilder workflow

Model Type → Geometric information → Material definition → Section definition → Boundary conditions

- **SIFBay**: xb1, xb2, ...
- **SIFMaterial**: steel, concrete, etc.
- **SIFSection**: I section, slab section, etc.
- **SIFBC**: Fixed, pinned, etc.

Start analysis → Data request → Fire → Load

- **SIFAnalyze**: Load, timestep; Fire, duration, timestep
- **SIFRecorder**: Nodal displacement, element forces, etc.
- **SIFFire**: Uniform fire, localised fire, etc.
- **SIFLoad**: Point load, UDL, etc.
2x2x2 Frame with slab

A large model of 2x2x2 frame with slab, under the localised fire / compartment fire.

An 2x2x2 frame example from Edinburgh wiki
2x2x2 Frame with slab

Geometry of the structure (bay lengths in each direction and storey heights in a Cartesian coordinate system);

```tcl
source DisplayPlane.tcl;  # procedure for displaying a plane in model
source DisplayModel2D.tcl;  # procedure for displaying 2D perspective of model
source DisplayModel3D.tcl

file mkdir HTData;  # define the directory for storing data
SIFBuilder;  # initialise SIFBuilder, (SIFBuilder frame) is accepted for defining frame only wi

#[BUILDING INFO]
SIFXBay 6 9;  # XBAY SPAN|<-----6m------>|<-------9m------>| along global x direction
SIFZBay 6 9;  # ZBAY SPAN|<-----6m------>|<-------9m------>| along global z direction
SIFStorey 5 4;  # Storey Height|<-----5m---->|<-----4m-->| along global y direction
```
2x2x2 Frame with slab

Material type and cross section type for the structural members;

```
[DEFINE MATERIAL AND SECTION]
AddMaterial steel 1 -type EC3 3e8 2e11; # E0 : 3e8 , fy: 2e11, EN-1993-1-2 Steel Mat
AddMaterial concrete 2 -type EC2 0 30; # moisture ratio:0 , fc :30, EN-1992-1-2 Concre
AddSection ISection 1 1 0.203 0.102 0.0054 0.009; # Sd $bf $tw $tf UB203x102x23
AddSection ISection 2 1 0.203 0.203 0.007 0.011; # Sd $bf $tw $tf UC203*203*46
AddSection SlabSection 3 2 0.1; #

ASSIGN SECTION]
AssignSection beams 1;
AssignSection columns 2;
AssignSection slabs 3;
```
**Boundary conditions** for the structural model, and define the **structural loading & Fire**.

```python
# [DEFINE BC AND LOAD]
SetBC fixedJoint -locy 0;  # set boundary condition
AddLoad -member allslabs -load 0 -1000 0;
AddFire -compartment 111 -type standard;
```
Mesh control for the thermo-mechanical analysis, and define the display.

```
# [BUILD MODEL]
BuildModel -MeshCtrl 6 6 6;     #Number of Els meshed for each member (along gl

# [Define DISPLAY]
set xPixels 800;    # height of graphical window in pixels
set yPixels 600;    # height of graphical window in pixels
set xLoc1 100;      # horizontal location of graphical window (0=upper left-most
set yLoc1 60;       # vertical location of graphical window (0=upper left-most
set ViewScale 1;    # scaling factor for viewing deformed shape, it depends on
DisplayModel3D DeformedShape $ViewScale $xLoc1 $yLoc1 $xPixels $yPixels 0
```
Define recorder and analysis

```plaintext
# [Define SIFRECORDER]
SIFRecorder Joint -file Joint111.out -joint 111 disp;  #Def
SIFRecorder Member -file XBeam111.out -xBeam 111 Midefect;  #Rec
SIFRecorder Member -file Slab111.out -slab 111 Midefect;  #Rec

# [RUN ANALYSES]
SIFAnalyze Load -dt 0.2 Fire -dt 30 -duration 1800 -output HTData;  #I
print domain.out
```
SIFBuilder exercises

- Change bay dimensions, material parameters, section types, etc
- Change standard fire durations
- Add standard fire to several compartments
- Apply localized fire at the centre column