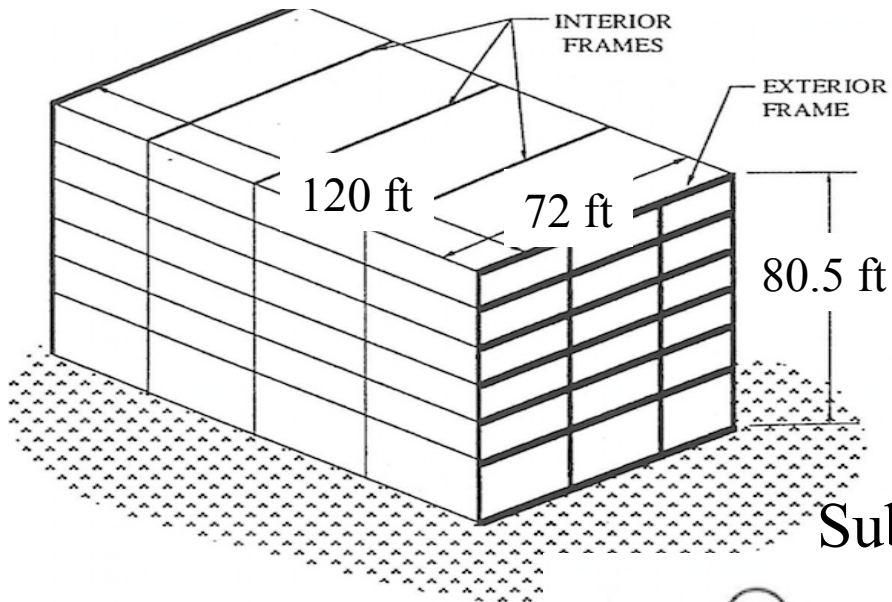


TO DO WHILE WAITING:

- If not already a NEES register at <https://nees.org/register/>
- After logging in launch the openseeslab tool: <https://nees.org/tools/openseeslab>
- If you are prompted for java update , you will need to update java.
- My presentation can be downloaded from:
<http://opensees.berkeley.edu/Workshops/OpenSeesWorkshop.pdf>



Max Roof Displacement = ?

Dead Load: 95psf typical, roof 80psf
 $E=29,000$, $F_y=50.0$, $b=0.003$

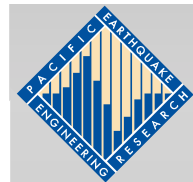
Subjected to el_centro 1940 N-S (Peknold)

	A	I _{xx}	ROOF	(A)	(B)	(C)	(D)	
W14X193	56.8	2400	6	W14 x 109	W24 x 104	W24 x 76	6th Story	12.5 ft
W14X159	46.7	1900	5			W24 x 76		
W14X109	32.0	1240	4		W27 x 146	W27 x 94	24 ft	
W30X173	51.0	8230	3			W27 x 94		
W27X146	43.1	5660	2		W30 x 173	W30 x 99	2nd Story	
W24X104	30.6	3100	1			W30 x 99		
W30X99	29.1	3990	G			1st Story	18 ft	
W27X94	27.7	3270						
W24X76	22.4	2100						

This figure is taken from the first paper in the SAC 95-05 report: Hall, John F. "Parameter Study of Response of Moment-Resisting Steel Frame Buildings to Near-Source Ground Motions"

OpenSees Workshop

Frank McKenna
UC Berkeley



Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Before We Begin: (A Disclaimer!)

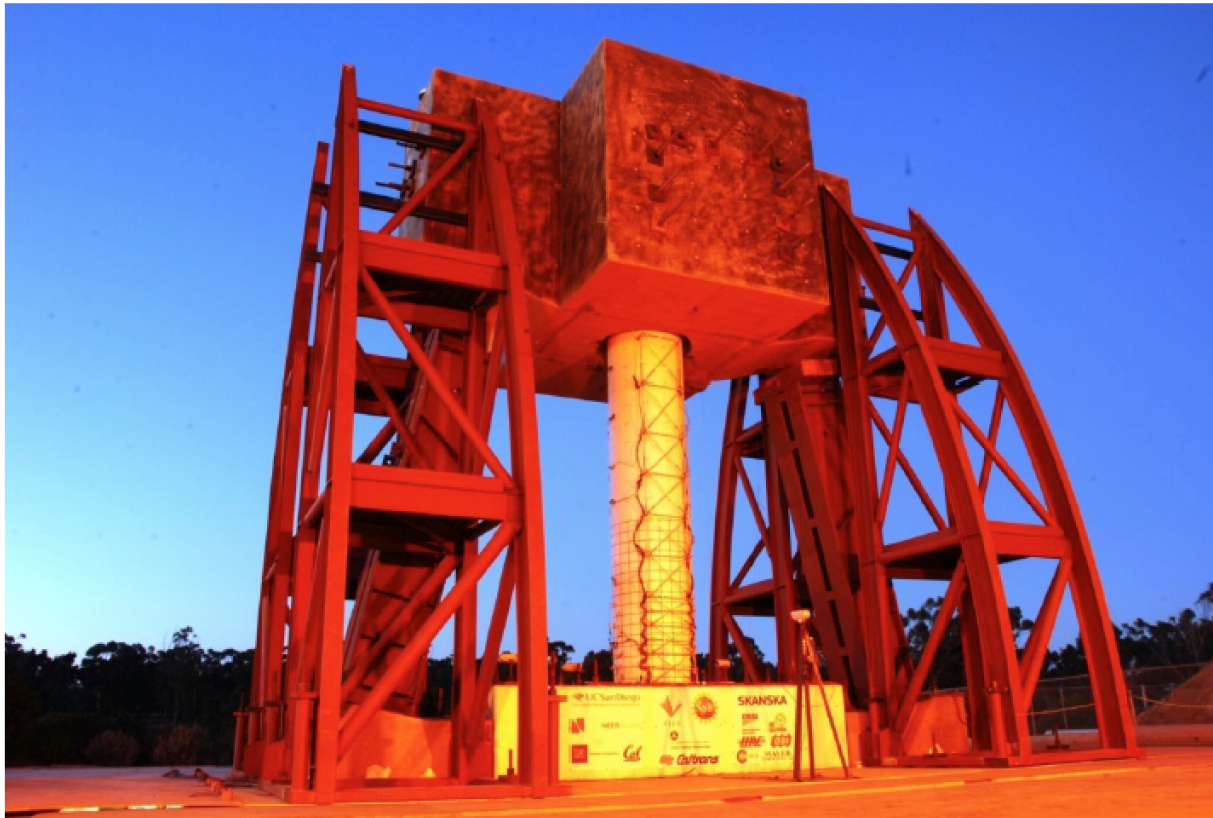
What is the Purpose of Simulation?

To develop a model capable of producing a credible prediction of the behavior of an existing/proposed structure.

And Just How Good are we Today at predicting this response?

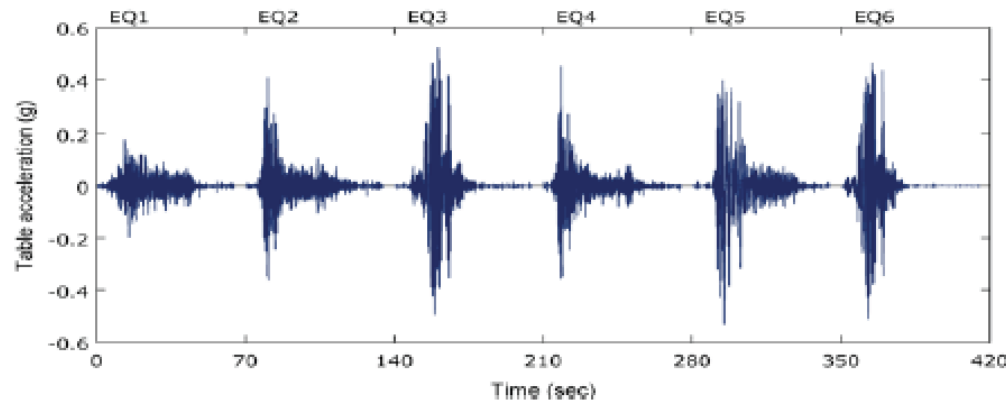
Today:

PEER Bridge Column Benchmark @ UCSD

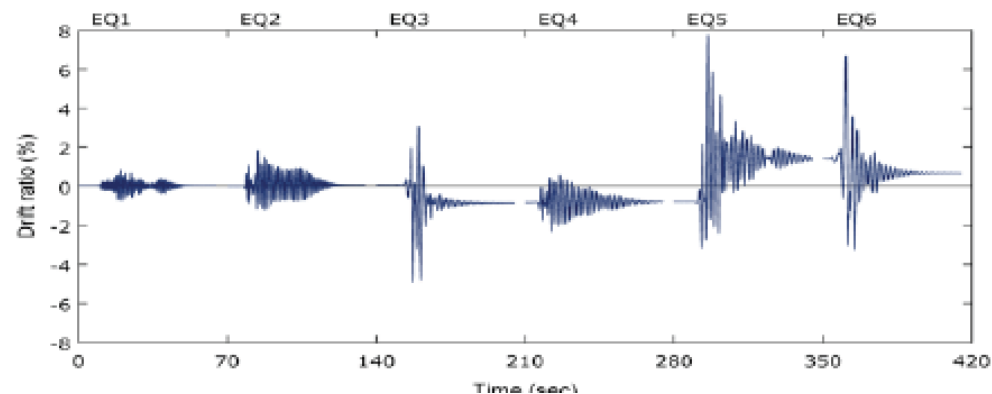


Today: PEER Bridge Column Benchmark @ UCSD

- Earthquake acceleration time histories



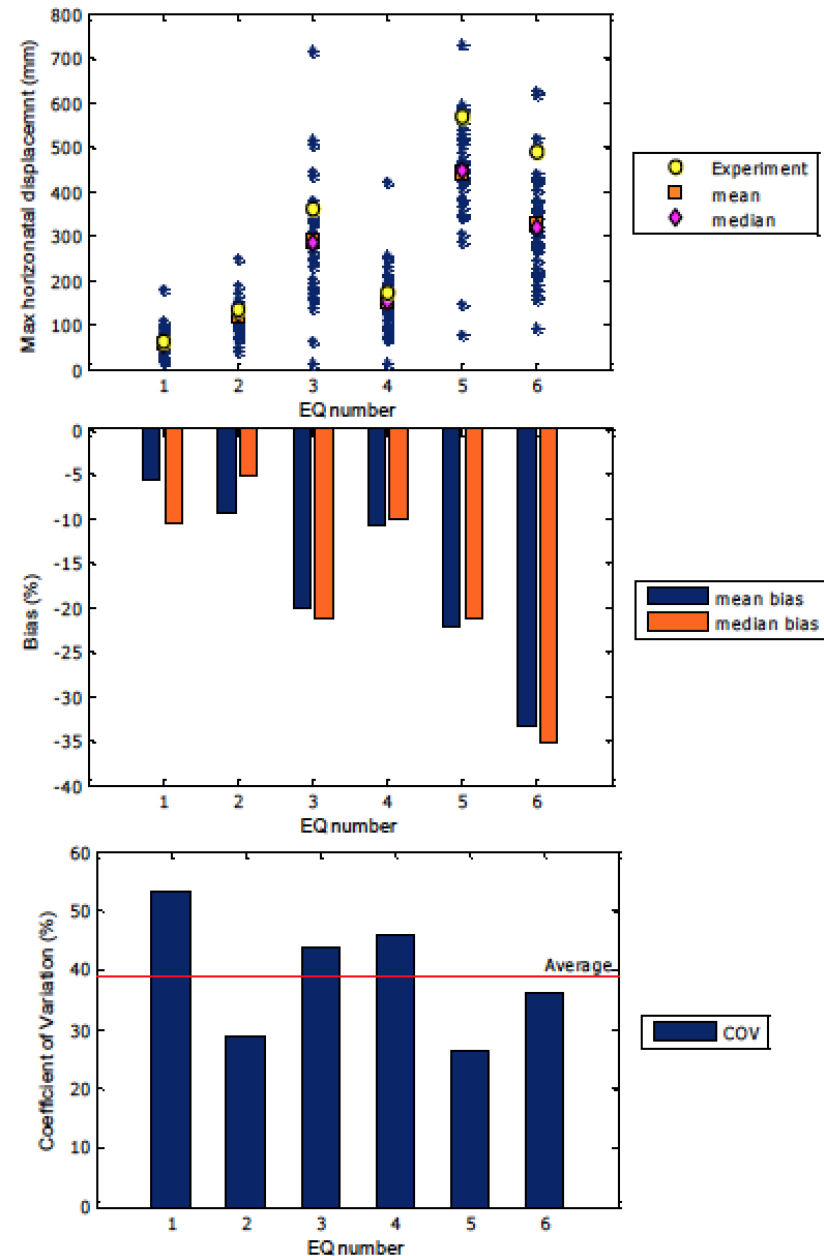
- Drift ratio time histories



Today

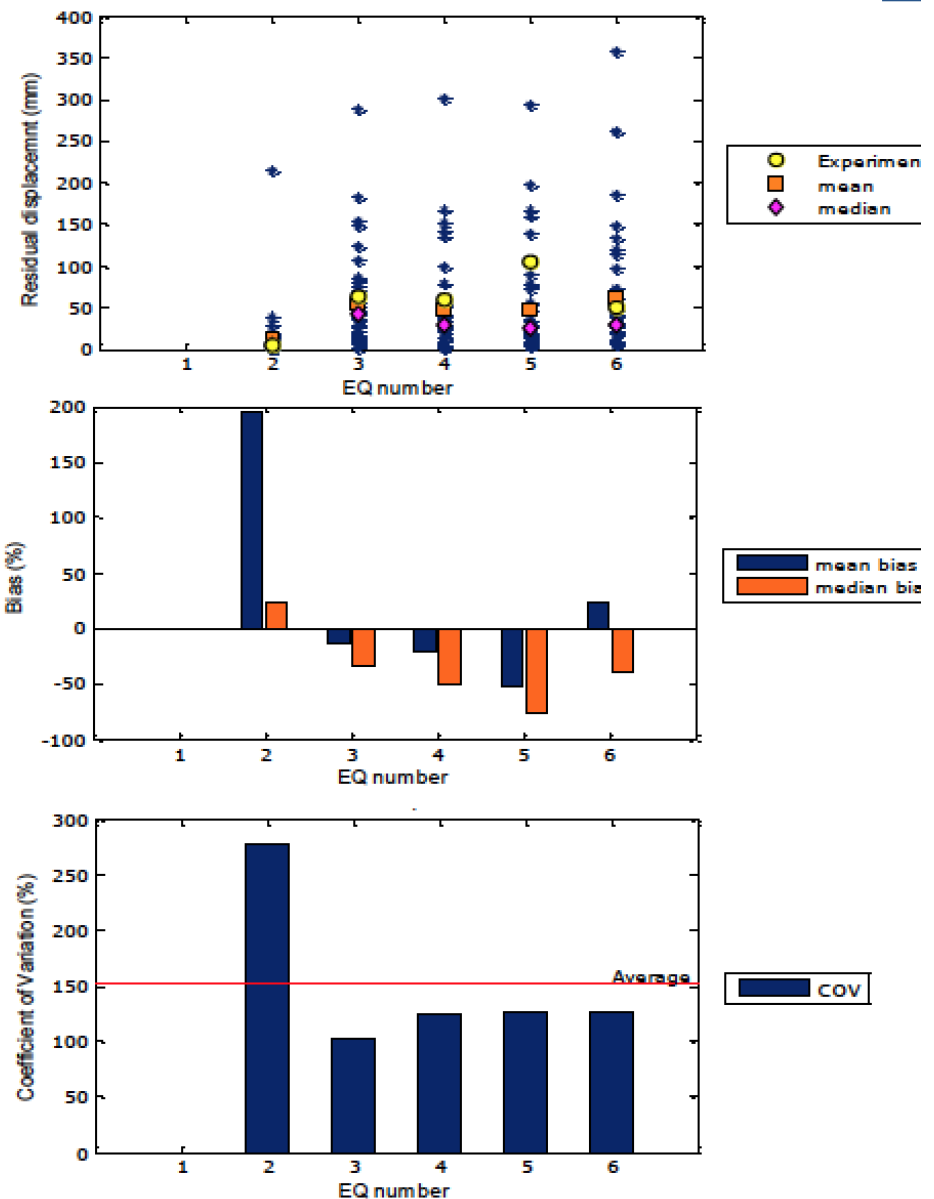
- Prediction of the relative displacement of the top of the column:
 - Very large scatter among the 41 simulations
 - Bias is negative (model underestimates the benchmark) and growing with increase of earthquake intensity
 - CoV is, on average, 39%

source: Prof. B. Stojadinovic



Today

- Prediction of the residual displacement of the top of the column:
 - HUGE scatter among the 41 simulations
 - Bias is mostly negative (model underestimates the benchmark) and there is no pattern
 - CoV is, on average, larger than 100%



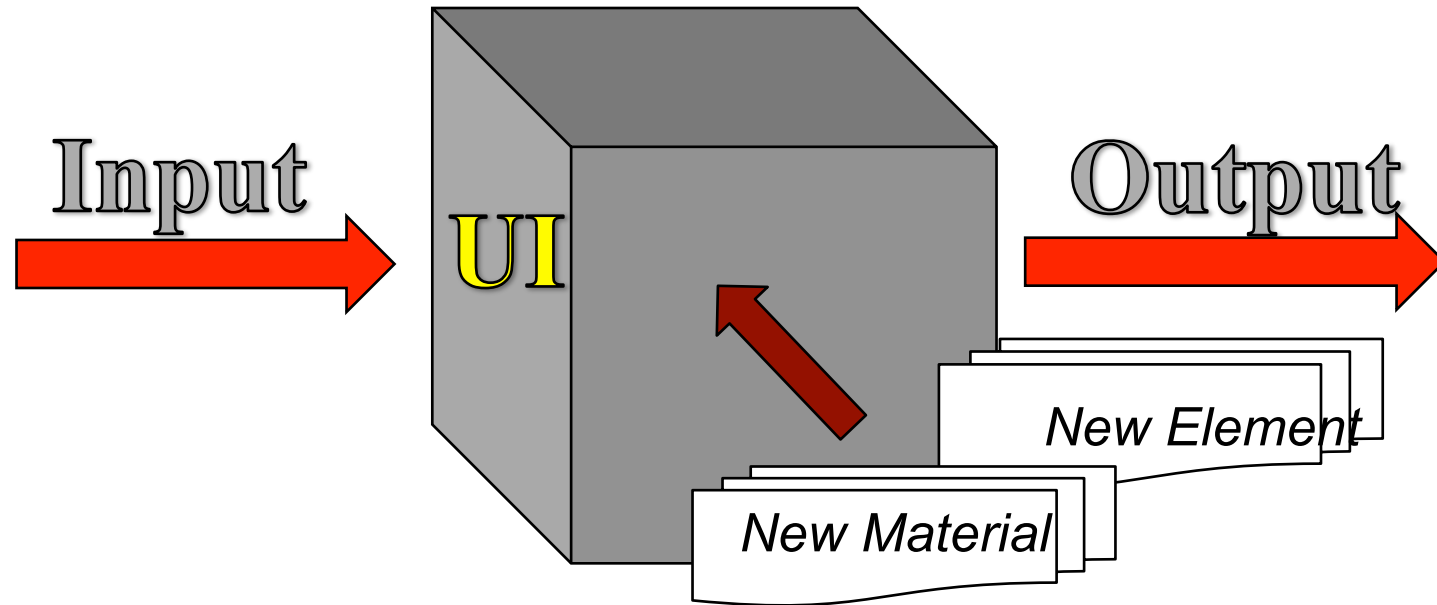
source: Prof. B. Stojadinovic

Current Status (simulation codes)

- Individual models (materials/elements) are typically well verified.
- They are typically validated against few experimental tests over a narrow performance range.
- Computer simulations of systems do not necessarily provide credible predictions and typically only provide a single deterministic response.

“The uncertainty is as important a part of the result as the estimate itself... . An estimate without a standard error is practically meaningless.” [Jeffreys (1967)]

What Was/Is Wrong With Existing Software



- Embedding of computational procedures in codes makes it difficult to experiment (integration schemes, algorithms, solvers) and take advantage of computing technology (Parallel & Grid Computing)
- “Closed-source” was/is the norm, whereas other fields have adopted “open-source” software for communities of users. Closed source is an impediment to new research. No sharing of code, building upon others ideas.

OpenSees

Open System for Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center

- I started writing code in 1995 so I could do my research.
- It has a very modular design.
- OpenSees has been under development by PEER since 1998. NEES supported since 2005.
- Windows application downloaded over 10,000 times a year.
- Parallel Applications utilize over 1,000,000 CPU hours on NSF XSEDE compute resources yearly.
- Open-source and royalty free license for non-commercial use and internal commercial use.
- License must be obtained for software developers including OpenSees code in their applications.
- Written in C++, C and Fortran (C++ being the main language)

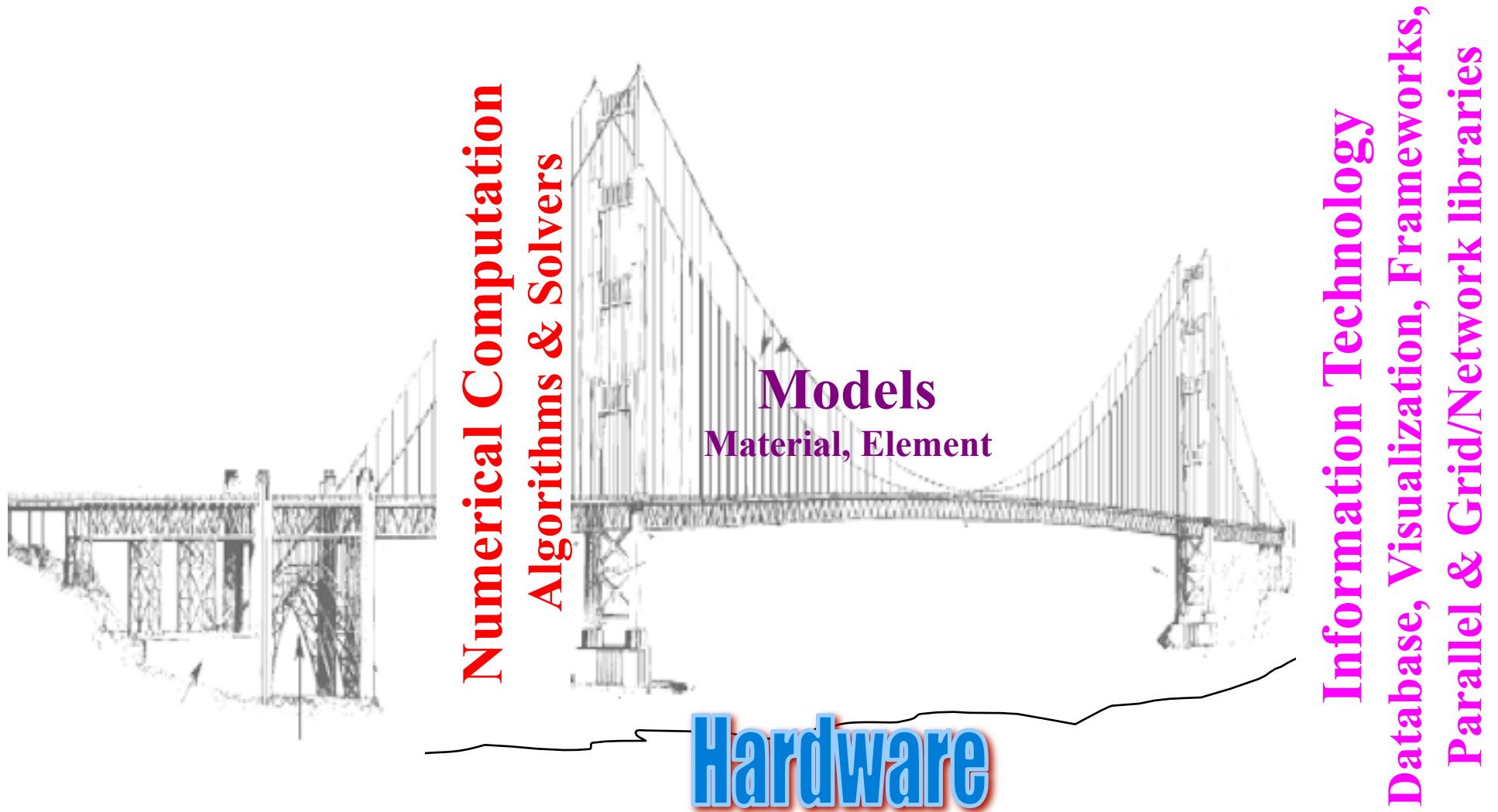
<http://opensees.berkeley.edu>

The screenshot shows the OpenSees website homepage. At the top, there is a navigation bar with links for HOME, USER, DEVELOPER, SUPPORT, PARALLEL, Copyright, and SITEMAP. Below this is a secondary navigation bar with links for About, News, Calendar, and Registration. The main content area features several articles and announcements:

- OpenSees Days 2012 & Call for Presentations**: A notice that NEES and PEER are hosting a two-day event on August 15-16 at the Richmond field Station, UC Berkeley. The event is free, but registration is required. A link is provided for registration and program information.
- Discovering OpenSees**: A notice about the next seminar in the web-based "Discovering OpenSees: Surfing the waves of OpenSees" learning series, which will occur on May 31, 2012. The session is titled "Parallel and Grid Computing With OpenSees" and will occur on May 31 at 4:00 PM and June 1 at 10:00 AM Pacific Time.
- OpenSees Challenge 2012**: A notice that this year they will be awarding two prizes:
 - an iPod to the developer of the best OpenSees powered tool added to NEEShub between the OpenSeesDays 2011 and the OpenSees Days 2012 events.
 - an iPod to the developer of the best new code development submitted to OpenSees between the OpenSeesDays 2011 and the OpenSees Days 2012 events.A link for "Rules & conditions" is provided.
- OpenSees Days Italy 2012**: A notice that OpenSees team members will be presenting at "OpenSeesDays, Rome, Italy" on May 24-25, 2012. A link is provided for registration and program information.

On the left side of the page, there is a sidebar with links for HOME, OPENSEESWIKI, MESSAGE BOARD, USER DOC, DOWNLOAD, SOURCE CODE, and BUG REPORT. There is also a search box and a note about customizing quicklinks by going to the Site Map.

Building Blocks for Modern Simulation Code



Open-Source - Leave it out there for community

PEER: OpenSees Goals (1998):

1. To use **modern software techniques** to evolve an extensible open-source finite element software platform for earthquake engineering that would encompass both **structural & geotechnical** engineering.
2. To provide **a common analytical research framework** for PEER researchers **to educate students & share** new knowledge.
3. To foster a mechanism whereby new research developed through PEER could be **disseminated to industry** for testing and implementation.



*Provost Gregory Fenves
UT Austin*

Has It Worked: Used Worldwide (2013)

Visitors

83,663

% of Total: 100.00% (83,663)

Visits

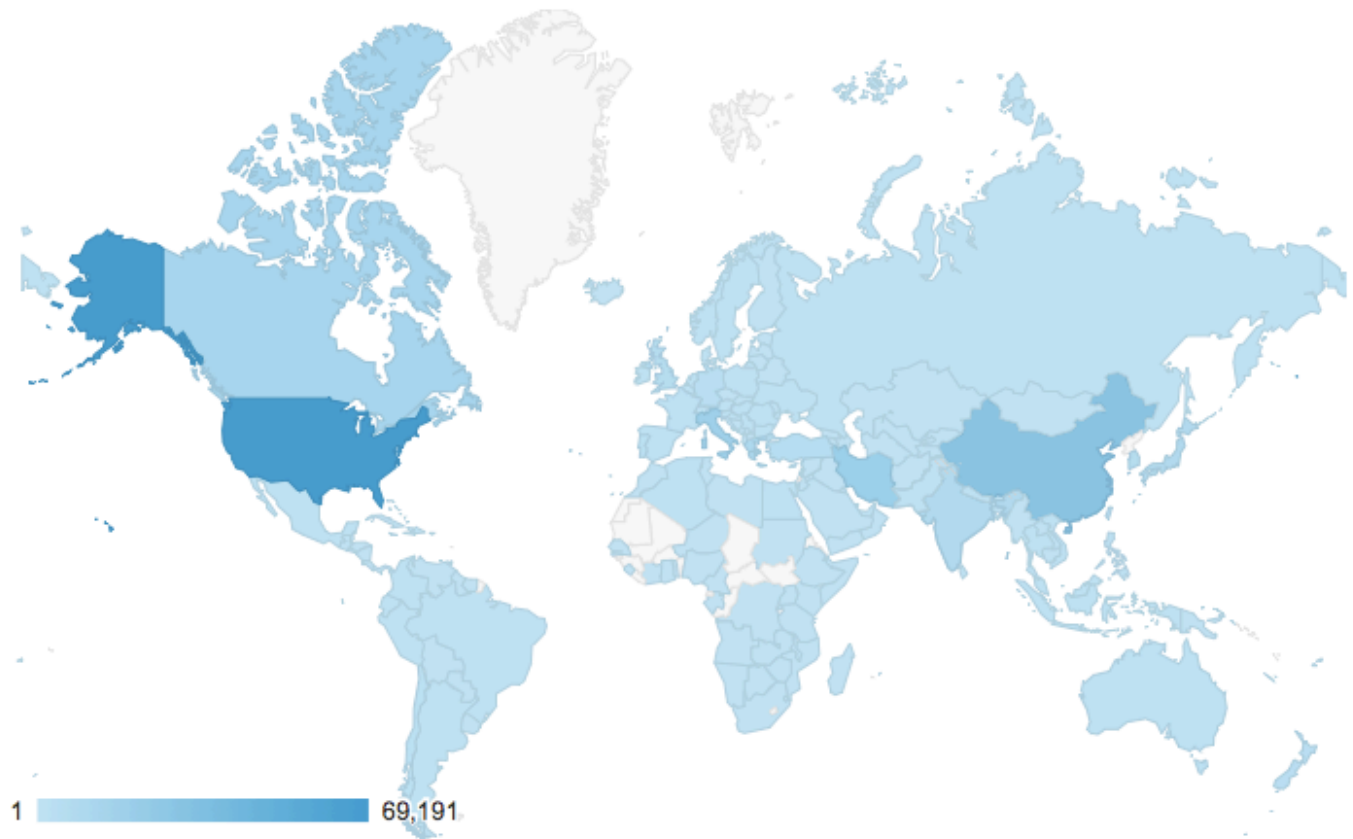
249,630

% of Total: 100.00% (249,630)

Pageviews

1,492,715

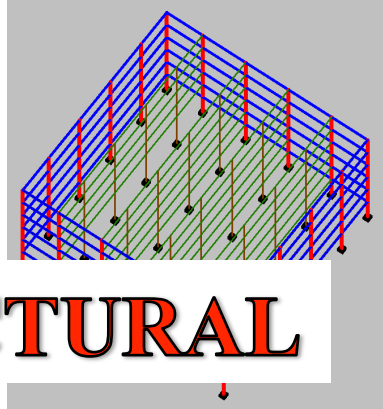
% of Total: 100.00%
(1,492,715)



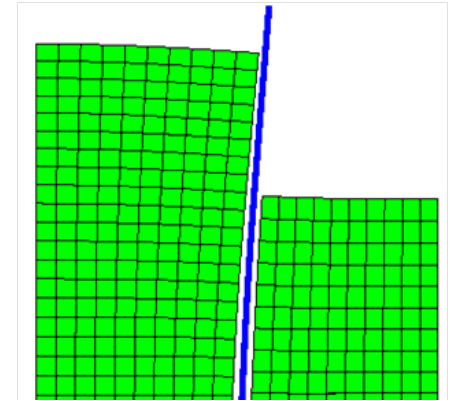
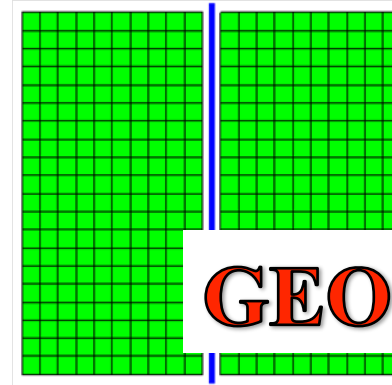
1. United States

2. China

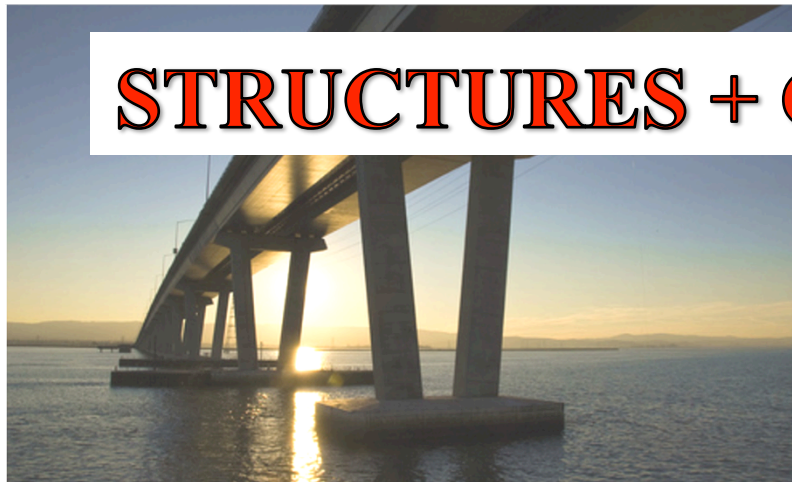
Example Usage



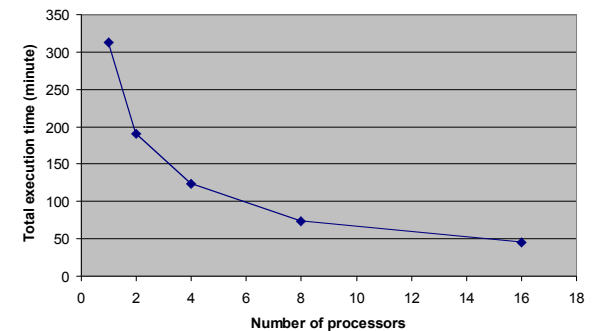
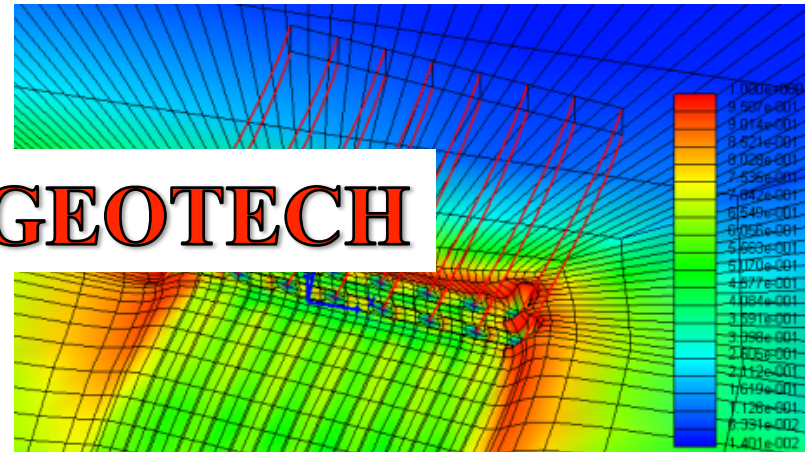
STRUCTURAL



GEOTECHNICAL



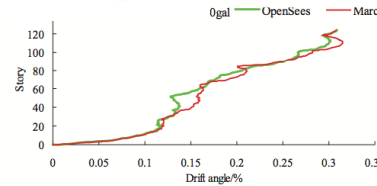
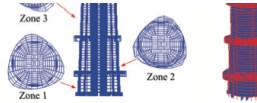
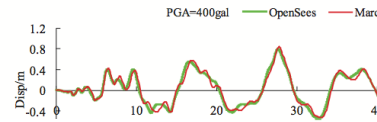
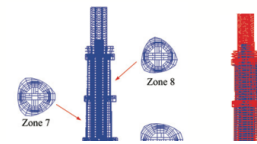
STRUCTURES + GEOTECH



Comparison/Validation

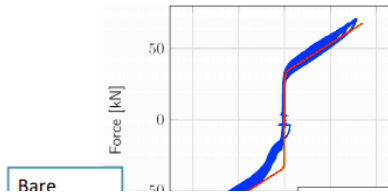
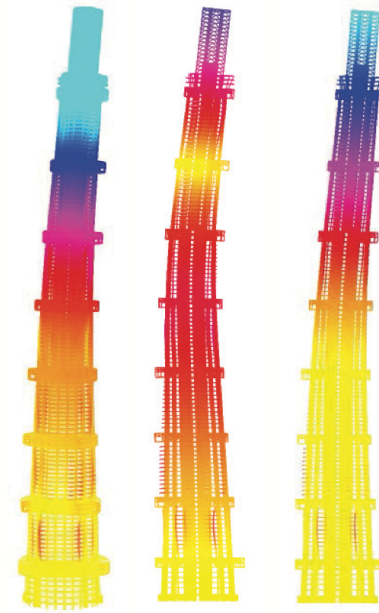
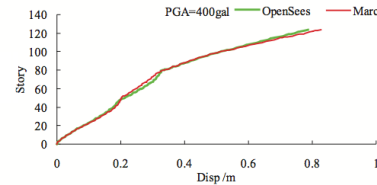


TBI Building 2 & Shanghai Tower



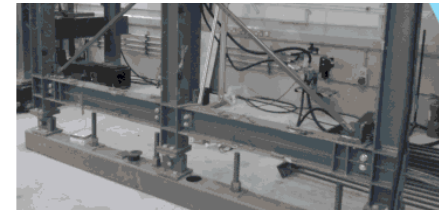
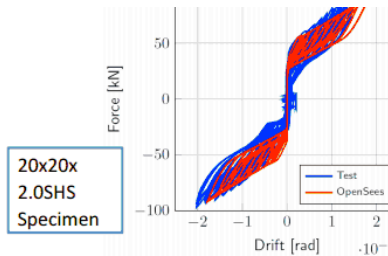
➤ Shanghai Tower

- H=632m, 124 stories
- 53,006 nodes
- 88,089 elements
- 48,774 fiber beam elements
- 39,315 multi-layer shell elements
- Memory used: 8.9GB



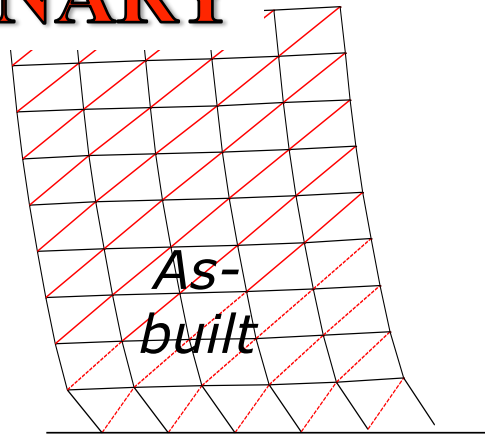
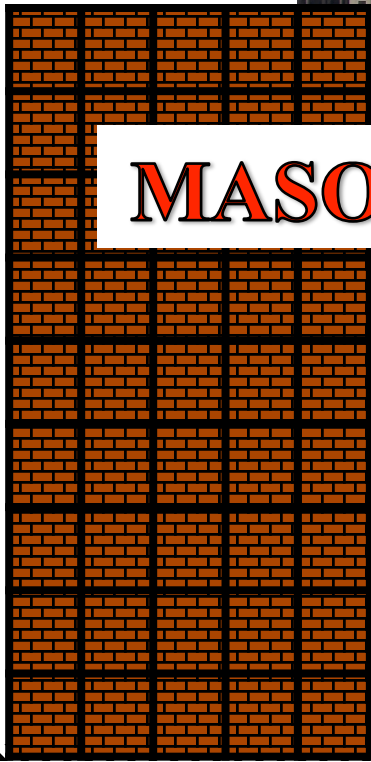
- Testing on bare frame demonstrated the bilinear elastic behavior of the frame and the OpenSees predicts the behavior very well.
- Testing with a 20x20x2.0SHS specimen showed the energy dissipation and self-centering of the frame and also that the OpenSees modeling predicts the behavior accurately.

Verification/Calibration/Validation



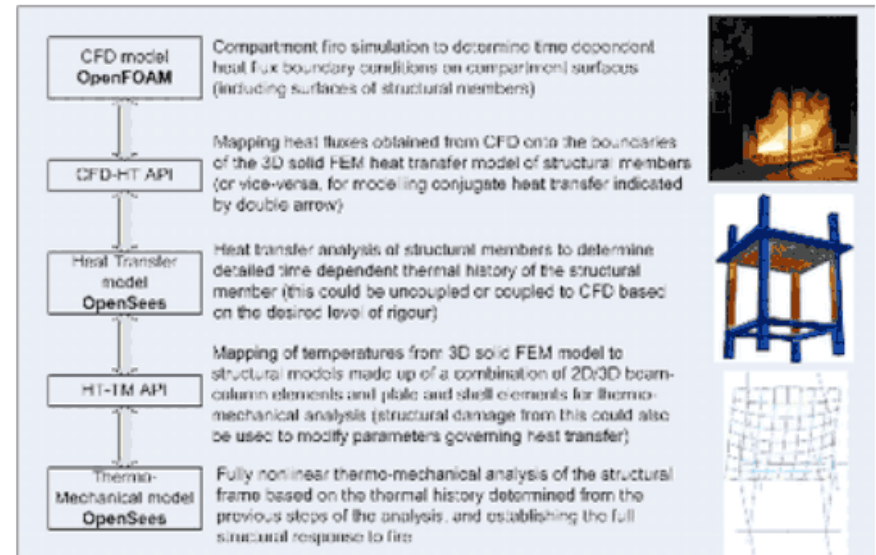


MASONRY



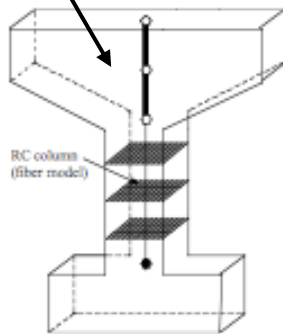
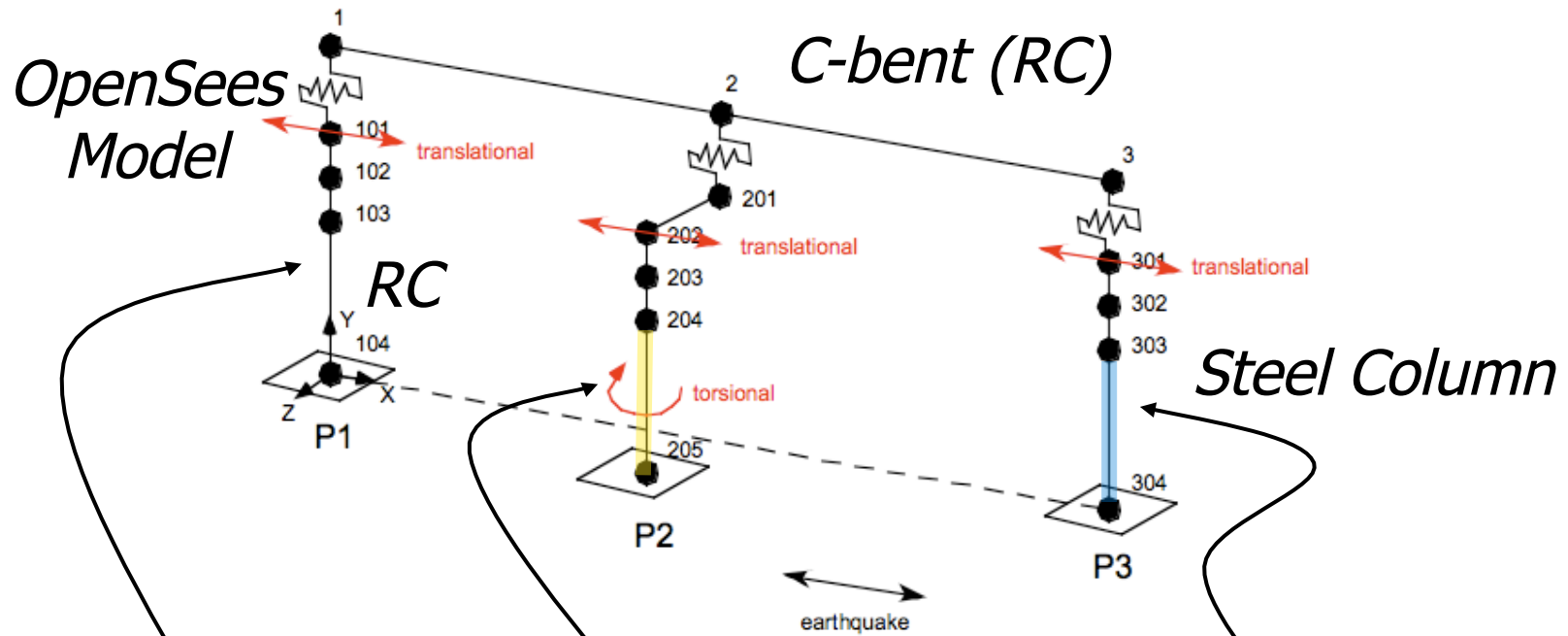
M.Talaat & K.Mosalam

FIRE

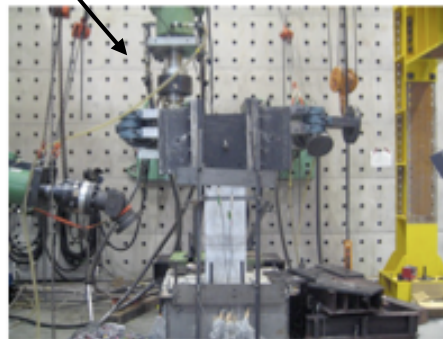


A. Usmani et al. , University of Edinburgh

HYBRID SIMULATION



Numerical Component
at Kyoto University



Experimental Component
at Kyoto University



Experimental Component
at nees@berkeley

Kyoto/Berkeley

New Directions:

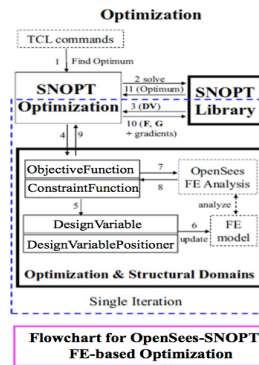
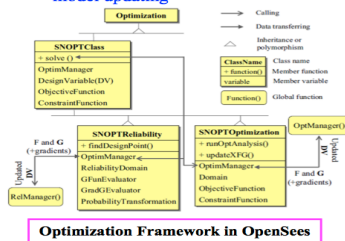
1. Optimization

J. Conte & P.Gill (UCSD), Q.Gu (Xiaman)

OpenSees-SNOPT Extended Framework

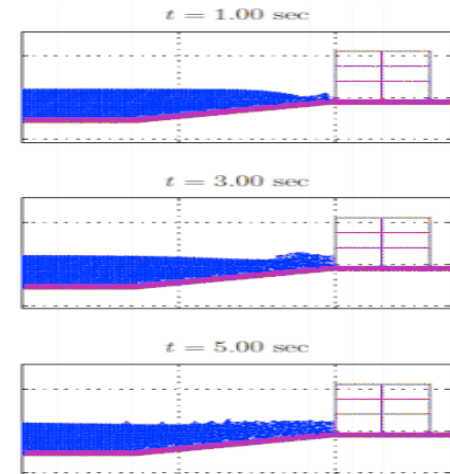
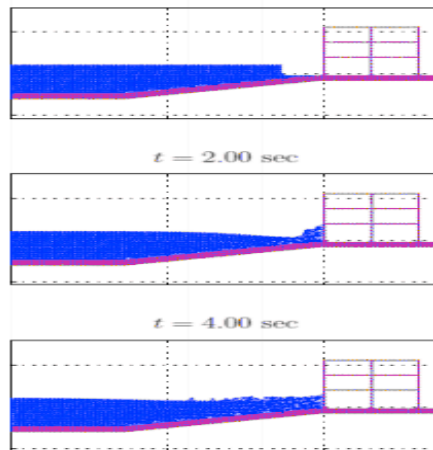
• **OpenSees & SNOPT Link:**

- After abstract class of **Optimization**, subclass **SNOPTClass** is used to encapsulate the *SNOPT* software package as an interface
- Subclass **SNOPTReliability** for the design point search problem in reliability analysis; while the **SNOPTOptimization** structural optimization & model updating



2. Fluid Structure Interaction

M.Scott, Oregon State



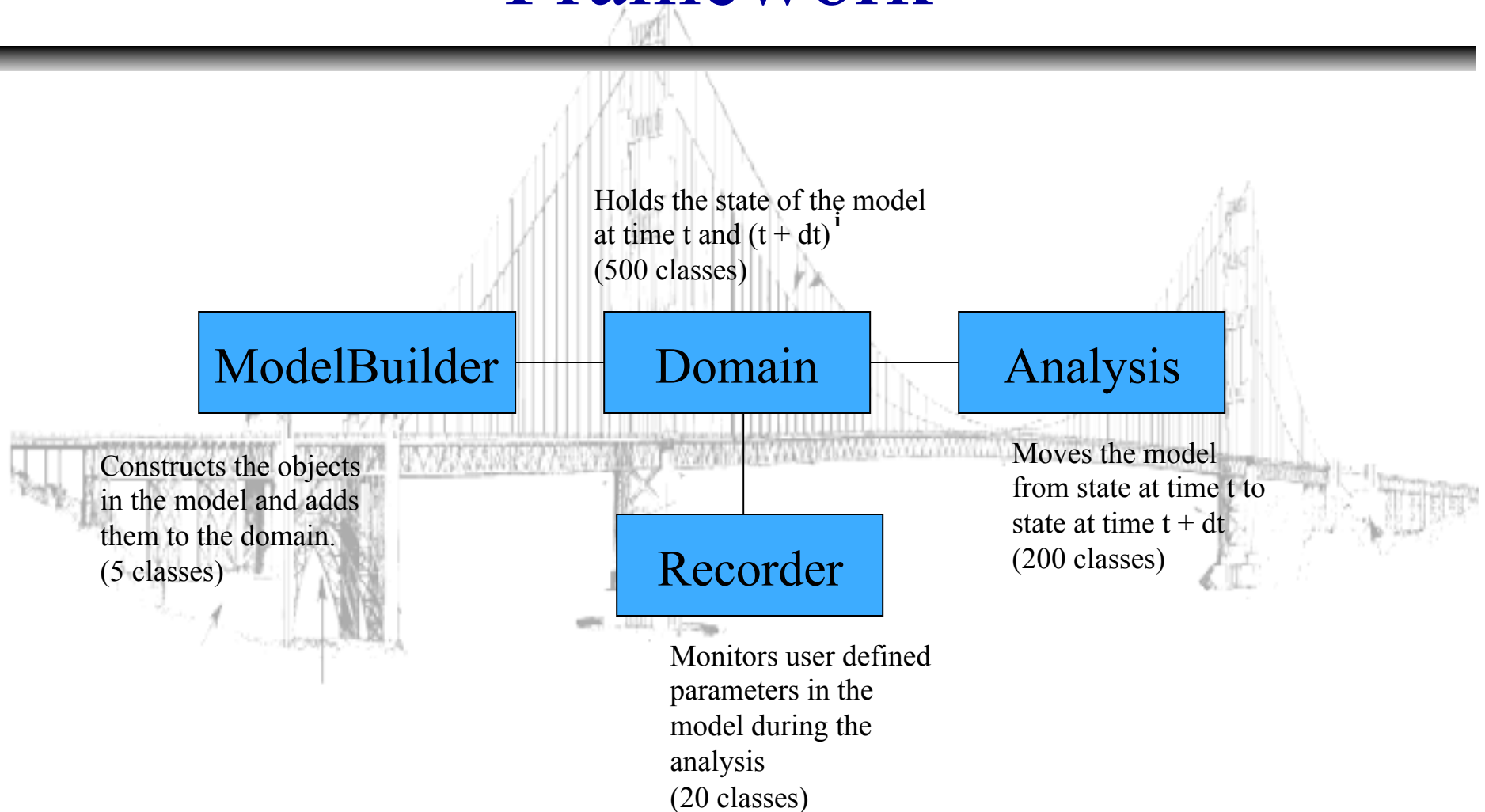
SO What is OpenSees?

- OpenSees is a software framework for building finite element applications in structural and geotechnical systems.
- These applications run on both sequential, parallel, and distributed computer systems.

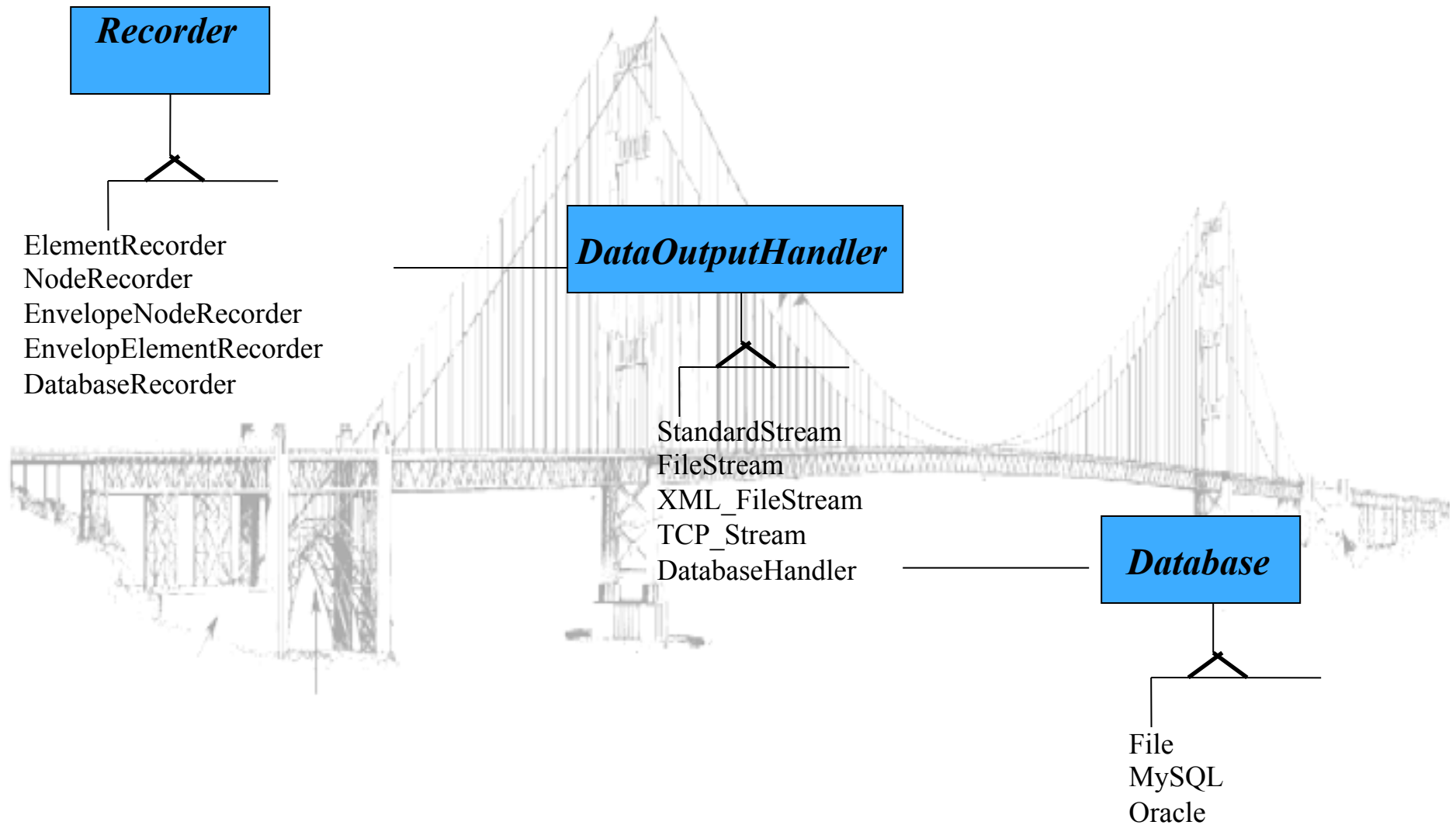
OpenSees is a Software Framework

- A framework is **NOT an executable**.
- A *framework* **IS** a set of cooperating software components for building applications in a specific domain.
- The OpenSees framework is written primarily in the object-oriented language C++; though other languages namely C and Fortran are also used.
- The abstract classes in the OpenSees framework define the interface. The concrete subclasses that exist in the framework provide the implementations.
- Other classes can be provided to extend the capabilities of the framework by developers using DLL's or providing the source code to the OpenSees repository.
- Currently over 1000 classes in the OpenSees framework.

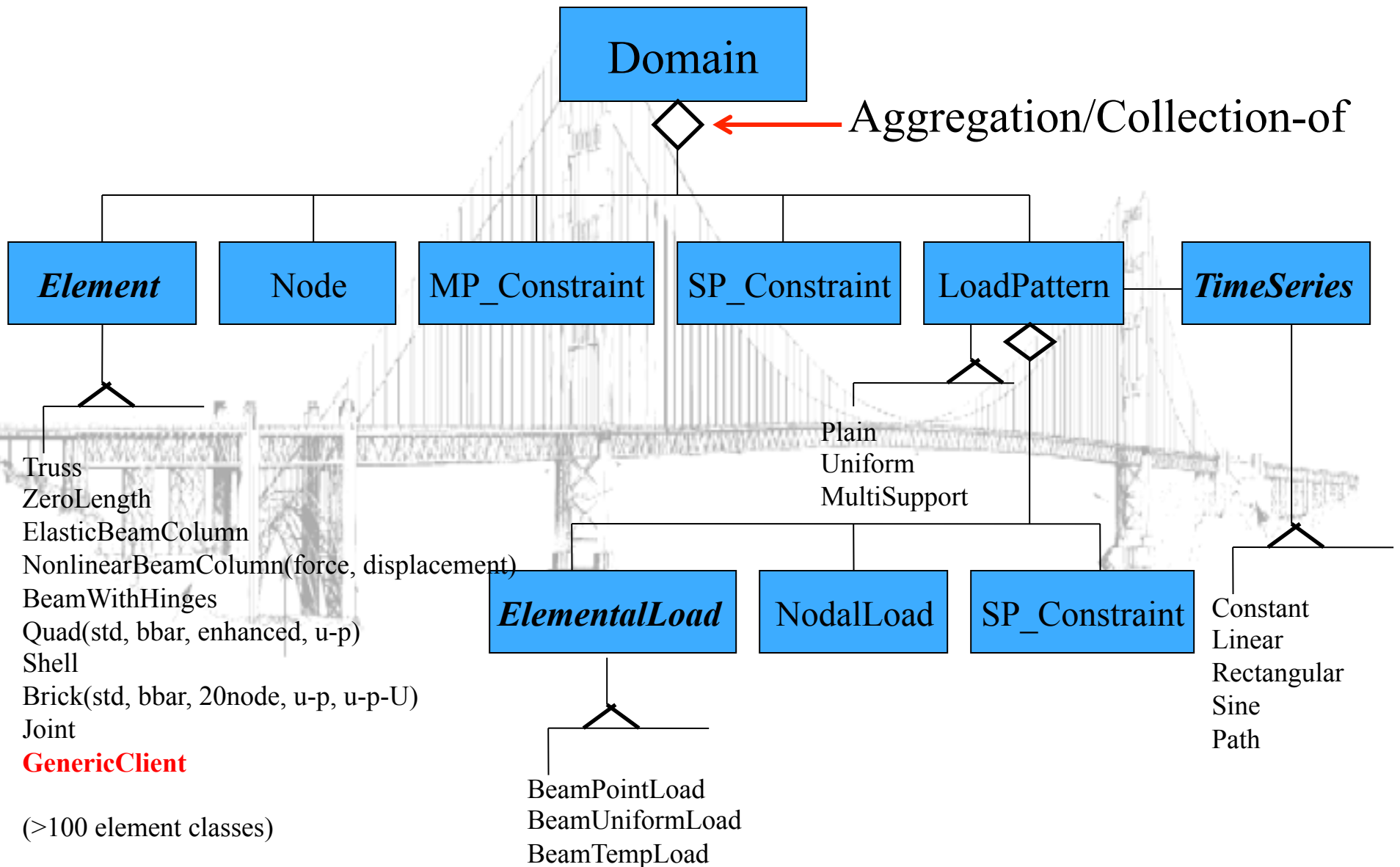
Main Abstractions in OpenSees Framework



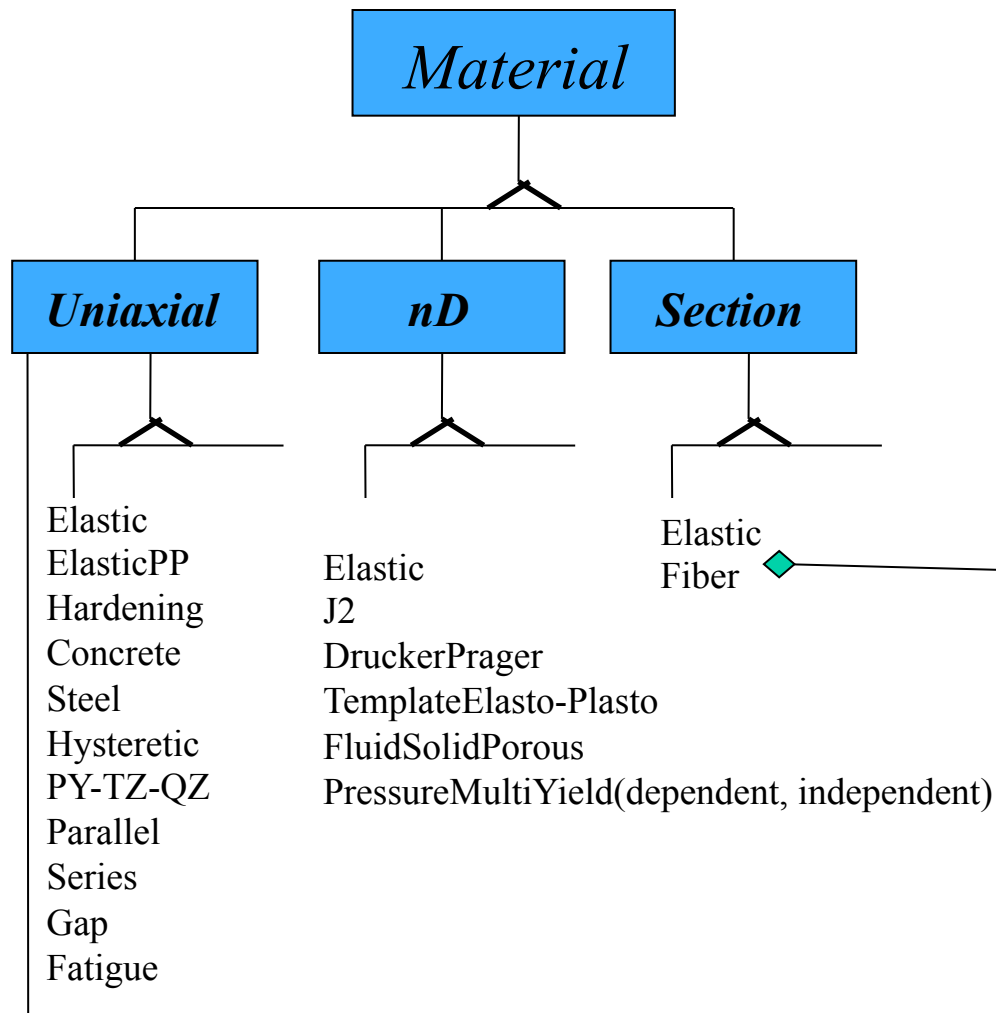
Recorder Options



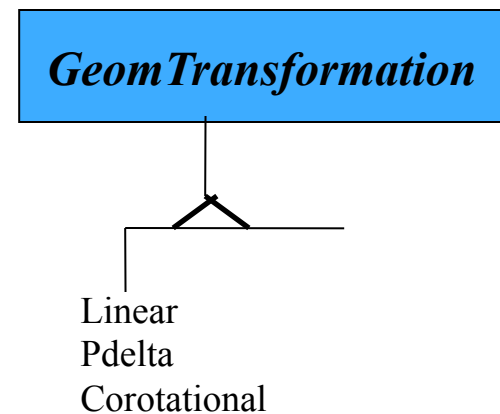
What is in a Domain?



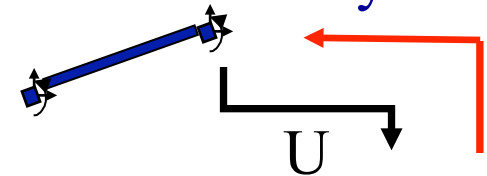
Some Other Classes associated with Elements:



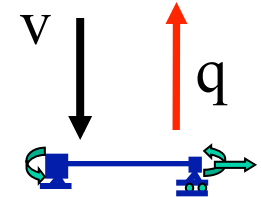
(over 250 material classes)



Element in Global System

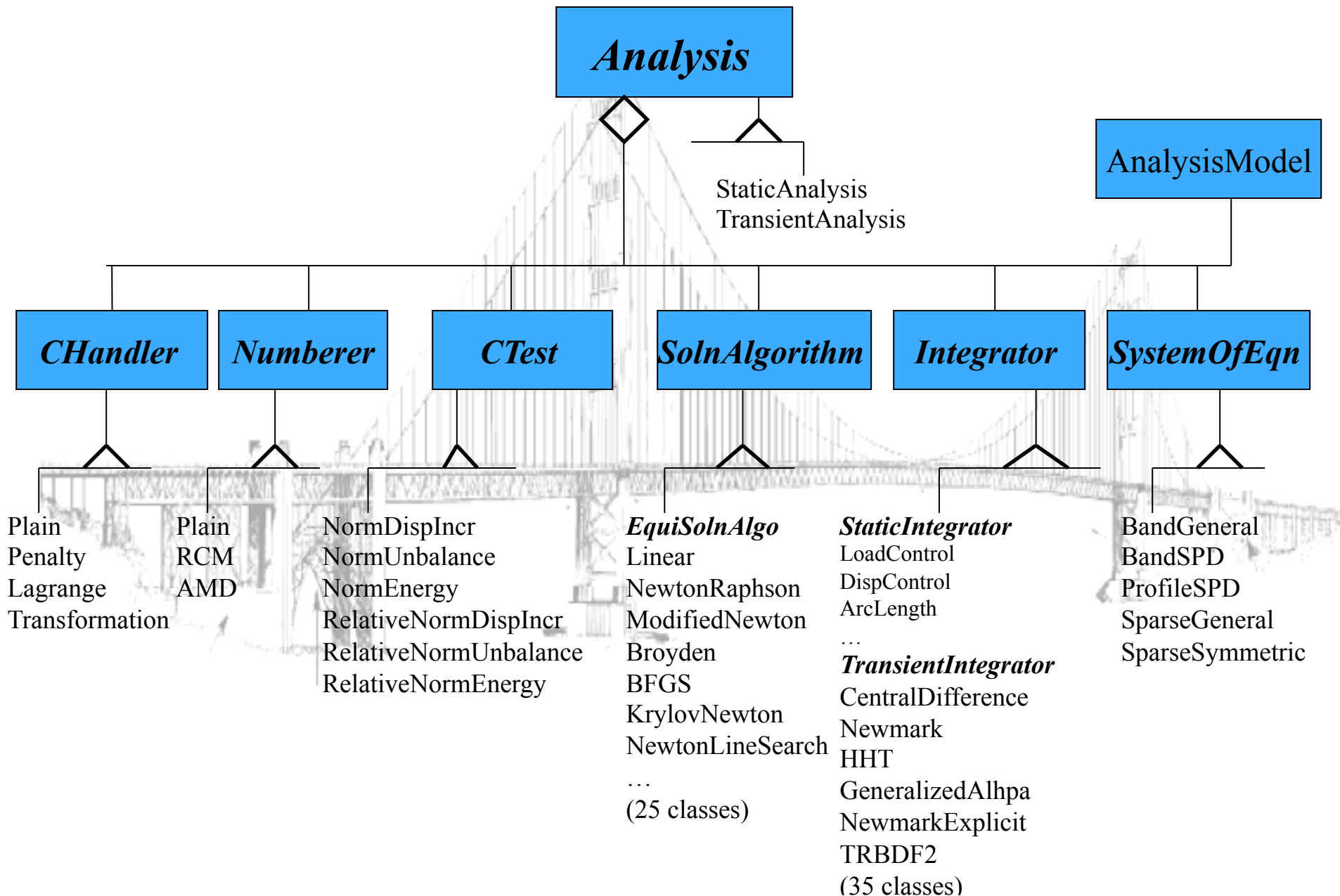


Geometric Transformation



Element in Basic System

What is an Analysis?



Outline of Workshop

- Introduction to OpenSees Framework

- OpenSees & Tcl Interpreters

- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

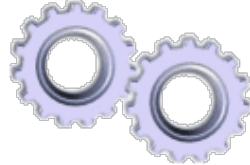
How Do People Use the OpenSees Framework?

- Provide their own main() function in C++ and link to framework.
- Use OpenSees interpreter^S. These are extensions of the Tcl interpreters, tclsh and wish, for performing finite element analysis.
 1. OpenSees.exe
 2. OpenSeesTk.exe
 3. OpseesSP.exe
 4. OpenSeesMP.exe

Tcl Interpreters

- **wish** and **tclsh** **are tcl interpreters.**
 - Interpreters (Perl, Matlab, Ruby) are programs that execute programs written in a programming language immediately.
 - There is no separate compilation & linking.
 - An interpreted program runs slower than a compiled one.

puts “sum of 2 and 3 is [expr 2 + 3]”



sum of 2 and 3 is 5

```
Terminal — tclsh8.4 — 85x9
fmk:~$ tclsh
% puts "sum of 2 and 3 is [expr 2 + 3]"
sum of 2 and 3 is 5
% █
```

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 Syntax Rules

(rules that define combinations that give a correctly structured program)

- 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
- The first word is the command name
- The remaining words are the command arguments

`commandName $arg1 $arg2 $arg3`

- The number of arguments depends on the command

Comments

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

```
# This is a comment
```

- Some people suggest writing comments before you write any code.
- **Judicious use of comments**, try to avoid to comment trivial things, like:

```
set a 5; # setting a to 5
```

Variables & Variable Substitution

- A variable is a symbolic name used to refer to some location in memory that has a value; the separation of name and value allows the value to be used independently of exact value.
- In Tcl to create a variable use **set** command.

```
set a 2.0
```

- To use the value of the variable use a **\$**

```
set b $a
```

- Use descriptive names

```
set PI 3.14159
```

puts

- puts is the command to produce output.
- The last argument is string to be output.
- Result sent to screen, unless the optional file id is provided.

puts <\$fileID> string

puts \$a

> 2

puts “value of a is: \$a”

> value of a is: 2

expr

- **expr** command is used to evaluate expressions
- Pretty much any mathematical expression is allowed

```
expr sqrt(($x*$x)+($y*$y))
```

- Typically you set some variable value equal to result of expression.

```
set length [expr sqrt(($x*$x)+($y*$y))]
```

- Watch out for **integer math**.

```
puts [expr 5/2]
```

```
> 2
```

```
puts [expr 5/2.]
```

```
> 2.5
```


if

- The if command has following general form

```
if expr1 body1 <elseif expr2 body2 elseif ... > <else bodyN>
```

- The else and elseif are optional
- The expr must return a boolean value
- The body is a set of commands enclosed in **{}**, use **{** on same line as the expression

```
if {$a == 2} {  
  set b 2  
}
```

```
if {$a == 2} {  
  set b 2  
} else {  
  set b 3  
}
```

```
if {$a == 2} {  
  set b 2  
} elseif {$a == 3} {  
  set b 3  
} else {  
  set b 4  
}
```

while

- The while command has the following form

```
while expression body
```

- The while command executes the body of the loop each time the expression evaluates to true.

```
set sum 0
set i 1
while {$i < 10} {
    set sum [expr $sum + $i]
    incr i 1
}
puts $sum
```

➤ 45

- An infinite loop will occur if expression always true
- Body never evaluated if expression initially false

for

- The for command is used when #cycles is known

```
for {initiation} {expression} {increment} {body}
```

- arg1 is set of commands executed at start, initiation
- arg2 is expression checked at start of each loop, when false for stops
- arg3 is set of commands to do at end of each loop
- arg4 is set of commands to do for body of the for loop

```
for {set i 1; set sum 0.} {$i < 10} {incr i 1} {  
    set sum [expr $sum + $i]  
}  
puts $sum
```

foreach

- The foreach command simplifies looping over **lists**

```
foreach varName1 list1 <varName2 list2 ...> {body}
```

- The foreach command successively assigns the values contained in the list to the variable named.
- If multiple names and lists, the lists must be of same size.

```
set sum 0
foreach v {1 2 3 4 5} {
    set sum [expr $sum + $v]
}
```

```
set sum 0
foreach v {1.0 9.0} s {2. 6.} {
    set sum [expr $sum + $v*$s]
}
```

```
set sum 0
foreach {v s} {1.0 2. 9.0 6.}
    set sum [expr $sum + $v*$s]
}
```

lists

- A list is an ordered collection of elements, of no fixed length. The list elements can be any string values, including numbers, variable names, other lists
- A list consists of elements separated by white spaces
- There are commands for operating on lists:

**list lappend linsert llength lindex lsearch
lreplace lrange lsort concat foreach**

```
set a "123 456"  
lappend a 789  
llength $a  
lindex $a 2  
set a [linsert $a 1 9]  
lsearch $a 789
```

```
>123 456  
>123 456 789  
>3  
>789  
>123 9 456 789  
>3
```

Lists start numbering at 0

array

- An array is an **associated** collection of elements –
 - A collection of **key-value** pairs.
 - key any string or number
- The array command provides several operations for working on arrays.

array option name \$arg.

options: **set exists get names size unset**

```
array set c {eleven 11 two 2}  
puts $c(eleven)  
set c(ten) 10  
puts [array names c]  
foreach {key value} [array get c] {  
    puts "$key $value"  
}  
puts [array get c eleven]
```

```
>  
>11  
>  
>two ten eleven  
>  
>two 2  
    ten 10  
    eleven 11  
>eleven 11
```

open & file

- The **open** command is used to open files for reading and writing. It returns the file descriptor.

```
open filename $mode
```

```
set inFile [open test.txt r]
set outFile [open test.out w]
set fileData [read $inFile]
puts $outFile $fileData
close $inFile; close $outFile;
```

?

- The **file** command provides several operations for working with the file system. Of form:

```
file option name $arg1 $arg2 ..
```

- example options include
copy exists delete isdirectory mkdir

proc

- The **proc** command creates procedures (your own commands)
 - First arg is procedure name.
 - Second arg defines the list of arguments
 - Third argument is body of the procedure
- The **return** command in the body returns the result and returns from the procedure

```
proc sum {a b} {  
  return [expr $a + $b]  
}
```

```
puts "sum 3 4 = [sum 3 4]"
```

```
➤ sum 3 4 = 7
```


proc

- The special variable name **args** allows procedures to take variable number of arguments
- Each Tcl procedure invocation maintains its own variables. Variables in a procedure have **local scope**.
- The **global** command inside a procedure marks variable as global

```
proc addSum {args} {  
    global sum; set localSum 0.  
    foreach num $args {  
        set localSum [expr $localSum + $num]  
    }  
    set sum [expr $sum + $localSum]  
    return "$localSum $sum"  
}
```

```
set sum 0  
puts [addSum 3 4 5]  
puts [addSum 3 4 5]
```

```
➤ 12 12  
➤ 12 24
```

Example Tcl

•comments

```
># this is a comment
```

•Variables

```
>set a 1
1
>set b a
a
>set b $a
1
```

•file manipulation

```
>set fileId [open tmp w]
??
>puts $fileId "hello"
>close $fileID
>type tmp
hello
```

•Load other files

```
>source Example1.tcl
```

•expression evaluation

```
>expr 2 + 3
5
>set b[expr 2 + $b]
3
```

•lists

```
>set a {1 2 three}
1 2 three
>set la [llength $a]
3
>set start [lindex $a 0]
1
>lappend a four
1 2 three four
```

•associative arrays

```
>array set a {one 1 three 3}
>set $a(fifty) blah
>array get a fifty
fifty blah
```

•procedures & control structures

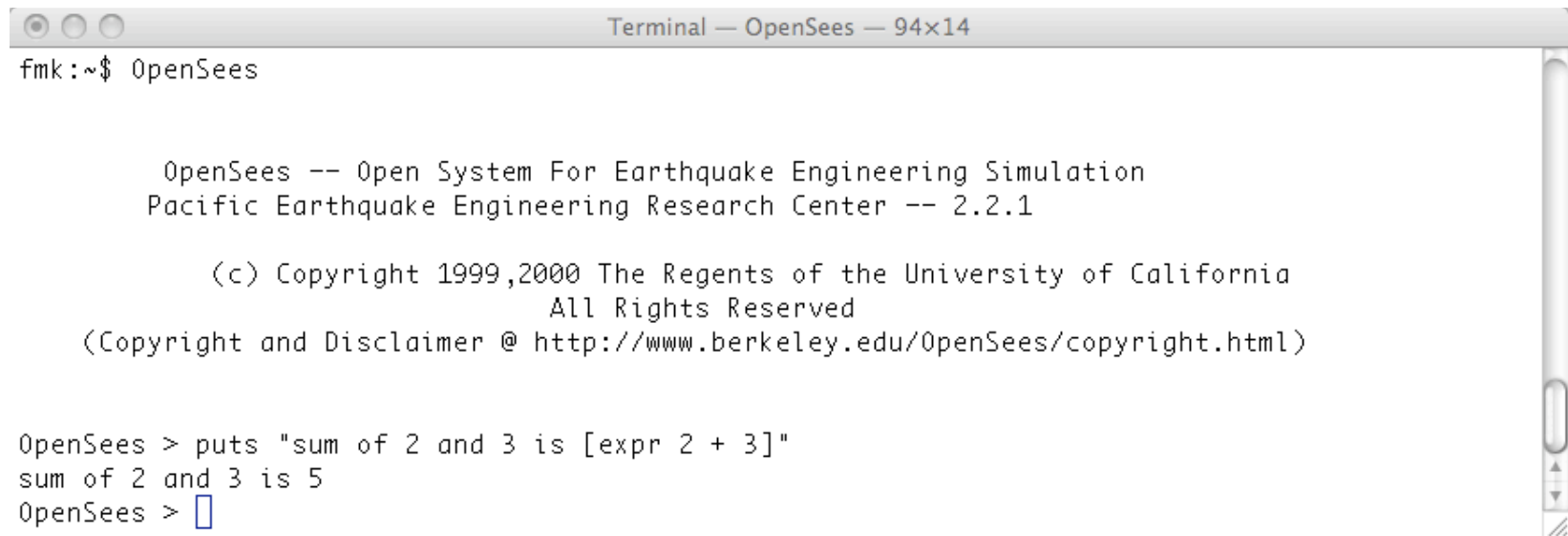
```
> for {set i 1} {$i < 10} {incr i 1}
puts "i equals $i"
}
...
> set sum 0
foreach value {1 2 3 4} {
    set sum [expr $sum + $value]
}
>puts $sum
10
>proc guess {value} {
    global sum
    if {$value < $sum} {
        puts "too low"
    } else {
        if {$value > $sum} {
            puts "too high"
        } else { puts "you got it!" }
    }
}
> guess 9
too low
```

OpenSees Interpreters

- The OpenSees interpreters are tcl interpreters which have been **extended** to include commands for finite element analysis:
 1. Modeling – create nodes, elements, loads and constraints
 2. Analysis – specify the analysis procedure.
 3. Output specification – specify what it is you want to monitor during the analysis.
- Being interpreters, this means that the files you create and submit to the OpenSees interpreters **are not input files**. You are creating and submitting **PROGRAMS**.

OpenSees.exe

- An interpreter that extends tclsh for FE analysis.



```
Terminal — OpenSees — 94x14
fmk:~$ OpenSees

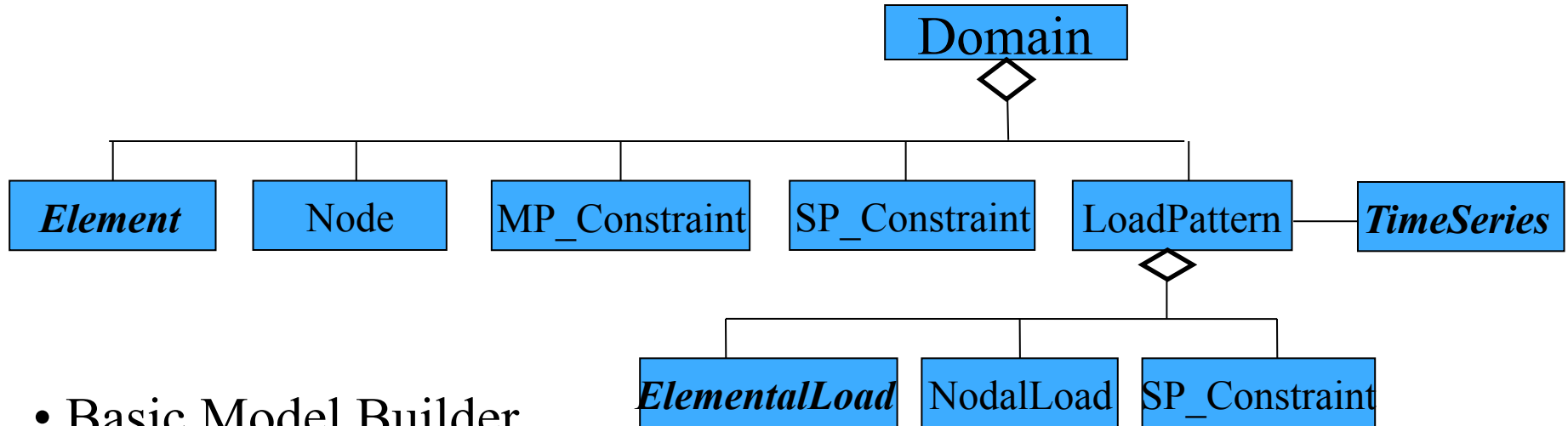
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

OpenSees > puts "sum of 2 and 3 is [expr 2 + 3]"
sum of 2 and 3 is 5
OpenSees > 
```

model Command

*Adds the modeling commands to the interpreter.



- Basic Model Builder

```
model Basic -ndm ndm? <-ndf ndf?>
```

This command now adds the following commands to the interpreter:

node mass element equalDOF fix fixX fixY fixZ
pattern timeSeries load eleLoad sp
uniaxialMaterial nDMaterial section geomTransf
fiber layer patch block2D block3D

```
model Basic -ndm 2 -ndf 2
```

```
#node $tag $xCrd $yCrd
```

```
node 1 0.0 0.0
```

```
node 2 144.0 0.0
```

```
node 3 168.0 0.0
```

```
node 4 72.0 96.0
```

```
#fix $nodeTag $xFix $yFix
```

```
fix 1 1 1
```

```
fix 2 1 1
```

```
fix 3 1 1
```

```
#uniaxialMaterial Elastic $tag $E
```

```
uniaxialMaterial Elastic 1 3000.0
```

```
#element truss $tag $iNode $jNode $A $matTag
```

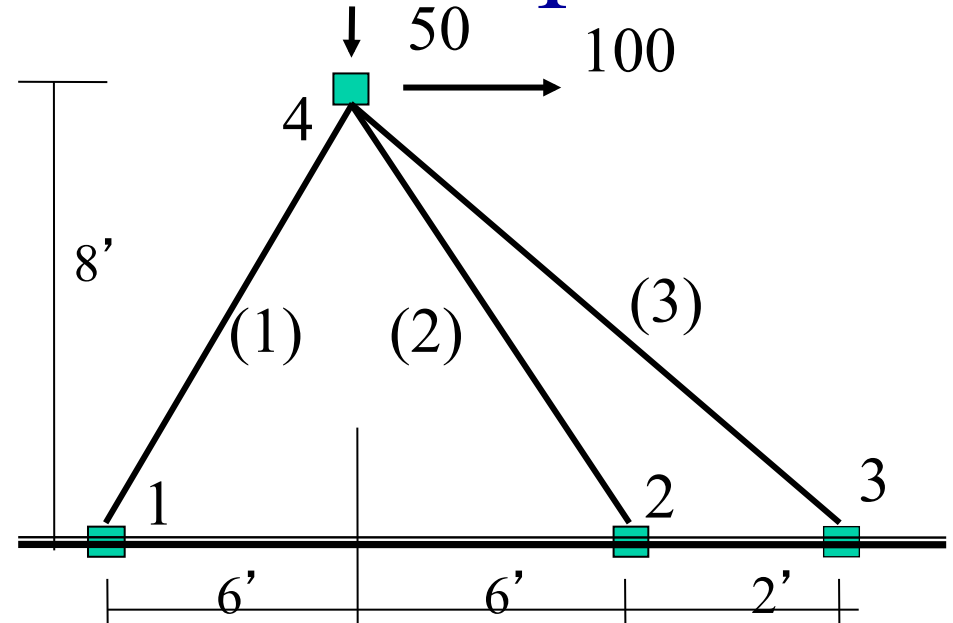
```
element truss 1 1 4 10.0 1
```

```
element truss 2 2 4 5.0 1
```

```
element truss 3 3 4 5.0 1
```

```
timeSeries Linear 1
```

Truss example:



	E	A
1	3000	10
2	3000	5
3	3000	5

```
#pattern Plain $tag $tsTag
```

```
pattern Plain 1 1 {
```

```
#load $nodeTag $xFrc $yFrc
```

```
load 4 100.0 -50.0
```

```
}
```

When should you Use **V**ariables In a Program?

- **NOT to document the commands**
(USE comments instead: i.e. more ‘#’ and less ‘set’)
- When you have a variable that you might change at some point later in time or in the script, or if value will be repeatedly used.

Truss example using variables:

```
model Basic -ndm 2 -ndf 2
```

```
set xCrd1 0.0
```

```
set xCrd2 144.0
```

```
set xCrd3 168.0
```

```
set xCrd4 62.0
```

```
set yCrd1 0
```

```
set yCrd2 96
```

```
set matTag1 1
```

```
set timeSeriesTag1 1
```

```
set patternTag1 1
```

```
set E1 3000
```

```
set nodeLoadTag 4
```

```
set nodeLoadx 100.
```

```
set nodeLoadY -50;
```

```
set A1 10
```

```
set A2 5.0
```

```
node 1 $xCrd1 $yCrd1
```

```
node 2 $xCrd2 $yCrd1
```

```
node 3 $xCrd3 $yCrd1
```

```
node 4 $xCrd4 $yCrd2
```

```
fix 1 1 1
```

```
fix 2 1 1
```

```
fix 3 1 1
```

```
uniaxialMaterial Elastic $matTag1 $E1
```

```
element truss 1 1 4 $A1 $matTag1
```

```
element truss 2 2 4 $A2 $matTag1
```

```
element truss 3 3 4 $A3 $matTag1
```

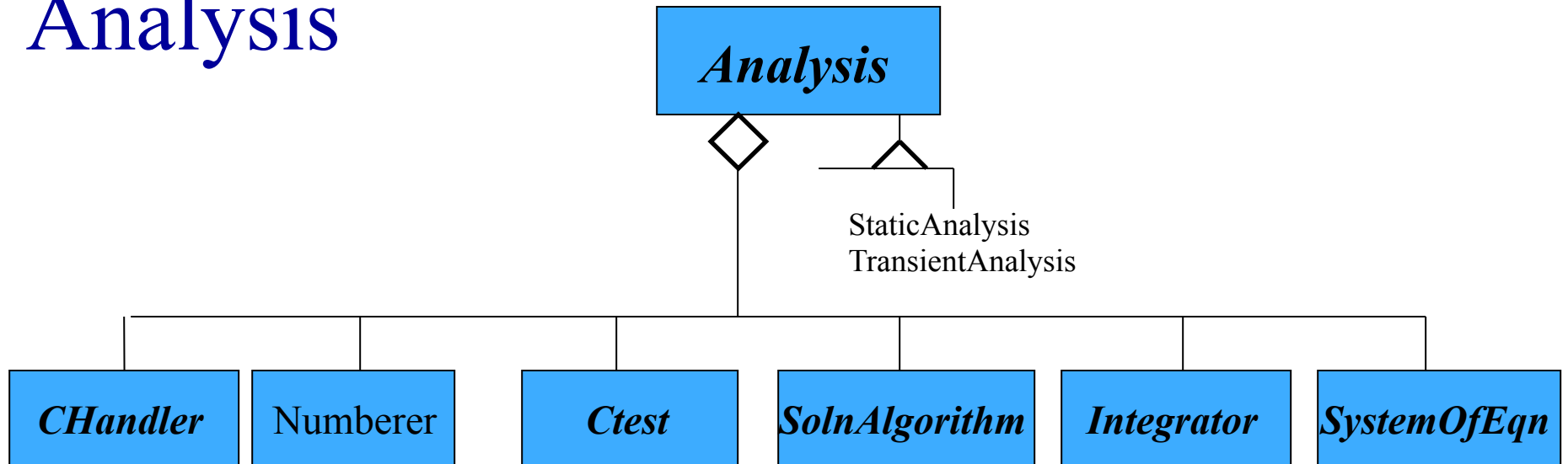
```
timeSeries Linear $tsTag1
```

```
pattern Plain $patternTag1 $tsTag1 {
```

```
  load 4 $nodeLoadX $nodeLoadY
```

```
}
```


Analysis



handler type? args...

numberer type? args...

test type? args...

algorithm type? args...

integrator type? args...

system type? args...

analysis type? args..

analyze args ...

Example Analysis:

- Static Nonlinear Analysis with LoadControl

```
constraints Transformation  
numberer RCM  
system BandGeneral  
test NormDispIncr 1.0e-6 6 2  
algorithm Newton  
integrator LoadControl 0.1  
analysis Static  
analyze 10
```

- Transient Nonlinear Analysis with Newmark

```
constraints Transformation  
numberer RCM  
system BandGeneral  
test NormDispIncr 1.0e-6 6 2  
algorithm Newton  
integrator Newmark 0.5 0.25  
analysis Transient  
analyze 2000 0.01
```

Commands That Return Values:

•analyze command

```
set ok [analyze numIter < $\Delta t$ >]
```

•getTime command

```
set currentTime [getTime]
```

•nodeDisp command

```
set disp [nodeDisp $node <$dof>]
```

•nodeVel command

```
set vel [nodeVel $node <$dof>]
```

•nodeAccel command

```
set acc [nodeAccel $node <$dof>]
```

•nodeEigen command

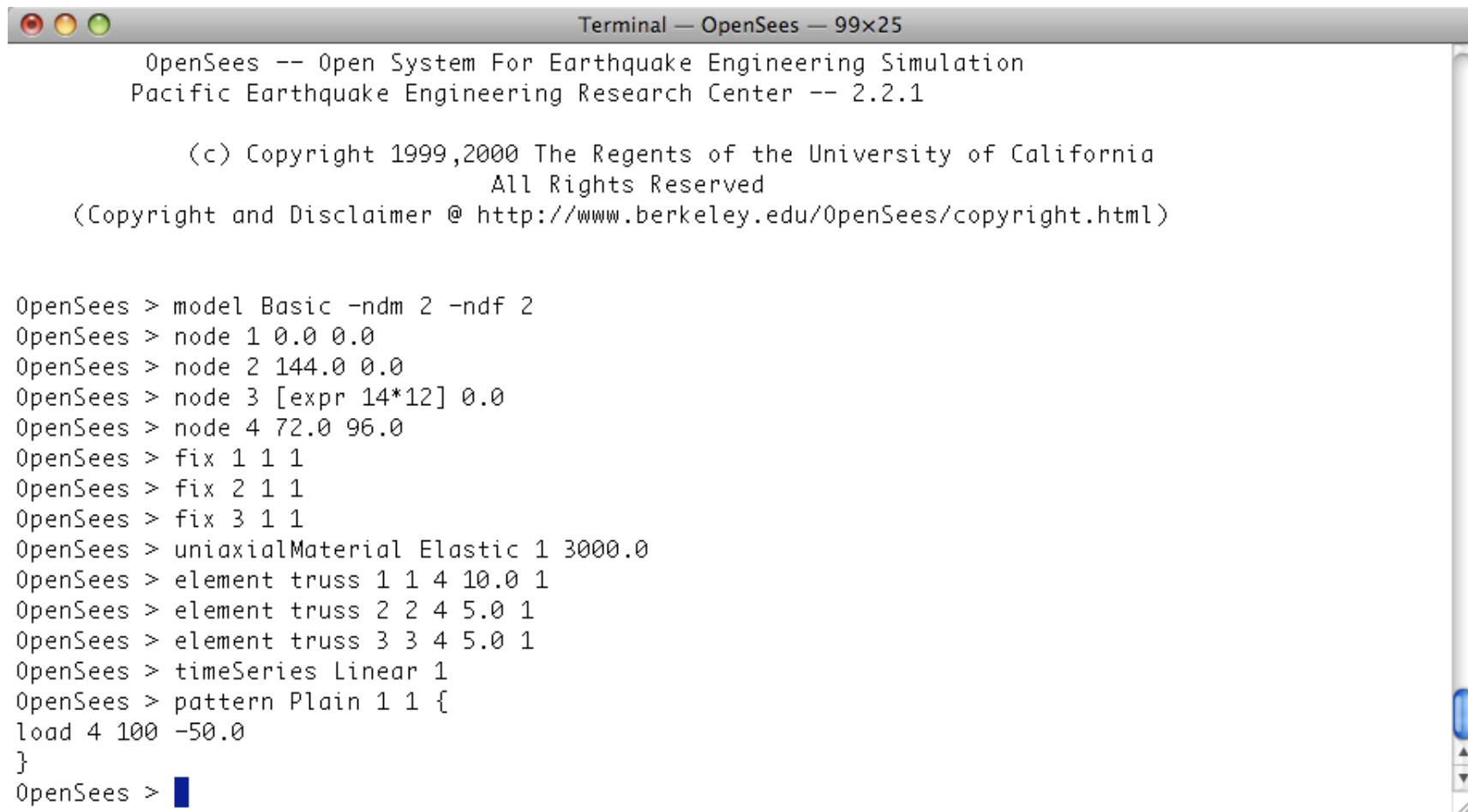
```
set eig [nodeEigen $node <$dof>]
```

•eleResponse command

```
set resp [eleResponse $eleTag $arg1 $arg2 ...]
```

3 Ways to Execute the commands

1. **Interactively** - the commands as we have shown can be input directly at the prompt



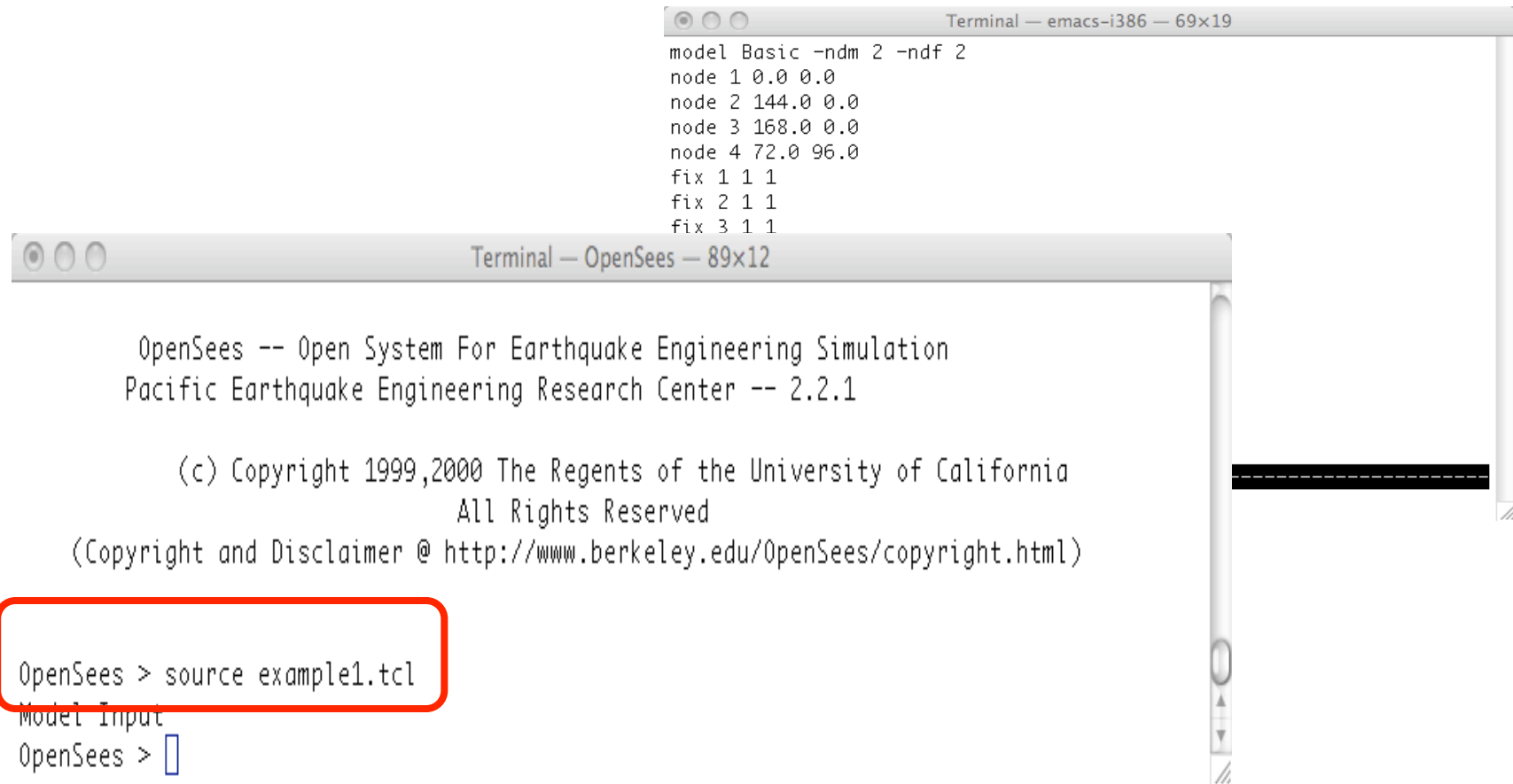
```
Terminal — OpenSees — 99x25
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/0penSees/copyright.html)

OpenSees > model Basic -ndm 2 -ndf 2
OpenSees > node 1 0.0 0.0
OpenSees > node 2 144.0 0.0
OpenSees > node 3 [expr 14*12] 0.0
OpenSees > node 4 72.0 96.0
OpenSees > fix 1 1 1
OpenSees > fix 2 1 1
OpenSees > fix 3 1 1
OpenSees > uniaxialMaterial Elastic 1 3000.0
OpenSees > element truss 1 1 4 10.0 1
OpenSees > element truss 2 2 4 5.0 1
OpenSees > element truss 3 3 4 5.0 1
OpenSees > timeSeries Linear 1
OpenSees > pattern Plain 1 1 {
load 4 100 -50.0
}
OpenSees > █
```

3 Ways to Execute the commands

2. Sourced from File- the commands are placed in a text file which is sourced in



```
Terminal — emacs-i386 — 69x19
model Basic -ndm 2 -ndf 2
node 1 0.0 0.0
node 2 144.0 0.0
node 3 168.0 0.0
node 4 72.0 96.0
fix 1 1 1
fix 2 1 1
fix 3 1 1

Terminal — OpenSees — 89x12
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

OpenSees > source example1.tcl
Model Input
OpenSees > 
```

3 Ways to Execute the commands

3. Batch Mode- the commands are placed in a text file which are executed at startup.

```
Terminal — emacs-i386 — 69x19
model Basic -ndm 2 -ndf 2
node 1 0.0 0.0
node 2 144.0 0.0
node 3 168.0 0.0
node 4 72.0 96.0
fix 1 1 1

Terminal — bash — 99x18
fmk:~$ OpenSees example1.tcl

OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

Model Input
fmk:~$
```

if batch mode - useful default variables: **argv** & **argc**

```
Terminal — emacs-i386 — 101x39
#parse input
if {$argc != 1} {
  puts "Incorrect Usage: OpenSees example2.tcl $E"
  exit
} else {
  set E [lindex $argv 0]
}

# model
model Basic -ndm 2 -ndf 2
node 1 0.0 0.0
node 2 144.0 0.0
node 3 168.0 0.0
node 4 72.0 96.0
fix 1 1 1
fix 2 1 1
fix 3 1 1
uniaxialMaterial Elastic 1 $E
element truss 1 1 4 10.0 1
element truss 2 2 4 5.0 1
timeSeries Linear 1
pattern Plain 1 1 {
  load 4 100.0 -50.0
}

#analysis
integrator LoadControl 1.0
algorithm Linear
numberer Plain
constraints Plain
system BandGeneral
analysis Static
analyze 2

#output
puts "node 4 disp [nodeDisp 4]"

Terminal — bash — 101x39
fmk:~$ OpenSees example2.tcl 3000.0

OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

node 4 disp          1.87500000000000000000          -0.88541666666666662966
fmk:~$ OpenSees example2.tcl 6000.0

OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

node 4 disp          0.93750000000000000000          -0.44270833333333331483
fmk:~$
```

OpenSees & Matlab

- Calling matlab from an OpenSees script (mScript.m)

```
# invoke matlab
```

```
if {[catch {exec matlab -nosplash -nodesktop -r "mScript; quit"}}}  
  {  
    puts "Ignore this $msg"  
  }  
}
```

- Calling OpenSees from a matlab script

```
# invoke matlab
```

```
!OpenSees opsScript.tcl
```


OpenSees Resources

<http://opensees.berkeley.edu>

- Message Board - **look for answers, post questions** and **ANSWERS**
<http://opensees.berkeley.edu/community/index.php>
- Getting Started Manual - basic how to for getting started
http://opensees.berkeley.edu/wiki/index.php/Getting_Started
- User Documentation - command documentation & theory!
http://opensees.berkeley.edu/wiki/index.php/Command_Manual
- User Examples
http://opensees.berkeley.edu/wiki/index.php/OpenSees_User
http://opensees.berkeley.edu/wiki/index.php/Examples_Manual
- Developers
http://opensees.berkeley.edu/wiki/index.php/OpenSees_Developer
<http://opensees.berkeley.edu/cgi-bin/cvsweb2.cgi/OpenSees/SRC/>

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters

- OpenSees & Output

- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Output Options

When you run OpenSees:

Input




No OUTPUT
Unless you request it!

3 ways to obtain output:

1. **puts** command

```
puts <$fileID> $string
```

2. **print** command

```
print <-file $fileName> <-node $nd1 $nd2 ..> <-ele $ele1 $ele2 ...>
```

3. **recorder** command

```
recorder $type $arg1 $arg2 ...
```

Example using puts (sdofExample1.tcl)

```
# create model & analysis
```

```
...
```

```
# open output file
```

```
set nodeOut [open node.out w]
```

```
set forceOut [open ele.out w]
```

```
#perform analysis
```

```
while {$ok == 0 && $t < $maxT} {
```

```
  set ok [analyze 1 $dT]
```

```
  set time [getTime]
```

```
  set d [nodeDisp 2 1]
```

```
  set forces [eleResponse 1 material stress]
```

```
  puts $nodeOut "$time $d"
```

```
  puts $forceOut "$time $forces"
```

```
  if {$d > $maxD} {
```

```
    set maxD $d
```

```
  } elseif {$d < [expr -$maxD]} {
```

```
    set maxD [expr -$d]
```

```
  }
```

```
  set t [expr $t + $dT]
```

```
}
```

```
#close the files
```

```
close $nodeOut
```

```
close $forceOut
```

```
puts "record: $record period: $Tn damping ratio: $dampRatio max disp: $maxD"
```



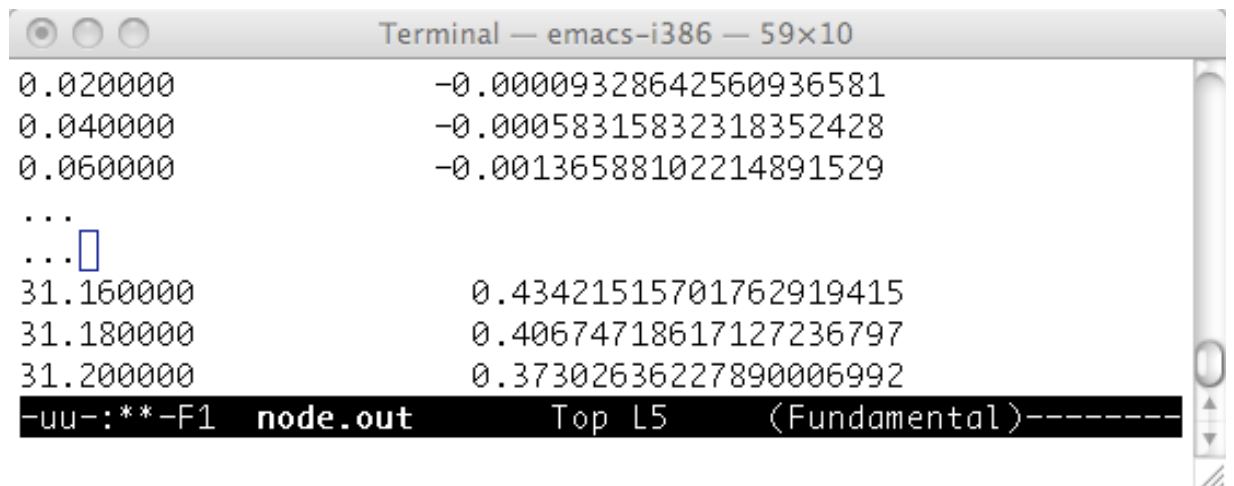
```
Terminal — bash — 110x13
OpenSees sdofExample1.tcl
```

```
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1
```

```
(c) Copyright 1999,2000 The Regents of the University of Calif
All Rights Reserved
```

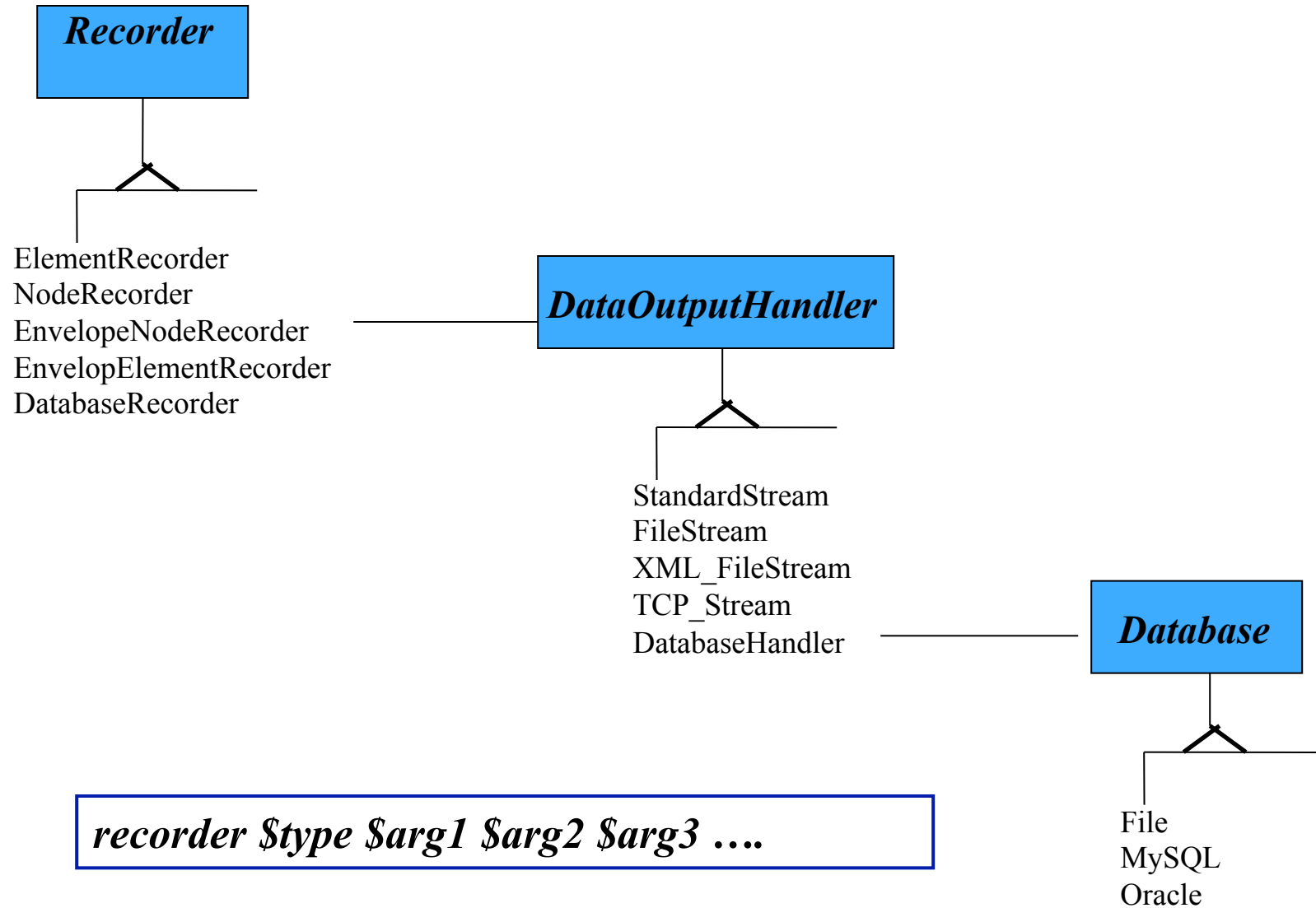
```
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright)
```

```
record: el_centro period: 1.0 damping ratio: 0.02 max disp: 5.96230501800
fmk:~/Desktop/Workshops/OpenSeesDays2010/OpenSeesDays2010/examples$
```



```
Terminal — emacs-i386 — 59x10
0.020000      -0.00009328642560936581
0.040000      -0.00058315832318352428
0.060000      -0.00136588102214891529
...
...
31.160000     0.43421515701762919415
31.180000     0.40674718617127236797
31.200000     0.37302636227890006992
-uu-:**-F1  node.out      Top L5      (Fundamental)-----
```

Recorder Options



Element/EnvelopeElement Recorders

- To monitor what's happening in the elements.

```
recorder Element <-file $fileName> <-time> <-ele $tg1 $tg2 ...> $arg1 $arg2 ...  
                <-xml $fileName>                <-eleRange $tgS $tgE>  
                <-binary $fileName>                <-region $rTag>  
                <-tcp $inetAddr>
```

- The response you can ask vary from element to element. There are of course some each element will respond to, e.g. forces.

```
recorder Element -file ele.out -ele 1 2 forces
```

```
recorder Element -file ele1sect1fiber1.out -ele 1 2 section 1 fiber 1stress
```

- The EnvelopeElement takes exactly same args

```
recorder EnvelopeElement <-file $fileName> <-time> <-ele $tg1 $tg2 ...> $arg1 $arg2 ...  
                        <-xml $fileName>                <-eleRange $tgS $tgE>  
                        <-binary $fileName>                <-region $rTag>  
                        <-tcp $inetAddr>
```


Example using recorders(s dofExample2.tcl)

```
# create model & analysis
```

```
...
```

```
#create recorders
```

```
recorder Node -file node1.out -time -node 2 -dof 1 disp
```

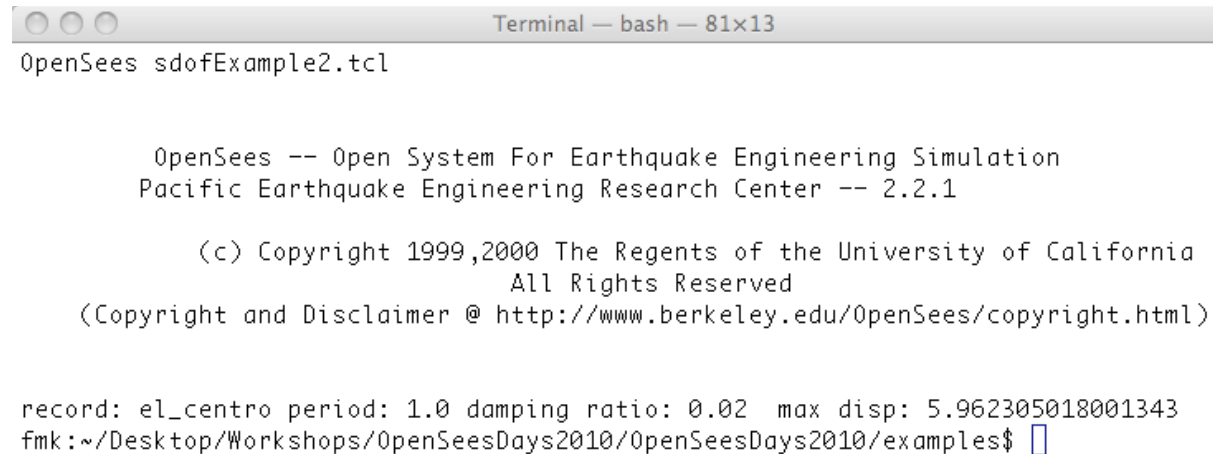
```
recorder Element -file ele1.out -time -ele 1 material stress
```

```
#perform analysis
```

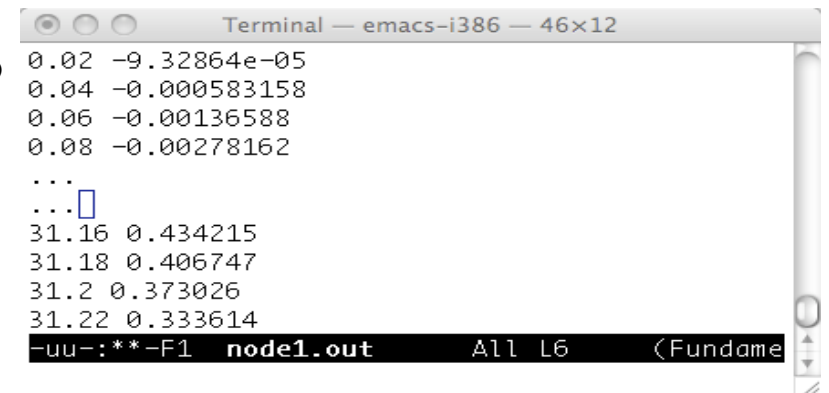
```
while {$ok == 0 && $t < $maxT} {  
  set ok [analyze 1 $dT]  
  set time [getTime]  
  set d [nodeDisp 2 1]  
  if {$d > $maxD} {  
    set maxD $d  
  } elseif {$d < [expr -$maxD]} {  
    set maxD [expr -$d]  
  }  
  set t [expr $t + $dT]  
}
```

```
puts "record: $record period: $Tn damping ratio: $dampRatio
```

```
wipe
```



```
Terminal — bash — 81x13  
OpenSees dofExample2.tcl  
  
OpenSees -- Open System For Earthquake Engineering Simulation  
Pacific Earthquake Engineering Research Center -- 2.2.1  
  
(c) Copyright 1999,2000 The Regents of the University of California  
All Rights Reserved  
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)  
  
record: el_centro period: 1.0 damping ratio: 0.02 max disp: 5.962305018001343  
fmk:~/Desktop/Workshops/OpenSeesDays2010/OpenSeesDays2010/examples$
```



```
Terminal — emacs-i386 — 46x12  
0.02 -9.32864e-05  
0.04 -0.000583158  
0.06 -0.00136588  
0.08 -0.00278162  
...  
31.16 0.434215  
31.18 0.406747  
31.2 0.373026  
31.22 0.333614  
-uu-: ** -F1 node1.out All L6 <Fundame
```



```
Terminal — emacs-i386 — 59x10
0.020000      -0.00009328642560936581
0.040000      -0.00058315832318352428
0.060000      -0.00136588102214891529
...
...
31.160000     0.43421515701762919415
31.180000     0.40674718617127236797
31.200000     0.37302636227890006992
-uu-:**-F1  node.out      Top L5      (Fundamental)-----
```

```
Terminal — emacs-i386 — 46x12
0.02 -9.32864e-05
0.04 -0.000583158
0.06 -0.00136588
0.08 -0.00278162
...
...
31.16 0.434215
31.18 0.406747
31.2 0.373026
31.22 0.333614
-uu-:**-F1  node1.out    All L6      (Fundame
```

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output

• Modeling in OpenSees

- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion



KISS is an acronym for "**Keep it simple, stupid**" as a design principle noted by the U.S. Navy in 1960. The KISS principle states that most systems work best if they are kept simple rather than made complex; therefore simplicity should be a key goal in design and unnecessary complexity should be avoided. source: wikipedia

Keep it simple, student! source: wikipedia

```
model Basic -ndm 2 -ndf 2
```

```
#node $tag $xCrd $yCrd
```

```
node 1 0.0 0.0
```

```
node 2 144.0 0.0
```

```
node 3 168.0 0.0
```

```
node 4 72.0 96.0
```

```
#fix $nodeTag $xFix $yFix
```

```
fix 1 1 1
```

```
fix 2 1 1
```

```
fix 3 1 1
```

```
#uniaxialMaterial Elastic $tag $E
```

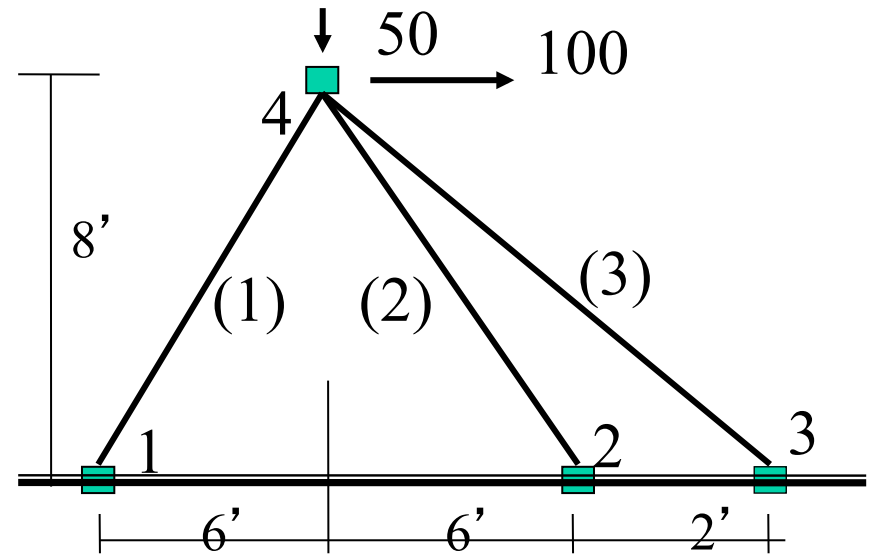
```
uniaxialMaterial Elastic 1 3000.0
```

```
#element truss $tag $iNode $jNode $A $matTag
```

```
element truss 1 1 4 10.0 1
```

```
element truss 2 2 4 5.0 1
```

```
element truss 3 3 4 5.0 1
```



	E	A
1	3000	10
2	3000	5
3	3000	5

```
timeSeries Linear 1
```

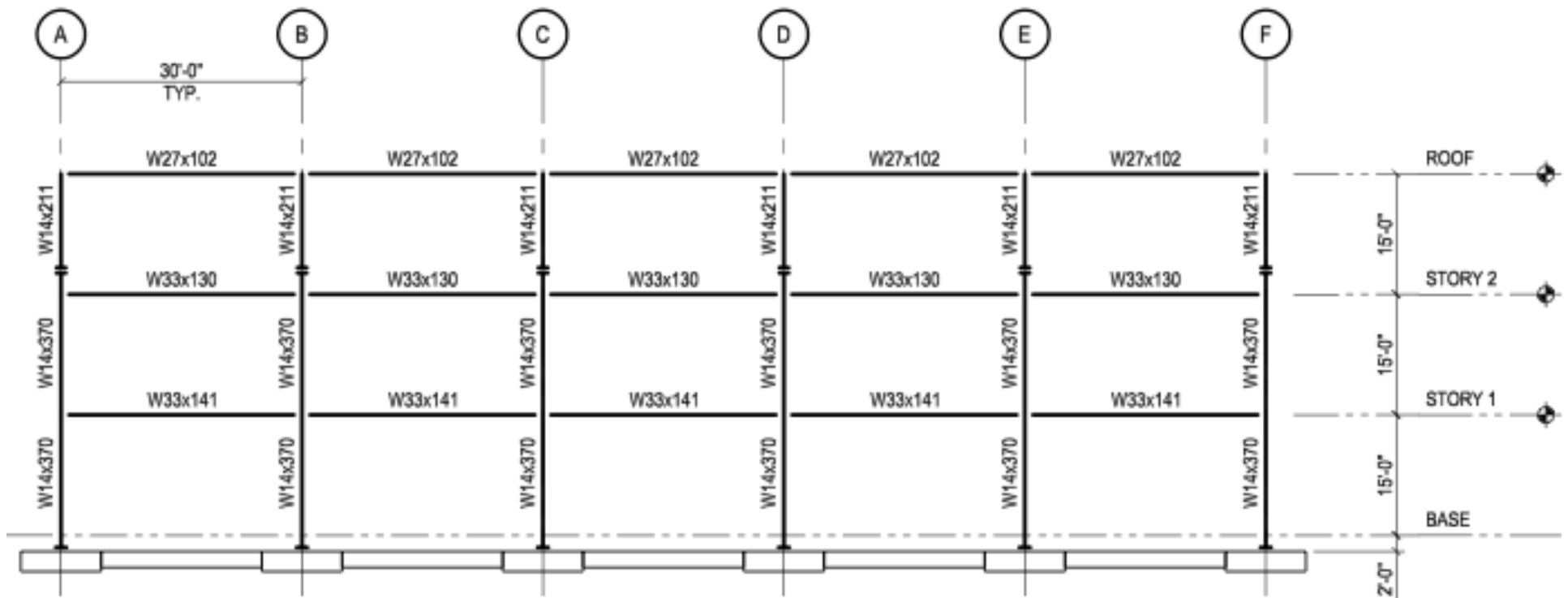
```
#pattern Plain $tag $tsTag
```

```
pattern Plain 1 1 {
```

```
  #load $nodeTag $xForce $yForce
```

```
  load 4 100.0 -50.0
```

```
}
```



MF ELEVATION ON LINE 1 (LINE 5 SIM.)

1"=20'

```

# define structure-geometry parameters
set NStories 3. # number of stories
set NodalMass2V 0.105; # mass at each column node on Floor
set

```

3 Story 5 Bay Moment Frame

Original Code >350 lines

```

noc uniaxial # command: forceBeamColumn $eleTag $iNode $jNode $numInt
noc # eleID convention: "1xy" where 1 = col, x = Pier #, y

```

If I had to do this I would
want a GUI too!

```

$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
noc element forceBeamColumn 112 122 132 $NIPcol 12 $PDeltaT
node 32 $Pier3 $Floor2 -mass $NodalMass2H $NodalMass2V 0.0;
node 42 $Pier4 $Floor2 -mass $NodalMass2H $NodalMass2V 0.0;

```

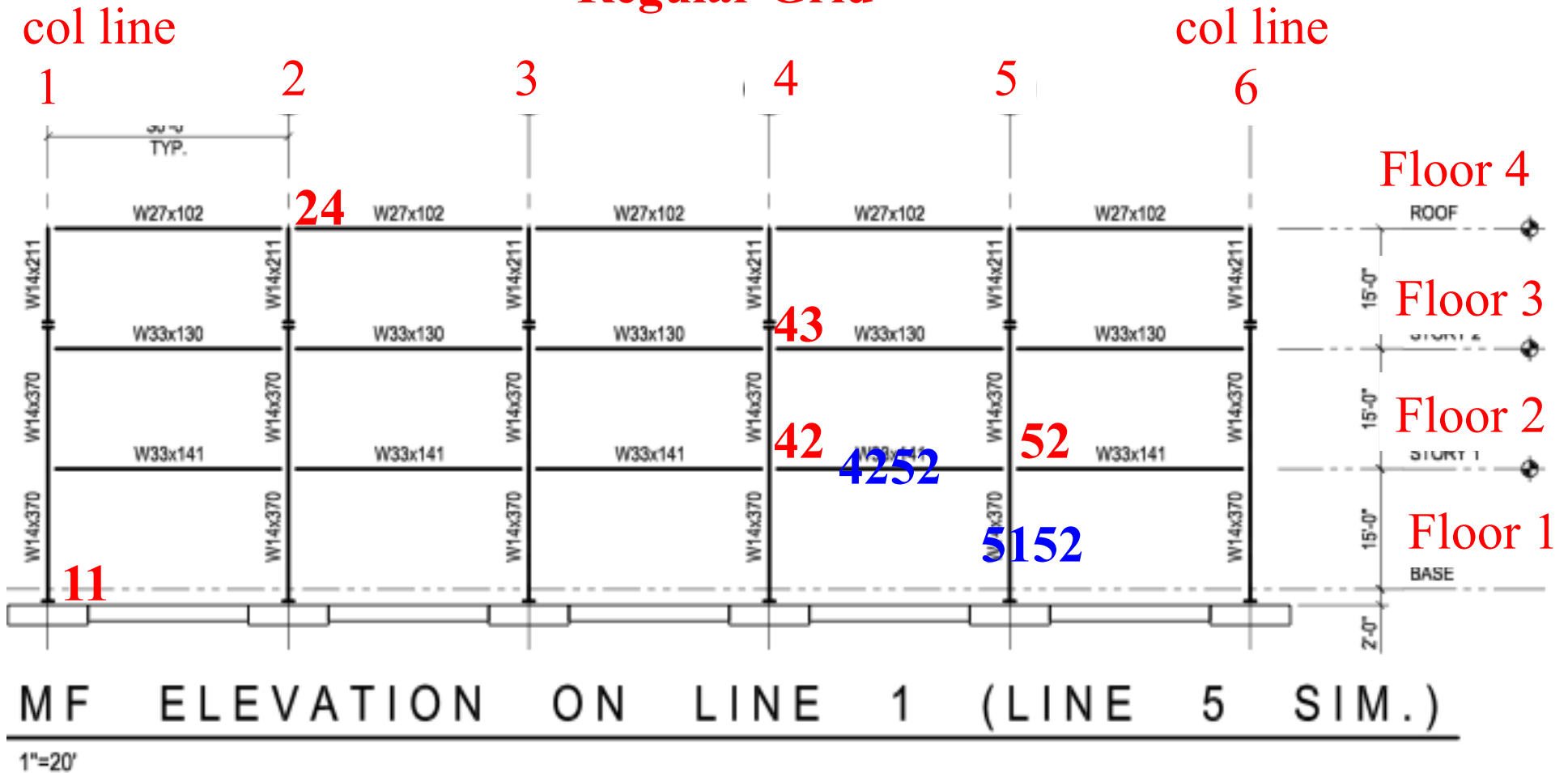
The SCRIPTS you are submitting
to OpenSees
are PROGRAMS
you are submitting to a powerful
INTERPRETER ..

**SO LETS START
PROGRAMMING!**



THINK Before You Type

Steel W Sections & Regular Grid



Nodes # \$col\$floor
Elements # \$iNode\$jNode

(if more than 10 col lines or floors,
 start numbering at 10,
 if > 100, at 100,)

MRF1.tcl

Same
model
in 35
lines!

```
model Basic -ndm 2 -ndf 3
source Steel2d.tcl

# set some lists containing floor and column line locations and nodal masses
set floorLocs {0. 204. 384. 564.}; # floor locations in inches
set colLocs {0. 360. 720. 1080. 1440. 1800.}; #column line locations in inches
set massesX {0. 0.419 0.419 0.430}; # mass at nodes on each floor in x dirn
set massesY {0. 0.105 0.105 0.096}; # " " " " " " in y dirn

# add nodes at each floor at each column line location & fix nodes if at floor 1
foreach floor {1 2 3 4} floorLoc $floorLocs massX $massesX massY $massesY {
  foreach colLine {1 2 3 4 5 6} colLoc $colLocs {
    node $colLine$floor $colLoc $floorLoc -mass $massX $massY 0.
    if {$floor == 1} {fix $colLine$floor 1 1 1}
  }
}

#uniaxialMaterial Steel02 $tag $Fy $E $b $R0 $scr1 $scr2
uniaxialMaterial Steel02 1 50.0 29000. 0.003 20 0.925 0.15; # material to be used for steel elements

# set some list for col and beam sizes
set colSizes {W14X370 W14X370 W14X211}; #col sizes stories 1, 2 and 3
set beamSizes {W33X141 W33X130 W27X102}; #beams sizes floor 1, 2, and 3

# add columns at each column line between floors
geomTransf PDelta 1
foreach colLine {1 2 3 4 5 6} {
  foreach floor1 {1 2 3} floor2 {2 3 4} {
    set theSection [lindex $colSizes [expr $floor1 -1]]; # obtain section size for column
    ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1 -nip 5
  }
}

#add beams between column lines at each floor
geomTransf Linear 2
foreach colLine1 {1 2 3 4 5} colLine2 {2 3 4 5 6} {
  foreach floor {2 3 4} {
    set theSection [lindex $beamSizes [expr $floor -2]]; # obtain section size for floor
    ForceBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2
  }
}
```

Steel2.tcl – contains a library of procedures

```
#WinXlb/f "Area(in2) d(in) bf(in) tw(in) tf(in) Ixx(in4) Iyy(in4)"
array set WSection {
  W44X335    "98.5 44.0 15.9 1.03 1.77 31100 1200 74.7"
  W44X290    "85.4 43.6 15.8 0.865 1.58 27000 1040 50.9"
  W44X262    "76.9 43.3 15.8 0.785 1.42 24100 923 37.3"
  W44X230    "67.7 42.9 15.8 0.710 1.22 20800 796 24.9"
  W40X593    "174 43.0 16.7 1.79 3.23 50400 2520 445"
  W40X503    "148 42.1 16.4 1.54 2.76 41600 2040 277"
  ...
}

proc ElasticBeamWSection2d {eleTag iNode jNode sectType E transfTag {Orient XX}} {
  global WSection
  global in
  set found 0
  foreach {section prop} [array get WSection $sectType] {
    set propList [split $prop]
    set A [expr [lindex $propList 0]*$in*$in]
    set Ixx [expr [lindex $propList 5]*$in*$in*$in*$in]
    set Iyy [expr [lindex $propList 6]*$in*$in*$in*$in]
    if {$Orient == "YY" } {
      element elasticBeamColumn $eleTag $iNode $jNode $A $E $Iyy $transfTag
    } else {
      element elasticBeamColumn $eleTag $iNode $jNode $A $E $Ixx $transfTag
    }
  }
}
}
```

```

proc ForceBeamWSection2d {eleTag iNode jNode sectType matTag transfTag args} {

  global FiberSteelWSection2d
  global ElasticSteelWSection2d

  set Orient "XX"
  if {[lsearch $args "YY"] != -1} {
    set Orient "YY"
  }

  set nFlange 8
  if {[lsearch $args "-nFlange"] != -1} {
    set loc [lsearch $args "-nFlange"]
    set nFlange [lindex $args [expr $loc+1]]
  }

  set nWeb 4
  if {[lsearch $args "-nWeb"] != -1} {
    set loc [lsearch $args "-nWeb"]
    set nWeb [lindex $args [expr $loc+1]]
  }

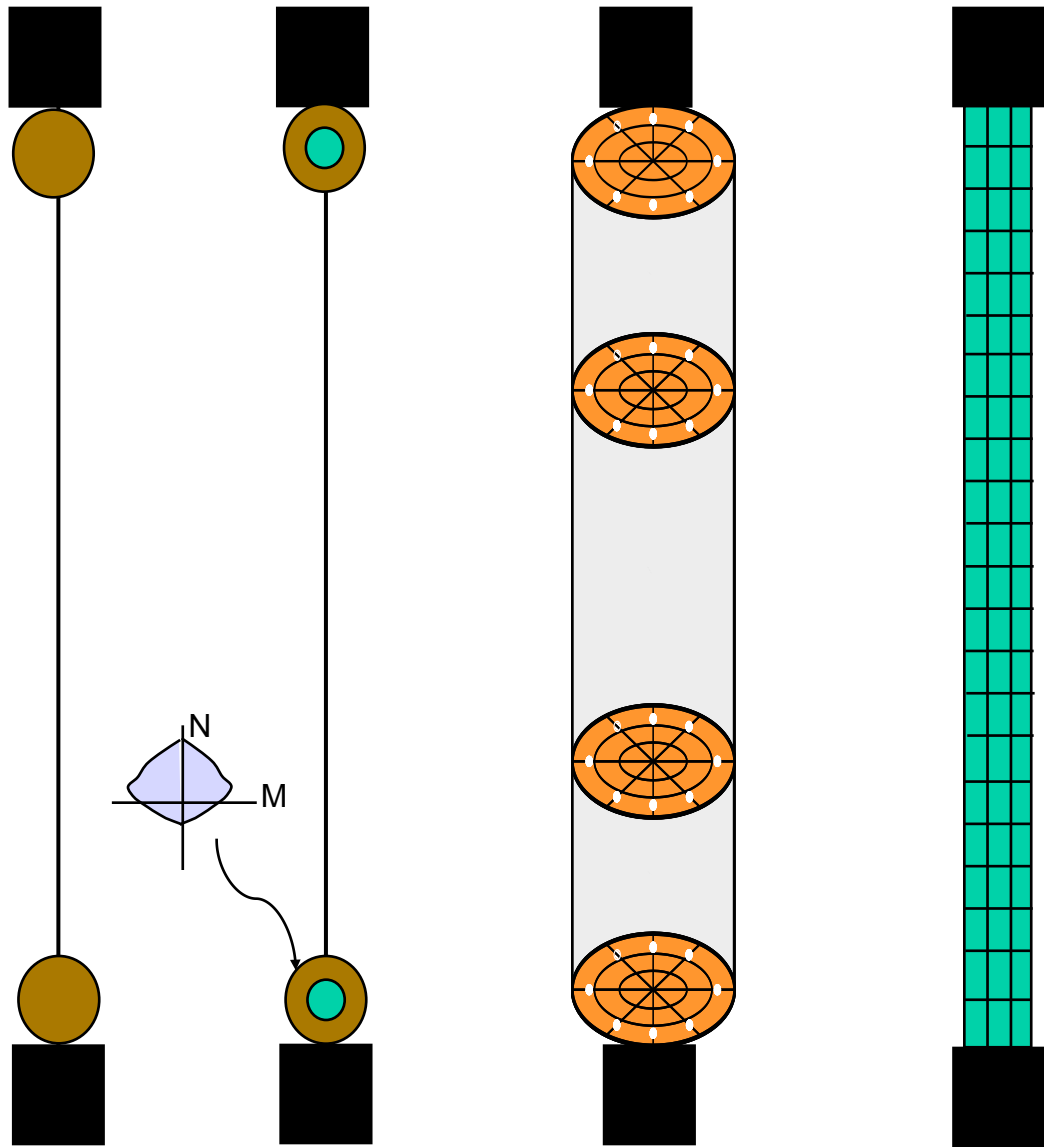
  set nip 4
  if {[lsearch $args "-nip"] != -1} {
    set loc [lsearch $args "-nip"]
    set nip [lindex $args [expr $loc+1]]
  }

  if {[lsearch $args "-elasticSection"] != -1} {
    set loc [lsearch $args "-elasticSection"]
    set E [lindex $args [expr $loc+1]]
    ElasticSteelWSection2d $eleTag $sectType $E $Orient
  } else {
    FiberSteelWSection2d $eleTag $sectType $matTag $nFlange $nWeb $Orient
  }

  element forceBeamColumn $eleTag $iNode $jNode $nip $eleTag $transfTag
}

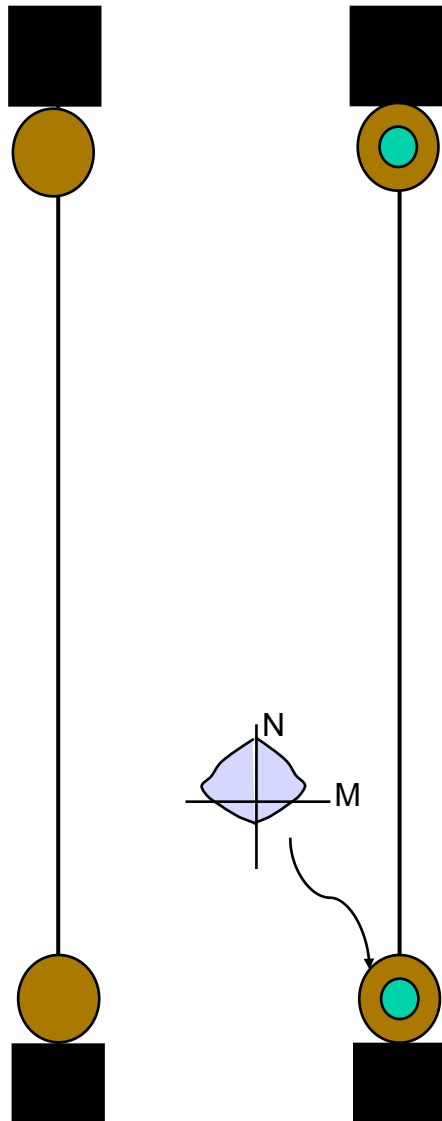
```

Nonlinear Structural Beam-Column Models



Source: Filip Filippou, UC Berkeley

Beam-Column Models: Concentrated Inelasticity

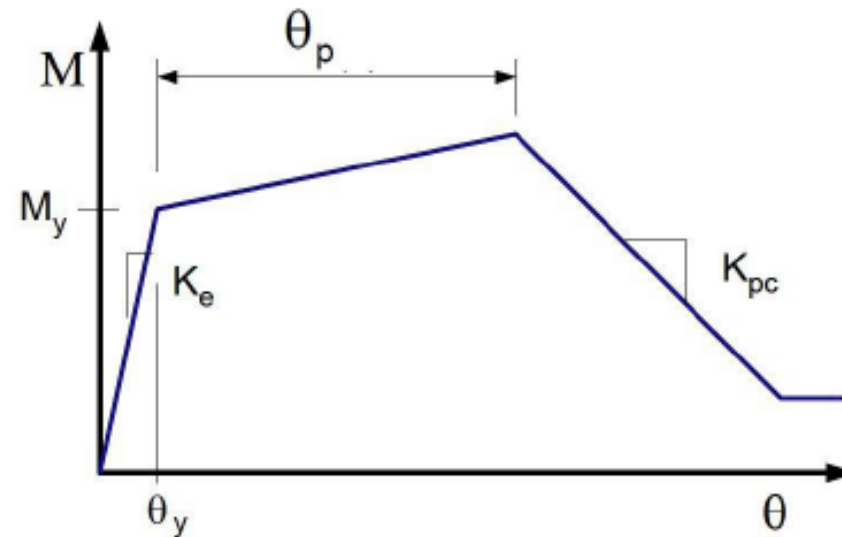


- Concentrated plasticity models = one or more rotational springs at each end + elastic element
 - Advantages: relatively simple, good(?) for interface effects (e.g. shear sliding, rotation due to bar pull-out)
 - Disadvantages: properties of rotational spring depend on geometry and moment distribution; relation to strains requires “plastic hinge length”; interaction of axial force, moment and shear ???; generality??? numerical robustness???

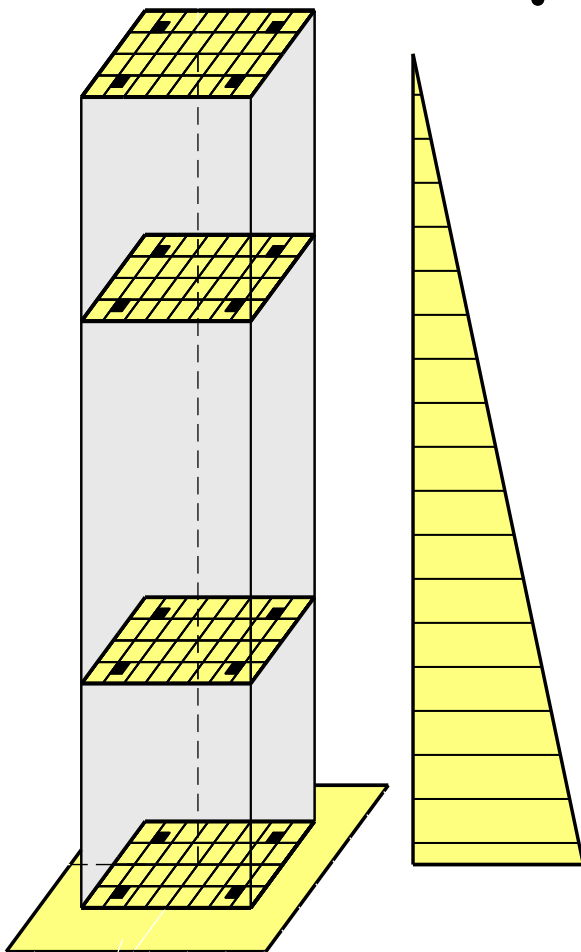


Elastic Beam Column elements
 With Modified Stiffness
 (Ibarra and Krawinkler, 2005)

Zero Length elements
 (rotational springs)
 (PEER/ATC 2010)
 Hysteretic Material



Beam-Column Models: Distributed Inelasticity



- Distributed inelasticity models (1d FE model) = consistent integration of section response at specific “control” or monitoring points
 - Advantages:
 - versatile and consistent;
 - section response from integration of material response thus N-My-Mz interaction (Bernoulli) (shear and torsion?? Timoshenko, ...)
 - thus numerical robustness is possible
 - Disadvantages:
 - can be expensive (what is the price \$\$\$\$) with “wasted sections” for localized inelasticity
 - inaccuracy of local response (localization)
 - thus better understanding of theory for interpretation of local response and damage is necessary

2 Element Types to Choose From:

1. **dispBeamColumn** – displacement based

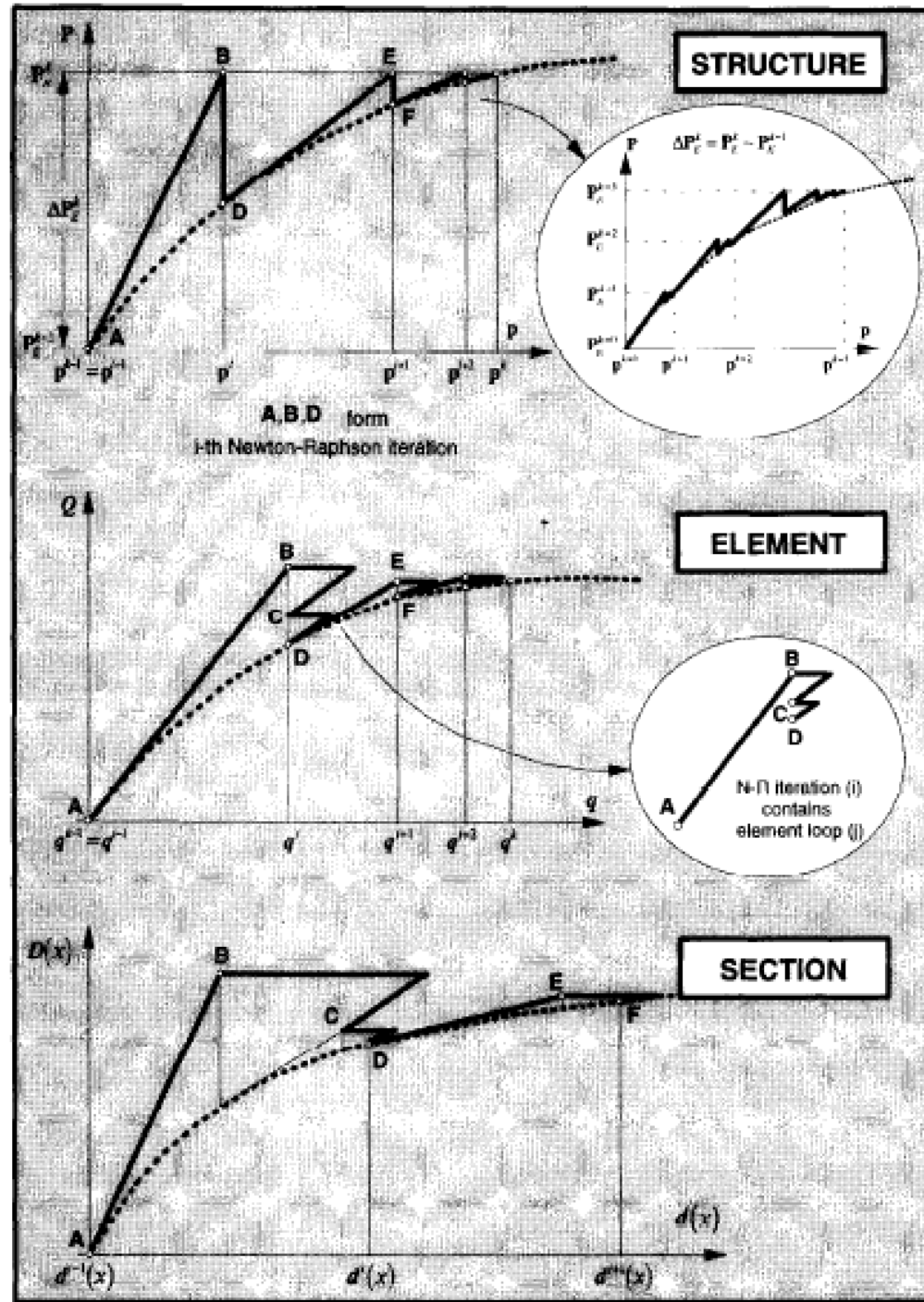
- Follows standard finite element procedures where we interpolate deformations at sections (integration points) using an approximate displacement field (constant axial deformation and linear curvature distribution)
- Like in typical finite element analysis, mesh refinement is needed to capture higher order distributions

2. **forceBeamColumn** – force based element

- Uses fact that we do know what the force distribution is along the element. No approximation.
- Using the force approach means iteration to determine compatible forces given nodal displacements (does not always converge).

MIXED FORMULATION OF NONLINEAR BEAM FINITE ELEMENT

E. Spacone,† V. Ciampiti and F. C. Filippou†

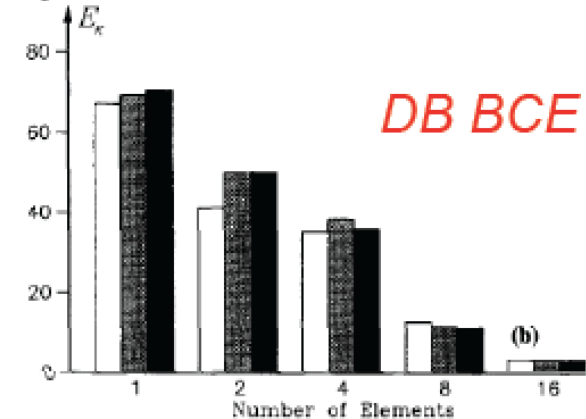
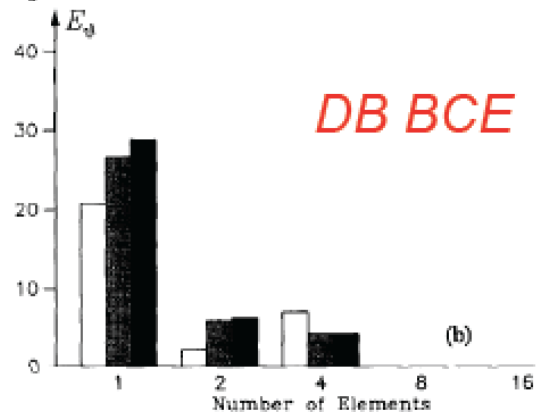
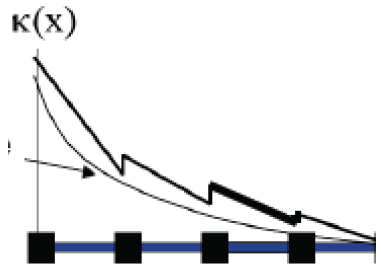
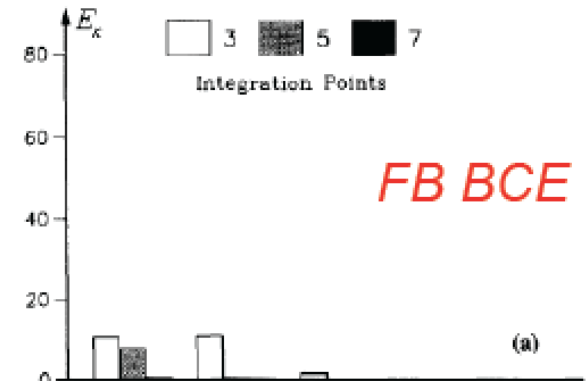
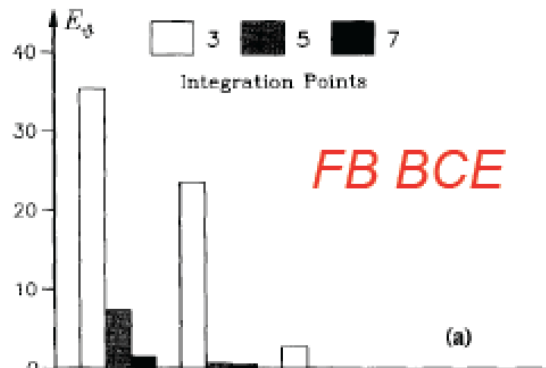
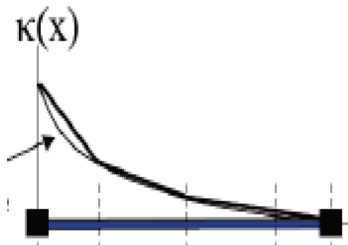


Which is more 'accurate' ?

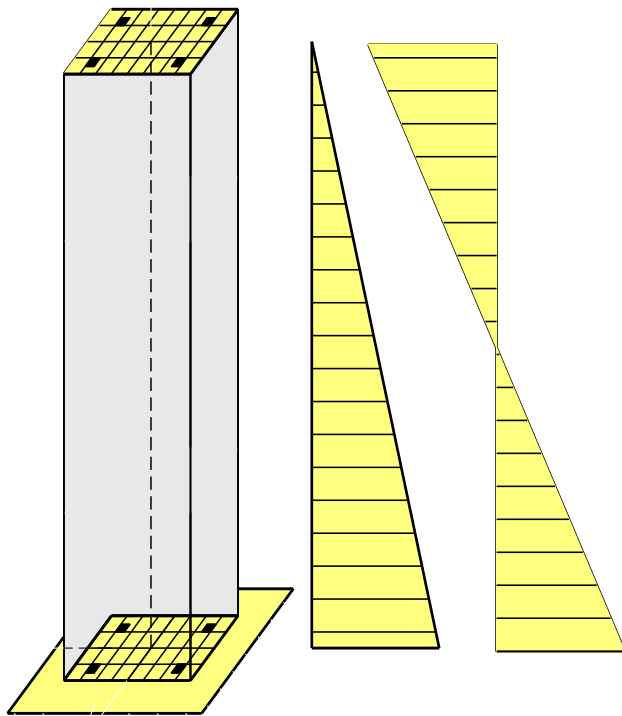
Neuenhofer, A., and F. C. Filippou, (1997). "Evaluation of Nonlinear Frame Finite Element Models." *Journal of Structural Engineering*, 123(7): 958-966.

Rotation error (node B)
Global response

Curvature error (node A)
Local response

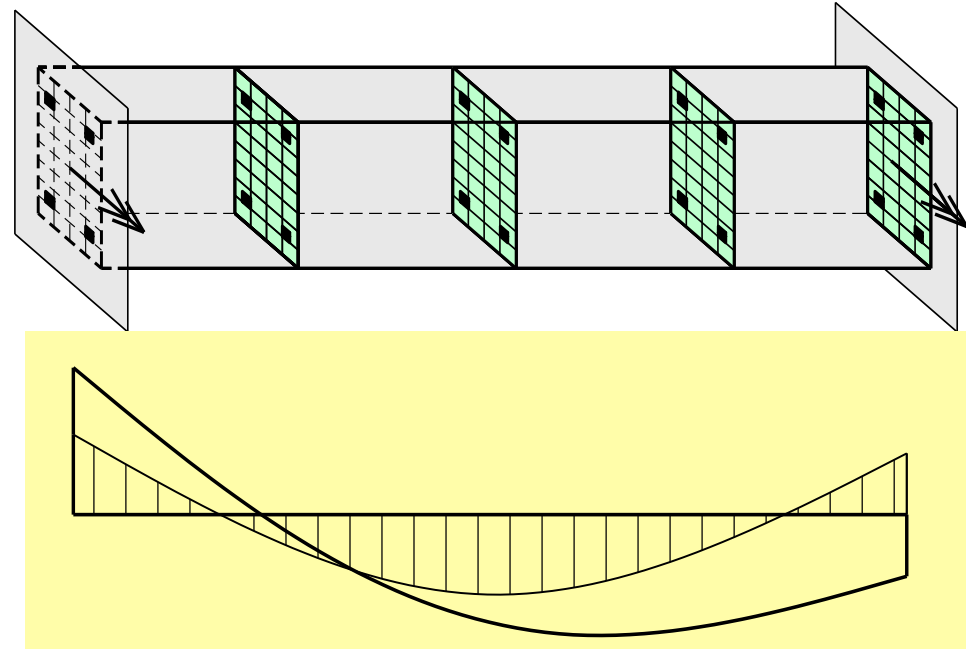


“Economic” Distributed inelasticity models for Columns



- **beamWithHinges**
- 2 monitoring points at element ends (inelastic zones), elastic section in middle
 - Good for columns and girders with low gravity loads
 - N-My-Mz interaction straightforward
 - shear and torsion ???
 - Consistent location of integration point?
 - Value of fixed length of inelastic zone?
 - **Good for softening response** (Fenves/Scott, ASCE 2006)
 - Hardening response → Spreading inelastic zone element SIZE (Lee/Filippou, ASCE 2009 to appear)

Good Distributed inelasticity models for Girders



- For girders with significant gravity loads
 - 4-5 integration points are advisable (“good plastic hinge length value with corresponding integration weights)
 - One element per girder -> avoid very high local deformation values

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees

• Nonlinear Analysis in OpenSees

- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Why Nonlinear Analysis

- **Geometric Nonlinearities** - occur in model when applied load causes large displacement and/or rotation, large strain, or a combo of both
- **Material nonlinearities** - nonlinearities occur when material stress-strain relationship depends on load history (plasticity problems), load duration (creep problems), temperature (thermoplasticity), or combo of all.
- **Contact nonlinearities** - occur when structure boundary conditions change because of applied load.

Nonlinear Analysis is Harder

- It requires **much** more thought when setting up the model
- It requires more thought when setting up the analysis
- It takes more computational time.
- It does not always converge.
- It does not always converge to the correct solution.

BUT if you are using a Finite element code the Problem probably Requires a
Nonlinear Analysis

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

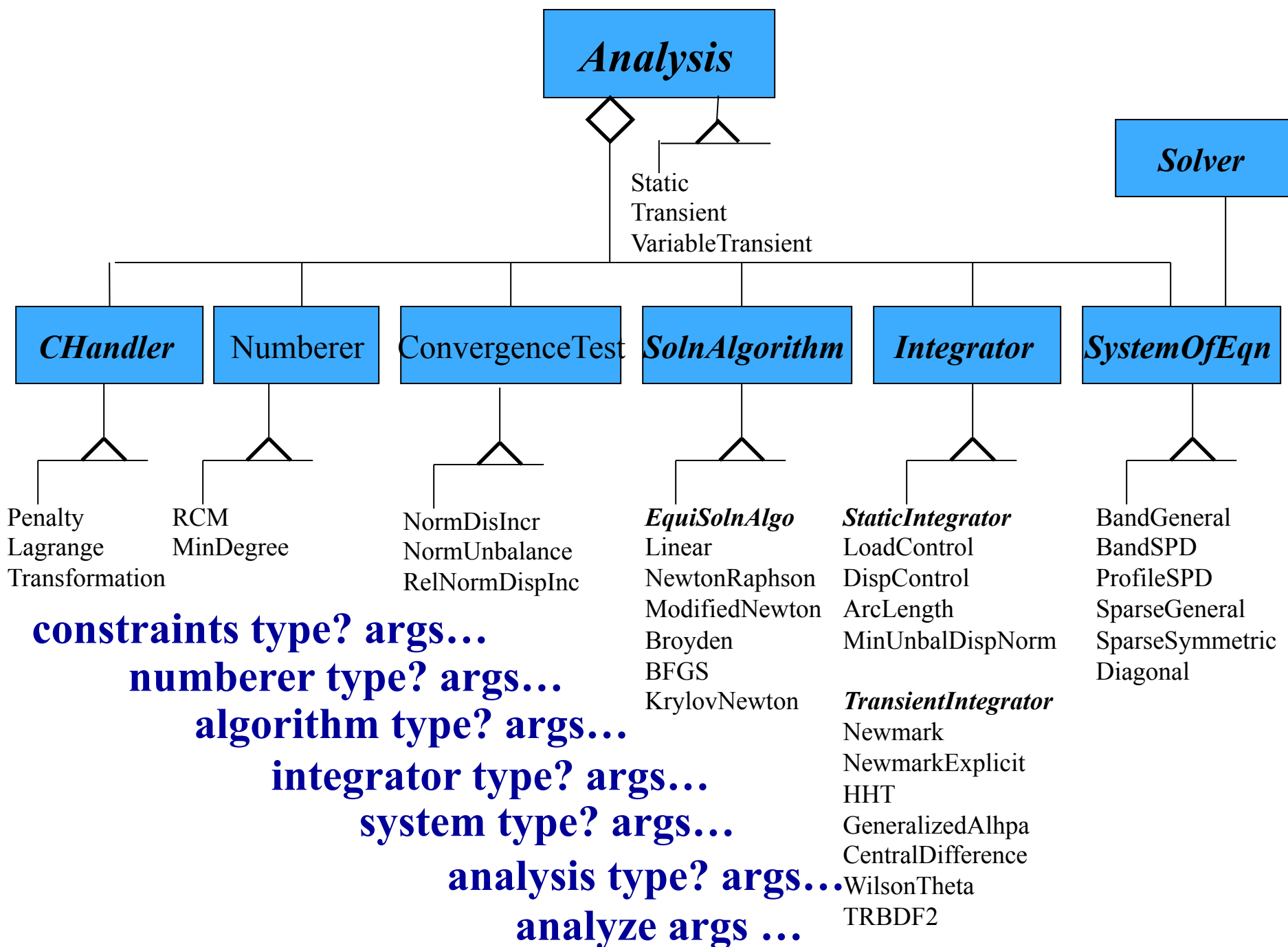
CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

99% Probability: if Fails in First Step THERE IS A PROBLEM IN YOUR MODEL



test command:

- to specify when convergence has been achieved

all look at system: $\mathbf{KU} = \mathbf{R}$

- Norm Unbalance

$$\sqrt{\mathbf{R}^T \mathbf{R}} < \mathbf{tol}$$

test NormUnbalance tol? numIter? <flag?>

- Norm Displacement Increment

$$\sqrt{\mathbf{U}^T \mathbf{U}} < \mathbf{tol}$$

test NormDispIncr tol? numIter? <flag?>

- Norm Energy Increment

$$\frac{1}{2} (\mathbf{U}^T \mathbf{R}) < \mathbf{tol}$$

test NormEnergyIncr tol? numIter? <flag?>

- Relative Tests

test RelativeNormUnbalance tol? numIter? <flag?>

test RelativeNormDispIncr tol? numIter? <flag?>

test RelativeNormEnergyIncr tol? numIter? <flag?>

numberer command:

- to specify how the degrees of freedom are numbered

- Plain Numberer

nodes are assigned dof arbitrarily

numberer Plain

- RCM Numberer

nodes are assigned dof using the Reverse Cuthill-McKee algorithm

numberer RCM

- AMD Numberer

nodes are assigned dof using the Approx. MinDegree algorithm

numberer AMD

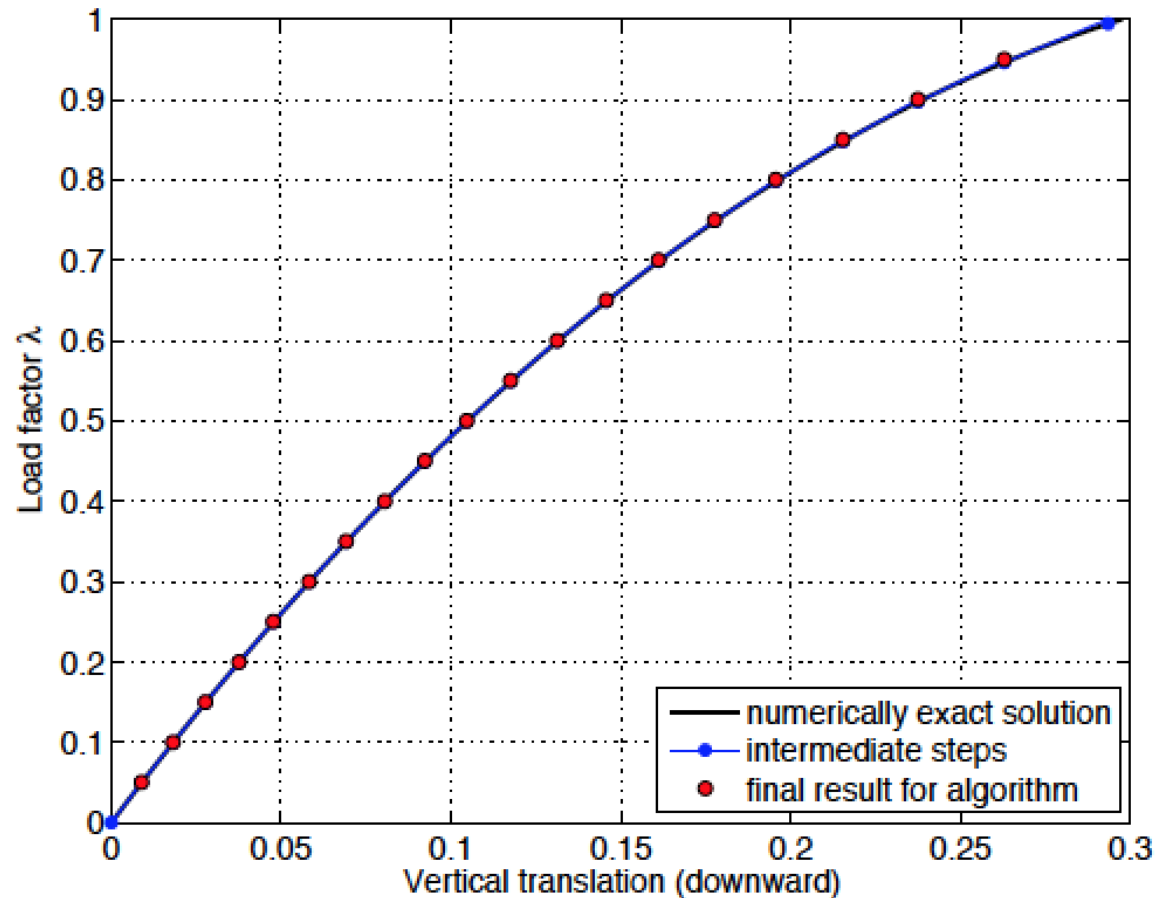
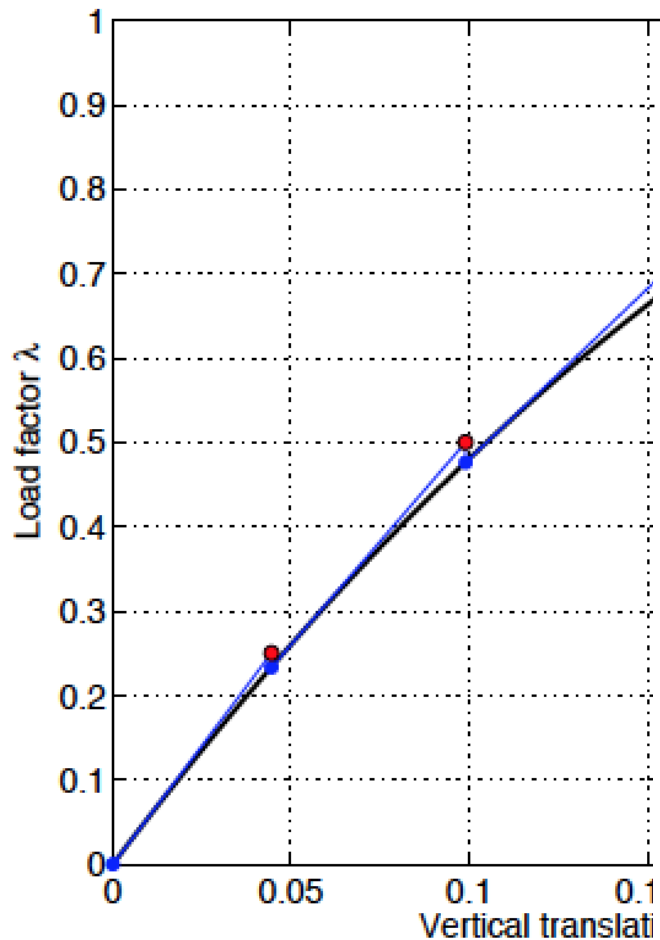
- numbering has an impact on performance of banded and profile solvers. The sparse solvers all use their own optimal numbering schemes.

algorithm command:

- Algorithm specifies the trial steps taken to solve the nonlinear equation for each step.
- Number of options (**ALL** have their place)

algorithm Linear

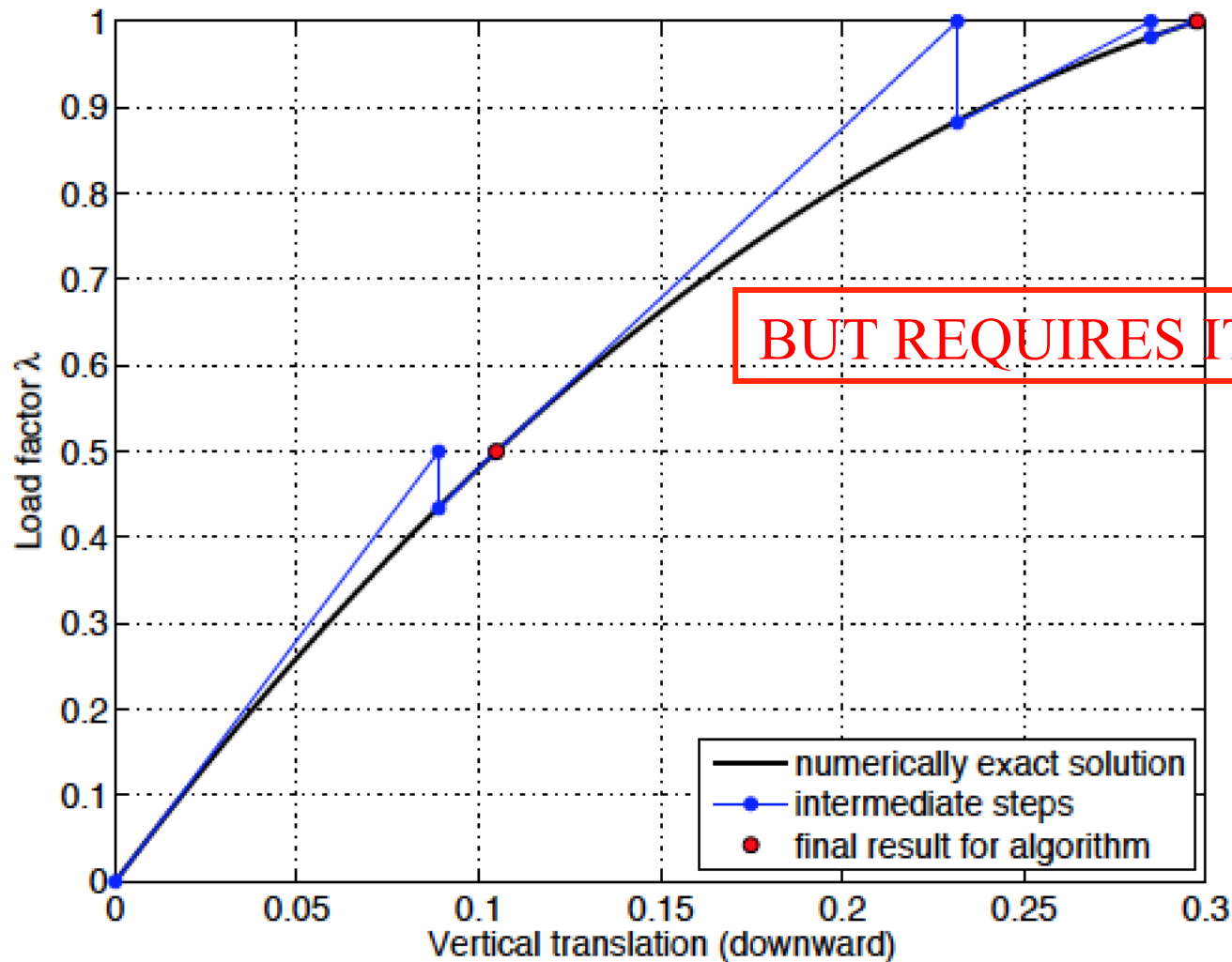
```
theIntegrator->formUnbalance();  
theIntegrator->formTangent();  
theSOE->solve()  
theIntegrator->update(theSOE->getX());
```



algorithm Newton

(Newton-Raphson)

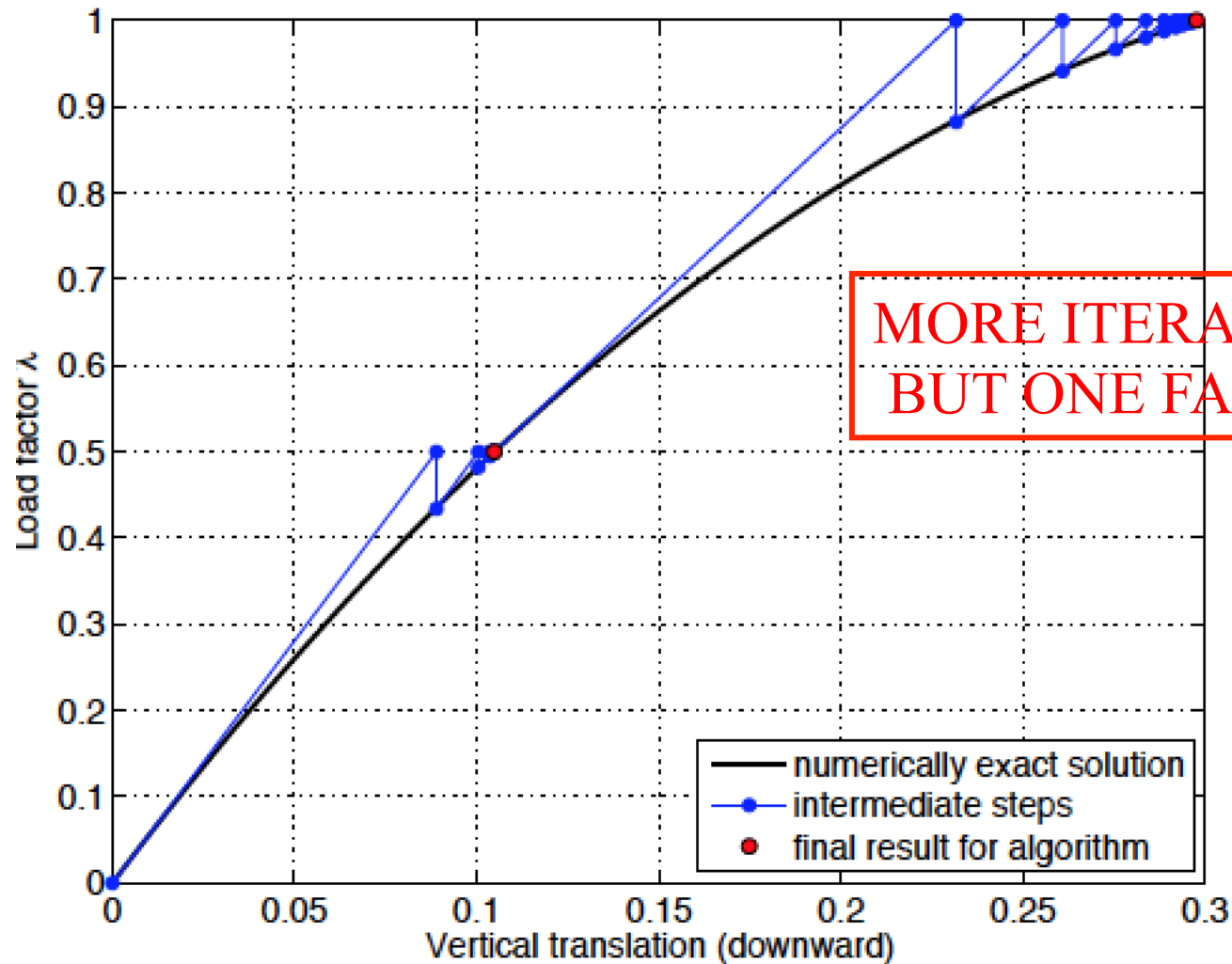
```
theIntegrator->formUnbalance();  
do {  
  theIntegrator->formTangent();  
  theSOE->solve()  
  theIntegrator->update(theSOE->getX());  
  theIntegrator->formUnbalance();  
} while (theTest->test() == fail)
```



algorithm ModifiedNewton

(Modified Newton-Raphson)

```
theIntegrator->formUnbalance();  
theIntegrator->formTangent();  
do {  
  theSOE->solve()  
  theIntegrator->update(theSOE->getX());  
  theIntegrator->formUnbalance();  
} while (theTest->test() == fail)
```



algorithm NewtonLineSearch

- Adds line search algorithm to the search algorithm

algorithm KrylovNewton

- A quasi-newton method – in which approximations of the inverse matrix are made from displacement vectors of previous trial steps. **"Nonlinear Finite Element Procedures" K.J.Bathe**

DISCLAIMER

All those functions were nice smooth continuous functions. We model with material discontinuities, contact, ... and thus are NOT SMOOTH ..

CONVERGENCE IS NOT GAURANTEED ..

CONVERGENCE TO CORRECT SOLUTION IS NOT GAURANTEED.



constraints command:

- to specify how the constraints are enforced

$$\mathbf{U}_c = \mathbf{C}_r \mathbf{C}_c \mathbf{U}_r$$

$$\mathbf{C} \mathbf{U} = \mathbf{0}$$

$$\mathbf{T} \mathbf{U}_r = [\mathbf{U}_r \ \mathbf{U}_c]^T$$

$$[\mathbf{C}_r \ \mathbf{C}_c] [\mathbf{U}_r \ \mathbf{U}_c]^T = \mathbf{0}$$

•Transformation Handler

$$\mathbf{K}^* \mathbf{U}_r = \mathbf{R}^* \quad \mathbf{K}^* = \mathbf{T}^T \mathbf{K} \mathbf{T}$$
$$\mathbf{R}^* = \mathbf{T}^T \mathbf{R}$$

constraints Transformation

in OpenSees currently don't allow retained node in one constraint to be a constrained node in another constraint

•Lagrange Handler

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{R} \\ \mathbf{Q} \end{bmatrix}$$

constraints Lagrange

•Penalty Handler

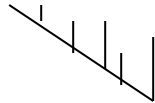
$$[\mathbf{K} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{C}] \mathbf{U} = [\mathbf{R} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{Q}]$$

constraints Penalty α_{sp} ? α_{mp} ?

system command:

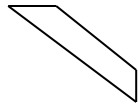
- to specify how matrix equation $KU = R$ is stored and solved

- Profile Symmetric Positive Definite (SPD)



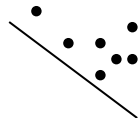
system ProfileSPD

- Banded Symmetric Positive Definite



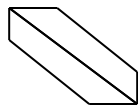
system BandSPD

- Sparse Symmetric Positive Definite



system SparseSPD

- Banded General



system BandGeneral

- Sparse Symmetric

system SparseGeneral

**If you have a large system
Use one of these**

system Umfpack

integrator command:

- determines the predictive step for time $t+\delta t$
- specifies the tangent matrix and residual vector at any iteration
- determines the corrective step based on ΔU

1. Static Integrators for Use in Static Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{P}^* - \mathbf{F}_R(\mathbf{U})$$

2. Transient Integrators for Use in Transient Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \dot{\mathbf{U}}, \ddot{\mathbf{U}}) = \mathbf{P}(t) - \mathbf{F}_I(\ddot{\mathbf{U}}) - \mathbf{F}_R(\mathbf{U}, \dot{\mathbf{U}})$$

Static Integrators

$$\mathbf{R}(\mathbf{U}, \boldsymbol{\lambda}) = \boldsymbol{\lambda} \mathbf{P}^* - \mathbf{F}(\mathbf{R}(\mathbf{U}))$$

at each step solving for $\boldsymbol{\lambda}$

▪ Load Control

$$\boldsymbol{\lambda}_n = \boldsymbol{\lambda}_{n-1} + \Delta \boldsymbol{\lambda} \quad \text{integrator LoadControl } \Delta \boldsymbol{\lambda}$$

*does not require a reference load, i.e. loads in load patterns with Linear series and all other loads constant.

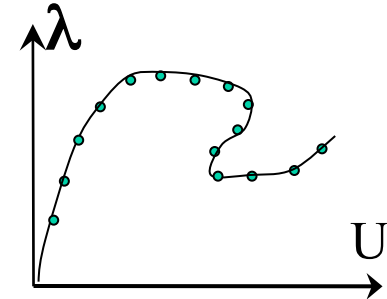
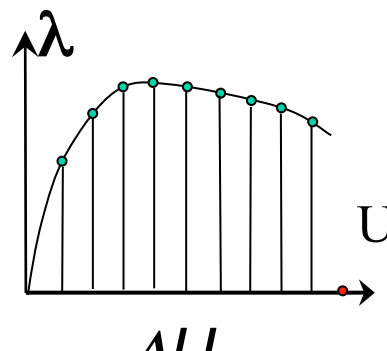
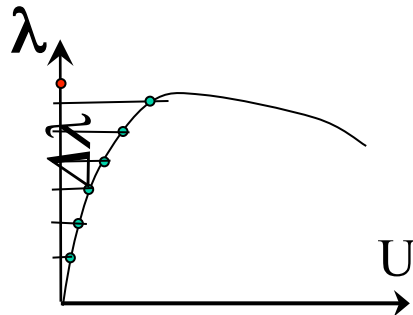
▪ Displacement Control

$$\mathbf{U}_{j_n} = \mathbf{U}_{j_{n-1}} + \Delta \mathbf{U}_j$$

integrator DisplacementControl node dof $\Delta \mathbf{U}$

▪ Arc Length

$$\Delta \mathbf{U}_n^T \Delta \mathbf{U}_n + \alpha^2 \Delta \boldsymbol{\lambda}_n^2 = \Delta s^2 \quad \text{integrator ArcLength } \alpha \Delta s$$



Transient Integrators

- Explicit:

$$D_{n+1} = f(P_n, D_n, V_n, A_n, D_{n-1}, V_{n-1}, A_{n-1}, \dots)$$

integrator CentralDifference

integrator NewmarkExplicit \$gamma

integrator HHTExplicit \$alpha

....

1. all Need Linear Algorithm

2. in absence of damping all require Positive Definite mass matrix.

- Implicit

$$D_{n+1} = f(V_{n+1}, A_{n+1}, D_n, V_n, A_n, D_{n-1}, V_{n-1}, A_{n-1}, \dots)$$

integrator Newmark \$gamma \$beta

integrator HHT \$alpha

integrator TRBDF2

...

Stability & Linear Systems

- Stability (bounded solution) and Accuracy are the most talked about properties of transient integration schemes.
- For most integration schemes, the stability and accuracy provisions you read about are provided FOR LINEAR DYNAMICAL SYSTEMS.
- Conditionally Stable: numerical procedure leads to a BOUNDED solution if time step is smaller than some stability limit. Conditional stability requires time step to be inversely proportional to highest frequency.
- Unconditionally Stable: solution is bounded regardless of the time step.

Stability Limits Common Integrators

Central Difference is conditionally stable if:

$$\frac{\Delta t}{T_n} < \frac{1}{\pi} \quad (.318)$$

Newmark is unconditionally stable if:

$$\frac{\Delta t}{T_n} \leq \frac{1}{\pi\sqrt{2}} \frac{1}{\sqrt{\gamma - 2\beta}}$$

Average Acceleration
(Trapezoidal)

$$(\gamma = \frac{1}{2}, \beta = \frac{1}{4})$$

Unconditionally Stable

Linear Acceleration

$$(\gamma = \frac{1}{2}, \beta = \frac{1}{6})$$

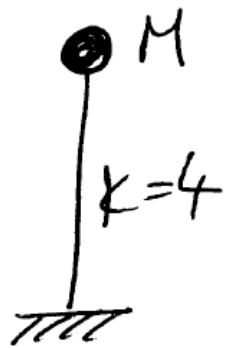


But Conditionally stable if: $\frac{\Delta t}{T_n} < 0.55$

Example

(see "Dynamics of Structures" A.K. Chopra, section 5.5)

Free vibration (exl. bcl)
(ex 2. bcl)

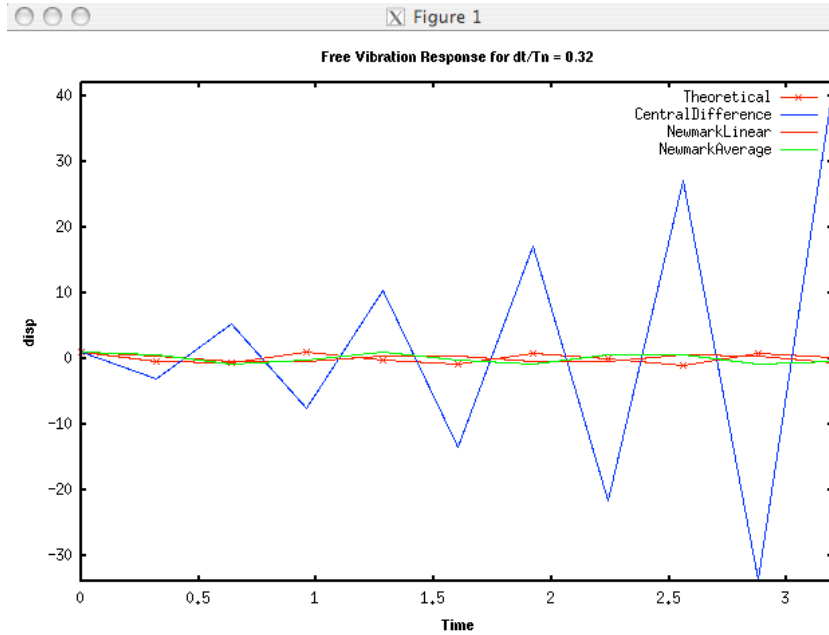


$$u(0) = 1$$
$$u'(0) = 0$$
$$p(t) = \phi$$

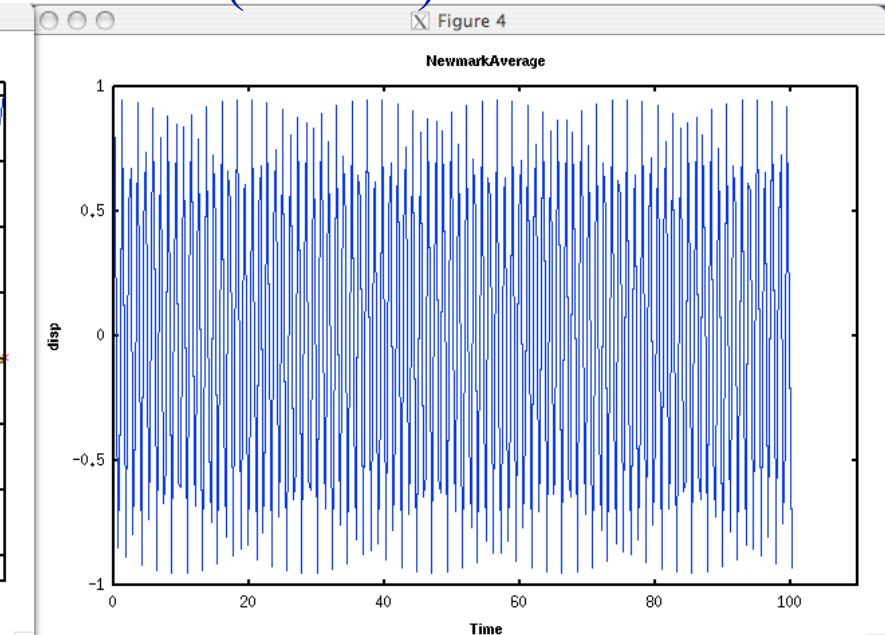
M is chosen to give desired period.

exact soln: $u(t) = u(0) \cos \omega_n t$.

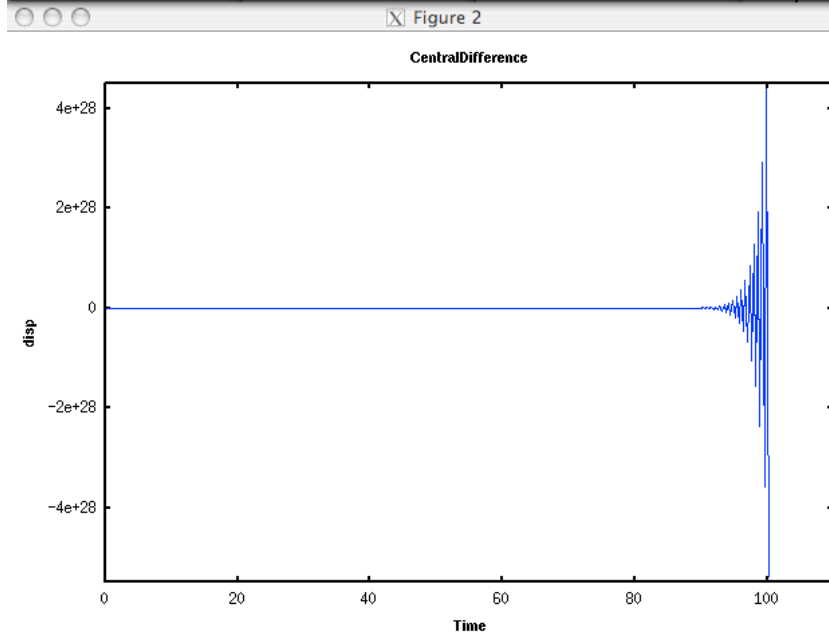
$$dT/T_n = 0.32 \quad (\text{ex1.tcl})$$



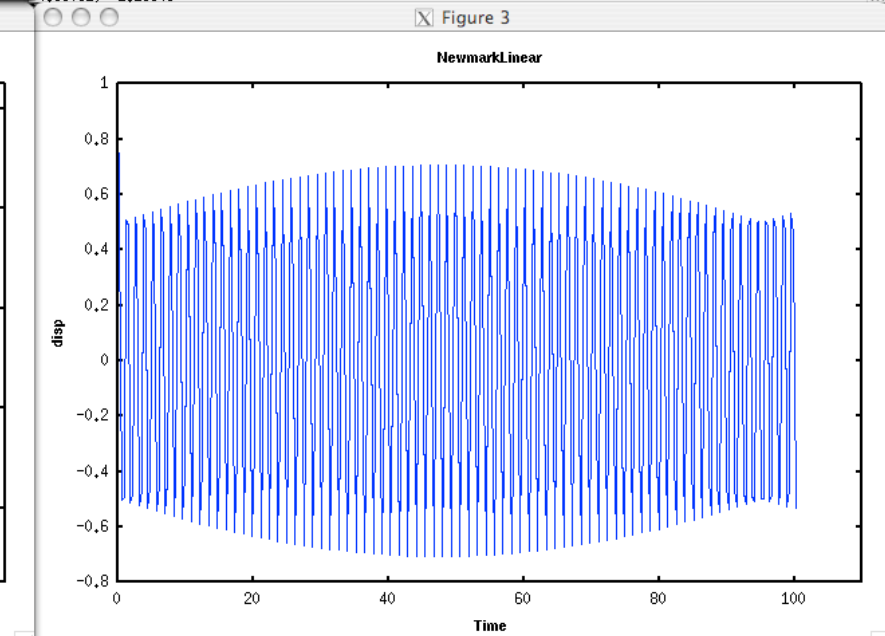
2,85543, -13,0933



4,98031, -1,15348

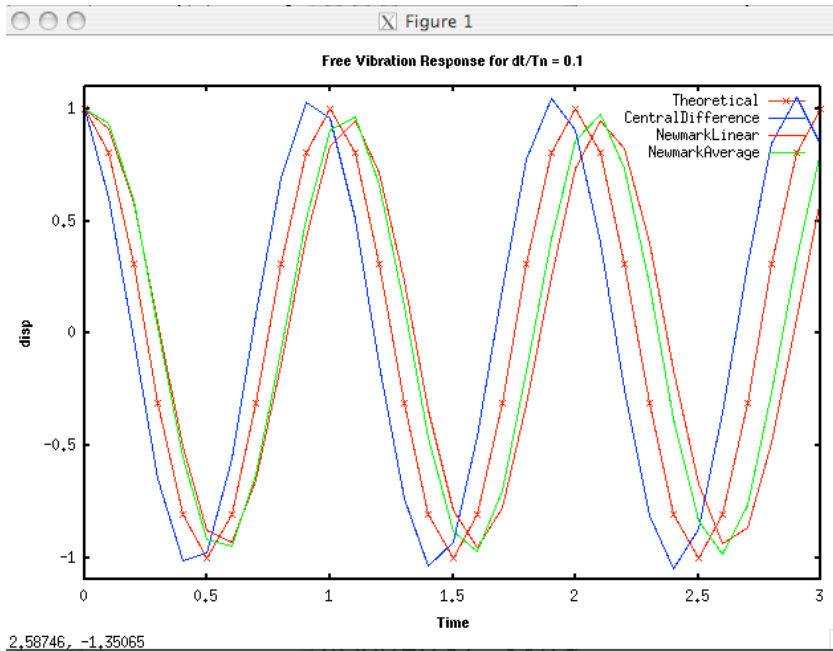


70,2672, 5,39138e+28

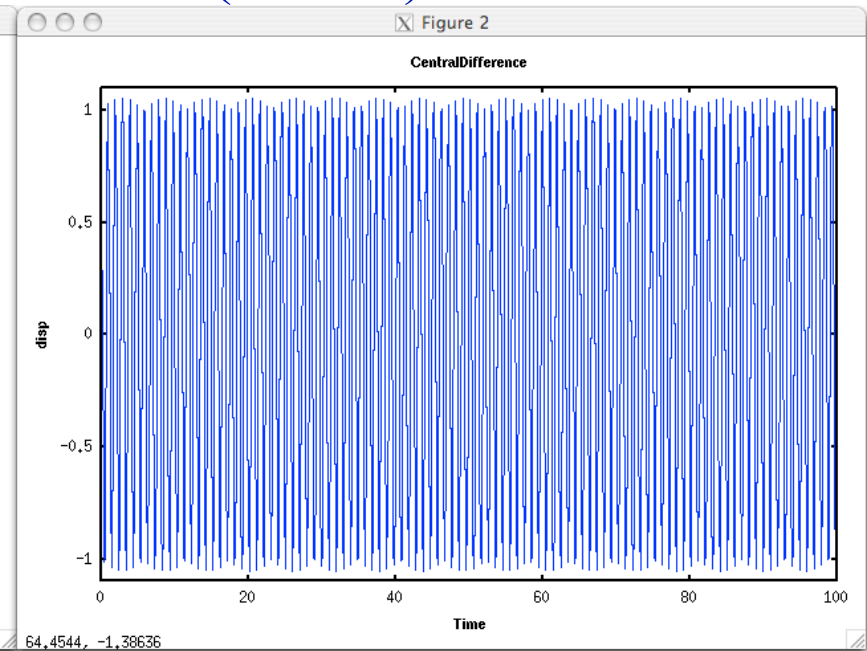


113,062, -0,947698

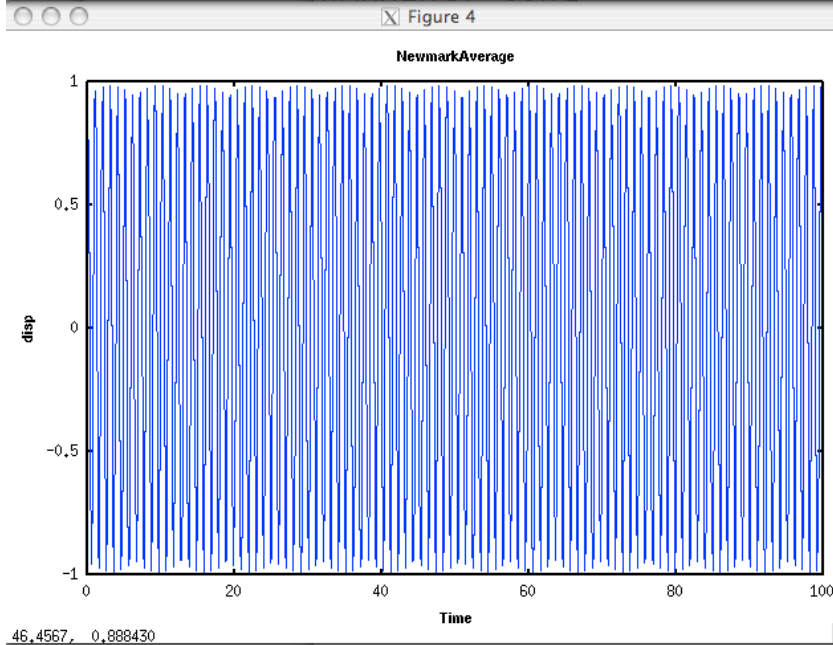
$$dT/T_n = 0.1 \quad (\text{ex1.tcl})$$



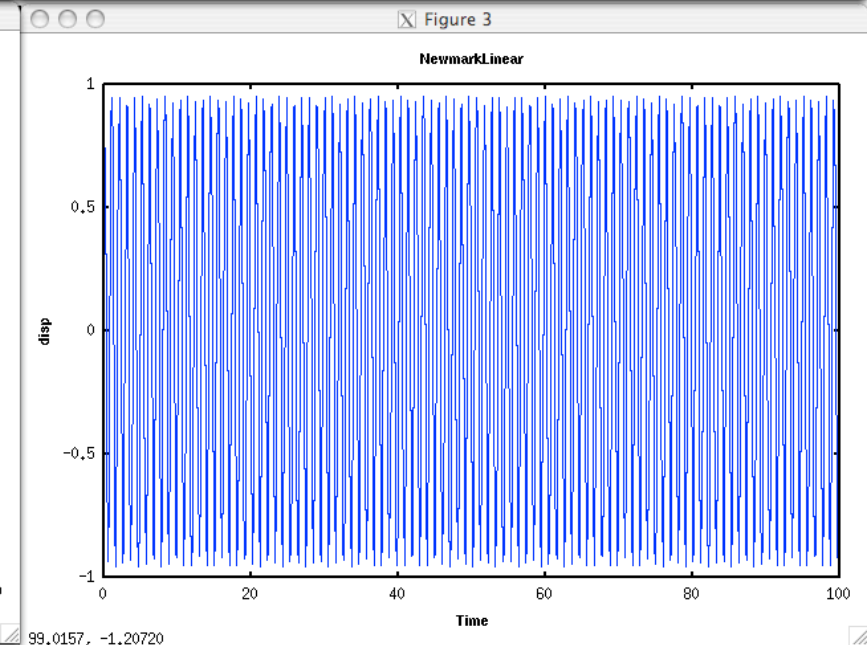
2,58746, -1,35065



64,4544, -1,38636

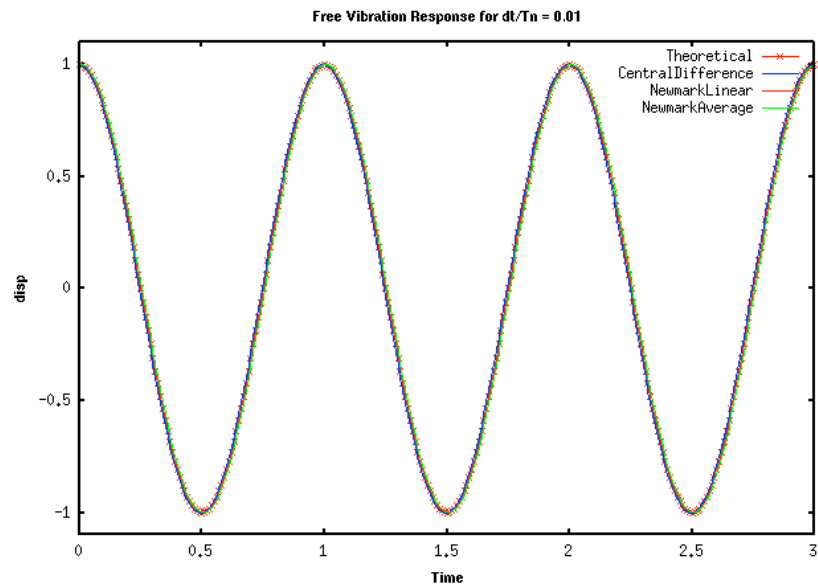


46,4567, 0,888430



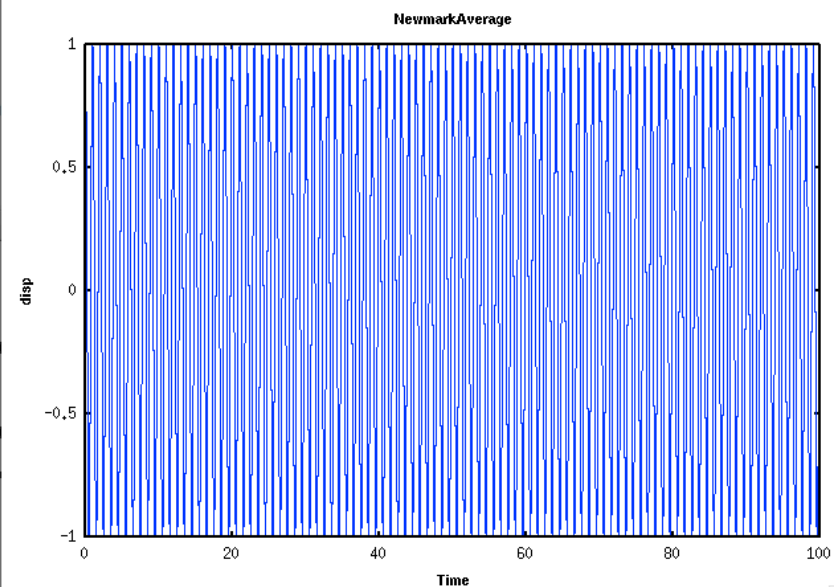
99,0157, -1,20720

$$dT/T_n = 0.01 \quad (\text{ex1.tcl})$$



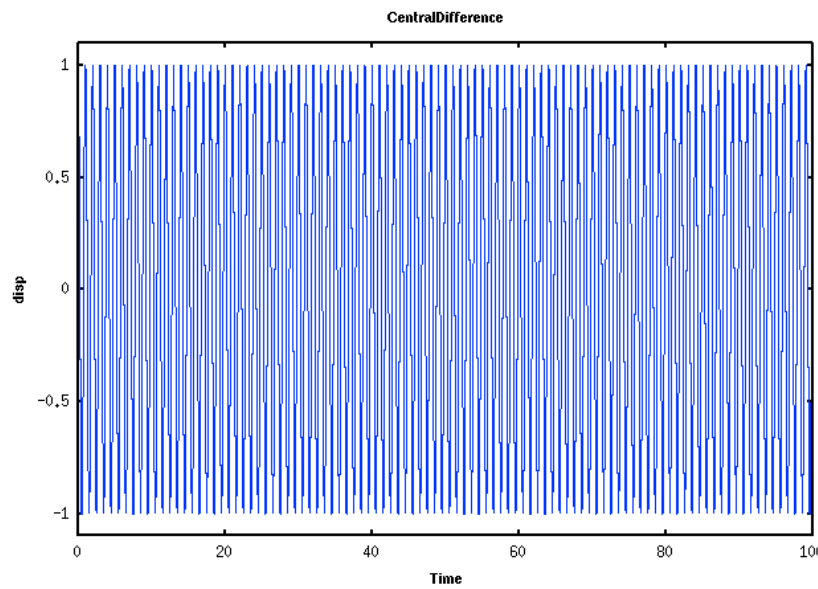
3,02447, -1,11494

Figure 2

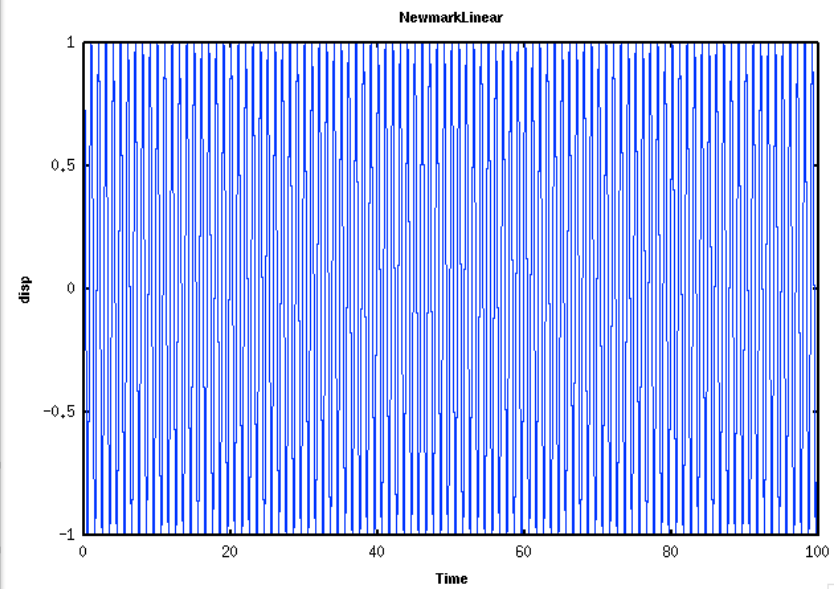


0,0281215, 0,0342385

Figure 3



3,02447, -1,11494



0,0281215, 0,0342385

Stability & Nonlinear Systems

- Algorithms that are stable for linear dynamical systems **ARE NOT NECESSARY STABLE** in nonlinear case.

“.. even with Newton–Raphson iterations carried out to very tight convergence tolerances, a scheme that is unconditionally stable in linear analysis may become unstable in a nonlinear solution.”

Source: “On a composite implicit time integration procedure for nonlinear dynamic systems” ,Klaus-Jurgen Bathe, Mirza M. Irfan Baig

- A sufficient condition in non-linear systems for stability is the conservation of total energy within a step, expressed:

$$U_{n+1} - U_n + K_{n+1} - K_n \leq W_{\text{ext}}$$

where U = strain energy and K = kinetic energy

- There are 3 groups of algorithms which **ATTEMPT** to satisfy this criterion:

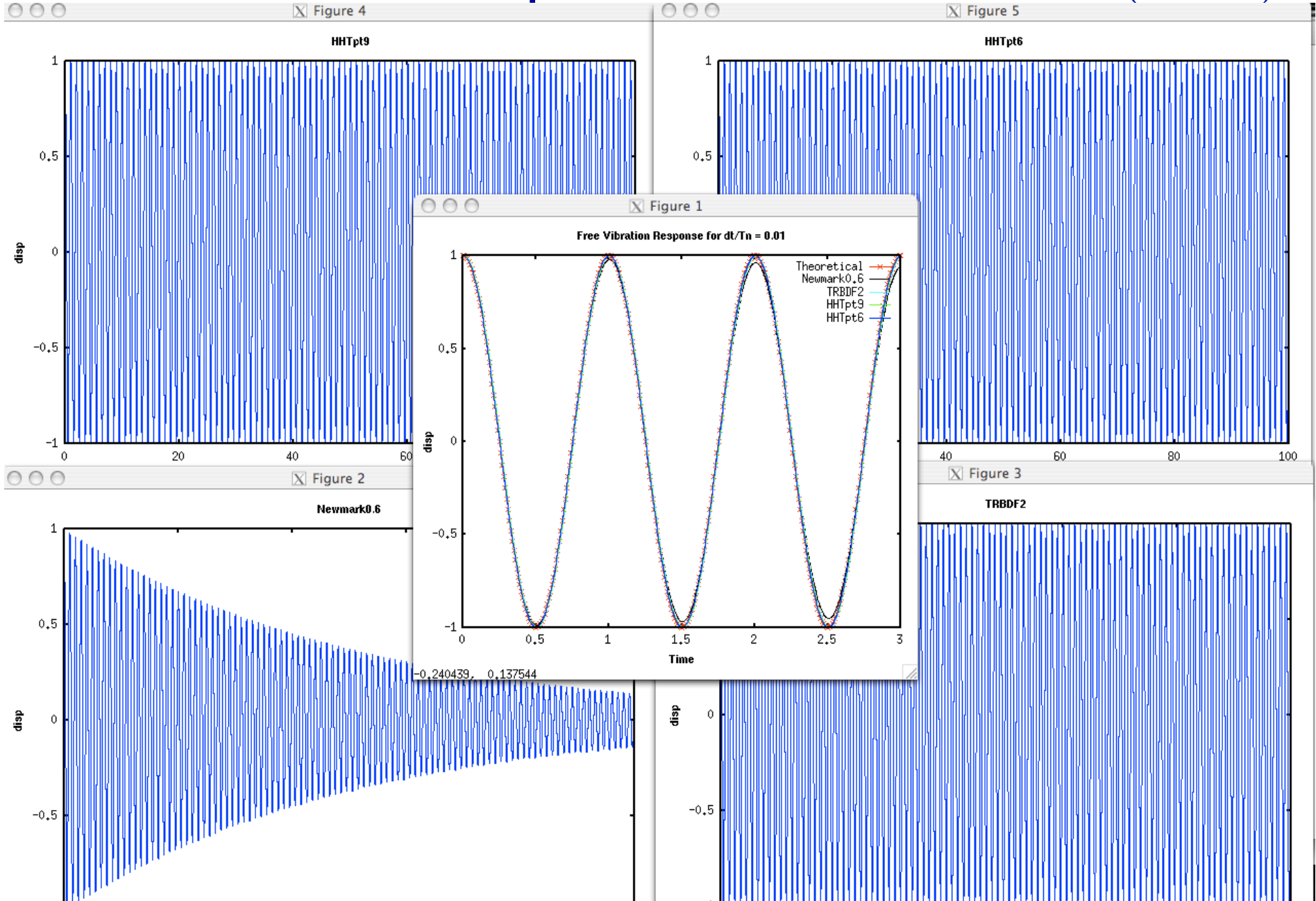
1. Numerical Dissipation
2. Enforced Conservation of Energy
3. Algorithmic Conservation of Energy

**Neither Types
Are Available
in OpenSees**

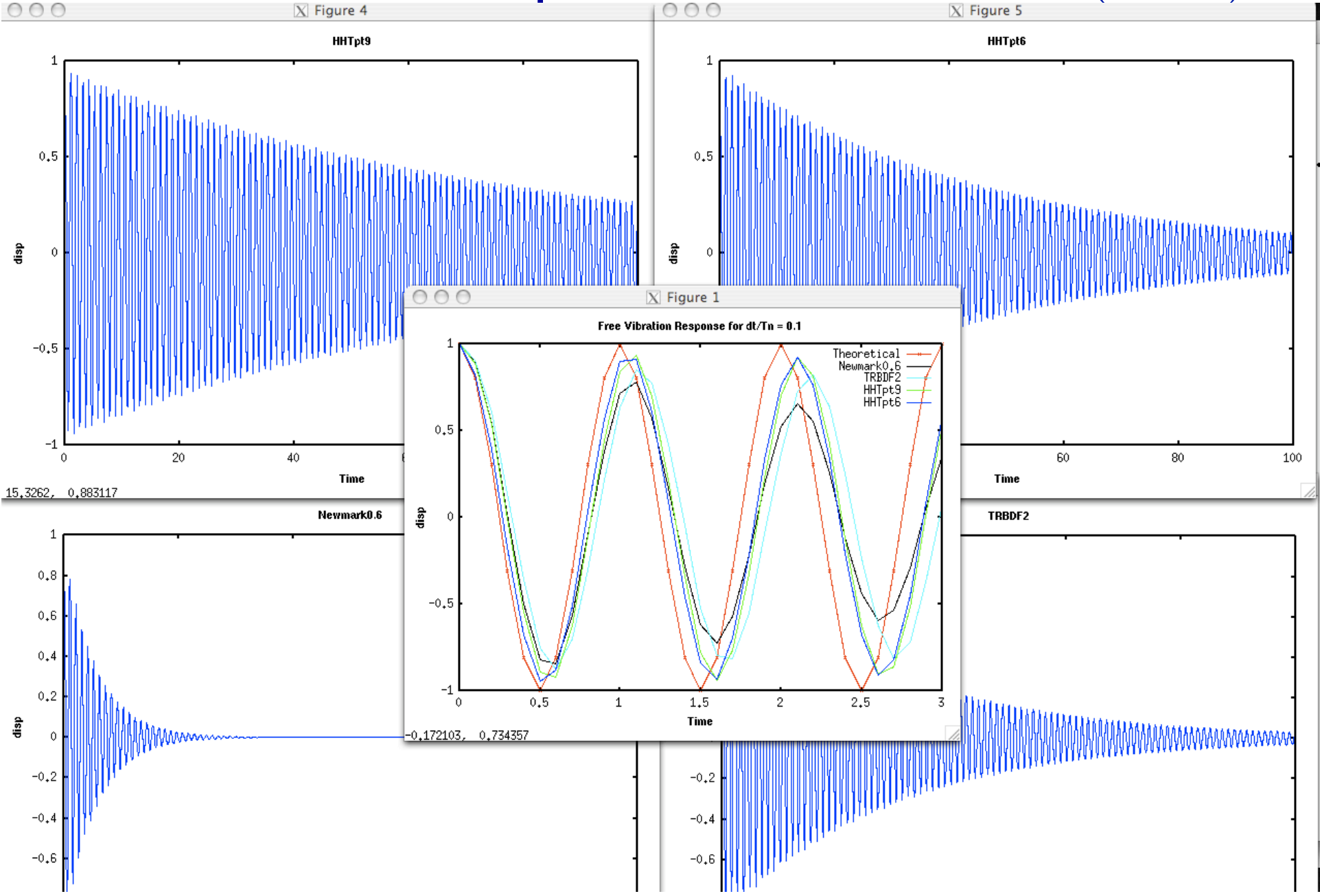
Dissipation Algorithms

- They were developed for large linear systems where typically only the low modes of response are of interest and the engineer wants to remove the high frequency noise (sometimes you don't want to do this!)
- These controlled dissipation of high frequency modes is used in an **ATTEMPT** to conserve energy.
- For nonlinear systems they do not guarantee the dissipation of enough energy to always satisfy the conservation of energy.
- EXAMPLES: Newmark ($\gamma > 0.5$), HHT, TRBDF2, TRBDF3

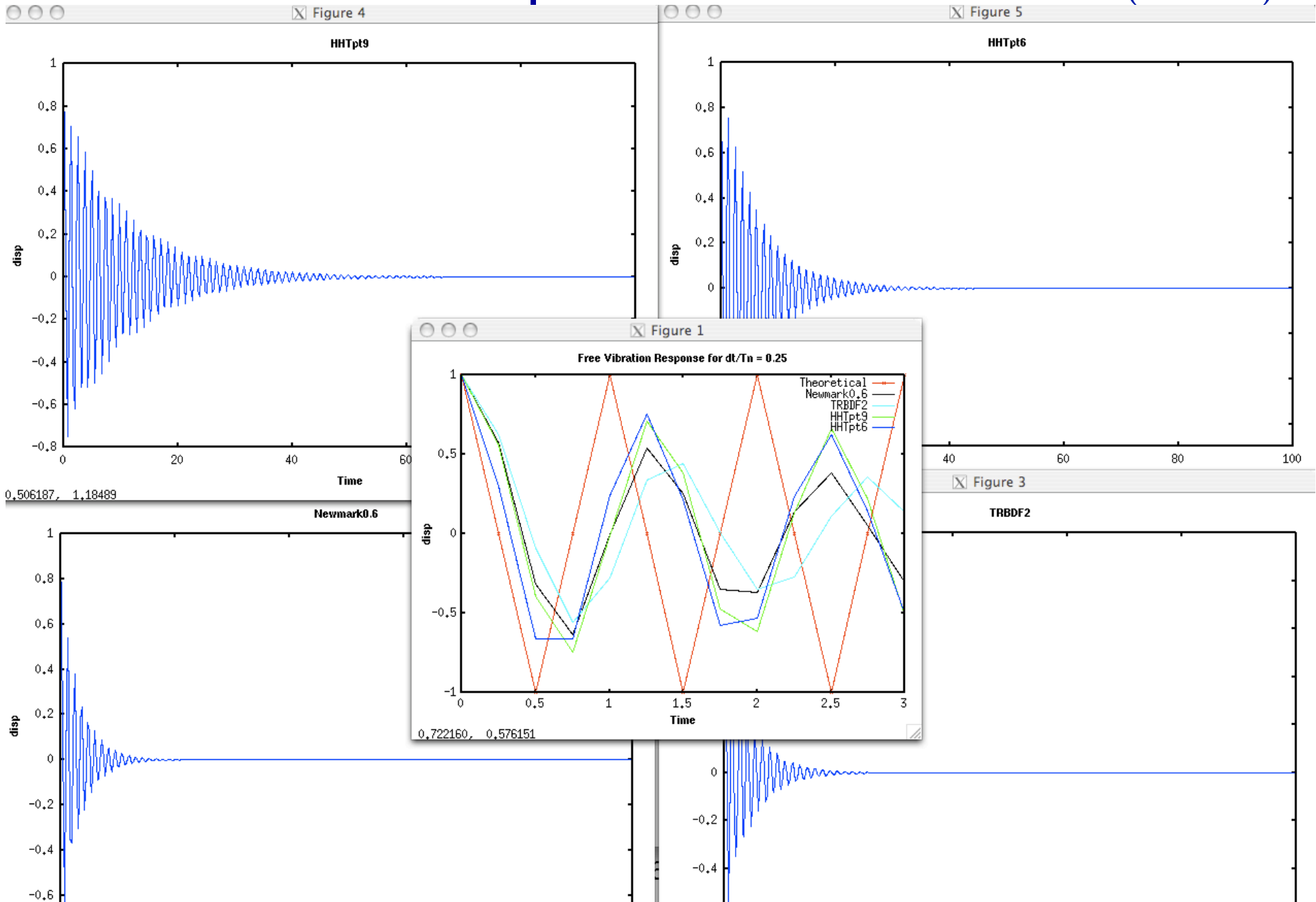
Numerical Dissipation: $dT/T_n = 0.01$ (ex2.tcl)



Numerical Dissipation: $dT/T_n = 0.1$ (ex2.tcl)



Numerical Dissipation: $dT/T_n = 0.25$ (ex2.tcl)



Example

(see "Dynamics of Structures" A.K. Chopra, section 3.1)

Harmonic vibration (ex 3.1d)



$$u(0) = 0$$

$$u'(0) = 0$$

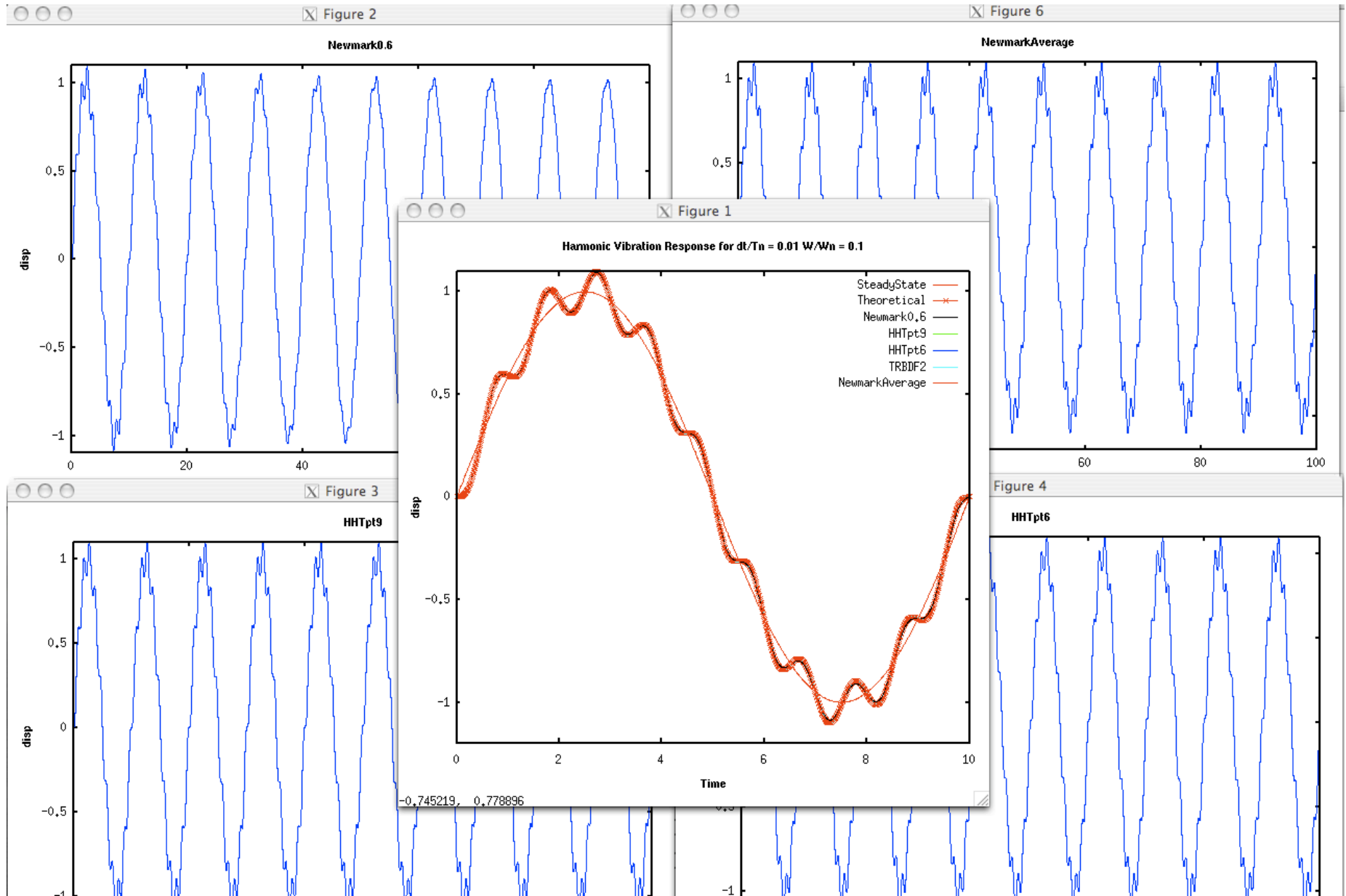
$$P(t) = \sin \omega t$$

M again is chosen to give desired period
 P is chosen such that $P/k = 1$.

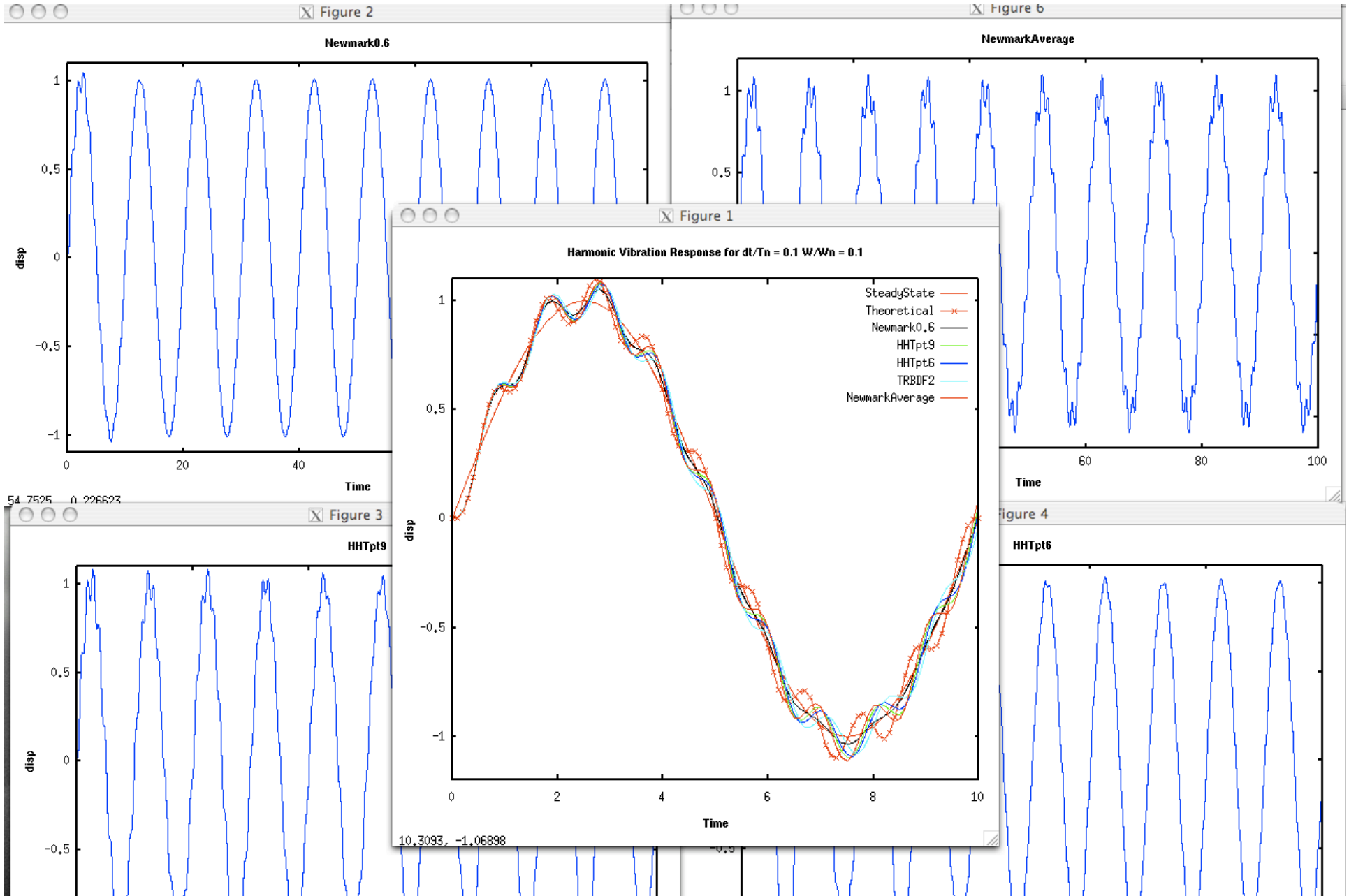
exact soln:

$$u(t) = \frac{P_0}{k} \frac{1}{1 - (\omega/\omega_n)^2} \left(\sin \omega t - \frac{\omega}{\omega_n} \sin \omega_n t \right)$$

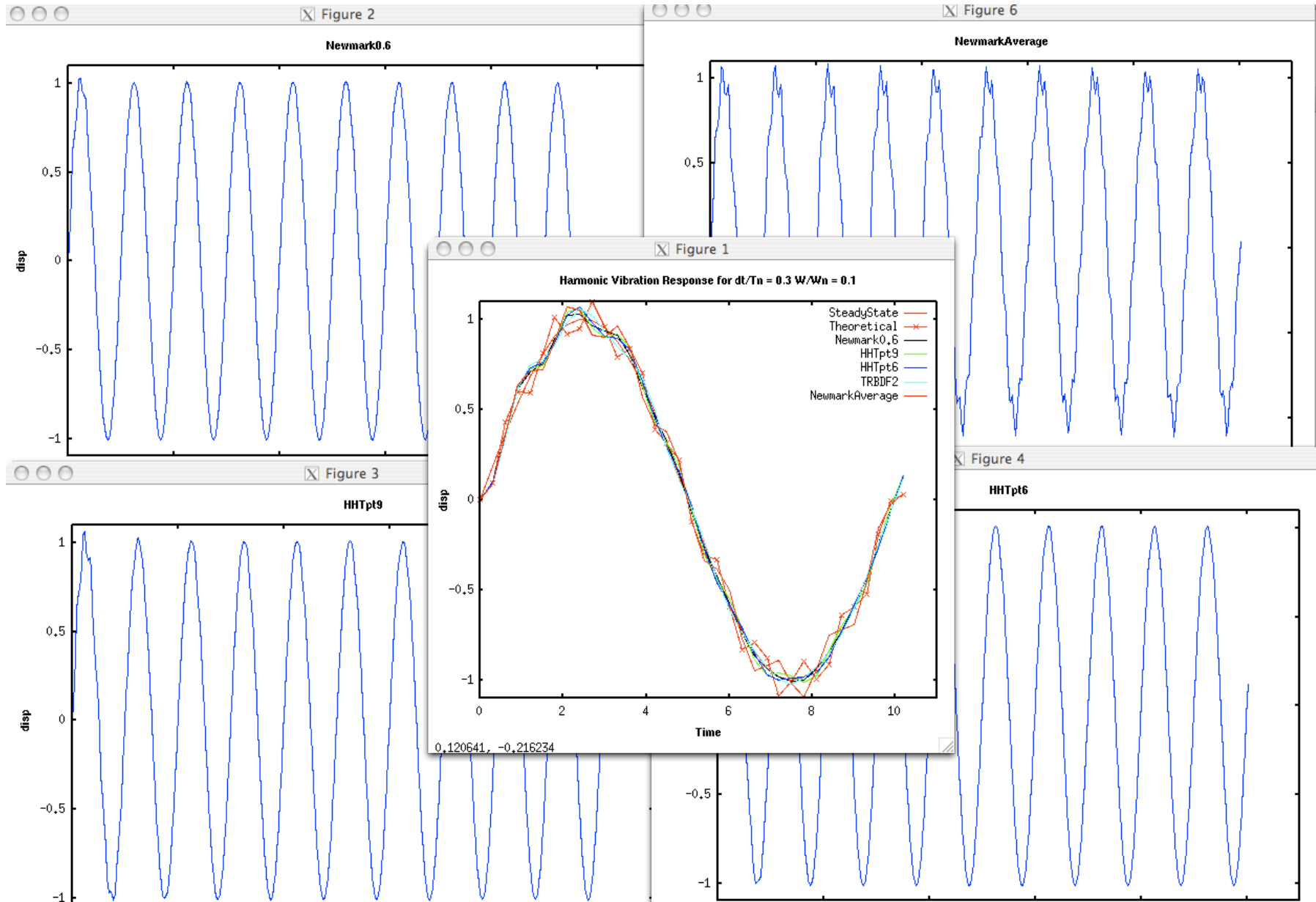
$$T/T_n = 0.1; dT/T_n = 0.01 \quad (\text{ex3.tcl})$$



$$T/T_n = 0.1; dT/T_n = 0.1 \quad (\text{ex3.tcl})$$



$$T/T_n = 0.1; dT/T_n = 0.3 \quad (\text{ex3.tcl})$$



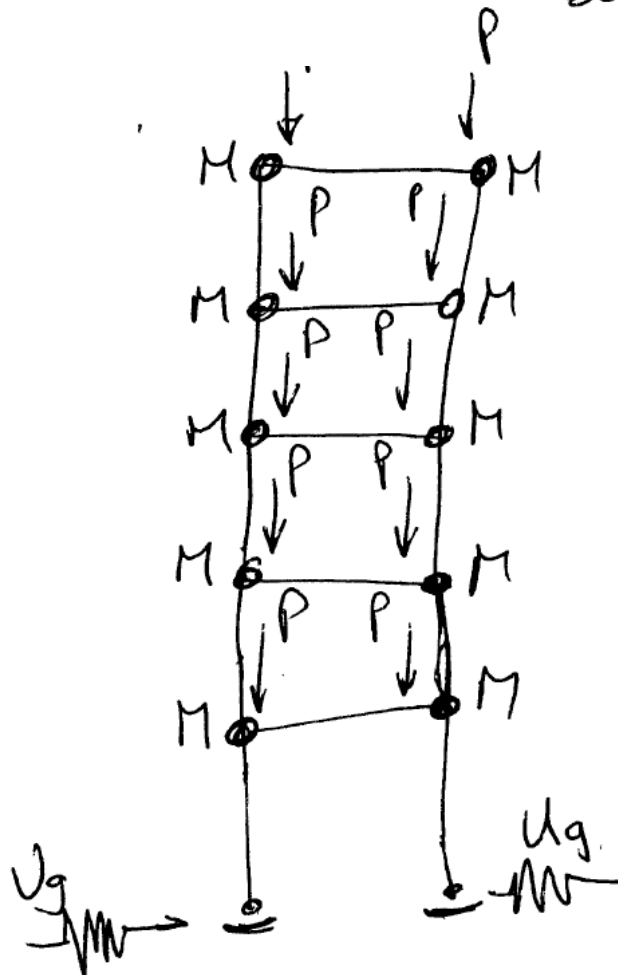
Remember

- Rayleigh Damping will can also be used to provide numerical damping
- When using dissipation to damp out higher frequencies, the choice of dT is as important as choice of integrator parameters.
- Why damp out higher frequencies?
 1. Not interested in spurious modes
 2. Contact
 3. (I know I am repeating but again) In nonlinear problems try to remove energy and hopefully allow conservation of energy (not guaranteed)

EXAMPLE

(Andreas Schellenberg, Rutherford and Chекenne)

- Building on friction pendulum bearings subjected to earthquake.



RESULTS

Periods Start: 0.50 sec to 0.0015 sec

Periods Just Before Impact: 380590.94 sec to 0.0045 sec

Periods if successful Analysis: 1.29 sec to 0.0015

Eigenvalues at end of transient:

n	lambda	omega	period
0	2.725479e-10	1.6509024804633374e-5	380590.9423199949
1	4.841790e-10	2.2004067805749007e-5	285546.5345156761
2	1.900059e-09	4.358966620656781e-5	144143.9188225047
3	2.437622e+03	49.37227967189686	0.12726139746704931
4	5.094668e+03	71.37694305586363	0.08802822085364471
5	9.174452e+03	95.78335972391028	0.06559787968693609
			0.05515718416484677
			0.04385979937343223
			0.016858990250615526
			0.016472394298701693
			0.008726856690187374
			0.00867526394323706
			3 0.006170819544074681
			3 0.006170819544074681
			7 0.006170057818824233
			4 0.006166930562842862
			0.00615859764023062
17	1.042444e+06	1021.0014691468372	0.006153943453607273
18	1.046875e+06	1023.1690964840562	0.006140906062126648
19	1.055707e+06	1027.476033783757	0.006115164831671342
20	1.555125e+06	1247.0465107605248	0.005038453059258967
21	1.560327e+06	1249.130497586221	0.0050300471562587
22	1.934602e+06	1390.8997088215958	0.00451735324073283
23	1.939489e+06	1392.6553773277867	0.00451165838258974

**Just before
point of
failure**

Eigenvalues at end of transient:

n	lambda	omega	period
1	2.365794e+01	4.863942845058935	1.291788474357259
2	2.444645e+03	49.443351423624186	0.12707846709957166
3	3.354993e+03	57.92230140455401	0.10847609909860358
4	7.038561e+03	83.89613221120506	0.07489243117146242
5	1.297724e+04	113.91768958331274	0.05515548401799733
6	2.045299e+04	143.01395036848677	0.0439340728019222
			775 0.029451923488762022
			442 0.016661966331203787
			365 0.010499457351349708
			4962 0.008700707004971624
			5294 0.006672111636214162
			19744 0.006170795735890136
			32478 0.006170010220361572
			38169 0.0061668889774234605
			39512 0.0061622426545781695
			34738 0.006158571014872712
17	1.040000e+06	1020.1713390700259	0.0061408913973288815
18	1.055708e+06	1027.4765204129972	0.006115161935431908
19	1.465409e+06	1210.5407882430068	0.005190395373871768
20	1.557795e+06	1248.1165810932887	0.005034133351289849
21	1.909504e+06	1381.8480379549699	0.00454694375546404
22	1.937262e+06	1391.855595958144	0.004514250850034686
23	1.823510e+07	4270.257603470779	0.001471383202285675

**At end of
successful
analysis**

For a dT=0.001

Newmark Average Acceleration and HHT 0.9 failed

HHT 0.6, TRBDF2, and Newmark 0.6 0.3025 worked

Max recorded roof displacements: 6.03, 5.88, 5.87 respectively

analysis command:

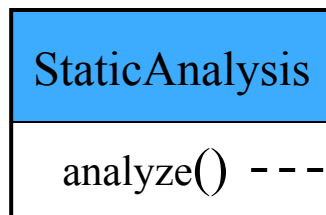
- Static Analysis

analysis Static

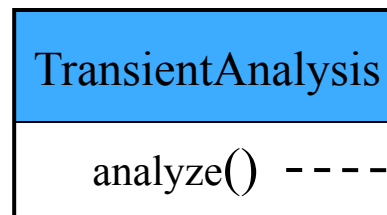
- Transient Analysis

analysis Transient

- both incremental solution strategies



```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep();  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```



```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep(dt);  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```

- Eigenvalue

- general eigenvalue problem

$$(\mathbf{K}-\lambda\mathbf{M})\Phi=0$$

eigen numModes? -general

- standard eigenvalue problem

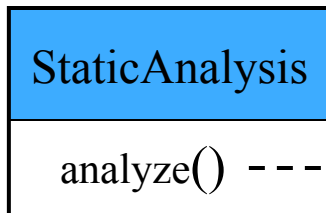
$$(\mathbf{K}-\lambda)\Phi=0$$

eigen numModes? -standard

analyze command:

- to perform the static/transient analysis

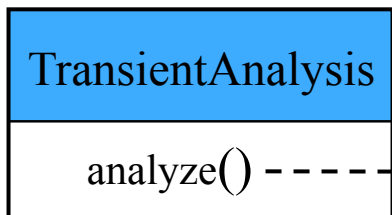
•Static Analysis



```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep();  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```

analyze numIter?

•Transient Analysis



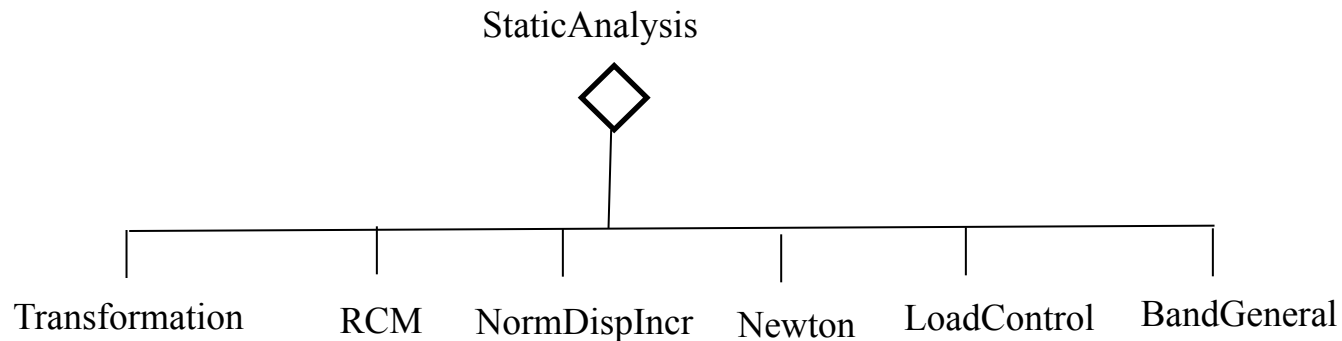
```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep(dt);  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```

analyze numIter? Δt ?

Example Static Analysis:

- Static Nonlinear Analysis with LoadControl

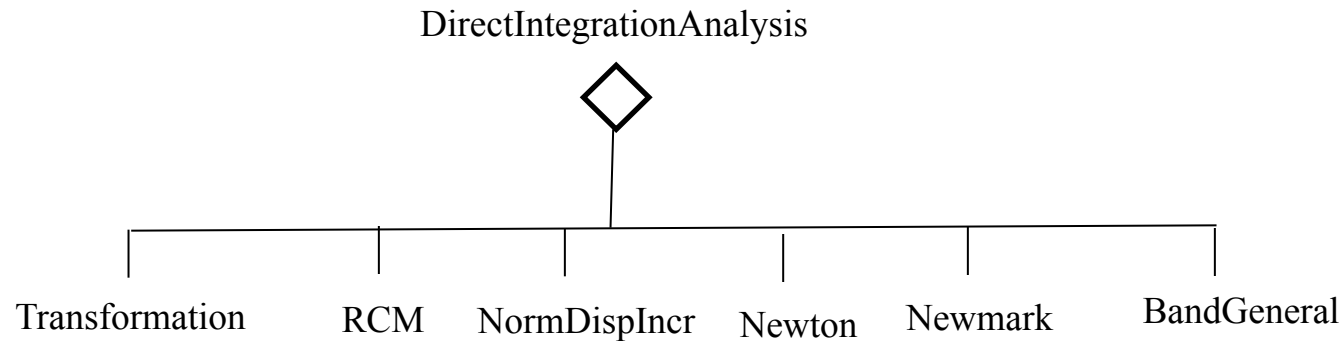
```
constraints Transformation  
numberer RCM  
system BandGeneral  
test NormDispIncr 1.0e-6 6 2  
algorithm Newton  
integrator LoadControl 0.1  
analysis Static  
analyze 10
```



Example Dynamic Analysis:

- Transient Nonlinear Analysis with Newmark

constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
analyze 2000 0.01



Remember that nonlinear
analysis does not always
converge

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

CHECK YOUR MODEL

Commands that Return Values

- analyze command

The analyze command returns 0 if successful.
It returns a negative number if not

```
set ok [analyze numIter <Δt>]
```

- getTime command

The getTime command returns pseudo time in Domain.

```
set currentTime [getTime]
```

- nodeDisp command

The nodeDisp command returns a nodal displacement.

```
set disp [nodeDisp node dof]
```


Example Usage – Displacement Control

```
set maxU 15.0; set dU 0.1
constraints transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator DispControl 3 1 $dU
analysis Static
set ok 0
set currentDisp 0.0
while {$ok == 0 && $currentDisp < $maxU} {
    set ok [analyze 1]
    if {$ok != 0} {
        test NormDispIncr 1.0e-6 1000 1
        algorithm ModifiedNewton -initial
        set ok [analyze 1]
        test NormDispIncr 1.0e-6 6 2
        algorithm Newton
    }
    set currentDisp [nodeDisp 3 1]
}
```

Example Usage – Transient Analysis

```
set tFinal 15.0;
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
set ok 0
set currentTime 0.0
while {$ok == 0 && $currentTime < $tFinal} {
    set ok [analyze 1 0.01]
    if {$ok != 0} {
        test NormDispIncr 1.0e-6 1000 1
        algorithm ModifiedNewton -initial
        set ok [analyze 1 0.01]
        test NormDispIncr 1.0e-6 6 2
        algorithm Newton
    }
    set currentTime [getTime]
}
}
```

Still Not Working!

1. Search the Message Board
2. Post Problem on the Message Board

To check which scale of elcentro earthquake makes the SDF inelastic, the following file was used. By trial and error, scale 10 was found in OpenSees, which is inconsistent with the results(scale 4) from using other programs.

Is there anything wrong in this file?

```
# create ModelBuilder (with two-dimensions and 2 DOF/node)
model BasicBuilder -ndm 1 -ndf 1

# Define geometry for model
# -----
puts "Define geometry for model"
set k1 2.75
set uy 1.35
```

i suggest you check your other input files .. if you have a look at chopra's book he plots the response spectrum for this e.q. .. for a period of 0.1, D for an elastic system is with 0% damping is about .11 (fig 6.8.1 in my version) .. so you need a scale factor of about 12 [1.35/.11] to reach the ultimate.
(note using Newmark 0.5 0.25 you get .11)

to compute the scale factor for yield i suggest you also stop playing with trying to predict the scale factor & just divide yield disp by the max response from elastic system.

Segmentation Faults, etc:

- Email: fmckenna@ce.berkeley.edu

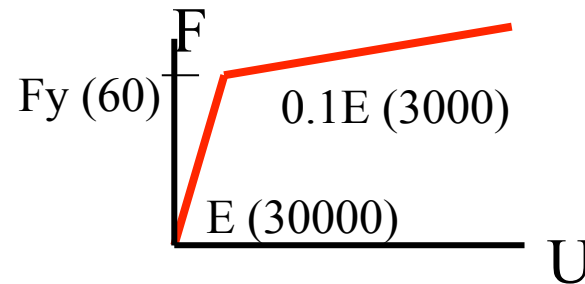
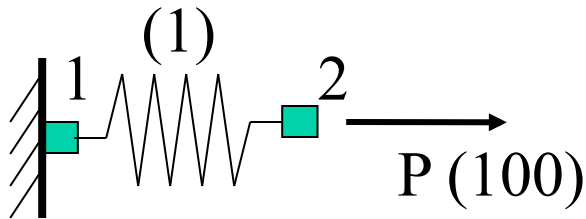
NOTE: Zip up your files in **1** directory and send them to us

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
 - Parallel & Distributed Processing
 - OpenSees on NEEShub
 - Hands on Exercise
 - Adding Your Code to OpenSees
 - Advanced Model Development
 - Conclusion

Spring Example - Load Control

a.tcl



```
# create the model builder
model Basic -ndm 1 -ndf 1
# create 2 nodes
node 1 0.0
node 2 0.0
# fix node 1
fix 1 1
# create material
set Fy 60.0
set E 30000.0
set b 0.1
uniaxialMaterial Steel01 1 $Fy $E $b
# create element
element zeroLength 1 1 2 -mat 1 -dir 1
# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

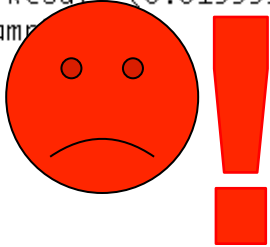
```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-6 6 0
algorithm Newton
system ProfileSPD
integrator LoadControl 0.1
analysis Static
# perform the analysis
analyze 10
# Output
print node 2
set exact [expr $Fy/$E + (100-$Fy)/($b*$E)]
set res [lindex [nodeDisp 2] 0]
if {$exact == $res} {
  puts "Exact ($exact) EQUALS Result ($res)"
} else {
  puts "Exact ($exact) NOT EQUAL Result ($res)"
}
```

```
Terminal — bash — 80x19
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

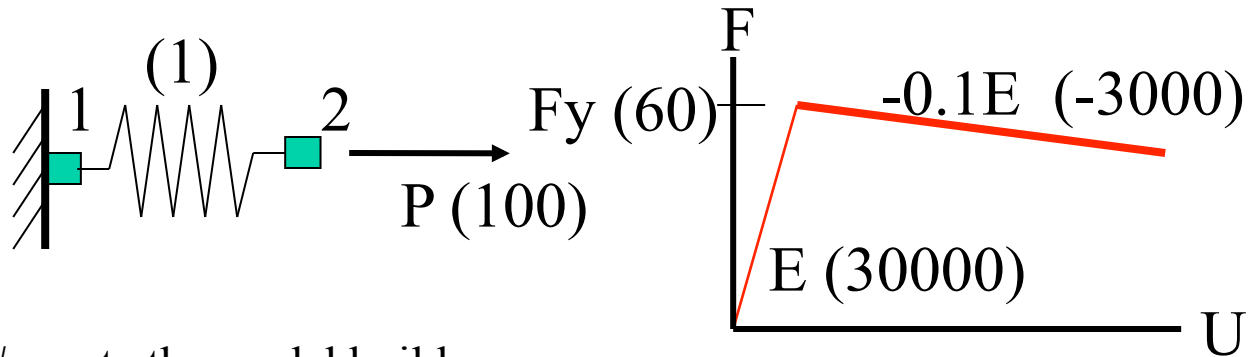
Node: 2
Coordinates : 0
Disps: 0.0153333
Velocities : 0
unbalanced Load: 100
ID : 0

Exact (0.015333333333333334) IS NOT EQUAL TO Result (0.0153333333333333250)
fmk:~/Desktop/Workshops/ChinaWorkshop2011/exam
```



Computers cannot represent all numbers exactly
and
Computer math involves roundoff

b.tcl



```
# create the model builder
model Basic -ndm 1 -ndf 1
# create 2 nodes
node 1 0.0
node 2 0.0
# fix node 1
fix 1 1
# create material
set Fy 60.0
set E 30000.0
set b -0.1
uniaxialMaterial Steel01 1 $Fy $E $b
# create element
element zeroLength 1 1 2 -mat 1 -dir 1
# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-6 6 0
algorithm Newton
system ProfileSPD
integrator LoadControl 0.1
analysis Static
# perform the analysis
analyze 10
# Output
print node 2
```


Another common situation for which the solve fails message occurs: **NOT ENOUGH BOUNDARY CONDITIONS!**

```
Terminal — bash — 87x20

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

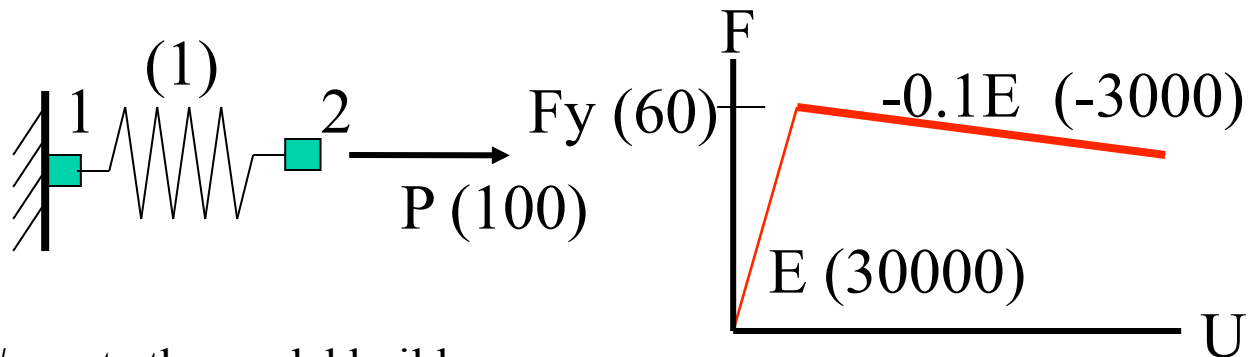
ProfileSPDlinDirectSolver::solve() - aii < 0 (i, aii): (0,0)
WARNING NewtonRaphson::solveCurrentStep() -the LinearSysOfEqn failed in solve()
StaticAnalysis::analyze() - the Algorithm failed at iteration: 5 with domain at load fa
ctor 0.6
OpenSees > analyze failed, returned: -3 error flag

Node. 2
Coordinates : 0
Disps: 0.00166667
Velocities : 0
unbalanced Load: 50
ID : 0

fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples$
```

change **system ProfileSPD** to **system BandGen**

change **LoadControl 0.1** to **LoadControl 0.0999999**



```
# create the model builder
model Basic -ndm 1 -ndf 1
# create 2 nodes
node 1 0.0
node 2 0.0
# fix node 1
fix 1 1
# create material
set Fy 60.0
set E 30000.0
set b -0.1
uniaxialMaterial Steel01 1 $Fy $E $b
# create element
element zeroLength 1 1 2 -mat 1 -dir 1
# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-6 6 0
algorithm Newton
system BandGen
integrator LoadControl 0.09999
analysis Static
# perform the analysis
analyze 10
# Output
print node 2
```

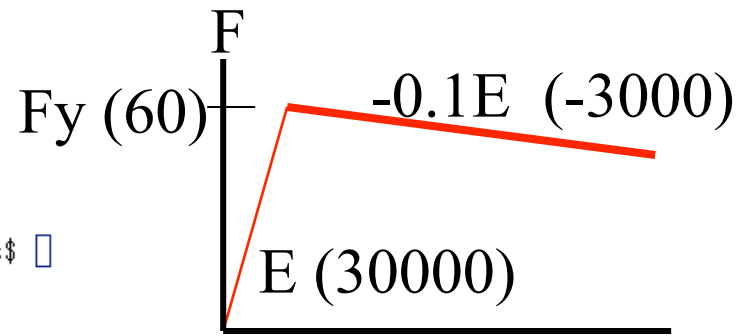
```
Terminal — bash — 104x22
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

WARNING: CTestNormDispIncr::test() - failed to converge
after: 6 iterations
NewtnRaphson::solveCurrentStep() -the ConvergenceTest object failed in test()
StaticAnalysis::analyze() - the Algorithm failed at iteration: 6 with domain at load factor 0.7
OpenSees > analyze failed, returned: -3 error flag

Node: 2
Coordinates : 0
Disps: 0.002
Velocities : 0
unbalanced Load: 60
ID : 0

fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples$
```

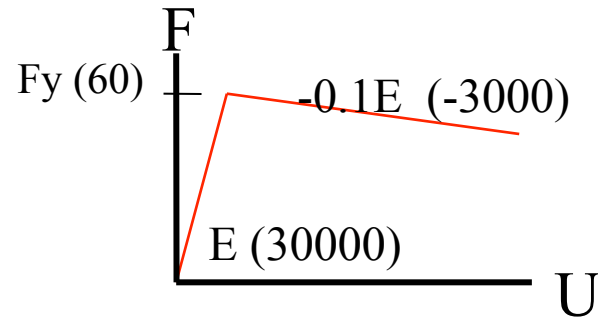
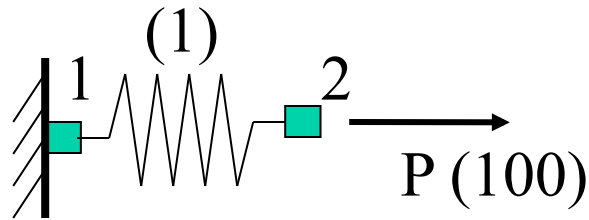


With a Yield Strength of 60
This is as far as we can push the
Model using LoadControl

We can go further using a Displacement Control scheme

Spring Example - Displacement Control

d.tcl



```
# create the model builder
model Basic -ndm 1 -ndf 1
# create 2 nodes
node 1 0.0
node 2 0.0
# fix node 1
fix 1 1
# create material
set Fy 60.0
set E 30000.0
set b -0.1
uniaxialMaterial Steel01 1 $Fy $E $b
# create element
element zeroLength 1 1 2 -mat 1 -dir 1
# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-12 6 0
algorithm Newton
system BandGen
integrator DisplacementControl 2 1 0.001
analysis Static
# perform the analysis & print results
for {set i 0} {$i < 10} {incr i 1} {
  analyze 1
  set factor [getTime]
  puts "[expr $factor*$P] [lindex [nodeDisp 2] 0]"
}
print node 2
```

```
fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples$ OpenSees d.tcl
```

```
OpenSees -- Open System For Earthquake Engineering Simulation  
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha
```

```
(c) Copyright 1999,2000 The Regents of the University of California  
All Rights Reserved
```

```
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)
```

```
30.0 0.00100000000000000002  
60.0 0.00200000000000000004  
56.99999999999999 0.00300000000000000006  
54.0 0.00400000000000000008  
51.0 0.00500000000000000010  
48.0 0.00600000000000000012  
45.0 0.00700000000000000015  
42.0 0.00800000000000000017  
39.0 0.009000000000000000105  
36.0 0.010000000000000000194
```

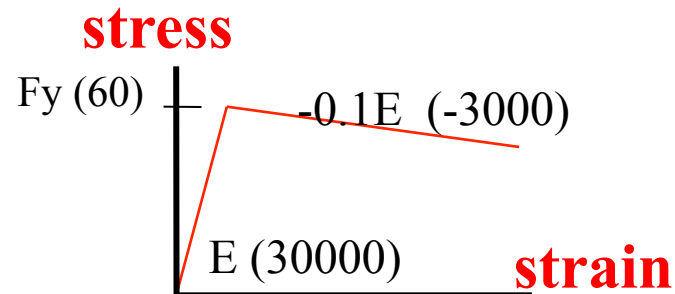
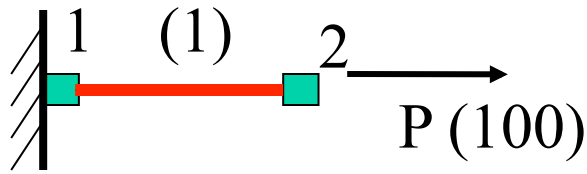
```
Node: 2
```

```
Coordinates : 0  
Disps: 0.01  
Velocities : 0  
unbalanced Load: 36  
ID : 0
```

```
fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples$ █
```

Truss Example - Displacement Control

e.tcl



```
# create the model builder
model Basic -ndm 1 -ndf 1

# create 2 nodes
node 1 0.0
node 2 1.0

# fix node 1
fix 1 1

# create material
set Fy 60.0
set E 30000.0
set b -0.1
uniaxialMaterial Steel01 1 $Fy $E $b

# create element
set A 1.0
element Truss 1 1 2 $A 1

# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-12 6 0
algorithm Newton
system BandGen
integrator DisplacementControl 2 1 0.001
analysis Static

# perform the analysis & print results
for {set i 0} {$i < 10} {incr i 1} {
  analyze 1
  set factor [getTime]
  puts "[expr $factor*$P] [lindex [nodeDisp 2] 0]"
}

print node 2
```

Terminal — bash — 92x30

fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples\$ openSees e.tcl

OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved

(Copyright and Disclaimer @ <http://www.berkeley.edu/OpenSees/copyright.html>)

```
30.0 0.00100000000000000002
60.0 0.00200000000000000004
56.999999999999999 0.00300000000000000006
54.0 0.00400000000000000008
51.0 0.00500000000000000010
48.0 0.00600000000000000012
45.0 0.00700000000000000015
42.0 0.00800000000000000017
39.0 0.009000000000000000105
36.0 0.010000000000000000194
```

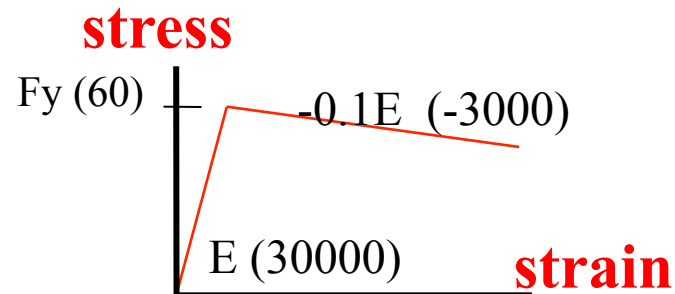
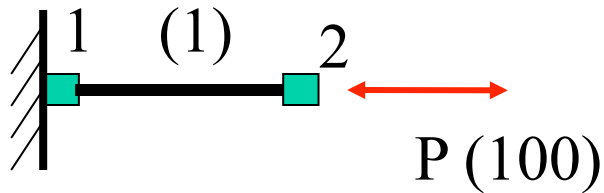
Node: 2

```
Coordinates : 1
Disps: 0.01
Velocities : 0
unbalanced Load: 36
ID : 0
```

fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples\$

Truss Example - Push & Pull

f.tcl



```
# create the model builder
model Basic -ndm 1 -ndf 1

# create 2 nodes
node 1 0.0
node 2 1.0

# fix node 1
fix 1 1

# create material
set Fy 60.0
set E 30000.0
set b -0.1
uniaxialMaterial Steel01 1 $Fy $E $b

# create element
set A 1.0
element Truss 1 1 2 $A 1

# create time series and load pattern
set P 100.0
timeSeries Linear 1
pattern Plain 1 1 {
  load 2 $P
}
```

```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-12 6 0
algorithm Newton
system BandGen
integrator DisplacementControl 2 1 0.001
analysis Static

# perform the analysis & print results
foreach numIter {10 20 10} dU {0.001 -0.001 0.001} {
  integrator DisplacementControl 2 1 $dU
  analyze $numIter
  set factor [getTime]
  puts "[expr $factor*$P] [lindex [nodeDisp 2] 0]"
}

print node 2
```


Terminal — bash — 93x23

fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples\$ OpenSees f.tcl

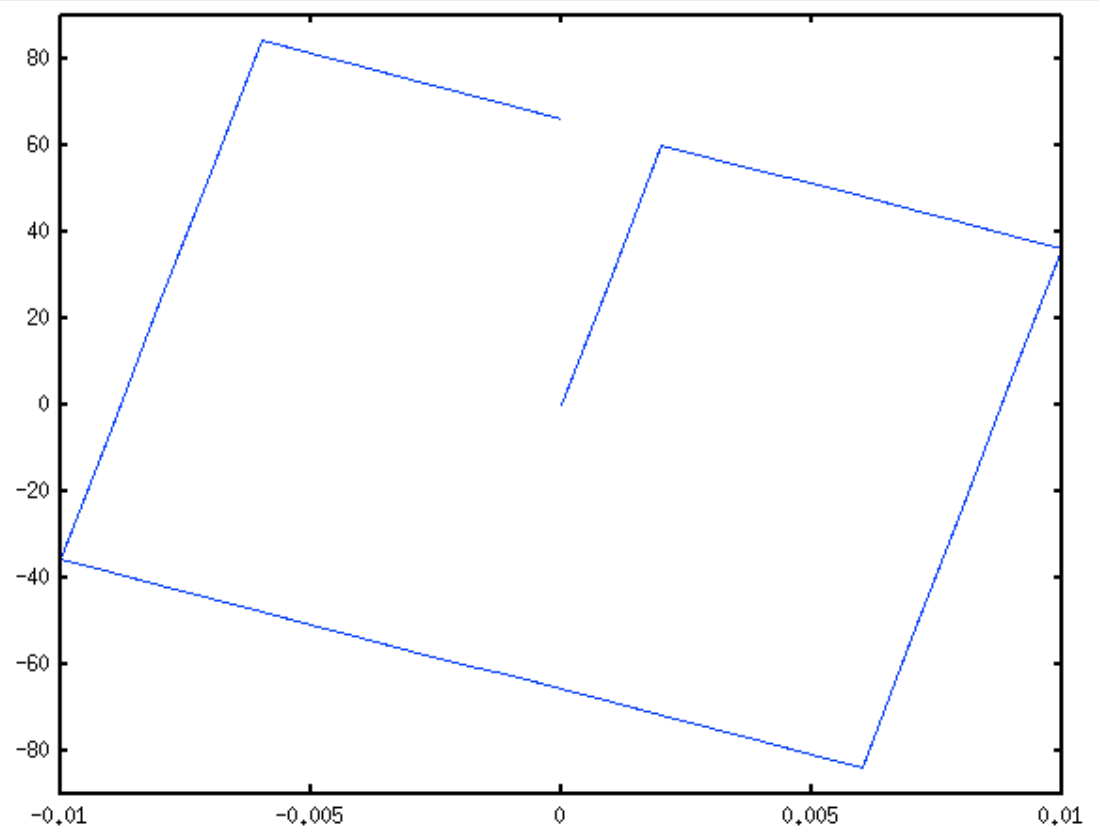
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ <http://www.berkeley.edu/OpenSees/copyright.html>)

```
36.0 0.010000000000000000194
-36.0 -0.010000000000000000194
66.0 0.000000000000000000000
```

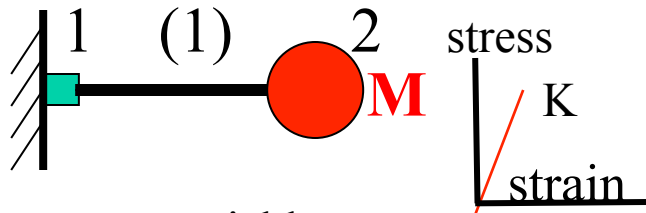
```
Node: 2
Coordinates : 1
Disps: 0
Velocities : 0
unbalanced Load: 66
ID : 0
```

fmk:~/Desktop/Workshops/ChinaWorkshop2011/



Truss Example - Uniform Excitation

j.tcl



```
# set some variables
set Tn 1.0
set K 4.0
set dampR 0.02
#set some constants
set g 386.4
set PI [expr 2.0 * asin(1.0)]
#derived quantities
set Wn [expr 2.0 * $PI / $Tn]
set M [expr $K / ($Wn * $Wn)]
set c [expr 2.0*$M*$Wn*$dampR]

# create the model
model basic -ndm 1 -ndf 1
node 1 0.0
node 2 1.0 -mass $M
fix 1 1
uniaxialMaterial Elastic 1 $K 0.0
uniaxialMaterial Elastic 2 0.0 $c
uniaxialMaterial Parallel 3 1 2

element truss 1 1 2 1.0 3
set dT 0.0;
set nPts 0;
```

```
# create the uniform excitation pattern
```

```
set record el_centro
```

```
source ReadRecord.tcl
```

```
ReadRecord $record.AT2 $record.dat dT nPts
```

```
timeSeries Path 1 -filePath $record.dat -dt $dT -factor $g
pattern UniformExcitation 1 1 -accel 1
```

```
# create the analysis
```

```
constraints Plain
```

```
integrator Newmark 0.5 [expr 1.0/6.0]
```

```
system ProfileSPD
```

```
test NormUnbalance 1.0e-12 6 0
```

```
algorithm Newton
```

```
numberer RCM
```

```
analysis Transient
```

```
# perform analysis
```

```
set t 0.0; set ok 0.0; set maxD 0.0;
```

```
set maxT [expr (1+$nPts)*$dT];
```

```
while {$ok == 0 && $t < $maxT} {
```

```
    set ok [analyze 1 $dT]
```

```
    set t [getTime]
```

```
    set d [nodeDisp 2 1]
```

```
    if {$d > $maxD} {
```

```
        set maxD $d
```

```
    } elseif {$d < [expr -$maxD]} {
```

```
        set maxD [expr -$d]
```

```
    }
```

```
}
```

```
puts "record: $record period: $Tn damping ratio: $dampR r
```

```
Terminal — bash — 81x13
examples> OpenSees j.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

record: el_centro period: 1.0 damping ratio: 0.02 max disp: 5.962305018001343
examples> █
```

Simply Supported Beam

n.tcl

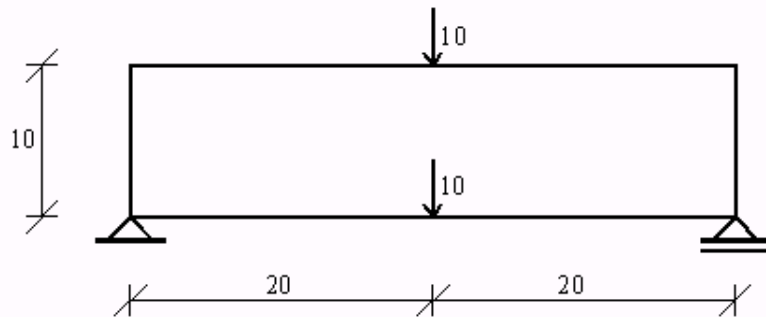


Fig. 1 Geometry and static loads

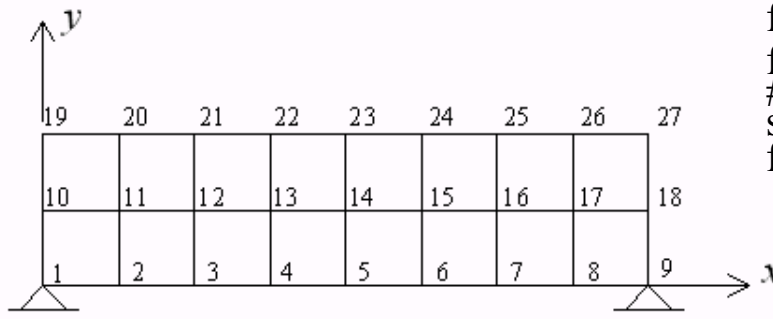


Fig. 2 Finite element mesh and node numbering

```
# some problem parameters
set L 40.0
set H 10.0
set thick 2.0
set P 10
set nX 9; # numNodes x dirn
set nY 3; # numNodes y dirn

# model builder
model Basic -ndm 2 -ndf 2

# create material
nDMaterial ElasticIsotropic 1 1000 0.25 3.0
```

```
# create nodes
set nodeTag 1
set yLoc 0.0;
for {set i 0} {$i < $nY} {incr i 1} {
  set xLoc 0.0;
  for {set j 0} {$j < $nX} {incr j 1} {
    node $nodeTag $xLoc $yLoc
    set xLoc [expr $xLoc+ $L/($nX-1.0)]
    incr nodeTag
  }
  set yLoc [expr $yLoc+ $H/($nY-1.0)]
}

# boundary conditions
fix 1 1 1
fix $nX 1 1
# create elements
set eleTag 1
for {set i 1} {$i < $nY} {incr i 1} {
  set iNode [expr 1+($i-1)*$nX];
  set jNode [expr $iNode+1];
  set kNode [expr $jNode+$nX]
  set lNode [expr $iNode+$nX]
  for {set j 1} {$j < $nX} {incr j 1} {
    element quad $eleTag $iNode $jNode $kNode $lNode \
      $thick "PlaneStress" 1
    incr eleTag; incr iNode; incr jNode; incr kNode; incr lNode
  }
}

# apply loads
set midNode [expr ($nX+1)/2]
timeSeries Linear 1
pattern Plain 1 1 {
  load $midNode 0 -$P
  load [expr $midNode + $nX*($nY-1)] 0 -$P
}
analysis Static;
analyze 1; print node $midNode
```

Simply Supported Beam

o.tcl

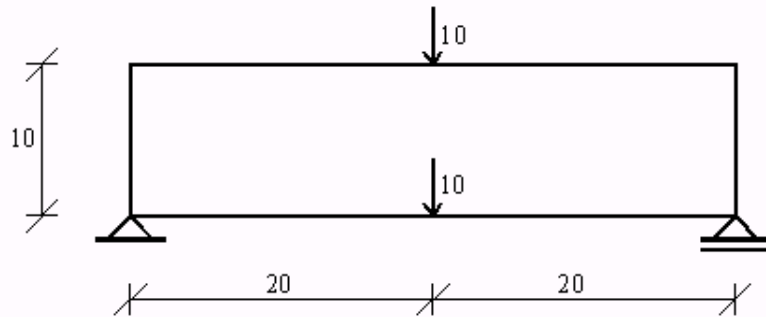


Fig. 1 Geometry and static loads

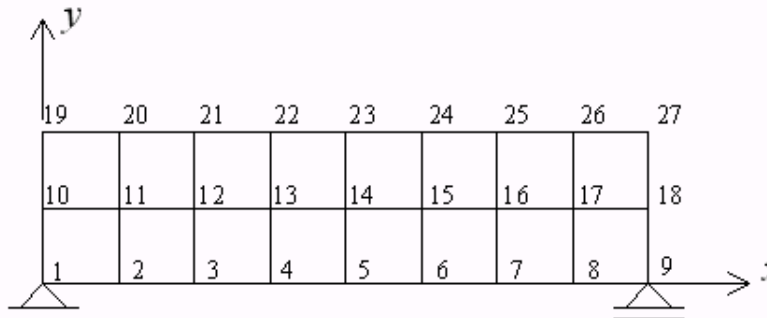


Fig. 2 Finite element mesh and node numbering

```
# some problem parameters
set L 40.0
set H 10.0
set thick 2.0

set P 10
set nX 9; # numNodes x dirn
set nY 3; # numNodes y dirn
```

```
# model builder
model Basic -ndm 2 -ndf 2
```

```
# create material
nDMaterial ElasticIsotropic 1 1000 0.25 3.0
```

```
# use block command
set cmd "block2D [expr $nX-1] [expr $nY-1] 1 1 \
quad \" $thick PlaneStress 1\" {
1 0 0
2 $L 0
3 $L $H
4 0 $H
}"
```

eval \$cmd

```
# apply loads
set midNode [expr ($nX+1)/2]
timeSeries Linear 1
pattern Plain 1 1 {
load $midNode 0 -$P
load [expr $midNode + $nX*($nY-1)] 0 -$P
}
analysis Static;
analyze 1;
print node $midNode
```

```
Terminal — bash — 85x37
examples> OpenSees n.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

Node: 5
  Coordinates : 20 0
  Disps: -1.37853e-16 -0.096041
  unbalanced Load: 0 -10
  ID : 26 27

examples> OpenSees o.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

Node: 5
  Coordinates : 20 0
  Disps: -1.37853e-16 -0.096041
  unbalanced Load: 0 -10
  ID : 26 27

examples> 
```

Cantilevered Circular Column

p.tcl

```

set P 1.0
set L 20.0
set R 1.0
set E 1000.0

set nz 20
set nx 6
set ny 6

set PI [expr 2.0 * asin(1.0)]
set I [expr $PI*pow((2*$R),4)/64.0]
puts "PL^3/3EI = [expr $P*pow($L,3)/(3.0*$E*$I)]"

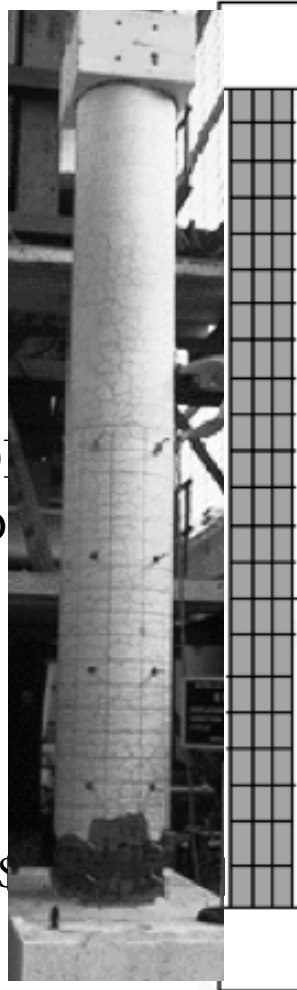
# Create ModelBuilder with 3 dimensions and 6 D
model Basic -ndm 3 -ndf 3

# create the material
nDMaterial ElasticIsotropic 1 $E 0.25 1.27

set eleArgs "1"
set element bbarBrick

set nn [expr ($nz)*($nx+1)*($ny+1) + (($nx+1)*($ny+1))]
set n1 [expr ($nz)*($nx+1)*($ny+1) + $nx]

```



```

# mesh generation
set sqrtR [expr sqrt($R/2.0)]
set cmd "block3D $nx $ny $nz 1 1 $eleArgs
  1 -$sqrtR -$sqrtR 0
  2 $sqrtR -$sqrtR 0
  3 $sqrtR $sqrtR 0
  4 -$sqrtR $sqrtR 0
  5 -$sqrtR -$sqrtR $L
  6 $sqrtR -$sqrtR $L
  7 $sqrtR $sqrtR $L
  8 -$sqrtR $sqrtR $L
  13 0 -$R 0
  14 $R 0 0
  15 0 $R 0
  16 -$R 0 0
  18 0 -$R $L
  19 $R 0 $L
  20 0 $R $L
  21 -$R 0 $L
  23 0 -$R [expr $L/2.0]
  24 $R 0 [expr $L/2.0]
  25 0 $R [expr $L/2.0]
  26 -$R 0 [expr $L/2.0]
}"

eval $cmd

# boundary conditions
fixZ 0.0 1 1 1

# Constant point load
pattern Plain 1 Linear {
  load $nn 0.0 $P 0.0
}

```

Terminal — OpenSees — 79x22

examples> OpenSees

OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.3.0.alpha

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved

(Copyright and Disclaimer @ <http://www.berkeley.edu/OpenSees/copyright.html>
)

OpenSees > source p.tcl

PL^3/3EI = 3.3953054526271007

Node: 1005

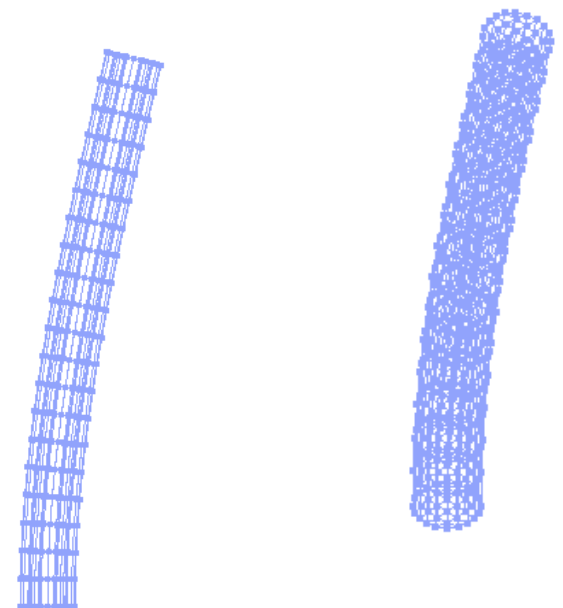
Coordinates : 0 0 20

Disps: 1.35333e-11 3.14833 1.24813e-14

unbalanced Load: 0 1 0

ID : 99 100 101

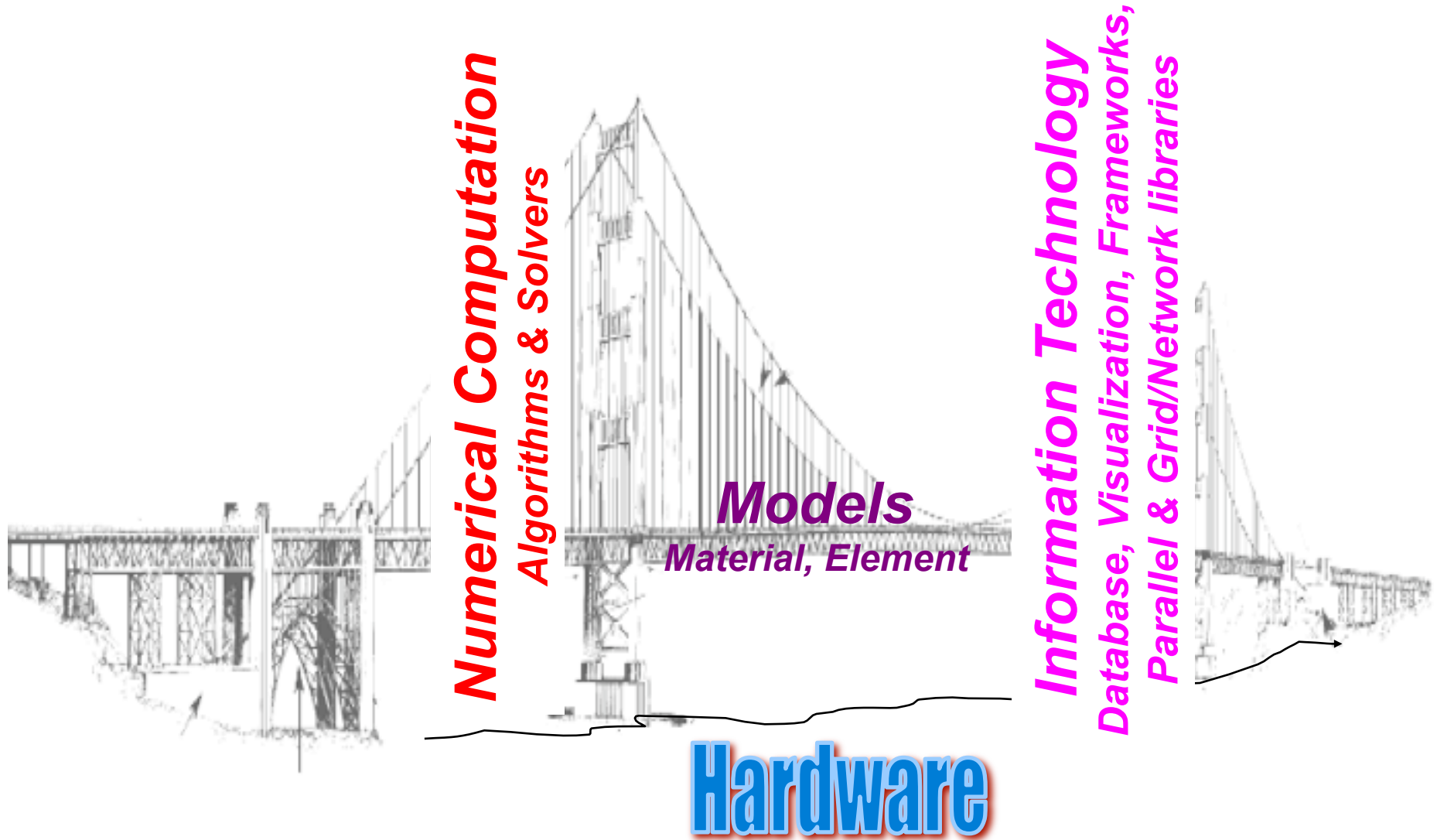
OpenSees > █



Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
 - OpenSees on NEEShub
 - Hands on Exercise
 - Adding Your Code to OpenSees
 - Advanced Model Development
 - Conclusion

Building Blocks for Simulation



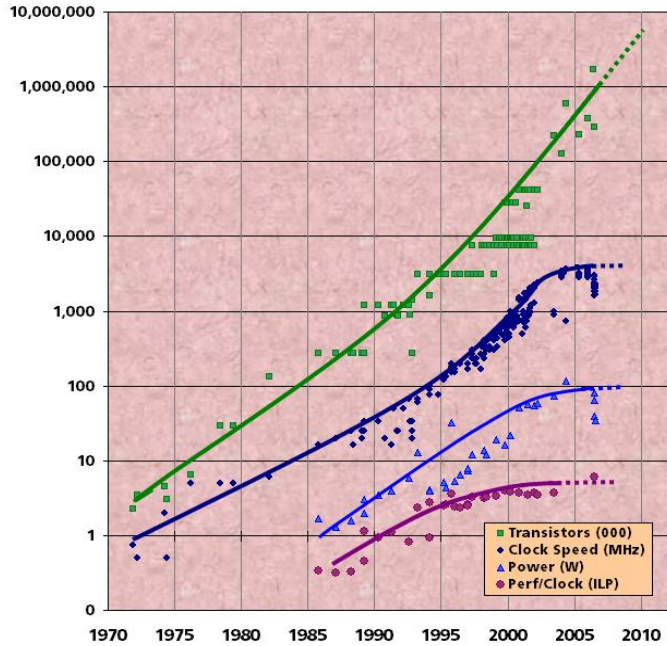
Bell's Law

Bell's Law of Computer Class formation

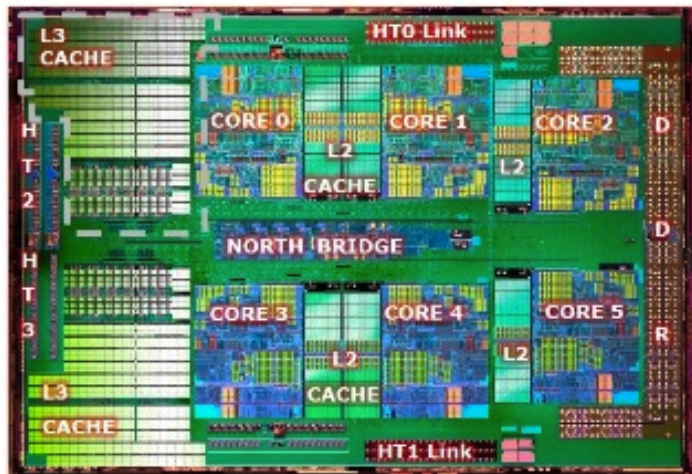
was discovered about 1972. It states that technology advances in semiconductors, storage, user interface and networking advance every decade enable a new, usually lower priced computing platform to form. Once formed, each class is maintained as a quite independent industry structure. This explains mainframes, minicomputers, workstations and Personal computers, the web, emerging web services, palm and mobile devices, and ubiquitous interconnected networks.

Gordon Bell, <http://research.microsoft.com/~GBell/Pubs.htm>

Technology is Changing



CHIP/Socket



Intel Processor Speed

Xeon E7 Server	72GFlop
i7 Desktop	55GFlop
i7 Mobile	30GFlop
i5 Desktop	40GFlop
i5 Mobile	22GFlop
Core2 Quad	48GFlop
Core2 Duo	25GFlop

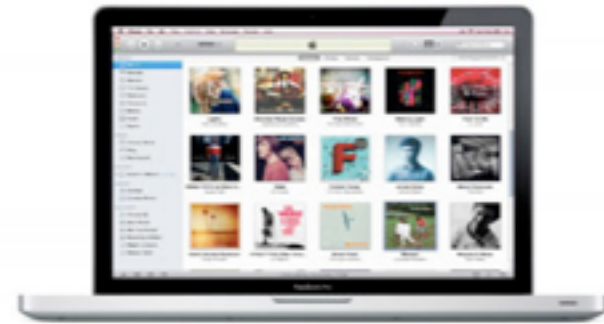
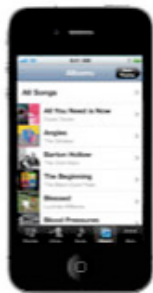


GPU & CO-Processors

STEVE JOBS KEYNOTE WWDC 2011

Some people think the cloud is just
A hard drive in the sky!

Cloud computing is internet-based computing ,
whereby shared resources, software, and information
are provided to computers and other devices on
demand, like the electricity grid. source: wikipedia



“..PC and Mac Demoted to a Device”

Cloud Computing **Compute Resources**

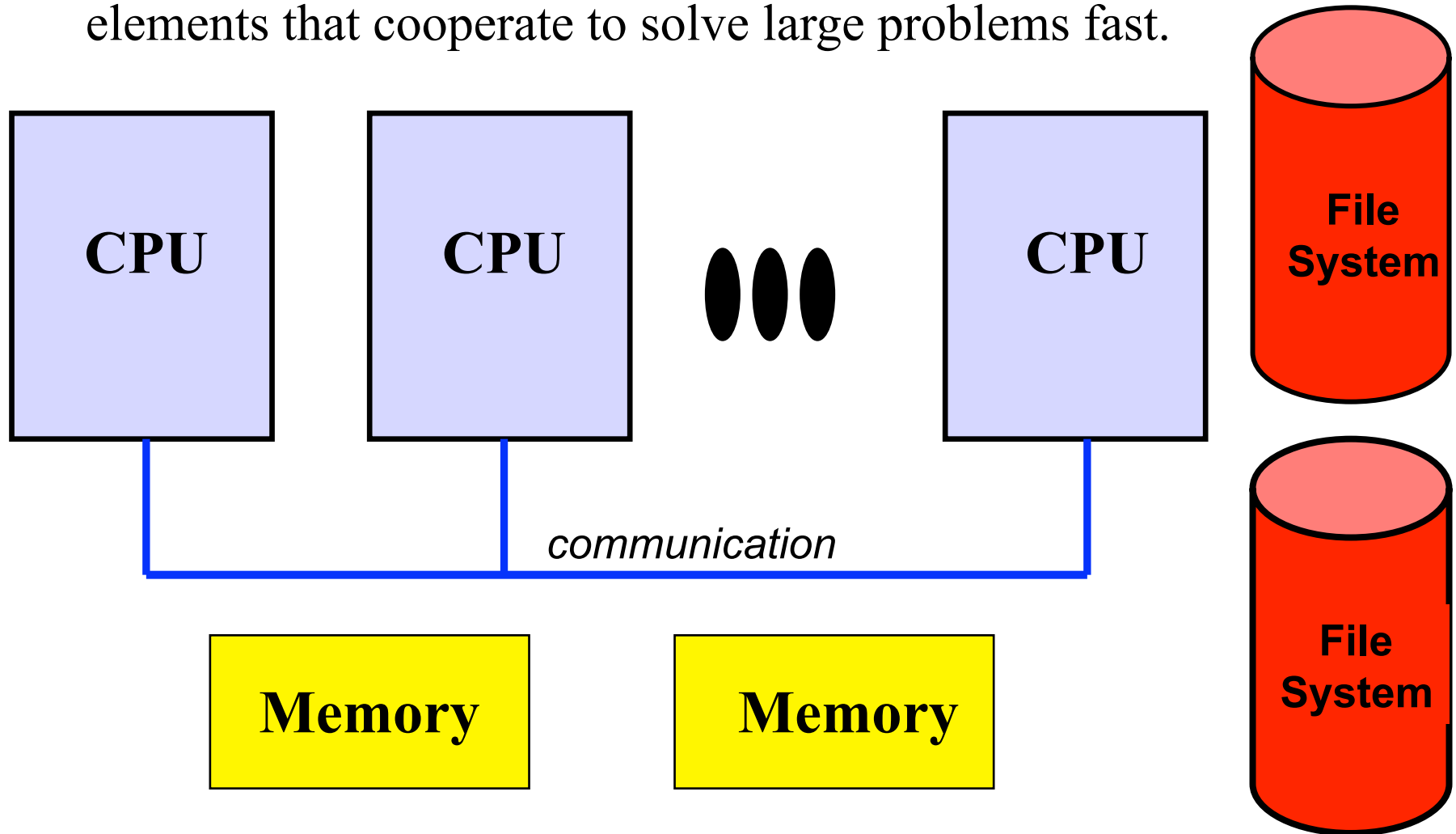
- Commercial



- Research
 - All of the above **PLUS**
 - Dedicated High Performance Computers
 - Grid (BOINC, OPEN SCIENCE GRID,)
 - **NEEShub (<http://nees.org>)**

What is a Parallel Computer?

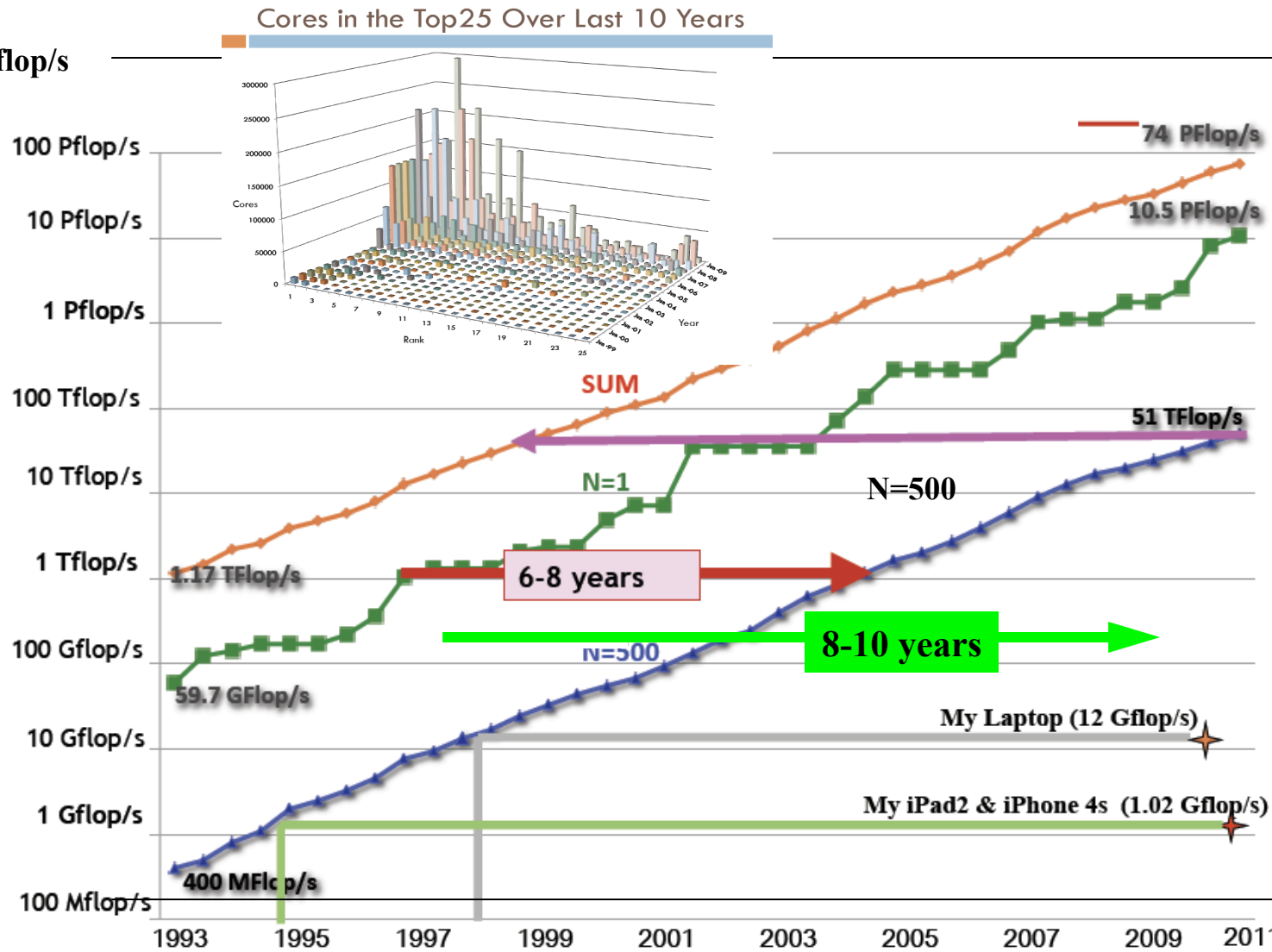
- A *parallel computer* is a collection of processing elements that cooperate to solve large problems fast.



Performance Projections

10e6 to 100e6
Cores!

1 Eflop/s



2019

Source: Jack Dongarra, 2011

Why should you care?

- They will save you time
- They will allow you to solve larger problems.
- They are **here** whether you like it or not!

HPC available through NEEShub

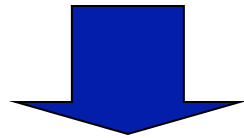


- Kraken (Tennessee)
 - Nodes: 9,408
 - Cores: 112,896
 - Memory: 142TB
 - Peak: 1.17PFlops
 - Disk: 3.3PB
- Stampede (Texas)
 - Nodes: 6,400
 - Cores: 102,400
 - Memory: 205TB
 - Peak: 9.5PFlops
 - Disk: 14PB

BEFORE YOU GET ALL EXCITED

Speedup & Amdahl's Law

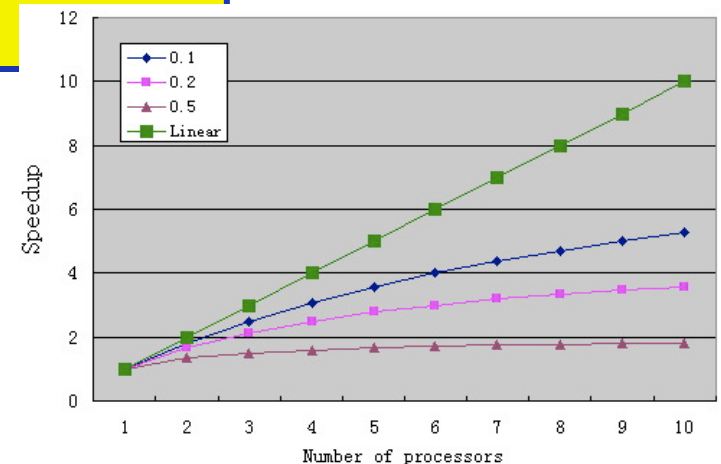
$$speedup_{PC}(p) = \frac{Time(1)}{Time(p)}$$



$$Speedup_{PC} = \frac{T_1}{\alpha T_1 + \frac{(1-\alpha)T_1}{n}} \rightarrow \frac{1}{\alpha} \text{ as } n \rightarrow \infty$$

Portion of sequential

of processors



Improving Real Performance

**Peak Performance grows exponentially,
a la Moore's Law**

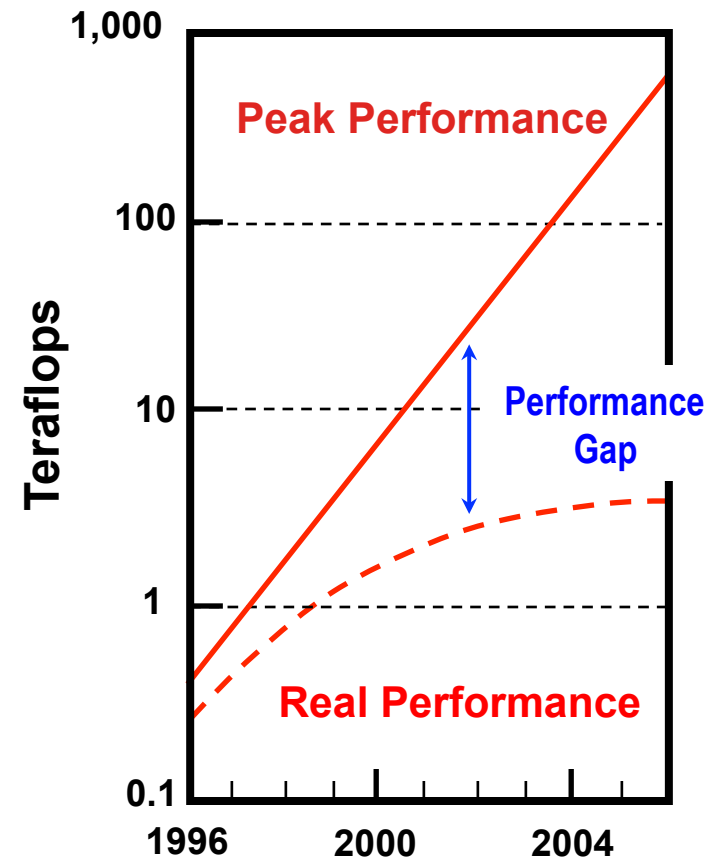
- In 1990's, peak performance increased 100x; in 2000's, it will increase 1000x

**But efficiency (the performance relative to
the hardware peak) has declined**

- was 40-50% on the vector supercomputers of 1990s
- now as little as 5-10% on parallel supercomputers of today

Close the gap through ...

- Mathematical methods and algorithms that achieve high performance on a single processor and scale to thousands of processors
- More efficient programming models and tools for massively parallel supercomputers

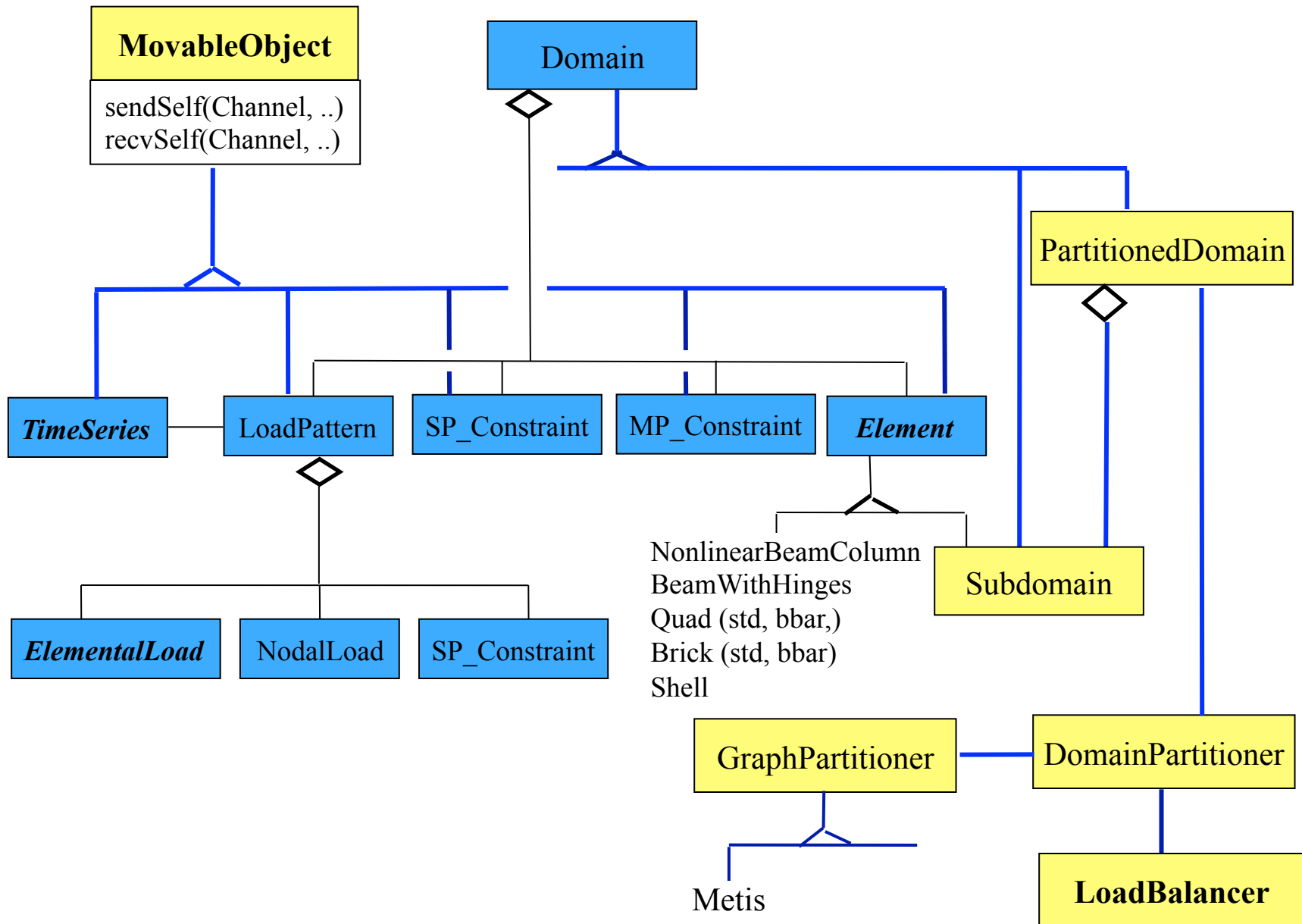


Source: Jim Demmell, CS267
Course Notes

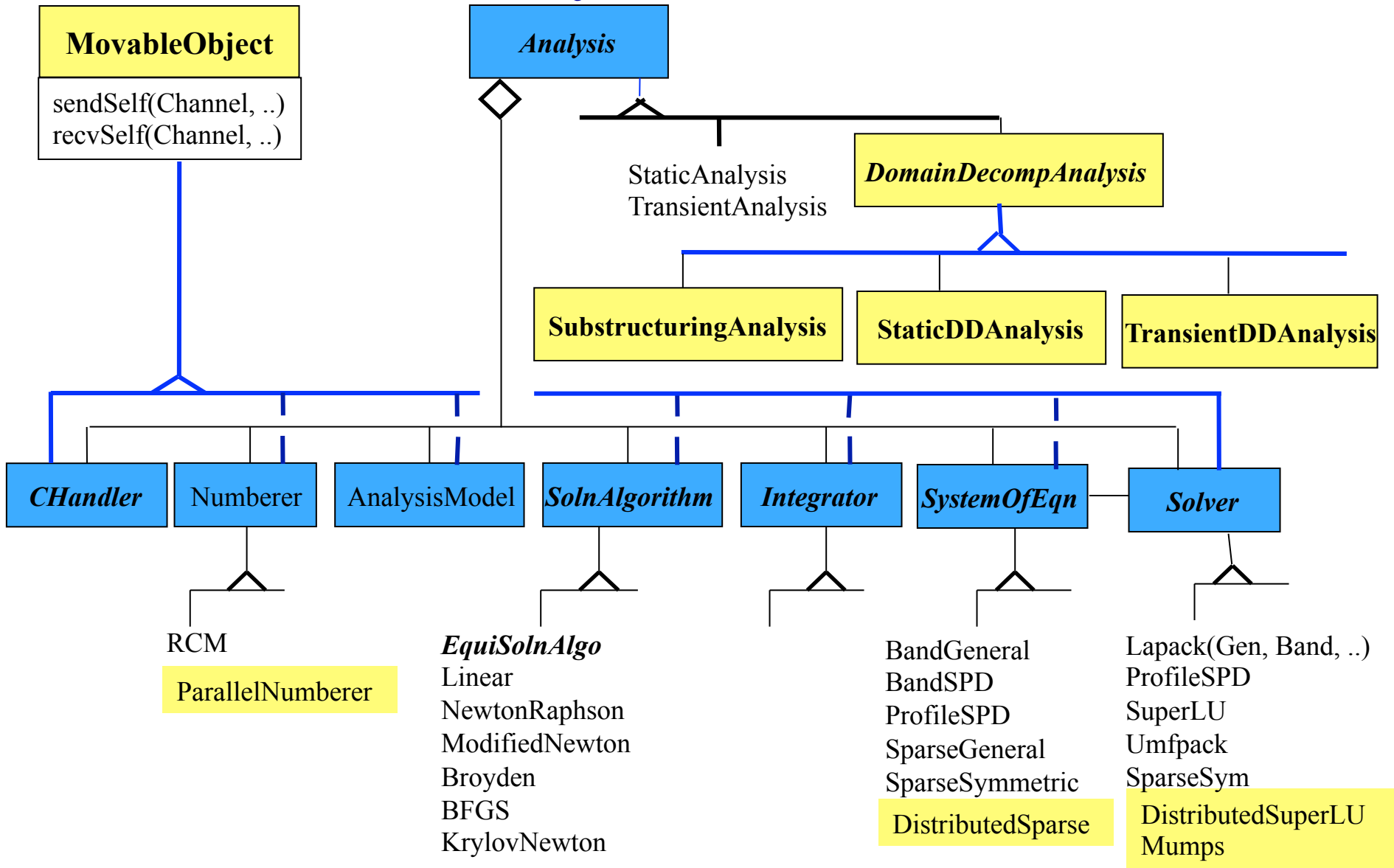
What is OpenSees?

- OpenSees is an Open-Source Software Framework written in C++ for developing nonlinear Finite Element Applications for both sequential and **PARALLEL** environments.

Domain Classes



Analysis Classes



The OpenSees Interpreters

- OpenSees.exe, OpenSeesSP.exe and OpenSeesMP.exe are applications that extend the Tcl interpreter for finite element.

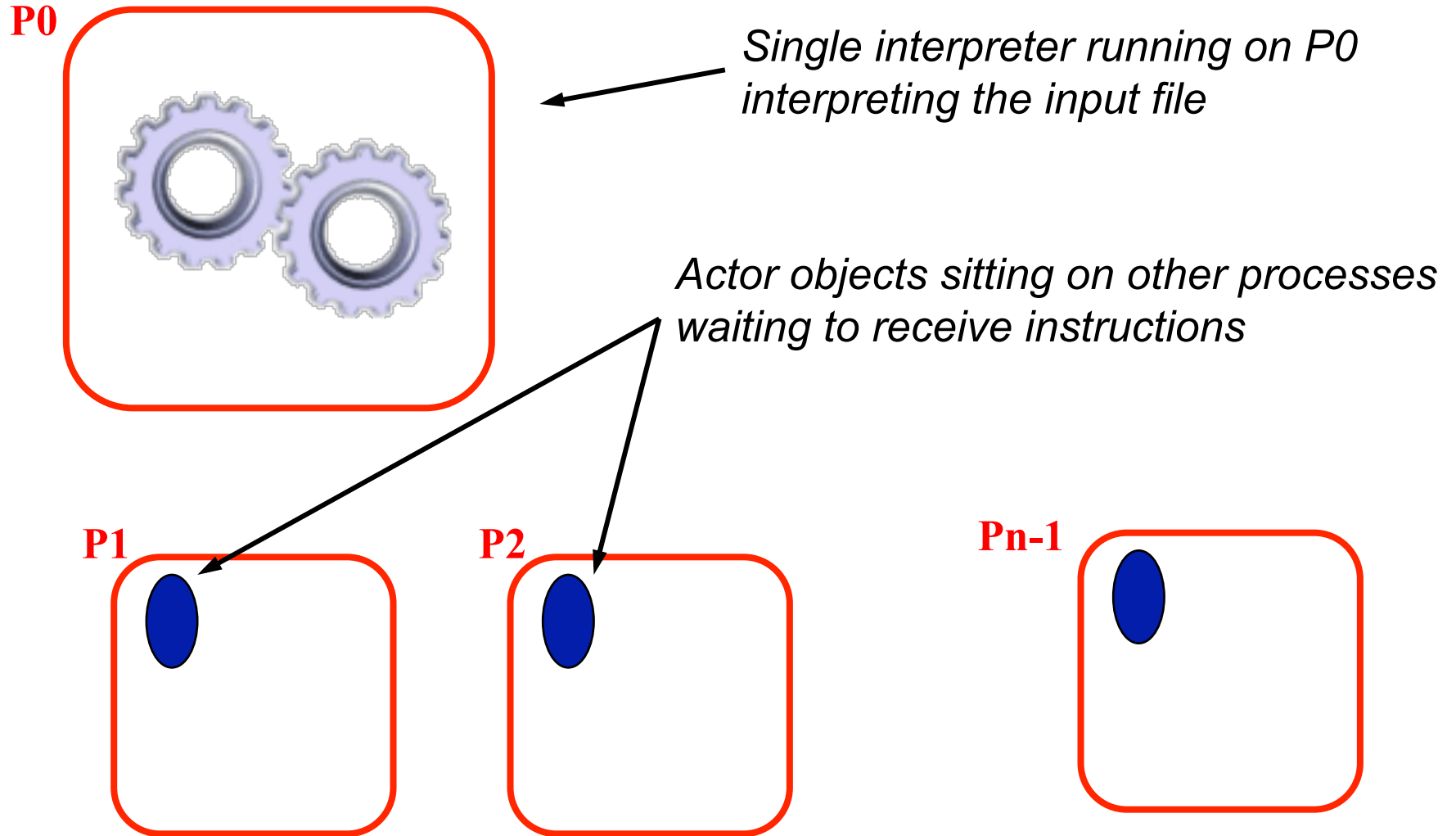
So What are OpenSeesSP.exe
and OpenSeesMP.exe ?

Parallel OpenSees Interpreters

- OpenSeesSP: An application for large models which will parse and execute the exact same script as the sequential application. The difference being the element state determination and equation solving are done in parallel.
- OpenSeesMP: An application for **BOTH** large models and parameter studies.

OpenSeesSP:

An application for Large Models



Modified Commands

- System command is modified to accept new parallel equation solvers

system Mumps

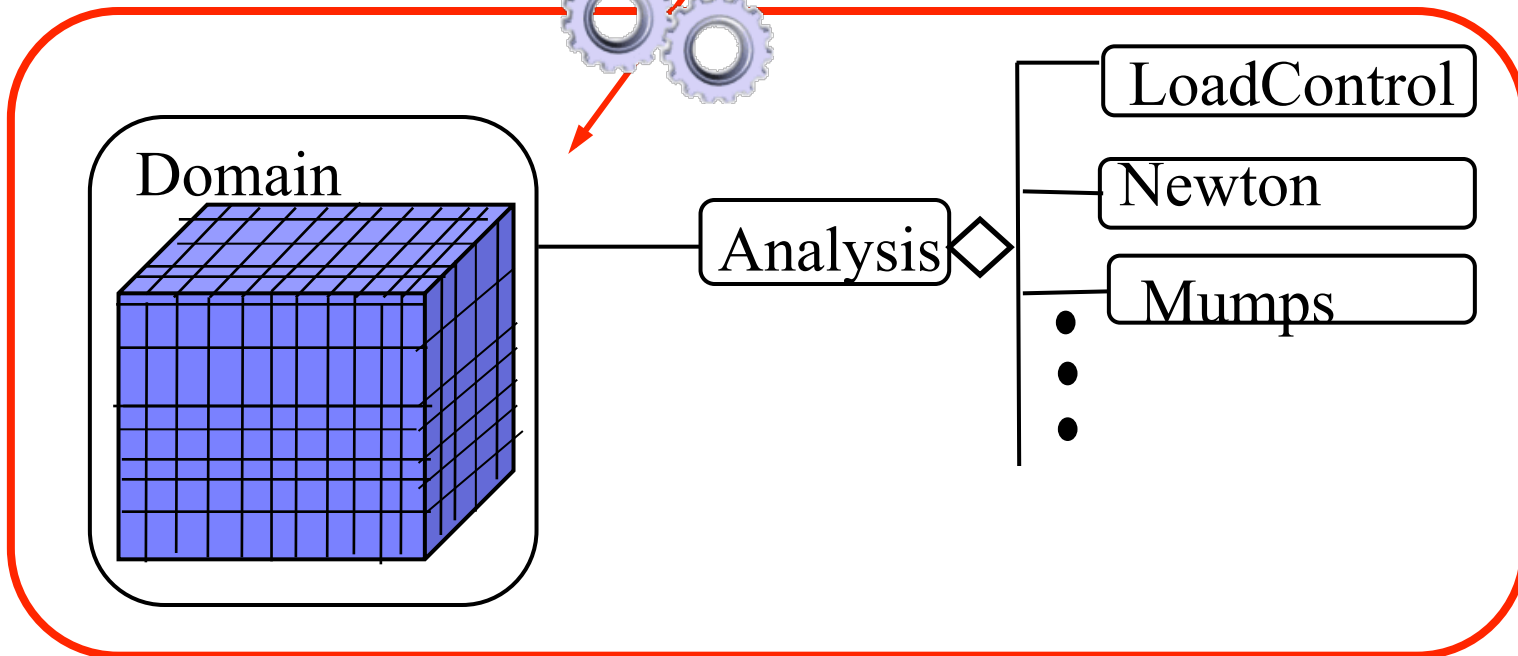
system Diagonal

Model Built and Analysis Constructed in P0

```
#build the model
source model.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDisplncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
```

Single interpreter running on P0
Interpreting the input file

P0



P1



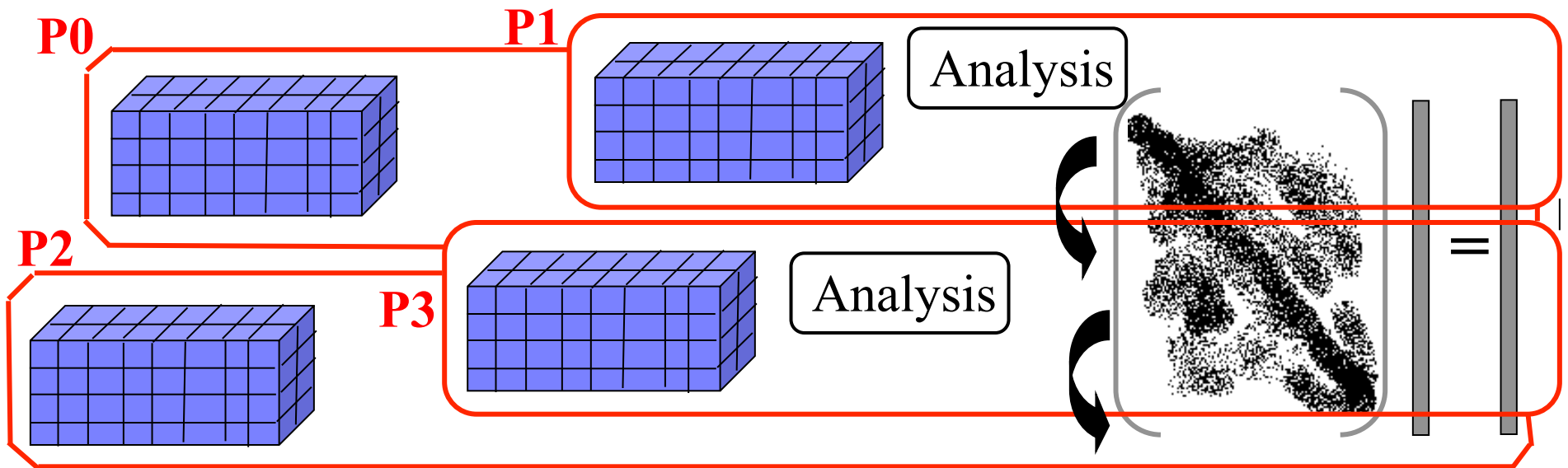
P2



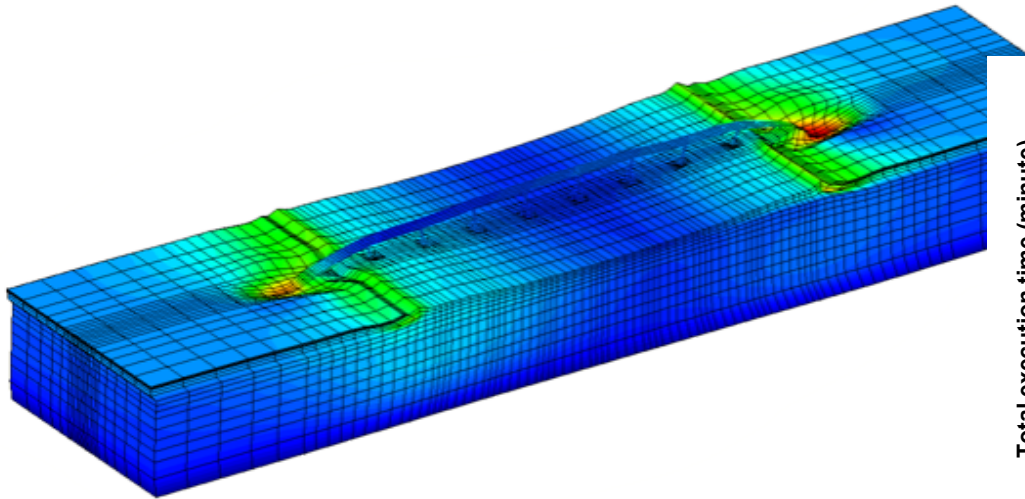
P3



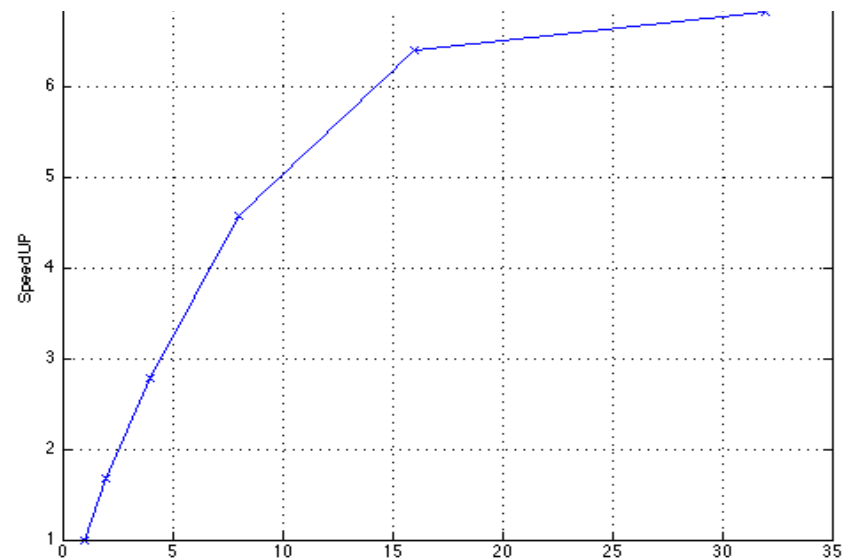
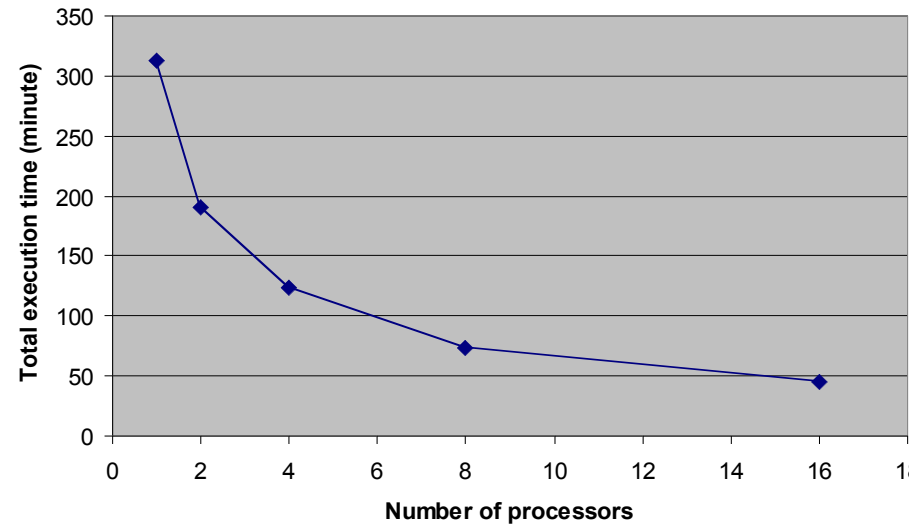
```
#build the model
source modelP.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDisplIncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
analyze 10
```



Example Results: Humboldt Bay Bridge Model

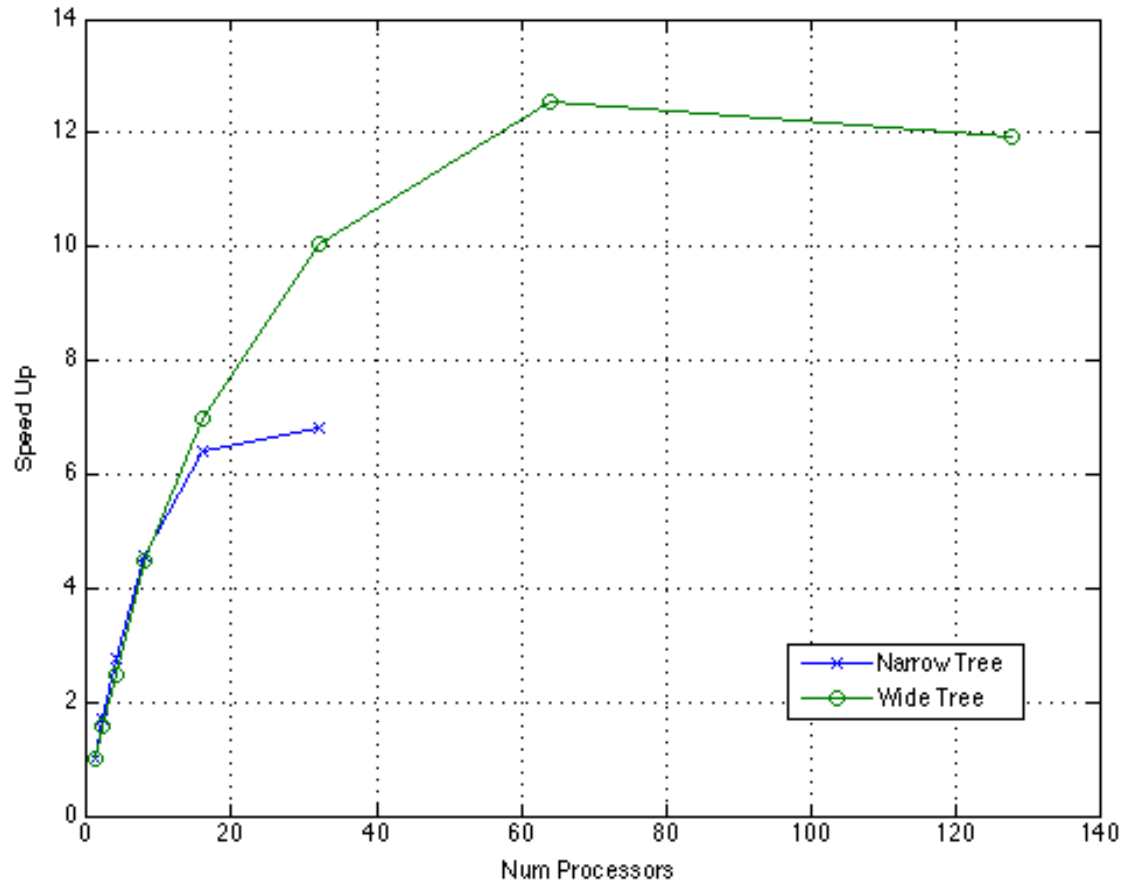


100,000+ DOF Model
Implicit Integration
Mumps Direct Solver



Effect of Shape of Elimination Tree on Performance:

50,000 DOF+ Models
Implicit Integration
Direct Solver



* Note the fatter the elimination tree, the better the parallel performance of the direct solver.

Run	el. size (m)	Elements	Nodes	DOFs
A	20	54,026	59,032	156,768
B	10	404,751	424,512	1,193,283
C	5	3,130,301	3,208,822	9,307,563
D	2.5	24,615,801	24,928,842	73,515,123

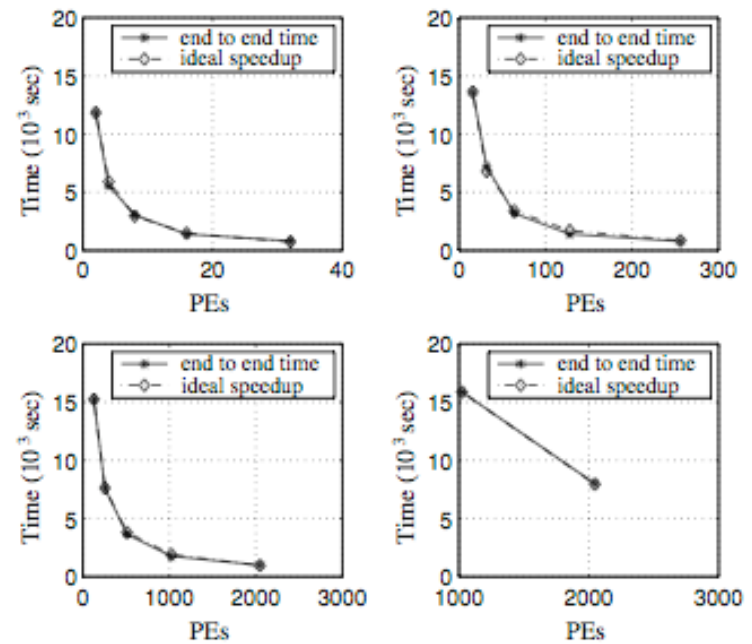
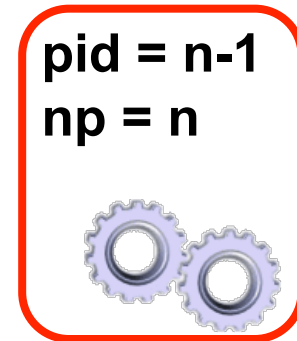
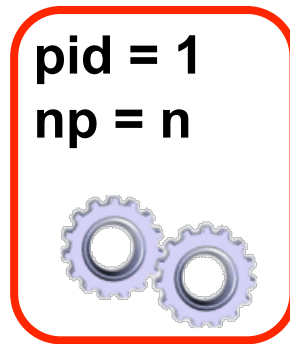
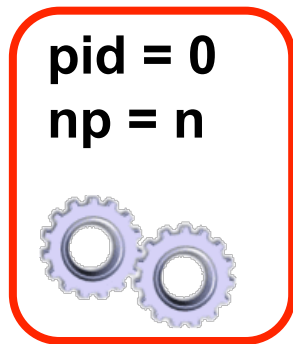


Fig. 18 Fixed-size, scalability plot at SDSC's DataStar. Upper row is runs A (*left*) and B (*right*), lower row is runs C (*left*) and D (*right*) (Table 3)

OpenSeesMP: An application for Large Models and Parameter Studies



Each process is running an interpreter and can determine its unique process number and the total number of processes in computation

Based on this script can do different things

```
# source in the model and analysis procedures
set pid [getPID]
set np [getNP]

# build model based on np and pid
source modelP.tcl
doModel {$pid $np}

# perform gravity analysis
system Mumps
constraints Transformation
numberer Parallel
test NormDisplncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl 0.1

analysis Static

set ok [analyze 10]
return $ok
```


New Commands added to OpenSeesMP:

- A Number of new commands have been added:
 1. `getNP` returns number of processes in computation.
 2. `getPID` returns unique process id {0,1, .. NP-1}
 3. `send -pid pid? data` pid = { 0, 1, .., NP-1}
 4. `recv -pid pid? variableName` pid = {0,1 .., NP-1, ANY}
 5. `barrier`
 6. `domainChange`
- These commands have been added to ALL interpreters (OpenSees, OpenSeesSP, and OpenSeesMP)

Example

ex2.tcl

```
set pid [getPID]
set np [getNP]
if {$pid == 0 } {
  puts "Random:"
  for {set i 1 } {$i < $np} {incr i 1} {
    recv -pid ANY msg
    puts "$msg"
  }
} else {
  send -pid 0 "Hello from Spid"
}
barrier
if {$pid == 0 } {
  puts "\nOrdered:"
  for {set i 1 } {$i < $np} {incr i 1} {
    recv -pid $i msg
    puts "$msg"
  }
} else {
  send -pid 0 "Hello from Spid"
}
```

```
Terminal — bash — 80x32
bin> mpirun -np 10 OpenSeesMP ex2.tcl
```

OpenSees -- Open System For Earthquake Engineering Si
Pacific Earthquake Engineering Research Center -- 1.7.

(c) Copyright 1999,2000 The Regents of the Univers
All Rights Reserved

(Copyright and Disclaimer @ <http://www.berkeley.edu/OpenSees>)

```
Random:
Hello from 1
Hello from 3
Hello from 5
Hello from 6
Hello from 8
Hello from 2
Hello from 4
Hello from 7
Hello from 9
Ordered:
Hello from 1
Hello from 2
Hello from 3
Hello from 4
Hello from 5
Hello from 6
Hello from 7
Hello from 8
Hello from 9
```

Steel Building Study

```
set pid [getPID]
set np [getNP]
set recordsFileID [open "peerRecords.txt" r]
set count 0;

foreach gMotion [split [read $recordsFileID] \n] {
  if {[expr $count % $np] == $pid} {

    source model.tcl
    source analysis.tcl

    set ok [doGravity]

    loadConst -time 0.0

    set gMotionList [split $gMotion "/"]
    set gMotionDir [lindex $gMotionList end-1]
    set gMotionNameInclAT2 [lindex $gMotionList end]
    set gMotionName [string range $gMotionNameInclAT2 0 end-4 ]

    set Gaccel "PeerDatabase $gMotionDir $gMotionName -accel 384.4 -dT dT -nPts nPts"
    pattern UniformExcitation 2 1 -accel $Gaccel

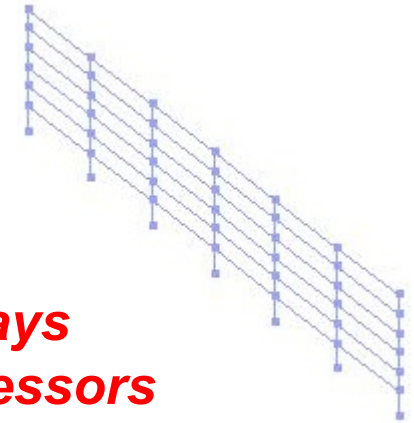
    recorder EnvelopeNode -file $gMotionDir$gMotionName.out -node 3 4 -dof 1 2 3 disp

    doDynamic [expr $dT*$nPts] $dT

    wipe
  }

  incr count 1;
}
```

7200 records
2 min a record
240 hours or 10 days
Ran on 2000 processors
on teragrid in less than 15 min.



Concrete Building Study

```
set pid [getPID]
set np [getNP]
set count 0;
source parameters.tcl
source ReadSMDFileNewFormat.tcl;
foreach GMfile $iGMFile {
  foreach Factor1248 $iFactor1248 {

    if {[expr $count % $np] == $pid} {

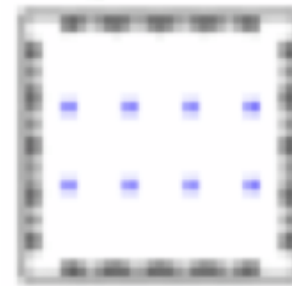
      set inFile $GMdir/$GMfile.AT2
      set outFile $GMdir/$GMfile.g3;
      ReadSMDFileNewFormat $inFile $outFile dt npts;

      wipe
      source GravityAnalysisScript.tcl

      loadConst -time 0.0;
      wipeAnalysis

      source EQ_Recorder.tcl
      source EQAnalysisScript.tcl

      if {$ok == 0} {
        puts "Process $pid $GMfile x $Factor1248 FINISHED OK modelTime [getTime]"
      } else {
        puts "Process $pid $GMfile x $Factor1248 FINISHED FAIL modeTime [getTime] desiredTime $TmaxAnalysis"
      }
      incr count 1
    }
  }
}
```



**113 records, 4 intensities
3 hour a record, 1356
hours or 56.5 days.**

**Ran on 452 processors
On XSEDE in less than 5
hours.**

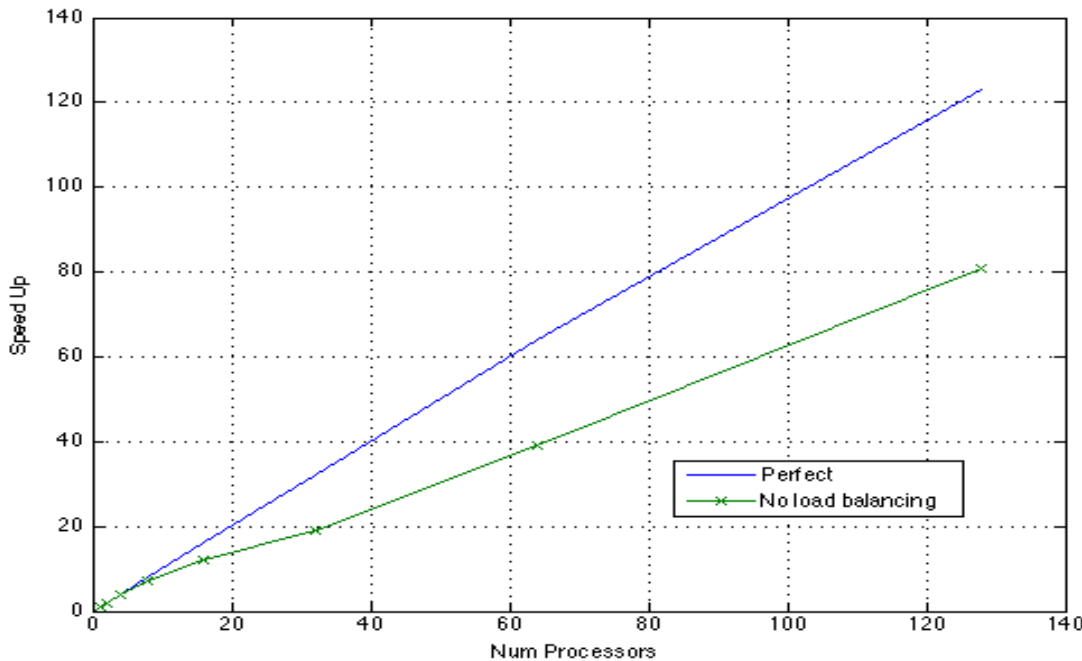
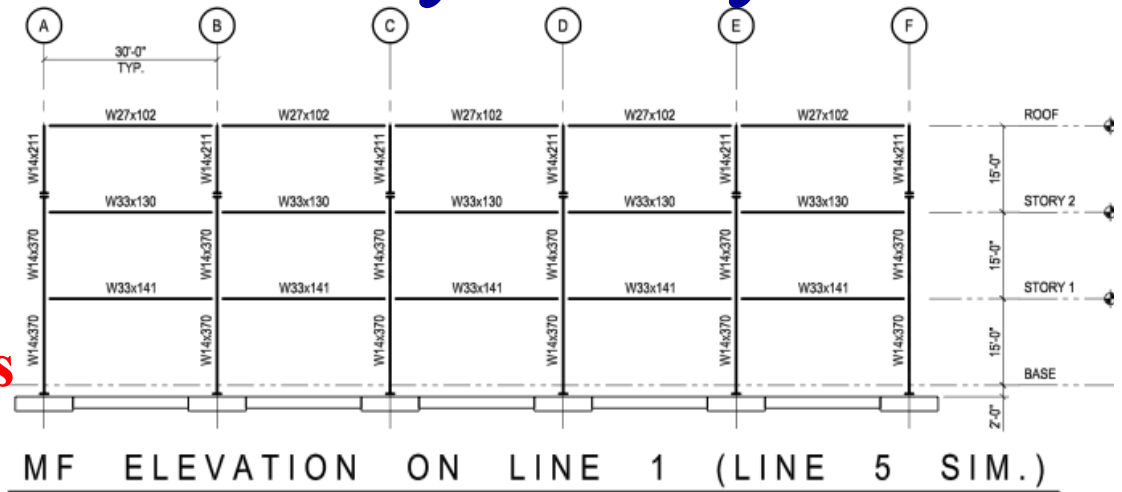
ATC-63/2 Project

***3 buildings, 2 config a building, 44 records, 12 intensities, 5 hour a record, would have, taken 15840 hours or 660 days or 1.8 years.
Ran on 44 processors of a XSEDE Ranger in less than 7 days.
(ATC-63/2 project using NEEShub)***

Example Results Parameter Study: Moment Frame Reliability Analysis

100+ DOF Model
Explicit Integration,
Mumps Direct Solver

8000 Earthquake Simulations



* If more simulations
than processors
SpeedUp
With “**Load Balancing**”
will be “**Linear**”

Modified Commands

- Some existing commands have been modified to allow analysis of large models in parallel:

1. numberer

numberer ParallelPlain

numberer ParallelRCM

2. system

system Mumps <-ICNTL14 %?>

3. integrator

integrator ParallelDisplacementControl node? Dof? dU?

- Use these **ONLY IF PARALLEL MODEL**

Example Parallel Model:

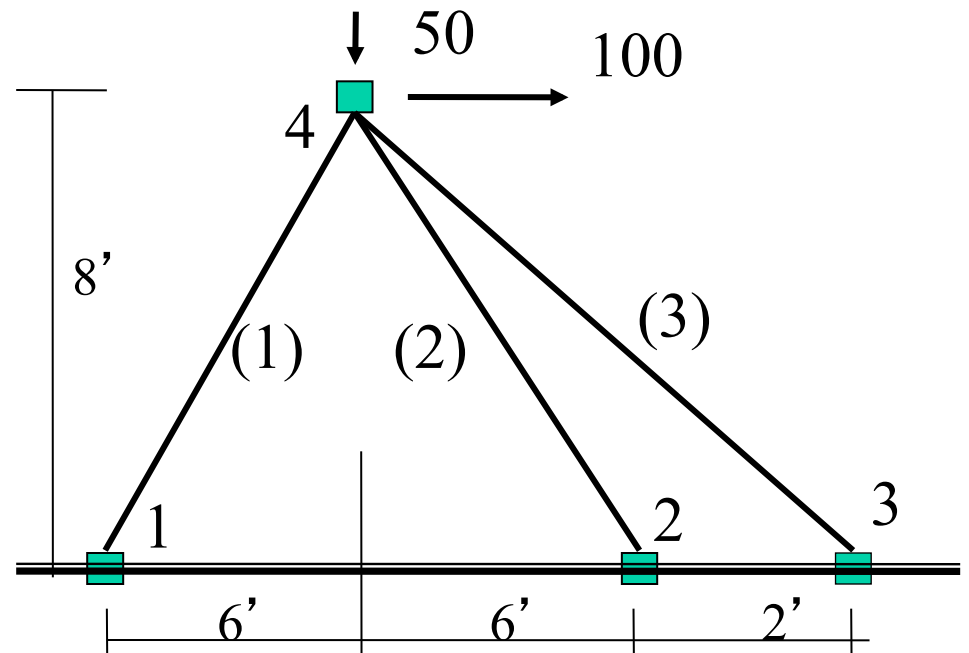
ex4.tcl

```

set pid [getPID]
set np [getNP]
if {$np != 2} exit

model BasicBuilder -ndm 2 -ndf 2
uniaxialMaterial Elastic 1 3000
if {$pid == 0} {
  node 1 0.0 0.0
  node 4 72.0 96.0
  fix 1 1 1
  element truss 1 1 4 10.0 1
  pattern Plain 1 "Linear" {
    load 4 100 -50
  }
} else {
  node 2 144.0 0.0
  node 3 168.0 0.0
  node 4 72.0 96.0
  fix 2 1 1
  fix 3 1 1
  element truss 2 2 4 5.0 1
  element truss 3 3 4 5.0 1
}

```



	E	A
1	3000	10
2	3000	5
3	3000	5

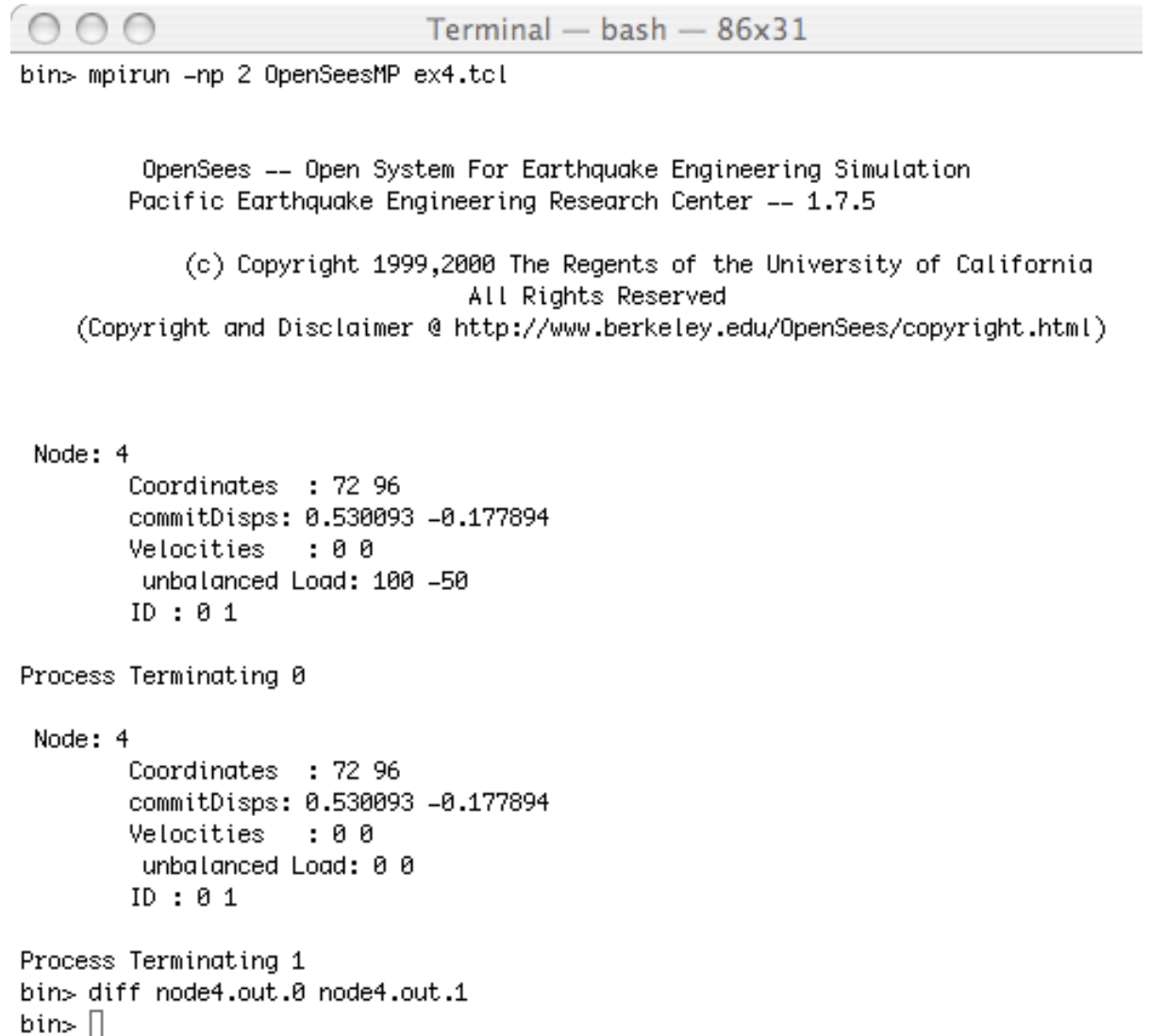
Example Parallel Analysis:

```
#create the recorder  
recorder Node -file node4.out.$pid -node 4 -dof 1 2 disp
```

```
#create the analysis  
constraints Transformation  
numberer ParallelPlain  
system Mumps  
test NormDispIncr 1.0e-6 6 2  
algorithm Newton  
integrator LoadControl 0.1  
analysis Static
```

```
#perform the analysis  
analyze 10
```

```
# print to screen node 4  
print node 4
```



```
Terminal — bash — 86x31  
bin> mpirun -np 2 OpenSeesMP ex4.tcl  
  
OpenSees -- Open System For Earthquake Engineering Simulation  
Pacific Earthquake Engineering Research Center -- 1.7.5  
  
(c) Copyright 1999,2000 The Regents of the University of California  
All Rights Reserved  
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)  
  
Node: 4  
Coordinates : 72 96  
commitDisps: 0.530093 -0.177894  
Velocities : 0 0  
unbalanced Load: 100 -50  
ID : 0 1  
  
Process Terminating 0  
  
Node: 4  
Coordinates : 72 96  
commitDisps: 0.530093 -0.177894  
Velocities : 0 0  
unbalanced Load: 0 0  
ID : 0 1  
  
Process Terminating 1  
bin> diff node4.out.0 node4.out.1  
bin> □
```

Parallel Displacement Control and domainChange!

ex5.tcl

```
source ex4.tcl

loadConst - time 0.0

if {$pid == 0} {
  pattern Plain 2 "Linear" {
    load 4 1 0
  }
}

domainChange

integrator ParallelDisplacementControl 4 1 0.1
analyze 10
```

```
Terminal — ba
Pacific Earthquake Engineering Research Center -- 1.

(c) Copyright 1999,2000 The Regents of the Unive
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/Oper

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 100 -50
ID : 0 1

Node: 4
Coordinates : 72 96
commitDisps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 0 0
ID : 0 1

Node: 4
Coordinates : 72 96
commitDisps: 1.53009 -0.194007
Velocities : 0 0
unbalanced Load: 200.668 -50
ID : 0 1

Process Terminating 0

Node: 4
Coordinates : 72 96
commitDisps: 1.53009 -0.194007
Velocities : 0 0
unbalanced Load: 0 0
```

Things to Watch For

1. Deadlock (program hangs)
 - send/recv messages
 - Opening files for writing & not closing them
2. Race Conditions (different results every time run problem)
 - parallel file system.
3. Load Imbalance
 - poor initial task assignment.

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

NEEShub (First Release July 2010)

The screenshot shows the NEEShub website homepage. The browser address bar displays <https://neeshub.org/home>. The page features a navigation menu with categories like Tools & Resources, Learning & Outreach, Project Warehouse, Sites, Collaborate, and Explore. A main banner area includes a video player for 'Earthquake Damage' and a text box describing the Network for Earthquake Engineering Simulation (NEES). Below this, there are sections for 'In the Spotlight' (highlighting virtual environment and laboratory tools), 'Use NEEShub to...' (listing access to projects, simulation tools, and learning resources), and 'NEES Videos on YouTube' (showing various engineering and construction videos). The bottom section is divided into three columns: 'Events and Activities' (featuring a calendar for July and August), 'News and Announcements' (listing recent news items like NSF Sensational 60 and NEEScomm updates), and 'Latest Earthquake Reports' (providing details on recent seismic events in Vanuatu and Papua New Guinea).

NEEShub
George E. Brown, Jr. Network for Earthquake Engineering Simulation

Logout | My Account | My NEEShub | About NEES | Sitemap | Feedback

477 New Messages
Thomas Hacker (thacker)

Tools & Resources | Learning & Outreach | Project Warehouse | Sites | Collaborate | Explore

Support

Network for Earthquake Engineering Simulation (NEES) is a shared national network of 14 experimental facilities, collaborative tools, a centralized data repository, and earthquake simulation software. Together, these resources provide the means for collaboration and discovery in the form of advanced research based on experimentation and computational simulations of the ways buildings, bridges, utility systems, coastal regions, and geomaterials perform during seismic events. [Learn more](#)

View Activity Map

In the Spotlight

- Visual Understanding Environment: Map your ideas and concepts in this mapping tool. - in Tools
- Virtual Laboratory - Nonlinear Two Story Building - in Tools
- Anchorage Detailing Effects on Lateral Deformation Components of ... - in Publications

Use NEEShub to...

- Access NEES projects- Project Warehouse
- Run simulators and other Tools
- Learn with earthquake data and simulators - NEES Academy
- Share research - Contribute Content

NEES Videos on YouTube

- 2008 NSLUC Engineering Program at UC Berkeley
- NEES@UTexas
- New Orleans Levee Centrifuge Models
- UC San Diego NEES Shake Table at Englekirk Center
- Grand Opening of nees@berkeley Facility
- 10 News coverage of School Quake Competition

Events and Activities

View All | Submit an event

JULY						
S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

JUL 24 9th US National & 10th Canadian Conference on Earthquake Engineering

AUG 10 International Workshop on Conservation of Heritage Structures Using FRM and SHM (CSHM-3 2010)

News and Announcements

View All | Subscribe to Newsletters

- National Science Foundation Sensational 60**
02 Jul 2010
NEES FEATURED IN NSF SENSATIONAL 60: The past 60 years have seen incredible developments in ...
- NEEScomm Update: June 2010**
02 Jul 2010
Julio Ramirez, Director of NEES Operations, provides the NEES community with an informal update of ...
- Gulf Oil Spill MRI RAPID Submission**
02 Jul 2010
In light of the recent oil spill in the Gulf of Mexico, the National Science Foundation (NSF) has ...
- NEEScomm Offers Free, Open WebEx Training Sessions To Community**
01 Jul 2010
NEEScomm will be offering--through WebEx--a series of free, open training sessions to provide you ...

Latest Earthquake Reports

View All

- JUL 22** M 6.2, Vanuatu
Thursday, July 22, 2010 05:04:01 UTC
Thursday, July 22, 2010 04:04:01 PM at epicenter
Depth: 35.20 km (21.87 mi)
- JUL 21** M 5.1, New Britain region, Papua New Guinea
Thursday, July 22, 2010 03:54:27 UTC
Thursday, July 22, 2010 01:54:27 PM at epicenter
Depth: 35.00 km (21.75 mi)
- JUL 21** M 5.3, Vanuatu
Wednesday, July 21, 2010 22:36:38 UTC
Thursday, July 22, 2010 09:36:38 AM at epicenter
Depth: 82.80 km (51.45 mi)

* The power behind NEES

NEEShub Tools and Resources

Documents, Learning Objects, Series & TOOLS

Simulation

The screenshot shows the inDEED software interface. On the left, there is a 'Name' list with columns for 'Name' and 'Type'. The list includes various sensors such as 'Table1_vel_y' (velocity), 'Table2_acc_x' (acceleration), 'Table2_acc_y' (acceleration), 'Table2_disp_x' (displacement), 'Table2_disp_y' (displacement), 'Table2_vel_x' (velocity), 'Table2_vel_y' (velocity), 'Table3_acc_x' (acceleration), 'Table3_acc_y' (acceleration), 'Table3_disp_x' (displacement), 'Table3_disp_y' (displacement), 'Table3_vel_x' (velocity), 'Table3_vel_y' (velocity), and 'T2YFrameAccel' (acceleration). Below the list is a 'Find sensors' section with a search box and a 'Plot selected sensors' section with a 'Plot' button. The main area shows a 3D model of a structure with various sensors placed on it. The interface includes a menu bar (File, Edit, View, Plot) and a 'Help' button.

OpenSeesLab

NEEShub

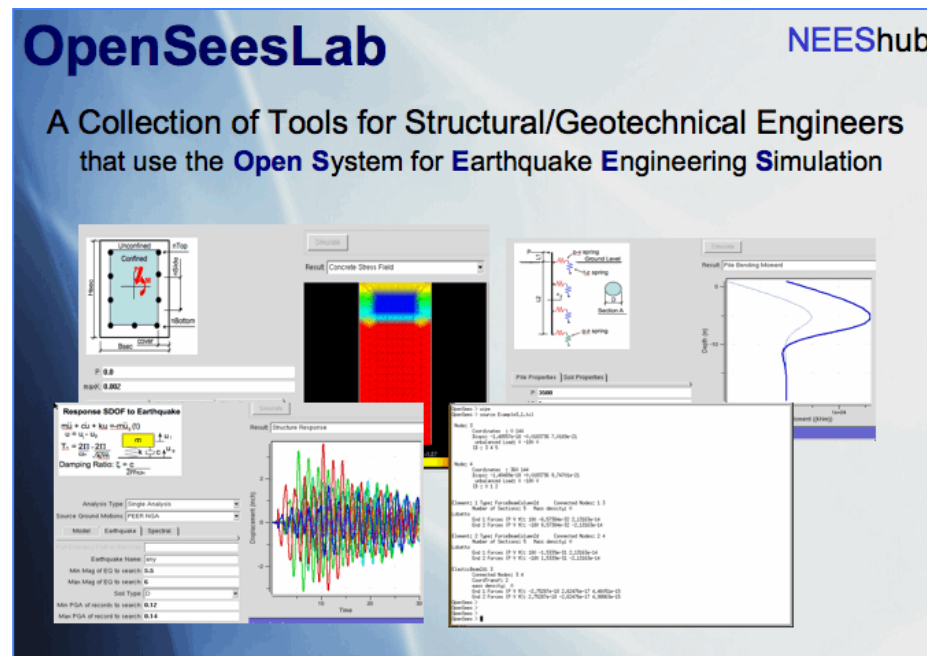
A Collection of Tools for Structural/Geotechnical Engineers that use the Open System for Earthquake Engineering Simulation

The screenshot shows the OpenSeesLab software interface. It displays several simulation results and plots. On the left, there is a 'Response SOOF to Earthquake' plot showing 'Component (mm)' vs 'Time'. The plot shows a complex, multi-colored oscillating signal. In the center, there is a 'Concrete Stress Field' plot showing a color-coded stress distribution. On the right, there is a 'Pile Bending Moment' plot showing 'Depth (ft)' vs 'Moment (k-ft)'. The interface includes a menu bar (File, Edit, View, Plot) and a 'Help' button. The bottom right corner shows a 'Results' window with a list of elements and their properties.

Data Management

The OpenSeesLab tool:

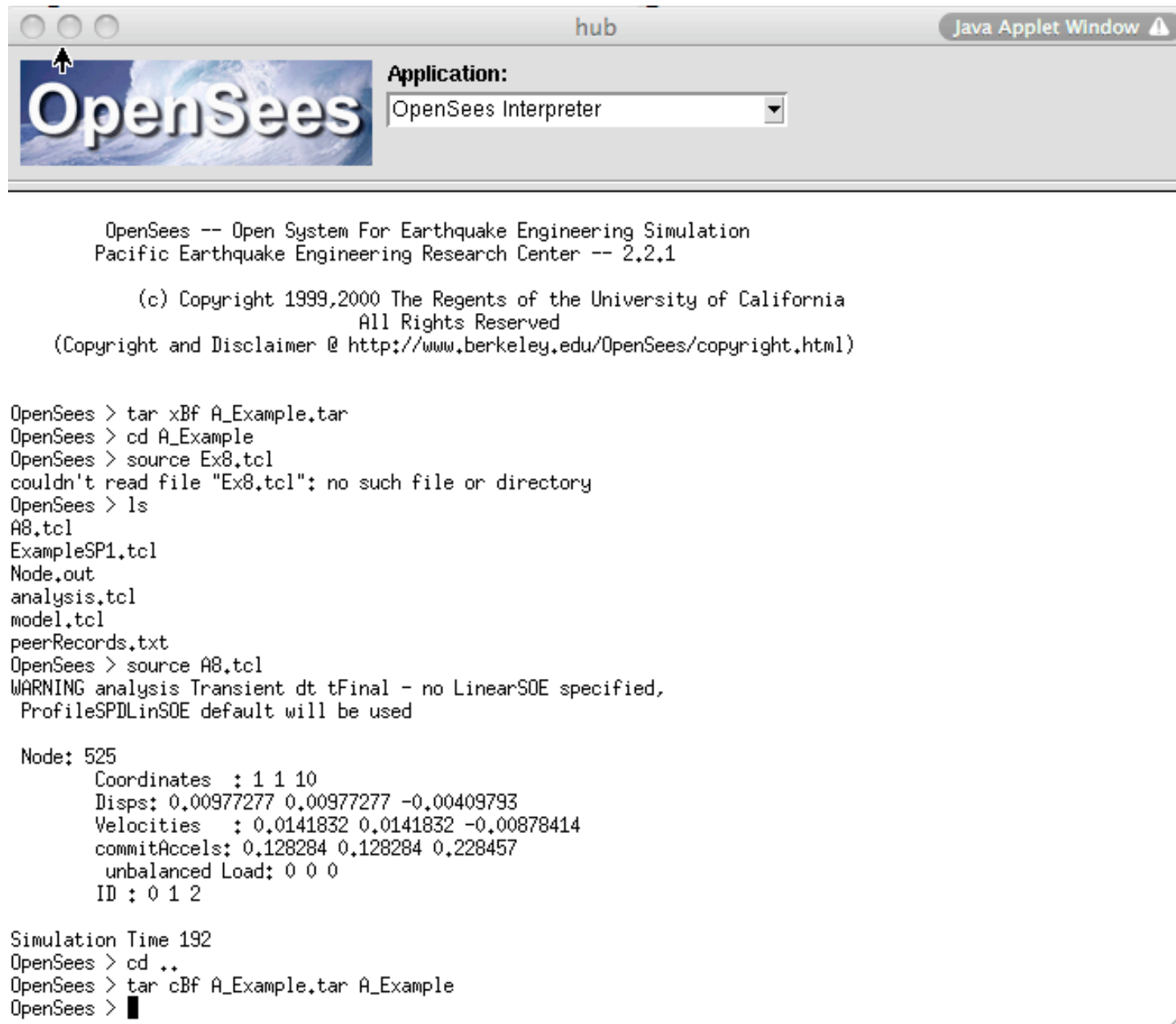
<http://nees.org/resources/tools/openseeslab>



Is a suite of Simulation Tools powered by OpnSees for:

1. Submitting OpenSees scripts (input files) to HUB resources
2. Educating students and practicing engineers

OpenSees Interpreter Tool



```
OpenSees -- Open System For Earthquake Engineering Simulation
Pacific Earthquake Engineering Research Center -- 2.2.1

(c) Copyright 1999,2000 The Regents of the University of California
All Rights Reserved
(Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

OpenSees > tar xBf A_Example.tar
OpenSees > cd A_Example
OpenSees > source Ex8.tcl
couldn't read file "Ex8.tcl": no such file or directory
OpenSees > ls
A8.tcl
ExampleSP1.tcl
Node.out
analysis.tcl
model.tcl
peerRecords.txt
OpenSees > source A8.tcl
WARNING analysis Transient dt tFinal - no LinearSOE specified,
ProfileSPDLinSOE default will be used

Node: 525
Coordinates : 1 1 10
Disps: 0.00977277 0.00977277 -0.00409793
Velocities : 0.0141832 0.0141832 -0.00878414
commitAccels: 0.128284 0.128284 0.228457
unbalanced Load: 0 0 0
ID : 0 1 2

Simulation Time 192
OpenSees > cd ..
OpenSees > tar cBf A_Example.tar A_Example
OpenSees > █
```


Parallel Script Submission Tool

The screenshot shows the OpenSees Parallel Job Submission Tool interface. The 'Application' dropdown is set to 'Parallel Job Submission'. The 'Main Script' field contains the path '/apps/openseesbuild/current/NEEShubExam'. The 'Resource' dropdown is set to 'Ranger'. The 'Ranger Options' section shows 'Application' set to 'OpenSeesMP', '# Processors' set to '512', and 'Walltime' set to '04:00:00'. A 'Simulate' button is visible. Red annotations include a box around the 'Main Script' field labeled 'Full Path to main script', a circle around the 'Resource' dropdown labeled 'To Select Machine', and a circle around the 'Application' dropdown in the 'Ranger Options' section labeled 'To Select Application'.

OpenSees Application: Parallel Job Submission

Main Script: /apps/openseesbuild/current/NEEShubExam

Resource: Ranger

Ranger Options

Application: OpenSeesMP

Processors: 512

Walltime: 04:00:00

Simulate new input parameters

Parallel Job Submission Tool

This tool can be used to submit parallel opensees jobs. The user is asked which parallel OpenSees application to use, which parallel machine to run on, how many processes to run and which parallel machine to run these on. The results from the analysis when completed will be placed in the users /scratch directory. The actual directory location will be shown in the result screen.

NOTE: the main script CANNOT be located in your home directory. It and the files it requires must be in a subdirectory.

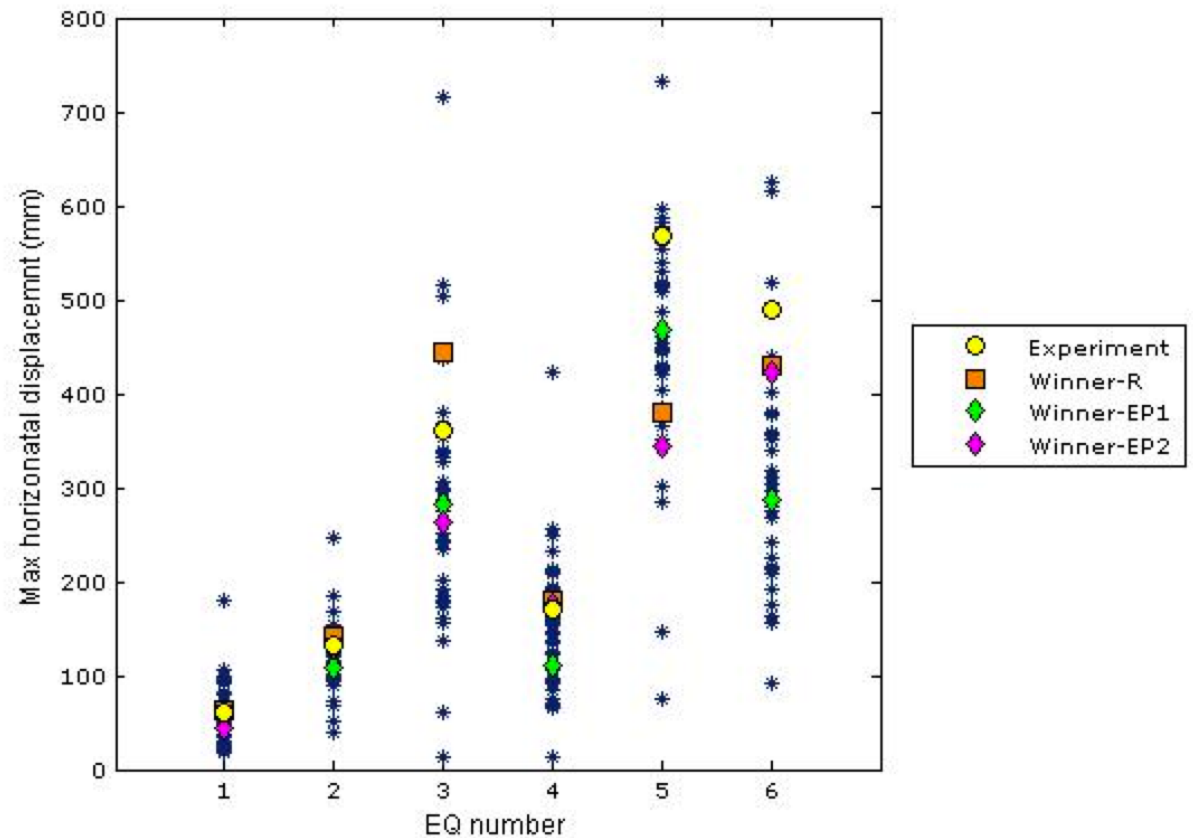
NOTE: control will return after the job has been submitted, AND NOT after the job has completed. This means you may have to wait awhile before the actual results are located in your output directory.

NOTE: as an example set the main script as:
/apps/openseesbuild/current/NEEShubExamples/SmallIMP/Example.tcl

DO NOT SELECT OpenSeesSP for this example.

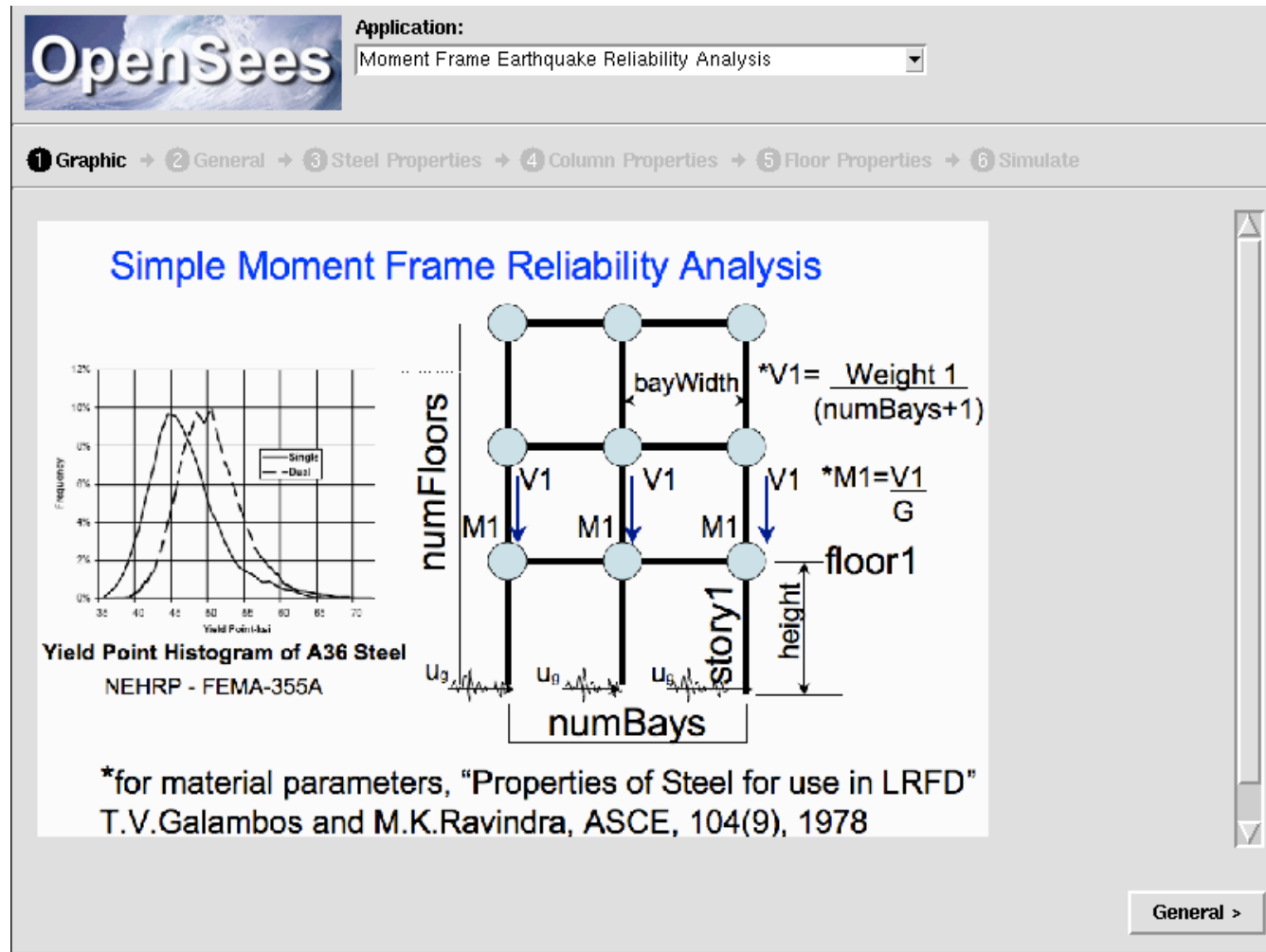
STRUCTURAL & Modeling Uncertainties

* PEER Concrete Column Blind Prediction Contest 2010



How Can This Uncertainty Be Ignored in your Evaluation?

Moment Frame Reliability Analysis



<http://nees.org/resources/tools/openseeslab>



Application:

Moment Frame Earthquake Reliability Analysis

1 Graphic → 2 **General** → 3 Steel Properties → 4 Column Properties → 5 Floor Properties → 6 Simulate

This tool performs a simplistic monte-carlo analysis of a steel moment frame building. The intent of the tool is to demonstrate the limitation of using a single finite element analysis to predict the response of a real structure.

1) The user inputs the number of simulations that they wish to perform. For each simulation, a random set of steel properties are created for each element in the model, i.e. each element in the model will have different properties. The default is a small number. If you play with the tool, you will see you will need a lot more if you want to get information on the tails.

2) The user also must specify a directory containing at least one earthquake motion. These files must follow the PEER At2 format, i.e. contain information about the number of points and deltaT as well as contain all the values. Examples are provided that you can look at.

3) The user also specifies the number of bays and floors in the frame. For now all columns and beams in a story or bay can only have the same steel section assigned to them. This will change if enough requests come in for it.

The user on the following pages will enter the model information, e.g. element types, physical dimensions. The results are displayed graphically for each earthquake. The red dot for each earthquake are the results for a simulation in which the mean properties alone were used. The graphics are somewhat lame. I will put in histograms if enough requests come in.

This tool utilizes Pegasus. Pegasus is a powerful workflow management software system for managing workflows (<http://pegasus.isi.edu>) that can be used for workflows that are typically more complicated than this.

#Simulations: **10**

Model Properties

num Floors: **3**

num Bays: **5**

Full Directory Path to Records:

Units:

< Graphic

Steel Properties >



Application:

Moment Frame Earthquake Reliability Analysis

1 Graphic → 2 General → 3 **Steel Properties** → 4 Column Properties → 5 Floor Properties → 6 Simulate

Steel Properties

Youngs Modulus E

Yield Strength

Hardening Ratio (Est/E)

Mean: **50**

CV%: **10**

Distribution: LogNormal

< General

Column Properties >

Application:

Moment Frame Earthquake Reliability Analysis

1 Graphic → 2 General → 3 Steel Properties → **4 Column Properties** → 5 Floor Properties → 6 Simulate

Element Type: ForceBeamColumn

Story1

Height: 180

Section: W14x370

Story2

Height: AsBelow

Section: AsBelow

Story3

Height: AsBelow

Section: W14x211

< Steel Properties

Floor Properties >

Application:

Moment Frame Earthquake Reliability Analysis

1 Graphic → 2 General → 3 Steel Properties → 4 Column Properties → **5 Floor Properties** → 6 Simulate

Element Type: ForceBeamColumn

Bay Widths: 360.0

Floor1

Section: W33x141

Weight: 972

Floor2

Section: W33x130

Weight: AsBelow

Floor3

Section: W27x102

Weight: 997

< Column Properties

Simulate >



Application:

Moment Frame Earthquake Reliability Analysis

1 Graphic → 2 General → 3 Steel Properties → **4 Column Properties** → 5 Floor Properties → 6 Simulate

Element Type: ForceBeamColumn

Story1

Height: 204.

Section: W14x370

Story2

Height: 180.

Section: AsBelow

Story3

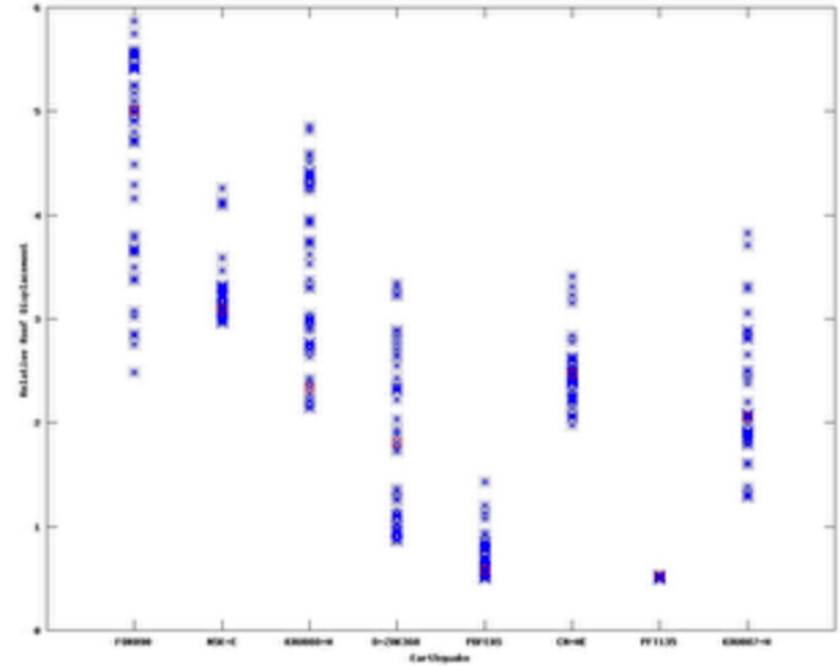
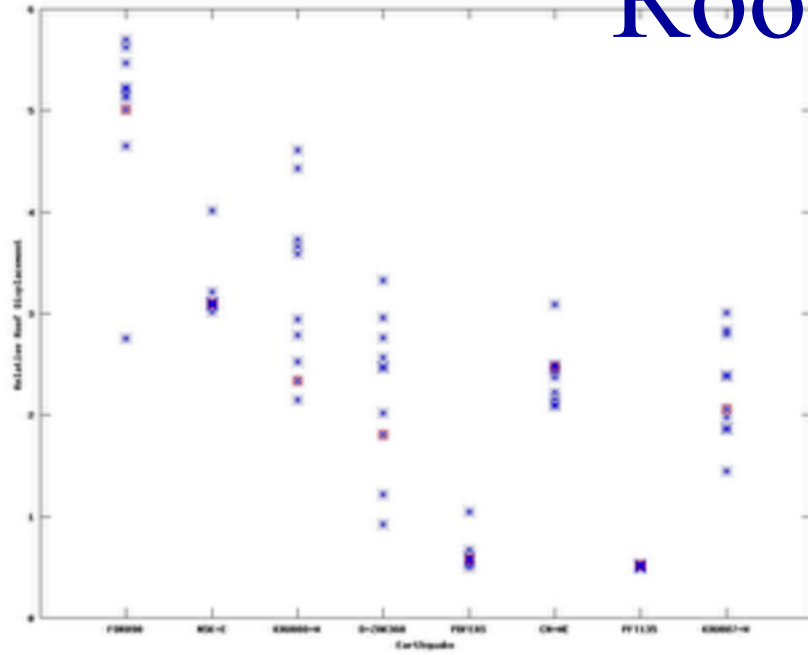
Height: AsBelow

Section: W14x211

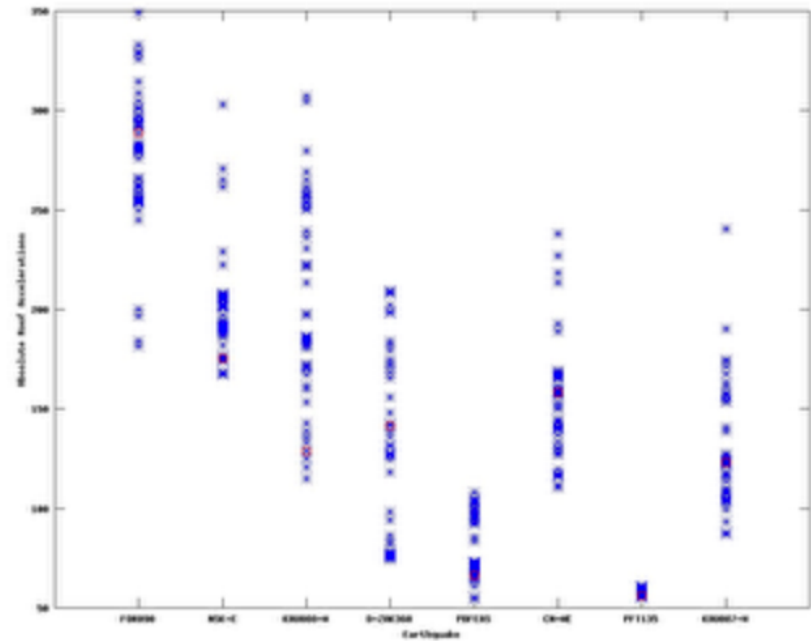
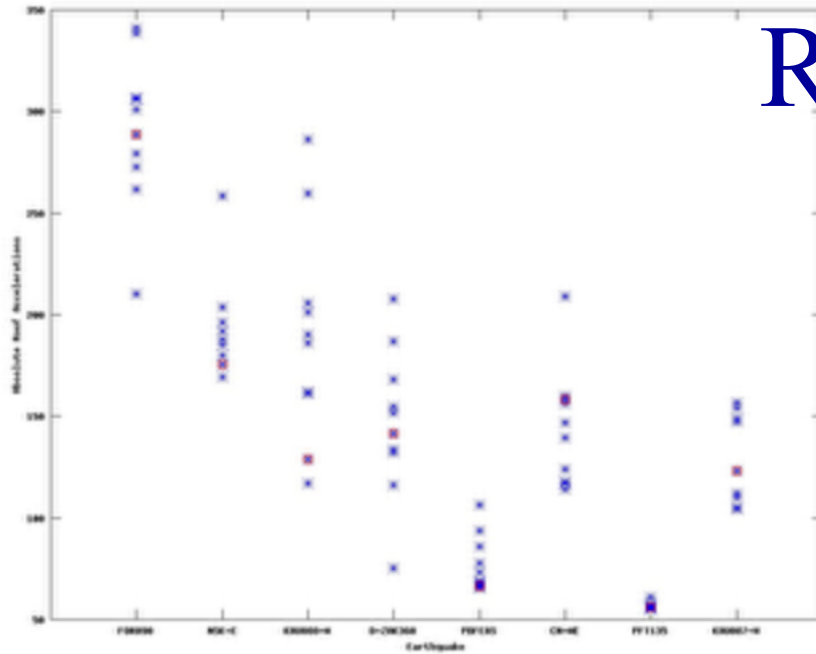
< Steel Properties

Floor Properties >

Roof Displacements



Roof Accelerations

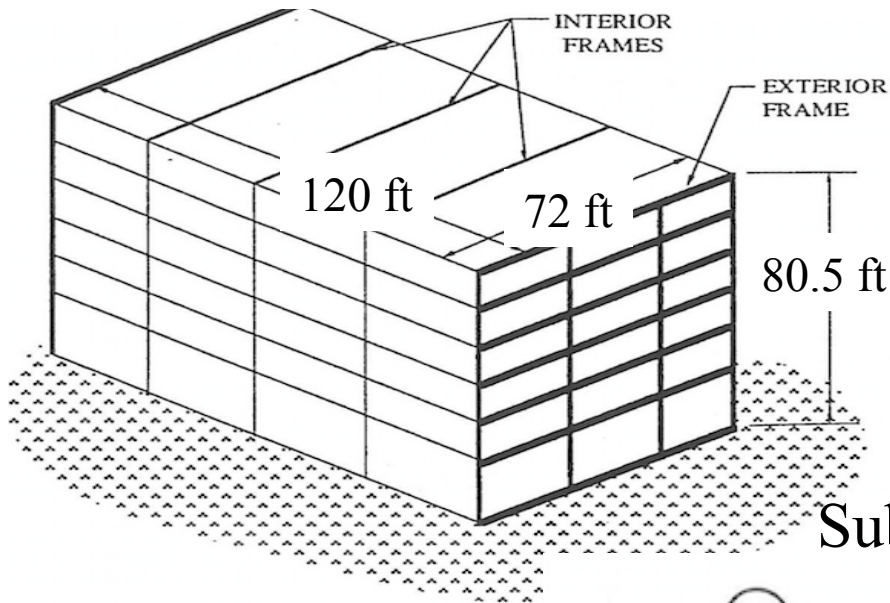


Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub

• Hands on Exercise

- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion



Max Roof Displacement = ?

Dead Load: 95psf typical, roof 80psf
 $E=29,000$, $F_y=50.0$, $b=0.003$

Subjected to el_centro 1940 N-S (Peknold)

	A	I _{xx}	ROOF	(A)	(B)	(C)	(D)	
W14X193	56.8	2400	6	W14 x 109	W24 x 104	W24 x 76	6th Story	12.5 ft
W14X159	46.7	1900	5			W24 x 76		
W14X109	32.0	1240	4		W27 x 146	W27 x 94	24 ft	
W30X173	51.0	8230	3			W27 x 94		
W27X146	43.1	5660	2		W30 x 173	W30 x 99	2nd Story	
W24X104	30.6	3100	1			W30 x 99		
W30X99	29.1	3990	G			1st Story	18 ft	
W27X94	27.7	3270						
W24X76	22.4	2100						

This figure is taken from the first paper in the SAC 95-05 report: Hall, John F. "Parameter Study of Response of Moment-Resisting Steel Frame Buildings to Near-Source Ground Motions"

Basic Linux Commands:

- **mkdir *dirname*** --- make a new directory
- **cd *dirname*** --- change directory
- **pwd** --- tells you where you currently are.
- **ls** --- list files in current directory
- **cp origFile newFile** --- copy a file
- **mv origFile newFile** --- move a file
- **rm filename** --- remove a file
- **wc -l filename** ---- how many lines in a file

To Get You Started – follow this:

- Start batchsubmit tool
- Type: **cd ~**
- On 1 single line type :
- **svn co svn://opensees.berkeley.edu/usr/local/svn/OpenSees/trunk/Workshops/OpenSeesDays Examples**
- This will have created Examples directory
- cd Examples and look at MRF1.tcl (spend some time looking at other files)
- Start OpenSeesLab tool and start OpenSeesInterpreter
- Go to editor, open MRF1.tcl and make changes

Outline of Workshop

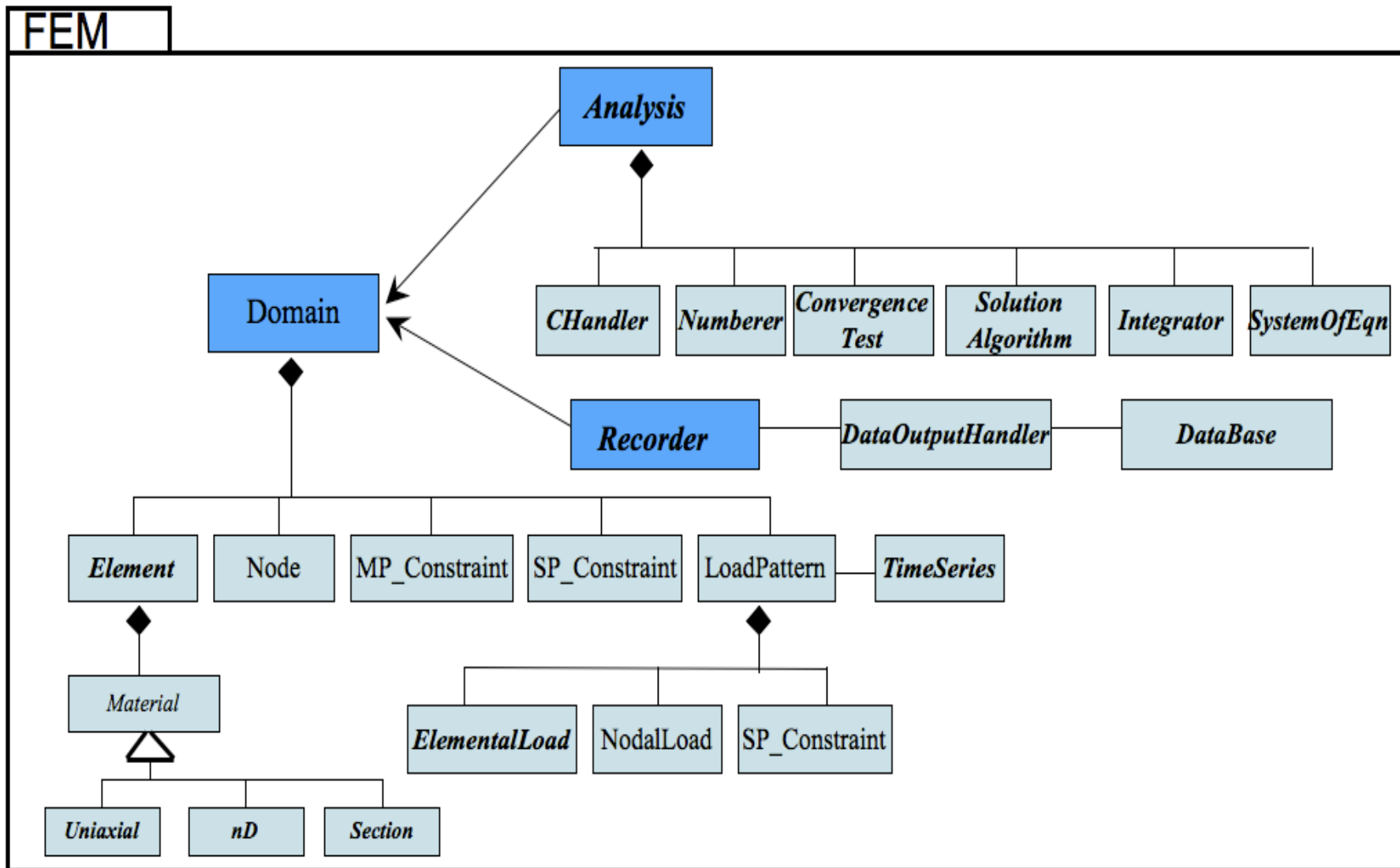
- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Traditional Finite Element Software

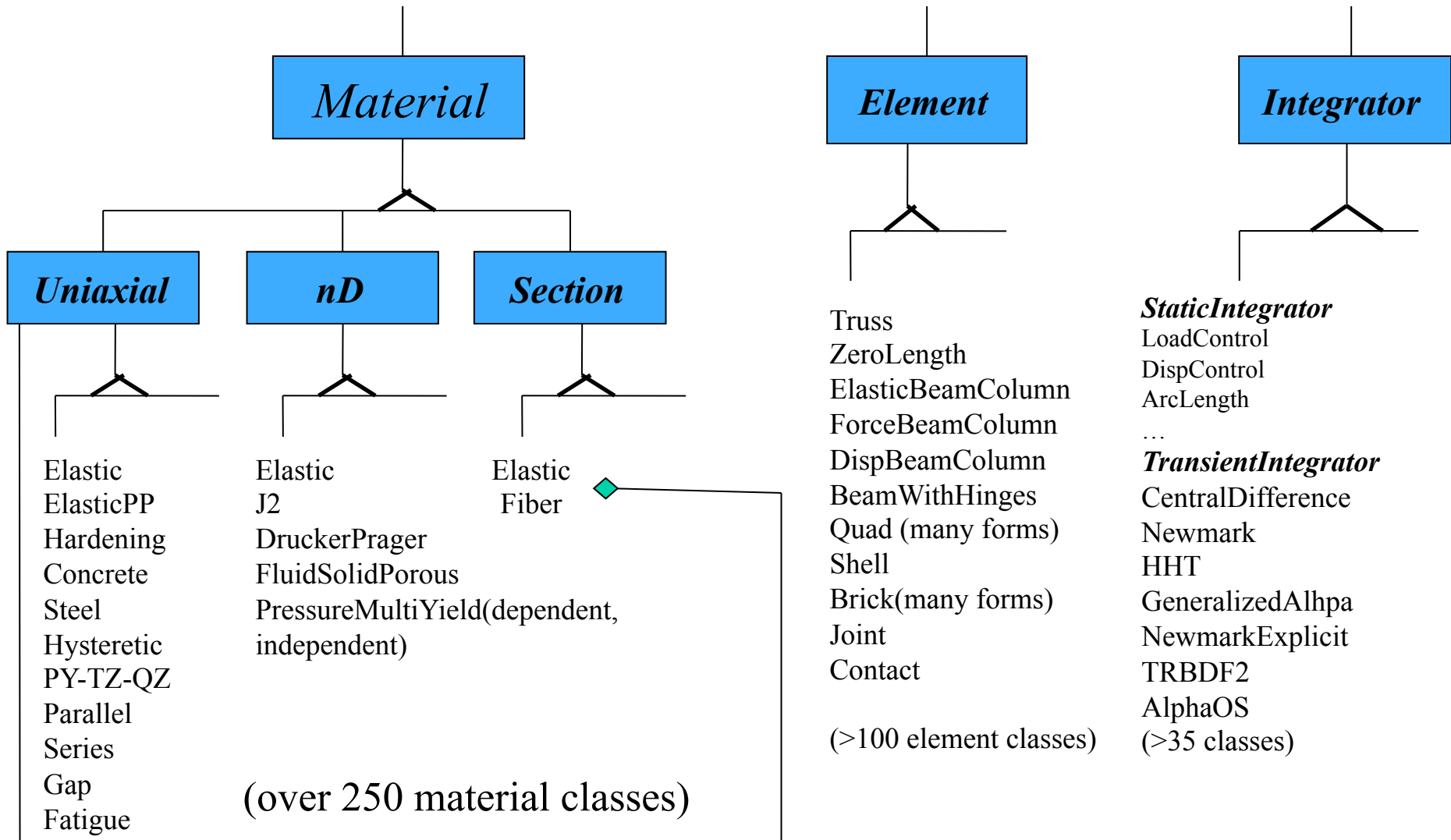
- Commercial fe codes **are large complex** software containing over one million lines of code. They **are closed-source** but **do allow new element and material routines to be added**. (at least the serious ones do)
- They are **slow to change** as hardware changes and they do **not allow researchers to play with other equally important aspects of the code**. They **do not promote the active sharing of new code**.

**THE ADVANTAGE OF A
SOFTWARE FRAMEWORK
SUCH AS OPENSEES IS THAT
YOU DON'T HAVE TO
UNDERSTAND ALL OF IT TO
BUILD APPLICATIONS OR
MAKE CONTRIBUTIONS
YOU JUST NEED TO
UNDERSTAND THAT PART
THAT CONCERNS YOU**

OPENSEES FEM ABSTRACT CLASSES

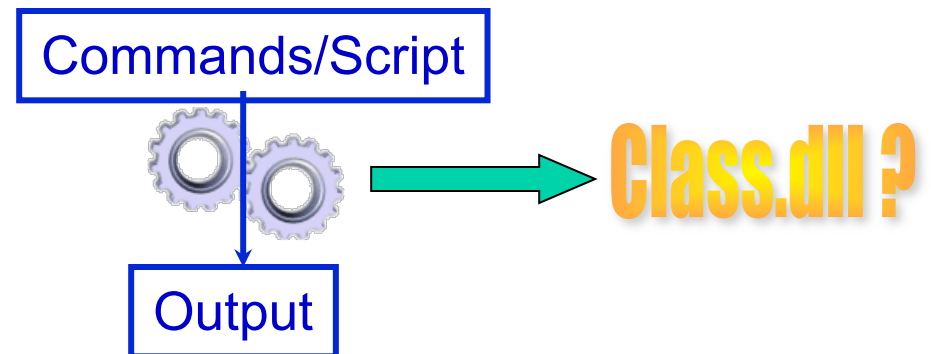


Framework Contains both Abstract Classes & Concrete Subclasses



Unknown Class Type

When OpenSees.exe is running and it comes across a class type it knows nothing about **BEFORE GIVING AN ERROR IT WILL TRY AND LOAD A LIBRARY OF THAT CLASSES NAME**. If it cannot find a **library of the appropriate name** or the **procedure of the appropriate name** in the library it will **FAIL**.



Command Prompt

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\PEER Center>cd DEVELOPER
C:\Users\PEER Center\DEVELOPER>cd material
C:\Users\PEER Center\DEVELOPER\material>cd cpp
C:\Users\PEER Center\DEVELOPER\material\cpp>OpenSees Example1.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.2

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      <Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html>

WARNING could not create uniaxialMaterial ElasticPPcpp
      while executing
"uniaxialMaterial ElasticPPcpp 1 3000 0.001"
      (file "Example1.tcl" line 21)

C:\Users\PEER Center\DEVELOPER\material\cpp>
```

To Add a New Class you must:

- 1) Provide Code that meets the Interface of the appropriate super-class
- 2) you must BUILD THE LIBRARY
- 3) make it accessible to the program.

**WHILE C++ IS THE GLUE LANGUAGE
THAT HOLDS OPENSEES TOGETHER**

**YOUR CODE DOES NOT HAVE TO BE
WRITTEN IN C++.**

**C and FORTRAN OPTIONS ARE ALSO
AVAILABLE**

UniaxialMaterial Interface

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

    virtual int setTrialStrain(double strain, double strainRate = 0.0) = 0;
    virtual double getStrain(void) = 0;
    virtual double getStress(void) = 0;
    virtual double getTangent(void) = 0;
    virtual double getInitialtangent(void) = 0;

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

    virtual UniaxialMaterial *getCopy(void) = 0;

    virtual Response *setResponse(const char **argv, int argc, OPS_Stream &theOutput);
    virtual int getResponse(int responseID, Information &info);
    virtual void Print(OPS_Stream &s, int flag = 0);

    virtual int sendSelf(int commitTag, Channel &theChannel)=0;
    virtual int recvSelf(int commitTag, Chanel &theChannel, FEM_ObjectBroker &theBroker)=0;
    // THERE ARE SOME OTHERS .. BUT THESE ARE PURE VIRTUAL ONES THAT MUST BE PROVIDED
protected:
private:
};
```

Must be overridden by subclass, “pure virtual”

Can be overridden by subclass

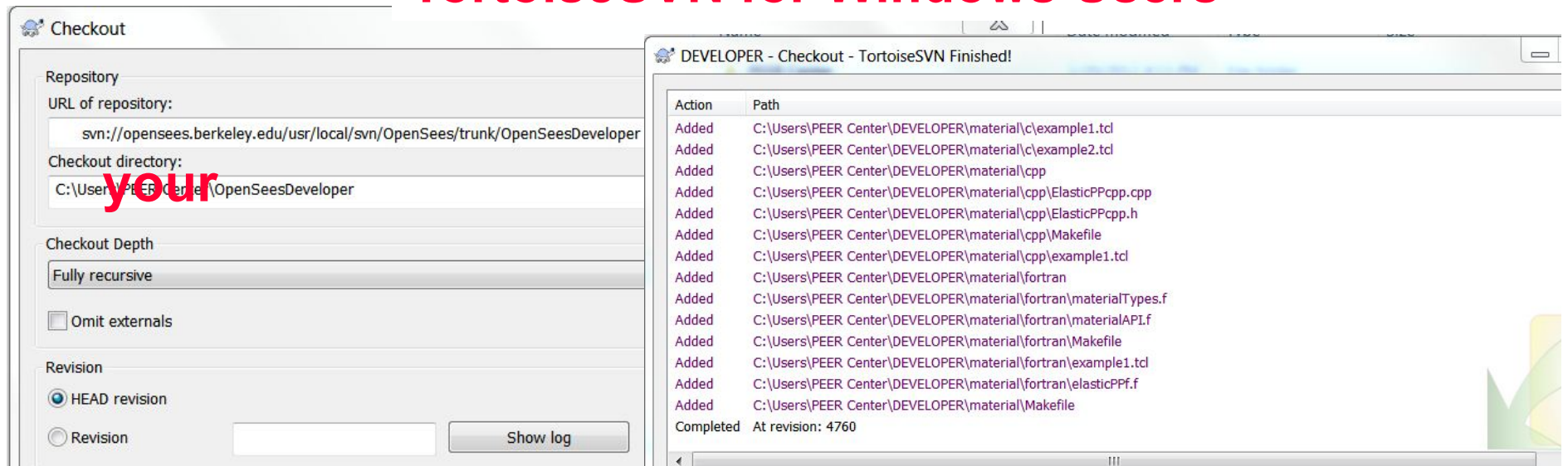
Adding New Code

For those new to Programming **NEVER EVER NEVER START WITH AN EMPTY FILE** .. TAKE SOMETHING SIMILAR THAT WORKS AND MAKE CHANGES TO THAT FILE.

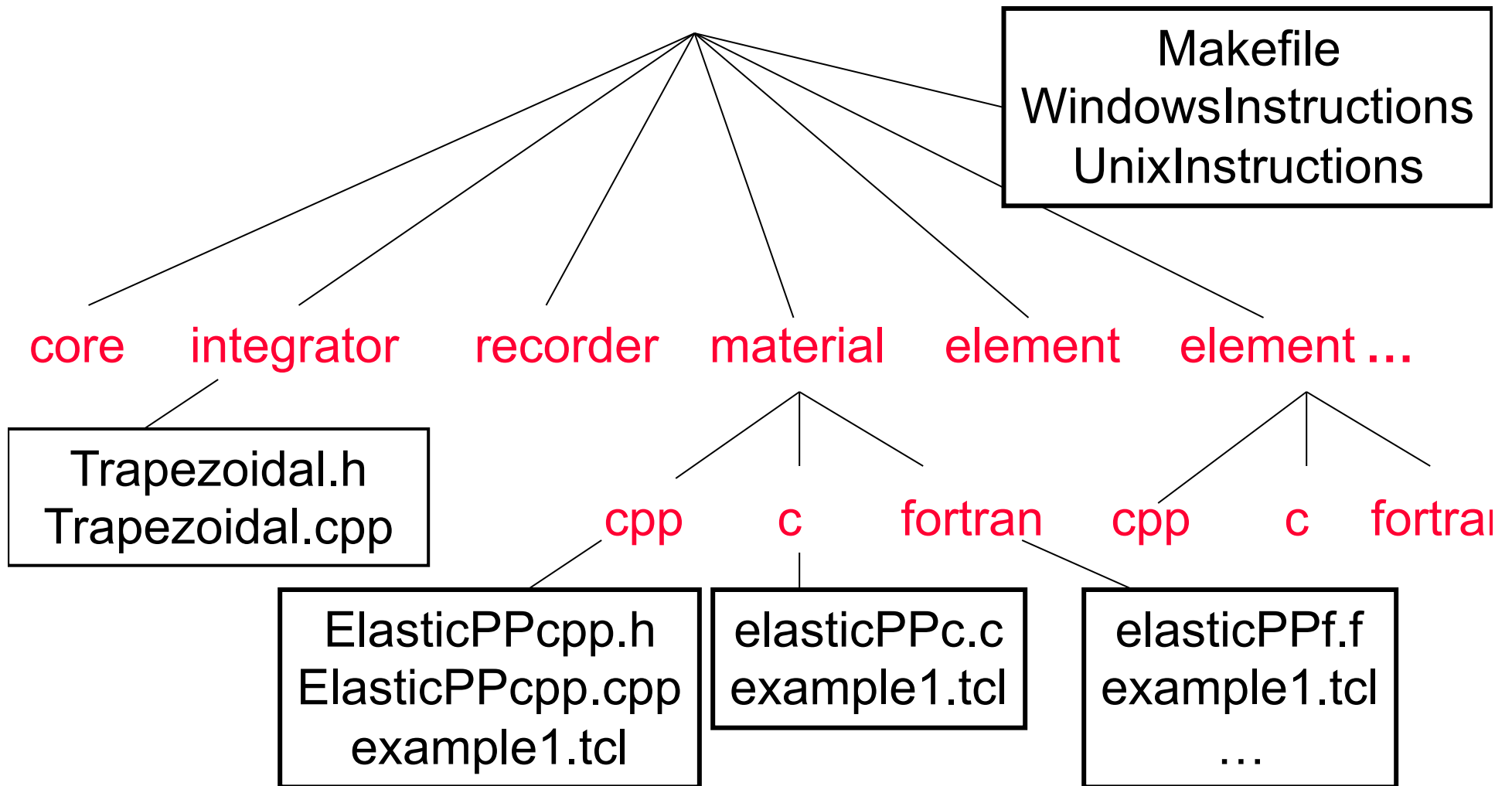
We provide C++, C and Fortran examples on-line using svn

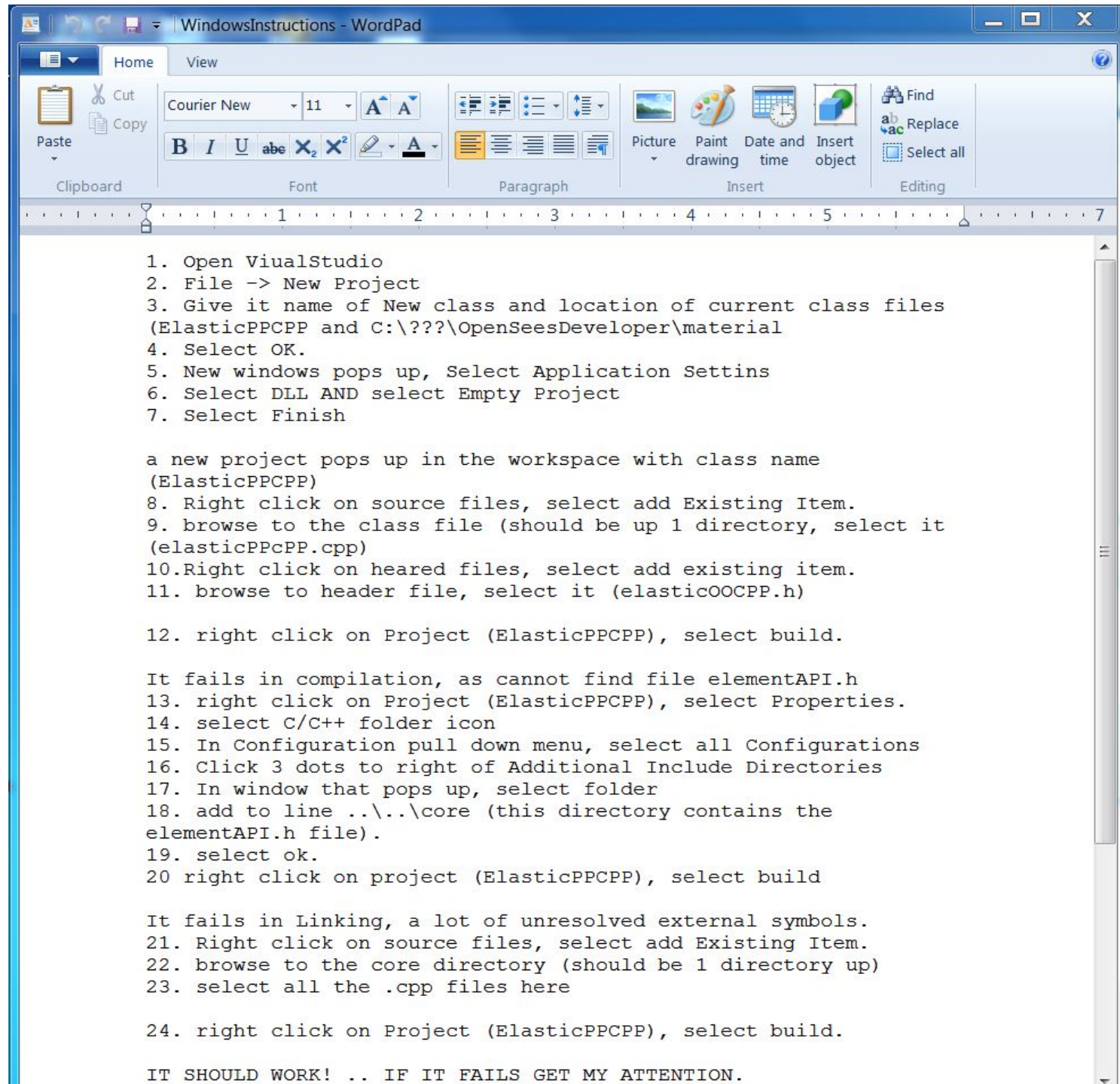
svn://opensees.berkeley.edu/usr/local/svn/OpenSees/trunk/DEVELOPER

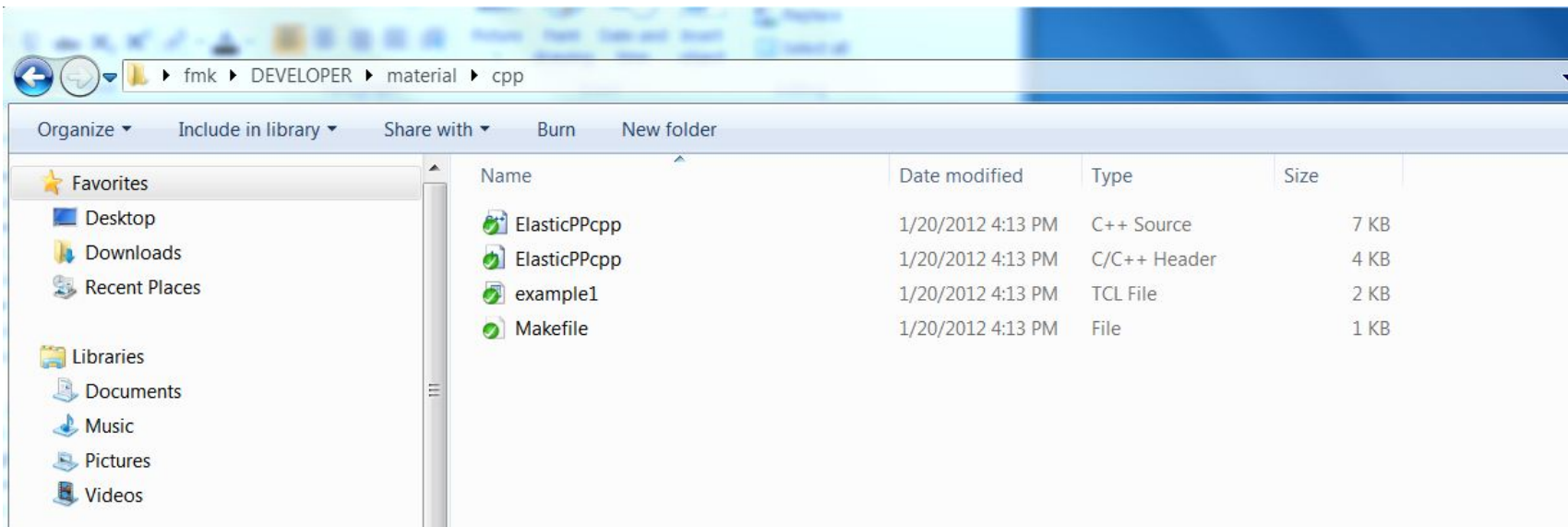
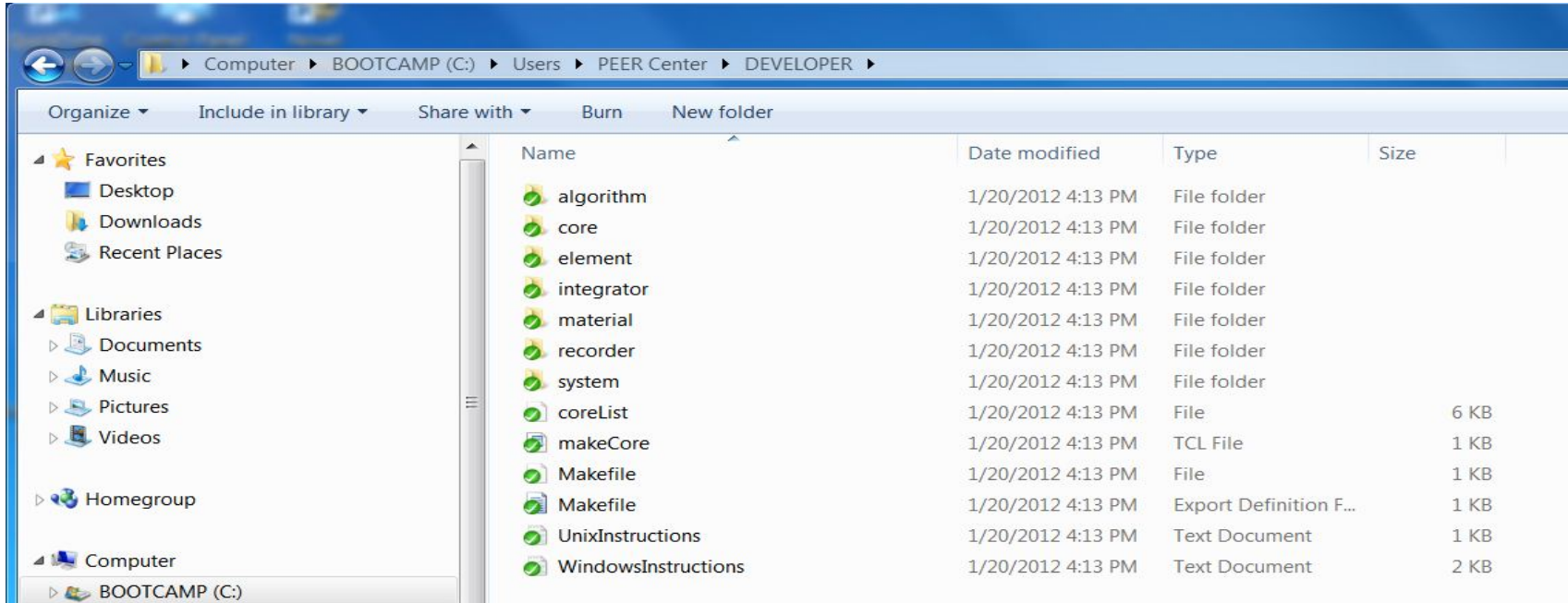
TortoiseSVN for Windows Users



Source Code Tree in DEVELOPER







Interface (.h file)

Material Name

```
class ElasticPPcpp: public UniaxialMaterial {
    public:
        ElasticPPcpp(int tag, double e, double eyp);
        ElasticPPcpp();
        ~ElasticPPcpp();
        int setTrialStrain(double strain, double strainRate=0.0);
        double getStrain(void);
        double getStress(void);
        double getTangent(void);
        double getInitialTangent(void);

        int commitState(void);
        int revertToLastCommit(void);
        int revertToStart(void);
        UniaxialMaterial *getCopy(void);
        int sendSelf(int commitTag, Channel &theChannel);
        int recvSelf(int commitTag, Channel &theChannel, FEM_ObjectBroker &theBroker);
        void Print(OPS_Stream &s, int flag = 0);
    private:
        double fyp, fyn; // pos & neg yield stress
        double ezero, E, ep; // init strain, elastic mod
        double ep; // plastic strain at yield point
        double trialStrain, trialStress, trialTangent;
        double commitStrain, commitStress, commitTangent;
};
```

material input properties

data unique to material includes:
Material parameters, & State variables

Implementation (.cpp file)

```
ElasticPPcpp::ElasticPPcpp(int tag, double e, double eyp)
    :UniaxialMaterial(tag, 0),
ezero(0), E(e), ep(0.0) trialStrain(0.0),trialStress(0.0),trialTangent(e),
    commitStrain(0.0),commitStress(0.0),commitTangent(e)
    {
        fyp=E*eyp;
        fyn = -fyp;
    }

ElasticPPcpp::ElasticPPcpp()
    :UniaxialMaterial(tag, 0)
    fyp(0),fyn(0),ezero(0), E(0),ep(0),
    trialStrain(0.0),trialStress(0.0),trialTangent(0),
    commitStrain(0.0),commitStress(0.0),commitTangent(e){
    }

ElasticPPcpp::~~ElasticPPcpp
    {
        // does nothing .. No memory to clean up
    }

    UniaxialMaterial *ElasticPPcpp::getCopy(void)
    {
        ElasticPPcpp *theCopy = new ElasticPPcpp(this->getTag(), E, fyp/E);
        return theCopy;
    };
```

Hardest Method to Write

Implementation

```
ElasticPPcpp::setTrialStrain(double strain, double strainRate)
{
    if (fabs(trialStrain - strain) < DBL_EPSILON)
        return 0;
    trialStrain = strain;
    double sigtrial; // trial stress
    double f; // yield function
    // compute trial stress
    sigtrial = E * ( trialStrain - ezero - ep );
    // evaluate yield function
    if ( sigtrial >= 0.0 )
        f = sigtrial - fyp;
    else
        f = -sigtrial + fyn;
    double fYieldSurface = - E * DBL_EPSILON;
    if ( f <= fYieldSurface ) {
        // elastic
        trialStress = sigtrial;
        trialTangent = E;
    } else {
        // plastic
        if ( sigtrial > 0.0 ) {
            trialStress = fyp;
        } else {
            trialStress = fyn;
        }
        trialTangent = 0.0;
    }

    return 0;
}
```


Implementation

```
double
ElasticPPcpp::getStrain(void)
{
    return trialStrain;
}

double
ElasticPPcpp::getStress(void)
{
    return trialStress;
}

double
ElasticPPcpp::getTangent(void)
{
    return trialTangent;
}

int
ElasticPPcpp::revertToLastCommit(void)
{
    trialStrain = commitStrain;
    trialTangent = commitTangent;
    trialStress = commitStress;

    return 0;
}
```

Interpreter looking for this function in lib

```
OPS_ELASTICPP ^
if (num
opserr << "ElasticPPcpp unaxi
num
```

You can give yourself some KUDOS, e.g. please reference ... if you used this

```
// Pointer to a uniaxial material that will be returned
```

```
UniaxialMaterial *theMaterial = 0;
```

```
int iData[1];
```

```
double d
```

```
int num
```

parse the script for three material parameters

```
numData
```

```
if (OPS_GetIntInput(&numData, iData) != 0) {
```

```
opserr << "WARNING invalid uniaxialMaterial ElasticPP tag" << endl;
```

```
return 0;
```

```
}
```

```
numData = 2;
```

```
if (OPS_GetDoubleInput(&numData, dData) != 0) {
```

```
opserr << "WARNING invalid E & ep\n";
```

```
return 0;
```

Function returns new material

```
theMaterial = new ElasticPPcpp(iData[0], dData[0], dData[1]);
```

```
return theMaterial;
```

```
}
```


C & Fortran Procedural Languages Can Also Be Used

```

OPS_Export void
elasticPPc (matObj *t
modelState *r
double *str
double *ta
double *str
int *isw,
int *result
{
*result = 0;

if (*isw == ISW_IN

double dData[4]
int iData[1]

/* get the input data - tag
int numData =
OPS_GetIntInput(&numData)
numData = 2
OPS_GetDoubleInput(&numData)

/* Allocate the element state
thisObj%state = iData

```

```

SUBROUTINE ELASTICPPF(matObj,model,strain,tang,stress

!DEC$ IF DEFINED (_DLL)
!DEC$ ATTRIBUTES DLLEXPORT :: ELASTICPPF
!DEC$ END IF
use materialTypes
use materialAPI
implicit none
IF (isw.eq.ISW_INIT) THEN

c   get the input data - tag? E? eyp?

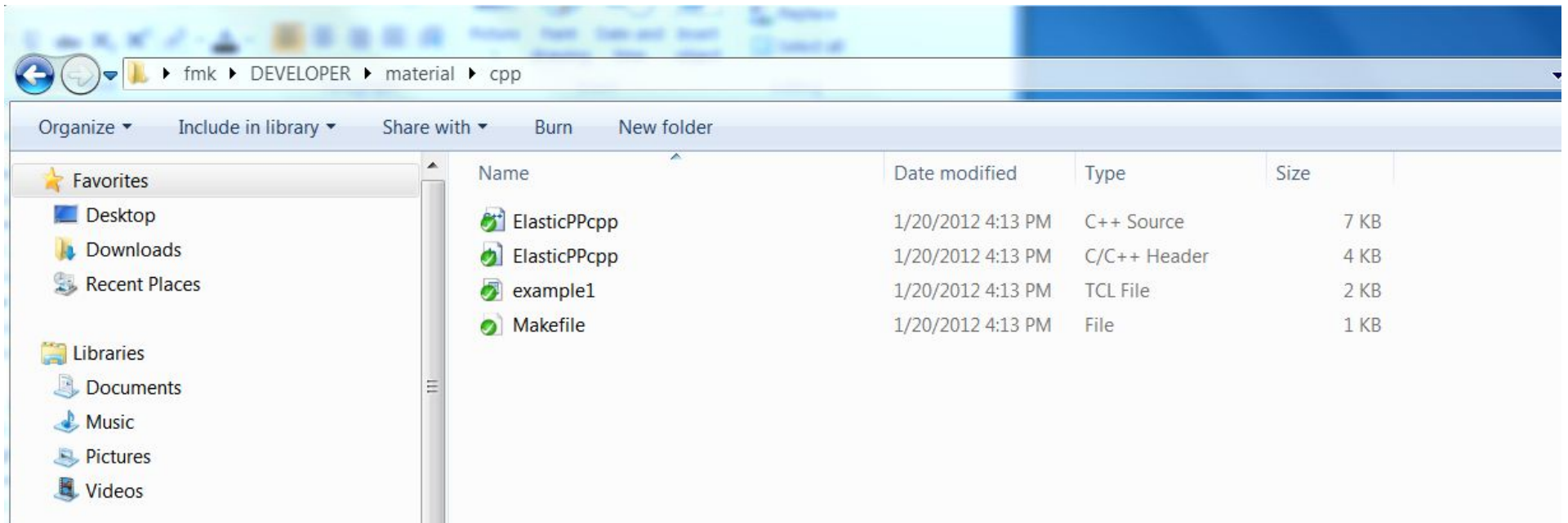
        numData = 1
        iPtr=>iData;
err = OPS_GetIntInput(numData, iPtr)
        numData = 2
        dPtr=>dData;
err = OPS_GetDoubleInput(numData, dPtr)

c   Allocate the element state
        matObj%tag = idata(1)
        matObj%nparam = 2

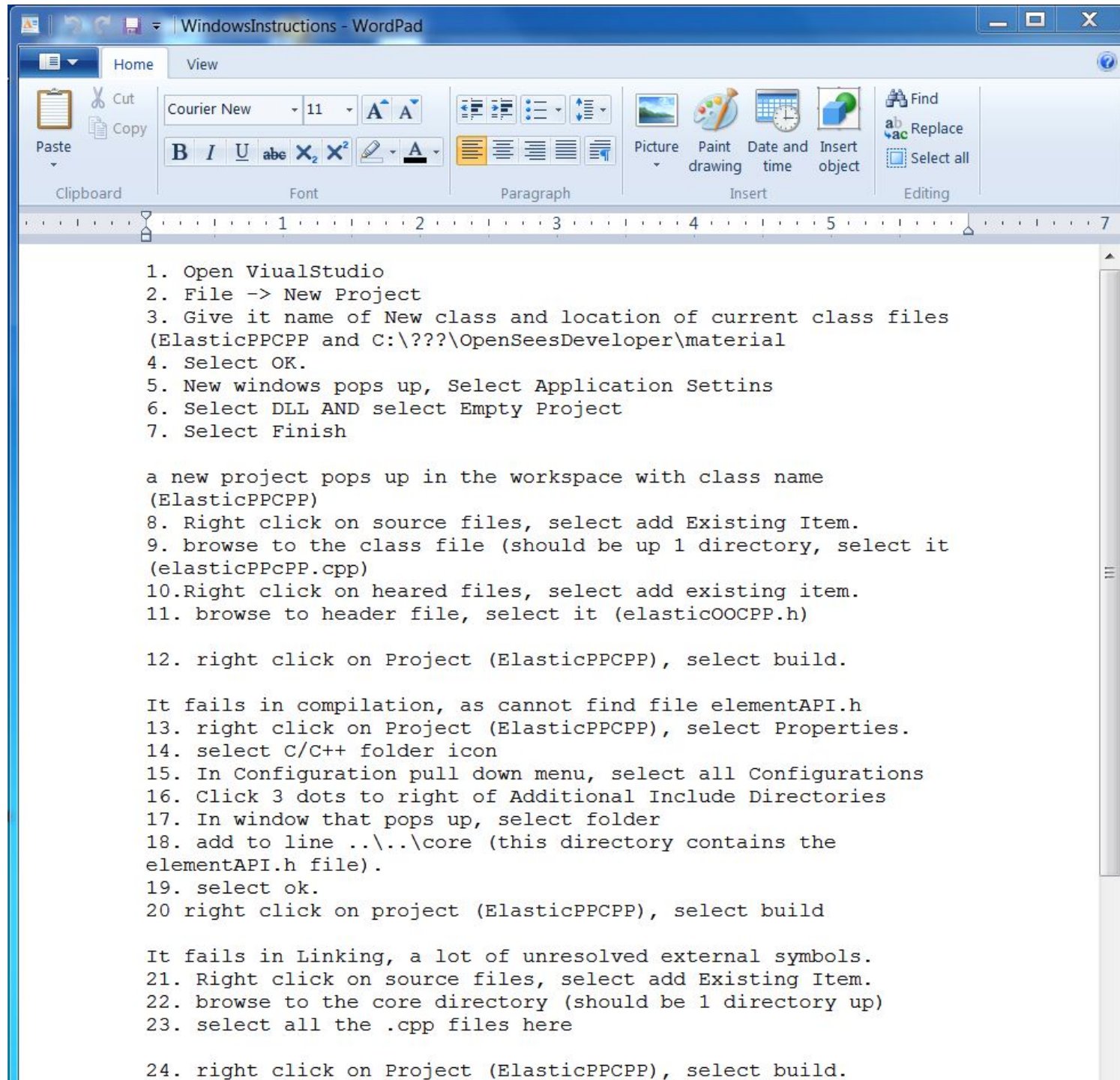
```

**What Follows are the Steps
required to Build
ElasticPPcpp.dll on a
Windows Machine with
Visual Studio 2010 Installed**

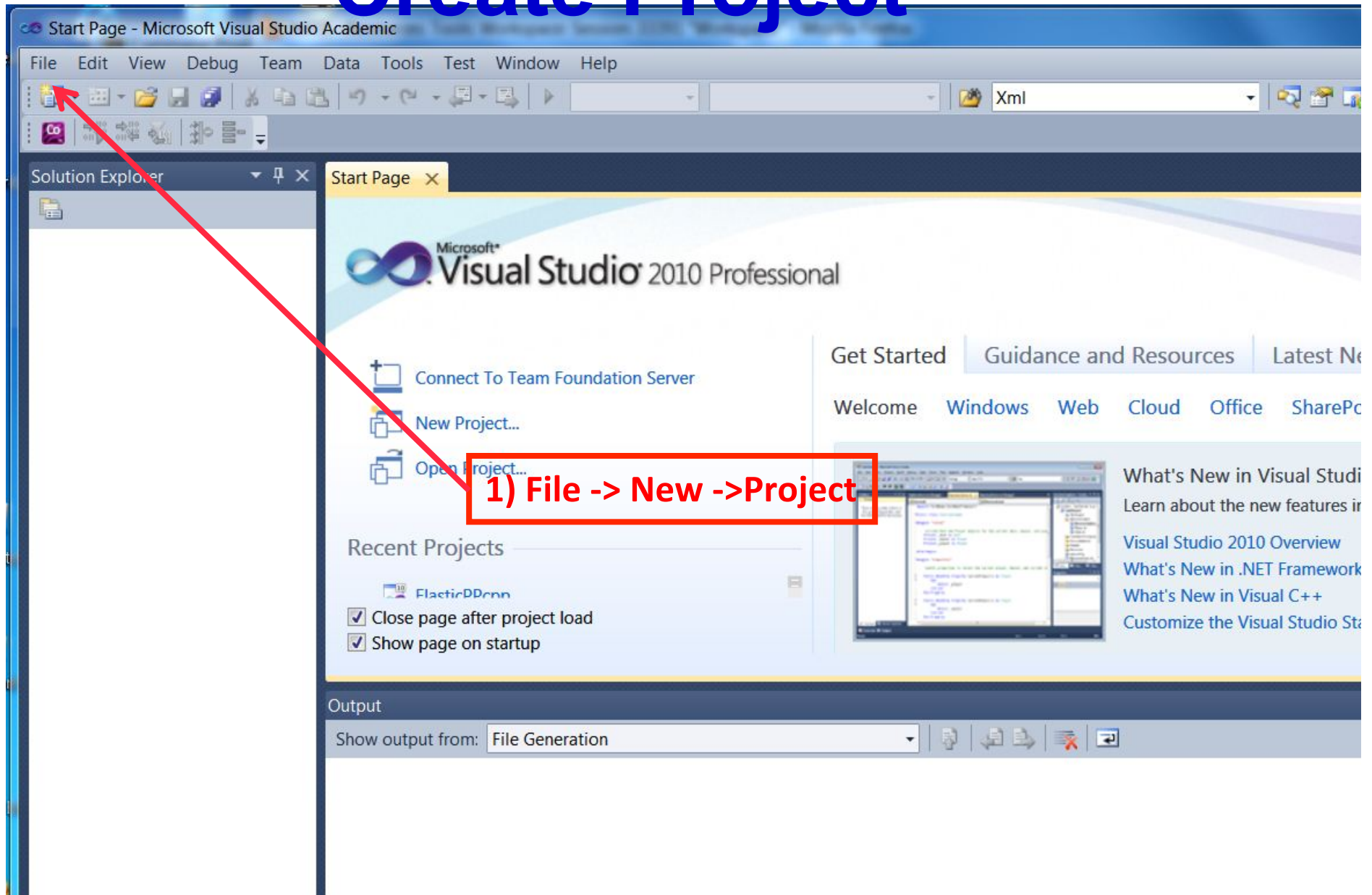
We Will Build the ElasticPPcpp.dll

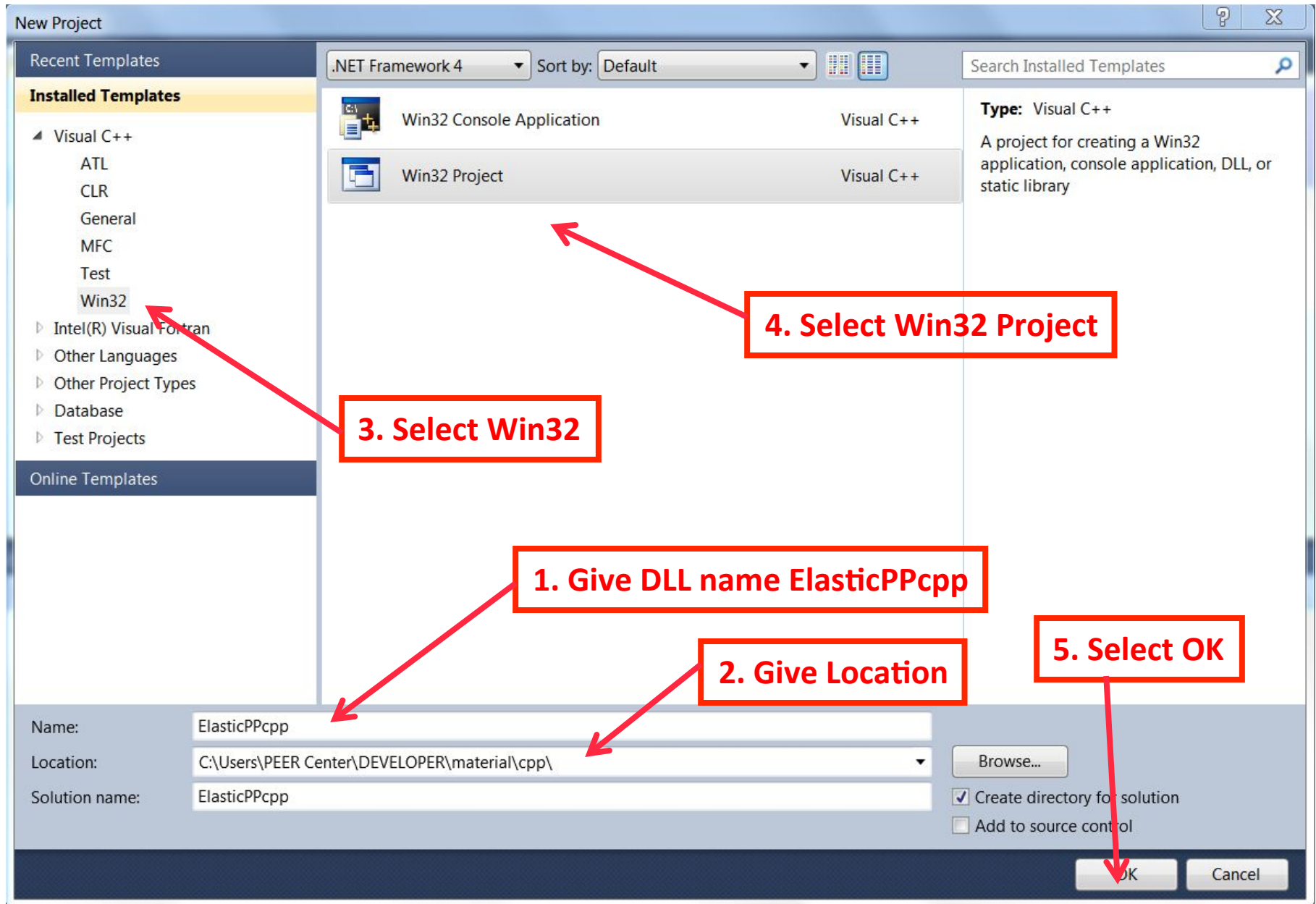


Source code and example are in
`/DEVELOPER/material/cpp`



Create Project







Welcome to the Win32 Application Wizard

Overview

Application Settings

These are the current project settings:

- Windows application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.



Select Application Settings

< Previous Next > Finish Cancel



Application Settings

Overview

Application Settings

Application type:

- Windows application
- Console application
- DLL
- Static library

Add common header files for:

- ATL
- MFC

Additional options:

- Empty project
- Export symbols
- Precompiled header

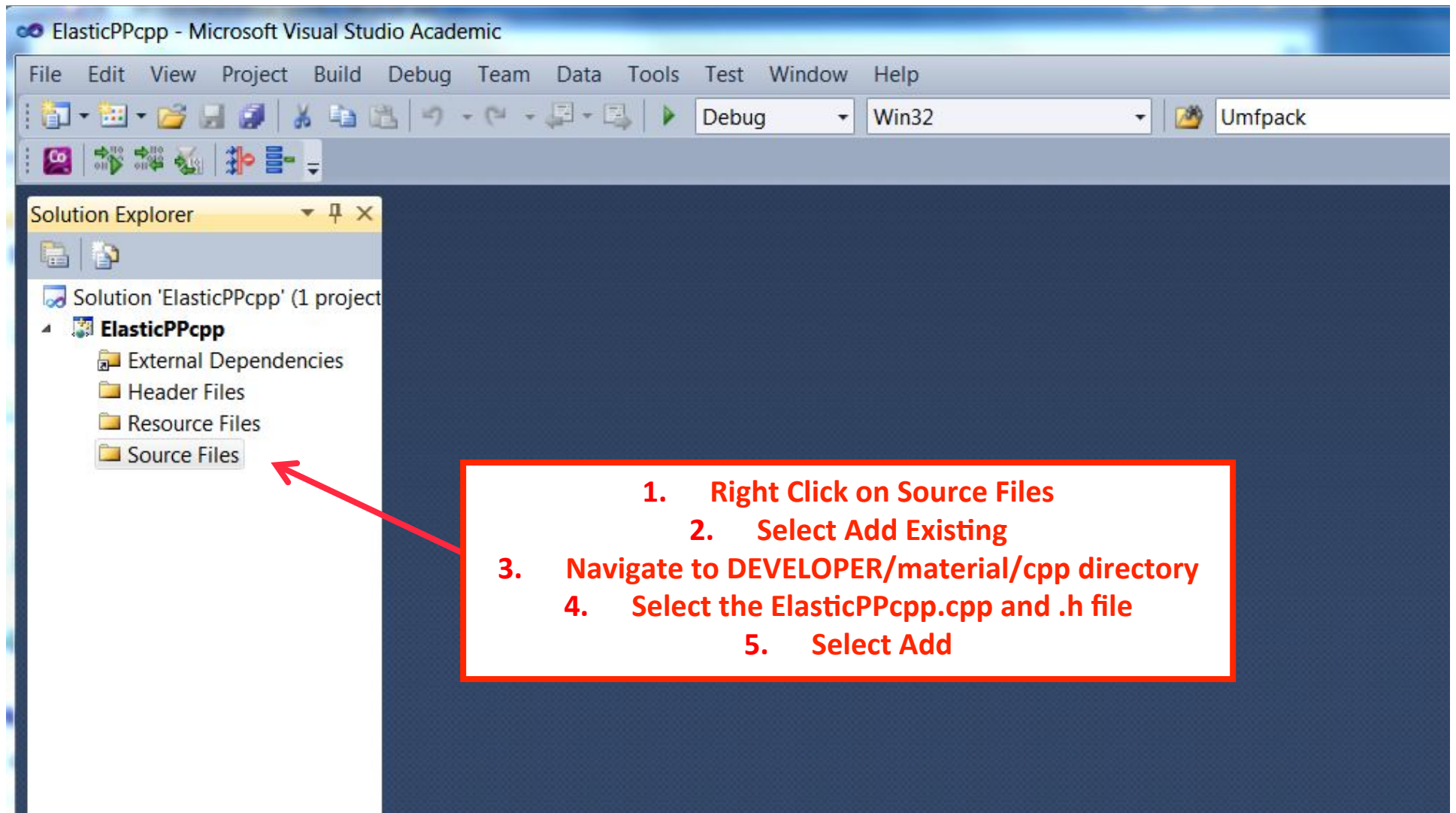
1. Select DLL

2. Select DLL

3. Select Finish

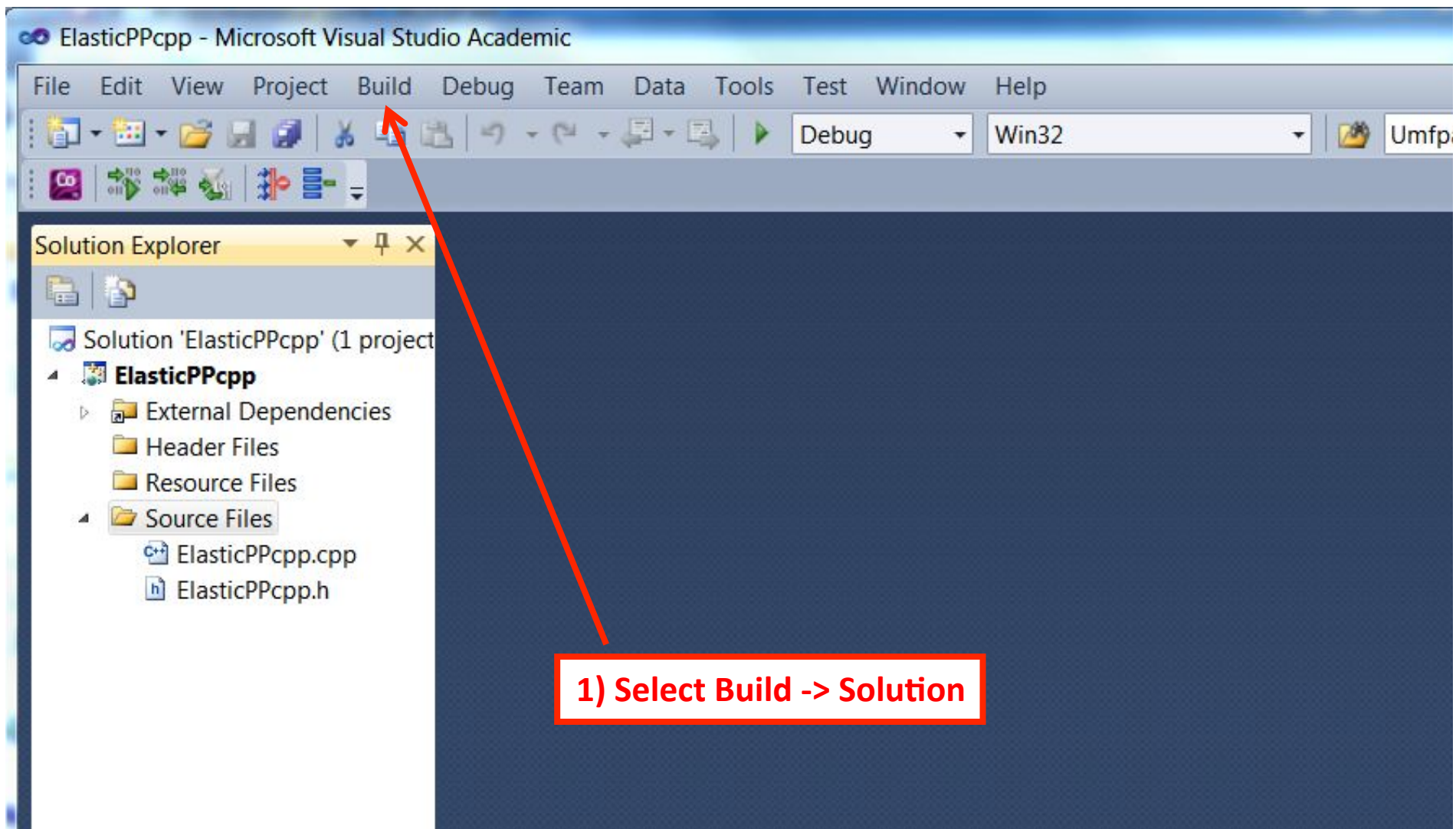
< Previous Next > Finish Cancel

Add Files To Project



The screenshot shows the Microsoft Visual Studio Academic interface. The Solution Explorer on the left displays the project structure for 'ElasticPPcpp', including folders for 'External Dependencies', 'Header Files', 'Resource Files', and 'Source Files'. A red arrow points from a text box to the 'Source Files' folder.

1. Right Click on Source Files
2. Select Add Existing
3. Navigate to DEVELOPER/material/cpp directory
4. Select the ElasticPPcpp.cpp and .h file
5. Select Add



1) Select Build -> Solution

ElasticPPcpp - Microsoft Visual Studio Academic

File Edit View Project Build Debug Team Data Tools Test Window Help

Debug Win32 Umfpack

Solution Explorer

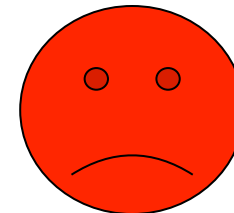
- Solution 'ElasticPPcpp' (1 project)
 - ElasticPPcpp**
 - External Dependencies
 - Header Files
 - Resource Files
 - Source Files
 - ElasticPPcpp.cpp
 - ElasticPPcpp.h

1. Right Click on ElasticPPcpp Project
2. Select Properties
3. Select C/C++

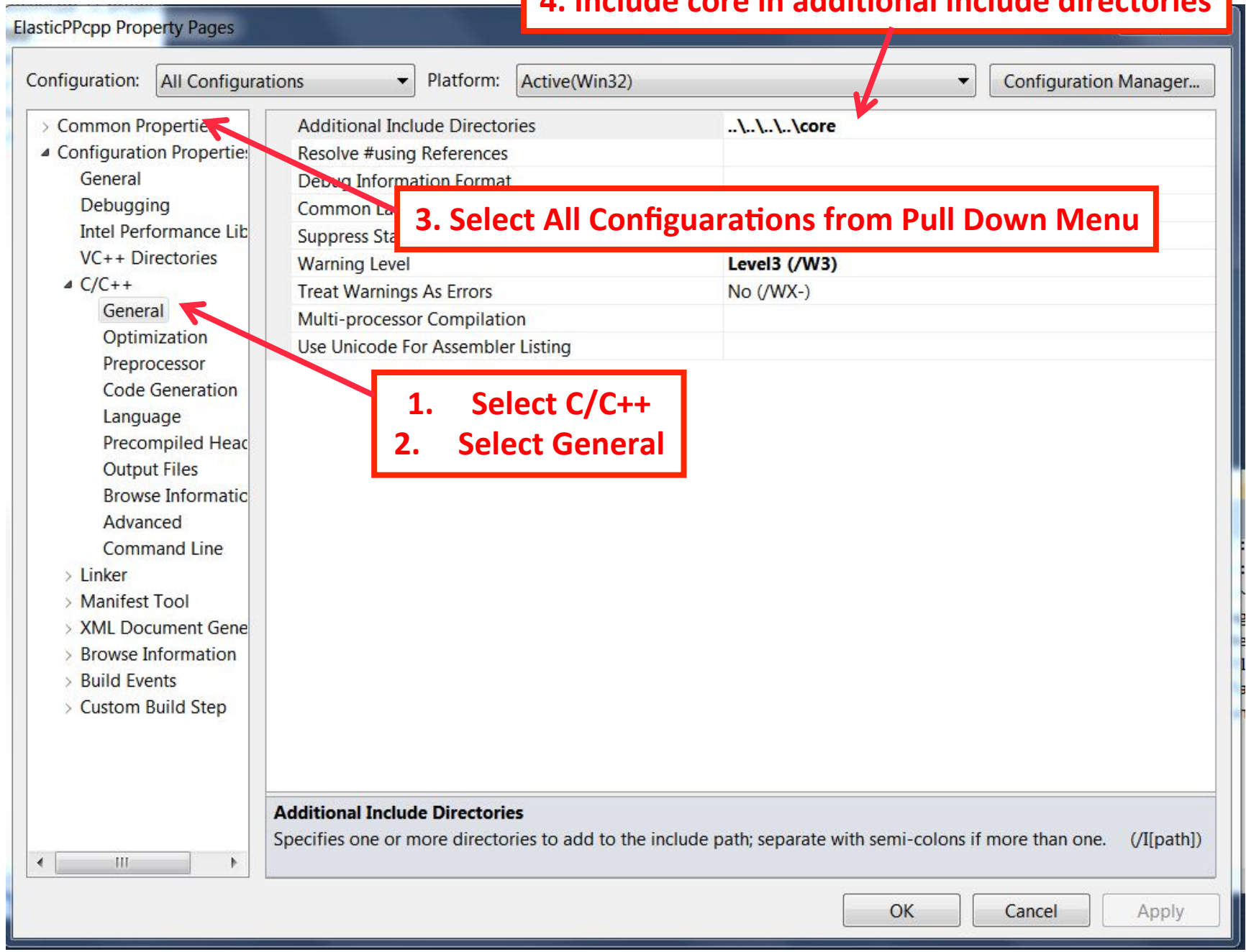
Output

Show output from: Build

```
1>Build started 1/20/2012 4:33:02 PM.
1>PrepareForBuild:
1> Creating directory "C:\Users\PEER Center\DEVELOPER\material\cpp\ElasticPPcpp\Debug\".
1>InitializeBuildStatus:
1> Creating "Debug\ElasticPPcpp.unsuccessfulbuild" because "AlwaysCreate" was specified.
1>ClCompile:
1> ElasticPPcpp.cpp
1>c:\users\peer center\developer\material\cpp\elasticppcpp.cpp(32): fatal error C1083: Cannot open include file: 'elementAPI.h': No such
1>
1>Build FAILED.
1>
1>Time Elapsed 00:00:00.55
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====
```



IT FAILS!



ElasticPPcpp - Microsoft Visual Studio Academic

File Edit View Project Build Debug Team Data Tools Test Window Help

Debug Win32 Umfpack

Solution Explorer

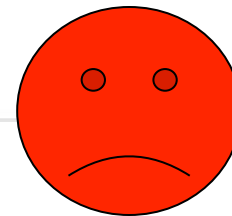
- Solution 'ElasticPPcpp' (1 project)
- ElasticPPcpp**
- External Dependencies
- Header Files
- Resource Files
- Source Files

1) Select Build -> Solution

Output

Show output from: Build

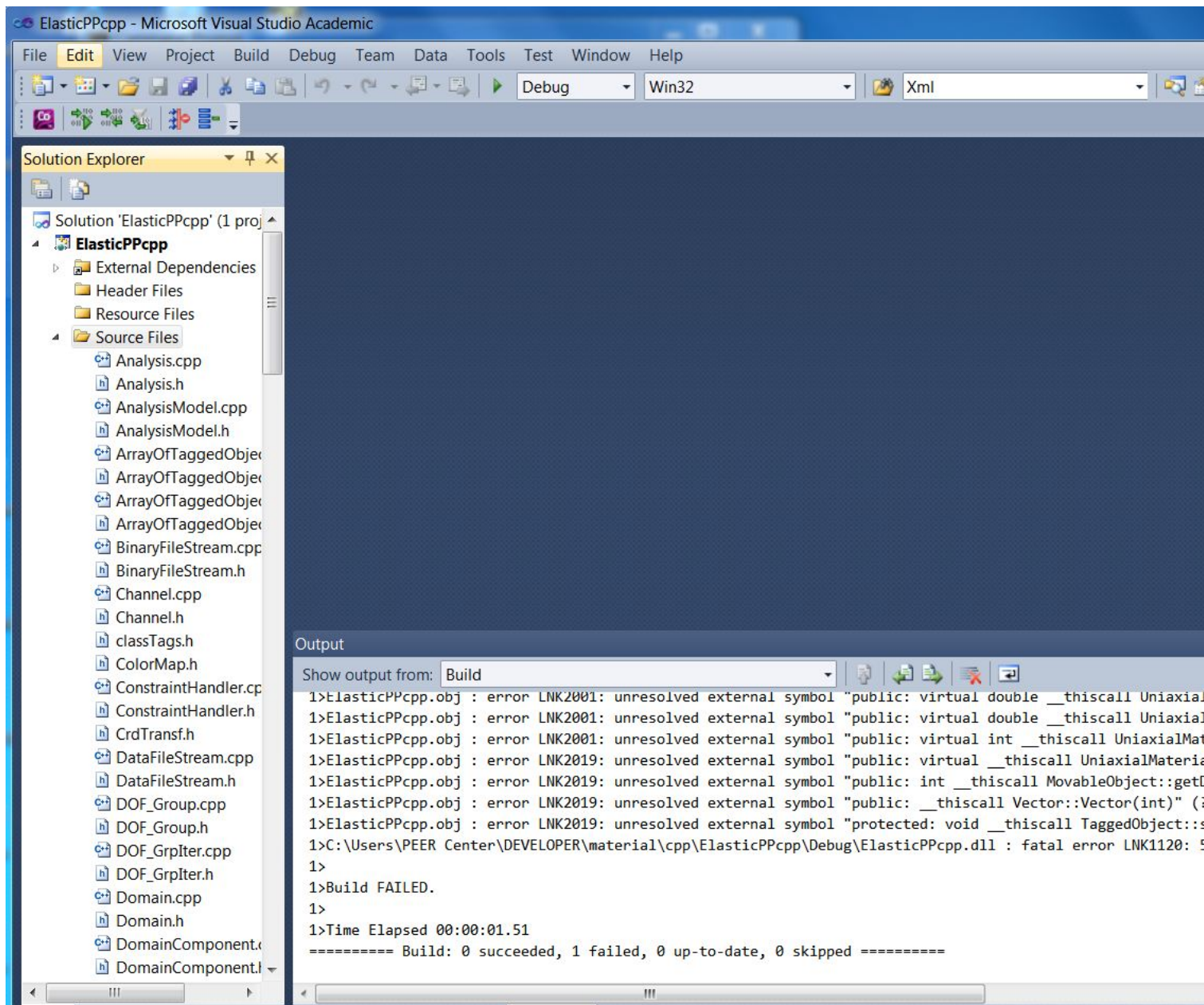
```
1>ElasticPPcpp.obj : error LNK2001: unresolved external symbol "public: virtual int __thiscall MovableObject::setVariable(char const *,class Information &
1>ElasticPPcpp.obj : error LNK2001: unresolved external symbol "public: virtual int __thiscall MovableObject::getVariable(char const *,class Information &
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: virtual __thiscall UniaxialMaterial::~UniaxialMaterial(void)" (??1UniaxialMaterial
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: int __thiscall MovableObject::getDbTag(void)const " (?getDbTag@MovableObject@@QBEH
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: __thiscall Vector::Vector(int)" (??0Vector@@QAE@H@Z) referenced in function "publi
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: __thiscall Vector::~Vector(void)" (??1Vector@@QAE@XZ) referenced in function "void
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "protected: void __thiscall TaggedObject::setTag(int)" (?setTag@TaggedObject@@IAEXH@Z) refe
1>C:\Users\PEER Center\DEVELOPER\material\cpp\ElasticPPcpp\Debug\ElasticPPcpp.dll : fatal error LNK1120: 31 unresolved externals
1>
1>Build FAILED.
1>
1>Time Elapsed 00:00:00.27
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====
```

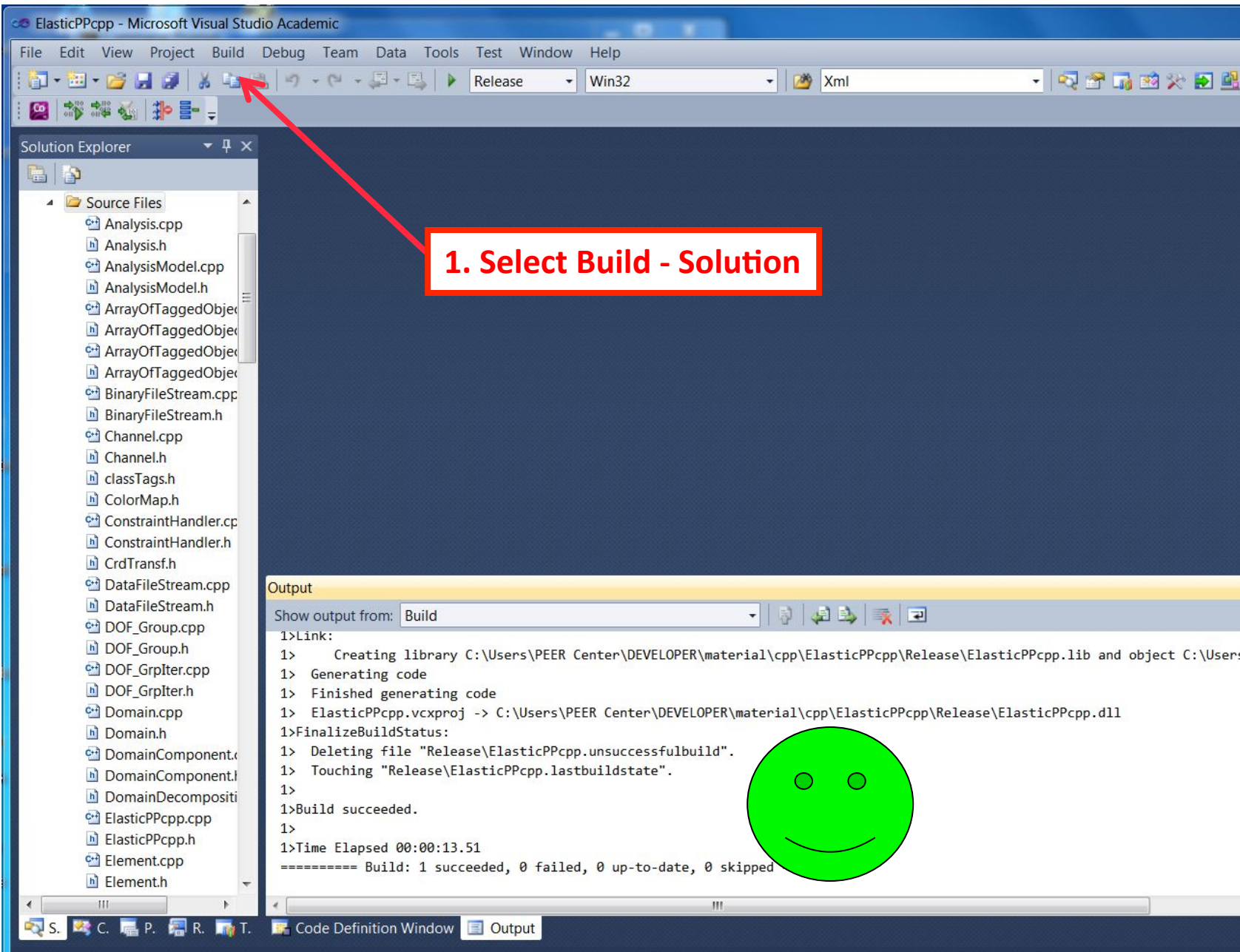


IT FAILS!

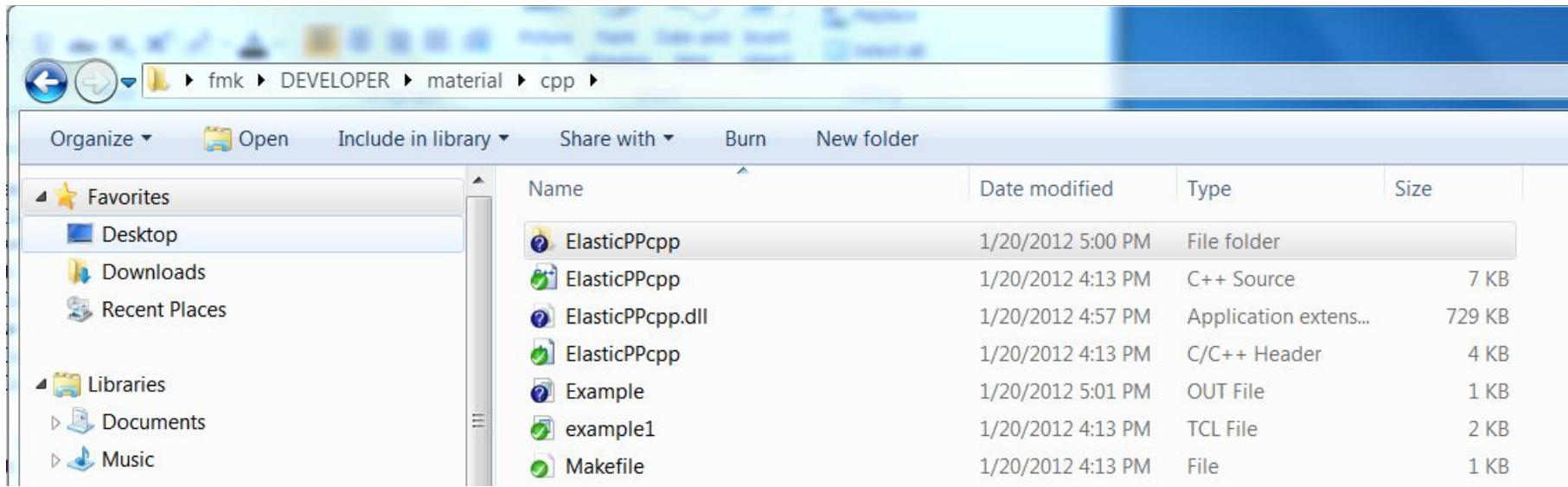
1. Right Click on Source Files
2. Select Add Existing
3. Navigate to DEVELOPER/core directory
4. Select the All .cpp and .h file
5. Select Add

```
1>Build FAILED.  
1>  
1>Time Elapsed 00:00:00.74  
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====
```





Copy ElasticPPcpp.dll from location into current directory



Now run Example

Command Prompt

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\PEER Center>cd DEVELOPER
C:\Users\PEER Center\DEVELOPER>cd material
C:\Users\PEER Center\DEVELOPER\material>cd cpp
C:\Users\PEER Center\DEVELOPER\material\cpp>OpenSees Example1.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.2

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

ElasticPPcpp unaxial material - Written by fmk UC Berkeley Copyright 2008 - Use
at your Own Peril

Node: 4
Coordinates : 72 96
Disps: 0.530093 -0.177894
Velocities : 0 0
unbalanced Load: 100 -50
ID : 0 1

Element: 1 type: Truss iNode: 1 jNode: 4 Area: 10 Mass/Length: 0
strain: 0.00146451 axial load: 30
unbalanced load: -18 -24 18 24
Material: ElasticPPcpp tag: 1
E: 3000
ep: 0.000464506
stress: 3 tangent: 0

Element: 2 type: Truss iNode: 2 jNode: 4 Area: 5 Mass/Length: 0
strain: -0.00383642 axial load: -15
unbalanced load: -9 12 9 -12
Material: ElasticPPcpp tag: 1
E: 3000
ep: -0.00283642
stress: -3 tangent: 0

Element: 3 type: Truss iNode: 3 jNode: 4 Area: 5 Mass/Length: 0
strain: -0.00368743 axial load: -15
unbalanced load: -10.6066 10.6066 10.6066 -10.6066
Material: ElasticPPcpp tag: 1
E: 3000
ep: -0.00268743
stress: -3 tangent: 0

C:\Users\PEER Center\DEVELOPER\material\cpp>
```

What Follows are the Steps required to Build ElasticPPcpp.so on a Linux Machine with gcc installed

**NOTE: We Will use
NEEShub
for this demonstration
(<http://nees.org>)**

3 Simple Steps!

1. Download code

`svn co svn://opensees.berkeley.edu/usr/local/svn/OpenSees/trunk/OpenSees/Developer Developer`

2. `cd DEVELOPER/material/cpp`

3. type `make`

if you type `ls` you should see the `.so` and you

Can test it using `>OpenSees example1.tcl`

NOTE: mac users must have xcode installed and must open `DEVELOPER/Makefile.def` and switch comments on lines 1 and 2 before step 3.

1: Download Source Code

NEES - Resources: Tools: Workspace: Session: 33391 "Workspace" - Mozilla Firefox

File Edit View History Bookmarks Tools Help

NEES - Resources: Tools: Works... +

https://nees.org/tools/workspace/session/33391

startnow Search with Bing Search Shopping Games Travel Amazon eBay Facebook Tw

NEEShub
George E. Brown, Jr. Network for Earthquake Engineering Simulation

Logout My Account 631 Frank

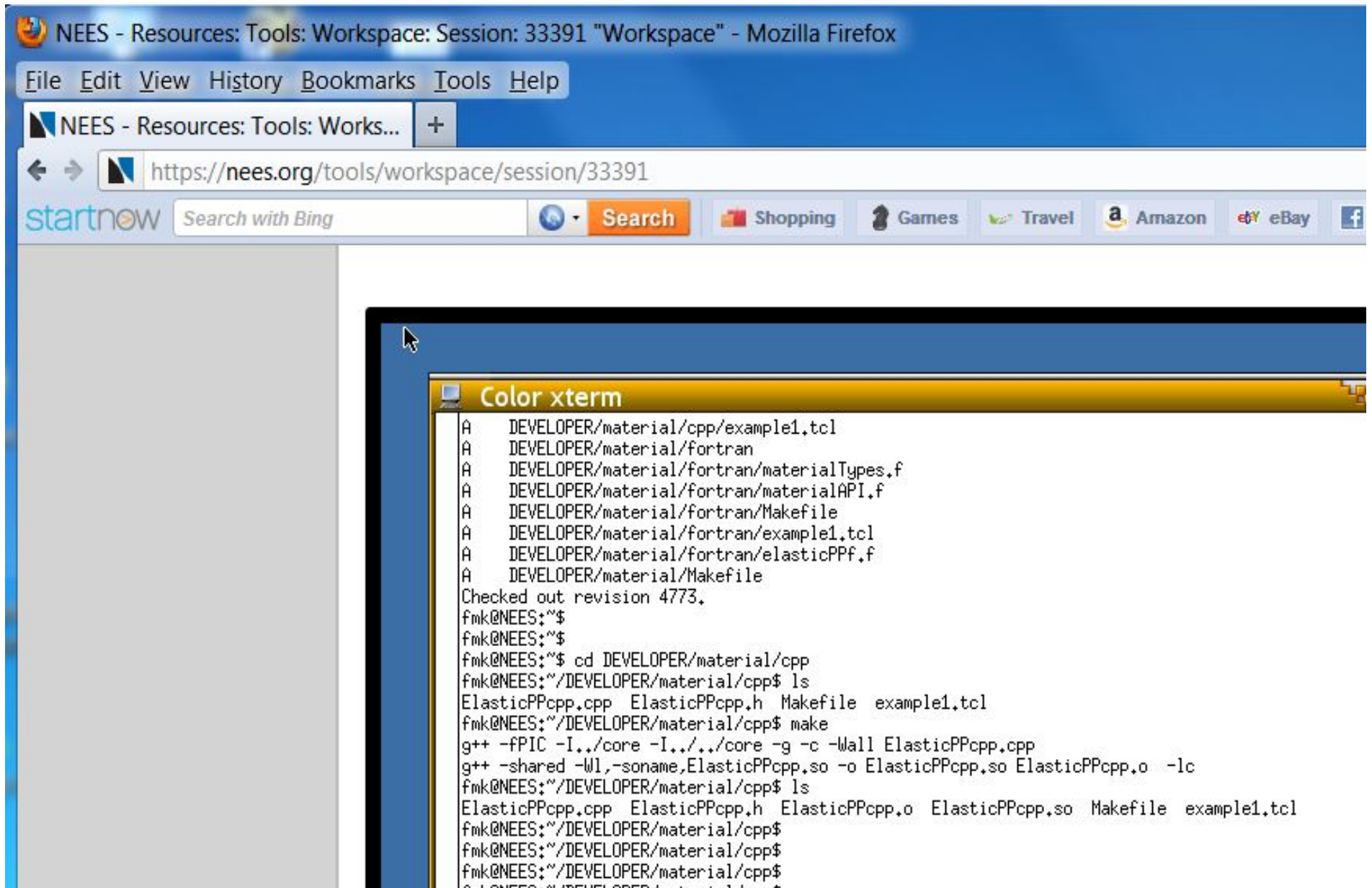
About NEES Tools & Resources Learning & Outreach Project Warehouse Simulation Sites Collaborate E

You are here: Home » Resources » Tools » Workspace » Session: 33391 "Workspace"

Workspace

```
Color xterm  
fmk@NEES:~$ svn co svn://opensees.berkeley.edu/usr/local/svn/OpenSees/trunk/DEVELOPER DEVELOPER
```

2/3: cd to Directory & Type make



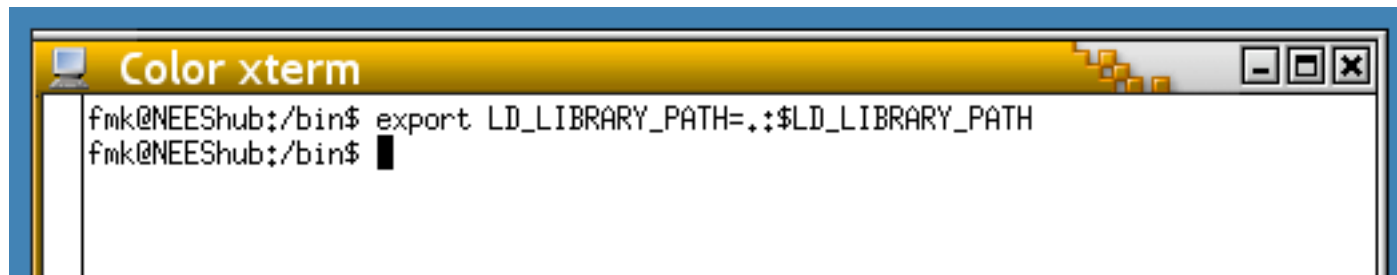
The screenshot shows a Mozilla Firefox browser window with the address bar displaying `https://nees.org/tools/workspace/session/33391`. The browser's menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. Below the browser window, a terminal window titled "Color xterm" is open, showing the following shell session:

```
A DEVELOPER/material/cpp/example1.tcl
A DEVELOPER/material/fortran
A DEVELOPER/material/fortran/materialTypes.f
A DEVELOPER/material/fortran/materialAPI.f
A DEVELOPER/material/fortran/Makefile
A DEVELOPER/material/fortran/example1.tcl
A DEVELOPER/material/fortran/elasticPPf.f
A DEVELOPER/material/Makefile
Checked out revision 4773.
fmk@NEES:~$
fmk@NEES:~$
fmk@NEES:~$ cd DEVELOPER/material/cpp
fmk@NEES:~/DEVELOPER/material/cpp$ ls
ElasticPPcpp.cpp ElasticPPcpp.h Makefile example1.tcl
fmk@NEES:~/DEVELOPER/material/cpp$ make
g++ -fPIC -I../core -I../..../core -g -c -Wall ElasticPPcpp.cpp
g++ -shared -Wl,-soname,ElasticPPcpp.so -o ElasticPPcpp.so ElasticPPcpp.o -lc
fmk@NEES:~/DEVELOPER/material/cpp$ ls
ElasticPPcpp.cpp ElasticPPcpp.h ElasticPPcpp.o ElasticPPcpp.so Makefile example1.tcl
fmk@NEES:~/DEVELOPER/material/cpp$
fmk@NEES:~/DEVELOPER/material/cpp$
fmk@NEES:~/DEVELOPER/material/cpp$
fmk@NEES:~/DEVELOPER/material/cpp$
```


Outline of Workshop

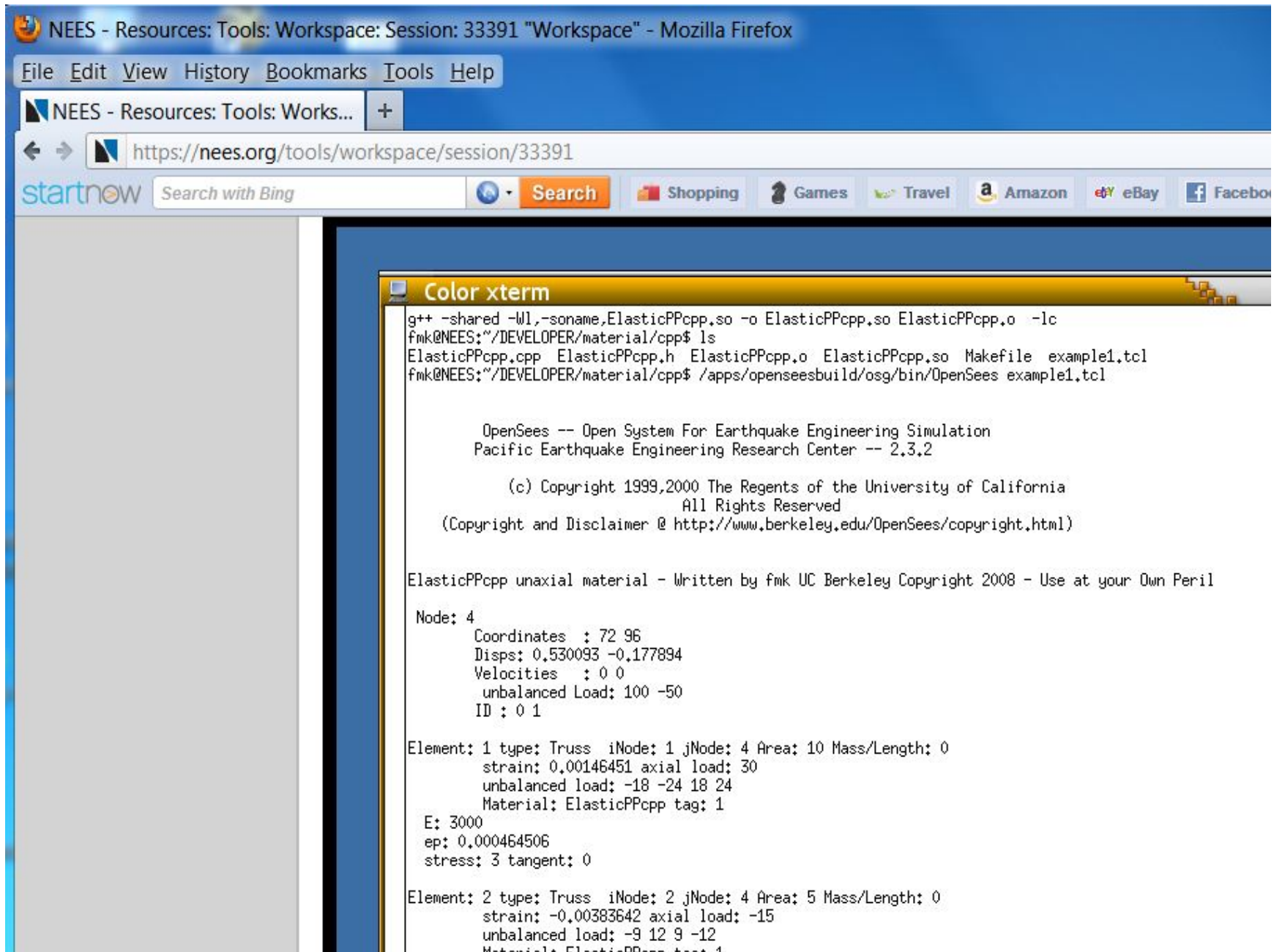
- Introduction
- OpenSees & Tcl Interpreters
- Output
- Modeling in OpenSees
- Nonlinear Analysis
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Adding Your Code to OpenSees
- Conclusion

Add the current directory
to your LD_LIBRARY_PATH



```
Color xterm
fmk@NEEShub:/bin$ export LD_LIBRARY_PATH=./:$LD_LIBRARY_PATH
fmk@NEEShub:/bin$
```


Run It



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `https://nees.org/tools/workspace/session/33391`. The browser's menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. Below the browser window, a terminal window titled "Color xterm" is open, showing the execution of a C++ program. The terminal output includes the command `g++ -shared -Wl,-soname,ElasticPPcpp.so -o ElasticPPcpp.so ElasticPPcpp.o -lc` and the execution of `OpenSees example1.tcl`. The output displays the OpenSees version (2.3.2), copyright information, and simulation results for a truss element.

```
g++ -shared -Wl,-soname,ElasticPPcpp.so -o ElasticPPcpp.so ElasticPPcpp.o -lc
fmk@NEES:~/DEVELOPER/material/cpp$ ls
ElasticPPcpp.cpp ElasticPPcpp.h ElasticPPcpp.o ElasticPPcpp.so Makefile example1.tcl
fmk@NEES:~/DEVELOPER/material/cpp$ /apps/openseesbuild/osg/bin/OpenSees example1.tcl

      OpenSees -- Open System For Earthquake Engineering Simulation
      Pacific Earthquake Engineering Research Center -- 2.3.2

      (c) Copyright 1999,2000 The Regents of the University of California
      All Rights Reserved
      (Copyright and Disclaimer @ http://www.berkeley.edu/OpenSees/copyright.html)

ElasticPPcpp uniaxial material - Written by fmk UC Berkeley Copyright 2008 - Use at your Own Peril

Node: 4
  Coordinates : 72 96
  Disps: 0.530093 -0.177894
  Velocities : 0 0
  unbalanced Load: 100 -50
  ID : 0 1

Element: 1 type: Truss iNode: 1 jNode: 4 Area: 10 Mass/Length: 0
  strain: 0.00146451 axial load: 30
  unbalanced load: -18 -24 18 24
  Material: ElasticPPcpp tag: 1
  E: 3000
  ep: 0.000464506
  stress: 3 tangent: 0

Element: 2 type: Truss iNode: 2 jNode: 4 Area: 5 Mass/Length: 0
  strain: -0.00383642 axial load: -15
  unbalanced load: -9 12 9 -12
  Material: ElasticPPcpp tag: 1
```

Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Advanced Modeling is about
developing scripts that are easier
to generate, easier to modify and
debug, easier for others to
decipher (think your advisor!)
and less error prone


```

# define structure-geometry parameters
set NStories 3. # number of stories
set NodalMass2V 0.105; # mass at each column node on Floor
set

```

3 Story 5 Bay Moment Frame

Original Code >350 lines

```

noc uniaxial # command: forceBeamColumn $eleTag $iNode $jNode $numInt
noc # eleID convention: "1xy" where 1 = col, x = Pier #, y

```

If I had to do this I would
want a GUI too!

```

$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
$PDeltaT
noc element forceBeamColumn 112 122 132 $NIPcol 12 $PDeltaT
node 32 $Pier3 $Floor2 -mass $NodalMass2H $NodalMass2V 0.0;
node 42 $Pier4 $Floor2 -mass $NodalMass2H $NodalMass2V 0.0;

```

The SCRIPTS you are submitting
to OpenSees
are PROGRAMS
you are submitting to a powerful
INTERPRETER ..

**SO LETS START
PROGRAMMING!**

A computer is a pretty stupid machine that stores a bunch of 0/1's and can just move 0/1's around very very very quickly. It only deals with 0/1's, it knows nothing about characters, files, and OpenSees elements!

Abstraction is what makes computers usable

- Computing is all about constructing, manipulating, and reasoning about abstractions. An important prerequisite for writing (good) computer programs is the ability to develop your own abstractions and handle them in a precise manner.
- An abstraction captures only those details about an object that are relevant to the current perspective.
- Through the process of abstraction, a programmer hides all but the relevant data about an object in order to reduce complexity and increase efficiency.

In 40 min I Cannot Show you
how to be a Programmer
But I can start you down the Path

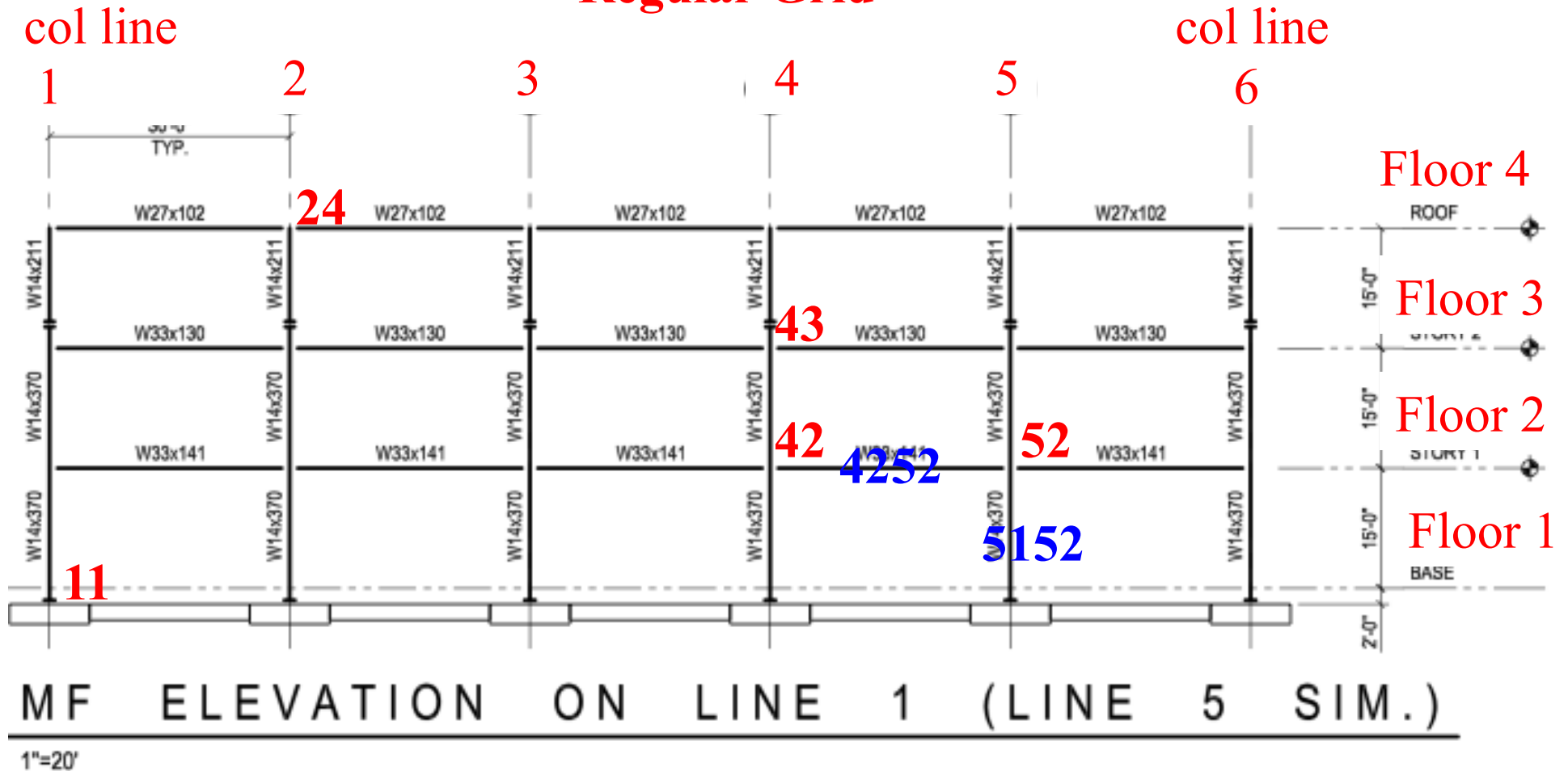


- “There are two ways of constructing a software design: one way is to make it so simple that there are *obviously* no deficiencies; the other way is to make it so complicated that there are no *obvious* deficiencies. The first method is far more difficult.” C.A. Hoare, *The Emperor’s Old Clothes*, 1980



KISS is an acronym
for "**Keep it simple,
stupid**"

Steel W Sections & Regular Grid



Nodes # \$col\$floor
Elements # \$iNode\$jNode

(if more than 10 col lines or floors,
 start numbering at 10,
 if > 100, at 100,)

MRF1.tcl

Same
model
in 35
lines!

```
model Basic -ndm 2 -ndf 3
source Steel2d.tcl

# set some lists containing floor and column line locations and nodal masses
set floorLocs {0. 204. 384. 564.};           # floor locations in inches
set colLocs {0. 360. 720. 1080. 1440. 1800.}; #column line locations in inches
set massesX {0. 0.419 0.419 0.430};         # mass at nodes on each floor in x dirn
set massesY {0. 0.105 0.105 0.096};         # " " " " " " " in y dirn

# add nodes at each floor at each column line location & fix nodes if at floor 1
foreach floor {1 2 3 4} floorLoc $floorLocs massX $massesX massY $massesY {
  foreach colLine {1 2 3 4 5 6} colLoc $colLocs {
    node $colLine$floor $colLoc $floorLoc -mass $massX $massY 0.
    if {$floor == 1} {fix $colLine$floor 1 1 1}
  }
}

#uniaxialMaterial Steel02 $tag $Fy $E $b $R0 $scr1 $scr2
uniaxialMaterial Steel02 1 50.0 29000. 0.003 20 0.925 0.15; # material to be used for steel elements

# set some list for col and beam sizes
set colSizes {W14X370 W14X370 W14X211}; #col sizes stories 1, 2 and 3
set beamSizes {W33X141 W33X130 W27X102}; #beams sizes floor 1, 2, and 3

# add columns at each column line between floors
geomTransf PDelta 1
foreach colLine {1 2 3 4 5 6} {
  foreach floor1 {1 2 3} floor2 {2 3 4} {
    set theSection [lindex $colSizes [expr $floor1 -1]]; # obtain section size for column
    ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1 -nip 5
  }
}

#add beams between column lines at each floor
geomTransf Linear 2
foreach colLine1 {1 2 3 4 5} colLine2 {2 3 4 5 6} {
  foreach floor {2 3 4} {
    set theSection [lindex $beamSizes [expr $floor -2]]; # obtain section size for floor
    ForceBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2
  }
}
```

Now We Can Do Fun Stuff

MRF2.tcl

How Many Fibers?

```
foreach nFlange {1 2 3 10 10 20 35} nWeb {4 4 4 5 10 20 30} {
```

```
  puts "nFlange: $nFlange nWeb: $nWeb"
```

```
  wipe;
```

```
  model BasicBuilder -ndm 2 -ndf 3;
```

```
  ....
```

```
  ....
```

```
# add columns at each column line between floors
```

```
geomTransf PDelta 1
```

```
foreach colLine {1 2 3 4 5 6} {
```

```
  foreach floor1 {1 2 3} floor2 {2 3 4} {
```

```
    set theSection [lindex $colSizes [expr $floor1 -1]]; # obtain section size for column
```

```
    ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1 -nFlange $nFlange -nWeb $nW
```

```
  }
```

```
}
```

```
#add beams between column lines at each floor
```

```
geomTransf Linear 2
```

```
foreach colLine1 {1 2 3 4 5} colLine2 {2 3 4 5 6} {
```

```
  foreach floor {2 3 4} {
```

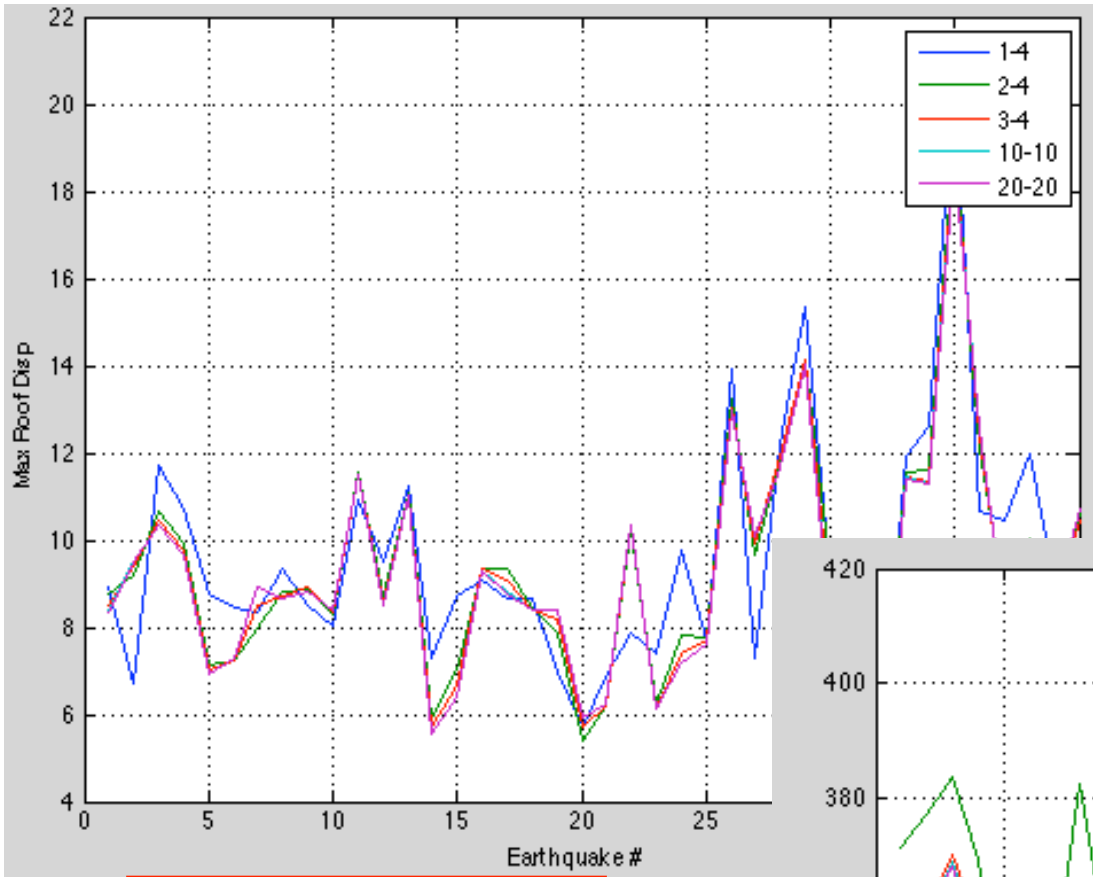
```
    set theSection [lindex $beamSizes [expr $floor -2]]; # obtain section size for floor
```

```
    ForceBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2 -nFlange $nFlange -nWeb $nW
```

```
  }
```

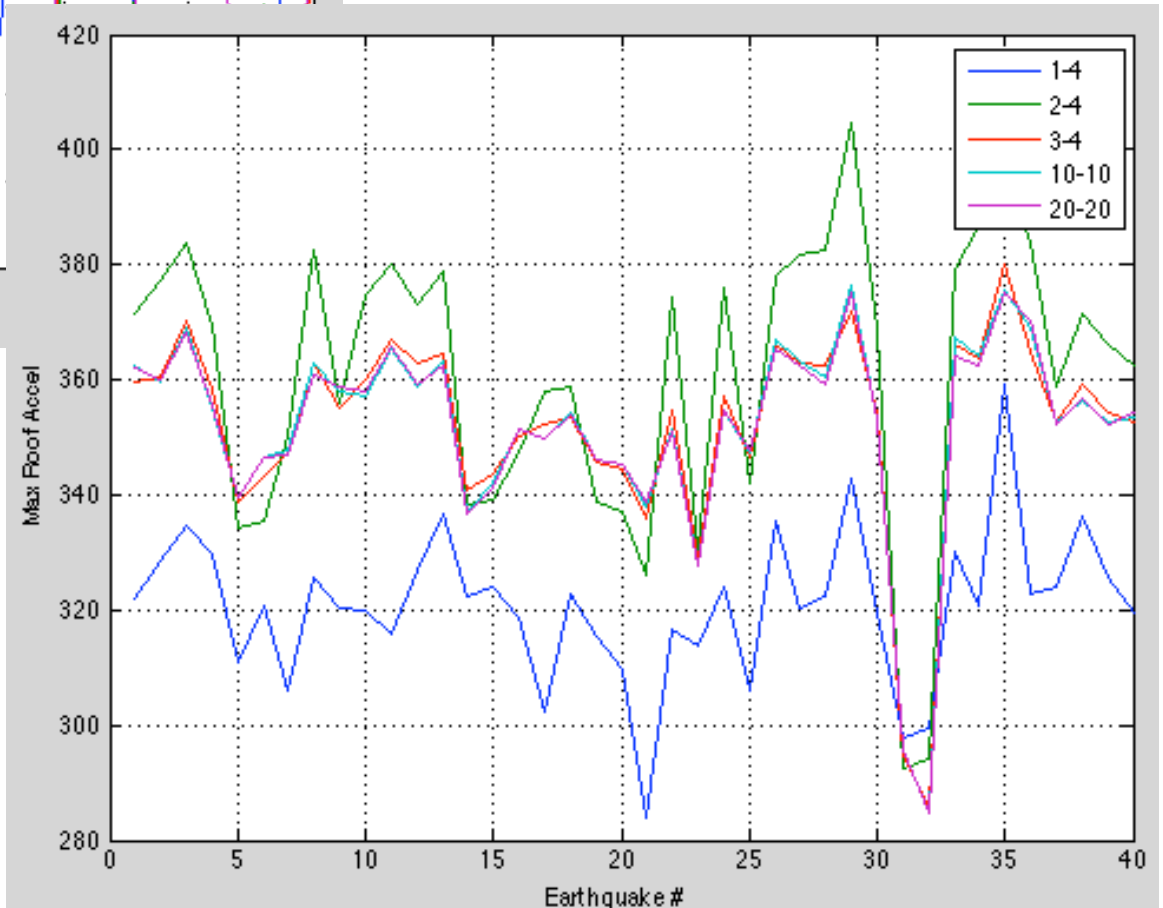
```
}
```

```
}
```

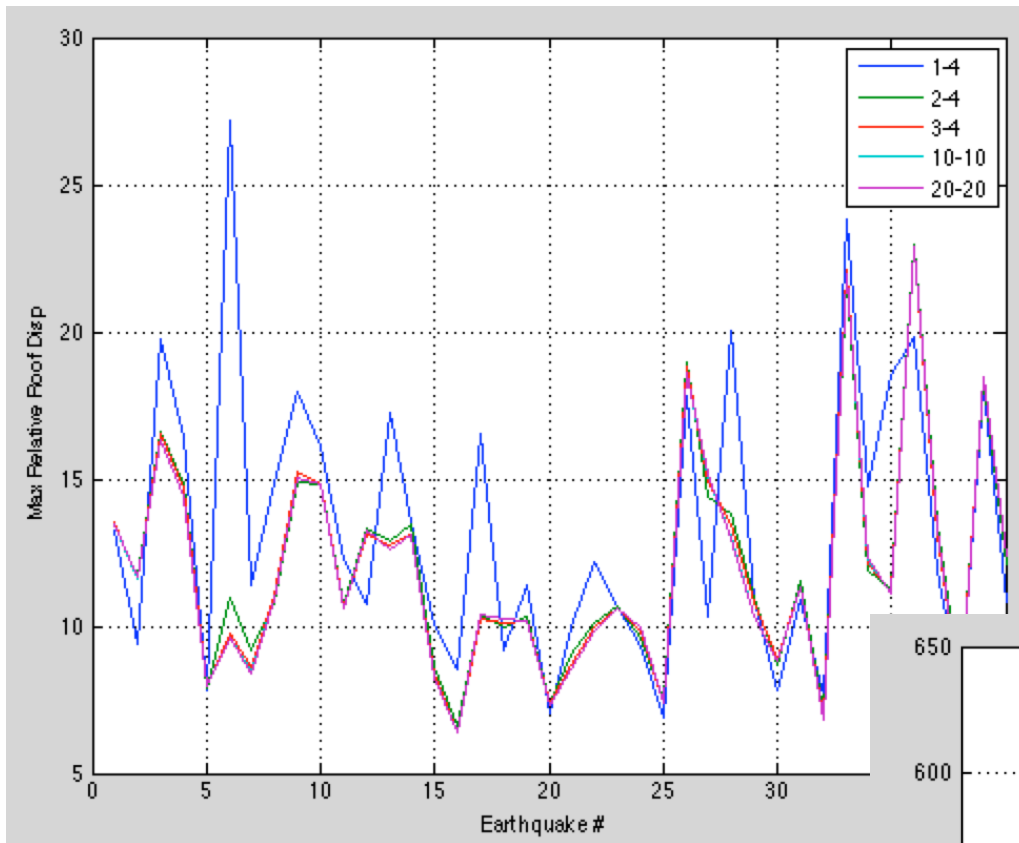


fibers flange- # fibers web

Results 10 % in 50year

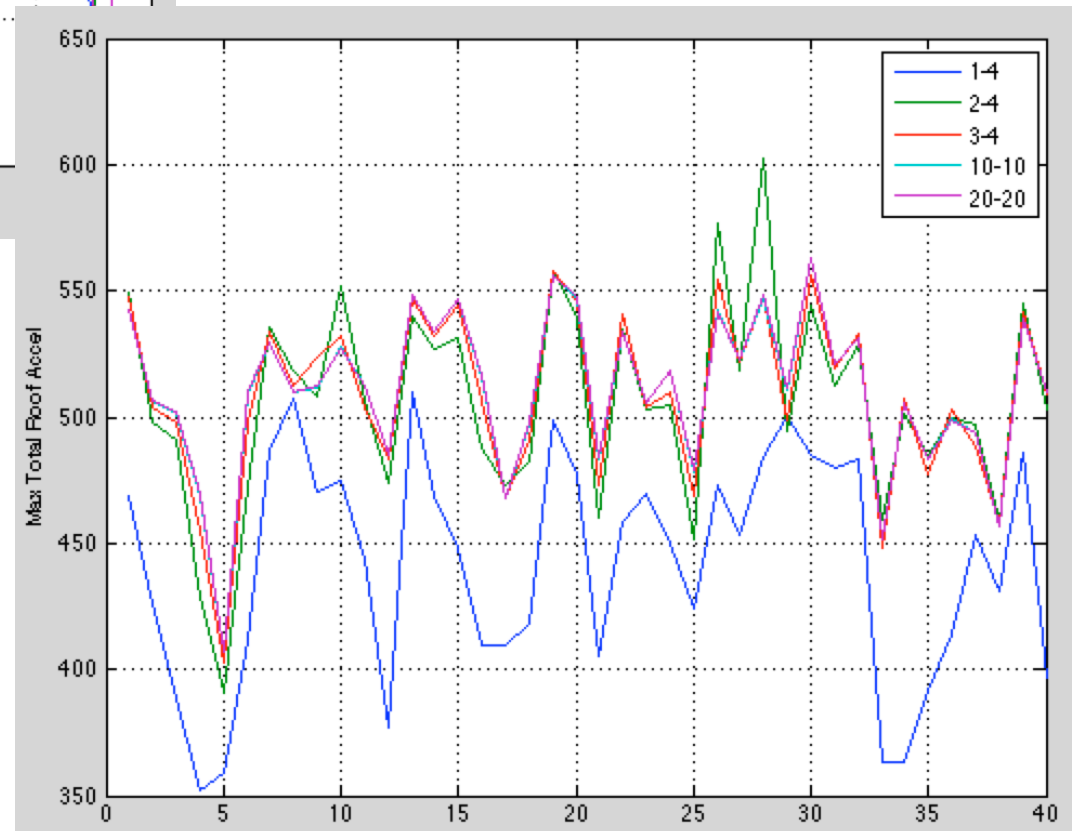


	Time
1-4	292sec
2-4	302sec
3-4	309sec
10-10	578sec
20-20	1001sec
35-30	1305sec



fibers flange- # fibers web

Results 2% in 50year



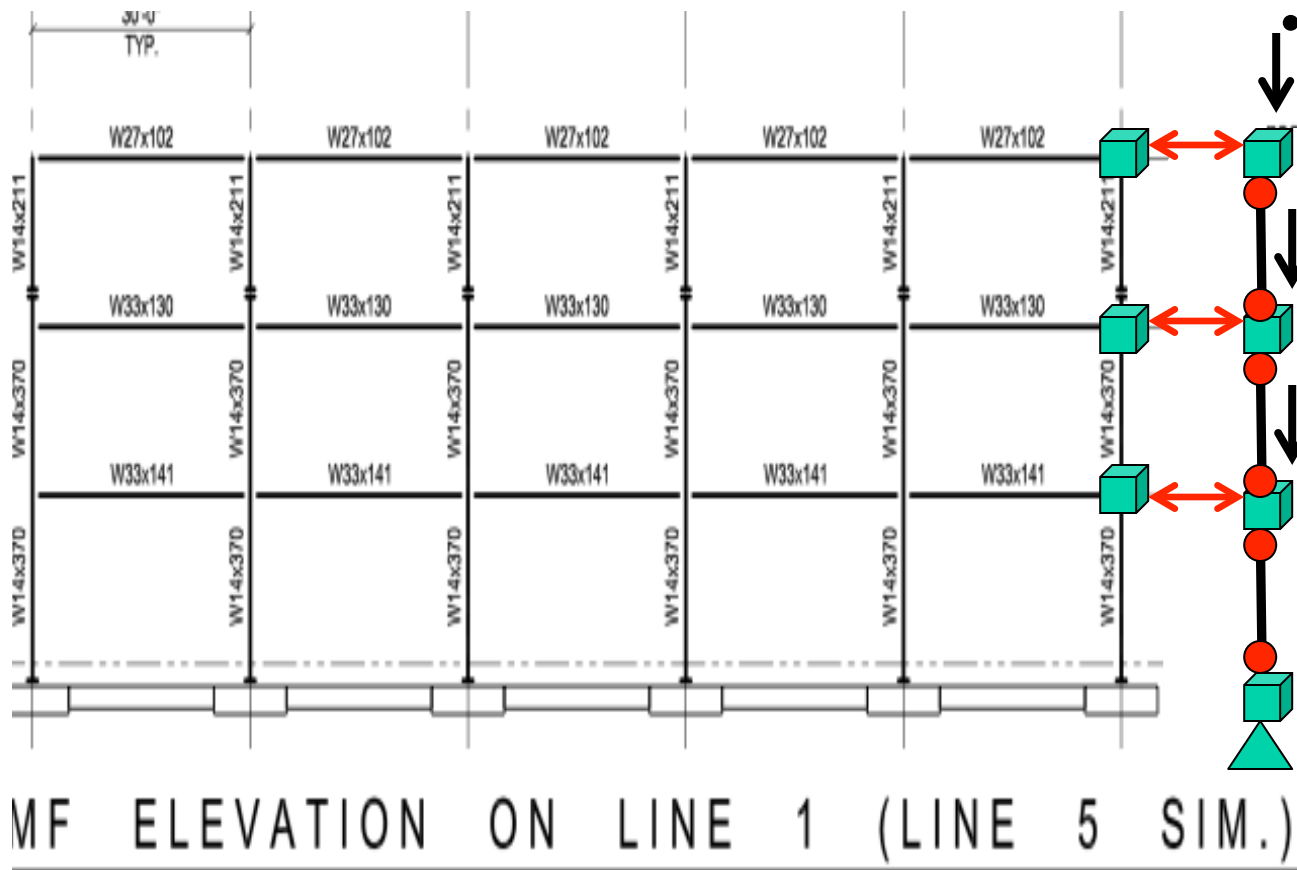
	Time
1-4	254sec
2-4	265sec
3-4	272sec
10-10	506sec
20-20	879sec
35-30	1539sec

PDelta Gravity Column

- To include PDelta effects from rest of structure on lateral system
- Impact seen when axial forces are large (think medium & high-rise)

Gravity column

- Pinned base
- Gravity Column Loads
- Truss with Pdelta
- Tied to lateral system with MP Constraint



1"=20'

- Biggest impact possible at bottom where axial loads largest.
- The bigger inter-story drift the bigger the impact.


```

geomTransf PDelta 3;
set Arigid 10000.0
set Irigid 100000.0
set Ismall 1.0e-1;
set gravityColsArea {806.7 806.7 477.7}; # half sum of area of gravity cols & orthogonal frame
columns

```

```

#nodes
set col6 6; set col7 7;
foreach floor {1 2 3 4} floorLoc {0 204. 384. 564.} {
  node $col7$floor 1900. $floorLoc
  if {$floor == 1} {fix $col7$floor 1 1 0}
}

```

for Pdelta Truss response, I am using elastic beam with small I to get PDelta

```

#elements
foreach floor1 {1 2 3} floor2 { 2 3 4} {
  set A [lindex $gravityColsArea [expr $floor1-1]]
  element elasticBeamColumn 7$floor1$floor2 7$floor1 7$floor2 $A $Es $Ismall 3;
  element Truss $col6$floor2$col7$floor2 $col6$floor2 $col7$floor2 $Arigid 3
  # equalDOF $col6$floor2 $col7$floor2 1
}

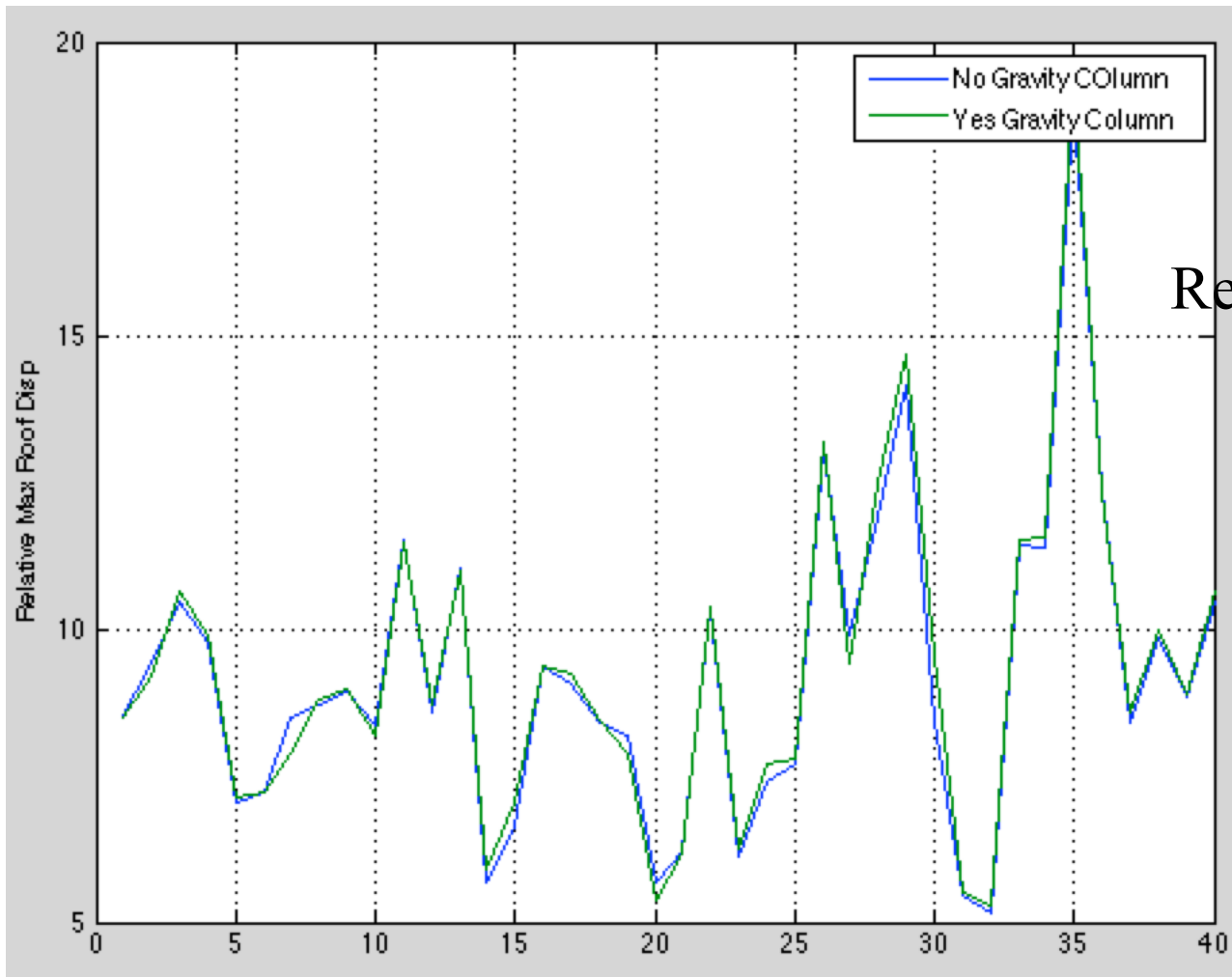
```

Either Works, Just wanted to look At forces being transmitted to frame.

```

# loads
pattern Plain 3 "Linear" {
  # loads on gravity PDelta column
  load 72 0.0 -748.48 0.0; # Floor 2
  load 73 0.0 -748.48 0.0; # Floor 3
  load 74 0.0 -792.84 0.0; # Floor 4
}

```



Results 10 % in 50year

Pdelta will only have an effect when axial loads are high .. Think tall building

When DONE with a Program

Critique It

(if you have the time this will make
you a better programmer)

```
source Steel2d.tcl
source ReadRecord.tcl;

# set some variables
set motion Oak_2_50_5_FN
set in 1.0;
set g 386.4;      # acceleration due to gravity
set floorOffsets {204. 180. 180.}
set colOffsets  {360. 360. 360. 360. 360.}
set massesX      {0. 0.419 0.419 0.400}
set massesY      {0. 0.105 0.105 0.096}
set colSizes     {W14X370 W14X370 W14X211};
set beamSizes    {W33X141 W33X130 W27X102};

set numFloor [expr [llength $floorOffsets]+1]
set numCline [expr [llength $colOffsets]+1]
set roofFloor [llength $numFloor]

set floorLocations 0; set floorLoc 0;
set colLocations 0; set colLoc 0;

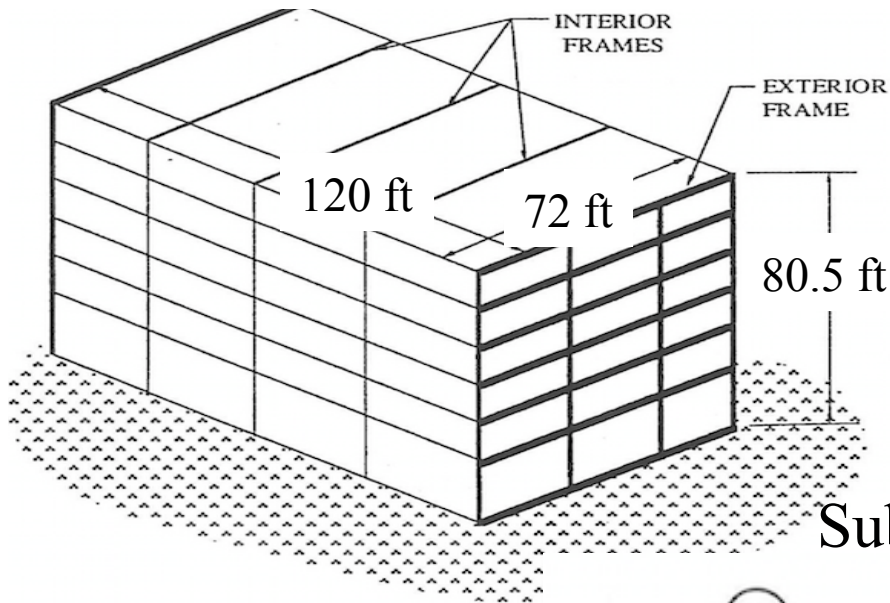
for {set i 1} {$i < $numFloor} {incr i 1} {
  set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $i-1]]]
  lappend floorLocations $floorLoc;
}

for {set i 1} {$i < $numCline} {incr i 1} {
  set colLoc [expr $colLoc + [lindex $colOffsets [expr $i-1]]]
  lappend colLocations $colLoc;
}
```

```

wipe;
model BasicBuilder -ndm 2 -ndf 3;
# build the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
    set floorLoc [lindex $floorLocations [expr $floor-1]]
    set massX [lindex $massesX [expr $floor-1]]
    set massY [lindex $massesY [expr $floor-1]]
    for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
        set colLoc [lindex $colLocations [expr $colLine-1]]
        node $colLine$floor $colLoc $floorLoc -mass $massX $massY 0.
        if {$floor == 1} { fix $colLine$floor 1 1 1}
    }
}
# define material
uniaxialMaterial Steel02 1 50.0 29000 0.003 20 0.925 0.15
# build the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
    for {set floor1 1} {$floor1 < $numFloor} {incr floor1 1} {
        set floor2 [expr $floor1+1]
        set theSection [lindex $colSizes [expr $floor1 -1]]
        ForceBeamWSection2d $colLine$floor1$colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1
    }
}
#build the beams
geomTransf Linear 2
for {set colLine1 1} {$colLine1 < $numCline} {incr colLine1 1} {
    set colLine2 [expr $colLine1 + 1]
    for {set floor 2} {$floor <= $numFloor} {incr floor 1} {
        set theSection [lindex $beamSizes [expr $floor -2]]
        ForceBeamWSection2d $colLine1$floor$colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2
    }
}

```



Max Roof Displacement = ?

Dead Load: 95psf typical, roof 80psf
 $E=29,000$, $F_y=50.0$, $b=0.003$

Subjected to el_centro 1940 N-S (Peknold)

	A	I _{xx}	ROOF	(A)	(B)	(C)	(D)	
W14X193	56.8	2400	6	W14 x 109	W24 x 104	W24 x 76	6th Story	12.5 ft
W14X159	46.7	1900	5			W24 x 76		
W14X109	32.0	1240	4		W27 x 146	W27 x 94	24 ft	
W30X173	51.0	8230	3			W27 x 94		
W27X146	43.1	5660	2		W30 x 173	W30 x 99	2nd Story	
W24X104	30.6	3100	1			W30 x 99		
W30X99	29.1	3990	G			1st Story	18 ft	
W27X94	27.7	3270						
W24X76	22.4	2100						

This figure is taken from the first paper in the SAC 95-05 report: Hall, John F. "Parameter Study of Response of Moment-Resisting Steel Frame Buildings to Near-Source Ground Motions"

```
source Steel2d.tcl
source ReadRecord.tcl;
```

Solution2014.tcl

```
# set some variables
set motion el_centro
set in 1.0;
set g 386.4;      # acceleration due to gravity
set floorOffsets {216. 150. 150. 150. 150. 150.}
set colOffsets   {288. 288. 288.}
set colSizes     {W30X173 W30X173 W27X146 W27X146 W24X76 W24X76};
set colExtSizes {W14X193 W14X193 W14X159 W14X159 W14X109 W14X109};
set beamSizes    {W30X99 W30X99 W27X94 W27X94 W24X76 W24X76}
set massesX      {0. 0.2655 0.2655 0.2655 0.2655 0.2655 0.223}
set massesY      {0. 0.2655 0.2655 0.2655 0.2655 0.2655 0.223}

set numFloor [expr [llength $floorOffsets]+1]
set numCline [expr [llength $colOffsets]+1]
set roofFloor [llength $numFloor]

set floorLocations 0; set floorLoc 0;
set colLocations 0; set colLoc 0;

for {set i 1} {$i < $numFloor} {incr i 1} {
    set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $i-1]]]
    lappend floorLocations $floorLoc;
}
for {set i 1} {$i < $numCline} {incr i 1} {
    set colLoc [expr $colLoc + [lindex $colOffsets [expr $i-1]]]
    lappend colLocations $colLoc;
}
```

Solution2014.tcl

```
wipe;
model BasicBuilder -ndm 2 -ndf 3;
# build the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
    set floorLoc [lindex $floorLocations [expr $floor-1]]
    set massX [lindex $massesX [expr $floor-1]]
    set massY [lindex $massesY [expr $floor-1]]
    for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
        set colLoc [lindex $colLocations [expr $colLine-1]]
        node $colLine$floor $colLoc $floorLoc -mass $massX $massY 0.
        if {$floor == 1} { fix $colLine$floor 1 1 1}
    }
}
# define material
uniaxialMaterial Steel02 1 50.0 29000 0.003 20 0.925 0.15
# build the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
    for {set floor1 1} {$floor1 < $numFloor} {incr floor1 1} {
        set floor2 [expr $floor1+1]
        if {colLine == 1 || colLine = $numCline} {
            set theSection [lindex $colSizes [expr $floor1 -1]]
        } else { set theSection [lindex $colSExtizes [expr $floor1 -1]]
        ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1
    }
}
# build the beams
geomTransf Linear 2
for {set colLine1 1} {$colLine1 < $numCline} {incr colLine1 1} {
    set colLine2 [expr $colLine1 + 1]
    for {set floor 2} {$floor <= $numFloor} {incr floor 1} {
        set theSection [lindex $beamSizes [expr $floor -2]]
        ForceBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2
    }
}
```



```
source Steel2d.tcl
source ReadRecord.tcl;

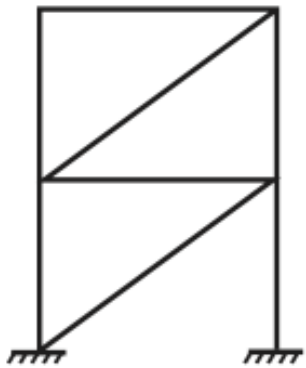
# set some variables
set motion Oak_2_50_5_FN
set in 1.0;
set g 386.4;      # acceleration due to gravity
set floorOffsets {204. 180. 180.}
set colOffsets   {360. 360. 360. 360. 360.}
set massesX     {0. 0.419 0.419 0.400}
set massesY     {0. 0.105 0.105 0.096}
set colSizes    {W14X370 W14X370 W14X211};
set colExtSizes {W14X370 W14X370 W14X211};
set beamSizes   {W33X141 W33X130 W27X102};

set numFloor [expr [llength $floorOffsets]+1]
set numCline [expr [llength $colOffsets]+1]
set roofFloor [llength $numFloor]

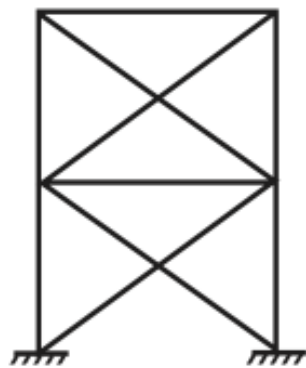
set floorLocations 0; set floorLoc 0;
set colLocations 0; set colLoc 0;

for {set i 1} {$i < $numFloor} {incr i 1} {
    set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $i-1]]]
    lappend floorLocations $floorLoc;
}
for {set i 1} {$i < $numCline} {incr i 1} {
    set colLoc [expr $colLoc + [lindex $colOffsets [expr $i-1]]]
    lappend colLocations $colLoc;
}
```

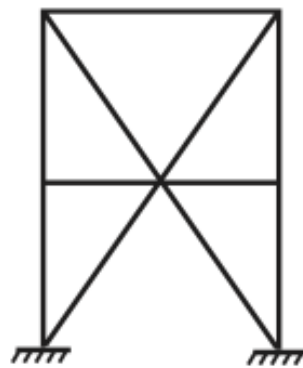
Some Braced Frame EXAMPLES: Typical Configurations



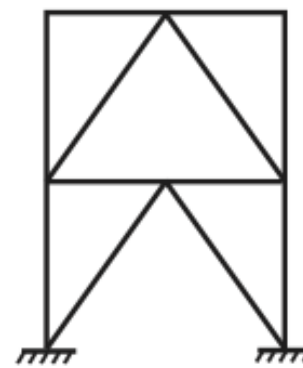
Diagonal bracing



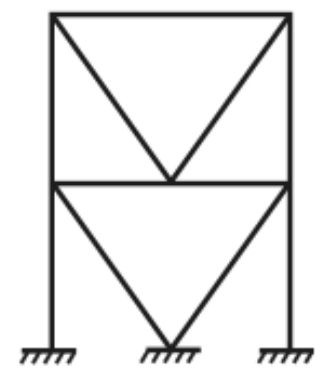
X-bracing



Multistory X-bracing

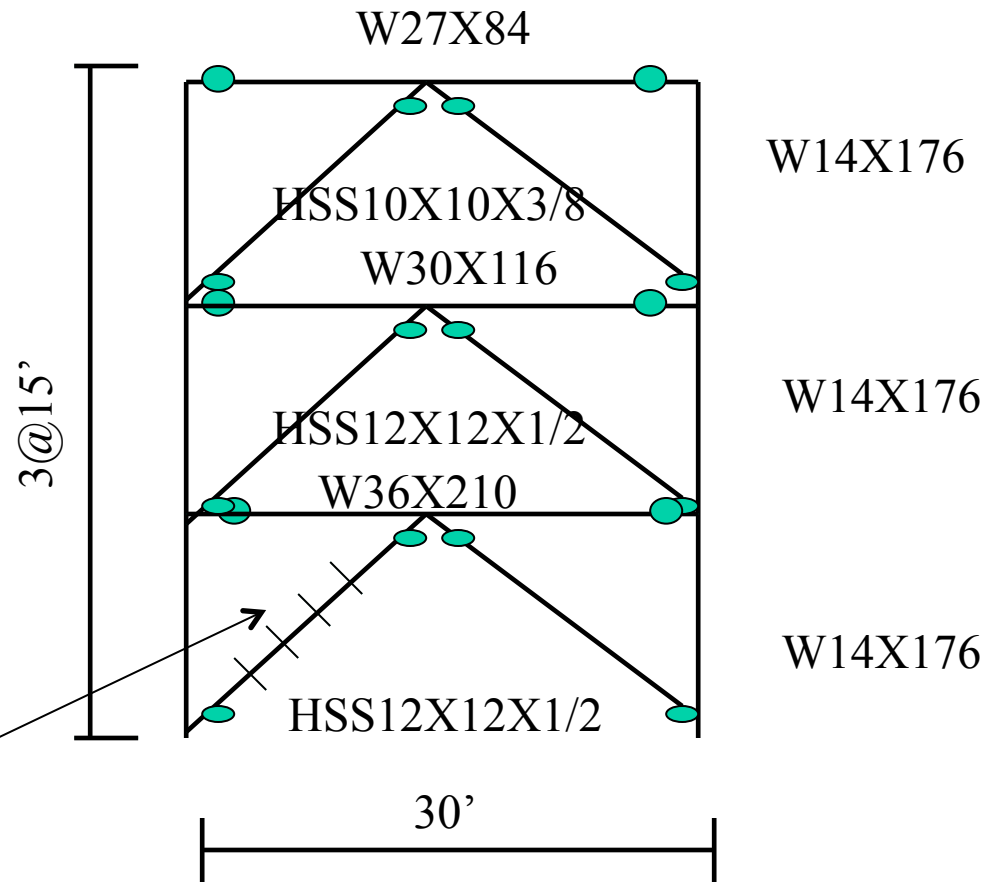
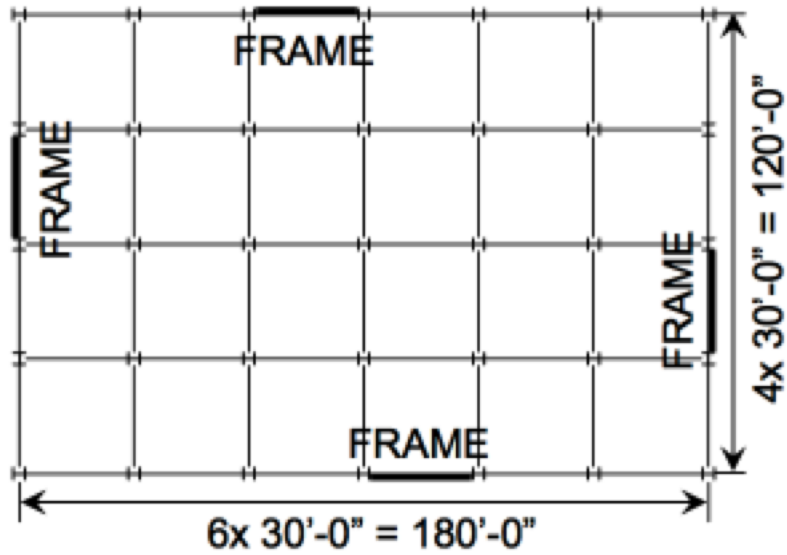


Inverted V-bracing
(Chevron)



V-bracing

- Beams pinned at column



#number of beam elements
pinned at either end

Imperfection so will buckle

```
proc HSSbrace $eleTag $iNode $jNode $secType $matTag $numSeg $imperfection $transfTag arg
```



D Lignos (McGill)

```

source Steel2d.tcl
# set up my structure
set colLocations {0. 180. 360.}
set floorLocations {0. 180. 360. 540.}
set masses {0. 0.419 0.419 0.400}
set colSizes {W14X176 W14X176 W14X176};
set beamSizes {W36X210 W30X116 W27X84};
set braceSizes {HSS12X12X1/2 HSS12X12X1/2 HSS10X10X3/8}

```

```

set numFloor [llength $floorLocations]
set numStory [expr $numFloor-1]
set numCline [llength $colLocations]

```

```

model BasicBuilder -ndm 2 -ndf 3;
#add the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
  set floorLoc [lindex $floorLocations [expr $floor-1]]
  set mass [lindex $masses [expr $floor-1]]
  for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
    set colLoc [lindex $colLocations [expr $colLine-1]]
    node $colLine$floor $colLoc $floorLoc -mass $mass $mass 0.
    if {$floor == 1} {
      fix $colLine$floor 1 1 1
    }
  }
}
# define material for col and beams
uniaxialMaterial Steel02 1 50.0 29000 0.03 20 0.925 0.15 0.0005 0.01 0.0005 0.01

```

```

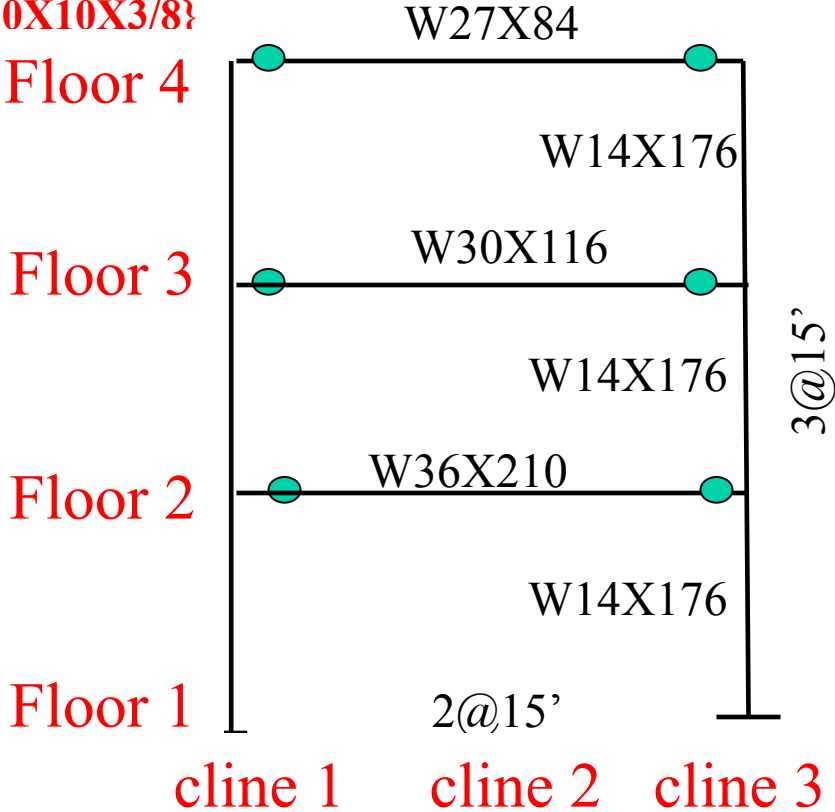
# add the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 2} {
  for {set floor1 1} {$floor1 < $numFloor} {incr floor1 1} {
    set floor2 [expr $floor1+1]
    set theSection [lindex $colSizes [expr $floor1 -1]]
    ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1 -nip 5
  }
}

```

```

# add the beams, pinned connection at column end
geomTransf Linear 2
for {set floor 2} {$floor <= $numFloor} {incr floor 1} {
  set colLine1 1; set colLine2 2; set colLine3 3;
  set theSection [lindex $beamSizes [expr $floor -2]]
  DispBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2 -release 1
}

```



* While diagonal and X braced models do not need me to break up beams, I am doing so as to keep all examples the same.

```

source Steel2d.tcl
# set up my structure
set colLocations {0. 180. 360.}
set floorLocations {0. 180. 360. 540.}
set masses {0. 0.419 0.419 0.400}
set colSizes {W14X176 W14X176 W14X176};
set beamSizes {W36X210 W30X116 W27X84};
set braceSizes {HSS12X12X1/2 HSS12X12X1/2 HSS10X10X3/8}

```

```

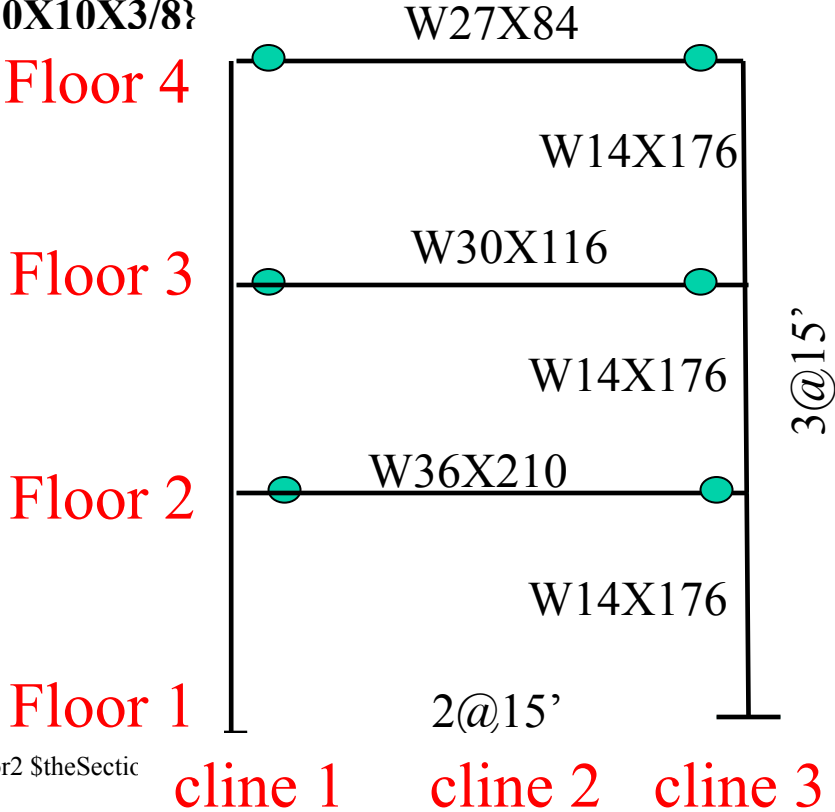
set numFloor [llength $floorLocations]
set numStory [expr $numFloor-1]
set numCline [llength $colLocations]

model BasicBuilder -ndm 2 -ndf 3;
#add the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
  set floorLoc [lindex $floorLocations [expr $floor-1]]
  set mass [lindex $masses [expr $floor-1]]
  for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
    set colLoc [lindex $colLocations [expr $colLine-1]]
    node $colLine$floor $colLoc $floorLoc -mass $mass $mass 0.
    if {$floor == 1} {
      fix $colLine$floor 1 1 1
    }
  }
}
# define material for col and beams
uniaxialMaterial Steel02 1 50.0 29000 0.03 20 0.925 0.15 0.0005 0.01 0.0005 0.01

# add the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 2} {
  for {set floor1 1} {$floor1 < $numFloor} {incr floor1 1} {
    set floor2 [expr $floor1+1]
    set theSection [lindex $colSizes [expr $floor1 -1]]
    ForceBeamWSection2d $colLine$floor1 $colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection
  }
}

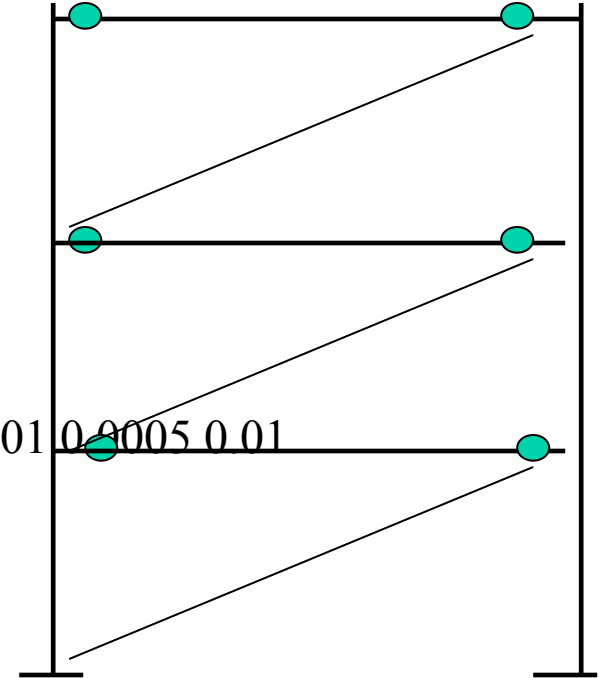
# add the beams, pinned connection at column end
geomTransf Linear 2
for {set floor 2} {$floor <= $numFloor} {incr floor 1} {
  set colLine1 1; set colLine2 2; set colLine3 3;
  set theSection [lindex $beamSizes [expr $floor -2]]
  DispBeamWSection2d $colLine1$floor $colLine2$floor $colLine1$floor $colLine2$floor $theSection 1 2 -release1
  DispBeamWSection2d $colLine2$floor $colLine3$floor $colLine2$floor $colLine3$floor $theSection 1 2 -release2
}

```



* While diagonal and X braced models do not need me to break up beams, I am doing so as to keep all examples the same.

CBF1.tcl (Diagonal bracing)



```
source Steel2d.tcl
# set up my structure
set colLocations {0. 180. 360.}
set floorLocations {0. 180. 360. 540.}
set masses {0. 0.419 0.419 0.400}
set colSizes {W14X176 W14X176 W14X176};
set beamSizes {W36X210 W30X116 W27X84};
set braceSizes {HSS12X12X1/2 HSS12X12X1/2 HSS10X10X3/8}
```

```
set numFloor [llength $floorLocations]
set numStory [expr $numFloor-1]
set numCline [llength $colLocations]

model BasicBuilder -ndm 2 -ndf 3;
#add the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
  set floorLoc [lindex $floorLocations [expr $floor-1]]
  set mass [lindex $masses [expr $floor-1]]
  for {set col line 1} {$col line <= $numCline} {incr col line 1}
```

define material for braces

```
set Fy_b 46.0;
```

```
set E0 0.095
```

```
set m -0.3
```

```
uniaxialMaterial Steel02 2 $Fy_b $Es $b 20 0.925 0.15 0.0005 0.01 0.0005 0.01
```

```
uniaxialMaterial Fatigue 3 2 -E0 $E0 -m $m -min -1.0 -max 0.04
```

```
set imperfection 0.001
```

add the X braces

```
geomTransf Corotational 3
```

```
for {set story 1} {$story <= $numStory} {incr story 1} {
```

```
  set colLine1 1; set colLine2 3;
```

```
  set floor1 $story; set floor2 [expr $story +1];
```

```
  set theSection [lindex $braceSizes [expr $story -1]]
```

```
  HSSbrace $colLine1$floor1$colLine2$floor2 $colLine1$floor1 $colLine2$floor2 $theSection$
```

```
}
```

CBF2.tcl (X bracing)

```
source Steel2d.tcl
# set up my structure
set colLocations {0. 180. 360.}
set floorLocations {0. 180. 360. 540.}
set masses {0. 0.419 0.419 0.400}
set colSizes {W14X176 W14X176 W14X176};
set beamSizes {W36X210 W30X116 W27X84};
set braceSizes {HSS12X12X1/2 HSS12X12X1/2 HSS10X10X3/8}
```

```
set numFloor [llength $floorLocations]
set numStory [expr $numFloor-1]
set numCline [llength $colLocations]
```

```
model BasicBuilder -ndm 2 -ndf 3;
```

```
#add the Nodes
```

```
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
  set floorLoc [lindex $floorLocations [expr $floor-1]]
  set mass [lindex $masses [expr $floor-1]]
  for {set col line 1} {$col line <= $numCline} {incr col line 1}
```

```
# define material for braces
```

```
set Fy_b 46.0;
```

```
set E0 0.095
```

```
set m -0.3
```

```
uniaxialMaterial Steel02 2 $Fy_b $Es $b 20 0.925 0.15 0.0005 0.01 0.0005 0.01
```

```
uniaxialMaterial Fatigue 3 2 -E0 $E0 -m $m -min -1.0 -max 0.04
```

```
set imperfection 0.001
```

```
# add the X braces
```

```
geomTransf Corotational 3
```

```
for {set story 1} {$story <= $numStory} {incr story 1} {
```

```
  set colLine1 1; set colLine2 2; set colLine3 3;
```

```
  set floor1 $story; set floor2 [expr $story +1];
```

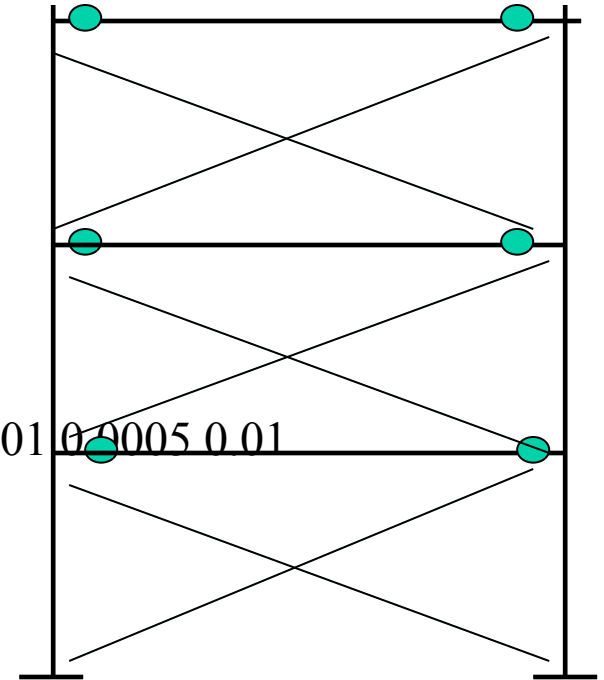
```
  set theSection [lindex $braceSizes [expr $story -1]]
```

```
  # proc HSSbrace {eleTag iNode jNode secType matTag numSeg Im transfTag args}
```

```
  HSSbrace $colLine1$floor1$colLine3$floor2 $colLine1$floor1 $colLine3$floor2 $theSection
```

```
  HSSbrace $colLine3$floor1$colLine1$floor2 $colLine3$floor1 $colLine1$floor2 $theSection
```

```
}
```



CBF3.tcl (V bracing)

```
source Steel2d.tcl
# set up my structure
set colLocations {0. 180. 360.}
set floorLocations {0. 180. 360. 540.}
set masses {0. 0.419 0.419 0.400}
set colSizes {W14X176 W14X176 W14X176};
set beamSizes {W36X210 W30X116 W27X84};
set braceSizes {HSS12X12X1/2 HSS12X12X1/2 HSS10X10X3/8}
```

```
set numFloor [llength $floorLocations]
set numStory [expr $numFloor-1]
set numCline [llength $colLocations]
```

```
model BasicBuilder -ndm 2 -ndf 3;
#add the Nodes
for {set floor 1} {$floor <= $numFloor} {incr floor 1} {
  set floorLoc [lindex $floorLocations [expr $floor-1]]
```

define material for braces

```
set Fy_b 46.0;
```

```
set E0 0.095
```

```
set m -0.3
```

```
uniaxialMaterial Steel02 2 $Fy_b $Es $b 20 0.925 0.15 0.0005 0.01 0.0005 0.01
```

```
uniaxialMaterial Fatigue 3 2 -E0 $E0 -m $m -min -1.0 -max 0.04
```

```
set imperfection 0.001
```

add the V braces

```
geomTransf Corotational 3
```

```
for {set story 1} {$story <= $numStory} {incr story 1} {
```

```
  set colLine1 1; set colLine2 2; set colLine3 3;
```

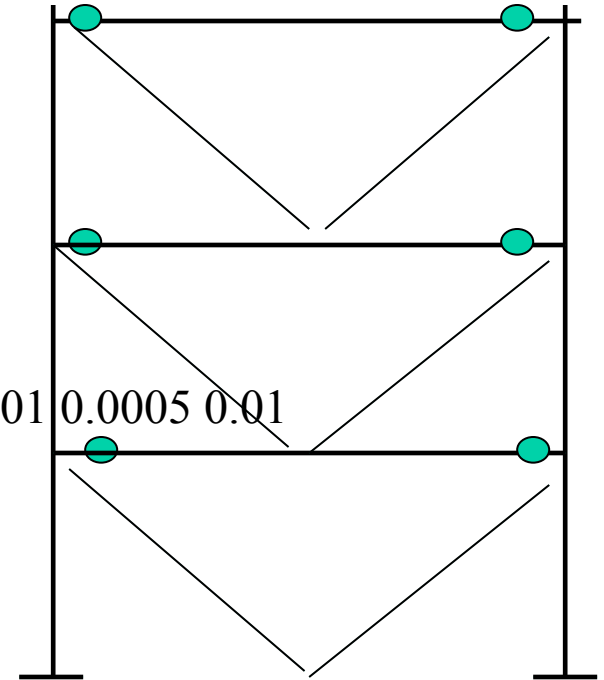
```
  set floor1 $story; set floor2 [expr $story +1];
```

```
  set theSection [lindex $braceSizes [expr $story -1]]
```

```
  HSSbrace $colLine2$floor1$colLine1$floor2 $colLine2$floor1 $colLine1$floor2 $theSection
```

```
  HSSbrace $colLine2$floor1$colLine3$floor2 $colLine2$floor1 $colLine3$floor2 $theSection
```

```
}
```



Outline of Workshop

- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development

• Conclusion

Conclusion

- OpenSees is a powerful tool for performing FE analysis
- It NEEDS contributions from others to grow
- The scripting aspect is something “STUDENTS LEARN TO LOVE”
- OpenSees LIKE ANY simulation tool requires the user to understand the theory and limitations
- OpenSees (if features are fully utilized!) will allow you to generate models faster and more accurately than you could with a GUI and will allow you to obtain information on the Uncertainty using parallel computing resources that is the dominant computer architecture available today.