OpenSees

OpenSees Workshop

http://opensees.berkeley.edu/OpenSees/workshops/OpenSeesWorkshop.pdf

Frank McKenna
UC Berkeley

NEES

PACIFIC EARTHQUAKE ENGINEERING RESEARCH

NSF

Outline of Workshop

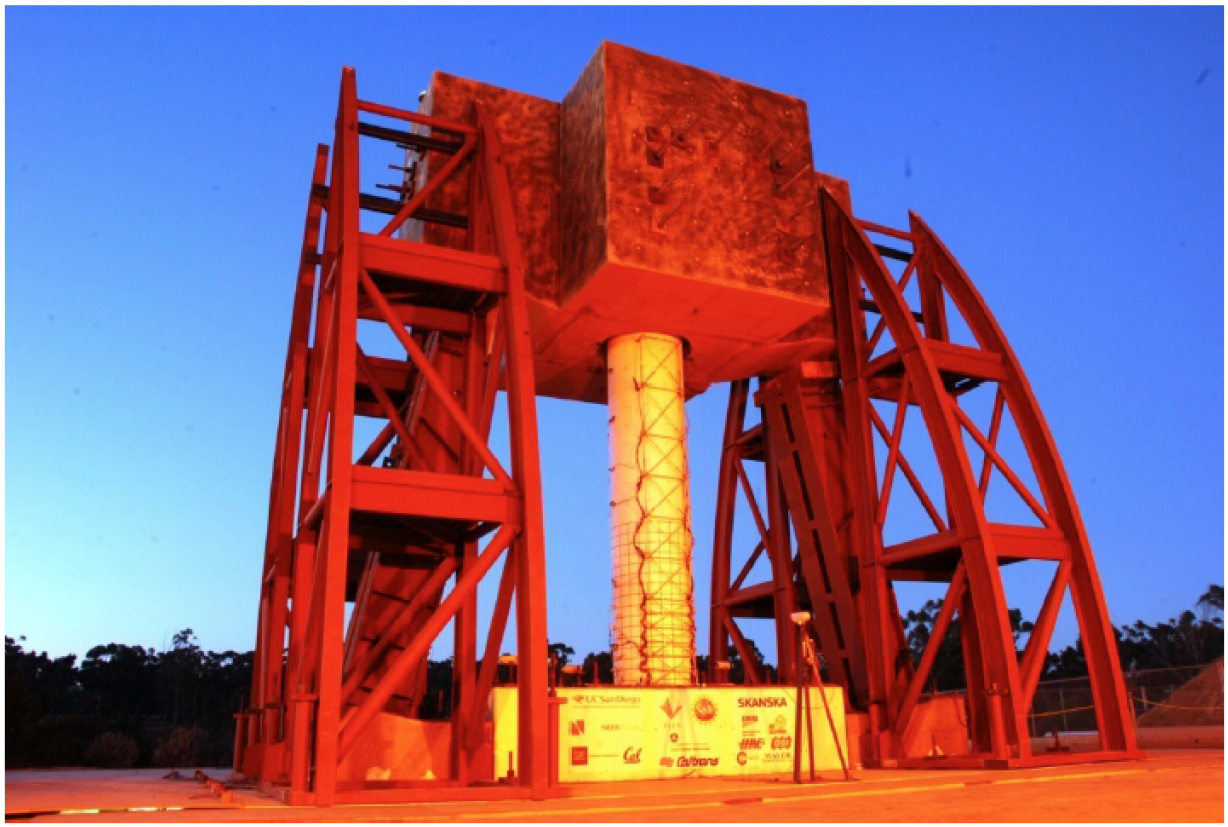
- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

What is Purpose of Simulation?

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

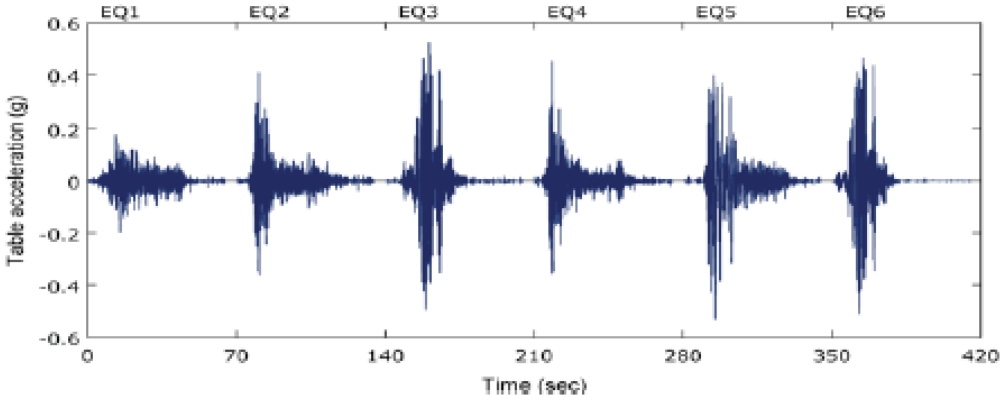
And Just How Good are we Today at predicting this response?

Today:
PEER Bridge Column Benchmark @ UCSD

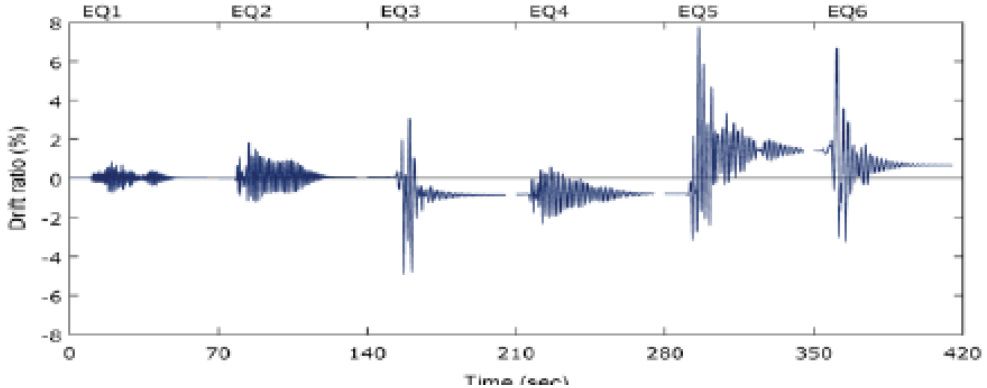


Today:
PEER Bridge Column Benchmark @ UCSD

- Earthquake acceleration time histories

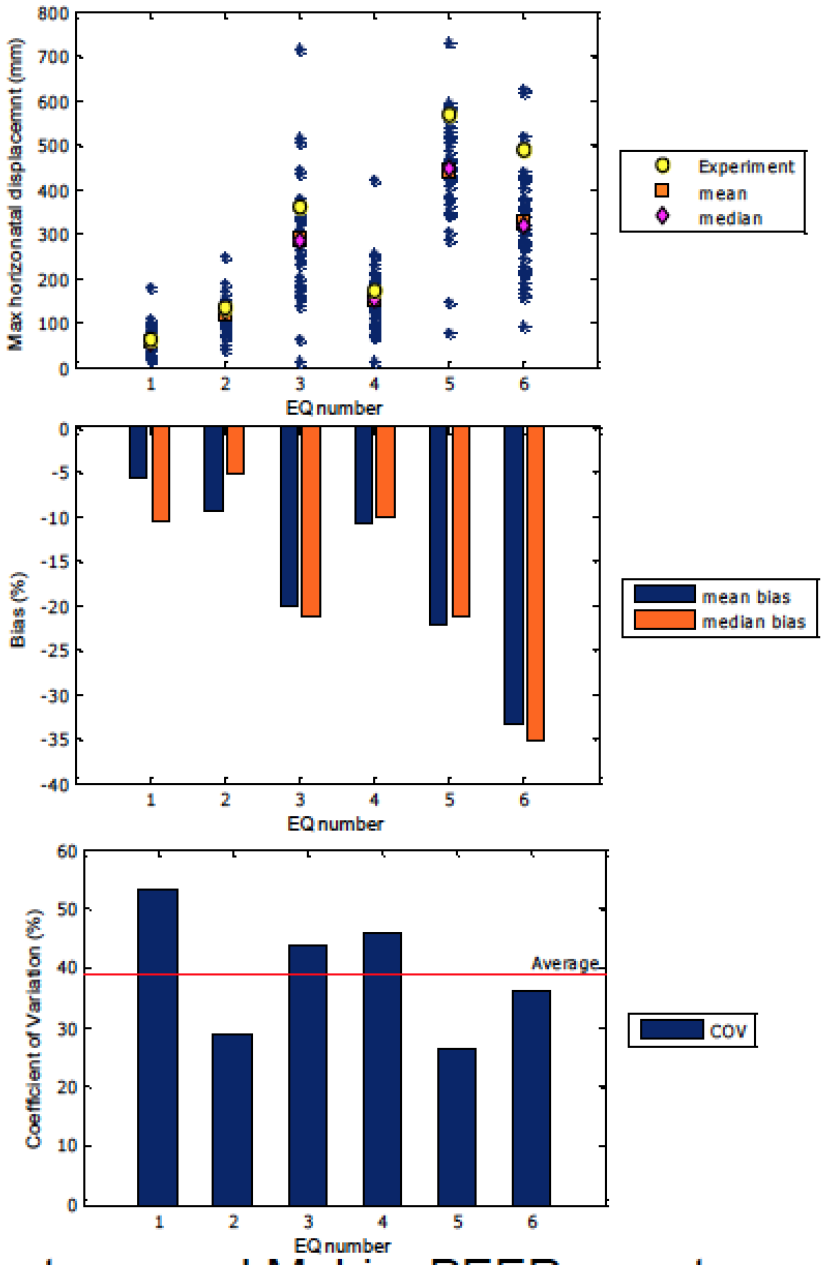


- Drift ratio time histories



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%

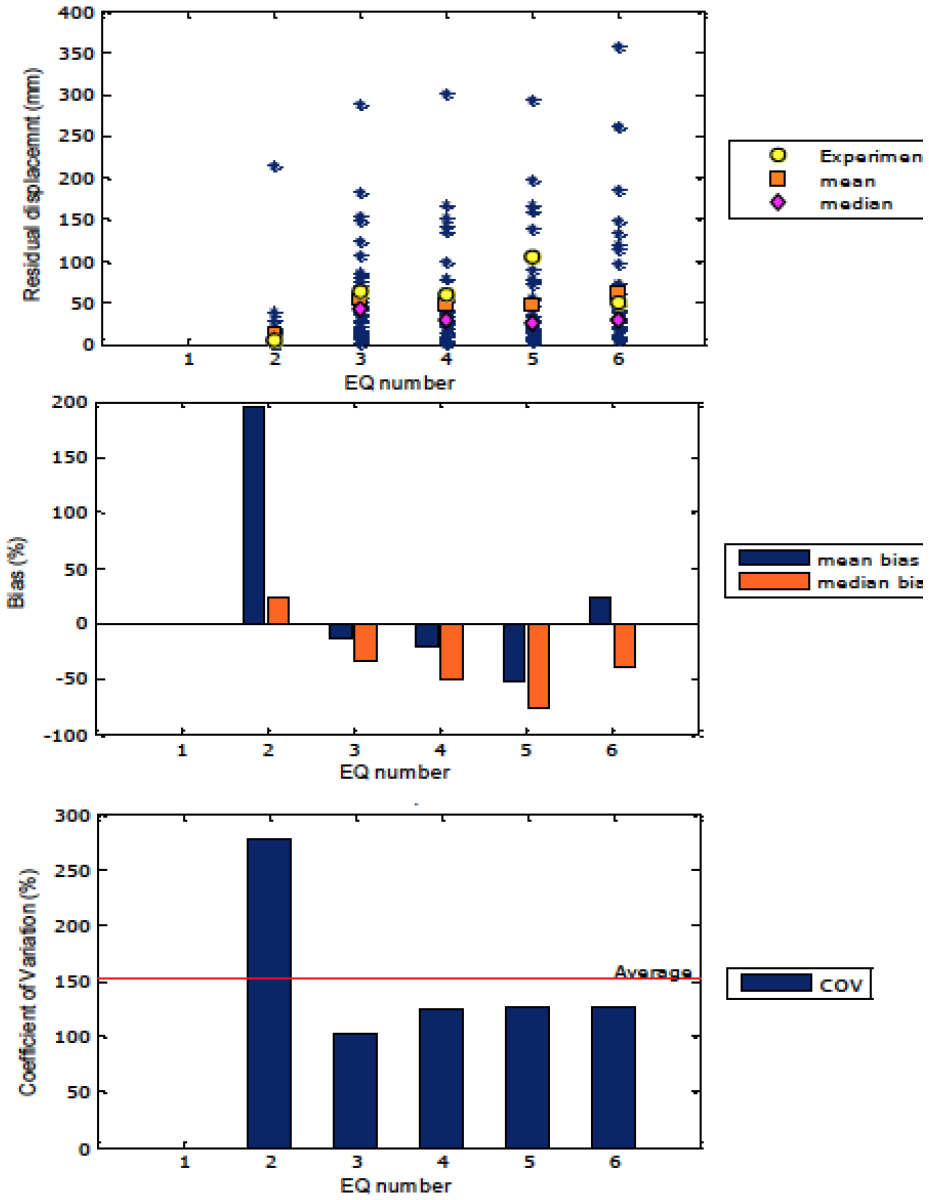


source: Prof. B. Stojadinovic

Terzic, Schoettler, Restrepo and Mahin, PEER report

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

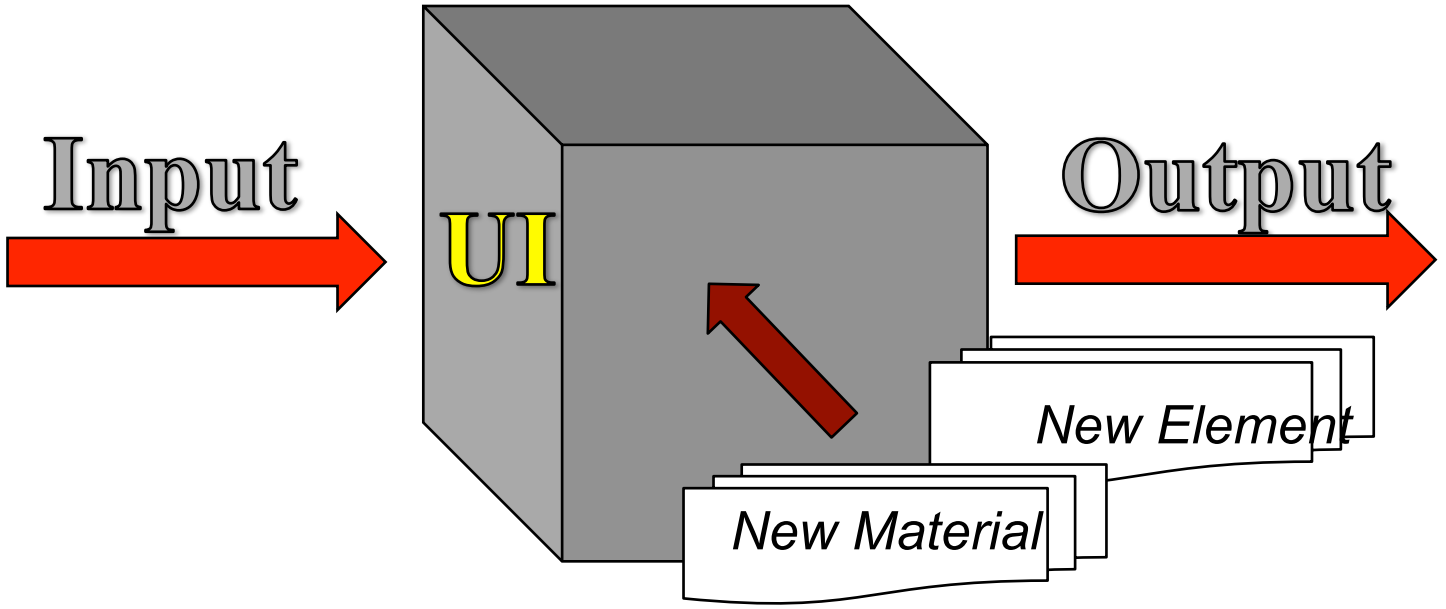
Terzic, Schoettler, Restrepo and Mahin, PEER report

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



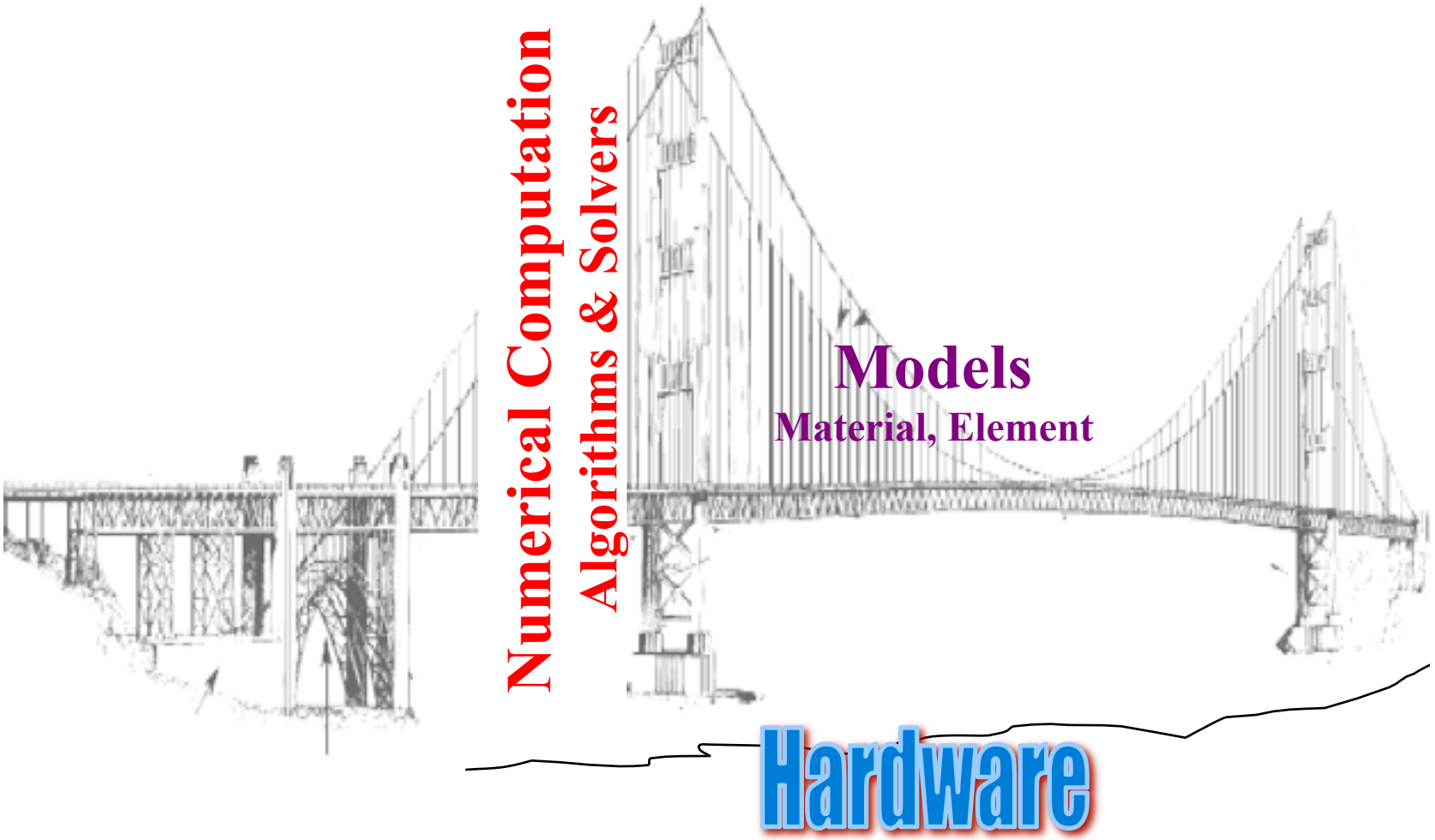
- Embedding of computational procedures in codes makes it difficult to experiment (integration schemes, algorithms, solvers) and take advantage of computing technology (Parallel & Grid Computing)
- "Closed-source" was/is the norm, whereas other fields have adopted "open-source" software for communities 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 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

Building Blocks for Modern Simulation Code



Open-Source - Leave it out there for community

PEER: OpenSees Goals (1998):

Provost Gregory Fenves
UT Austin

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.

Has It Worked:
Used Worldwide (2013)

Visitors

83,663

% of Total: **100.00% (83,663)**

Visits

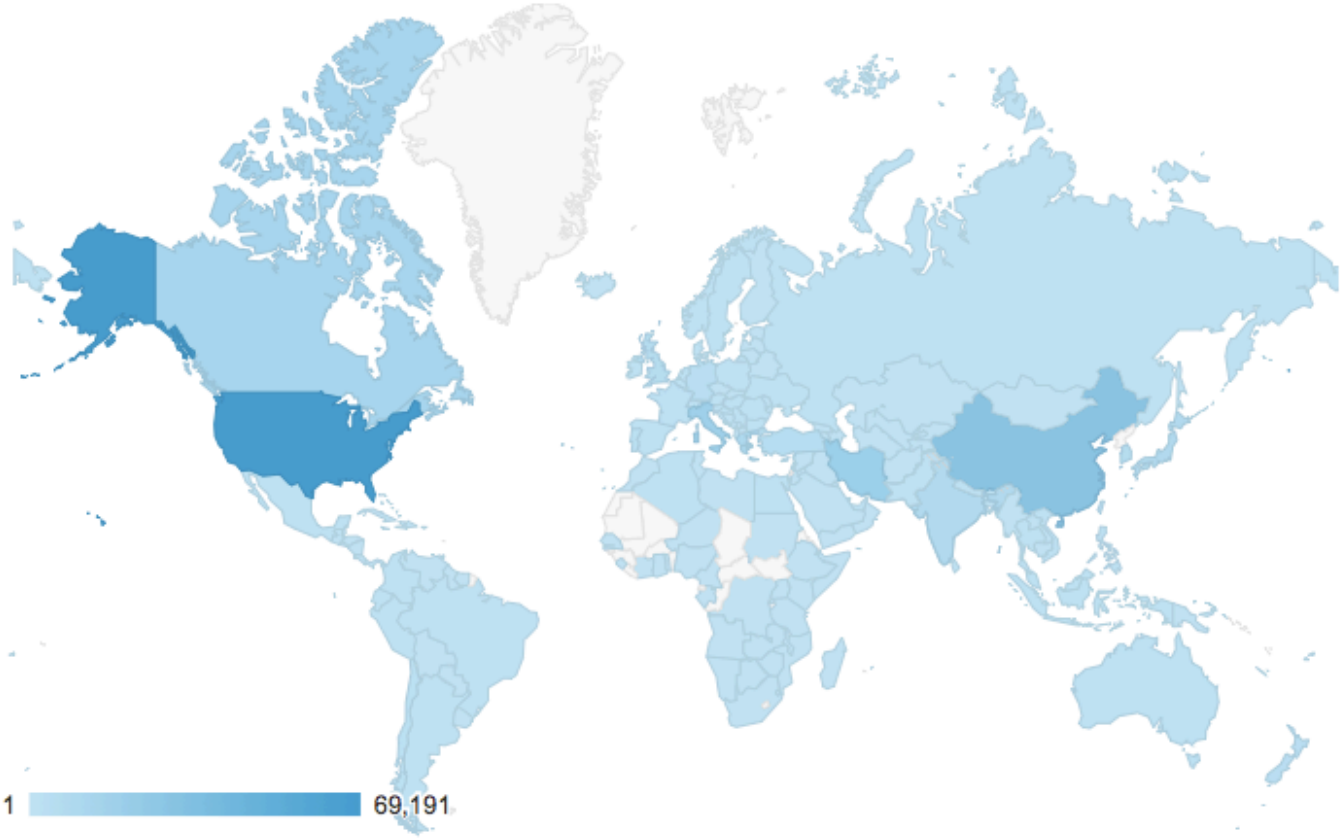
249,630

% of Total: **100.00% (249,630)**

Pageviews

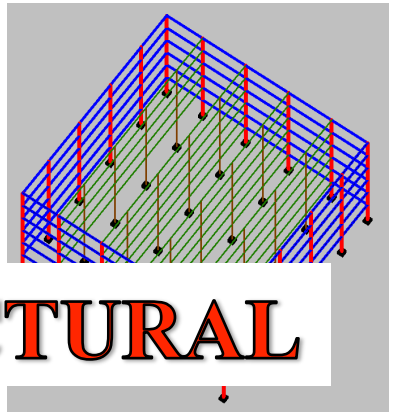
1,492,715

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

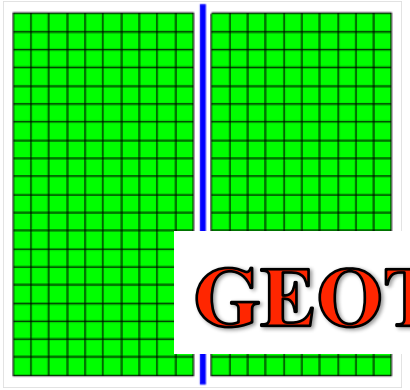


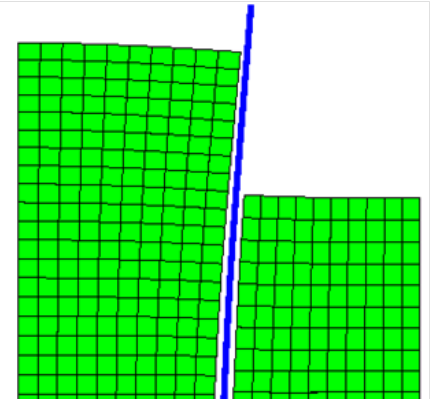
1. United States
2. China

Example Usage



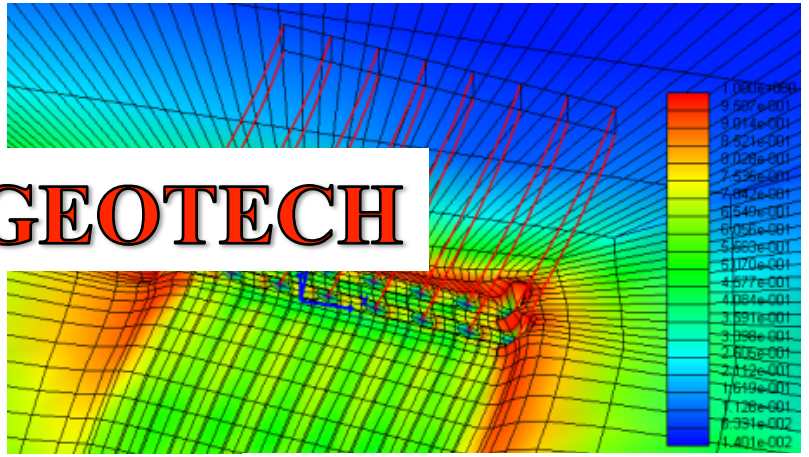
STRUCTURAL

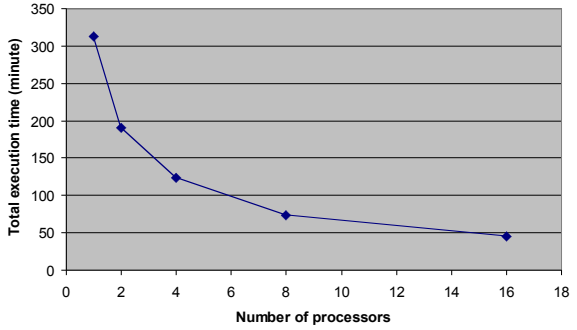




GEOTECHNICAL

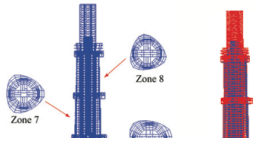
STRUCTURES + GEOTECH

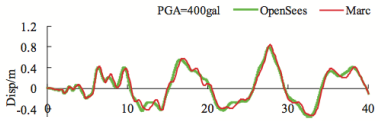


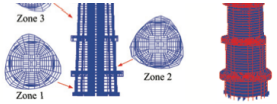


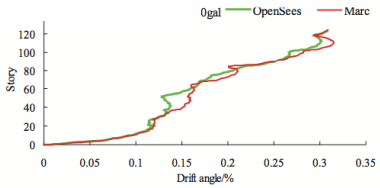
Comparison/Validation

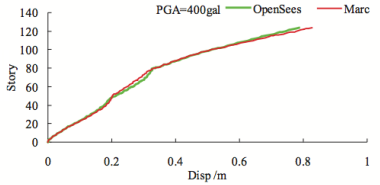
TBI Building 2 &
Shanghai Tower

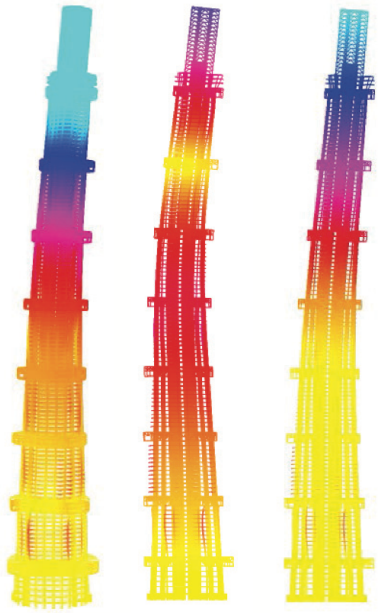






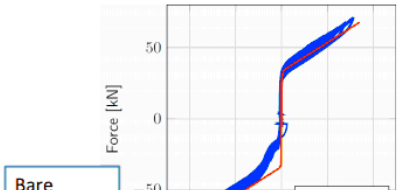






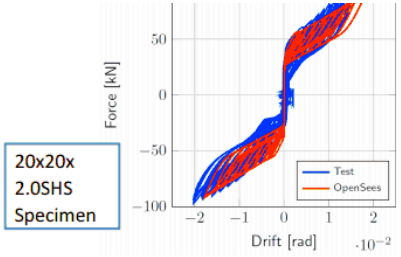
Shanghai Tower

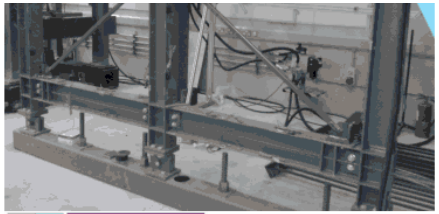
- H=632m, 124 stories
- 53,006 nodes
- 88,089 elements
- 48,774 fiber beam elements
- 39,315 multi-layer shell elements
- Memory used: 8.9GB



- 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

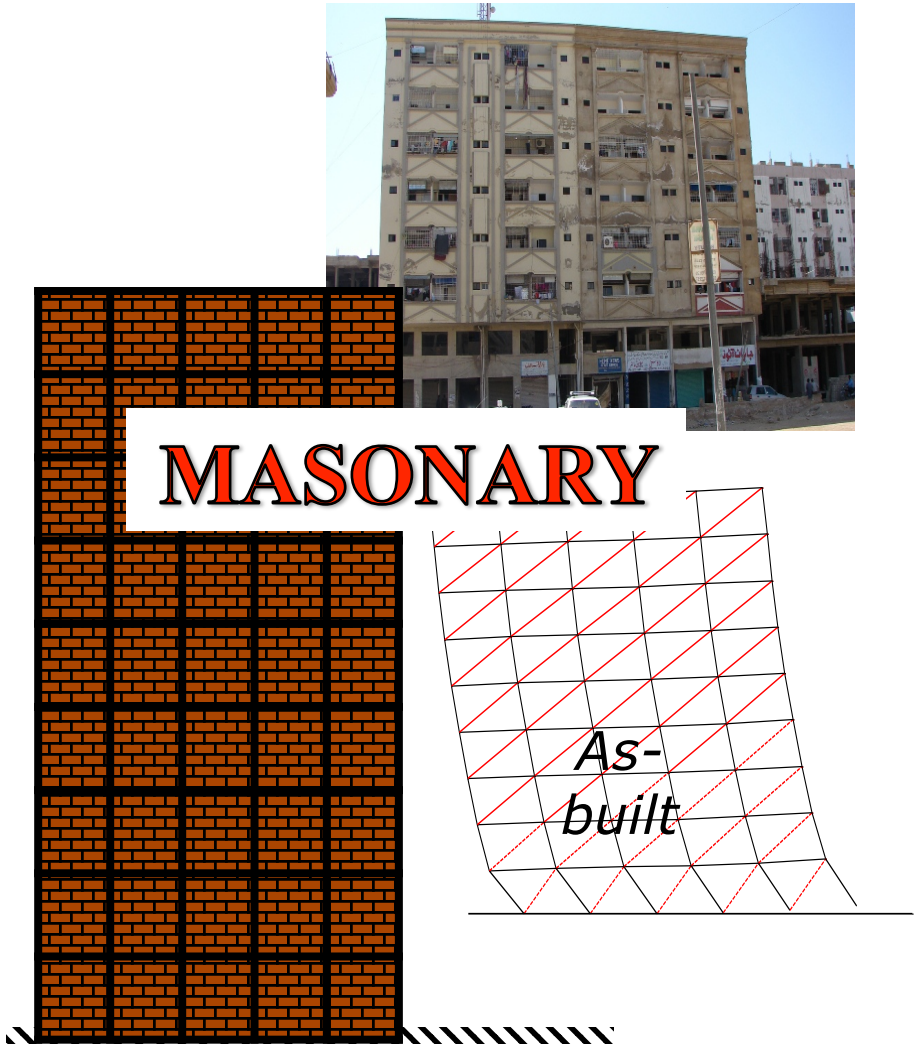
Verification/Calibration/Validation





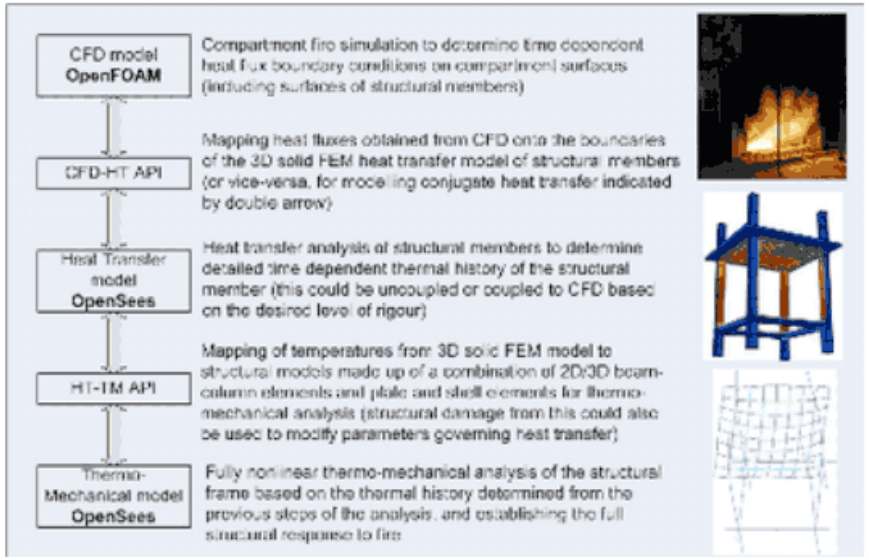
NUI Galway
OÉ Gaillimh

IRISH RESEARCH COUNCIL
COMHAIRLE TAIGHDE na hÉIREANN



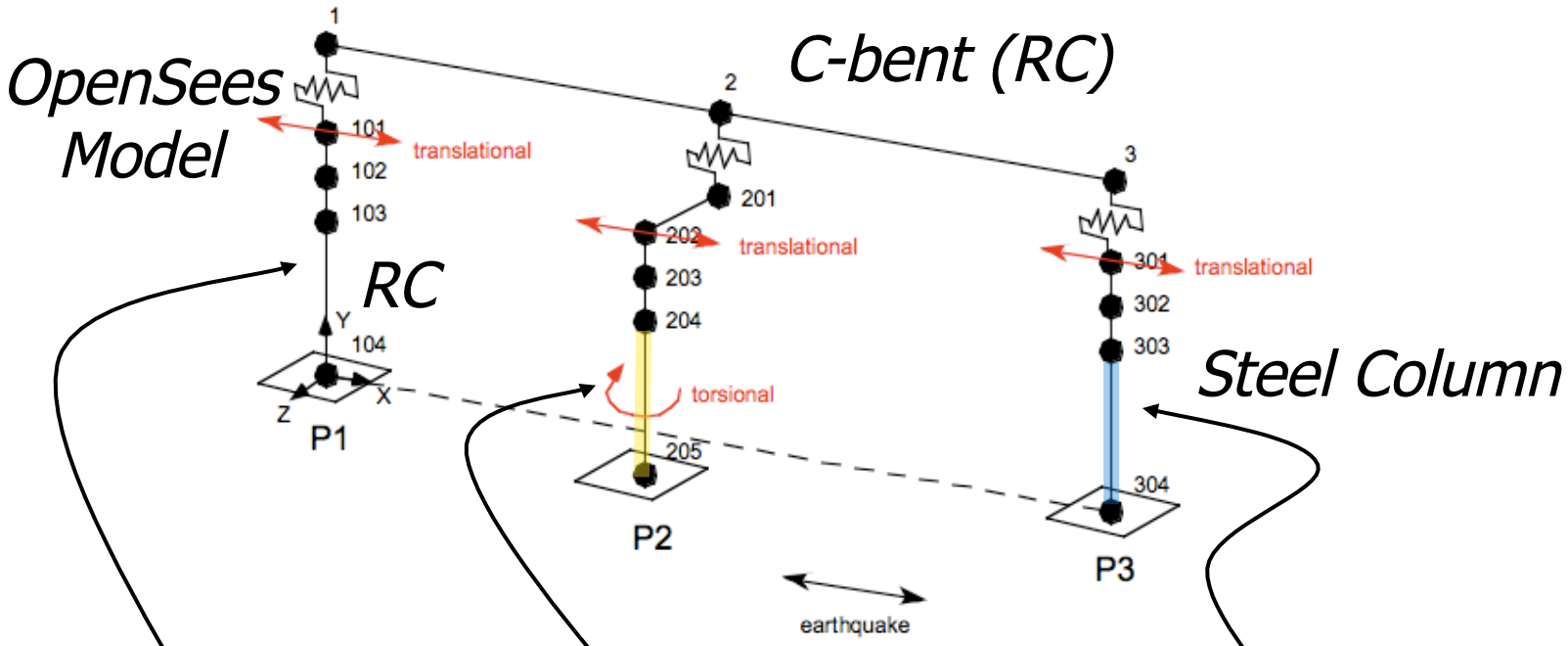
M.Talaat & K.Mosalam

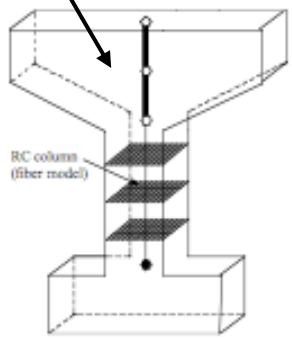
FIRE



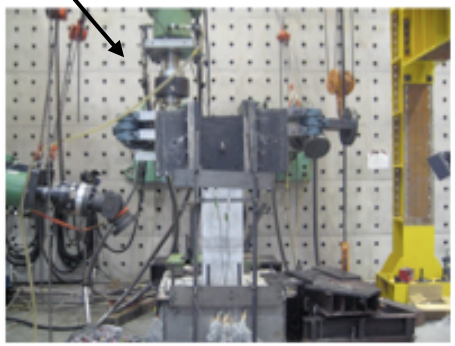
A. Usmani et al. , University of Edinburgh

HYBRID SIMULATION

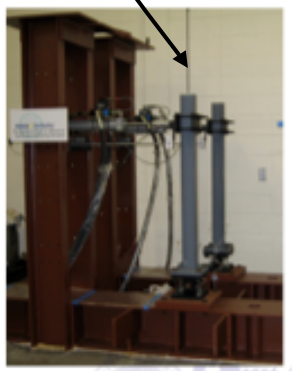




Numerical Component
at Kyoto University



Experimental Component
at Kyoto University



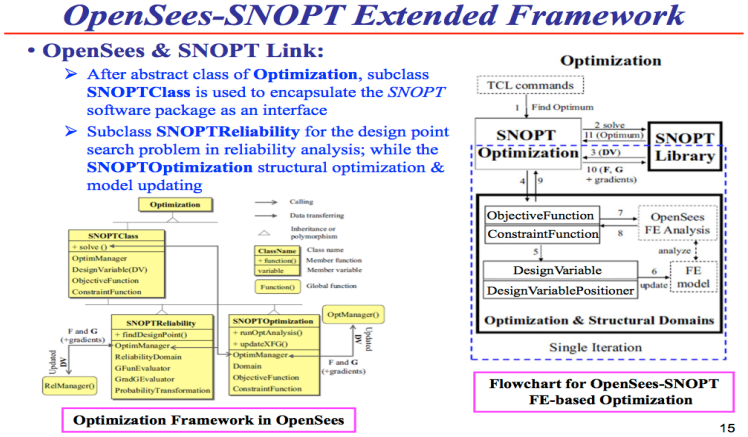
Experimental Component
at nees@berkeley

Kyoto/Berkeley

New Directions:

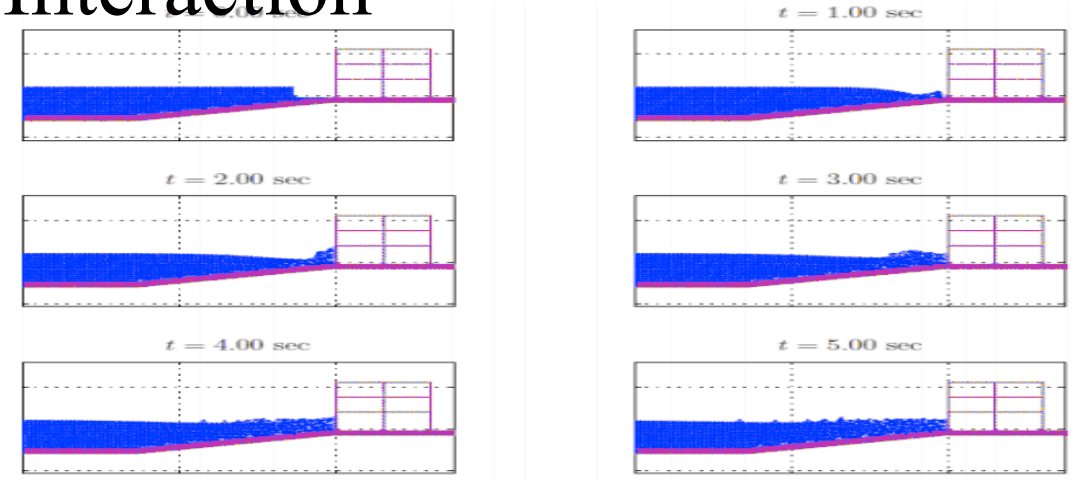
1. Optimization

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



2. Fluid Structure Interaction

M.Scott, Oregon State



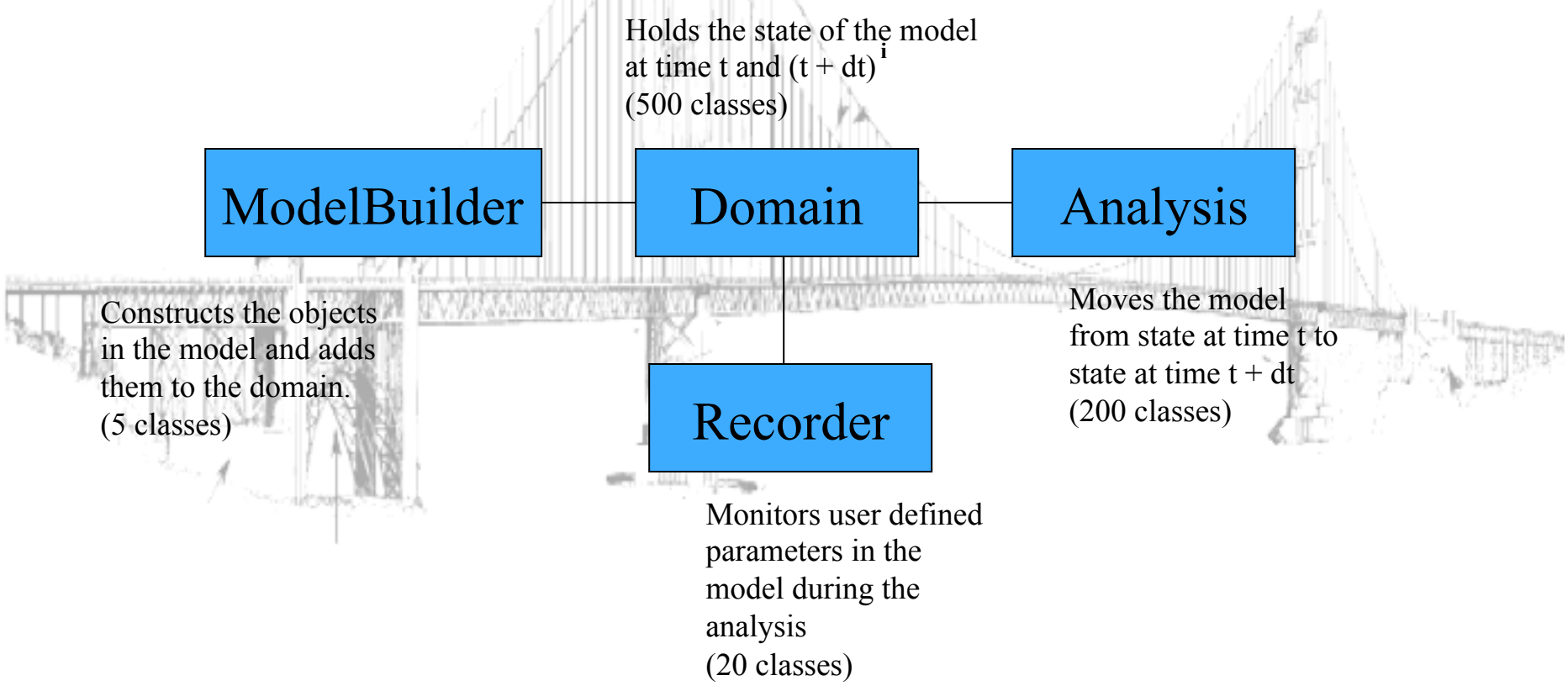
SO What is OpenSees?

- 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.

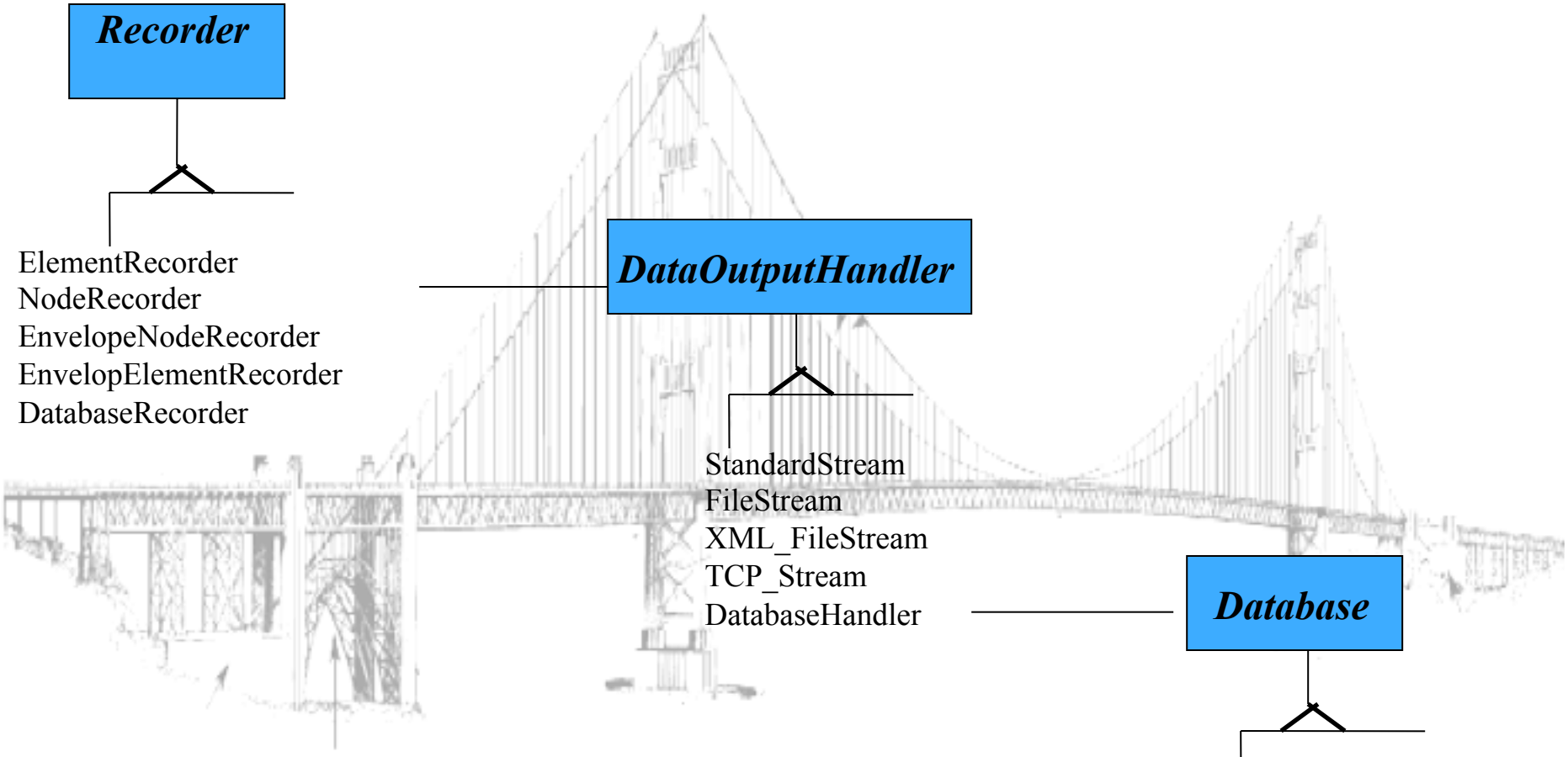
OpenSees is a Software Framework

- 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.

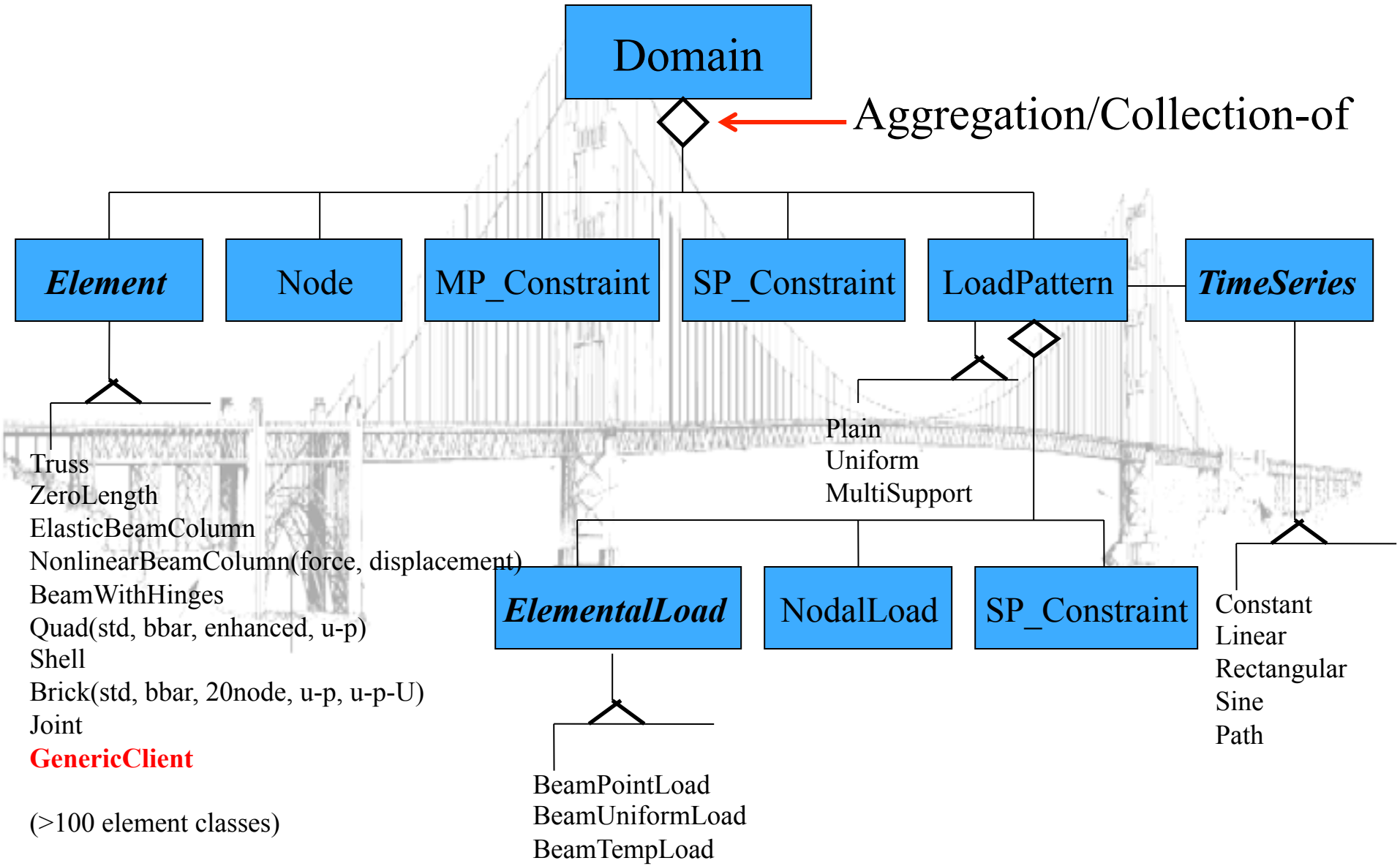
Main Abstractions in OpenSees Framework



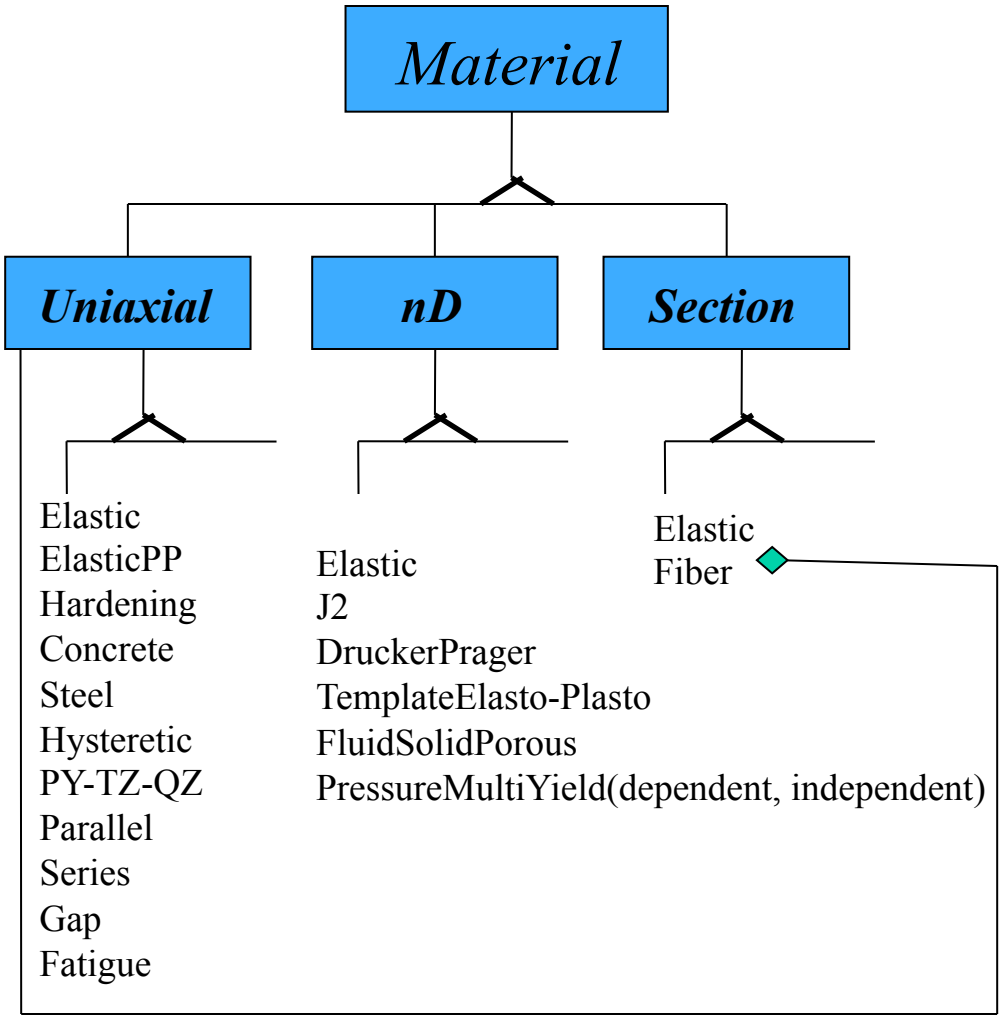
Recorder Options



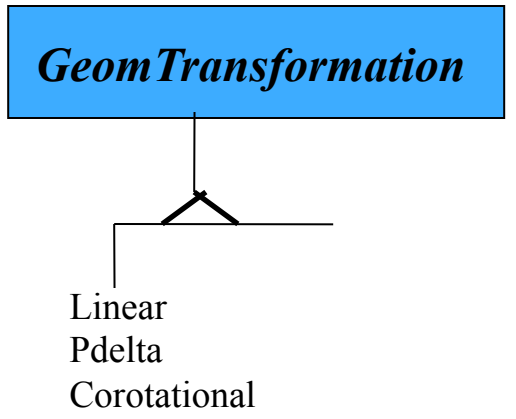
What is in a Domain?



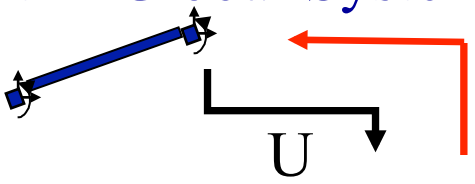
Some Other Classes associated with Elements:



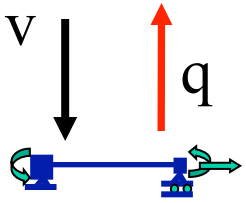
(over 250 material classes)



Element in Global System

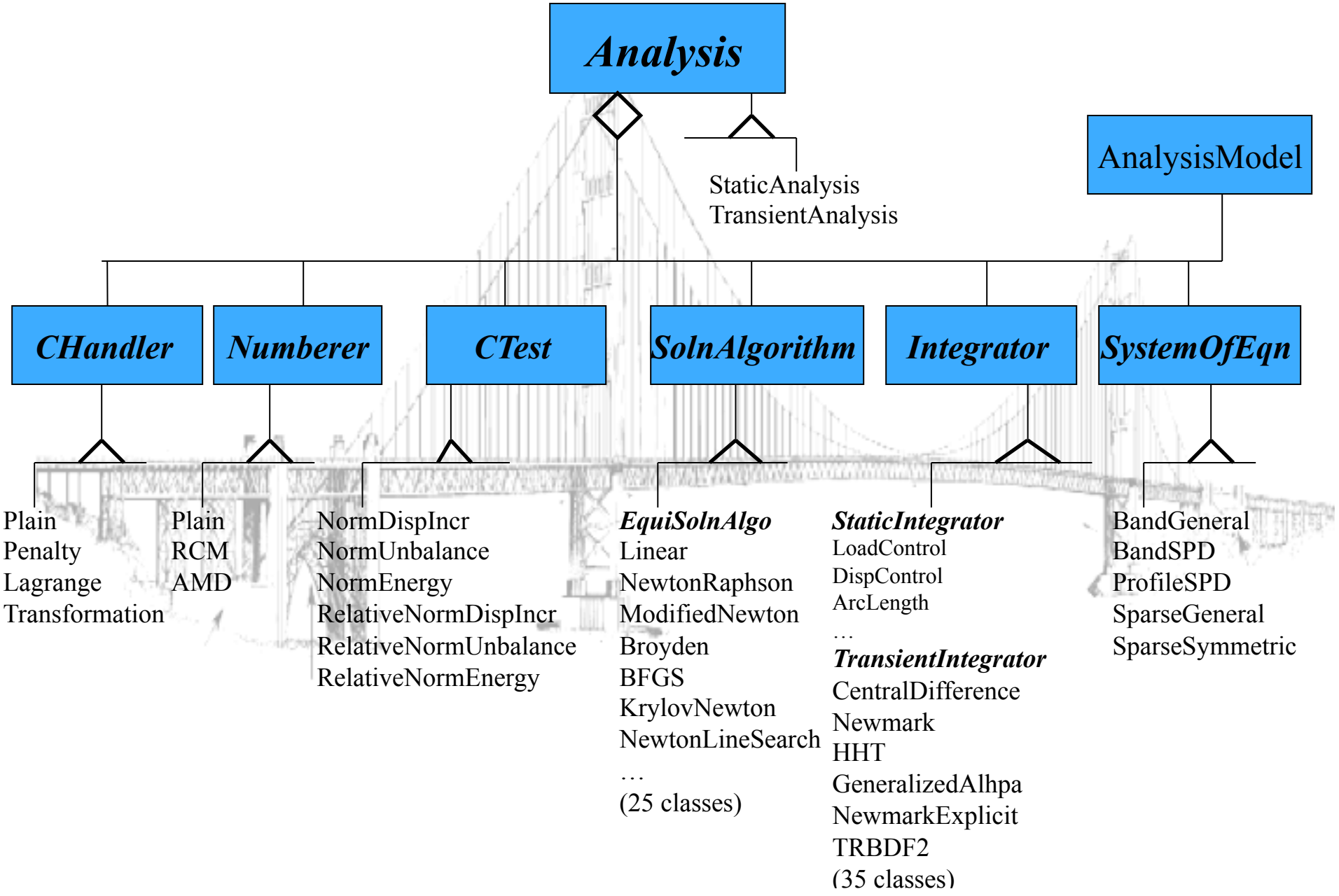


Geometric Transformation



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?

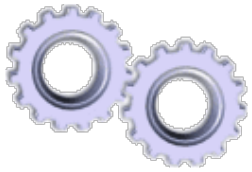
- Provide their own main() function in C++ and link to framework.

- 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. OpseseesSP.exe
    4. OpenSeesMP.exe

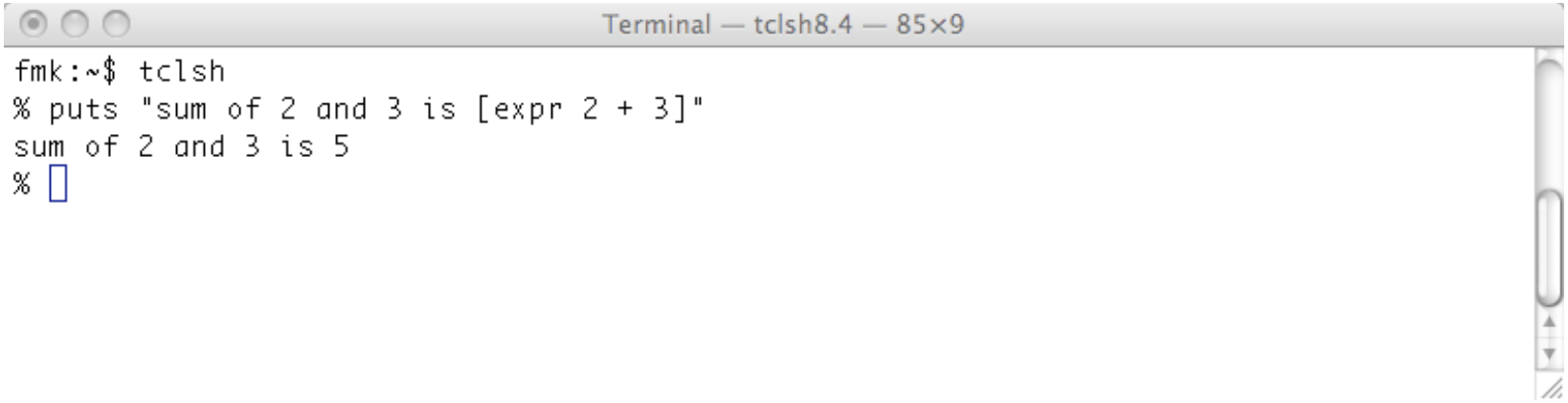
Tcl Interpreters

- **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



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
```

➢ 45

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
```

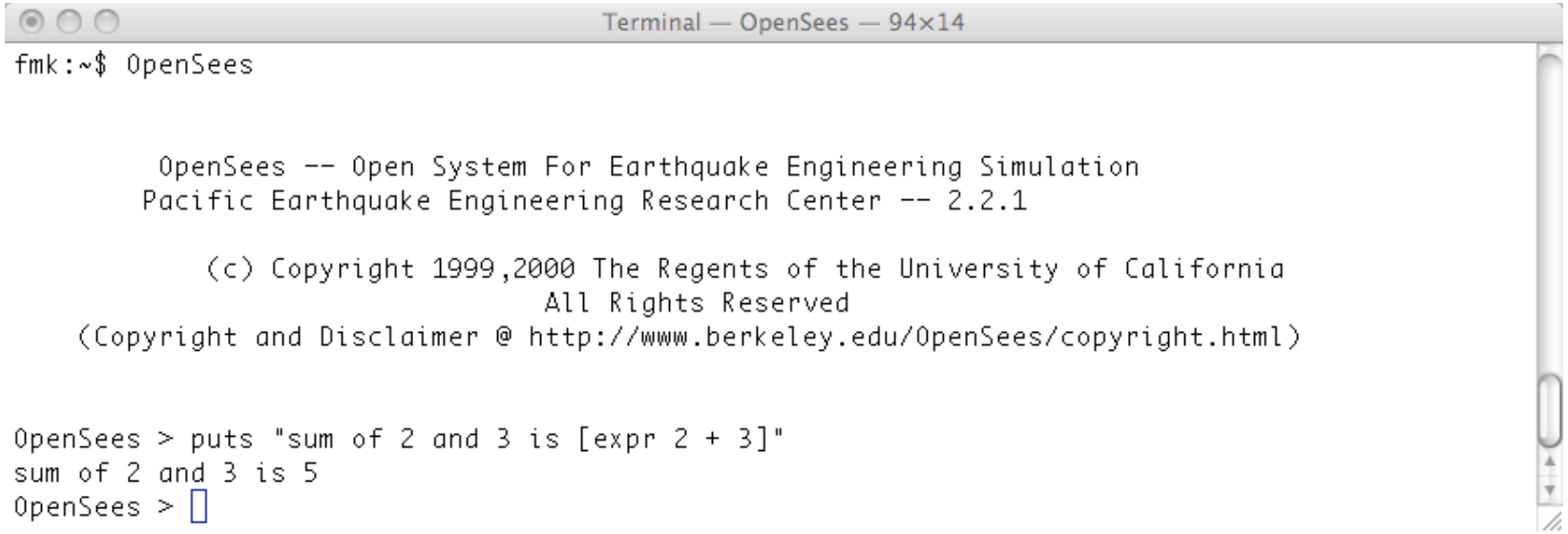
OpenSees Interpreters

- 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.

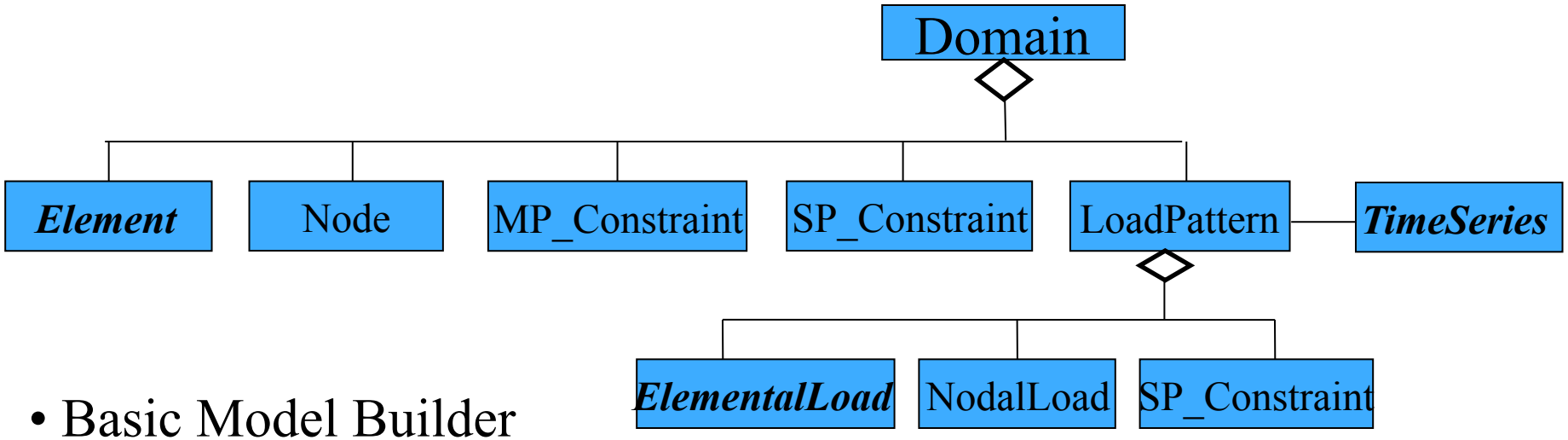
OpenSees.exe

- An interpreter that extends tclsh for FE analysis.



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:

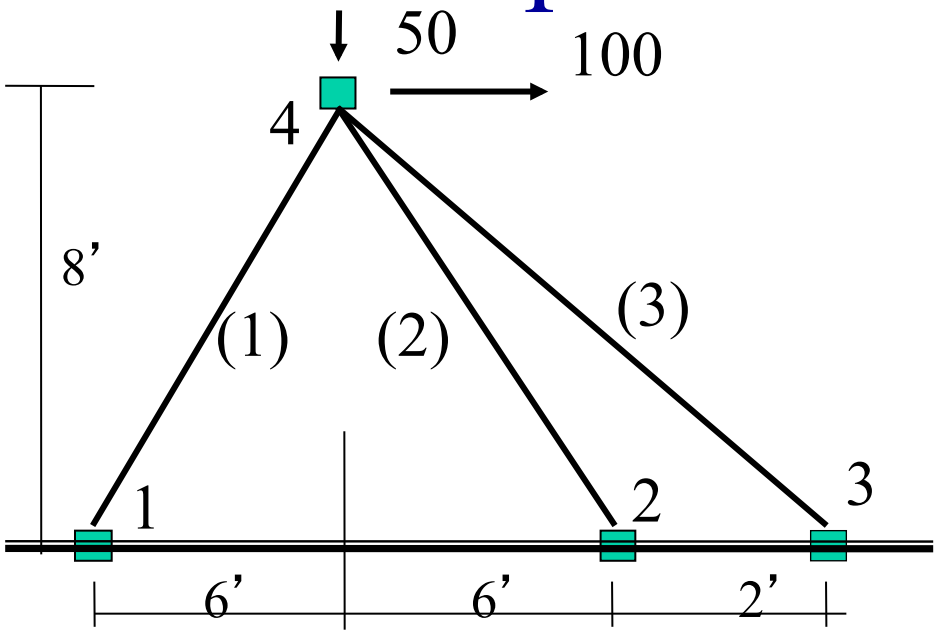
node mass element equalDOF fix fixX fixY fixZ

pattern timeSeries load eleLoad sp

uniaxialMaterial nDMaterial section geomTransf

fiber layer patch block2D block3D

Truss example:



	E	A
1	3000	10
2	3000	5
3	3000	5

```
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

#pattern Plain $tag $tsTag
pattern Plain 1 1 {
   #load $nodeTag $xFrc $yFrc
   load 4 100.0 -50.0
}
```

When should you Use Variables 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 wi be repeatedly used.

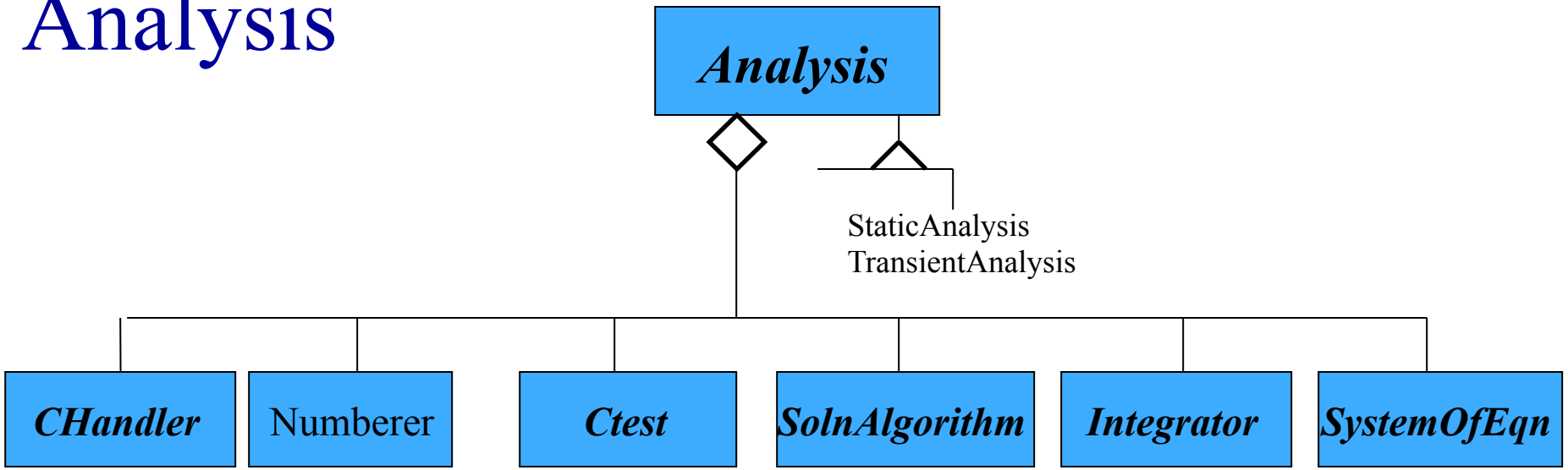
Truss example using variables:

```
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
set patternTag1
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
 }
```

DON'T DO THIS or STOP DOING THIS

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

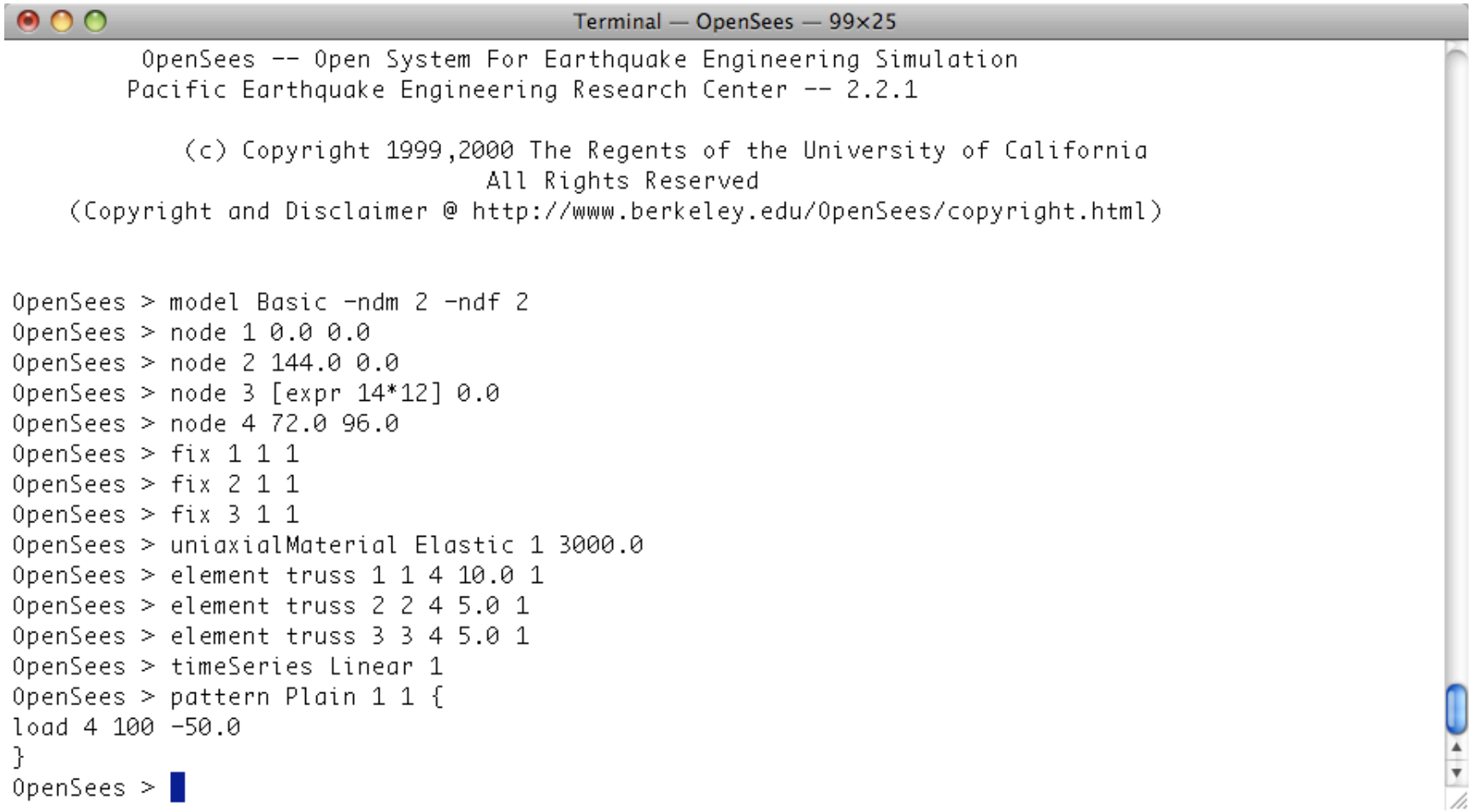
```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
analyze 2000 0.01
```

Commands That Return Values:

- analyze command — *set ok [analyze numIter <Δt>]*
- 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 ...]*

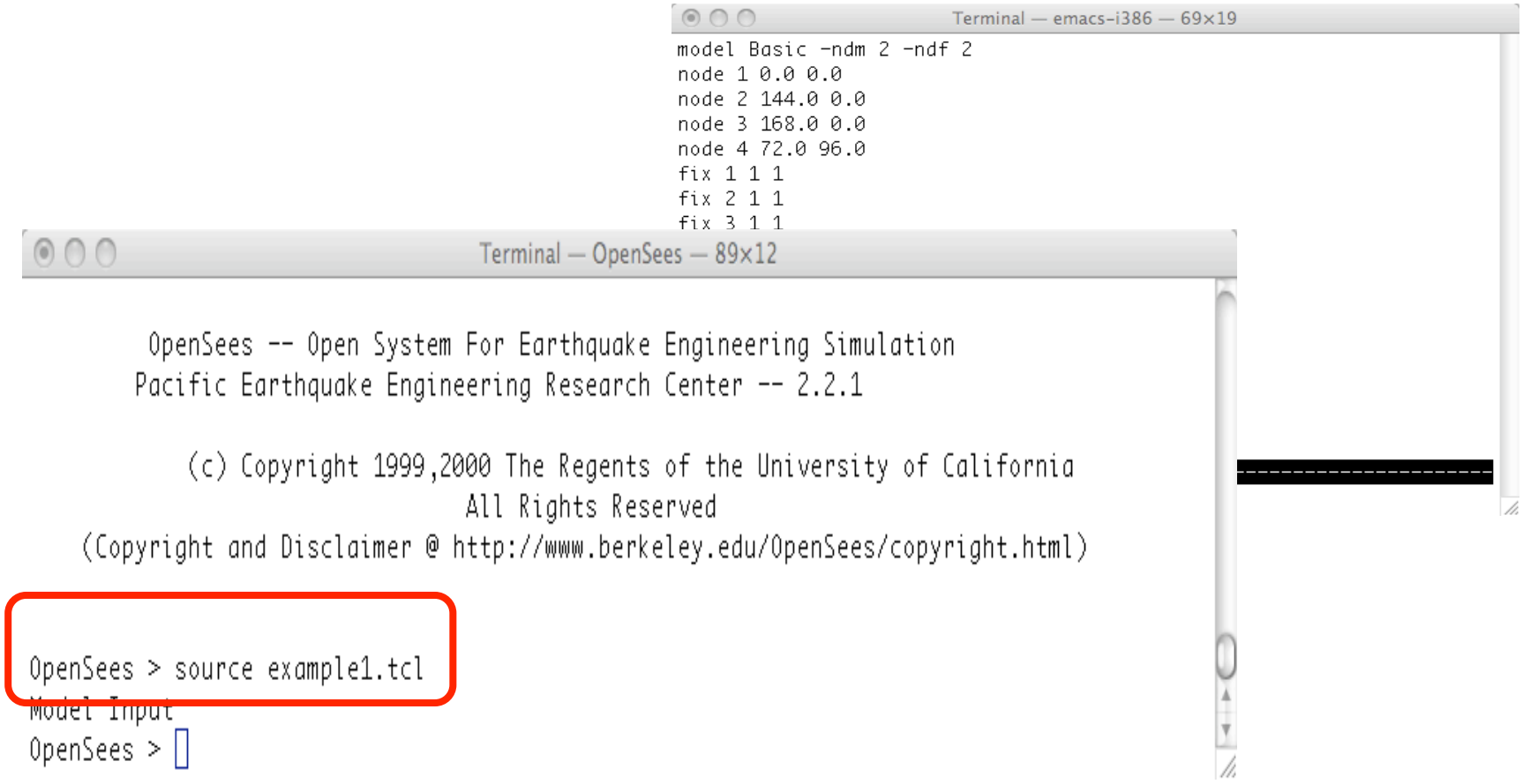
3 Ways to Execute the commands

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



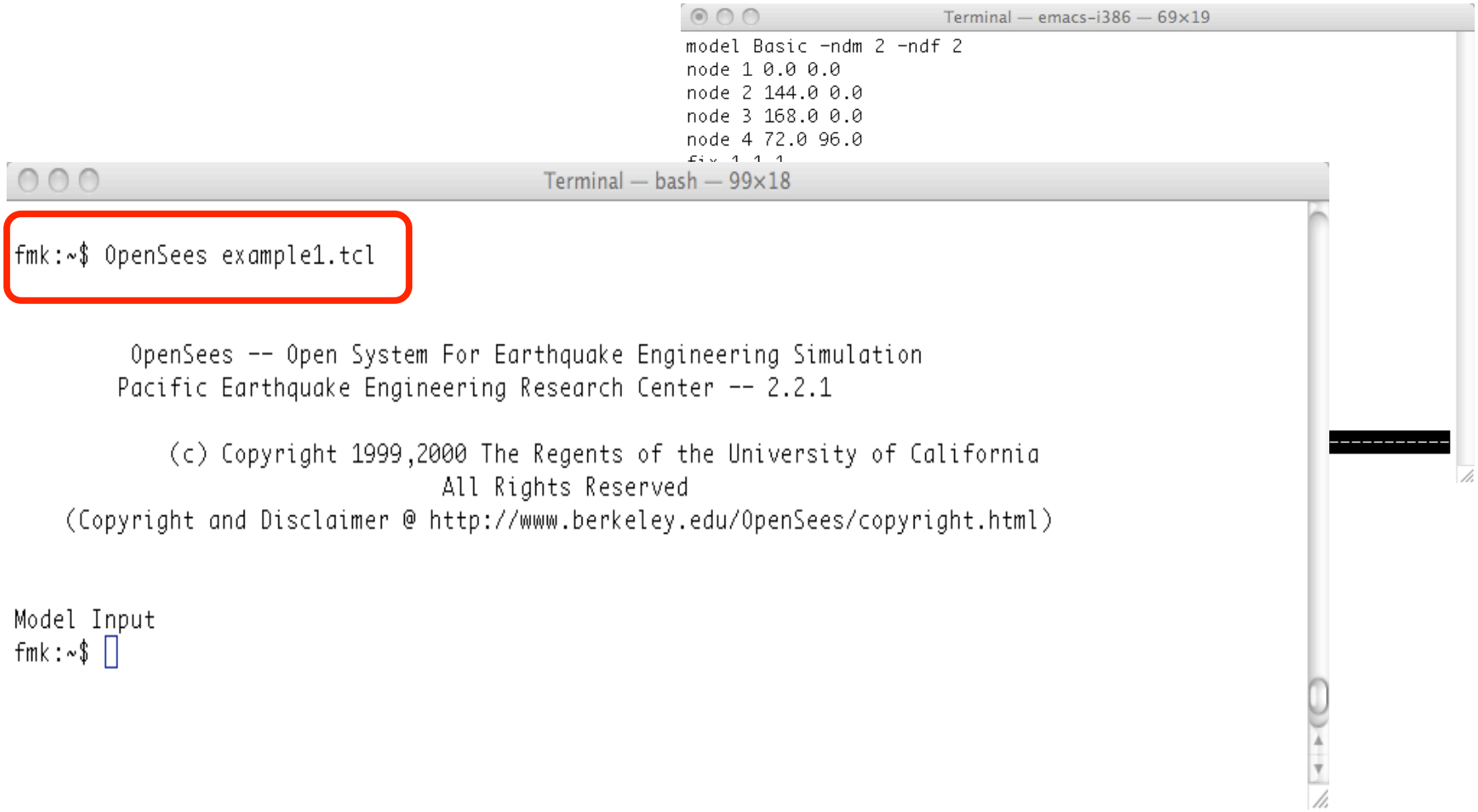
3 Ways to Execute the commands

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



3 Ways to Execute the commands

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



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

Terminal — emacs-i386 — 101×39

```
#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 — 101×39

```
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              1.8750000000000000000            -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.berkely.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/

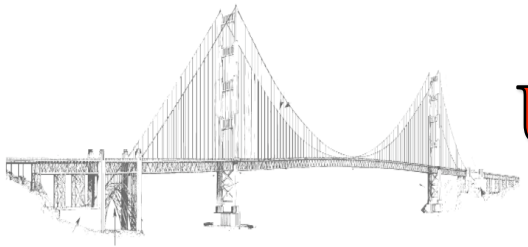
Outline of Workshop

- 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)

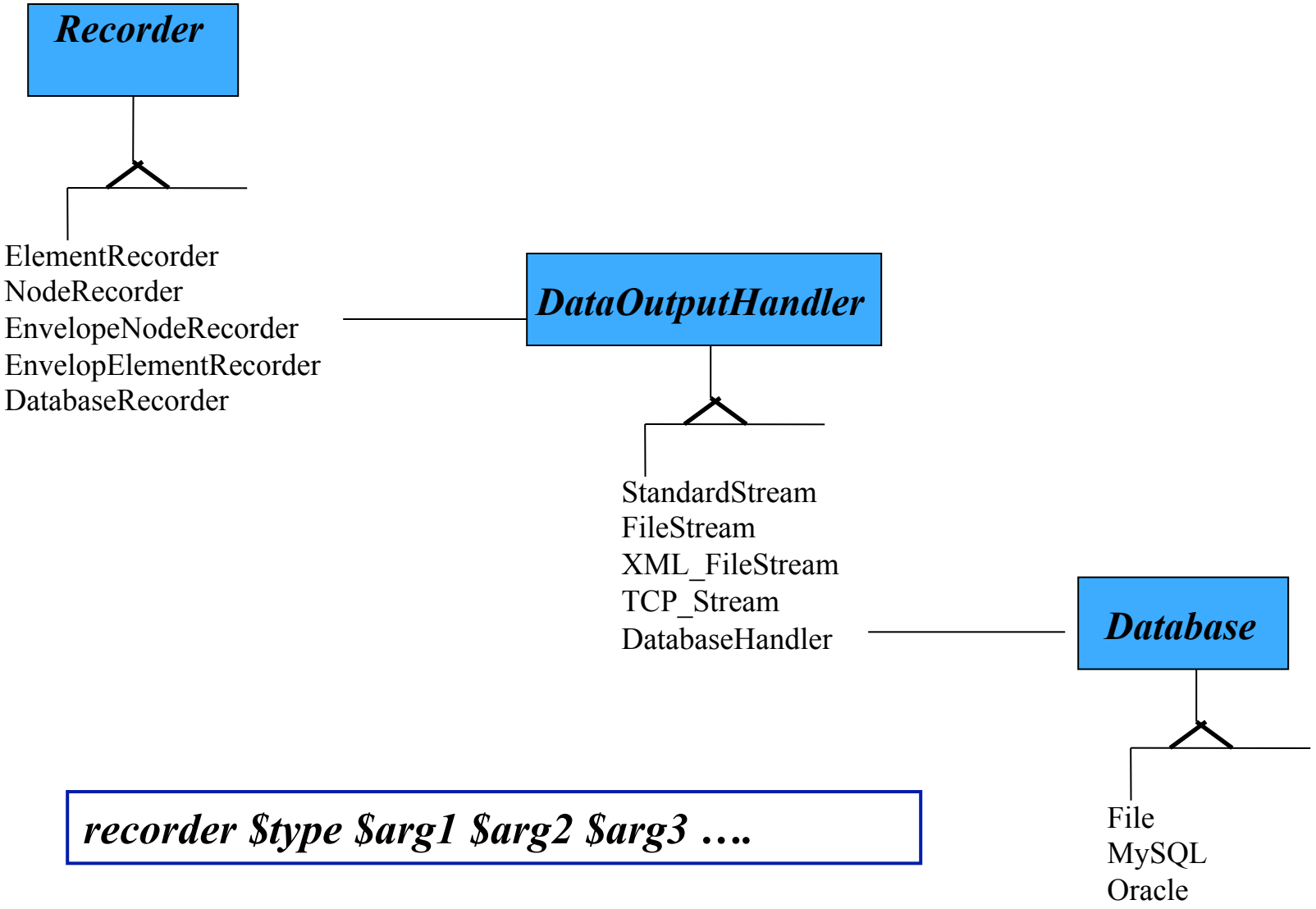
```
# create model & analysis
 ...

# 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
```

```
Terminal — bash — 110×13
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 — 59×10
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)--------
```

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

Recorder Options



recorder $type $arg1 $arg2 $arg3 ….

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>

Node/EnvelopeNode Recorders

- To monitor what's happening at the Nodes.

recorder Node <-file $fileName><-timeSeries $tsTag> <-time> <-node $tg1 $tg2 …> -dof $d1 $d2 .. disp
<-xml $fileName> <-nodeRange $tgS $tgE> vel
<-binary $fileName> <-region $rTag> accel
<-tcp $inetAddr> incrDisp
reaction

Example:

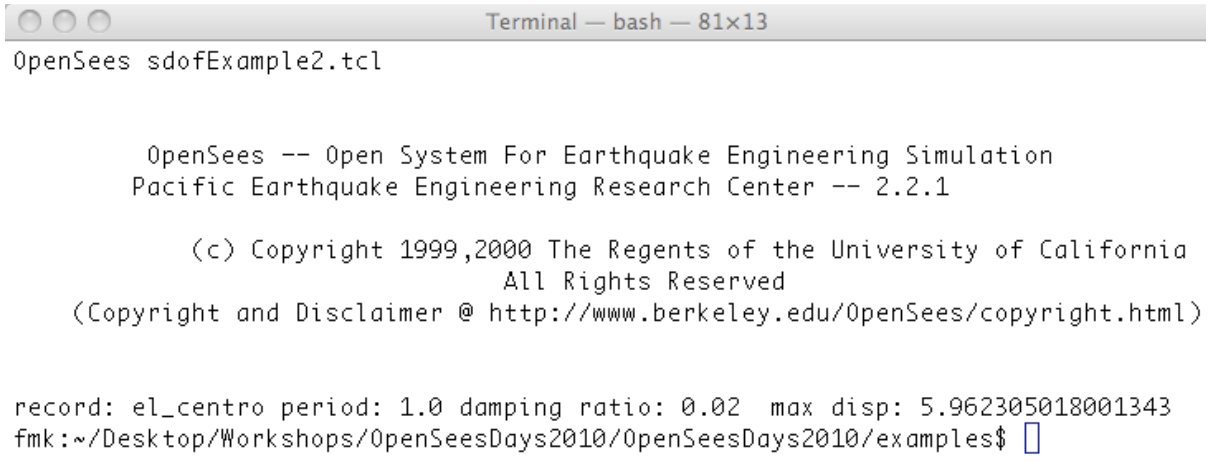
recorder Node -file nodeD.out -node 2 -dof 1 2 3 disp

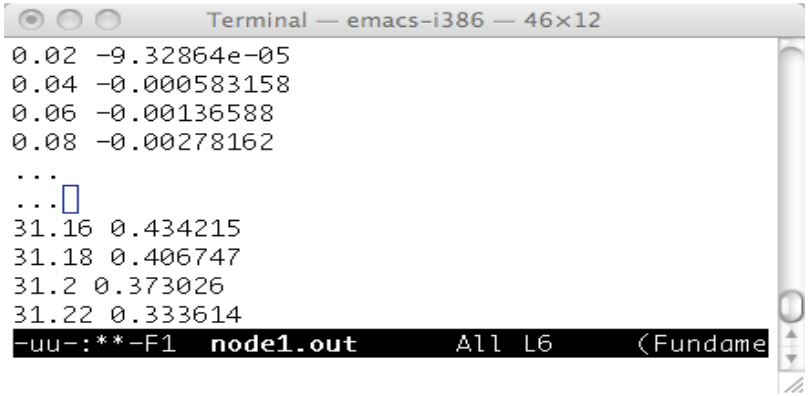
recorder Node -file nodeA.out -temeSeries 1 -node 2 -dof 1 accel

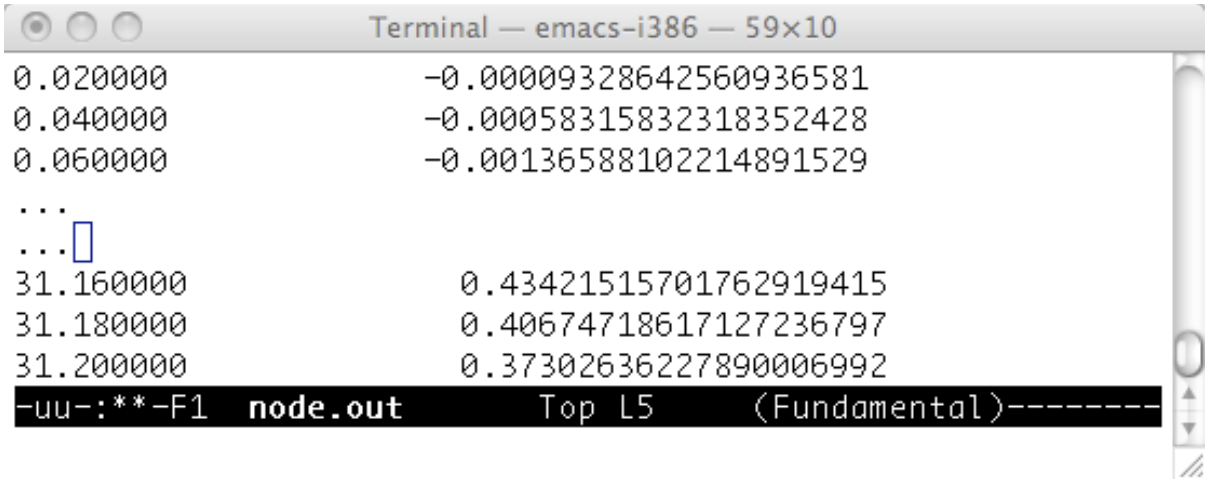
recorder EnvelopeNode <-file $fileName><-timeSeries $tsTag> <-time> <-node $tg1 $tg2 …> -dof $d1 $d2 .. disp
<-xml $fileName> <-nodeRange $tgS $tgE> vel
<-binary $fileName> <-region $rTag> accel
<-tcp $inetAddr> incrDisp
reaction

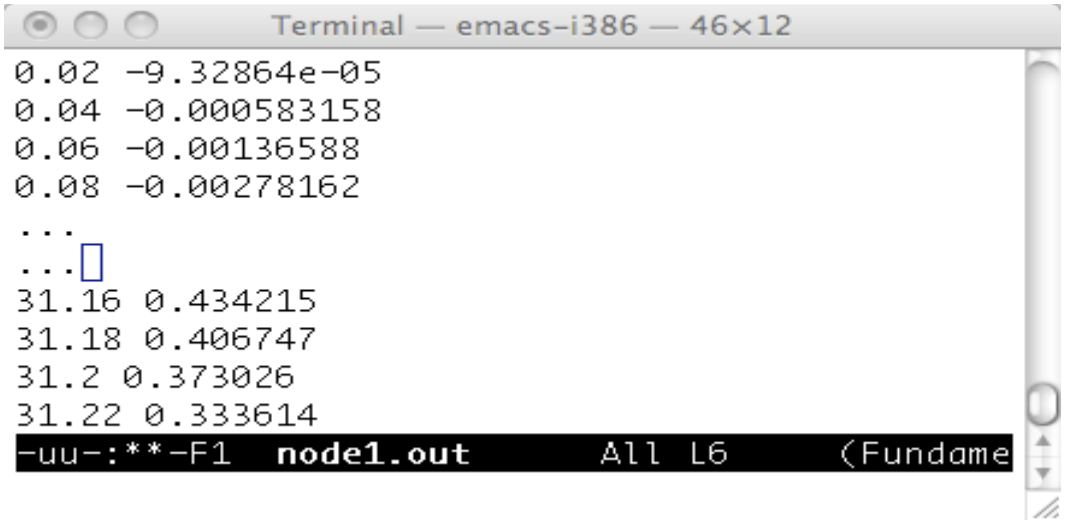
Example using recorders(sdofExample2.tcl)

```
# 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
```





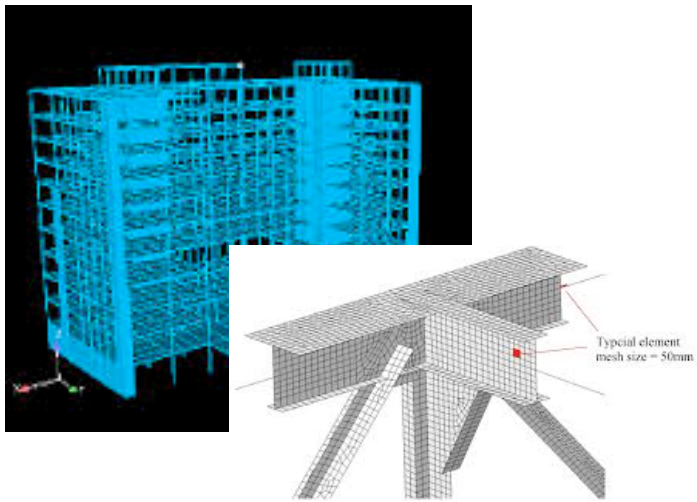

Terminal — emacs-i386 — 59×10
0.020000 -0.00009328642560936581
0.040000 -0.0005831583231835242 8
0.060000 -0.0013658810221489152 9
...
...
31.160000 0.43421515701762919415
31.180000 0.40674718617127236797
31.200000 0.37302636227890006992
-uu-:**-F1 node.out Top L5 (Fundamental)--------


Terminal — emacs-i386 — 46×12
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

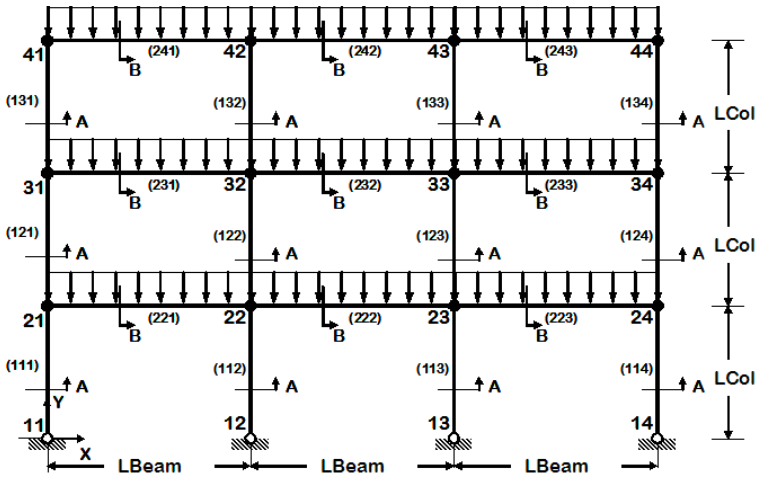
Outline of Workshop

- 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

To **develop a model** capable of producing a **credible prediction** of the **behavior of an existing/proposed structure. How complicated a model is required of our structure?**



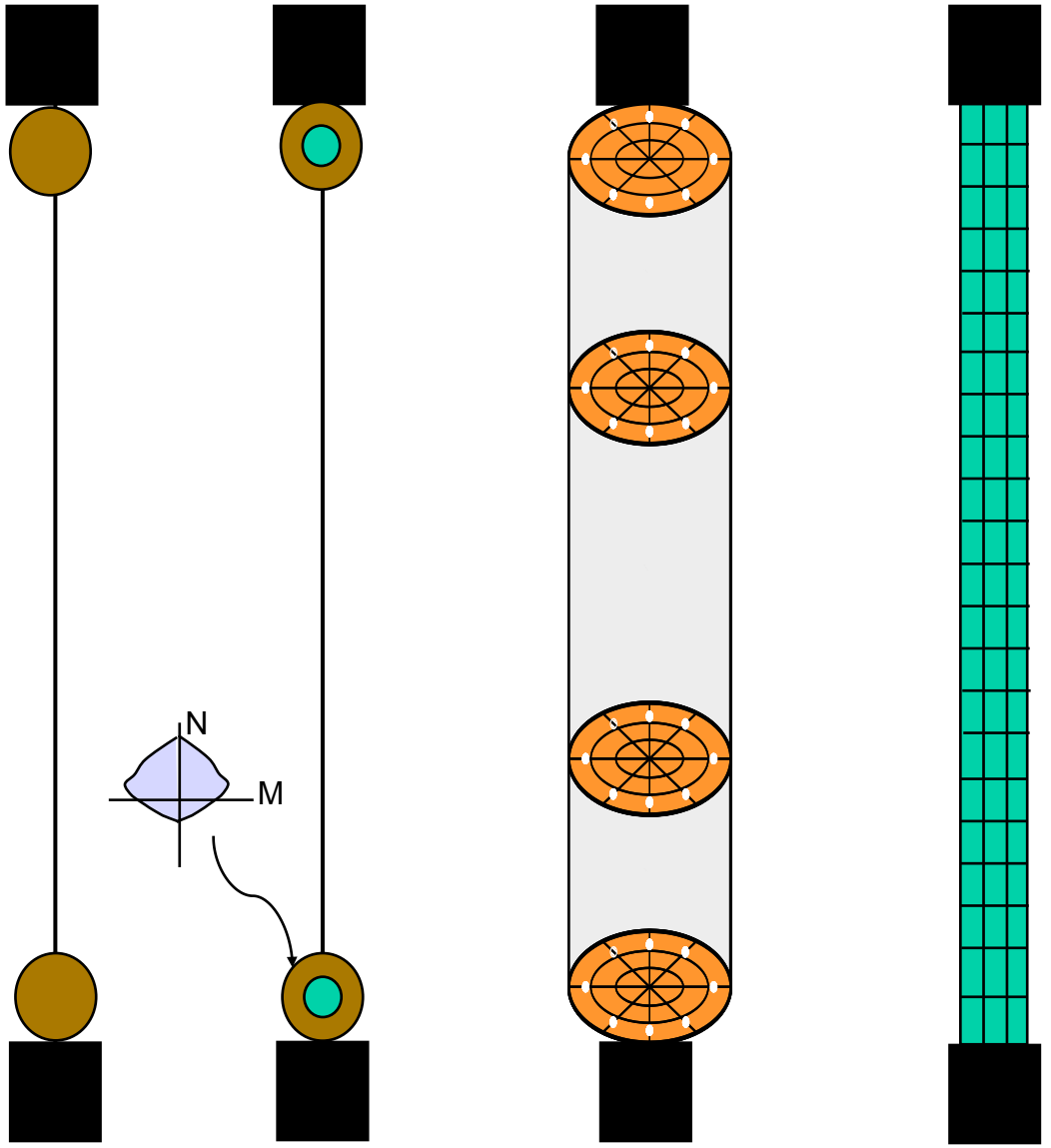
3d Continuum/Frame?



2d Frame?

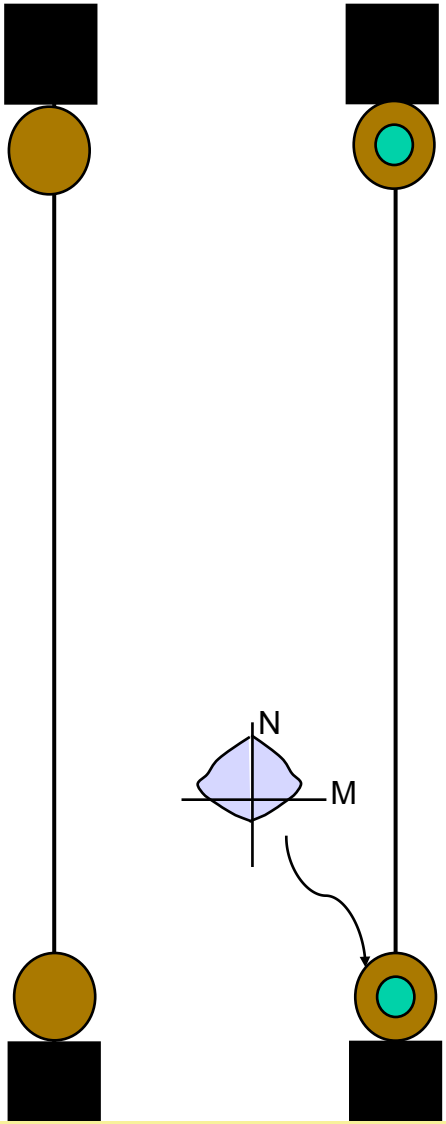
1d Lollipop?

Nonlinear Structural Beam-Column Models



Source: Filip Filippou, UC Berkeley

Beam-Column Models: Concentrated Plasticity

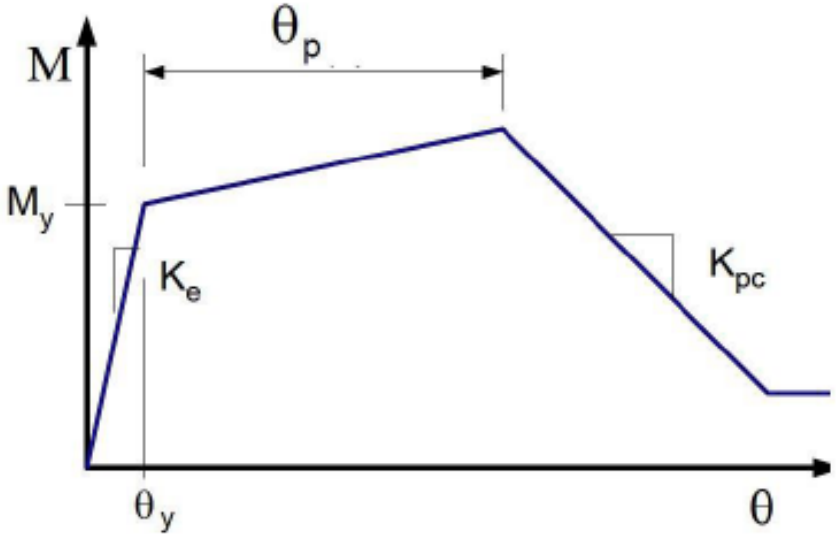


- Concentrated plasticity models =
  one or more rotational springs at each end +
  elastic element
  - Advantages:
    relatively simple, good(?) for interface effects (e.g. shear sliding, rotation due to bar pull-out)
  - Disadvantages:
    properties of rotational spring depend on geometry and moment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

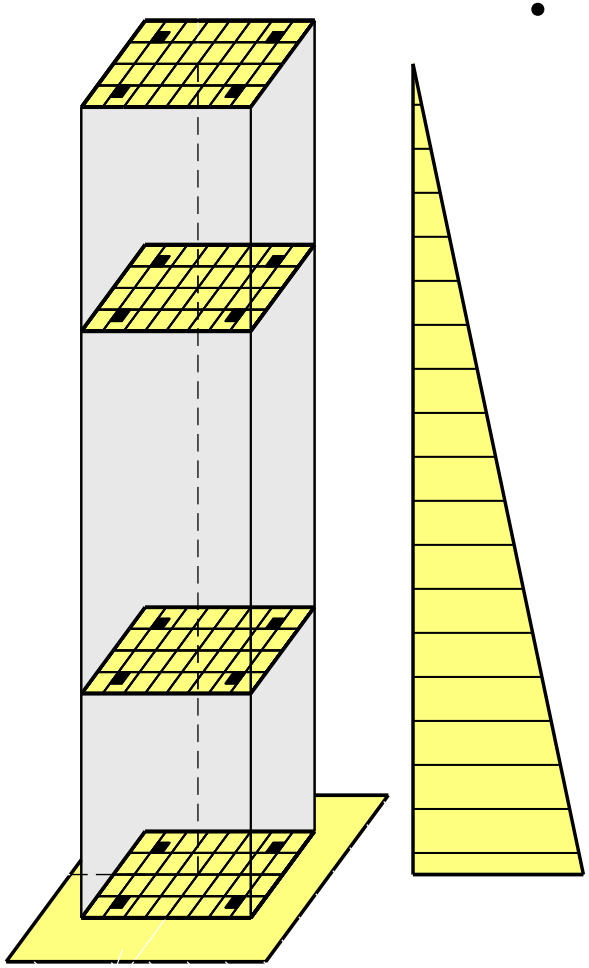
Source: Filip Filippou, UC Berkeley

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

Source: Filip Filippou, UC Berkeley

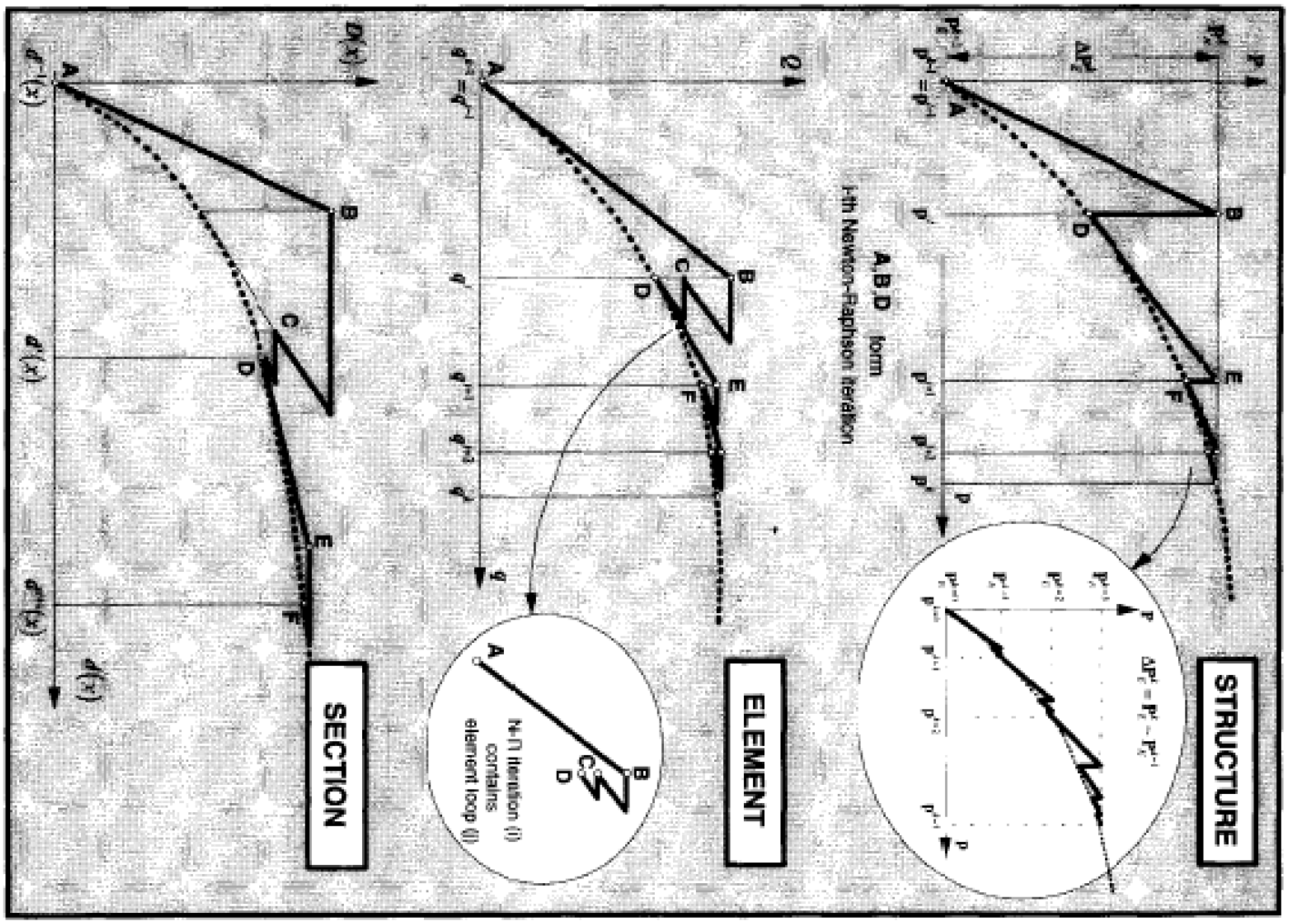
2 Element Types to Choose From:

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 (h 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).

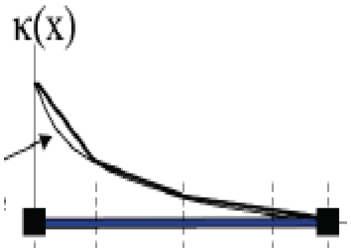
MIXED FORMULATION OF NONLINEAR BEAM FINITE ELEMENT

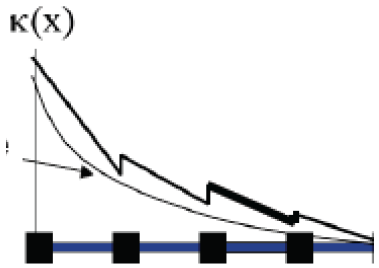
E. Spacone,† V. Ciampi‡ and F. C. Filippou†



Which is more ‘accurate’ ?

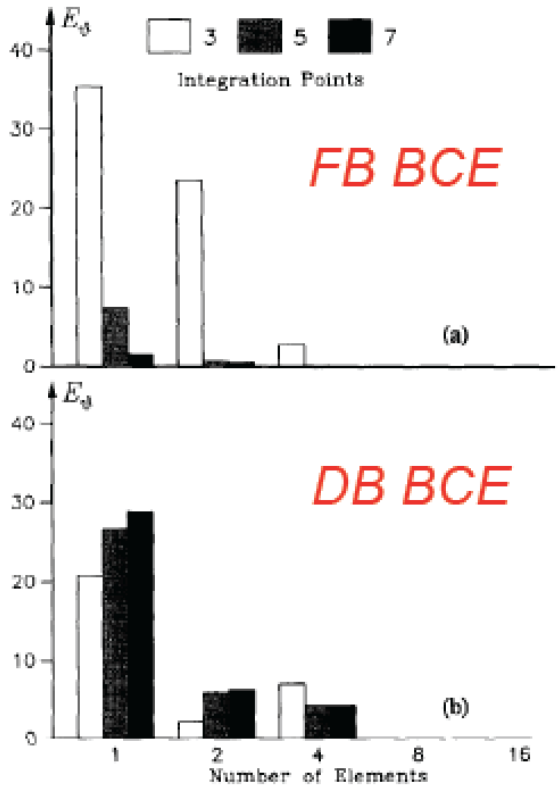
Neuenhofer, A., and F. C. Filippou, (1997). “Evaluation of Nonlinear Frame Finite Element Models.” *Journal of Structural Engineering*, 123(7): 958-966.





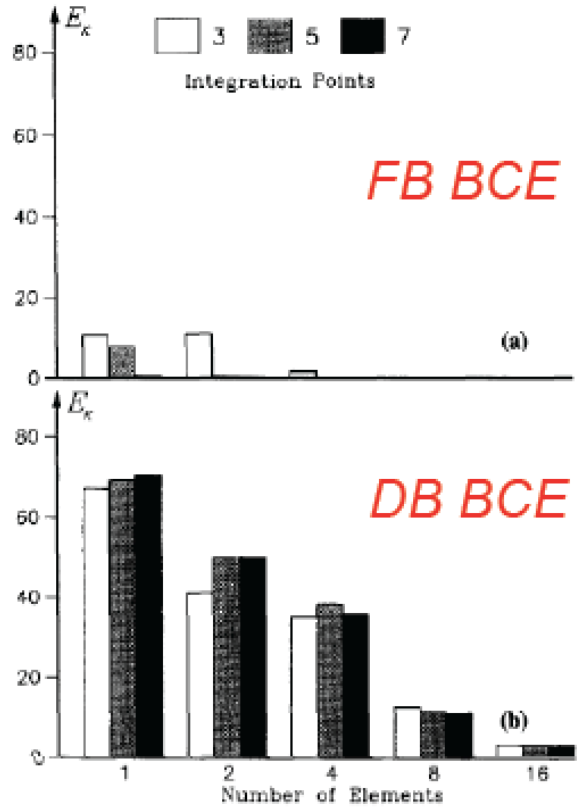
Rotation error (node B)

Global response

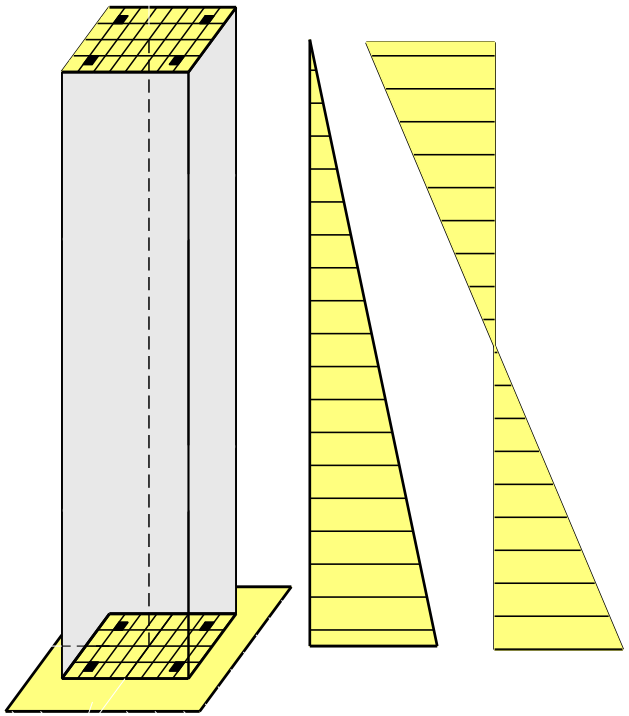


Curvature error (node A)

Local response



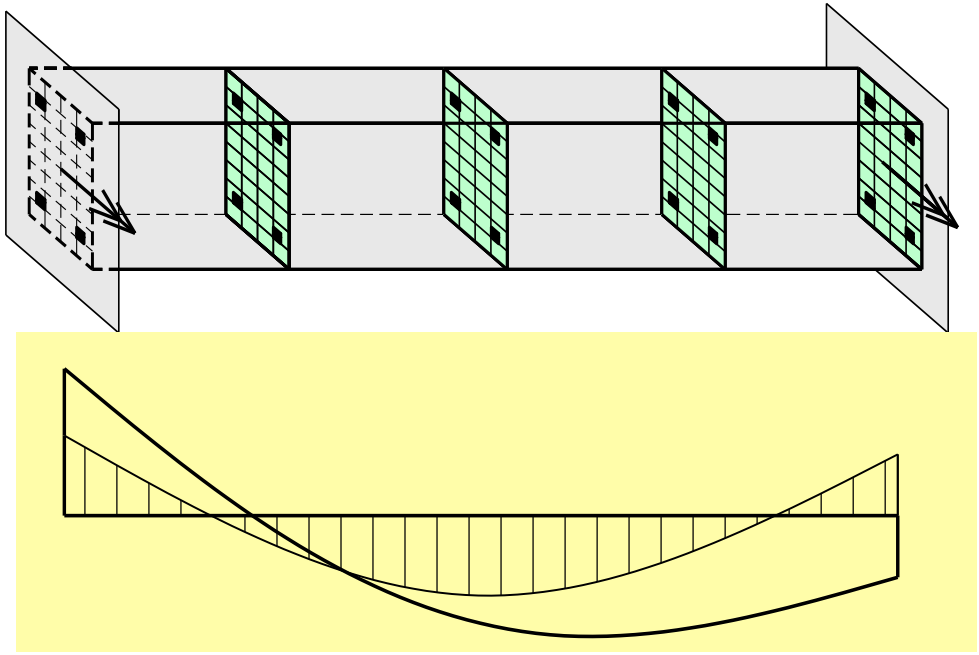
“Economic” Distributed inelasticity models for Columns



- **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)

Source: Filip Filippou, UC Berkeley

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

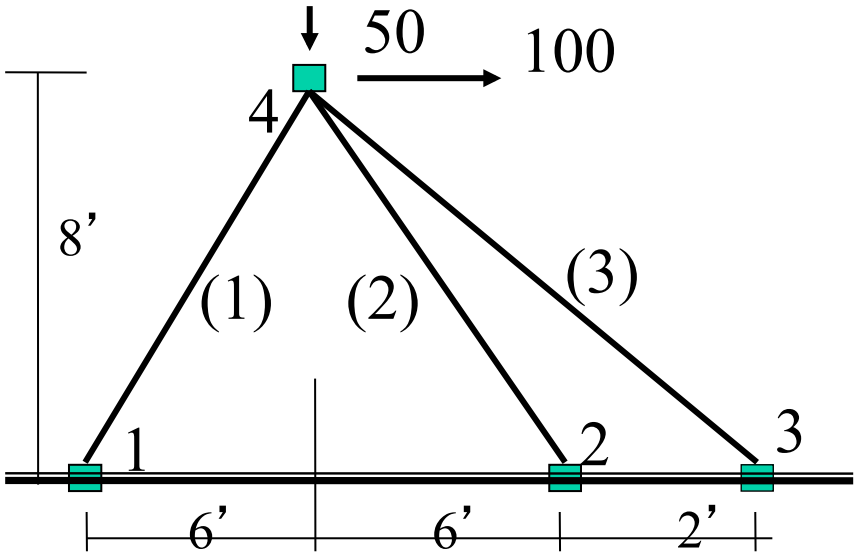
Source: Filip Filippou, UC Berkeley

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!

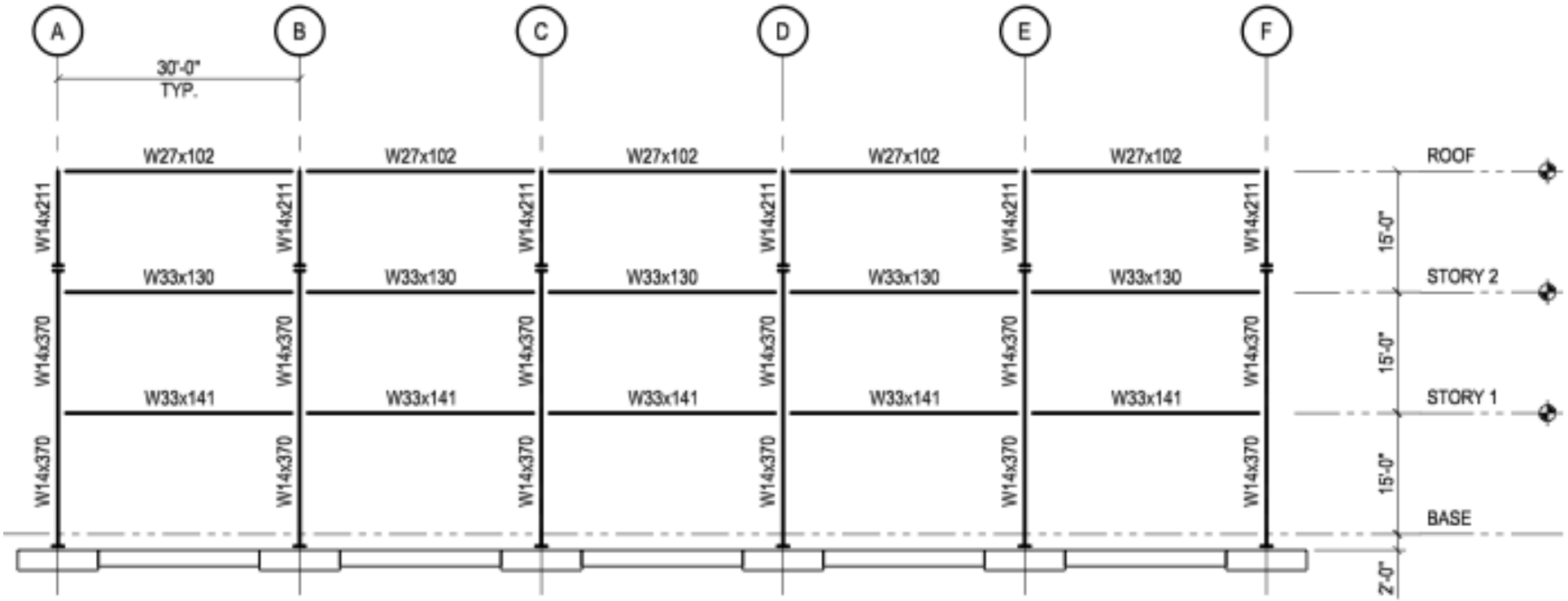
source: wikipedia

```
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'

3 Story 5 Bay Moment Frame Original Code >350 lines

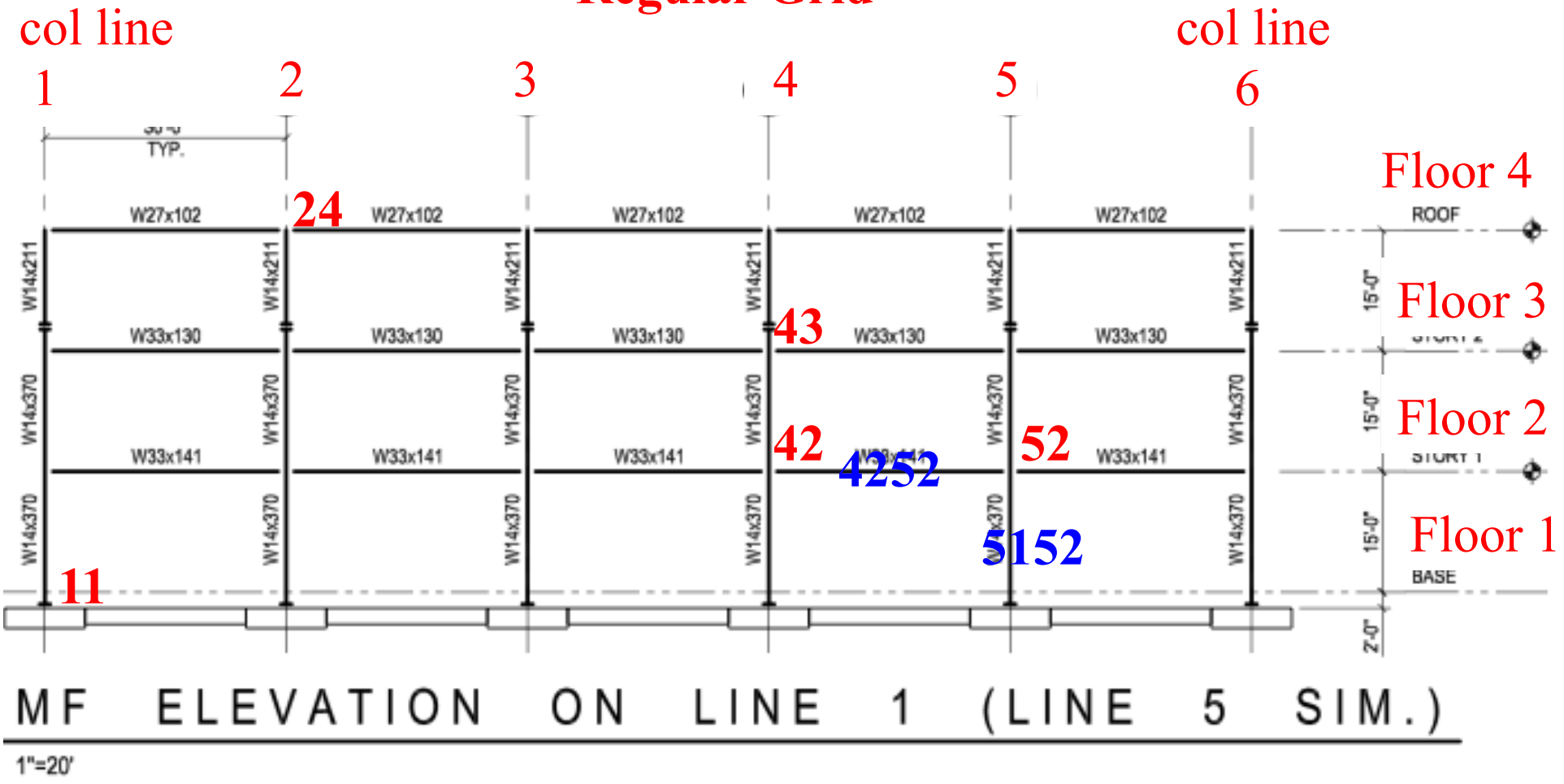
If I had to do this I would want a GUI too!

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 # colfloor

Elements # $iNode$jNode

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

MRF1.tcl

```
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 $cr1 $cr2
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
    }
}
```

Same model in 35 lines!

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
}
```

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

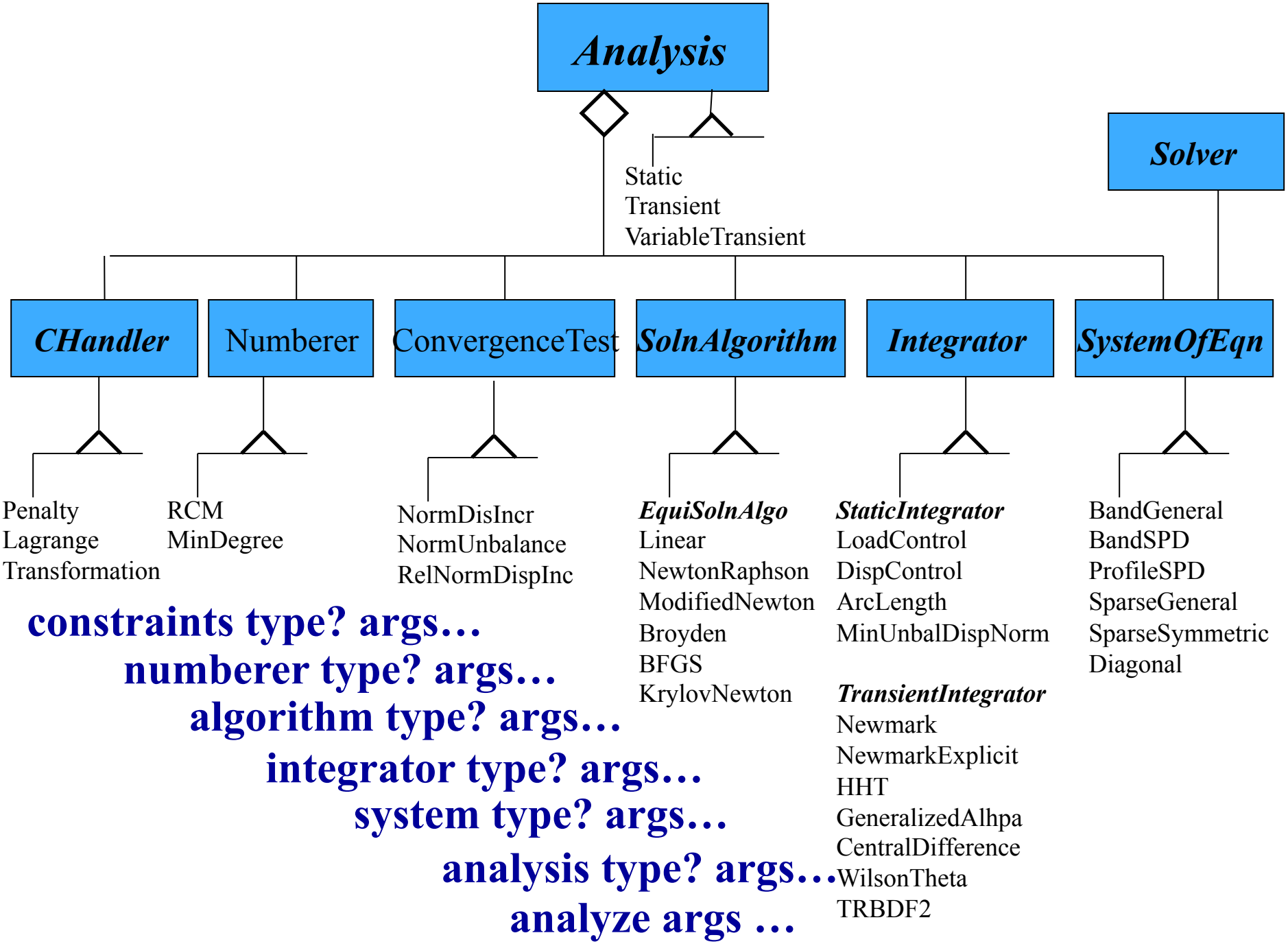
- It requires **much** more thought when setting up the model
- It requires more thought when setting up the analysis
- It takes more computational time.
- It does not always converge.
- It does not always converge to the correct solution.

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
CHECK YOUR MODEL

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


Analysis
Static
Transient
VariableTransient
Solver
CHandler
Numberer
ConvergenceTest
SolnAlgorithm
Integrator
SystemOfEqn
Penalty
Lagrange
Transformation
RCM
MinDegree
NormDisIncr
NormUnbalance
RelNormDispInc
EquiSolnAlgo
Linear
NewtonRaphson
ModifiedNewton
Broyden
BFGS
KrylovNewton
StaticIntegrator
LoadControl
DispControl
ArcLength
MinUnbalDispNorm
TransientIntegrator
Newmark
NewmarkExplicit
HHT
GeneralizedAlhpa
CentralDifference
WilsonTheta
TRBDF2
BandGeneral
BandSPD
ProfileSPD
SparseGeneral
SparseSymmetric
Diagonal
constraints type? args…
numberer type? args…
algorithm type? args…
integrator type? args…
system type? args…
analysis type? args…
analyze args …

test command:

- to specify when convergence has been achieved

all look at system: **KU = R**

- Norm Unbalance

$\sqrt{\mathbf{R\hat{}R}} < \mathbf{tol}$

test NormUnbalance tol? numIter? <flag?>

- Norm Displacement Increment

$\sqrt{\mathbf{U\hat{}U}} < \mathbf{tol}$

test NormDispIncr tol? numIter? <flag?>

- Norm Energy Increment

½ (U^R) < 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

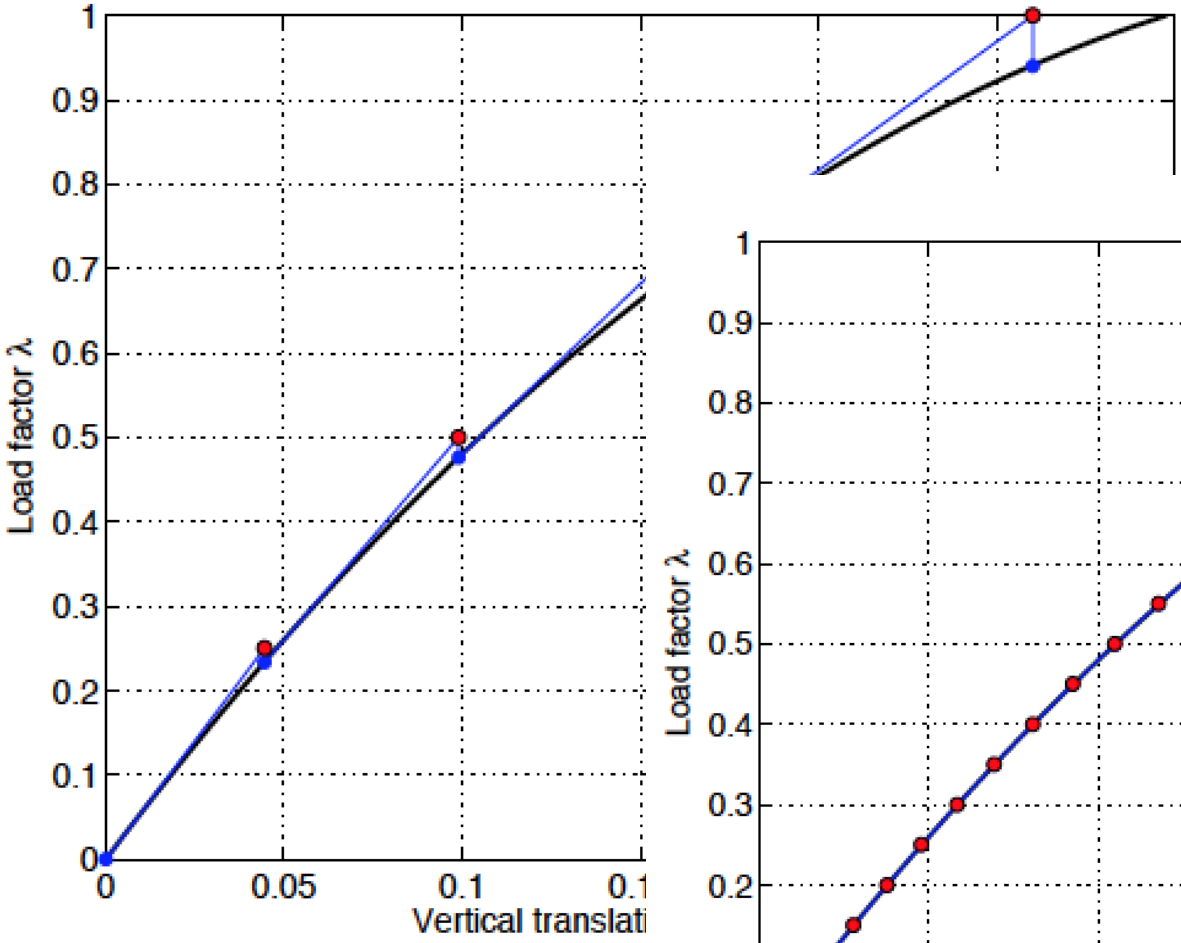
• numbering has an impact on performance of banded and profile solvers. The sparse solvers all use their own optimal numbering schemes.

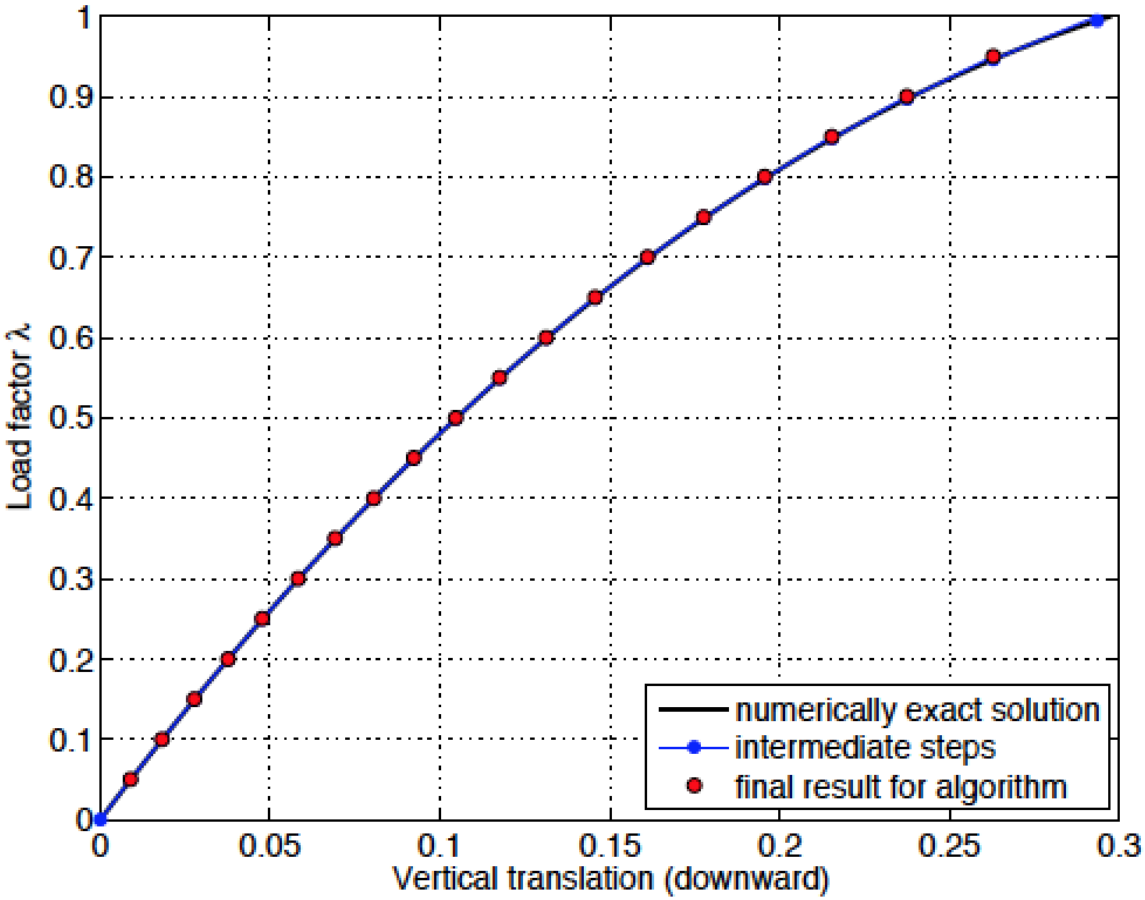
algorithm command:

- 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());
```

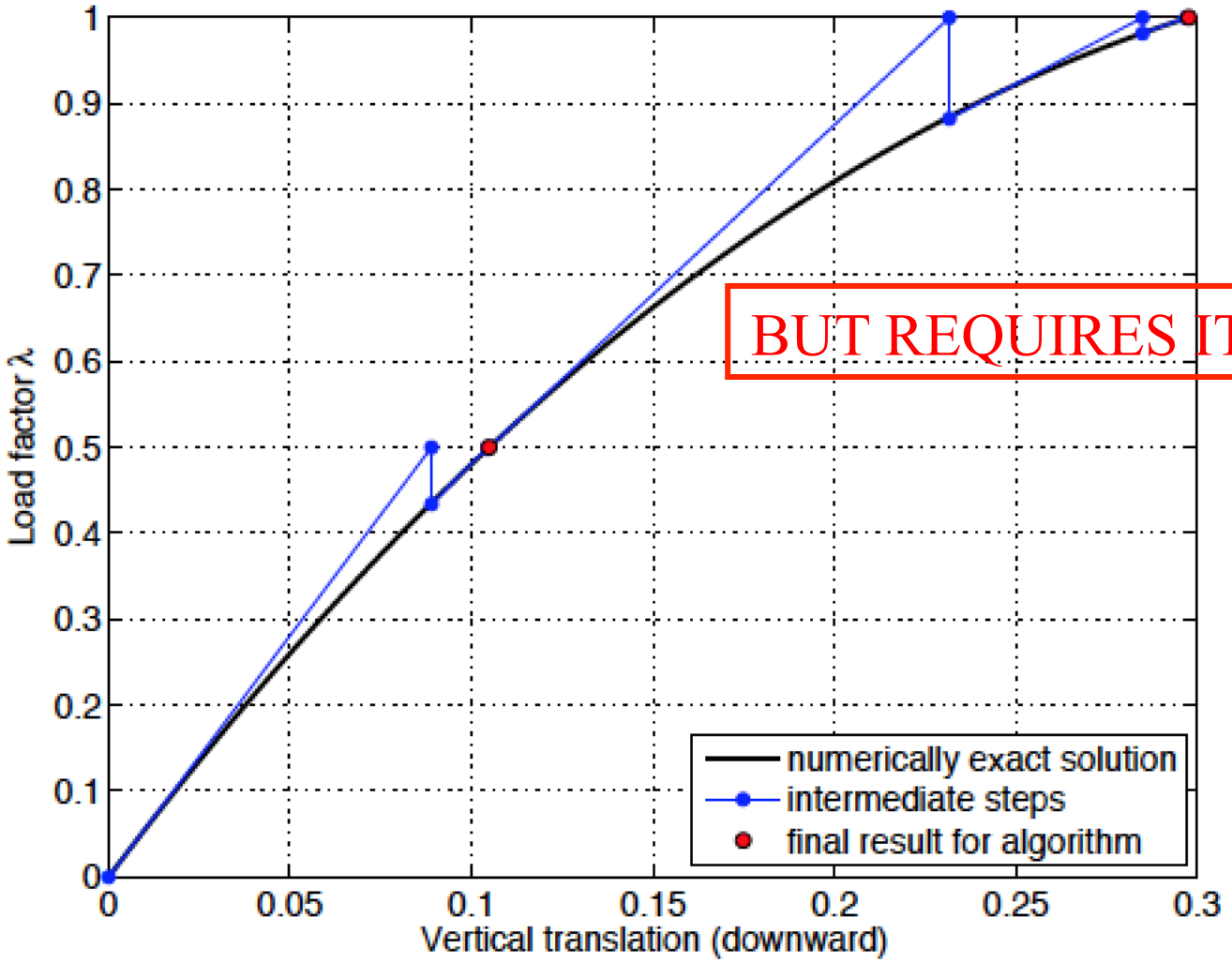




algorithm Newton

(Newton-Raphson)

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

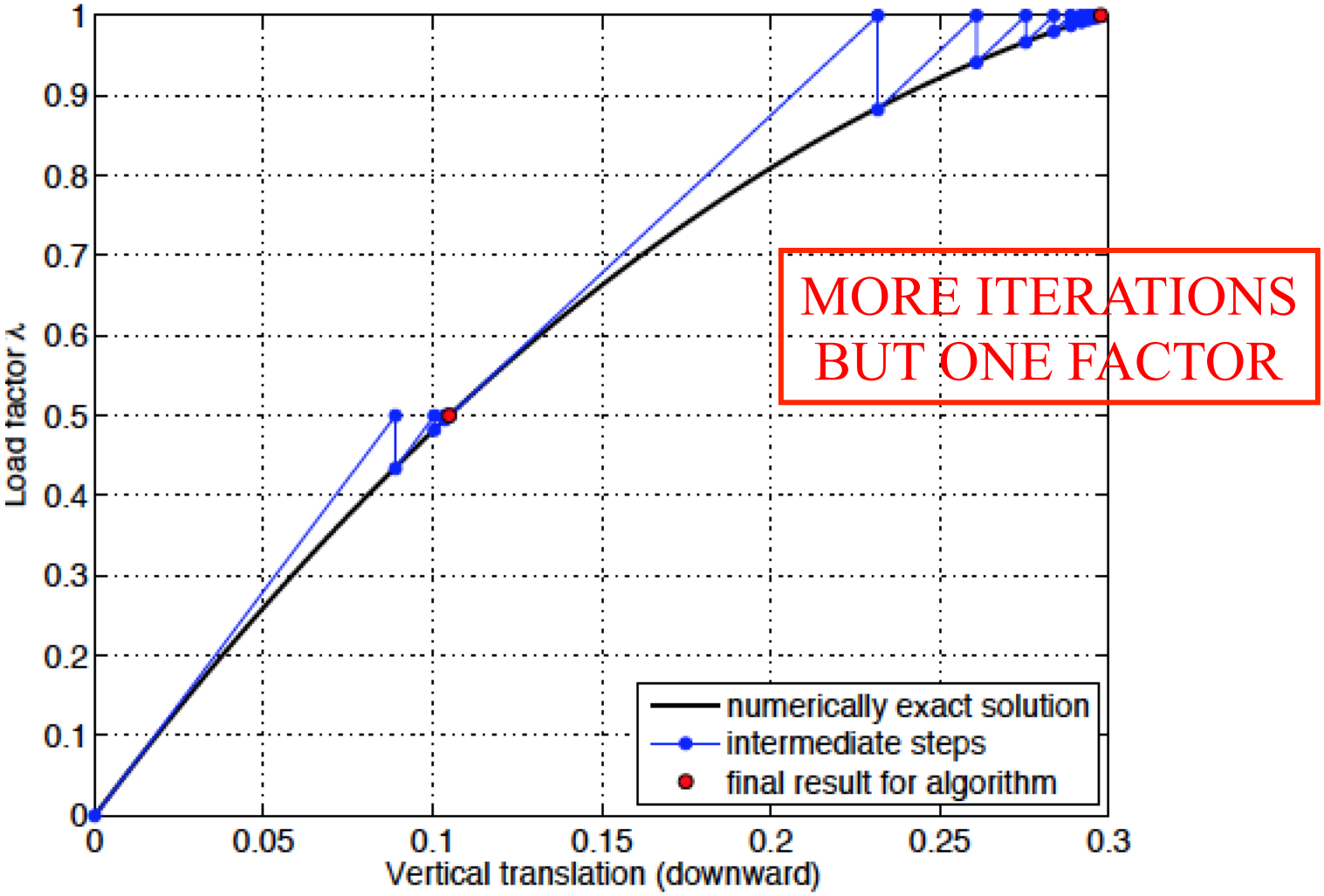


BUT REQUIRES ITERATION

algorithm ModifiedNewton

(Modified Newton-Raphson)

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



MORE ITERATIONS
BUT ONE FACTOR

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

$$\mathbf{U}c = \mathbf{C}rc\ \mathbf{U}r$$

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

$$\mathbf{T}\ \mathbf{U}r = [\mathbf{U}r\ \ \mathbf{U}c]\hat{}$$

$$[\mathbf{C}r\ \ \mathbf{C}c]\hat{}[\mathbf{U}r\ \ \mathbf{U}c] = 0$$

- Transformation Handler

$$K^* Ur = R^* \qquad K^*=T\hat{}KT \qquad R^*=T\hat{}R$$

constraints Transformation

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

- Lagrange Handler

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

constraints Lagrange

- Penalty Handler

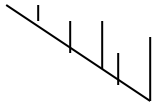
$$[\mathbf{K} + \mathbf{C}\hat{}\boldsymbol{\alpha}\mathbf{C}]\ \mathbf{U} = [\mathbf{R} + \mathbf{C}\hat{}\boldsymbol{\alpha}\mathbf{Q}]$$

constraints Penalty α_{sp}? α_{mp}?

system command:

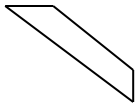
- to specify how matrix equation KU = R is stored and solved

•Profile Symmetric Positive Definite



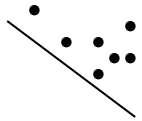
system ProfileSPD

•Banded Symmetric Positive Definite



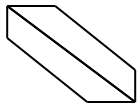
system BandSPD

•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:

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

2. Transient Integrators for Use in Transient Analysis

Nonlinear equation of the form:

$$\mathbf{R(U, \dot{U}, \ddot{U}) = P(t) - F_I(\ddot{U}) - F_R(U, \dot{U})}$$

Static Integrators

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

at each step solving for λ

- Load Control

$$\lambda_n = \lambda_{n-1} + \Delta\lambda$$

integrator LoadControl Δλ

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

- Displacement Control

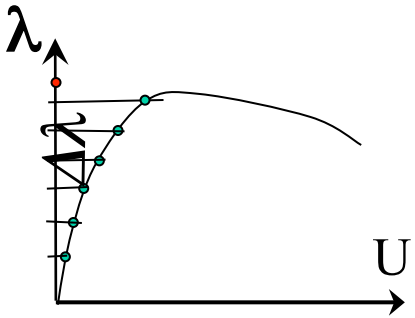
$$Uj_n = Uj_{n-1} + \Delta Uj$$

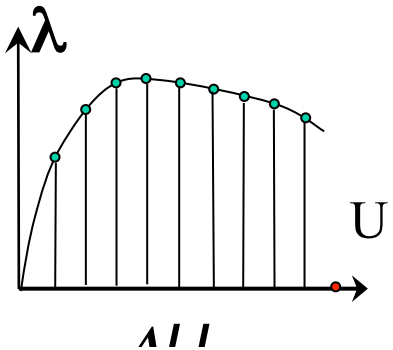
integrator DisplacementControl node dof ΔU

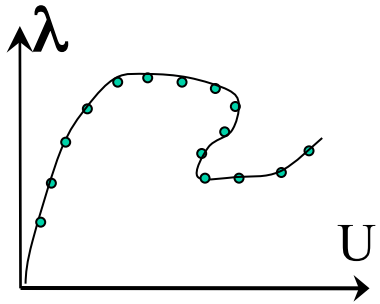
- Arc Length

$$\Delta U_n \hat{} \Delta U_n + \alpha^2 \Delta\lambda_n = \Delta s^2$$

integrator ArcLength α Δs







Transient Integrators

- Explicit:

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

  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

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

  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

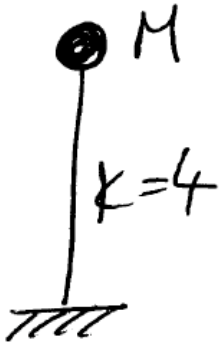
Linear Acceleration $(\gamma = \frac{1}{2}, \beta = \frac{1}{6})$

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

Example

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

Free vibration (ex1.tcl) (ex2.tcl)



$u(0) = 1$

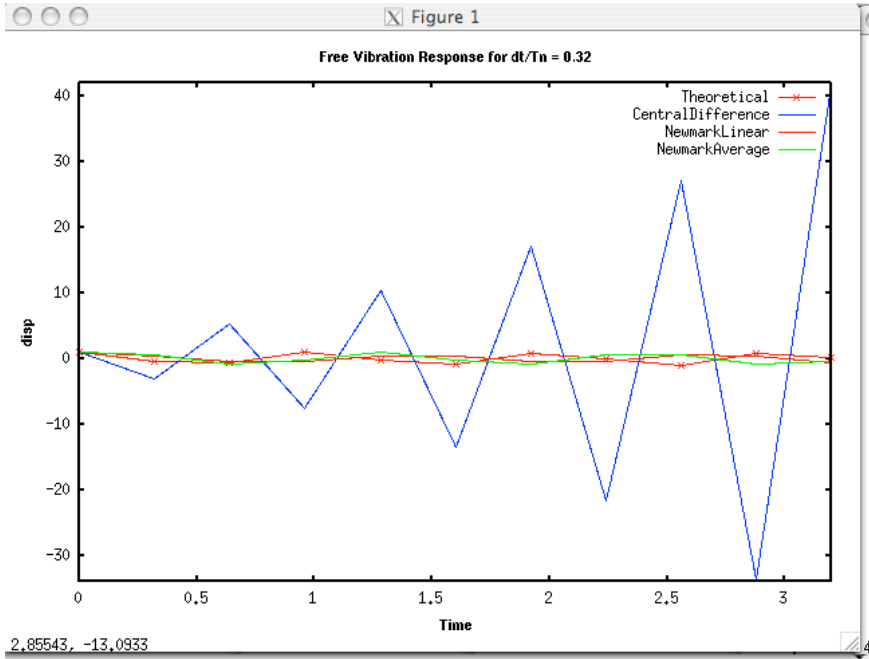
$u'(0) = 0$

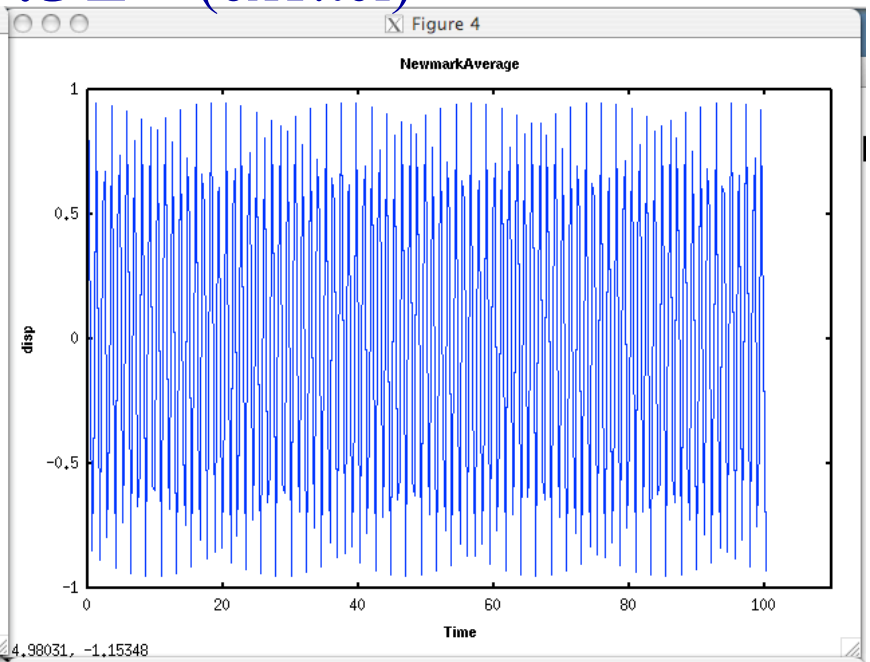
$p(t) = 0$

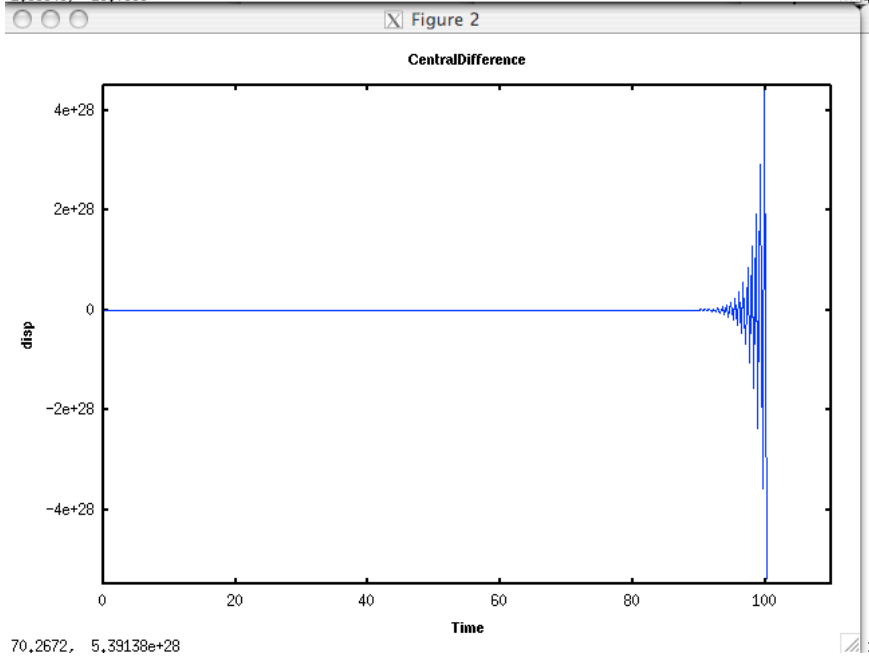
M is chosen to give desired period.

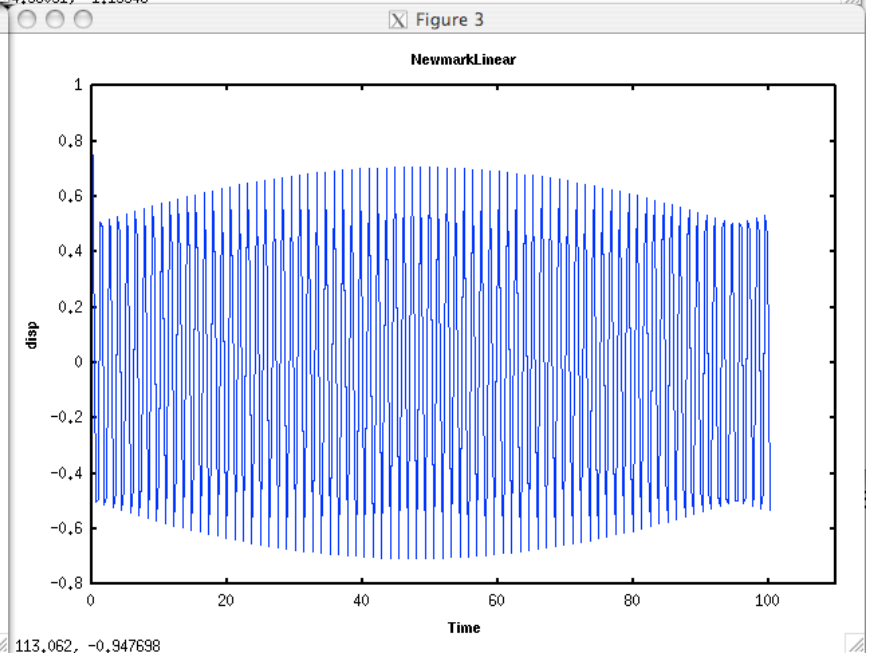
exact soln: $u(t) = u(0)\cos\omega_n t$.

dT/Tn = 0.32 (ex1.tcl)

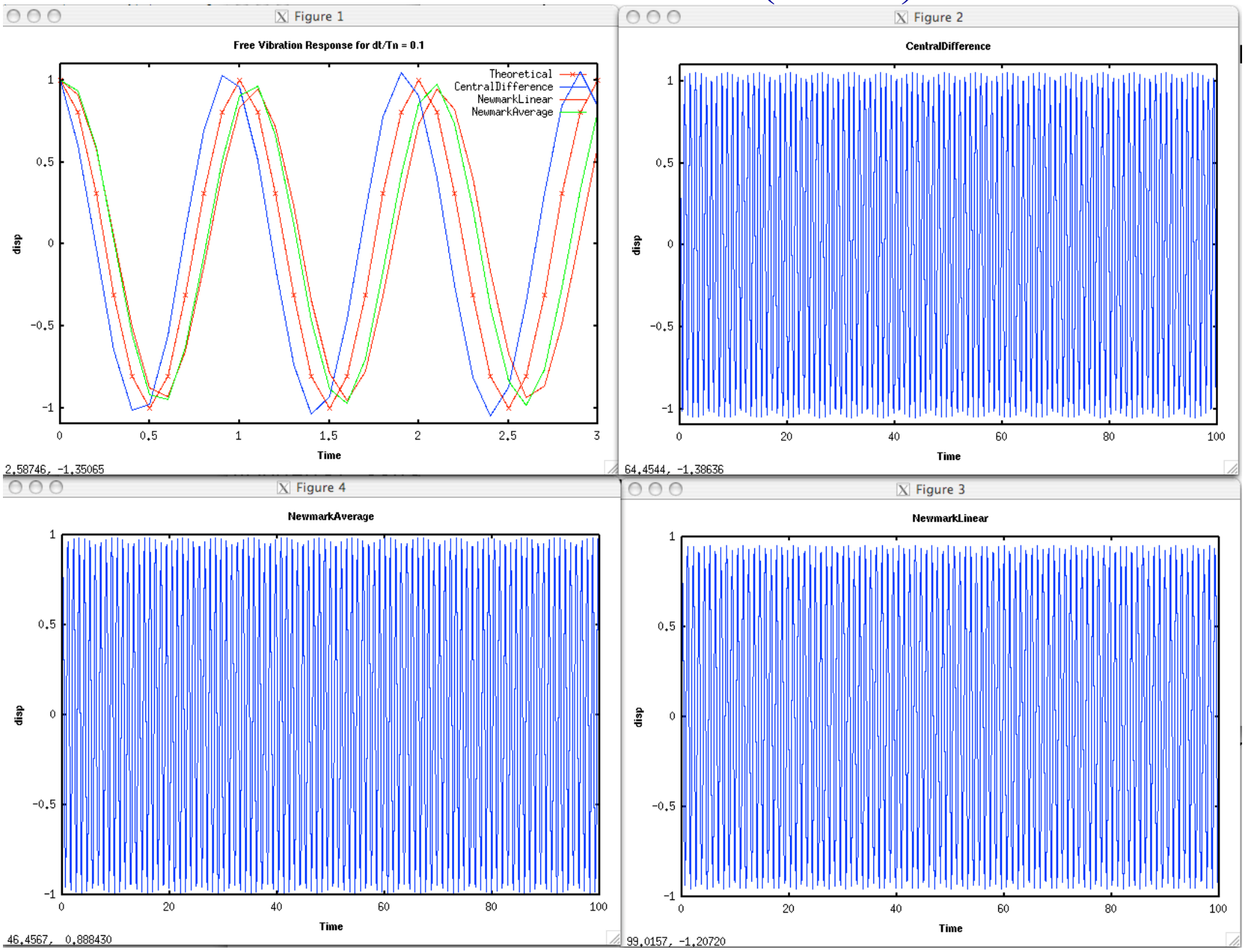




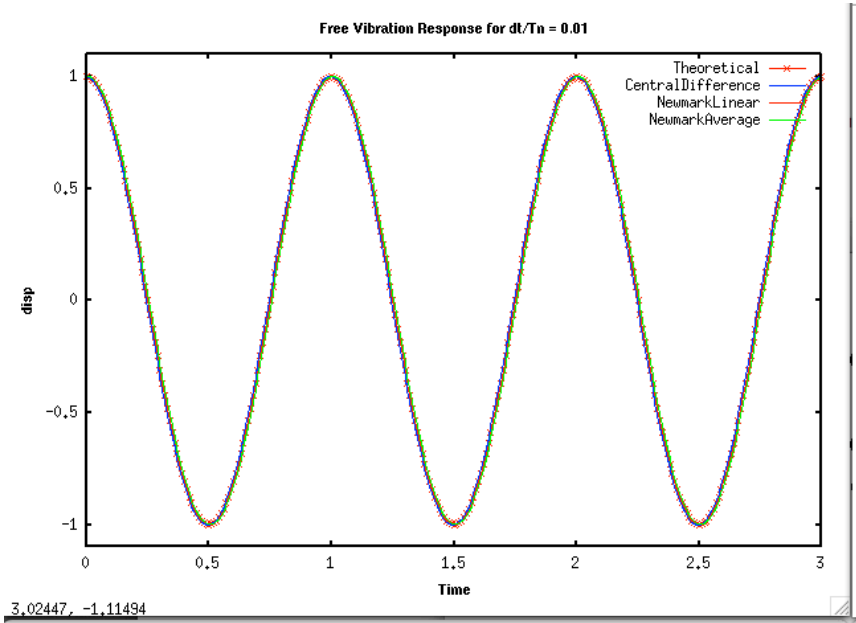


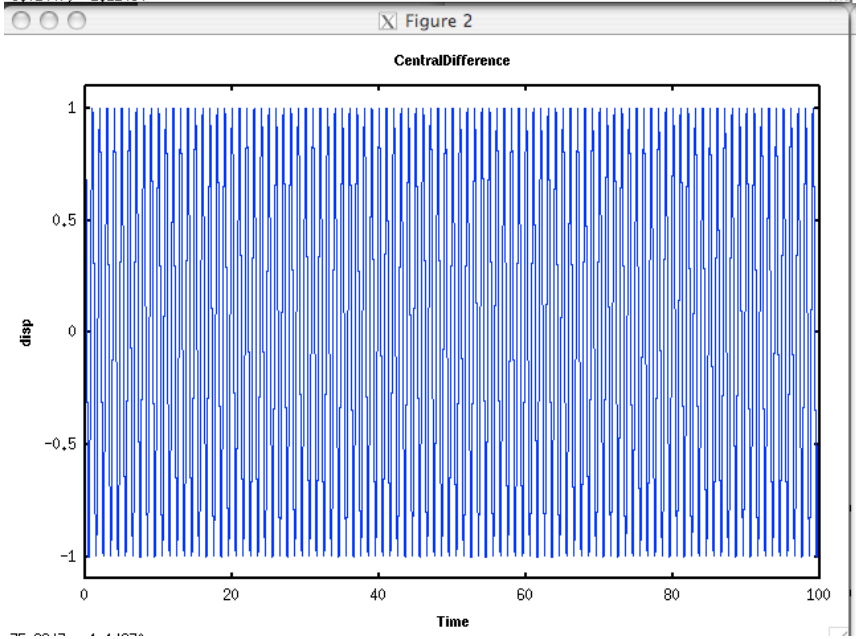


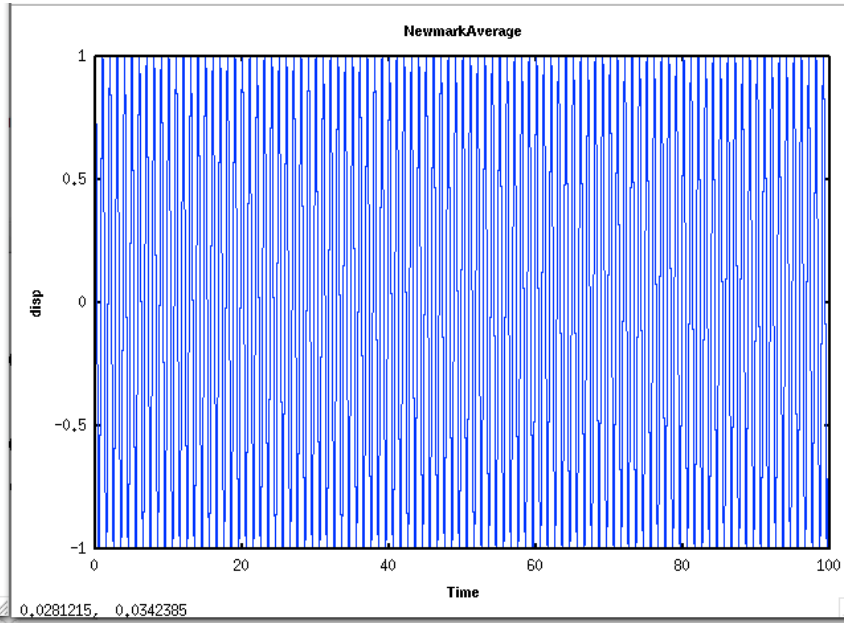
dT/Tn = 0.1 (ex1.tcl)

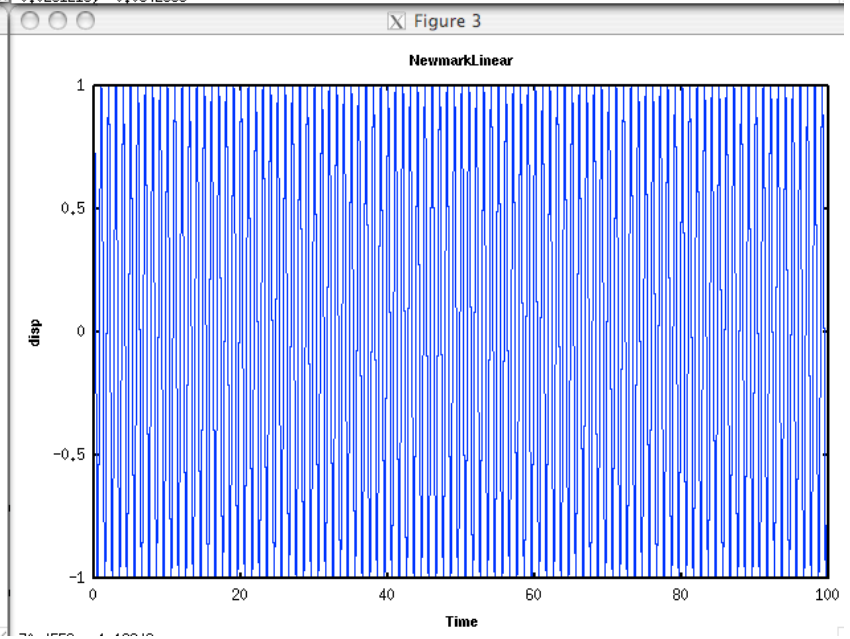


dT/Tn = 0.01 (ex1.tcl)









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:

$$U_{n+1} - U_n + K_{n+1} - K_n <= W_{ext}$$

where U = strain energy and K = kinetic energy

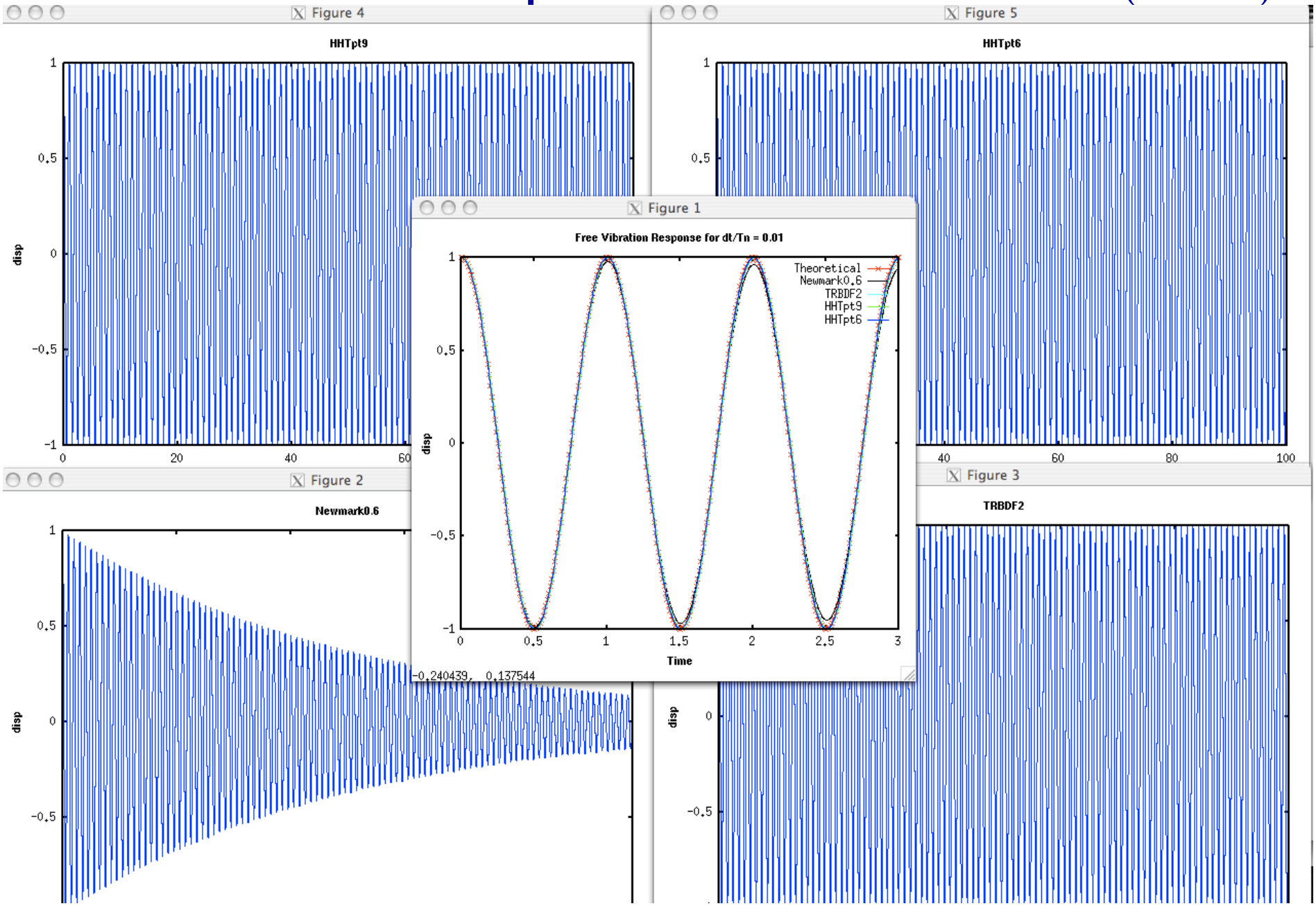
- There are 3 groups of algorithms which ATTEMPT to satisfy this criterion:
  1. Numerical Dissipation
  2. Enforced Conservation of Energy
  3. Algorithmic Conservation of Energy

} **Neither Types Are Available in OpenSees**

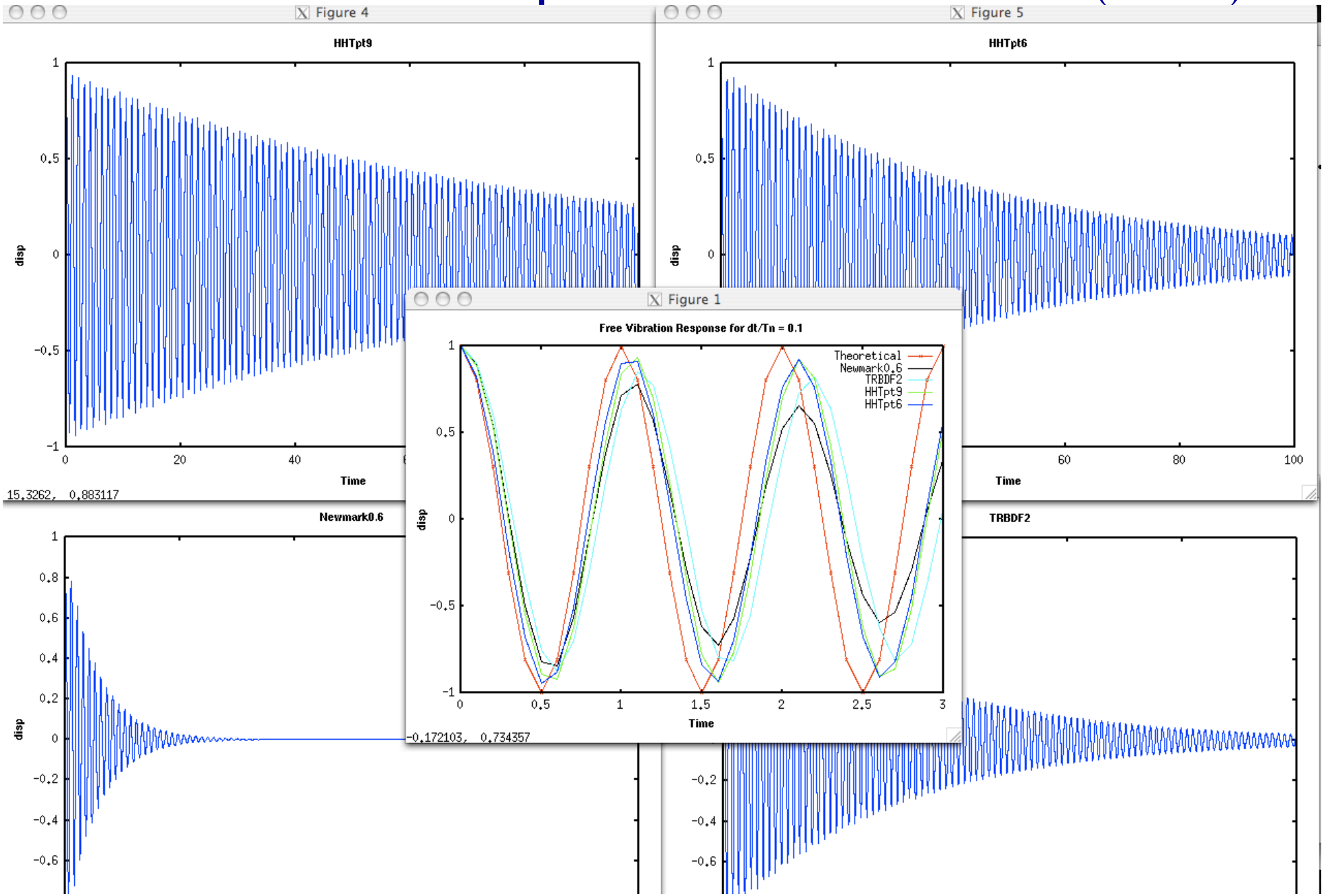
Dissipation Algorithms

- 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

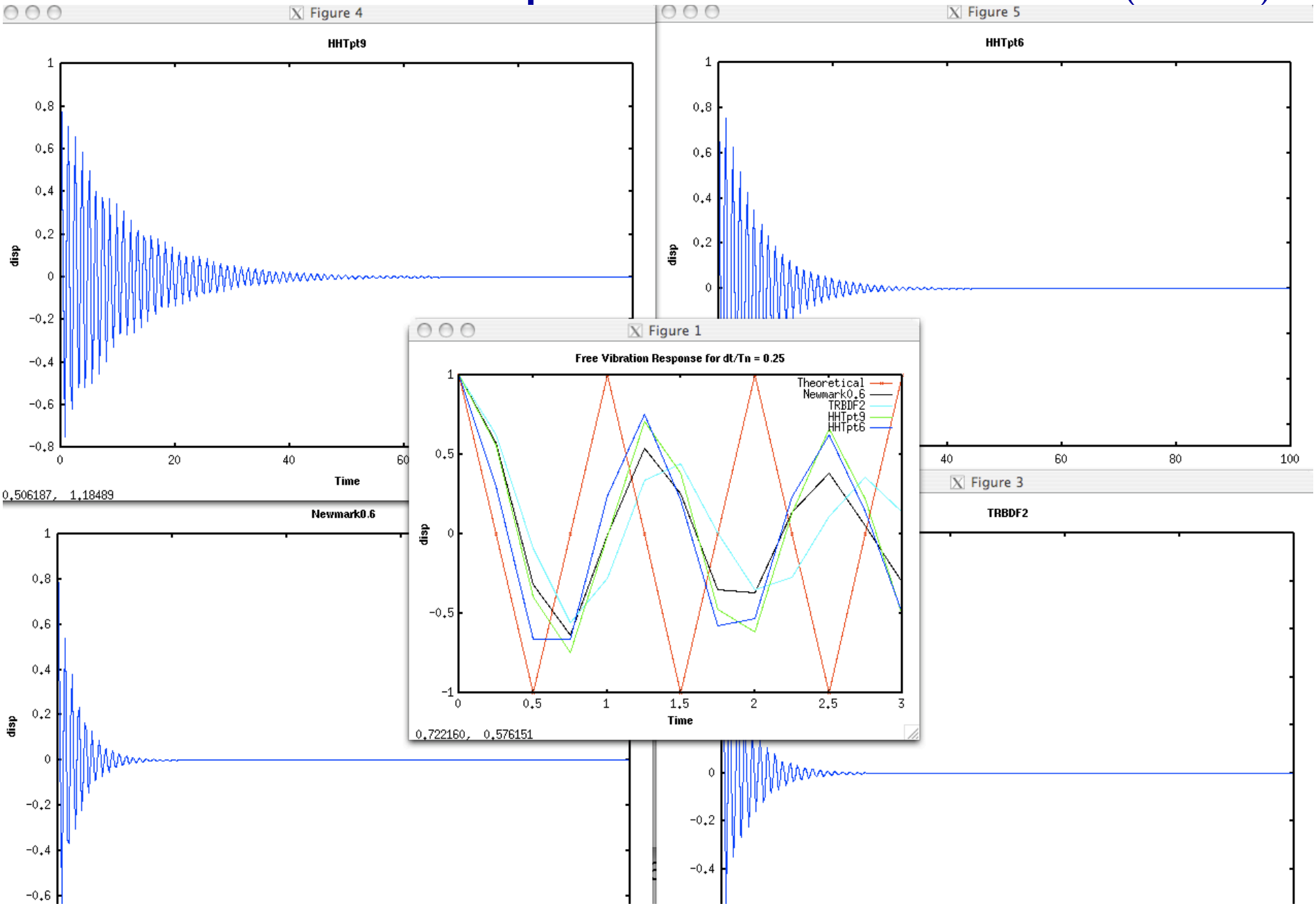
Numerical Dissipation: dT/Tn = 0.01 (ex2.tcl)



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



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



Example

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

Harmonic vibration (ex3.tcl)

$u(0) = 0$

$u'(0) = 0$

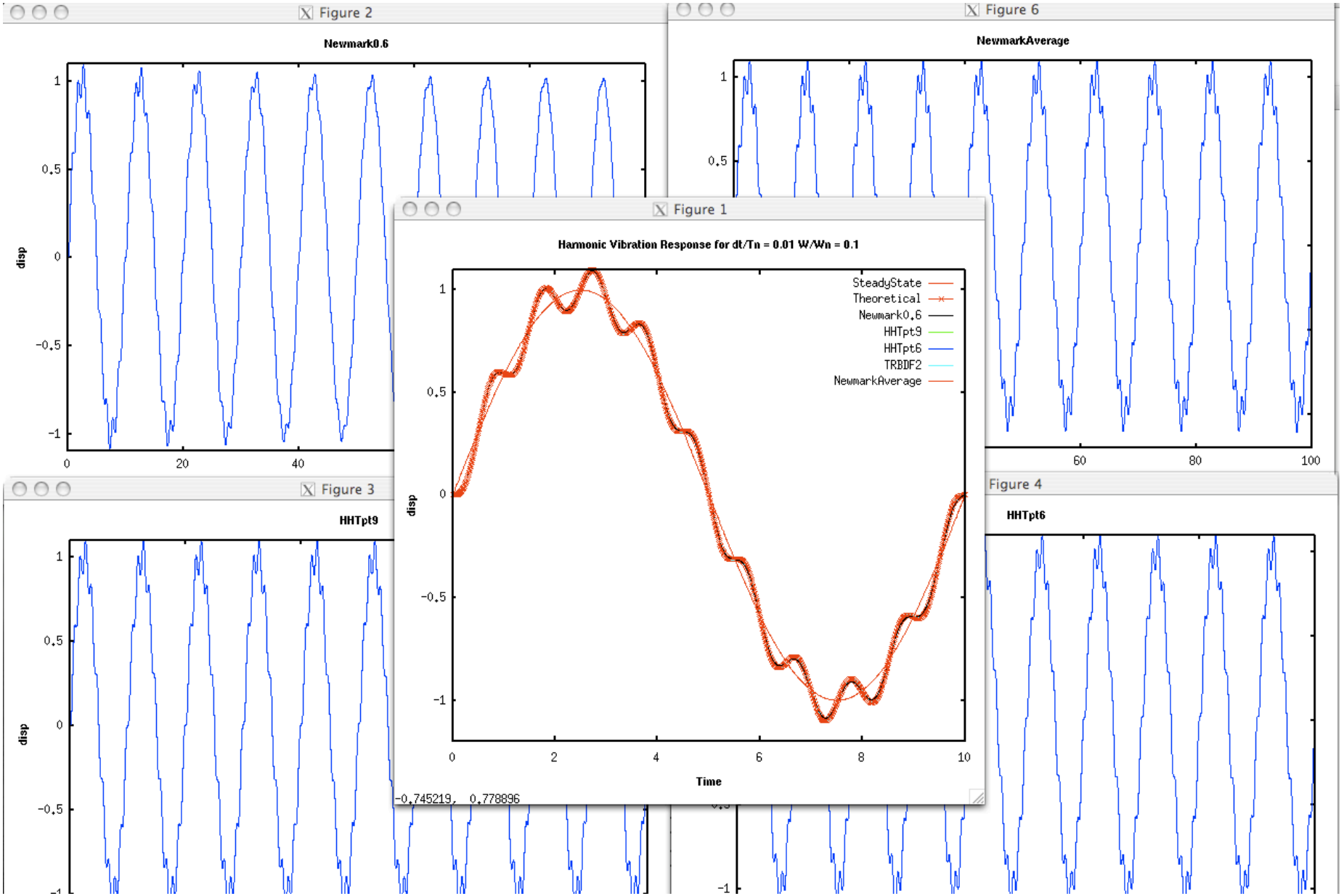
$p(t) = \sin \omega t$

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

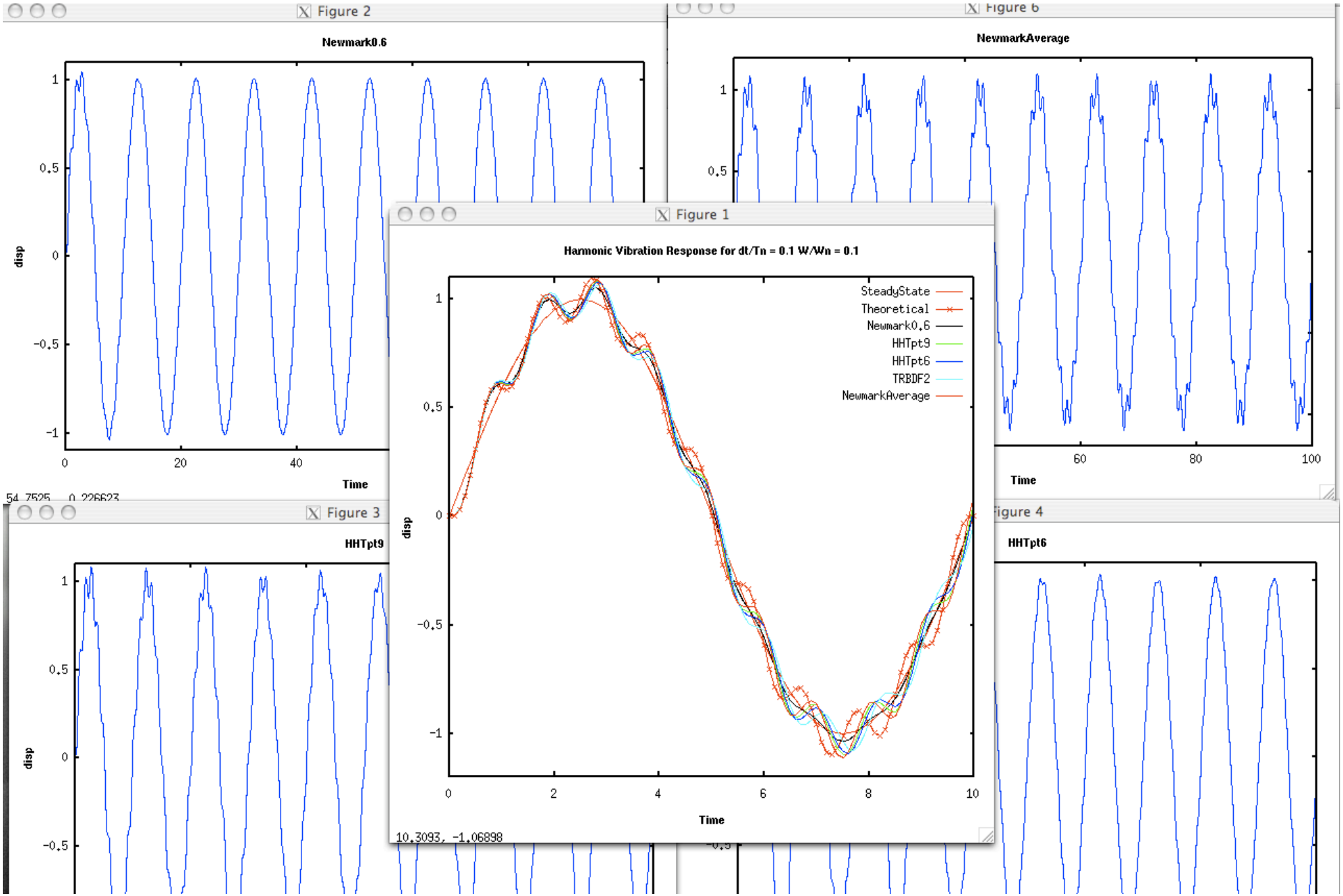
exact soln:

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

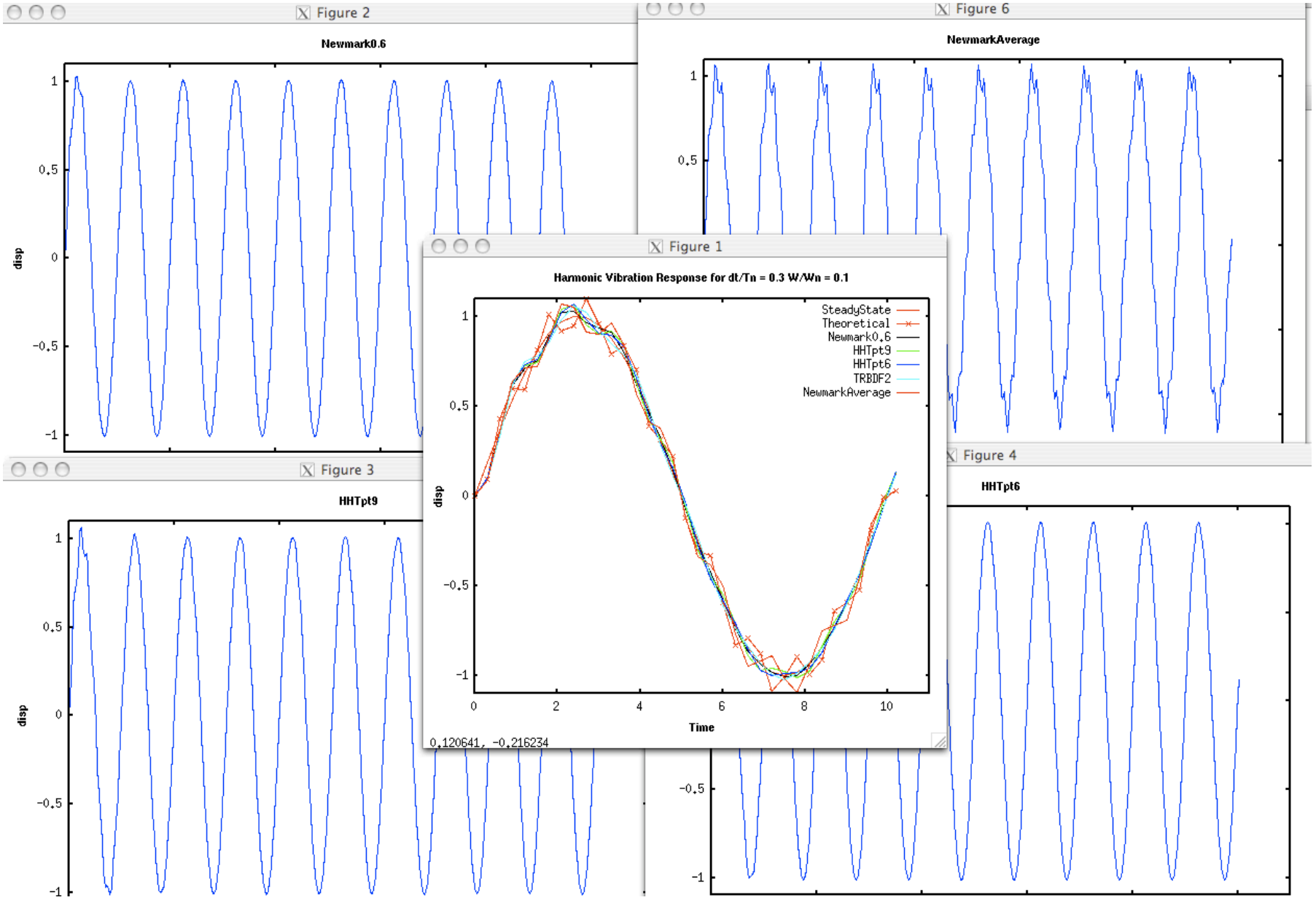
T/Tn = 0.1; dT/Tn = 0.01 (ex3.tcl)



T/Tn = 0.1; dT/Tn = 0.1 (ex3.tcl)



T/Tn = 0.1; dT/Tn = 0.3 (ex3.tcl)

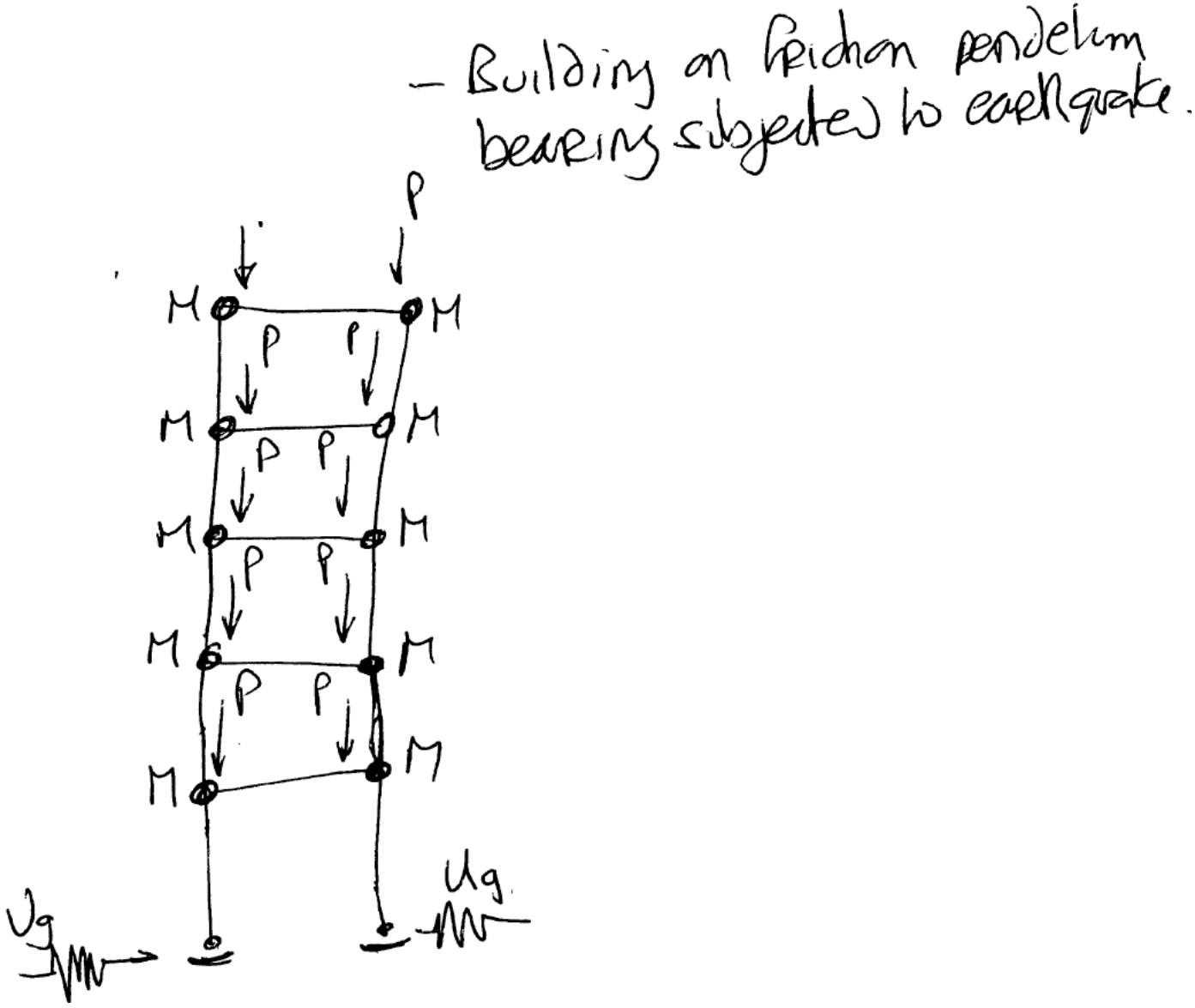


Remember

- 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 Chekenne)



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

Just before point of failure

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

At end of successful analysis

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
		865	0.010499457351349708
		4962	0.008700707004971624
		5294	0.006672111636214162
		19744	0.006170795735890136
		92478	0.006170010220361572
		98169	0.0061668889774234605
		89512	0.0061622426545781695
		84738	0.006158571014872712
17	1.048886e+06	1023.1715596700259	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

For a dT=0.001

Newmark Average Acceleration and HHT 0.9 failed

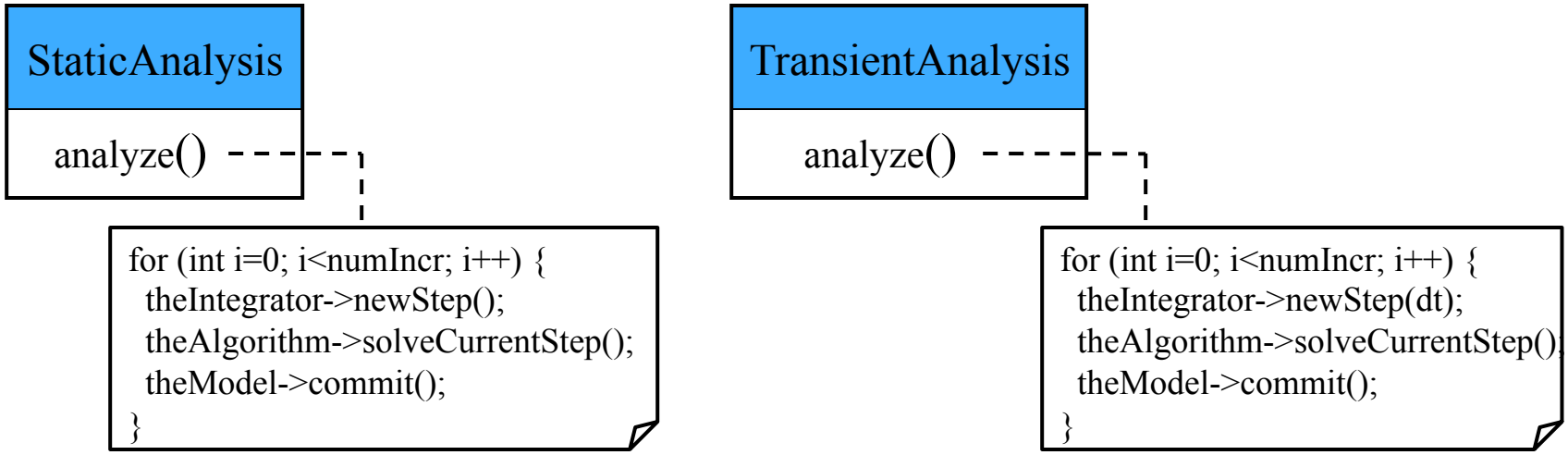
HHT 0.6, TRBDF2, and Newmark 0.6 0.3025 worked

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



- Eigenvalue
  - general eigenvalue problem

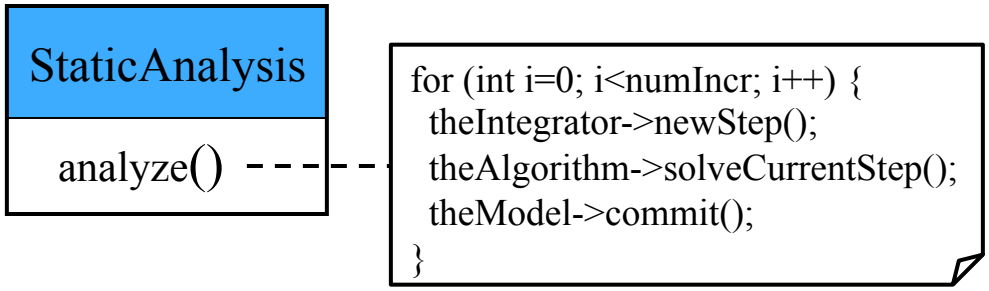
    $(K-\lambda M)\Phi=0$ — *eigen numModes? -general*
  - standard eigenvalue problem

    $(K-\lambda)\Phi=0$ — *eigen numModes? -standard*

analyze command:

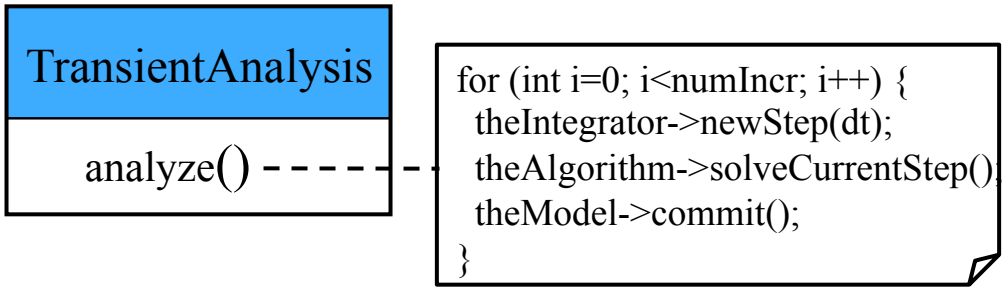
- to perform the static/transient analysis

- Static Analysis



analyze numIter?

- Transient Analysis

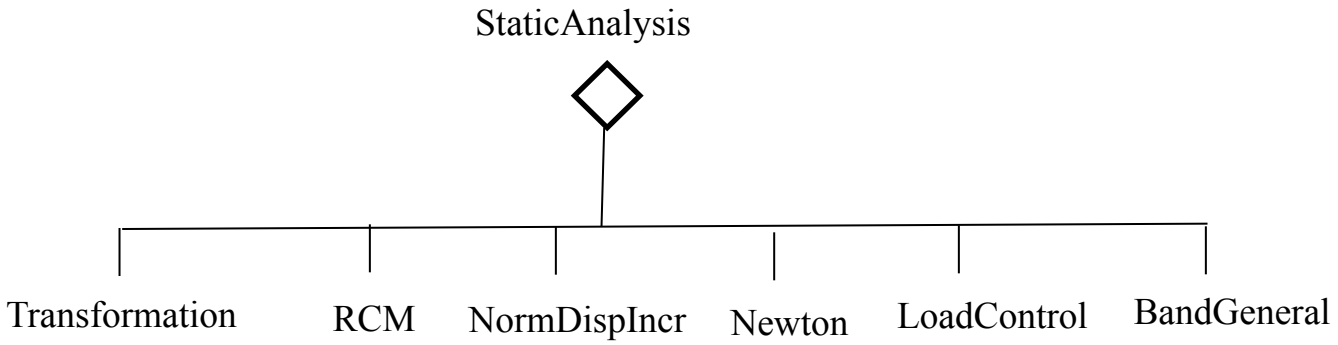


analyze numIter? Δt?

Example Static Analysis:

- Static Nonlinear Analysis with LoadControl

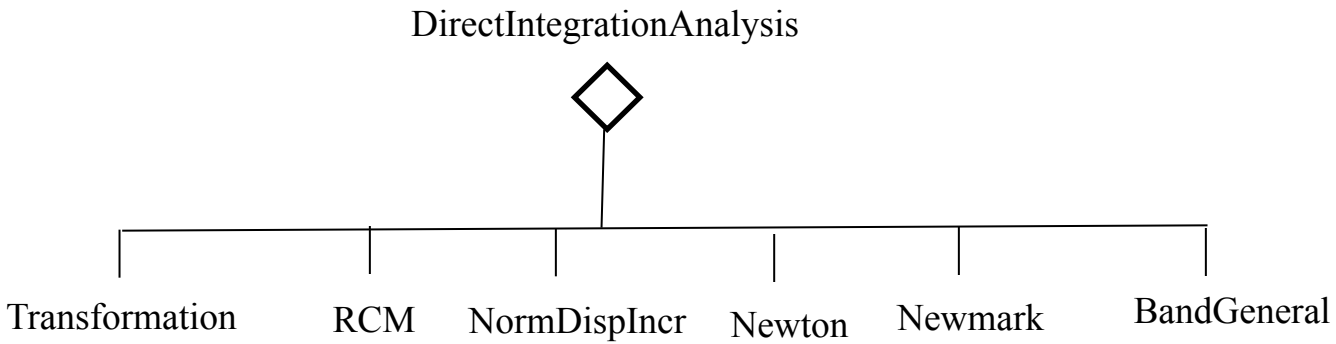
```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator LoadControl 0.1
analysis Static
analyze 10
```



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 [anal;yze 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 respone 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

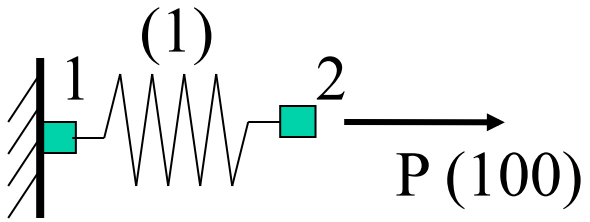
NOTE: Zip up your files in 1 directory and send them to us

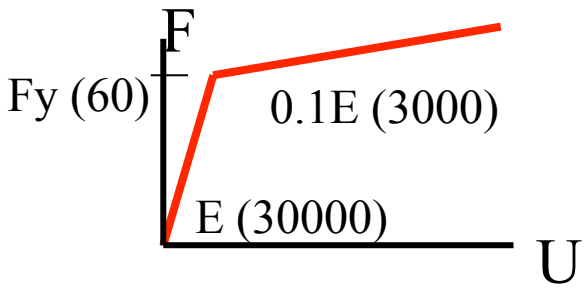
Outline of Workshop

- 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

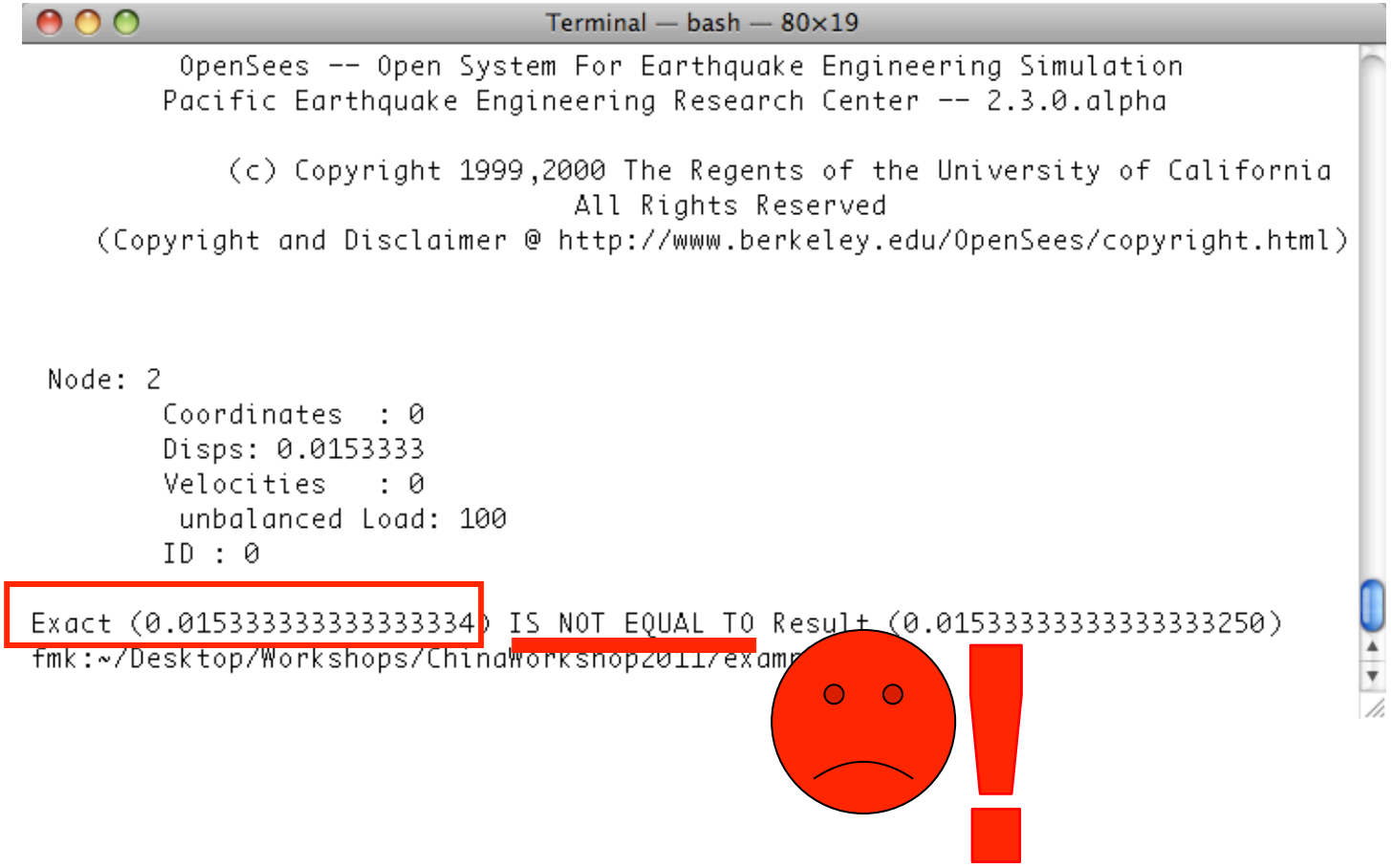
a.tcl






```
# 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)"
}
```

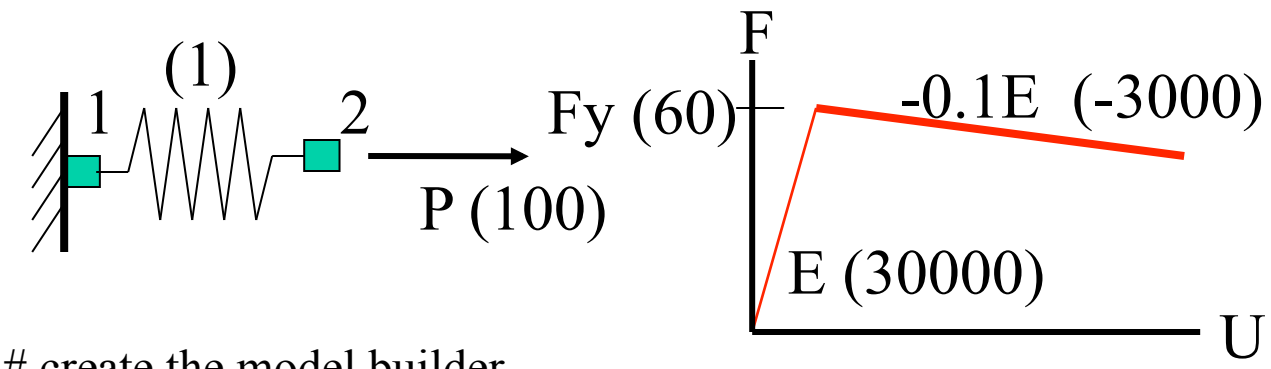


Computers cannot represent all numbers exactly

and

Computer math involves roundoff

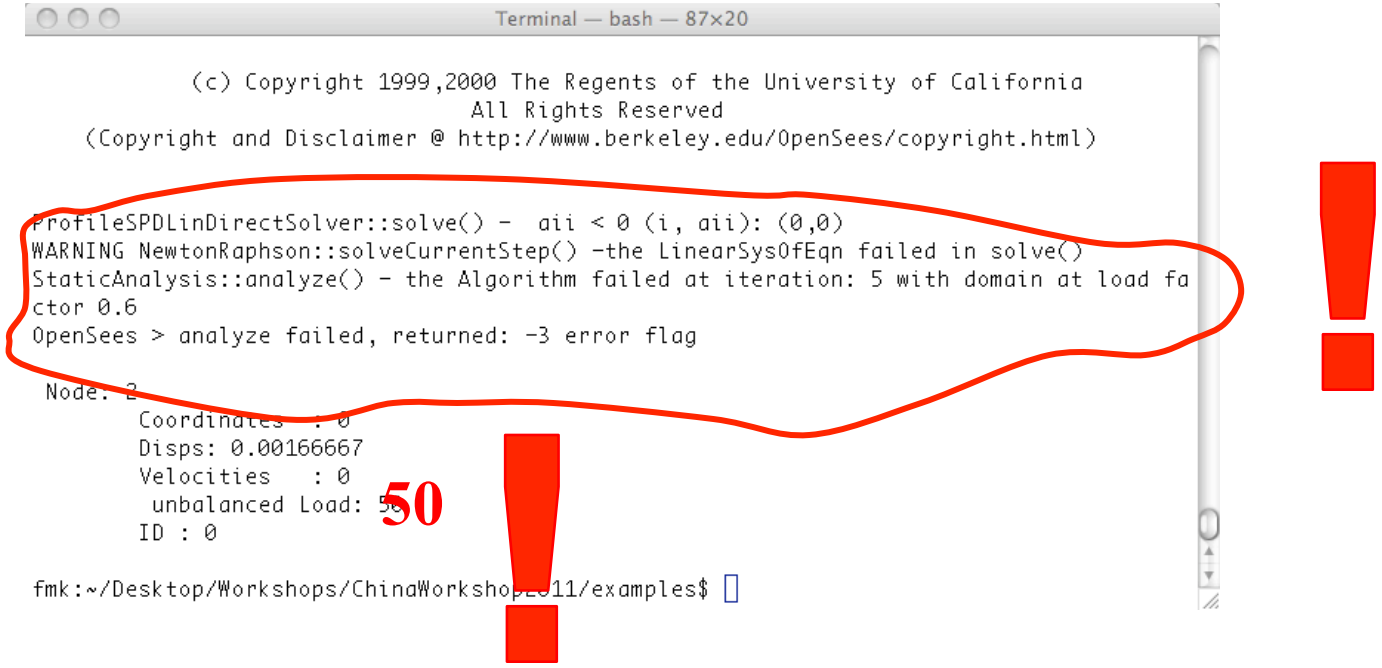
b.tcl




```
# 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
```

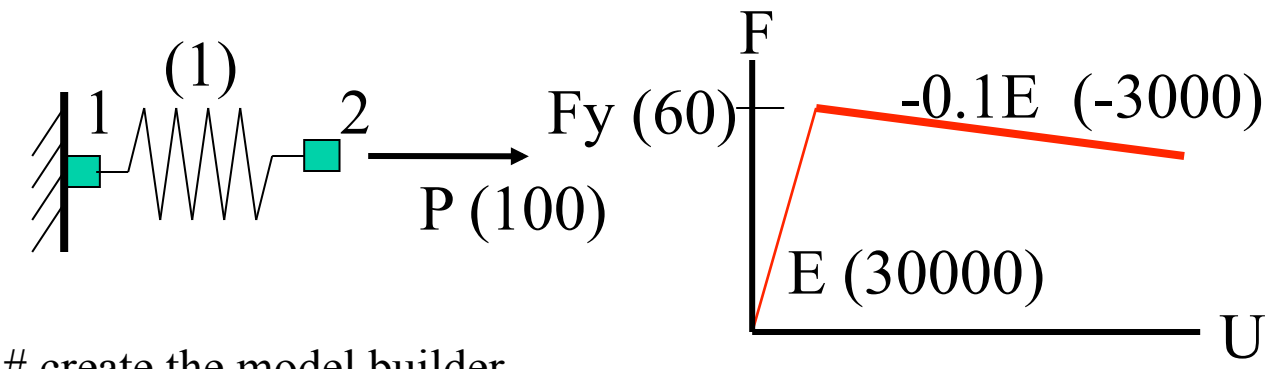
Another common situation for which the solve fails message occurs:
NOT ENOUGH BOUNDARY CONDITIONS!



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

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

c.tcl

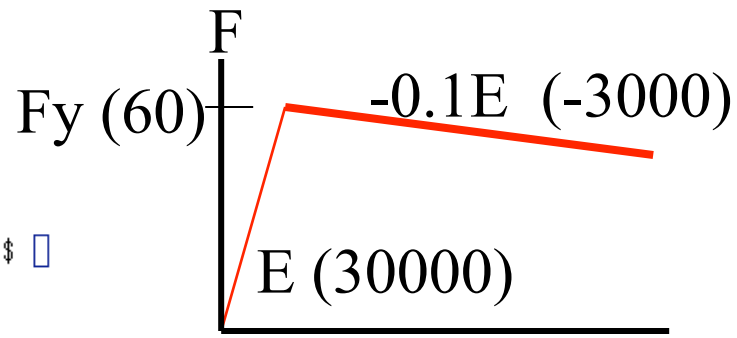



```
# create the model builder
model Basic -ndm 1 -ndf 1

# 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 BandGen
integrator LoadControl 0.09999
analysis Static

# perform the analysis
analyze 10

# Output
print node 2
```

Terminal — bash — 104×22

```
         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$
```

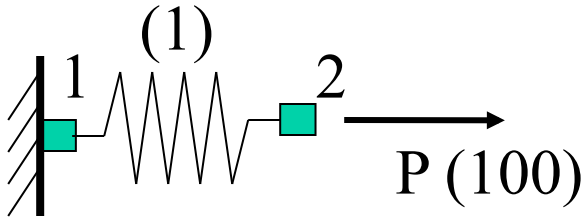


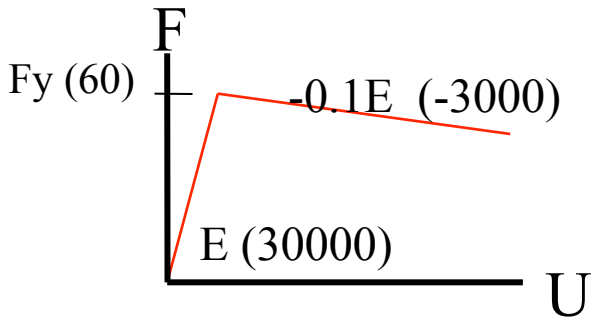
With a Yield Strength of 60
This is as far as we can push the
Model using LoadControl

We can go further using a Displacement Control scheme

Spring Example - Displacement Control

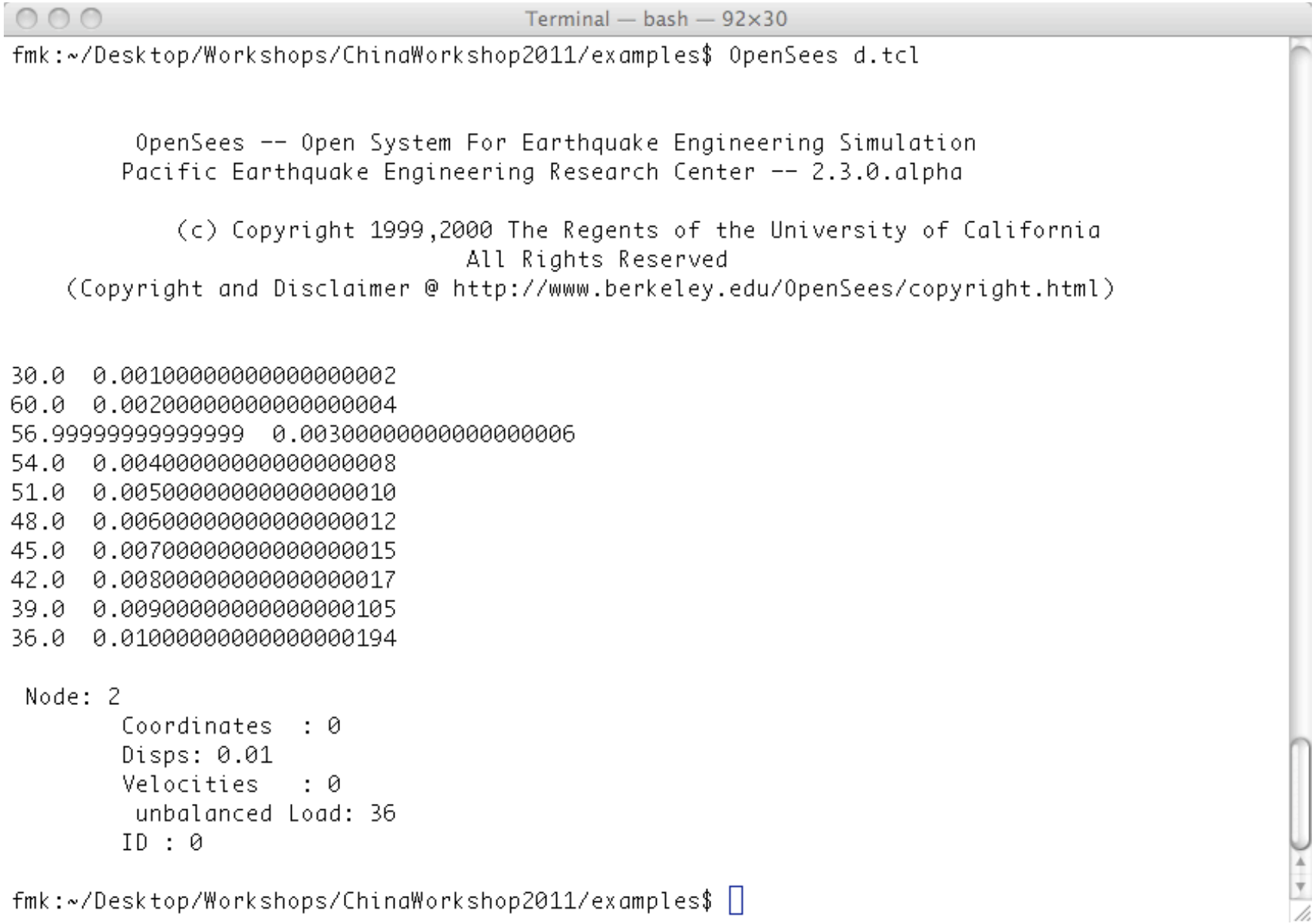
d.tcl





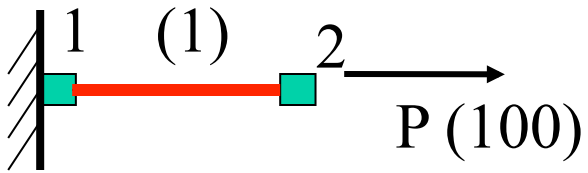

```
# create the model builder
model Basic -ndm 1 -ndf 1

# 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
}
```

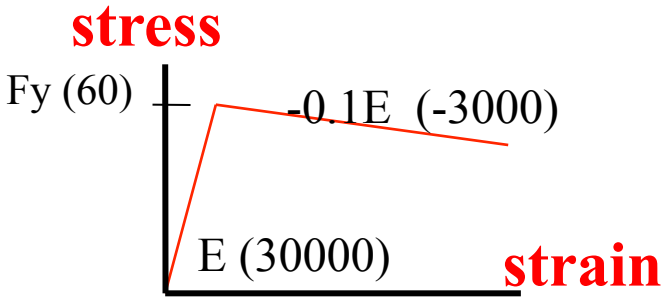
```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-12 6 0
algorithm Newton
system BandGen
integrator DisplacementControl 2 1 0.001
analysis Static

# 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
```


Terminal — bash — 92×30
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 0.0010000000000000002
60.0 0.0020000000000000004
56.99999999999999 0.0030000000000000006
54.0 0.0040000000000000008
51.0 0.0050000000000000010
48.0 0.0060000000000000012
45.0 0.0070000000000000015
42.0 0.0080000000000000017
39.0 0.0090000000000000105
36.0 0.0100000000000000194
Node: 2
Coordinates : 0
Disps: 0.01
Velocities : 0
unbalanced Load: 36
ID : 0
fmk:~/Desktop/Workshops/ChinaWorkshop2011/examples$

Truss Example - Displacement Control

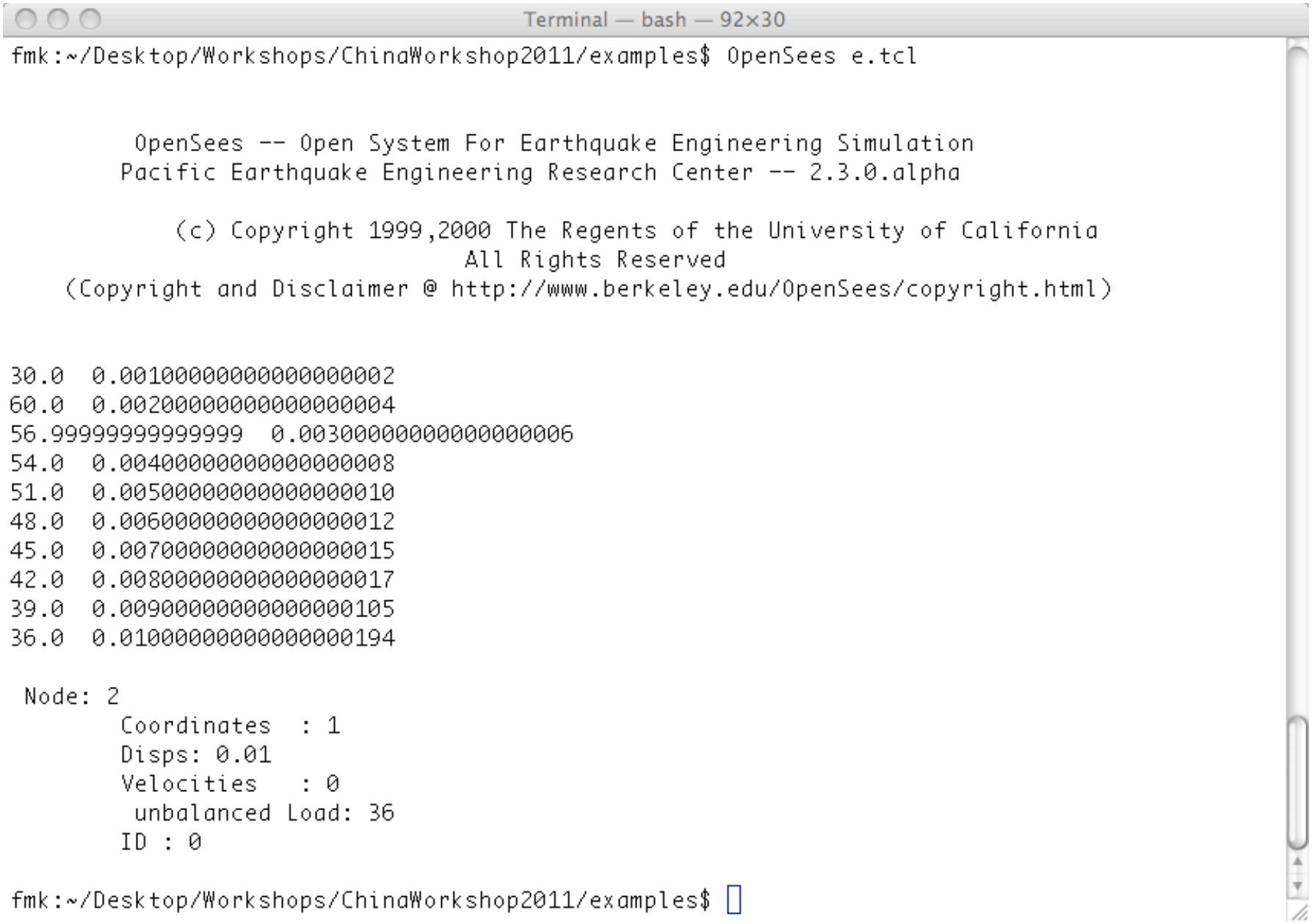
e.tcl





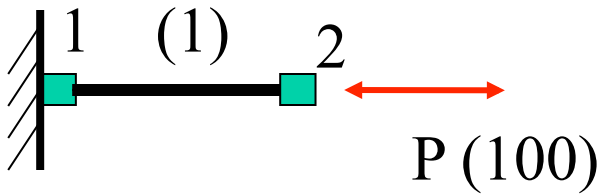

```
# create the model builder
model Basic -ndm 1 -ndf 1

# 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
}
```

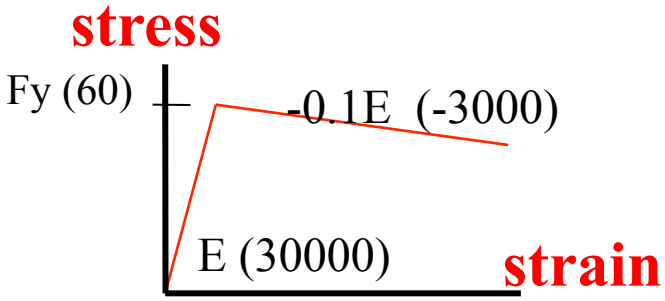
```
# create an analysis
constraints Plain
numberer RCM
test NormDispIncr 1.0e-12 6 0
algorithm Newton
system BandGen
integrator DisplacementControl 2 1 0.001
analysis Static

# 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 — 92×30
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 0.0010000000000000002
60.0 0.0020000000000000004
56.99999999999999 0.0030000000000000006
54.0 0.0040000000000000008
51.0 0.0050000000000000010
48.0 0.0060000000000000012
45.0 0.0070000000000000015
42.0 0.0080000000000000017
39.0 0.0090000000000000105
36.0 0.0100000000000000194
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



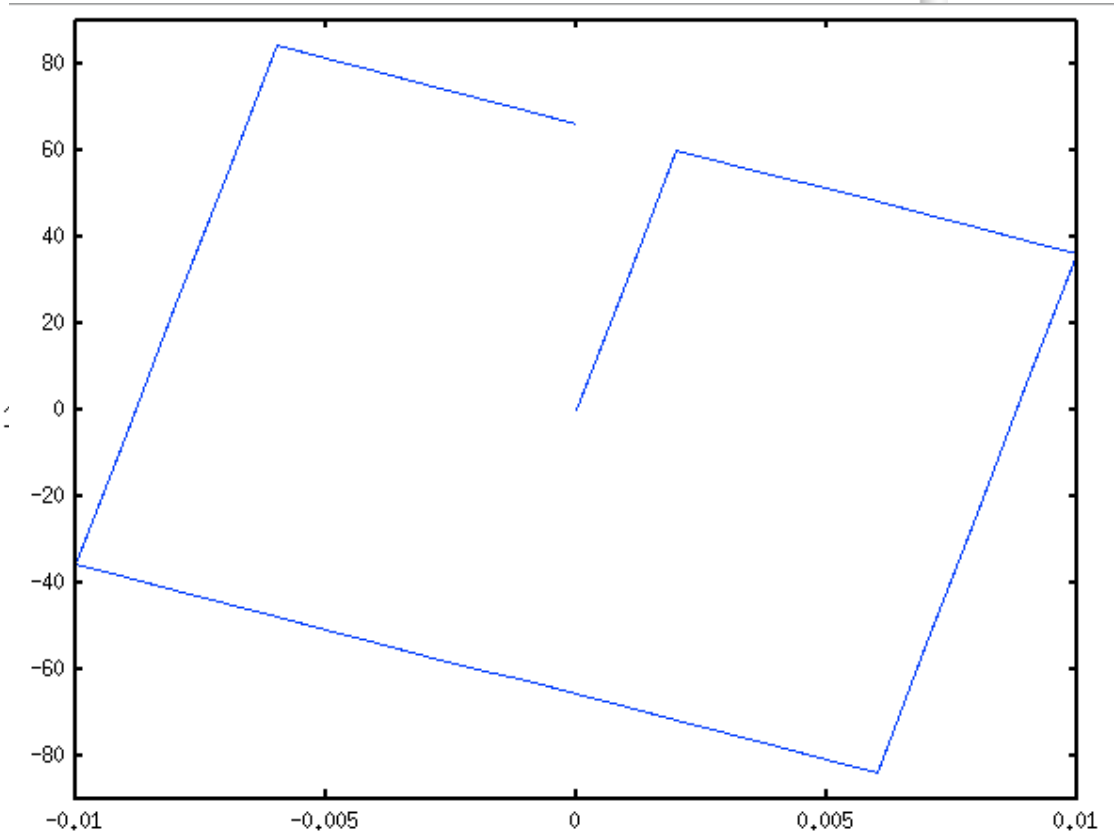



```
# create the model builder
model Basic -ndm 1 -ndf 1

# 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
```

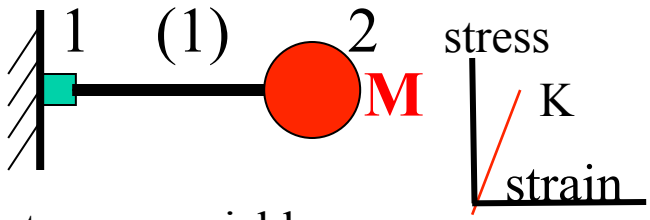
Terminal — bash — 93×23

```
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  0.010000000000000194
-36.0  -0.010000000000000194
66.0  0.0000000000000000000

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

fmk:~/Desktop/Workshops/ChinaWorkshop20
```



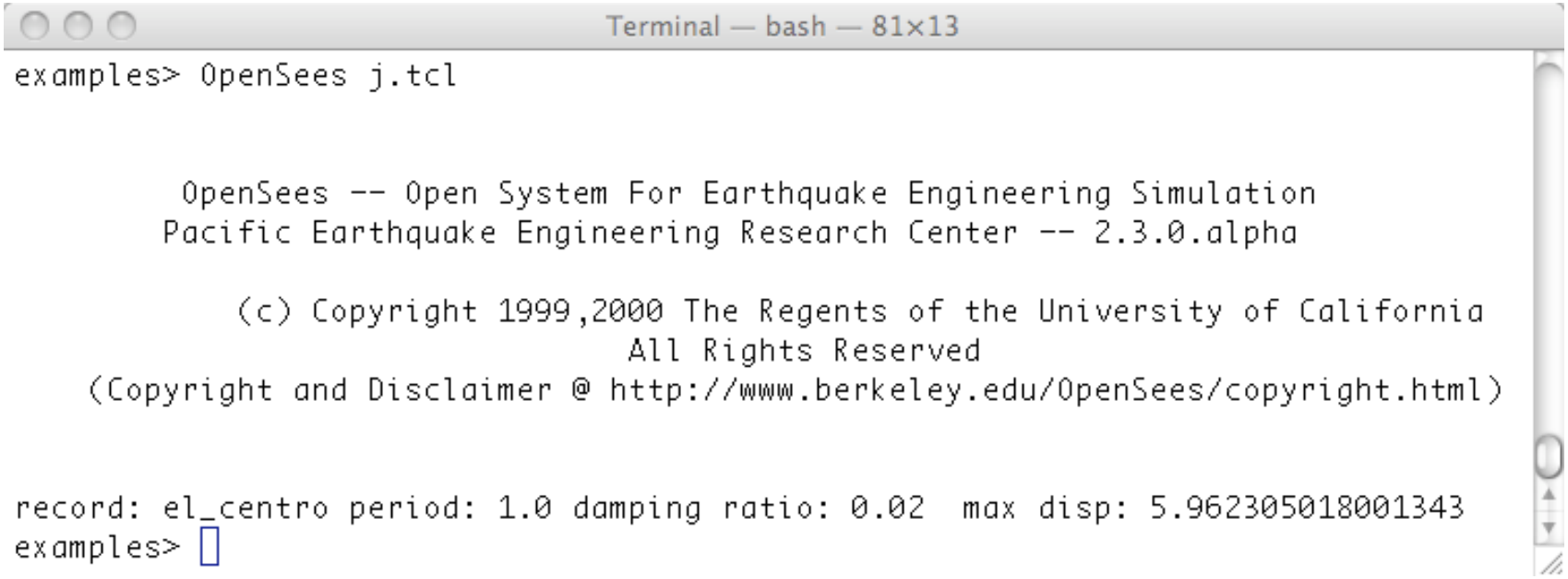
Truss Example - Uniform Excitation

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 quantaties
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 — 81×13
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

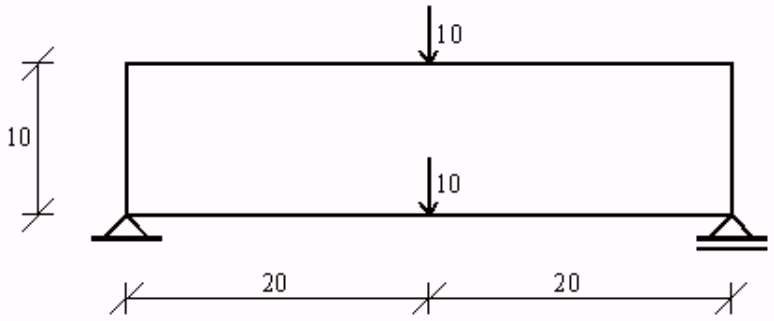


Fig. 1 Geometry and static loads

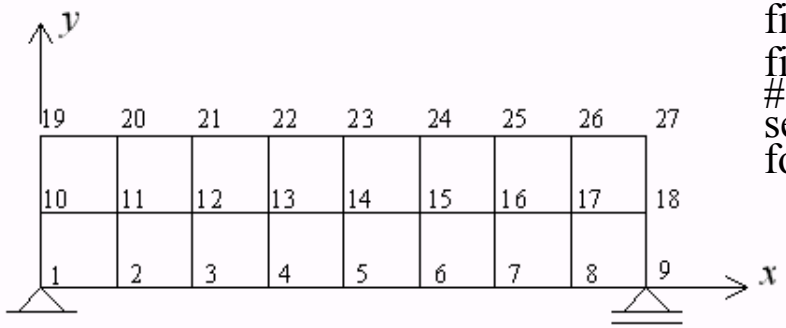


Fig. 2 Finite element mesh and node numbering

```
# 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 lN
    }
}

# 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

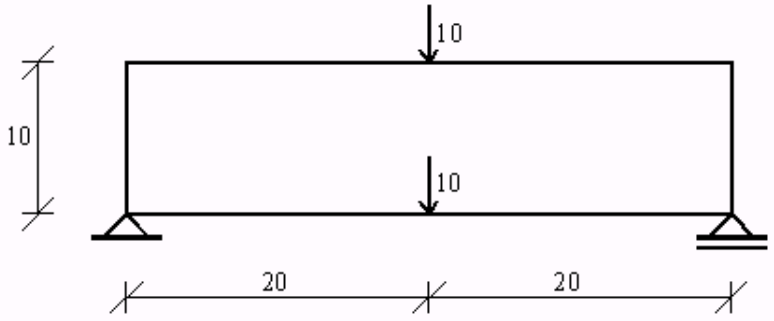


Fig. 1 Geometry and static loads

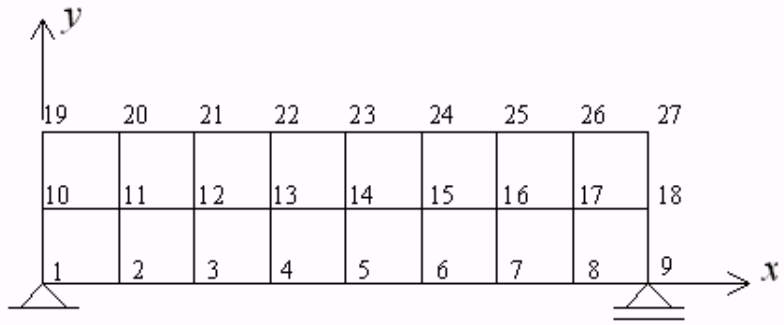
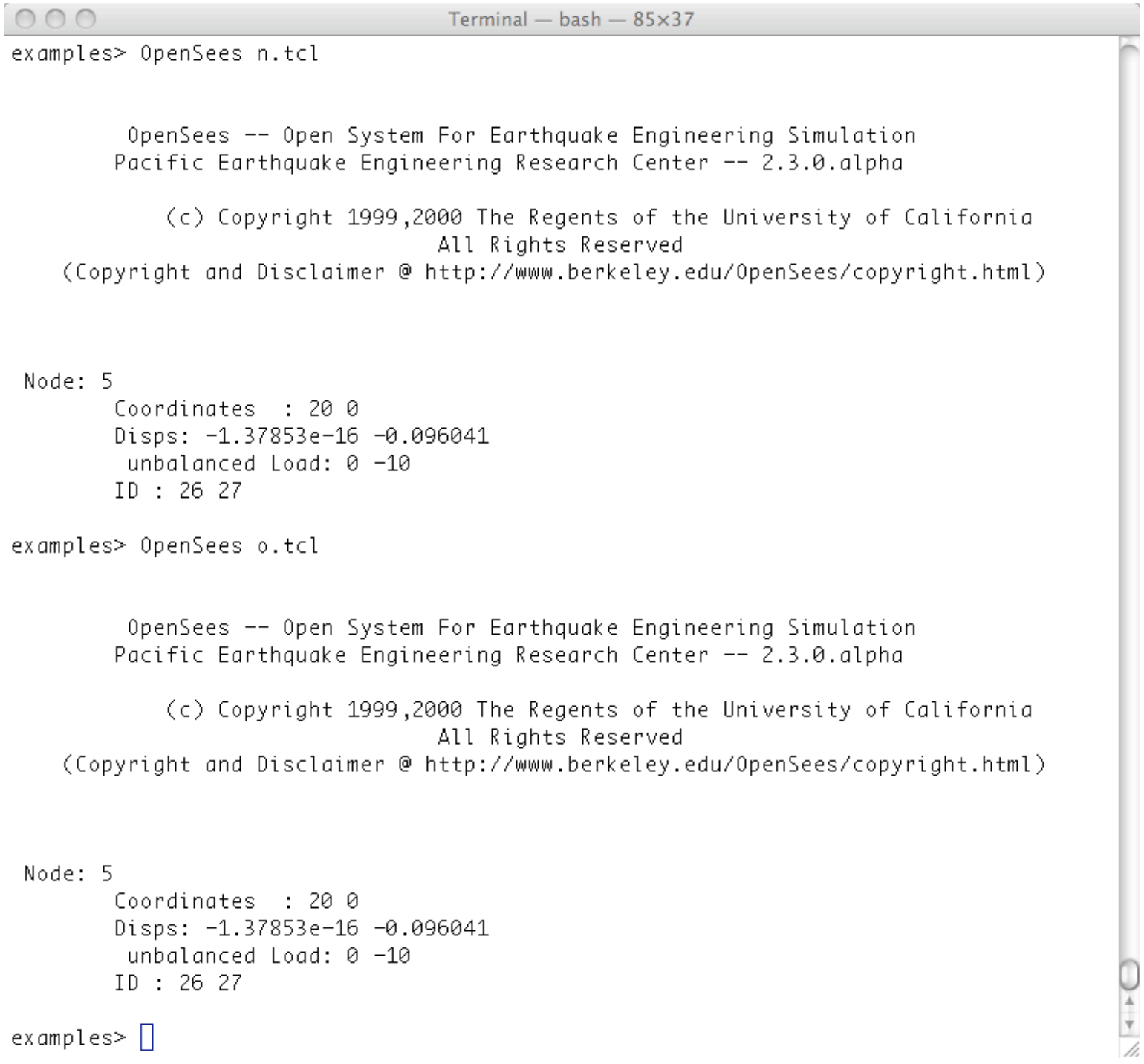


Fig. 2 Finite element mesh and node numbering

```
# 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
```

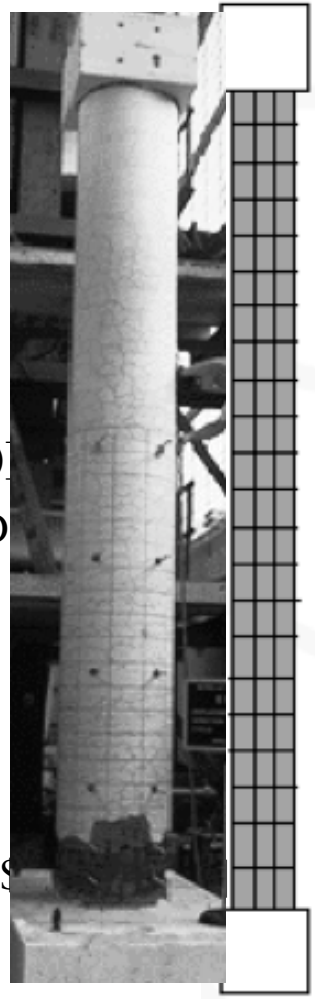
```
# use block command
set cmd "block2D [expr $nX-1] [expr $nY-1] 1 1 \
        quad \" $thick PlaneStress 1\" {
      1   0   0
      2  $L   0
      3  $L  $H
      4   0  $H
}"

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 — 85×37
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

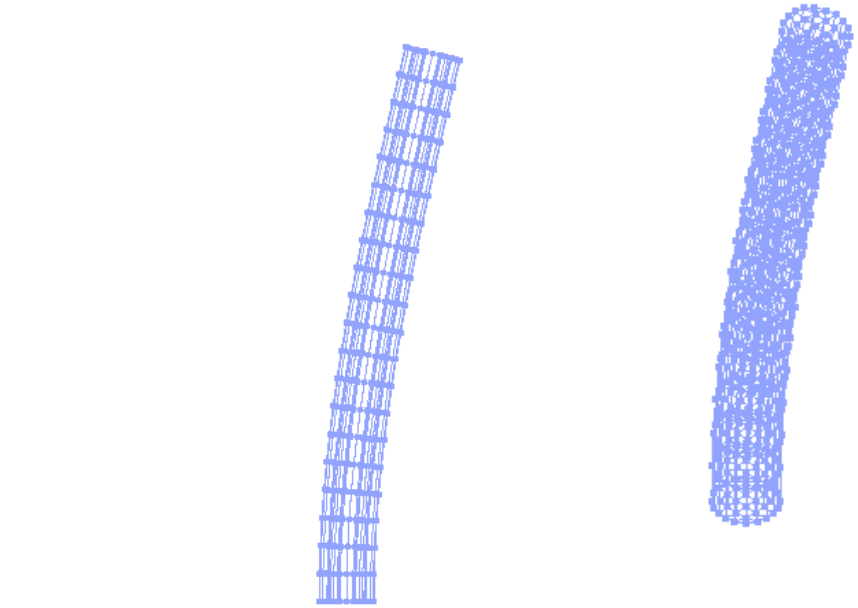
```
set P 1.0
set L 20.0
set R 1.0
set E 1000.0
set nz  20
set nx  6
set ny  6
set PI [expr 2.0 * asin(1.0)]
set I  [expr $PI*pow((2*$R),4)/64.0]
puts "PL^3/3EI = [expr $P*pow($L,3)/(3.0*$E*$I)]

# 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)*($
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  $ele
      1   -$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 — 79×22

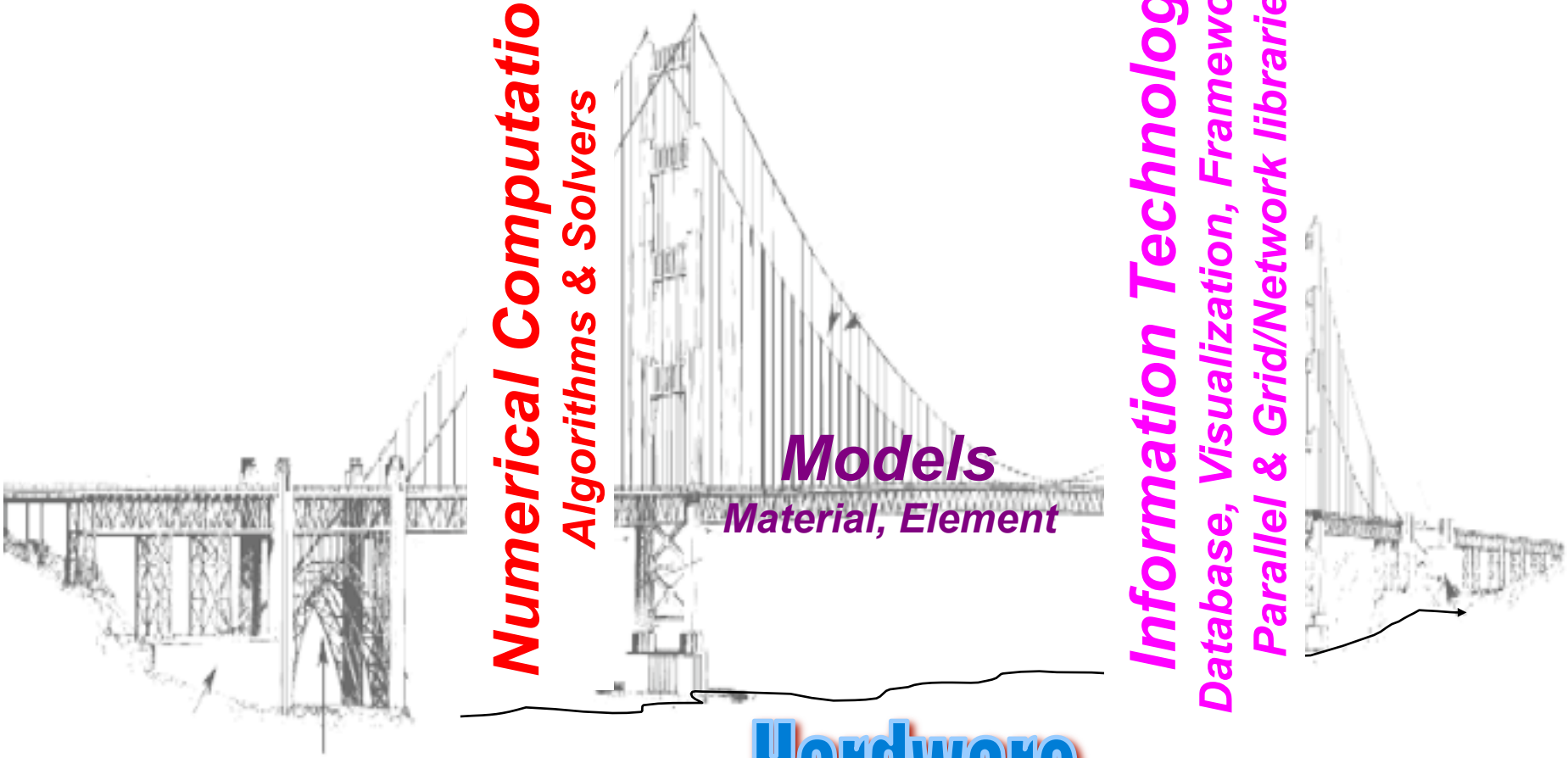
```
examples> OpenSees


         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

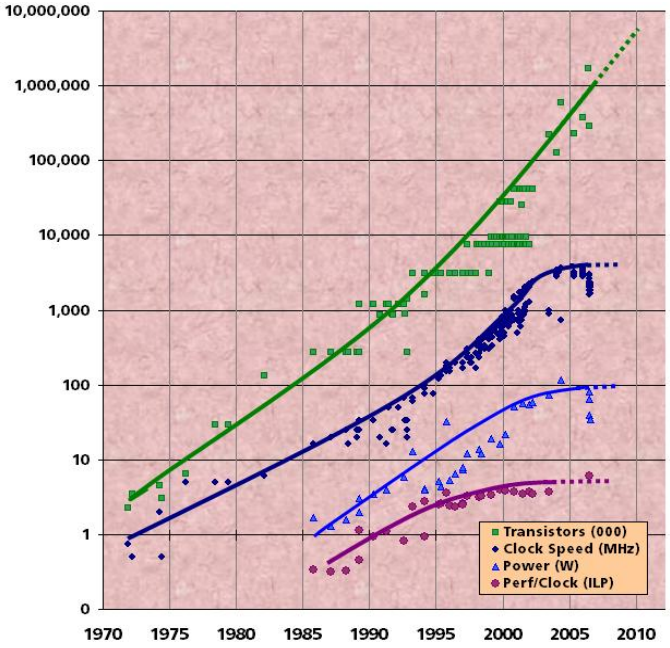


Bell's Law

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.

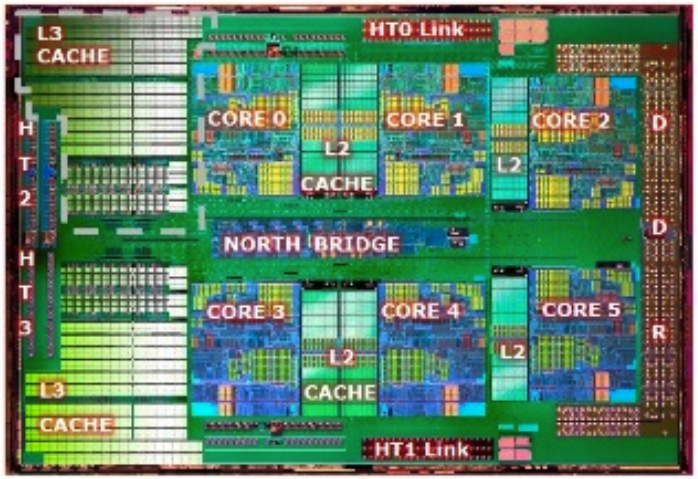
Gordon Bell, http://research.microsoft.com/~GBell/Pubs.htm

Technology is Changing



Intel Processor Speed	
XeonE7Server	72Gflop
i7Desktop	55GFlop
i7Mobile	30GFlop
i5Desktop	40GFlop
i5Mobile	22GFlop
Core2 Quad	48GFlop
Core2 Duo	25GFlop

CHIP/Socket

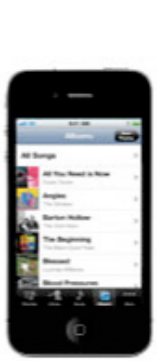


GPU &
CO-Processors

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”

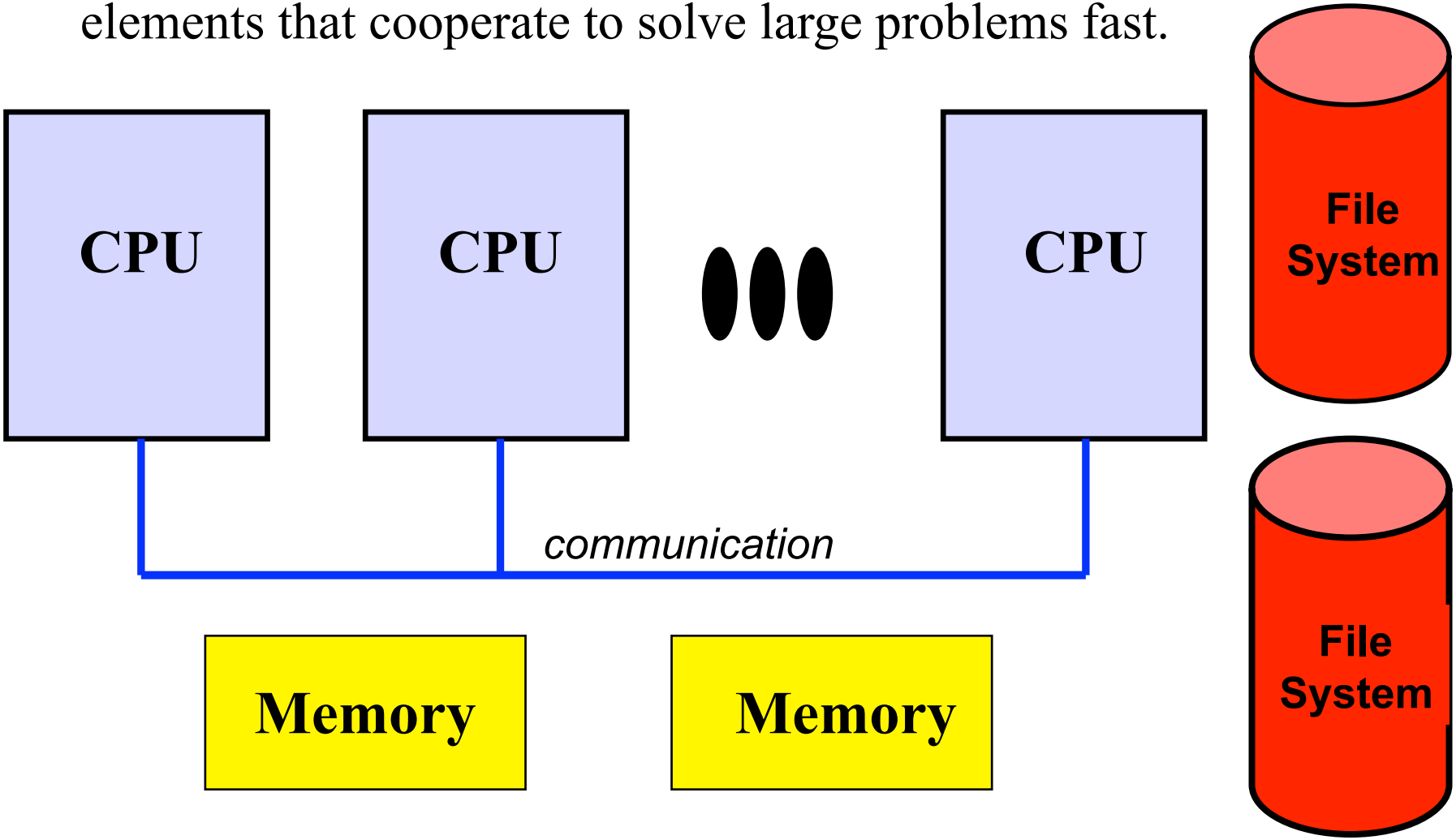
Cloud Computing **Compute Resources**

- Commercial

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

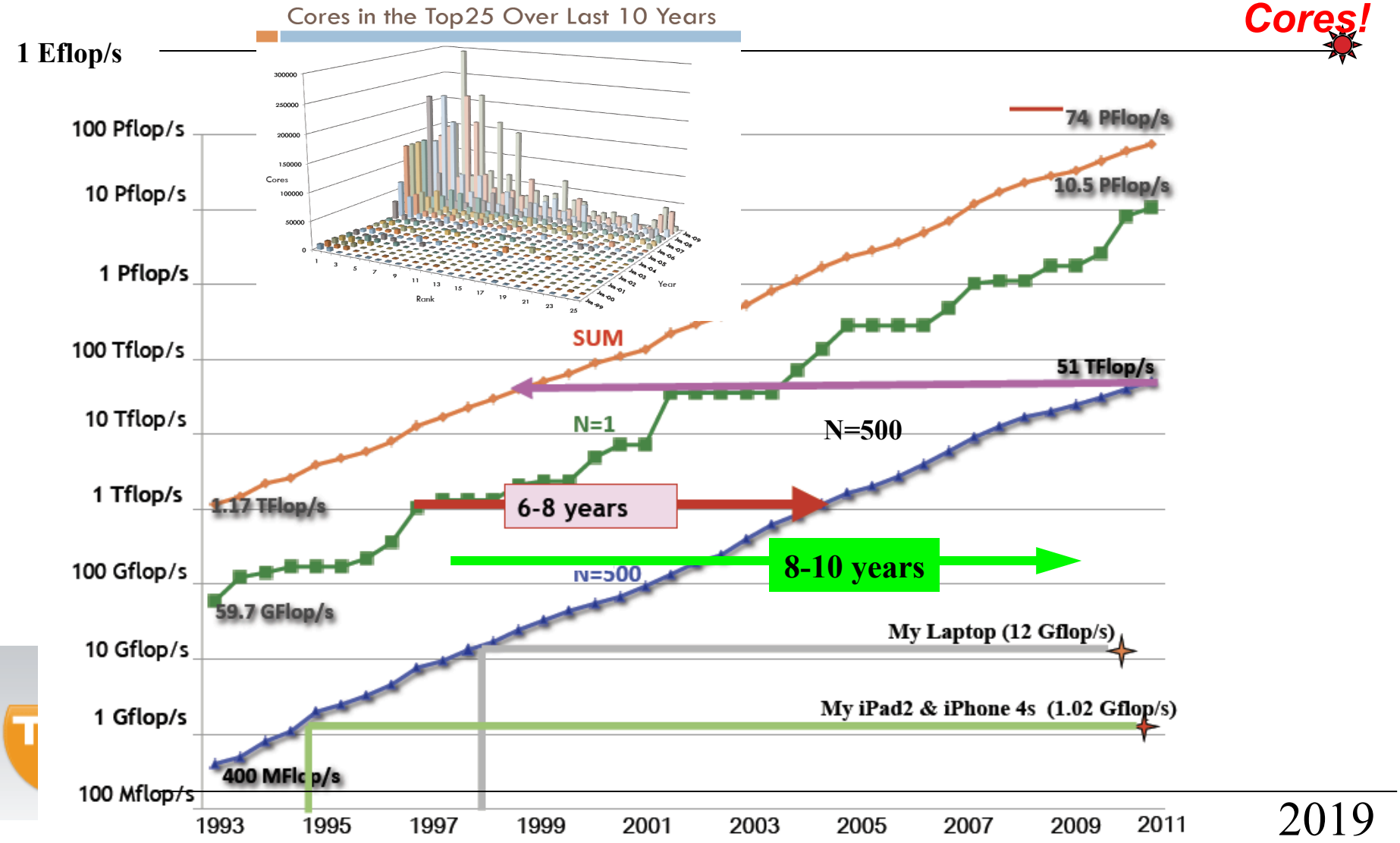
What is a Parallel Computer?

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



Projections

10e6 to 100e Cores!



Source:Jack Dongarra, 2011

Why should you care?

- They will save you time
- They will allow you to solver 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

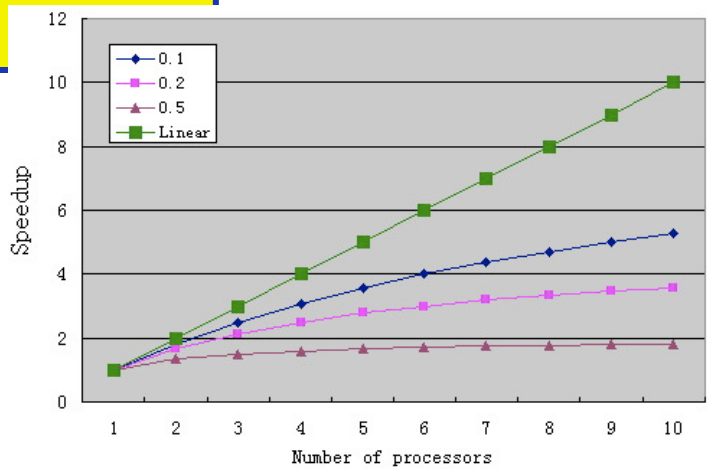
Speedup & Amdahl's Law

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

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

Portion of sequential

# of processors



Improving Real Performance

Peak Performance grows exponentially, a la Moore's Law

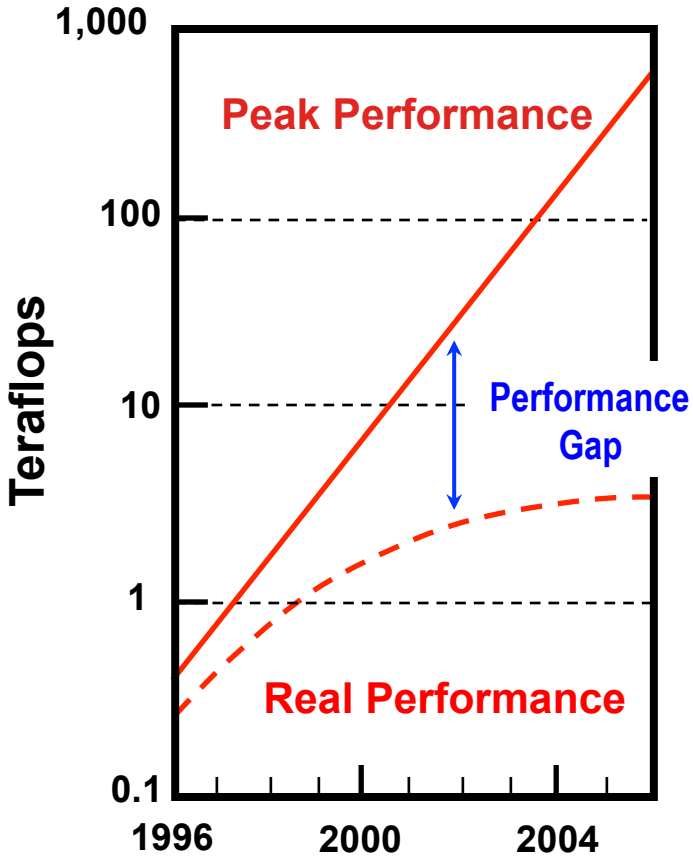
- In 1990's, peak performance increased 100x; in 2000's, it will increase 1000x

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

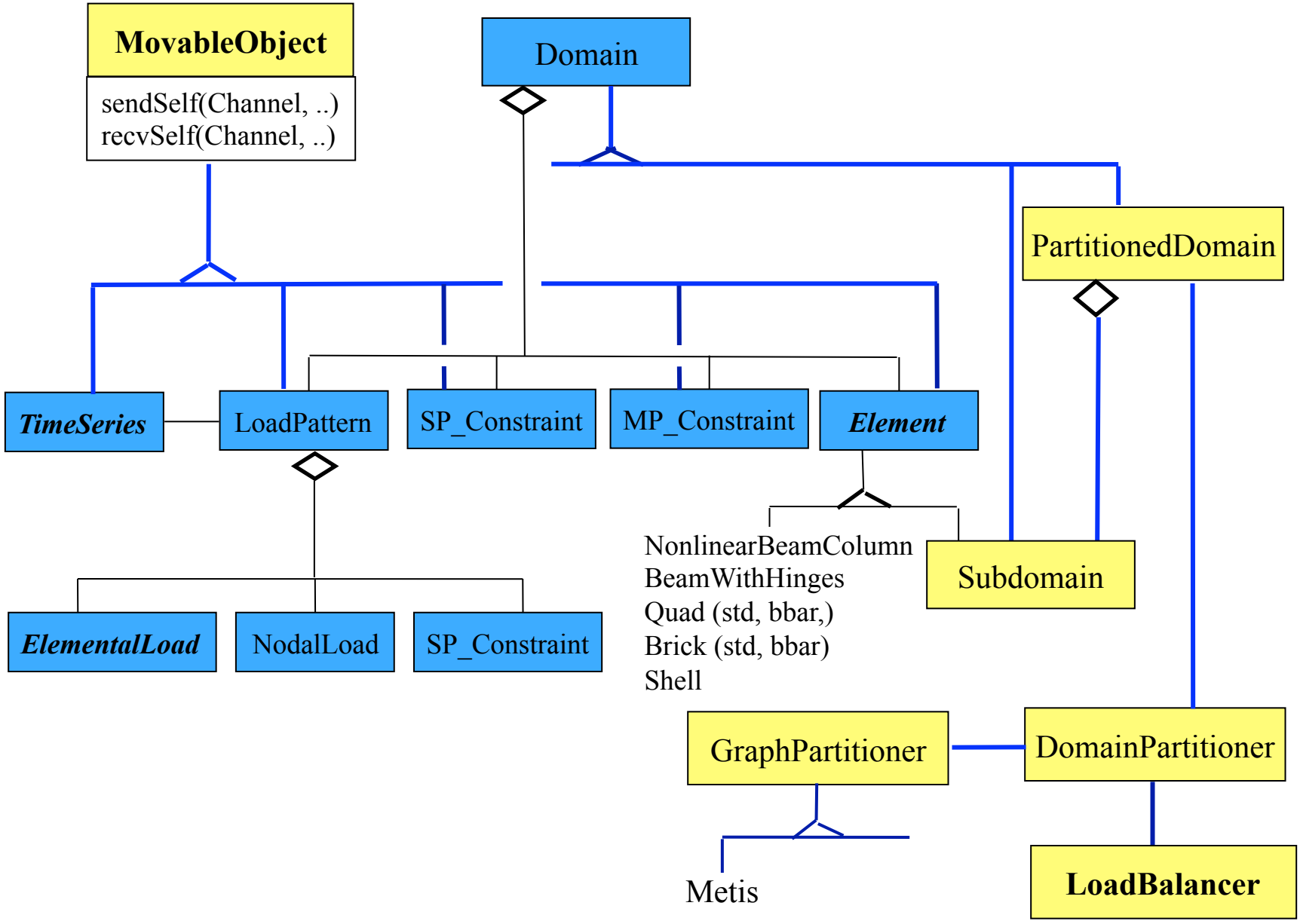


Source: Jim Demmell, CS267 Course Notes

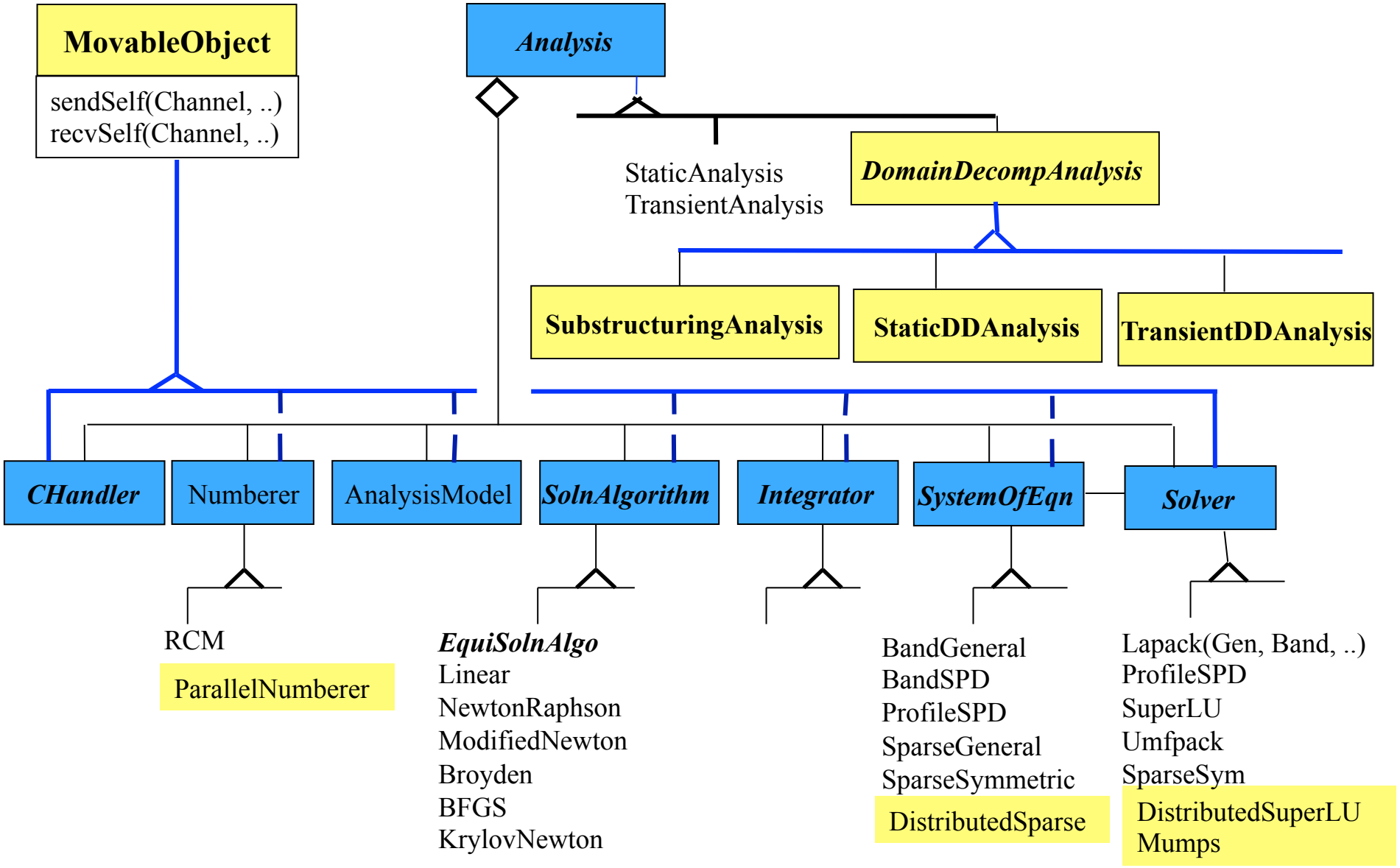
What is OpenSees?

- 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

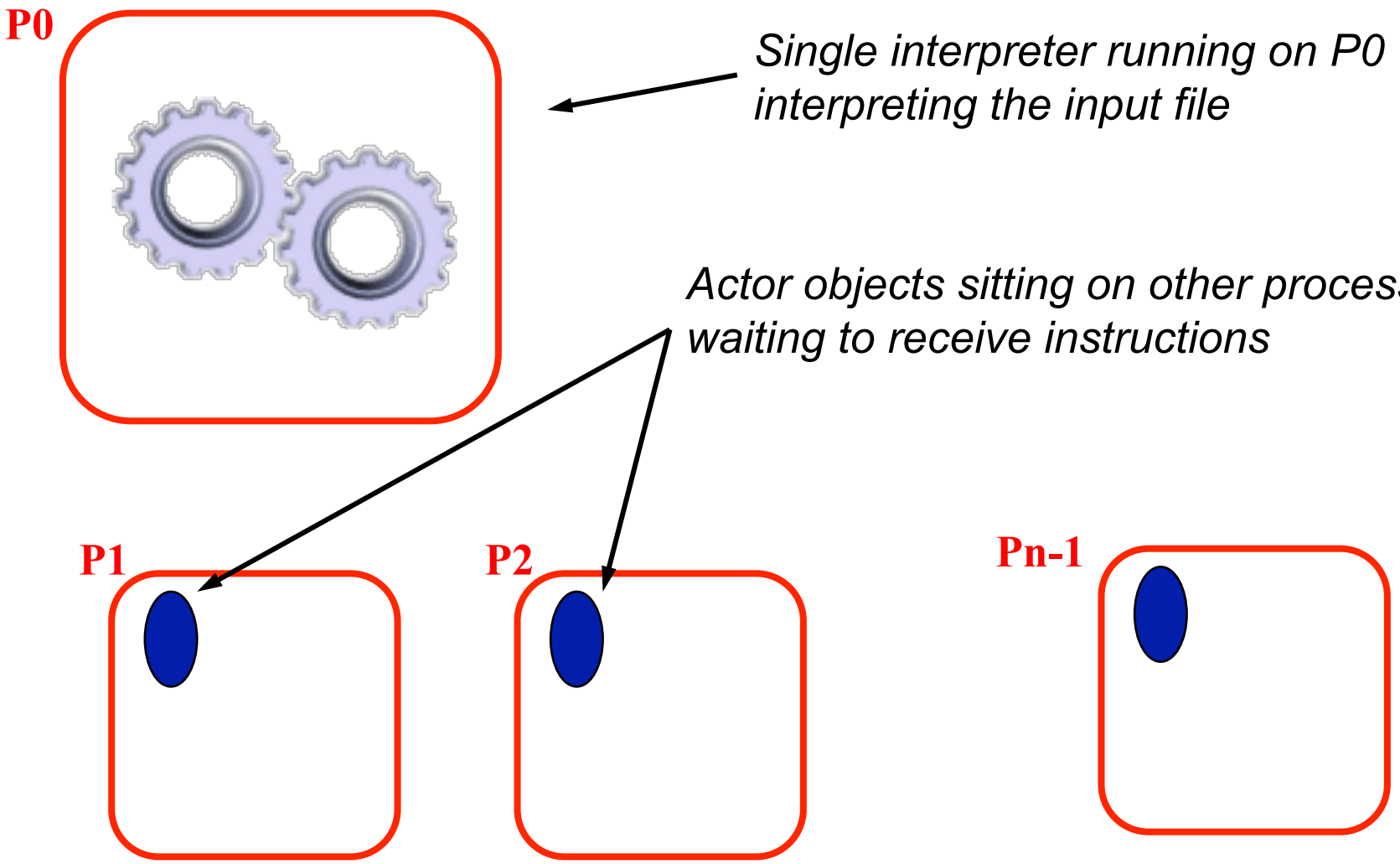
- OpenSees.exe, OpenSeesSP.exe and OpenSeesMP.exe are applications that extend the Tcl interpreter for finite element.

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

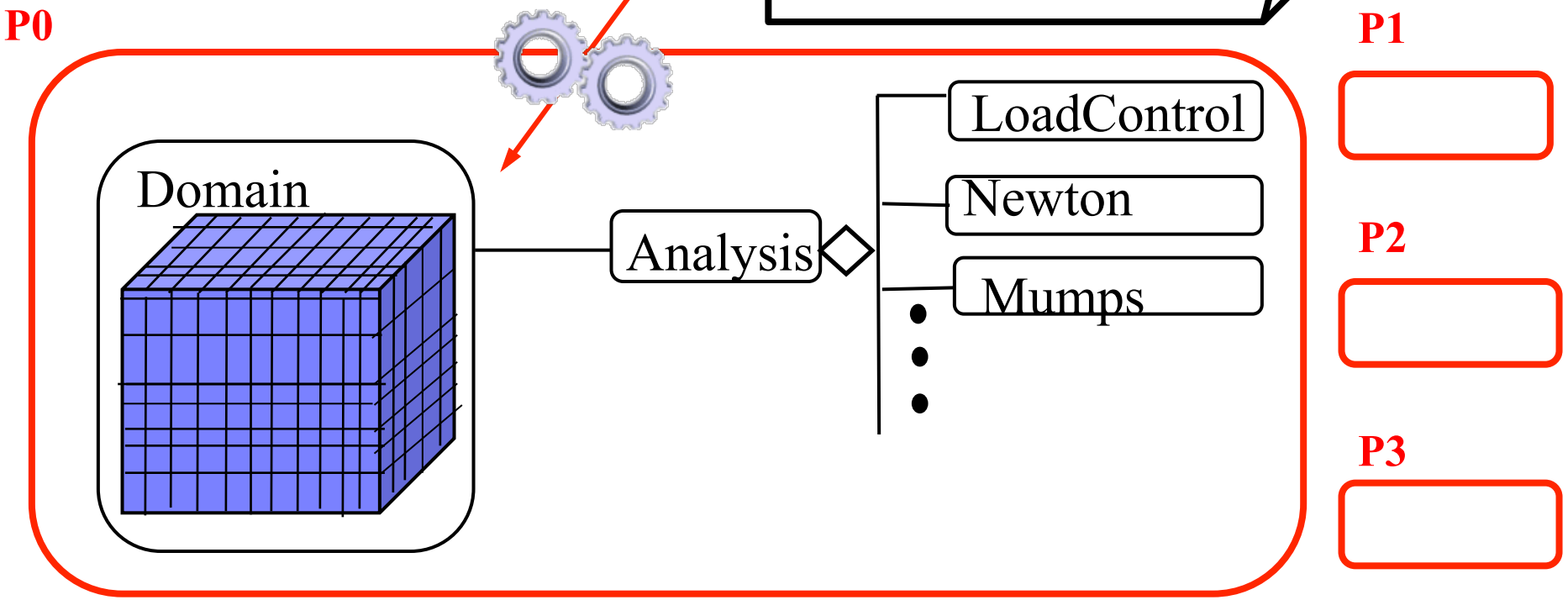
system Diagonal

Model Built and Analysis Constructed ir P0

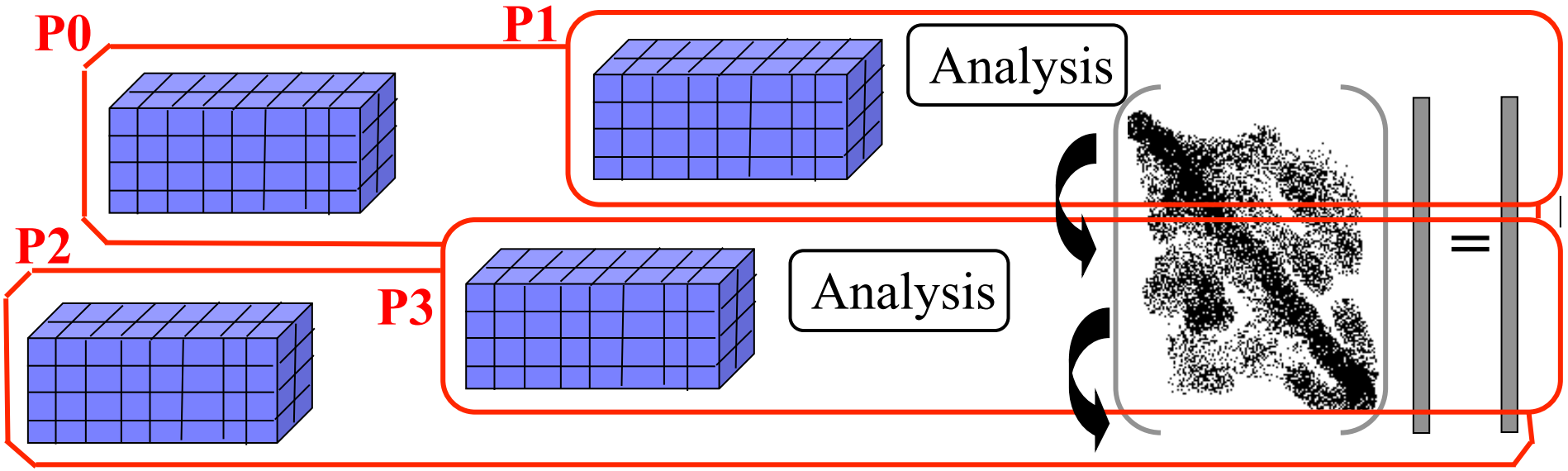
```
#build the model
source model.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDispIncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
```

Single interpreter running on P0

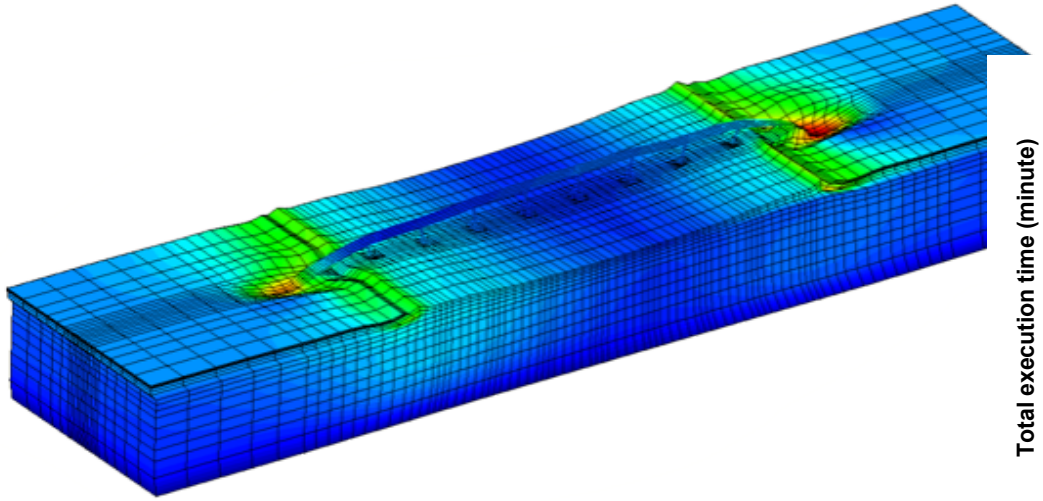
Interpreting the input file

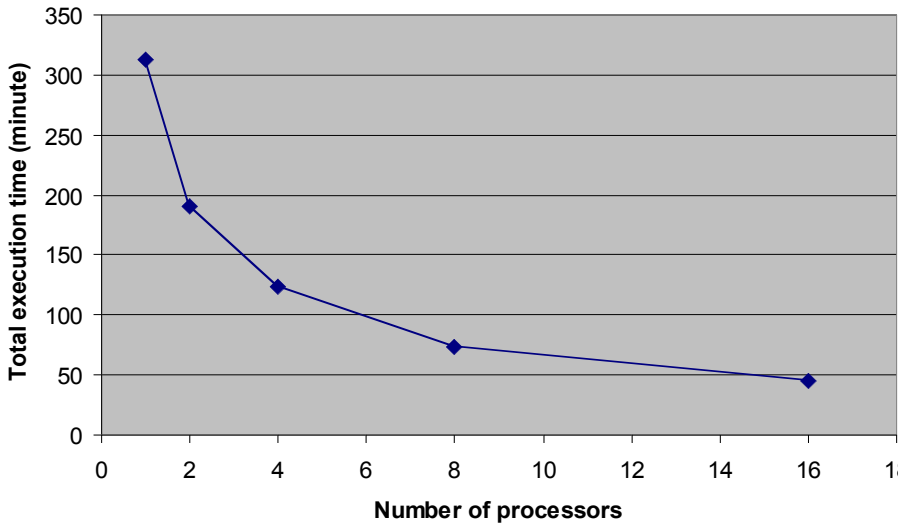


```
#build the model
source modelP.tcl
#build the analysis
system Mumps
constraints Transformation
numberer Plain
test NormDispIncr 1.0e-12 10 3
algorithm Newton
integrator LoadControl
analysis Static
analyze 10
```

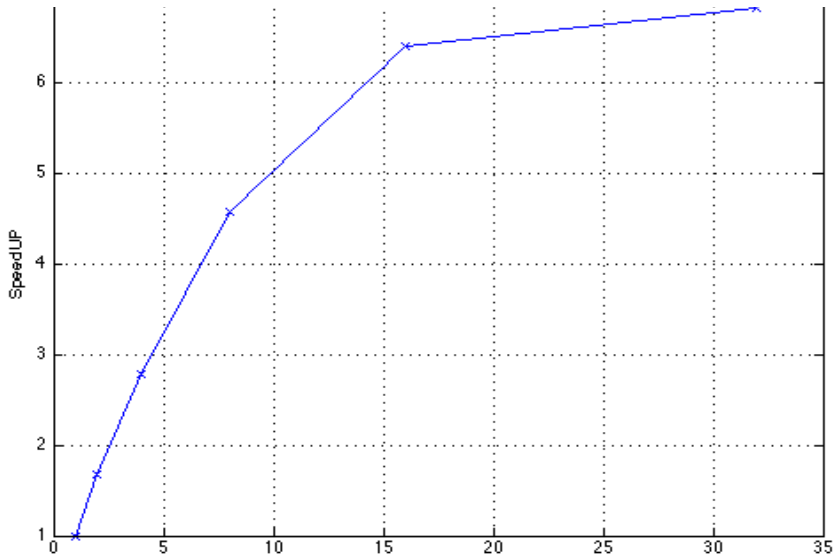


Example Results: Humboldt Bay Bridge Model



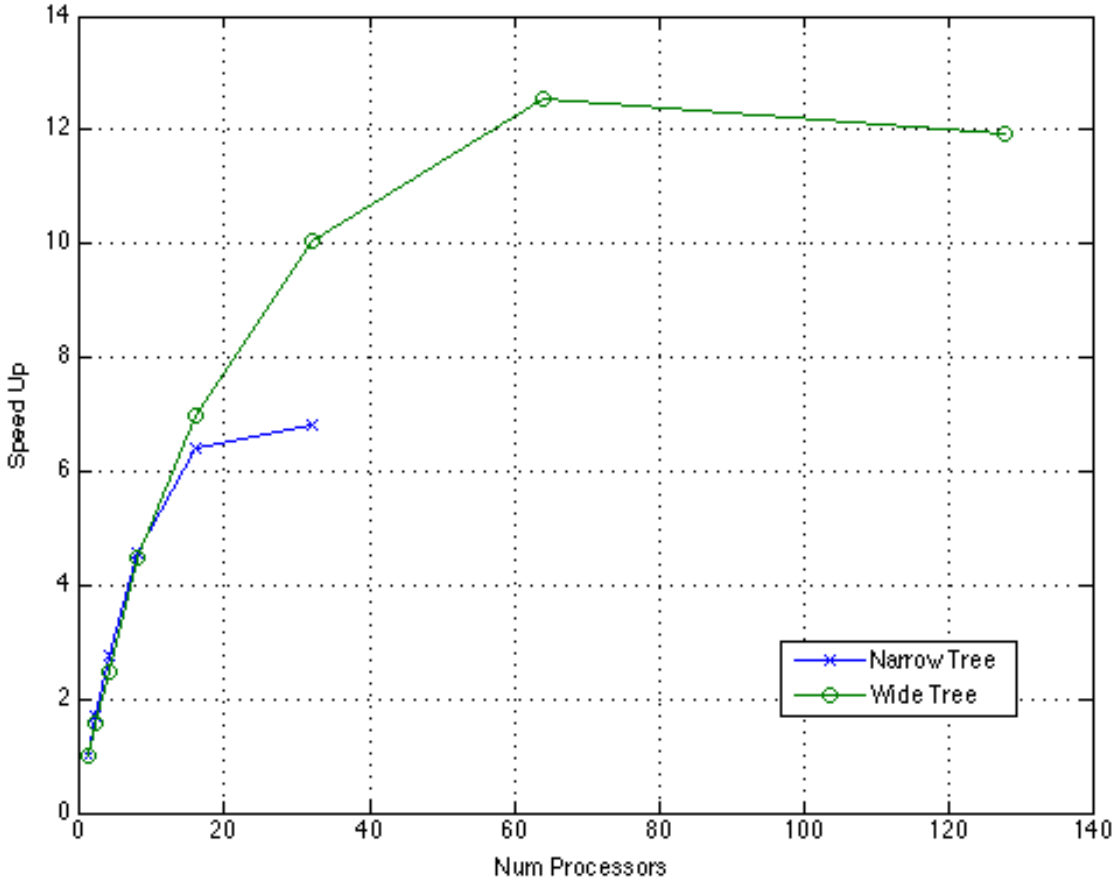


100,000+ DOF Model
Implicit Integration
Mumps Direct Solver



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

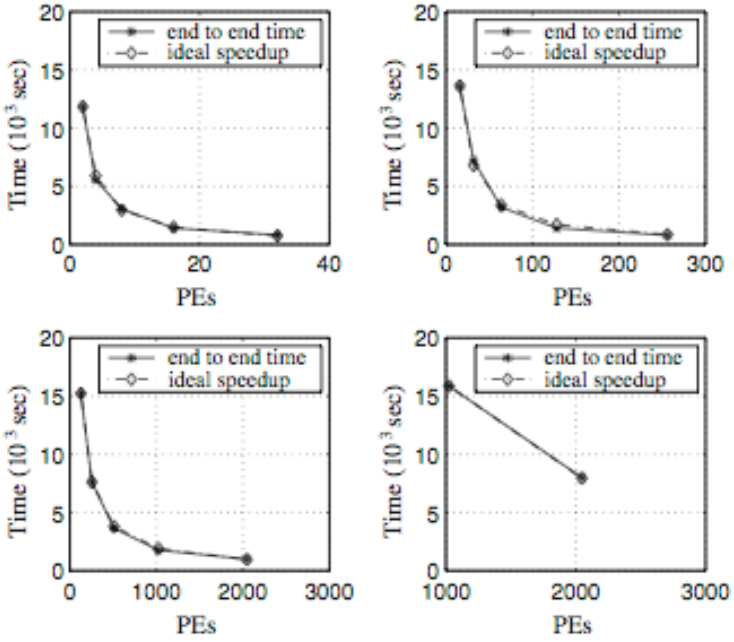
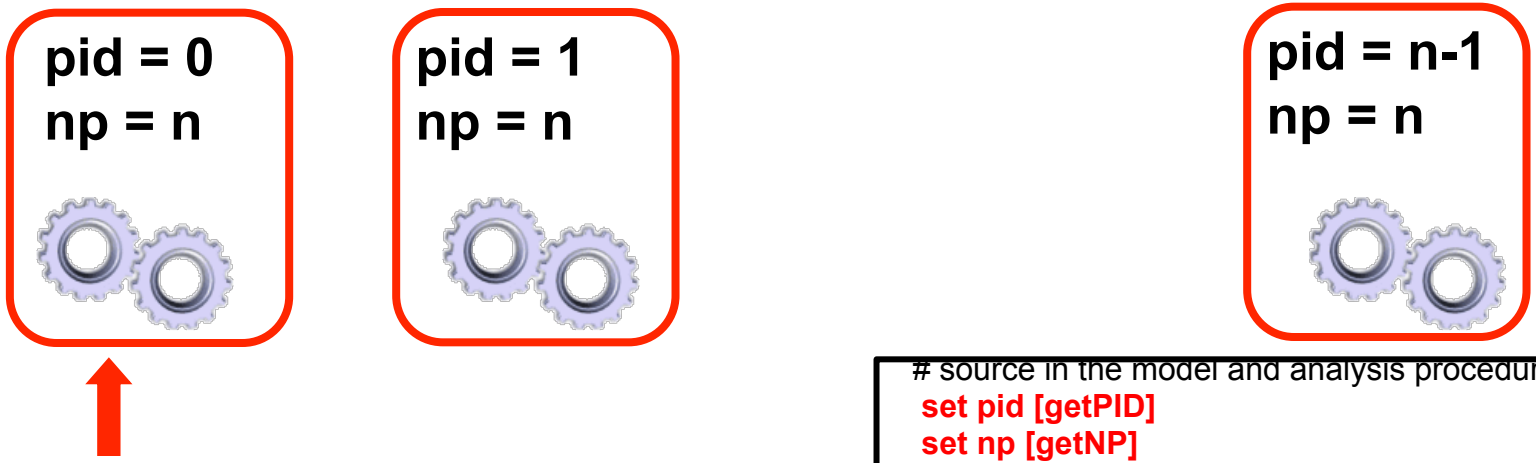


Fig. 18 Fixed-size, scalability plot at SDSC's DataStar. Upper row is runs A (*left*) and B (*right*), lower row is runs C (*left*) and D (*right*) (Table 3)

OpenSeesMP: An application for Large Models and Parameter Studies



Each process is running an interpreter and can determine it's 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 NormDispIncr 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 pocess 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

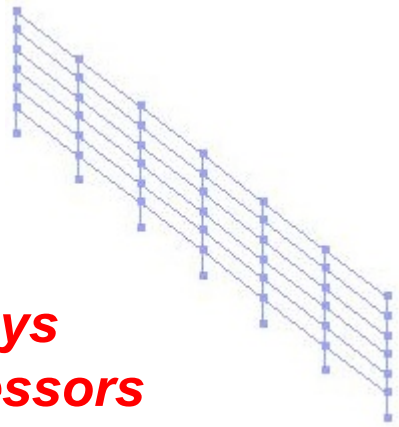
```
set pid [getPID]
set np [getNP]
if {$pid == 0 } {
   puts "Random:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv -pid ANY msg
      puts "$msg"
   }
} else {
   send -pid 0 "Hello from $pid"
}
barrier
if {$pid == 0 } {
   puts "\nOrdered:"
   for {set i 1 } {$i < $np} {incr i 1} {
      recv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

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/OpenSe


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

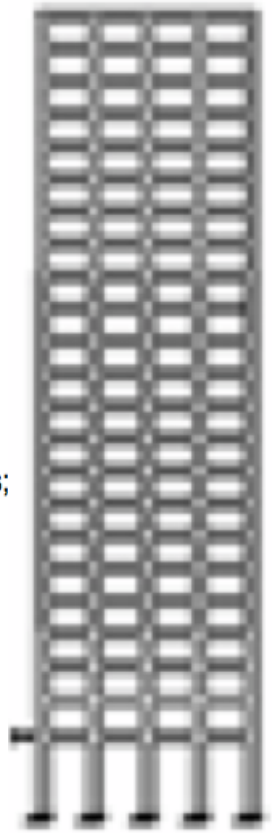
```
set pid [getPID]
set np [getNP]
set recordsFileID [open "peerRecords.txt" r]
set count 0;

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;
}
```

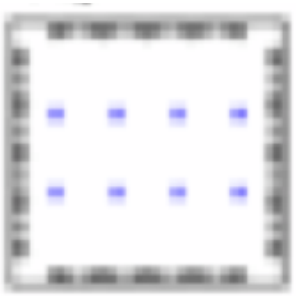


7200 records
2 min a record
240 hours or 10 days
Ran on 2000 processors
on teragrid in less than 15 min.

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.

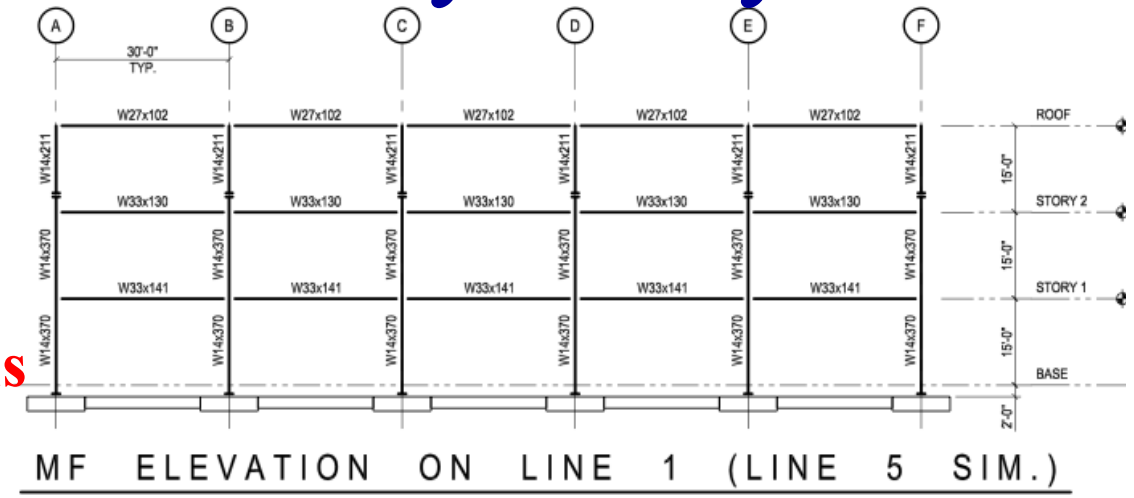
Ran on 452 processors
On XSEDE in less than 5 hours.

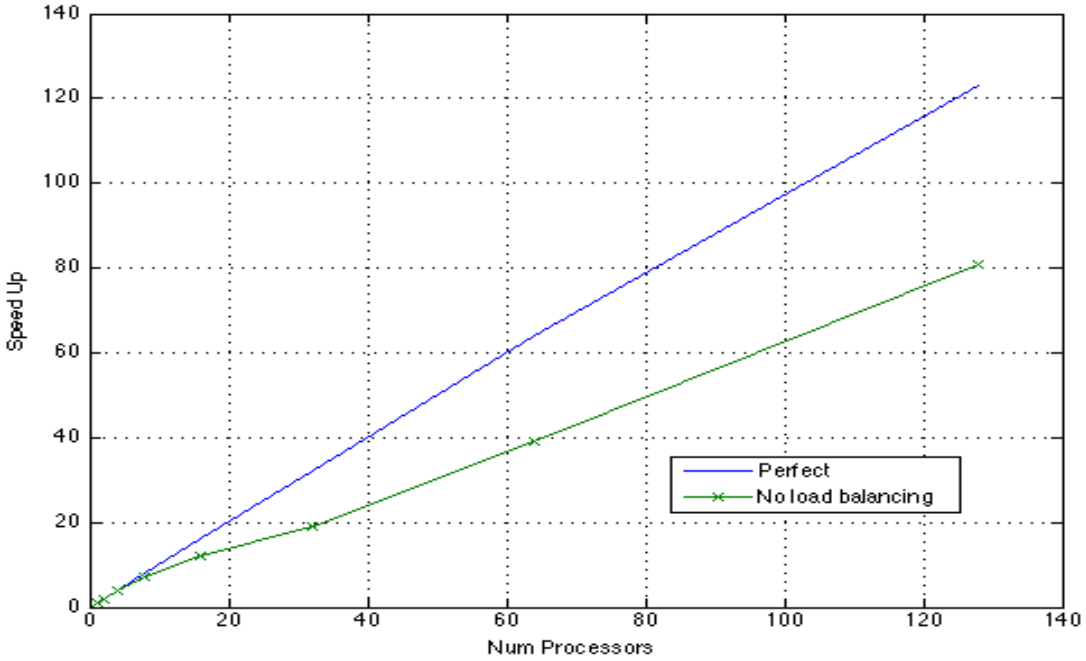
ATC-63/2 Project

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

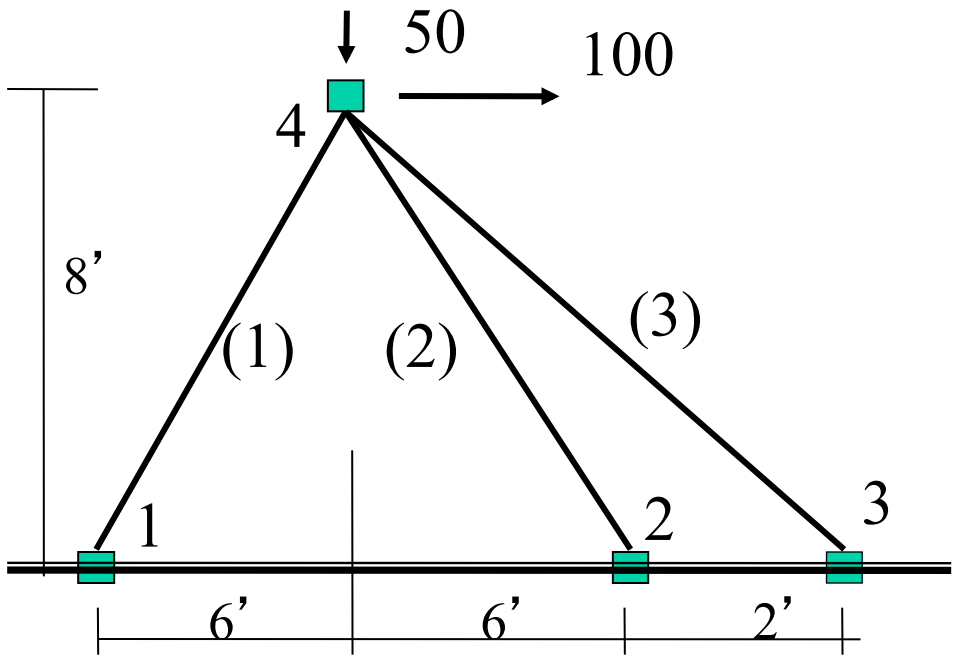
```
integrator ParallelDisplacementControl node? Dof? dU?
```

- Use these ONLY IF PARALLEL MODEL

Example Parallel Model:

ex4.tcl

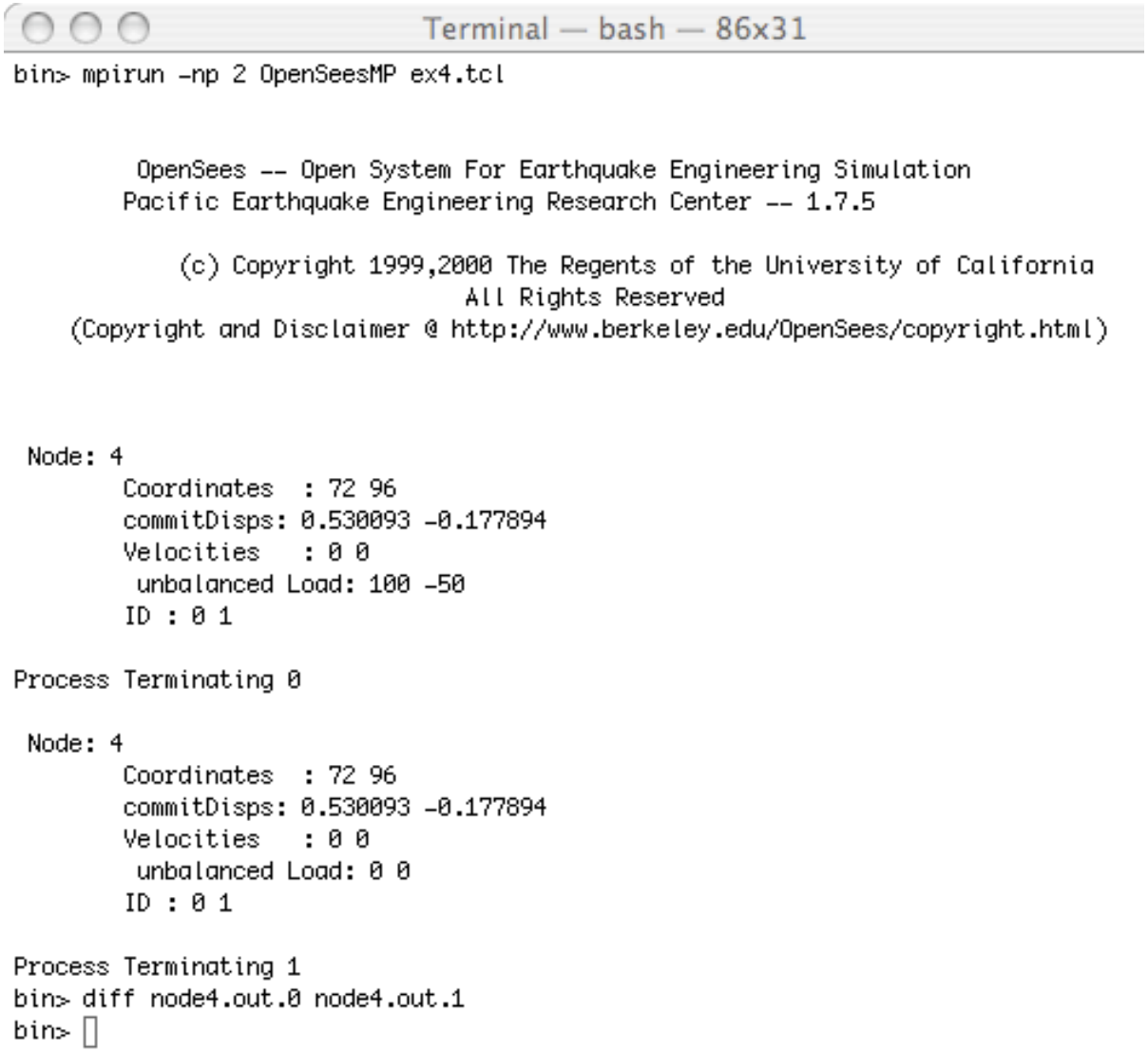
```
set pid [getPID]
set np [getNP]
if {$np != 2}  exit

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
```



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: 280.668 -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.

Things to Watch For Using Additional Commands

In addition to Load Imbalance:

1. Deadlock (program hangs)
   - send/recv messages
   - Opening files for writing & not closing them
2. Race Conditions (different results on re-run)
   - parallel file system.

These I am not going to demonstrate today .. But BE WARNED THEY DO HAPPEN AND YOU PROBABLY WILL COME ACROSS THEM!

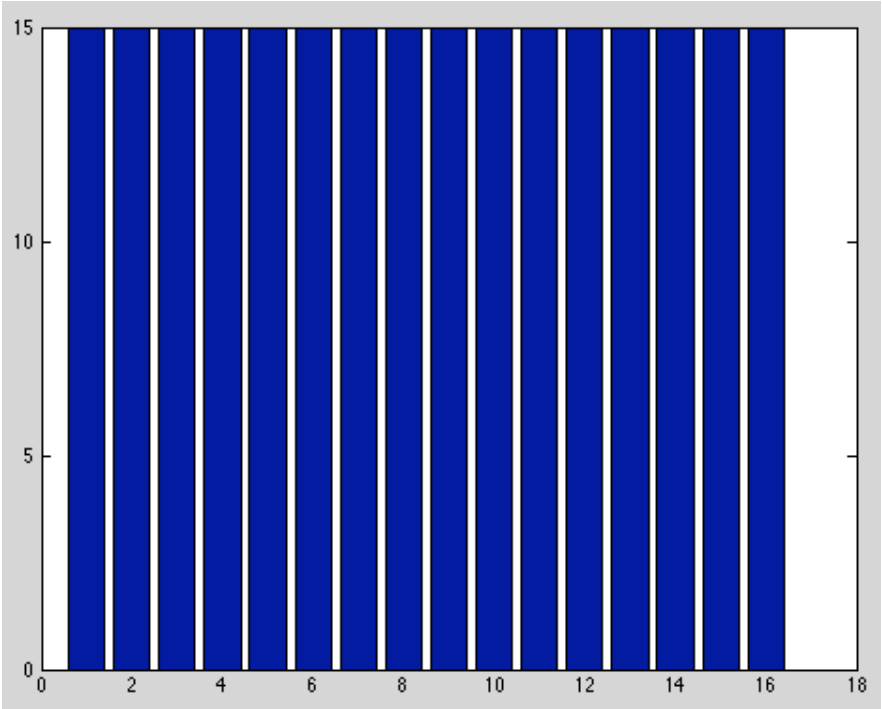
Parameter Study

MRF_MP1.tcl

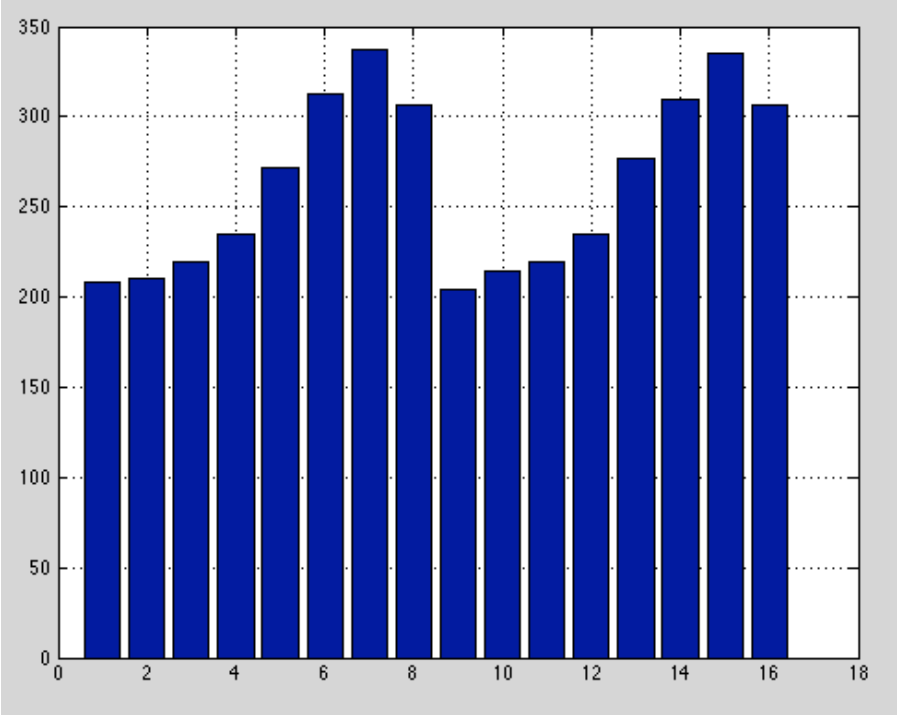
```
set np [getNP]
set pid [getPID]
set count 0
# set some parameters & open some local files
….
if {$pid == 0} {
   foreach Hazard $Hazards {
      set dataDir [format "Oak-%i-50-Output" $Hazard]
      file mkdir $dataDir/DriftAcceleration;
      file mkdir $dataDir/DriftDeformation;
}
barrier
foreach Hazard $Hazards {
   foreach eqDir $eqDirections {
      for { set i 1 } { $i <= $eqNum } { incr i} {
         if {[expr $count % $np] == $pid} {
            set tStart [clock clicks –milliseconds]
            set GMdir [format "./GMfiles/Oak-%i-50/" $Hazard]
            source buildModelWithGravity.tcl
            source addRecorders.tcl
            source Dynamic.EQ.tcl
            source processResults.tcl
            set tEnd [clock clicks -milliseconds]
            puts $timeOut "DURATION: $i $eqDir $Hazard [expr ($tEnd-$tStart\)/1000.]"
         }
         incr count 1
      }
   }
}
set tEnd [clock clicks -milliseconds]
puts $timeOut "TOTAL DURATION: [expr ($tEnd-$tStartAll)/1000.]"
puts  "PID: $pid DURATION: [expr ($tEnd-$tStartAll)/1000.]"
# close local files
….
```

To ensure only 1 processor does stuff to file system

Results using 16 Processors



 # Tasks Completed *(pid vs # tasks)*



Duration *(pid v time)*

Load Balancing Solutions:

The key question: When is information (task costs, task dependencies, and task locality) about the load balancing problem known. Leads to a spectrum of solutions:

- Static scheduling. All information is available to scheduling algorithm, which runs before any real computation starts.
  - Off-line algorithms, eg graph partitioning, DAG scheduling (Pegasus)
- Semi-static scheduling. Information may be known at program startup, or the beginning of each timestep, or at other well-defined points. Offline algorithms may be used even though the problem is dynamic.
- Dynamic scheduling. Information is not known until mid-execution.

Source: Prof. Demmel, CS267, UC Berkeley

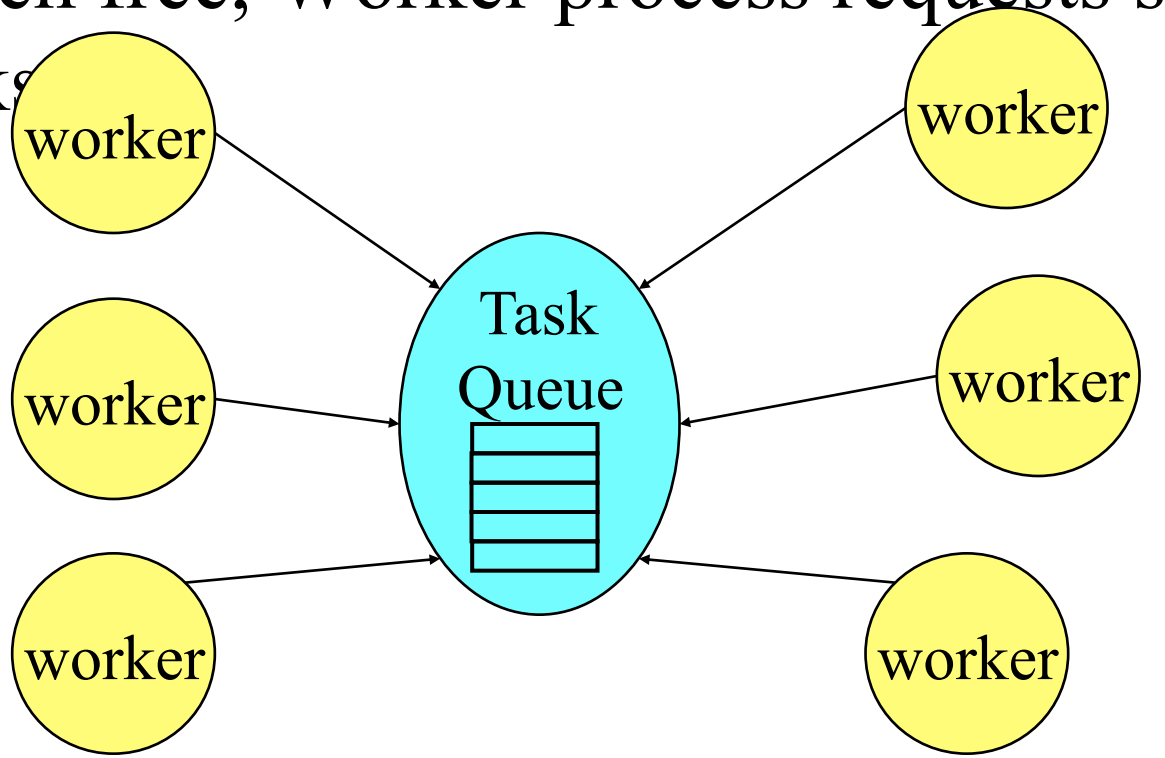
Dynamic Scheduling (Dynamic Load Balancing)

In dynamic load balancing we Adjust Task Assignment **as Processing Takes Place**. 2 Main Approaches:

- **Work Sharing**: processors balance the workload using a globally shared work queue(s) that contain tasks to be computed. If a task generates additional tasks, these can be pushed back to task queue to share.
- **Work Stealing**: processors keep own queues, idle processors search among the other processors in the computation in order to find surplus work.

Work Sharing (Centralized Scheduling /Self Scheduling)

- Keep a queue of tasks waiting to be done
  - May be done by manager process
  - Or a shared data structure protected by locks
- When free, Worker process requests some tasks



Source: Prof. Demmel, CS267, UC Berkeley

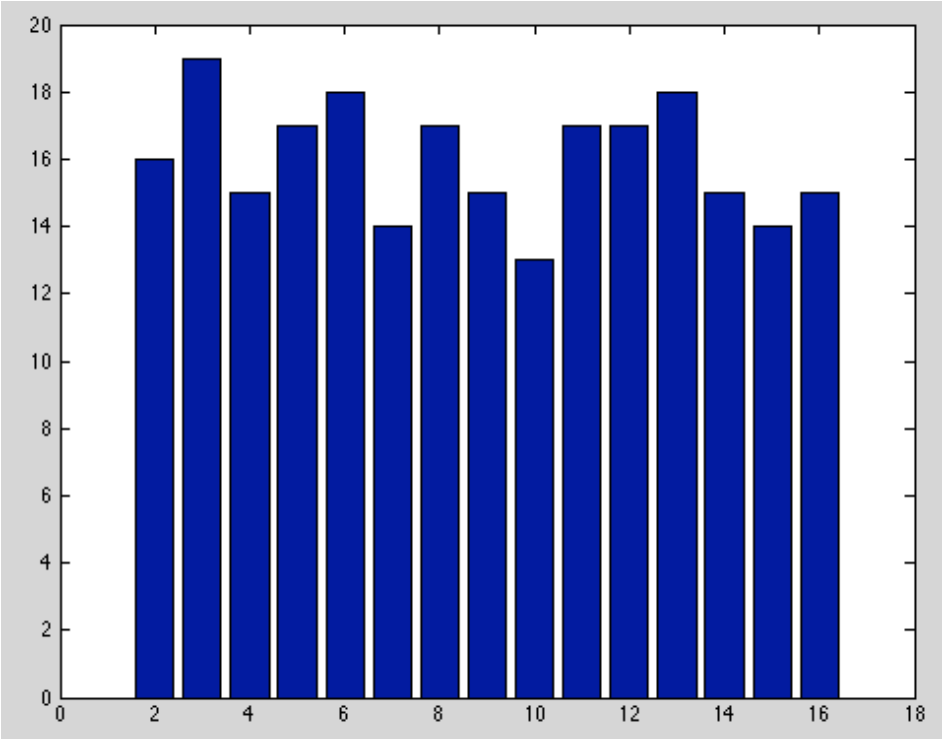
Work Sharing

MRF_MP2.tcl

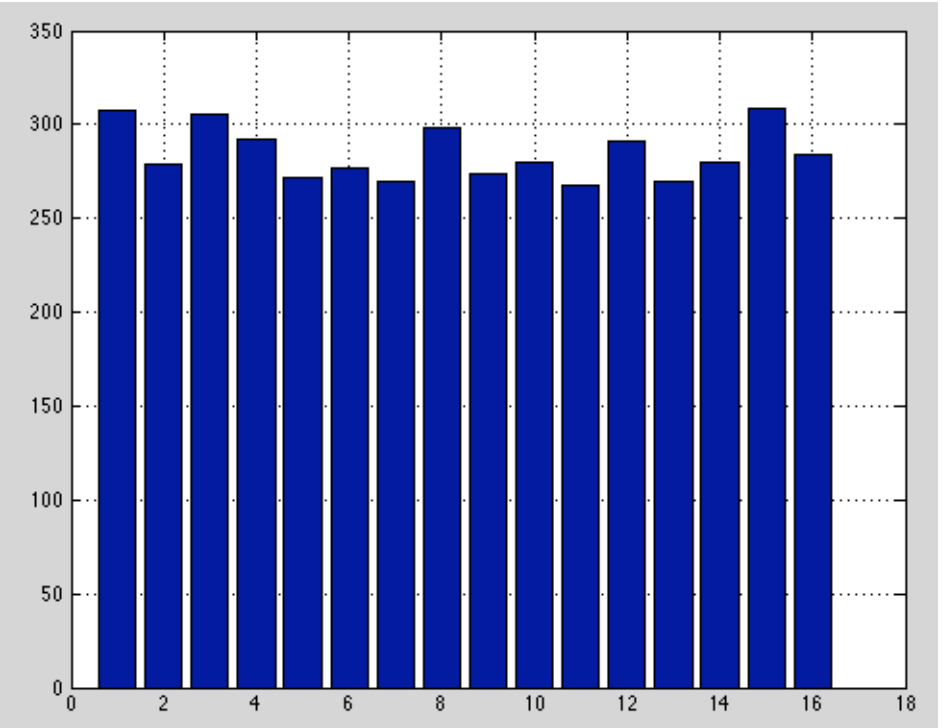
```
set np [getNP]
set pid [getPID]
# set some parameters & open some local files
….
if {$pid == 0} {
   foreach Hazard $Hazards {

      set dataDir [format "Oak-%i-50-Output" $Hazard] ;
      file mkdir $dataDir/DriftAcceleration;
      file mkdir $dataDir/ForceDeformation

      foreach eqDir $eqDirections {
         for {set i 1 } { $i <= $eqNum } { incr i} {
            recv -pid ANY pidWorker
            send -pid $pidWorker "$Hazard $eqDir $i"
         }
      }
   }
   # tell all processors we are done
   for {set i 1} {$i < $np} {incr i 1} {
      recv -pid ANY pidWorker
      send -pid $pidWorker "DONE"
   }
} else {
```

```
} else {
   set done NOT_DONE;
   while {$done != "DONE"} {
      send -pid 0 $pid
      recv -pid 0 task

      set Hazard [lindex $task 0]

      if {$Hazard == "DONE"} {
         set done "DONE"
         break;
      }
      set eqDir [lindex $task 1]
      set i    [lindex $task 2]

      set tStart [clock clicks -milliseconds]
      set GMdir [format "./GMfiles/Oak-%i-50/" $Hazard];
      source buildModelWithGravity.tcl
      source addRecorders.tcl
      source Dynamic.EQ.tcl
      source processResults.tcl

      set tEnd [clock clicks -milliseconds]
      puts $timeOut "DURATION: $i $eqDir $Hazard [expr ($tEnd-$tStart)/1000.]"
   }
}
 set tEnd [clock clicks -milliseconds]
puts $timeOut "TOTAL DURATION: [expr ($tEnd-$tStartAll)/1000.]"
puts  "PID: $pid DURATION: [expr ($tEnd-$tStartAll)/1000.]"
# close local files
….
```

Results using 16 Processors



 # Tasks Completed *(pid vs # tasks)*



Duration *(pid v time)*

Variations on Work Sharing

- Typically, don't want to grab smallest unit of parallel work, e.g., a single iteration
  - Too much contention at shared queue
- Instead, choose a chunk of tasks of size K.
  - If K is large, access overhead for task queue is small
  - If K is small, we are likely to have even finish times (load balance)
- (at least) Four Variations:
  - Use a fixed chunk size
  - Guided self-scheduling
  - Tapering
  - Weighted Factoring

Source: Prof. Demmel, CS267, UC Berkeley

Variation 1: Fixed Chunk Size

- Kruskal and Weiss give a technique for computing the optimal chunk size (IEEE Trans. Software Eng., 1985)
- Requires a lot of information about the problem characteristics
  - e.g., task costs, number of tasks, cost of scheduling
  - Probability distribution of runtime of each task (same for all)
  - Assumes distribution is IFR = "Increasing Failure Rate"
- Not very useful in practice
  - Task costs must be known at loop startup time
  - E.g., in compiler, all branches be predicted based on loop indices and used for task cost estimates

Variation 2: Guided Self-Scheduling

- Idea: use larger chunks at the beginning to avoid excessive overhead and smaller chunks near the end to even out the finish times.
  - The chunk size K_i at the i^{th} access to the task pool is given by

$$K_i = \text{ceiling}(R_i/p)$$

  - where R_i is the total number of tasks remaining and
  - p is the number of processors
- See Polychronopoulos & Kuck, "Guided Self-Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers," IEEE Transactions

Variation 3: Tapering

- Idea: the chunk size, K_i is a function of not only the remaining work, but also the task cost variance
  - variance is estimated using history information
  - high variance => small chunk size should be used
  - low variance => larger chunks OK

- See S. Lucco, "Adaptive Parallel Programs," PhD Thesis, UCB, CSD-95-864, 1994.
  - Gives analysis (based on workload distribution)
  - Also gives experimental results -- tapering always works at least as well as GSS, although difference is often small

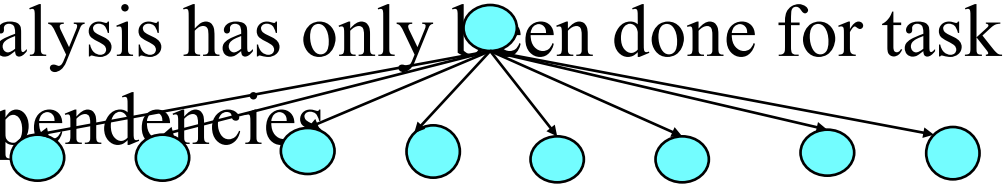
Variation 4: Weighted Factoring

- Idea: similar to self-scheduling, but divide task cost by computational power of requesting node
- Useful for heterogeneous systems
- Also useful for shared resource clusters, e.g., built using all the machines in a building
  - as with Tapering, historical information is used to predict future speed
  - "speed" may depend on the other loads currently on a given processor
- See Hummel, Schmit, Uma, and Wein, SPAA

When is Work Sharing a Good Idea?

Useful when:

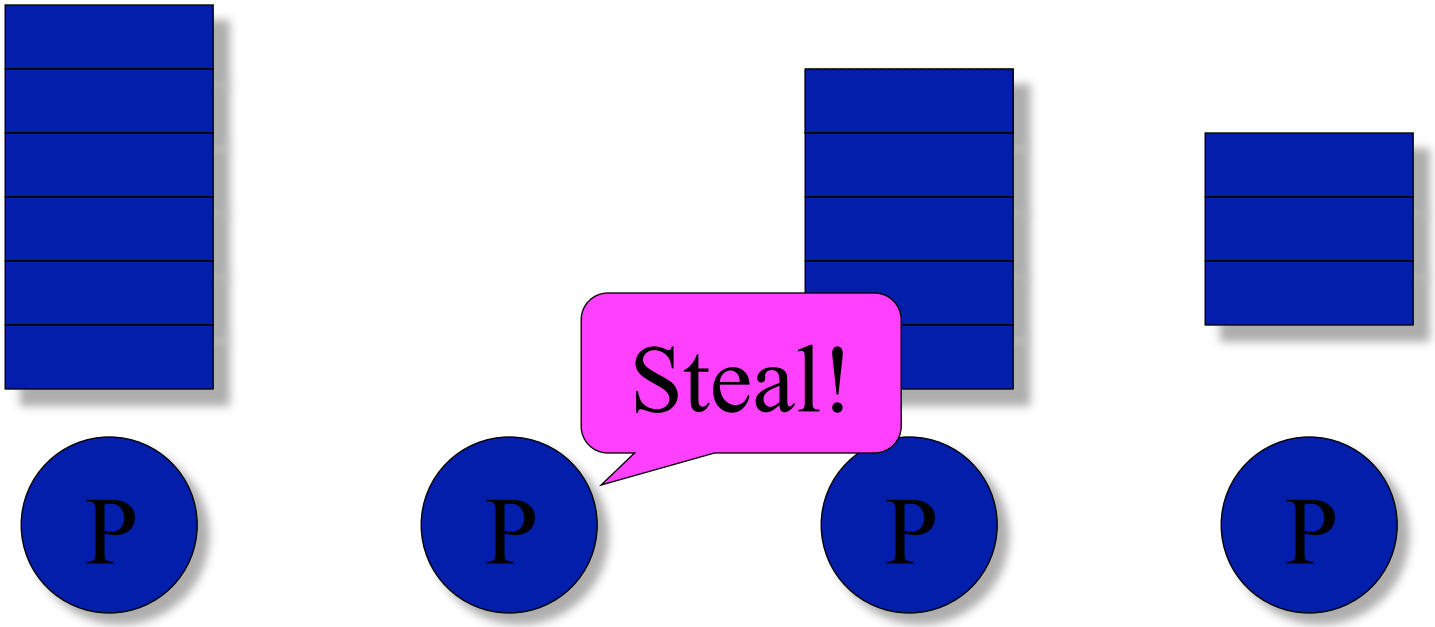
- A batch (or set) of tasks without dependencies
  - can also be used with dependencies, but most analysis has only been done for task sets without dependencies



- The cost of each task is unknown
- Locality is not important
- Shared memory machine, or at least number

Work Stealing (Work Crews/Distributed Load Balancing)

- Each processor processes tasks in its queue
- When finished, steals work from a processor that is busy
- Requires asynchronous communication



Source: Prof. Demmel, CS267, UC Berkeley

Work Stealing

MRF_MP3.tc

```
set np [getNP]
set pid [getPID]
# set some parameters & open some local files
# define some procedures for work stealing
….
if {$pid == 0} {

   # create task queues
   createTaskQueues $np

   # add tasks to the different queus
   set count 0
   foreach Hazard $Hazards {
      set dataDir [format "Oak-%i-50-Output" $Hazard] ;
      file mkdir $dataDir/DriftAcceleration;
      file mkdir $dataDir/ForceDeformation;

      foreach eqDir $eqDirections {
         for {set i 1 } { $i <= $eqNum } { incr i} {
            set thePID [expr $count % $np]
            addTask $thePID $Hazard $eqDir $i
            incr count 1
         }
      }
   }
}

barrier
```

Work Stealing

MRF_MP3.tcl

```
#
# process my own tasks
#

set done NOT_DONE;
while {$done != "DONE"} {
   set task [getTask $pid]

   if {[llength $task] == 0} {
      set done "DONE"
      break;
   }

   set Hazard [lindex $task 0]
   set eqDir  [lindex $task 1]
   set i      [lindex $task 2]

   set tStart [clock clicks -milliseconds]

   set GMdir [format "./GMfiles/Oak-%i-50/" $Hazard];
   source buildModelWithGravity.tcl
   source addRecorders.tcl
   source Dynamic.EQ.tcl
   source processResults.tcl

   set tEnd [clock clicks -milliseconds]
    puts $timeOut "DURATION: $i $eqDir $Hazard [expr ($tEnd-$tStart)/1000.]"
                                        }
```

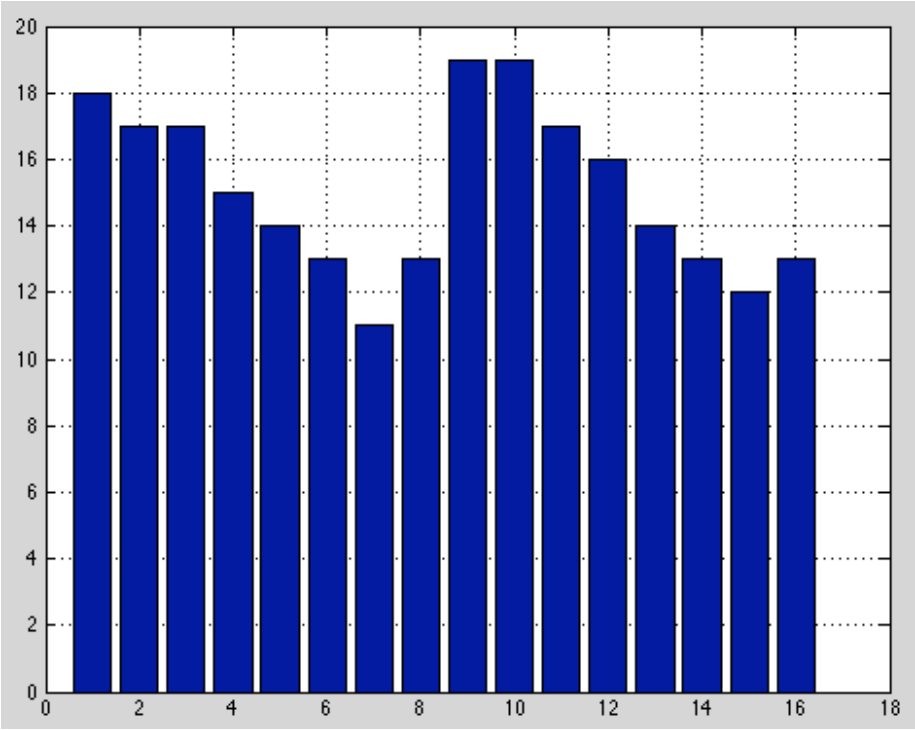
```
#
# done with my own queue, start stealing from others
#

set numDone 1;
set done “NOTDONE”
while {$done == “NOTDONE”} {

  set task [stealTask]

  if {$task == ""} {
     set done “DONE”
     break
  }

  set Hazard [lindex $task 0]
  set eqDir  [lindex $task 1]
  set i      [lindex $task 2]

  set tStart [clock clicks -milliseconds]

  set GMdir [format "./GMfiles/Oak-%i-50/" $Hazard];
  source buildModelWithGravity.tcl
  source addRecorders.tcl
  source Dynamic.EQ.tcl
  source processResults.tcl

  set tEnd [clock clicks –milliseconds]
  puts $timeOut "DURATION: $i $eqDir $Hazard [expr ($tEnd-$tStart)/1000.]"
                                        }
```

Work Stealing

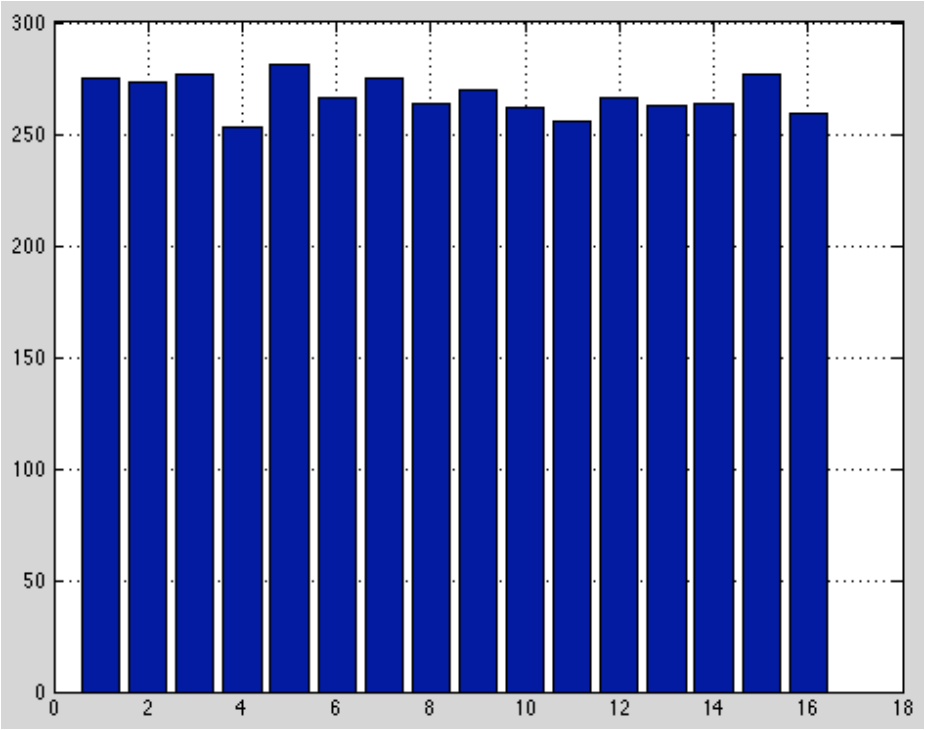
```
proc createTaskQueues {np} {
   # open some files to store tasks
   for {set i 0} {$i < $np} {incr i 1} {
                set taskFile [open tasks.$i w]
   }
   close $taskFile
}

proc addTask {thePID Hazard eqDir i} {
   set taskFile [open tasks.$thePID a]
   puts $taskFile "$Hazard $eqDir $i"
   close $taskFile
}

proc getTask {thePID} {
   if {[file size tasks.$thePID] == 0} {
     return ""
   }
   set taskFile [open tasks.$thePID r+]
   seek $taskFile 0
   gets $taskFile task
   set restData [read $taskFile]
   chan truncate $taskFile 0
   seek $taskFile 0
   set data [split $restData "\n"]
   foreach line $data {
      if {[llength $line] != 0} {
         puts $taskFile $line
      }
   }
   close $taskFile
   return $task
```

```
proc getRandomPID {np} {
   if {$np == 0} {
     return 0;
   }
   set maxFactor [expr $np + 1]
   set value [expr int([expr rand() * 100])]
   set value [expr int([expr $value % $maxFactor])]
   return [expr $value % $np]
}
proc stealTask {} {
   global pid
   global np
   global numDone
   global processesDone
   set foundOne 0
   set task ””;
   while {$foundOne == 0 && $numDone < $np} {
    set otherProcess [getRandomPID [expr $np-$numDone-1]]
    set task [getTask [lindex $processesDone $otherProcess]]
    if {[llength $task] == 0} {
      set newProcessesDone {}
      for {set i 0} {$i < $np-$numDone} {incr i 1} {
        if {$i != $otherProcess} {
          lappend newProcessesDone [lindex$processesDone $i]
        }
       }
       set numDone [expr $numDone+1]
       set processesDone $newProcessesDone
    } else {
       set foundOne 1
      }
   }
   return $task
}
```

Results using 16 Processors



 # Tasks Completed *(pid vs # tasks)*



Duration *(pid v time)*

How to Select a Donor Processor

Three basic techniques:

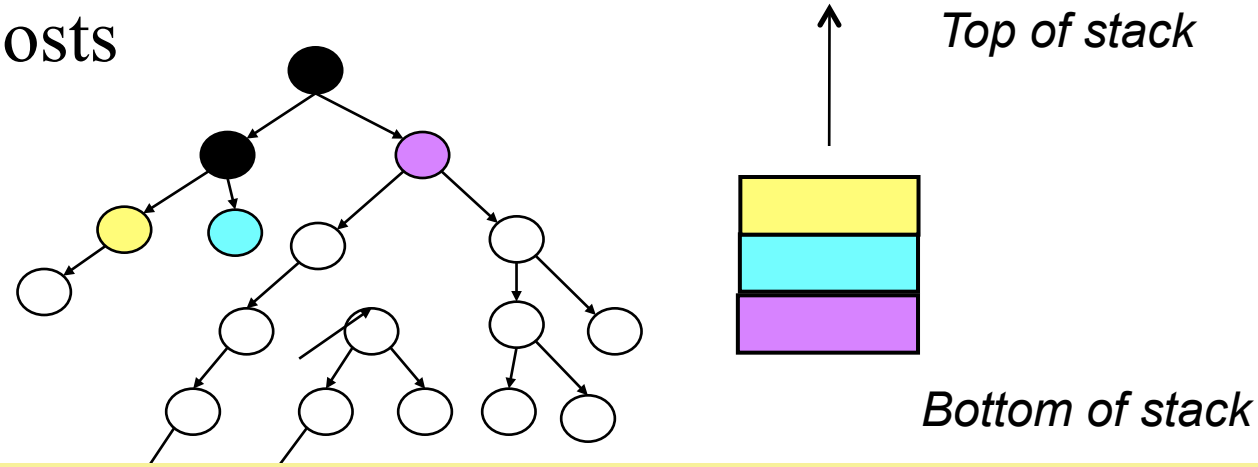
1. Asynchronous round robin
   - Each processor k, keeps a variable "$target_k$"
   - When a processor runs out of work, requests work from $target_k$
   - Set $target_k = (target_k + 1) \bmod procs$
2. Global round robin
   - Proc 0 keeps a single variable "target"
   - When a processor needs work, gets target, requests work from target
   - Proc 0 sets target = (target + 1) mod procs
3. Random polling/stealing
   - When a processor needs work, select a random processor and request work from it

- Repeat if no work is found

Source: Prof. Demmel, CS267, UC Berkeley

How to Split Work

- First question is number of tasks to split
  - Related to the work sharing variations, but total number of tasks is now unknown
- Second question is which one(s)
  - Send tasks near the bottom of the stack (oldest)
  - Execute from the top (most recent)
  - May be able to do better with information about task costs

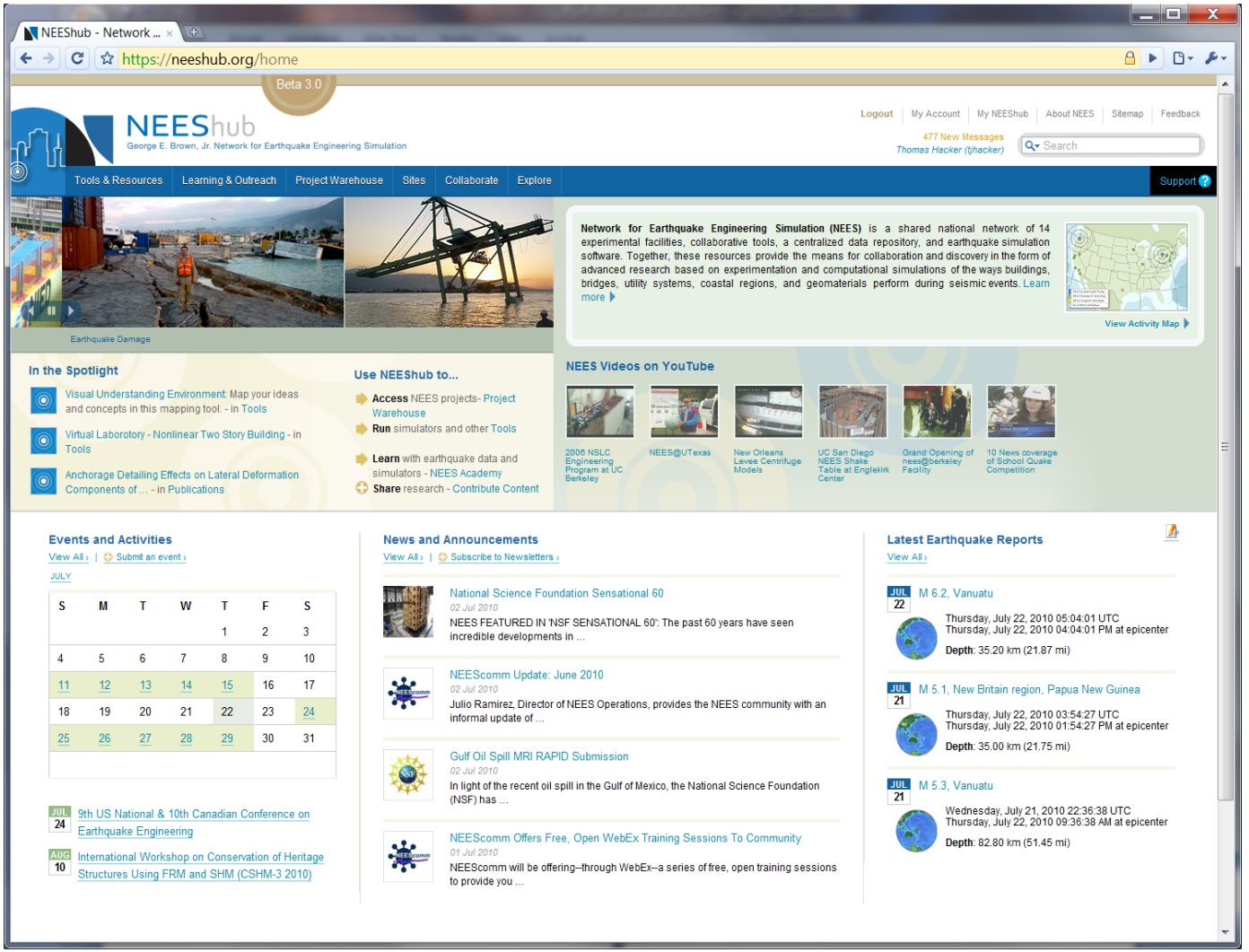


Source: Prof. Demmel, CS267, UC Berkeley

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

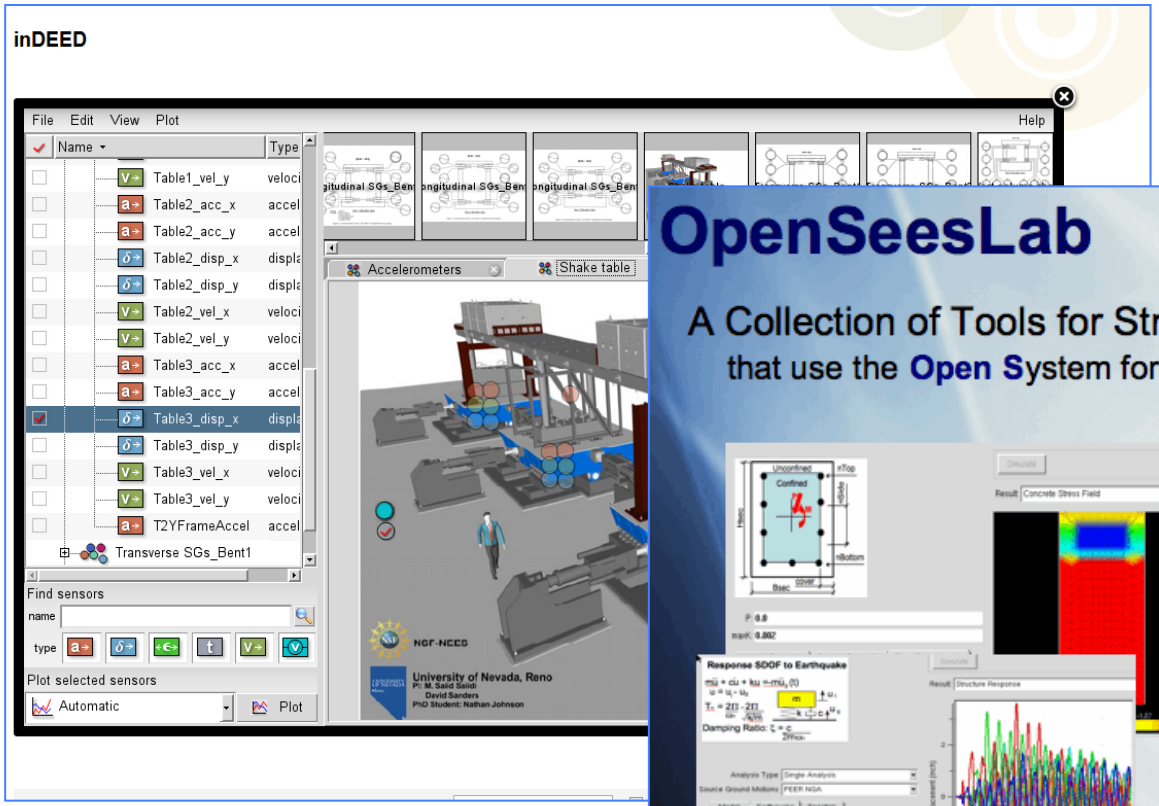
NEEShub (First Release July 2010)



* The power behind NEES

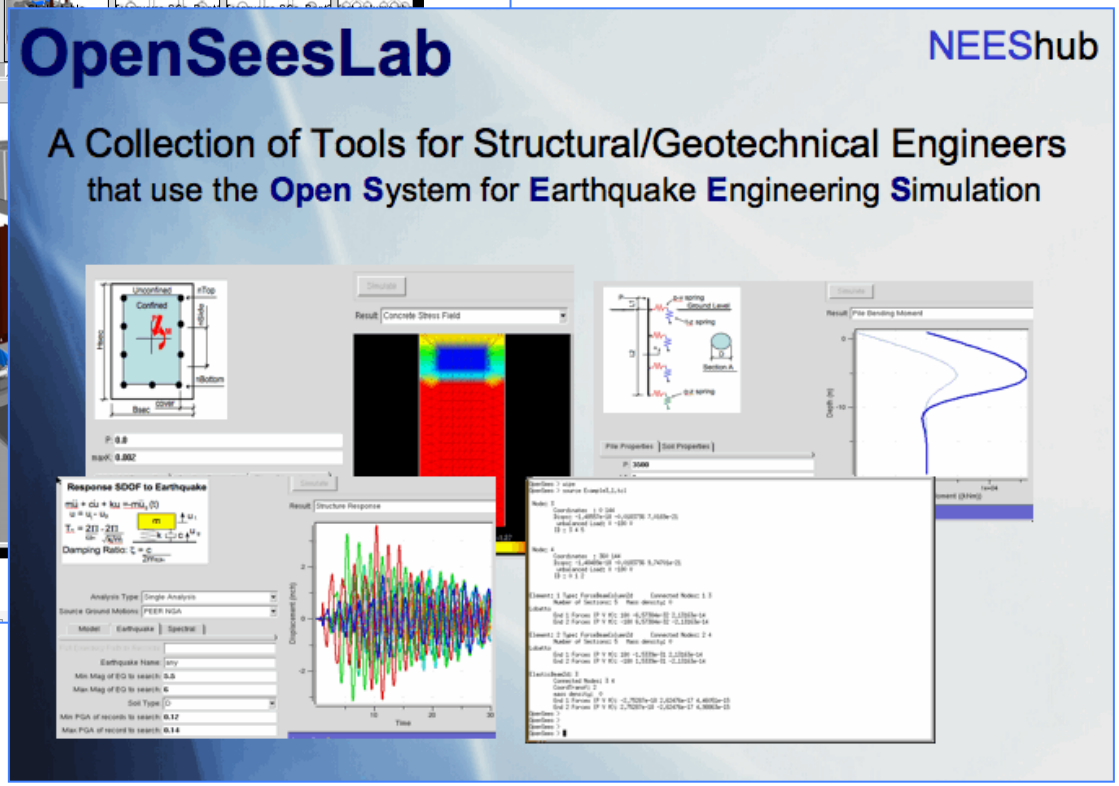
NEEShub Tools and Resources

Documents, Learning Objects, Series & TOOLS



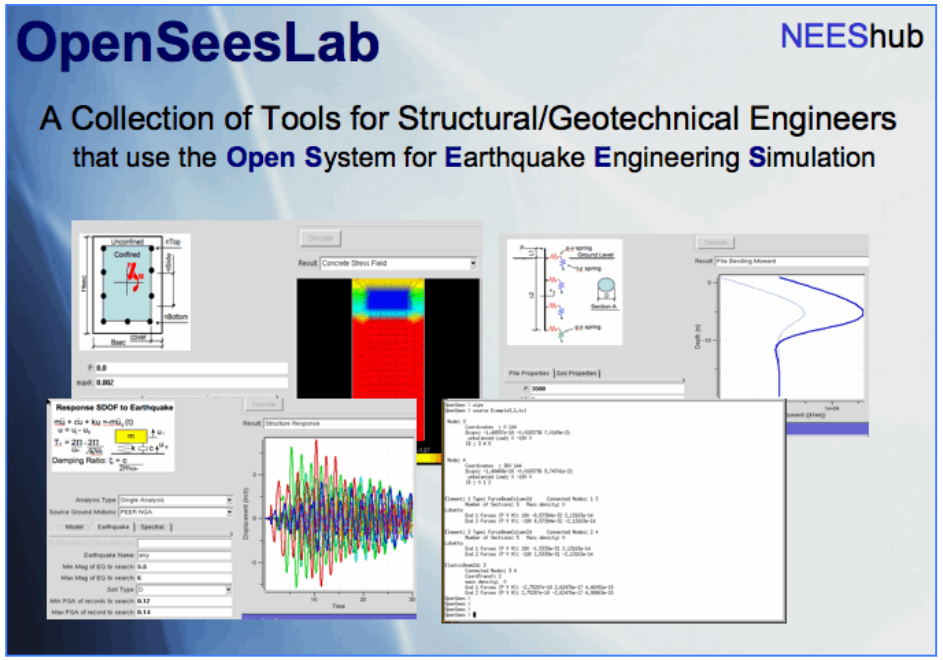
Data Management

Simulation



The OpenSeesLab tool:

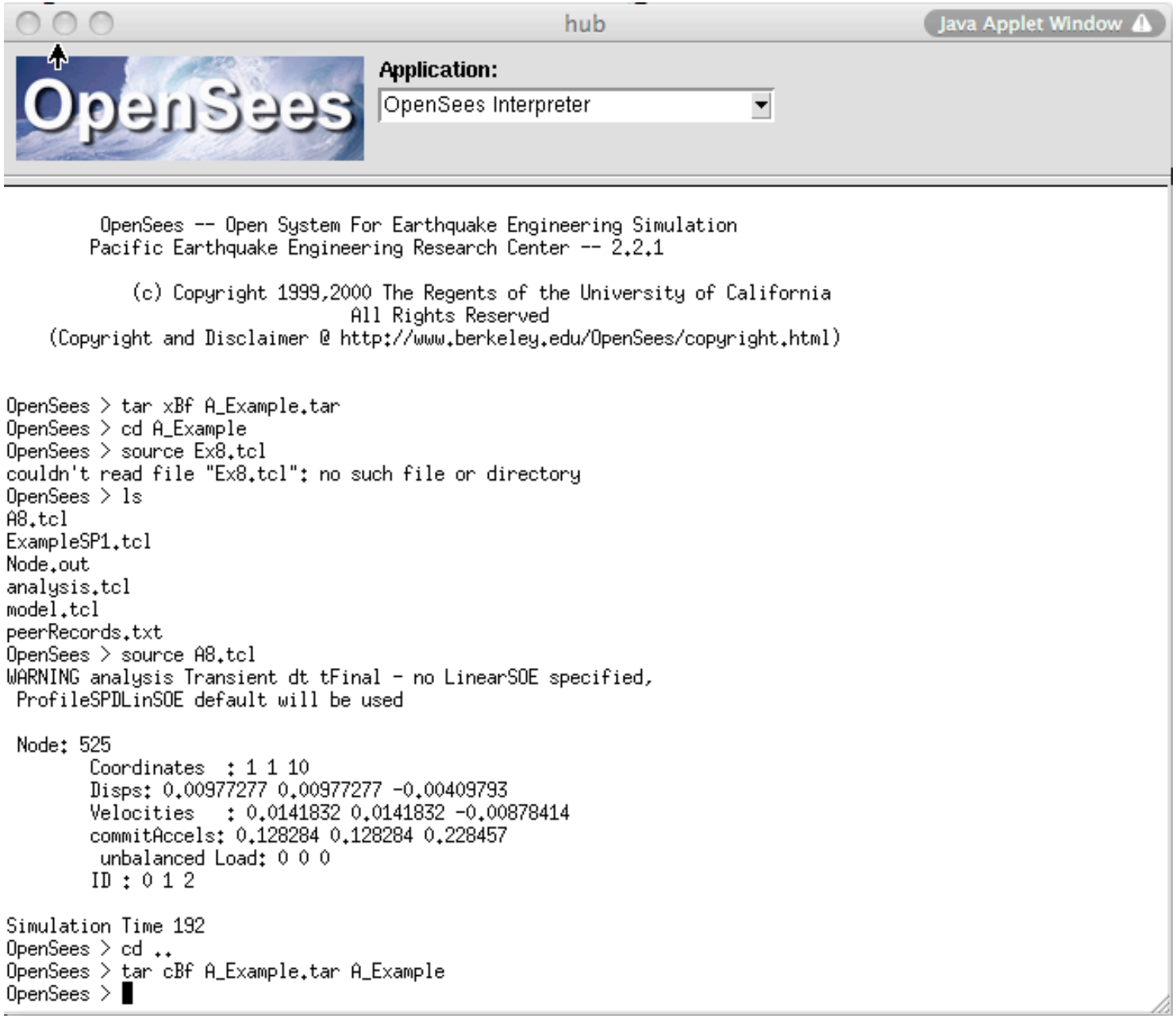
http://nees.org/resources/tools/openseeslab



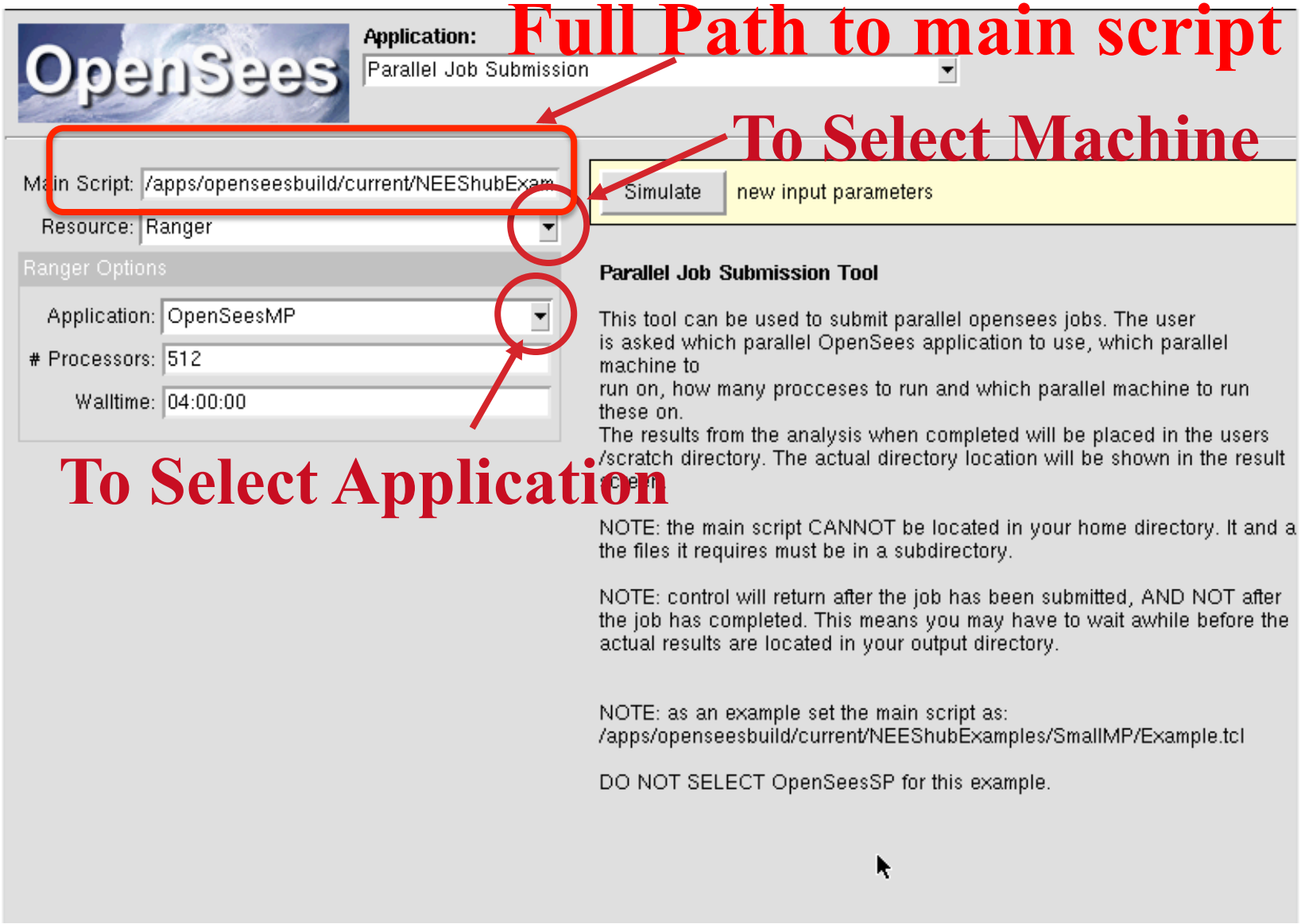
Is a suite of Simulation Tools powered by OpnSees for:

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

OpenSees Interpreter Tool

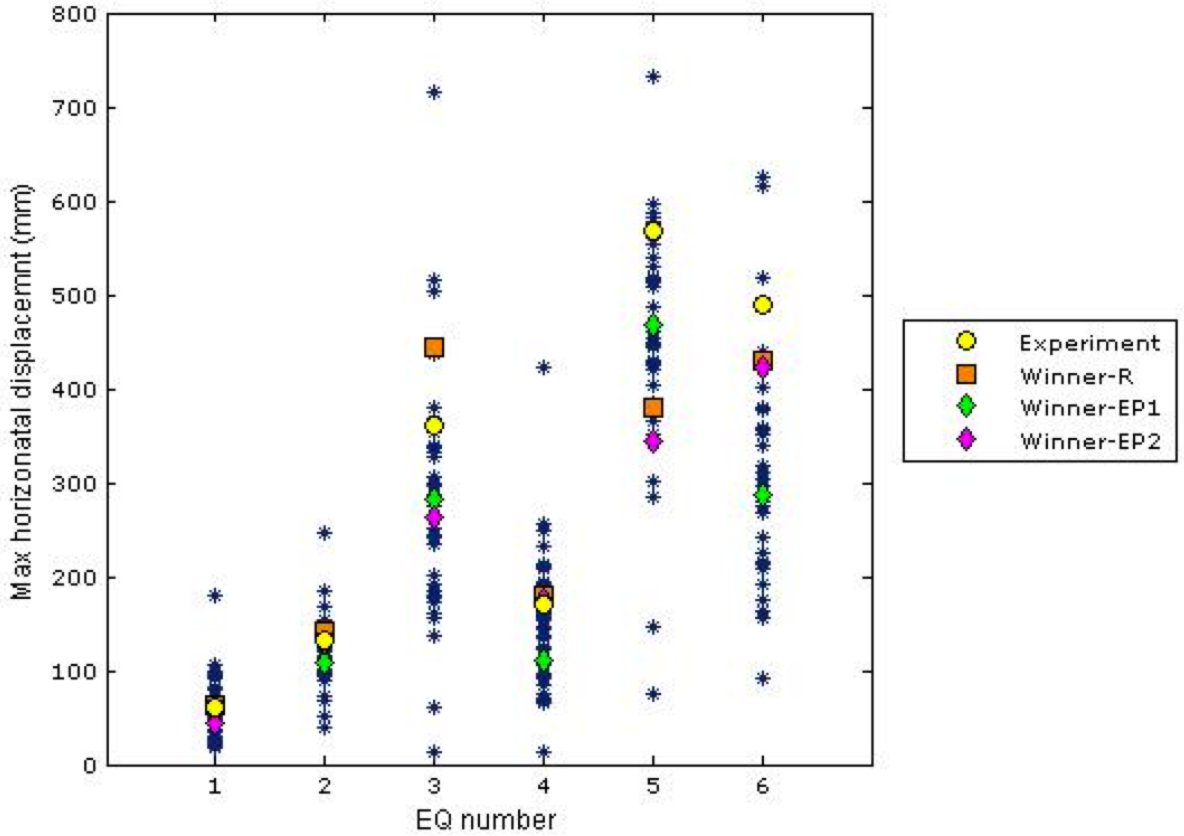


Parallel Script Submission Tool



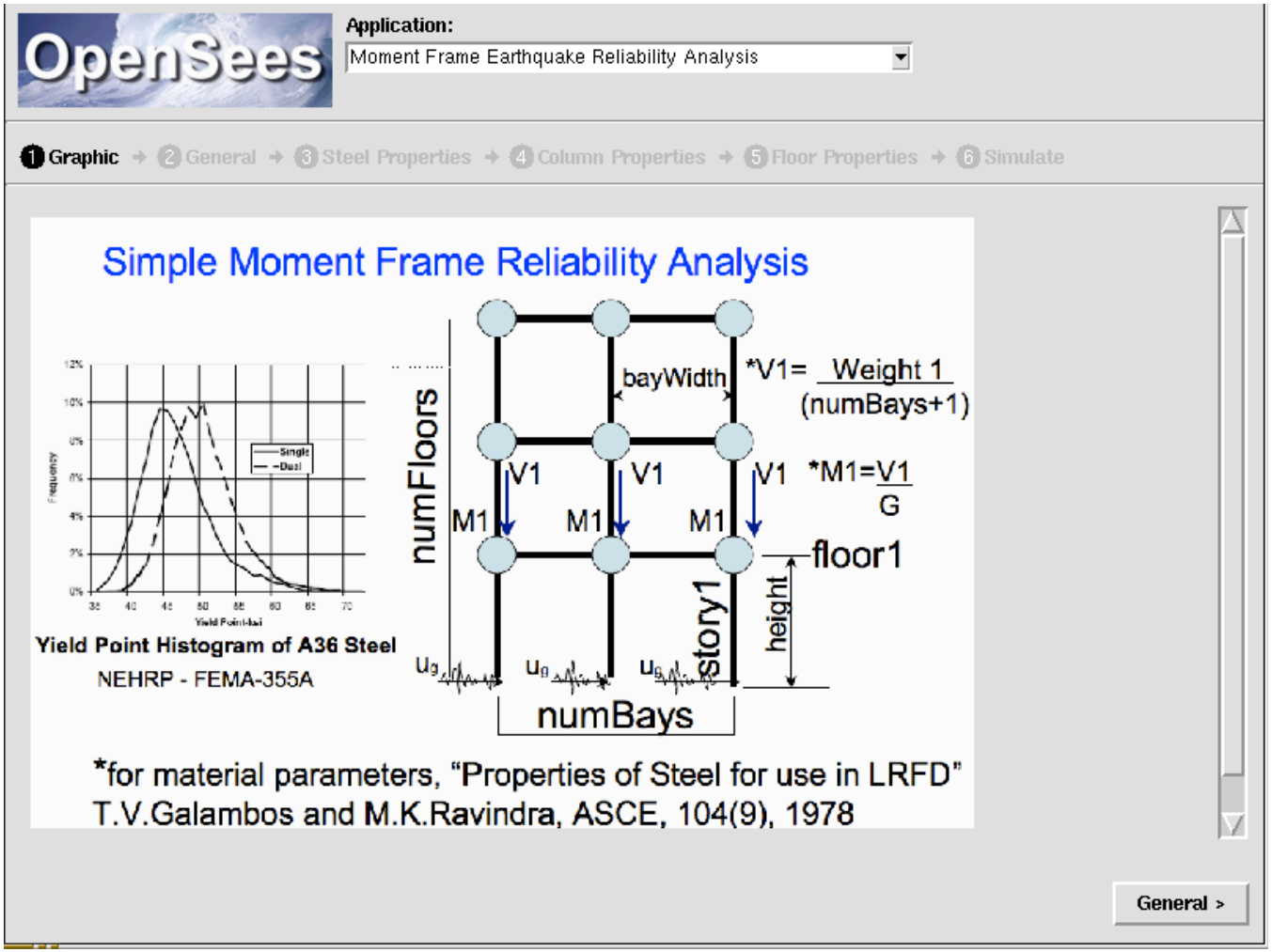
STRUCTURAL & Modeling Uncertainties

* 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 elmenet 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 containg at least one earthquake motion. These files must be 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 compilcated than this.

#Simulations: **10**

Model Properties

num Floors: **3**

num Bays: **5**

Full Directory Path to Records: /apps/openseeslab/dev/examples/records

Units: kip-in

« Graphic

Steel Properties »

OpenSees

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)

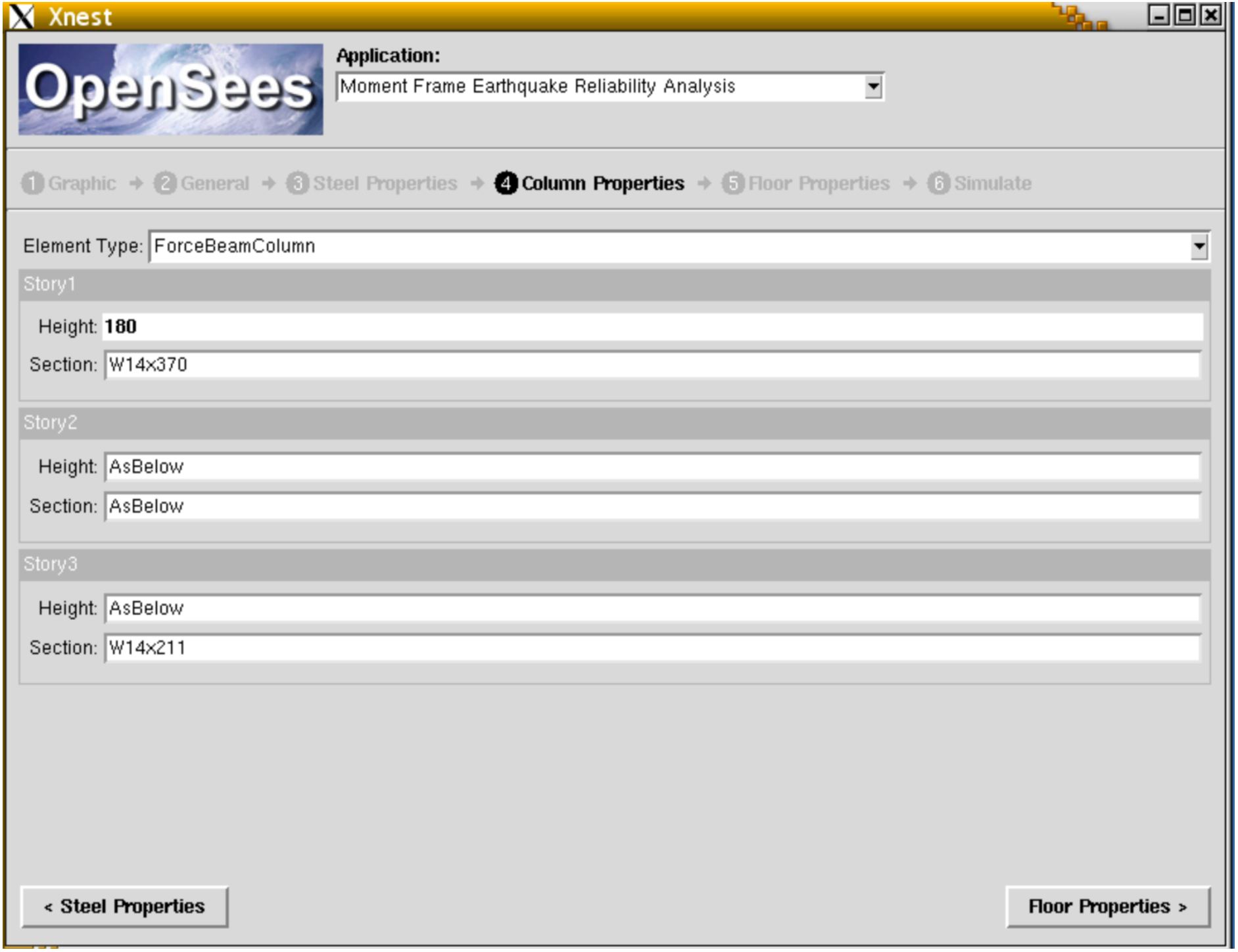
Mean: **50**

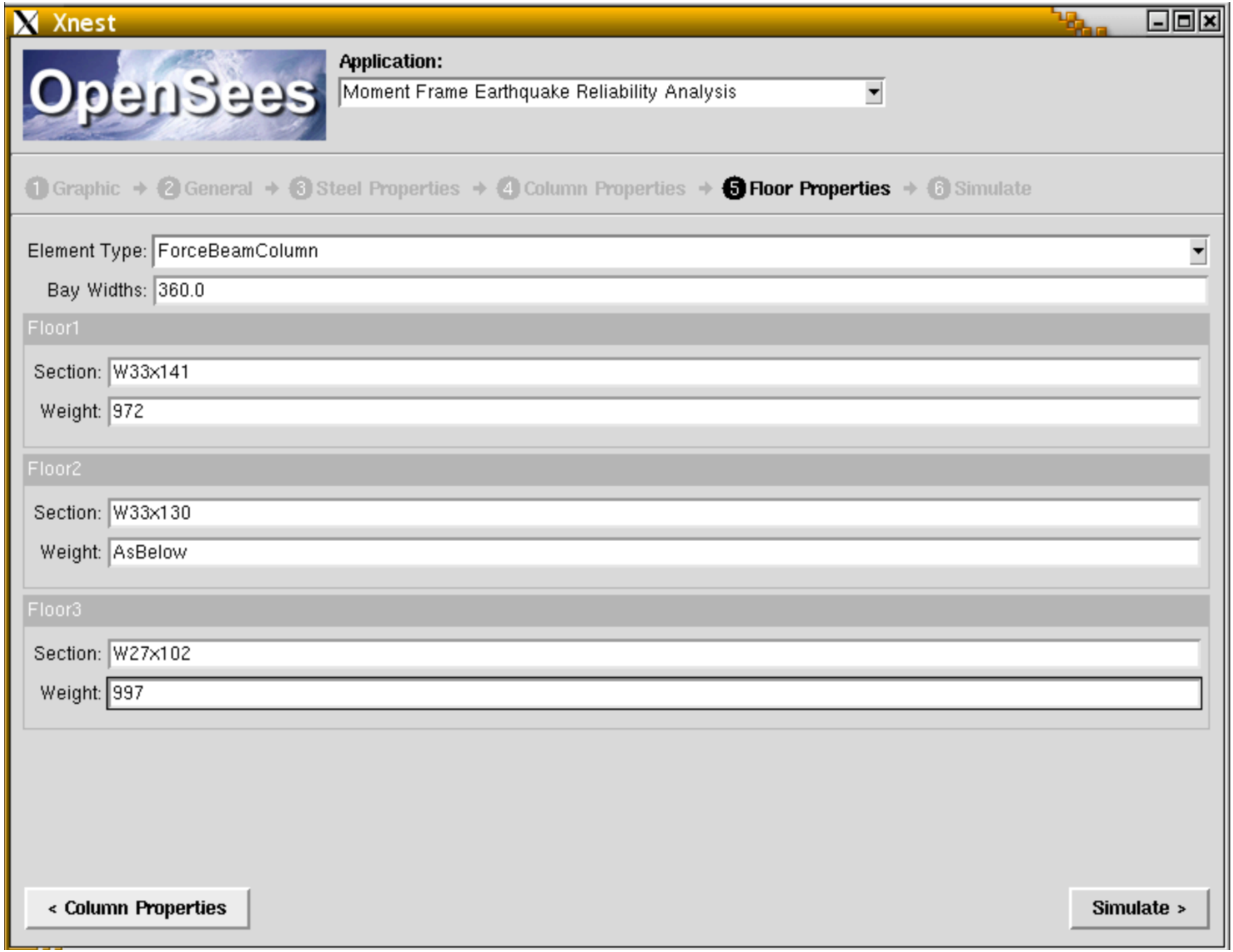
CV%: **10**

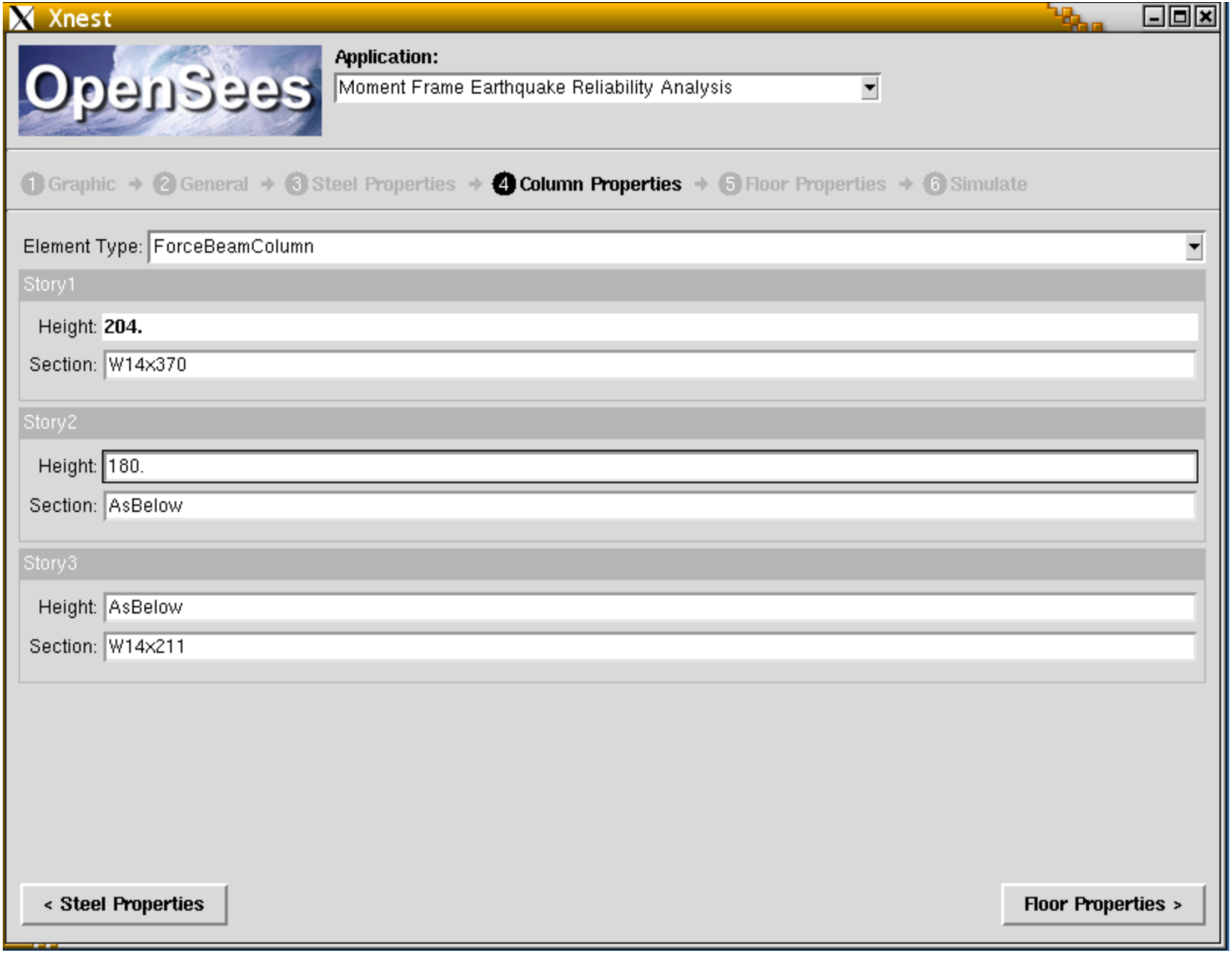
Distribution: LogNormal

« General

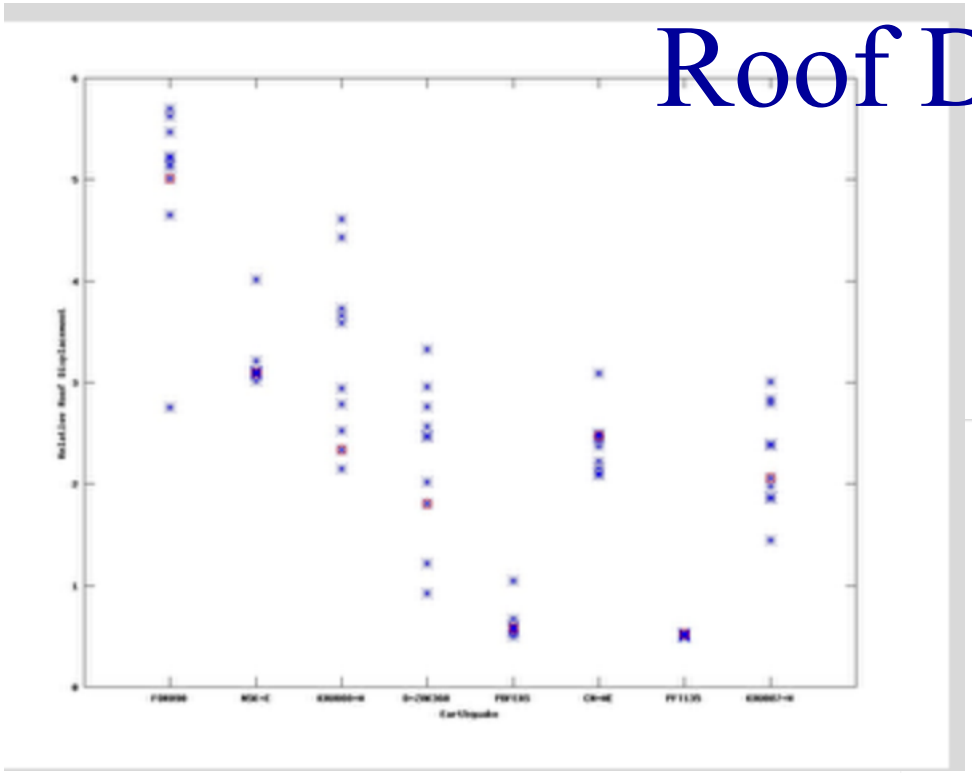
Column Properties »

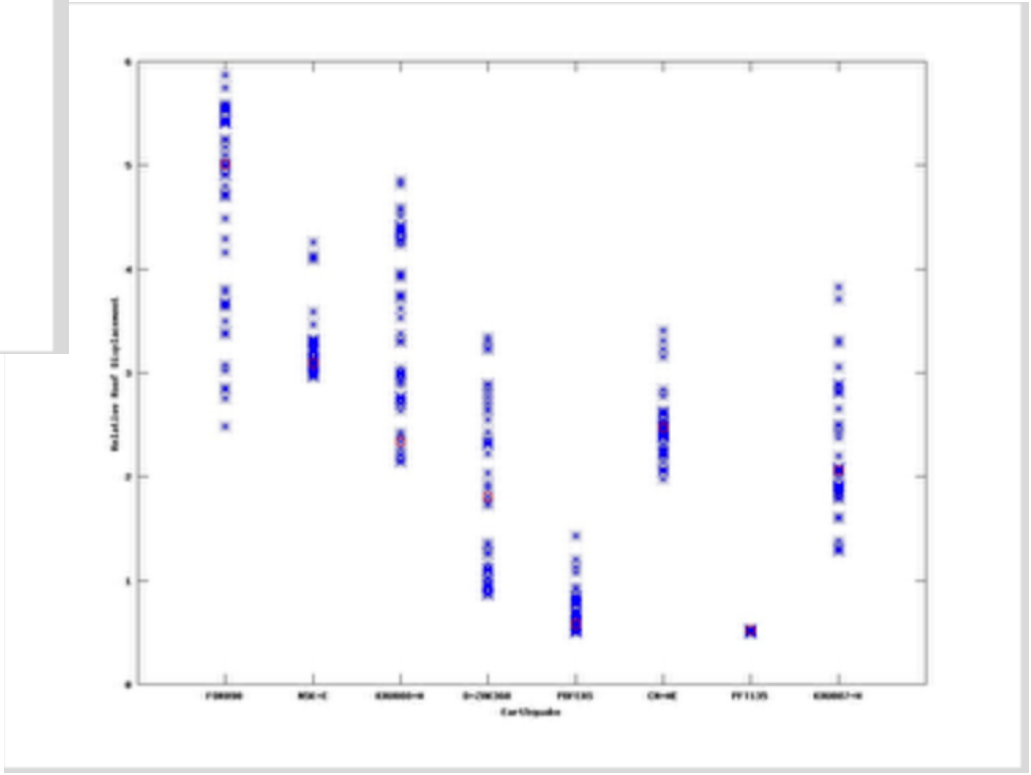

Xnest
OpenSees
Application:
Moment Frame Earthquake Reliability Analysis
1 Graphic → 2 General → 3 Steel Properties → 4 Column Properties → 5 Floor Properties → 6 Simulate
Element Type: ForceBeamColumn
Story1
Height: 180
Section: W14x370
Story2
Height: AsBelow
Section: AsBelow
Story3
Height: AsBelow
Section: W14x211
< Steel Properties
Floor Properties >


Xnest
OpenSees
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 >

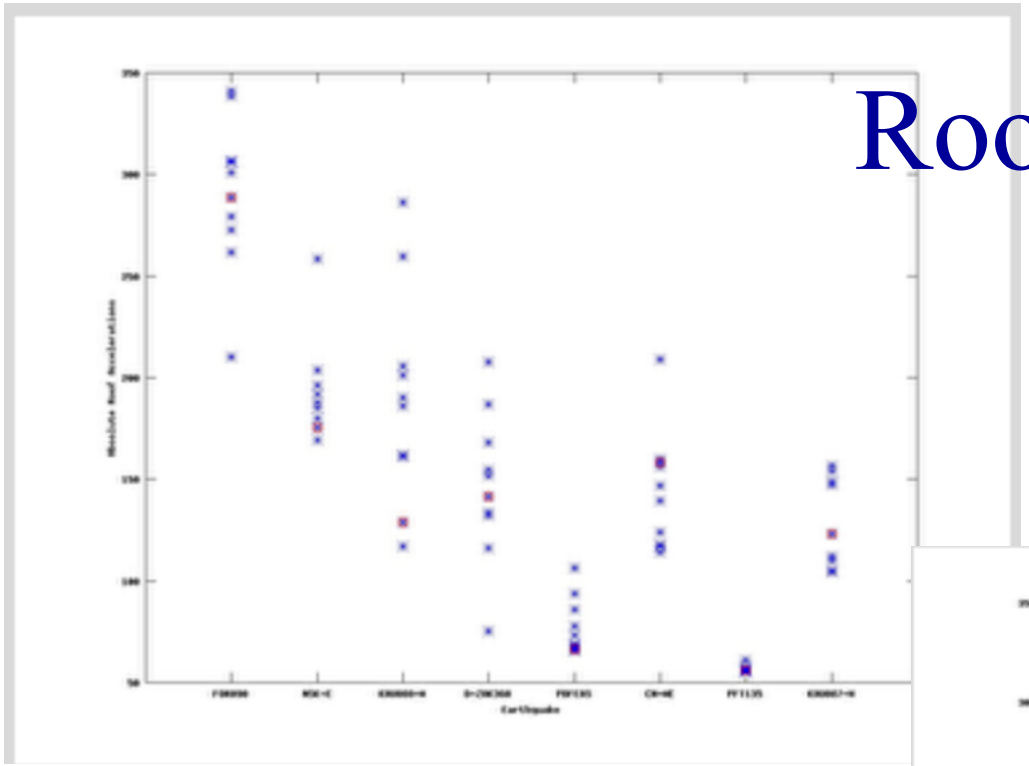

Xnest
OpenSees
Application:
Moment Frame Earthquake Reliability Analysis
1 Graphic → 2 General → 3 Steel Properties → 4 Column Properties → 5 Floor Properties → 6 Simulate
Element Type: ForceBeamColumn
Story1
Height: 204.
Section: W14x370
Story2
Height: 180.
Section: AsBelow
Story3
Height: AsBelow
Section: W14x211
< Steel Properties
Floor Properties >

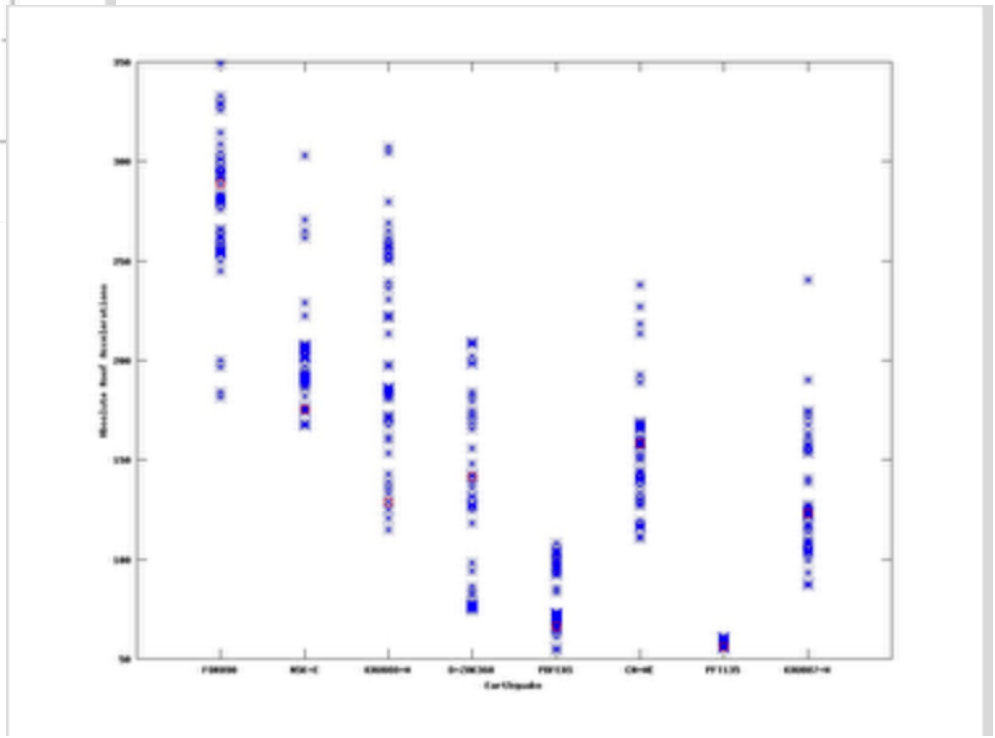
Roof Displacements





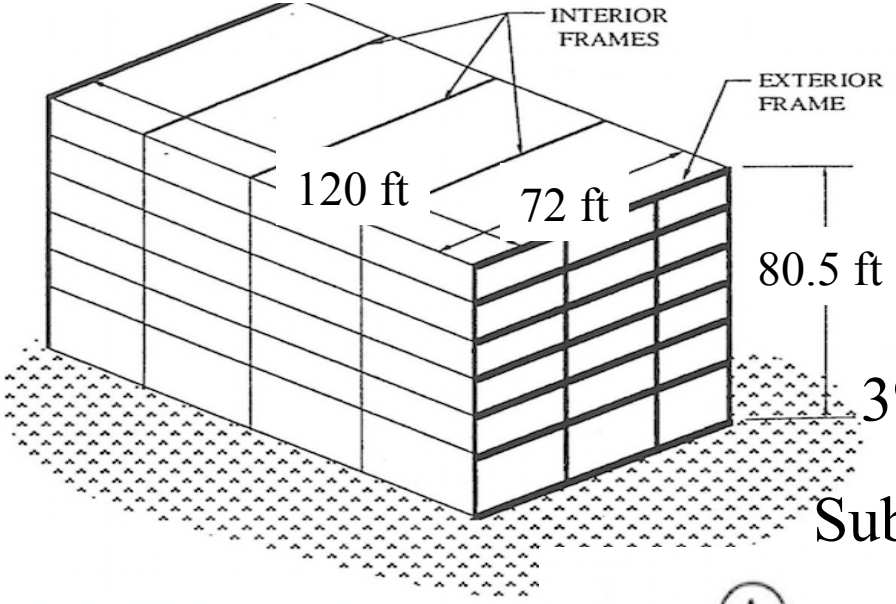
Roof Accelerations





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 = ?

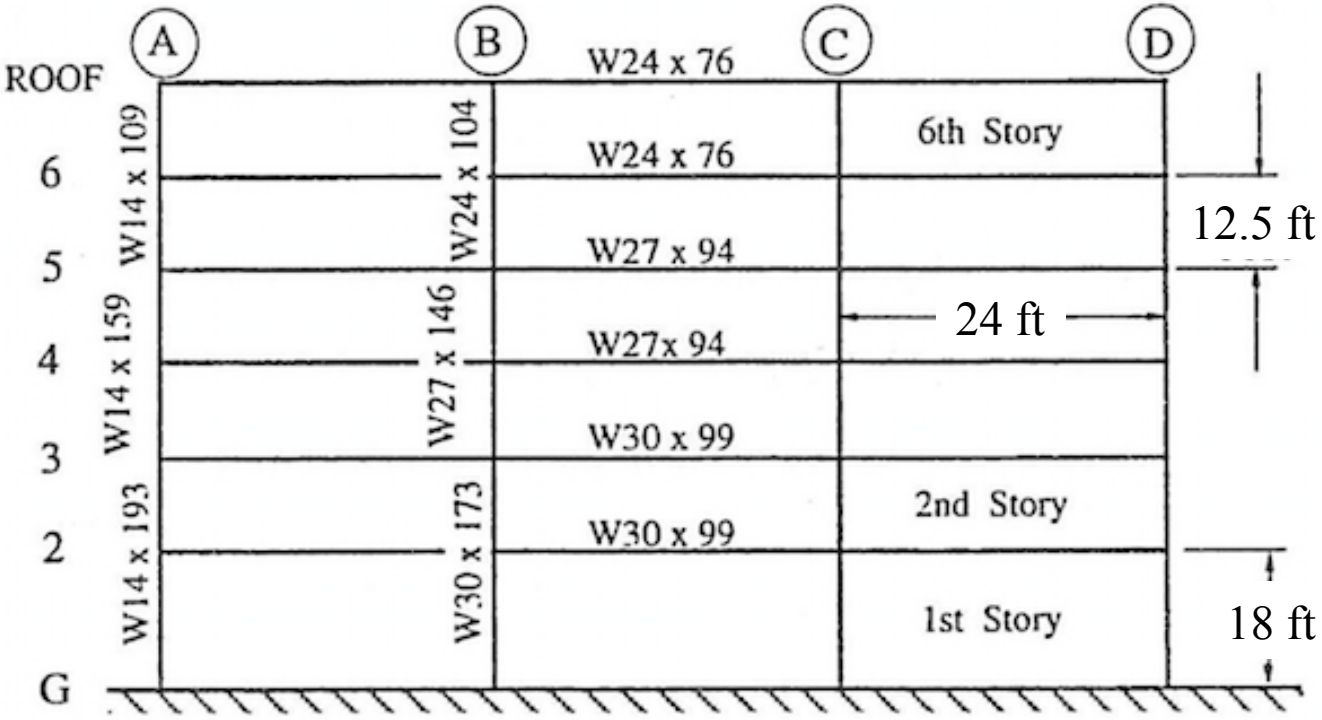
Dead Load: 95psf typical, roof 80psf

E=29,000, Fy=50.0, b =0.003

3% Rayleigh Damping 1st and 3rd Modes

Subjected to el_centro 1940 N-S (Peknold)

	A	Ixx
W14X193	56.8	2400
W14X159	46.7	1900
W14X109	32.0	1240
W30X173	51.0	8230
W27X146	43.1	5660
W24X104	30.6	3100
W30X99	29.1	3990
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

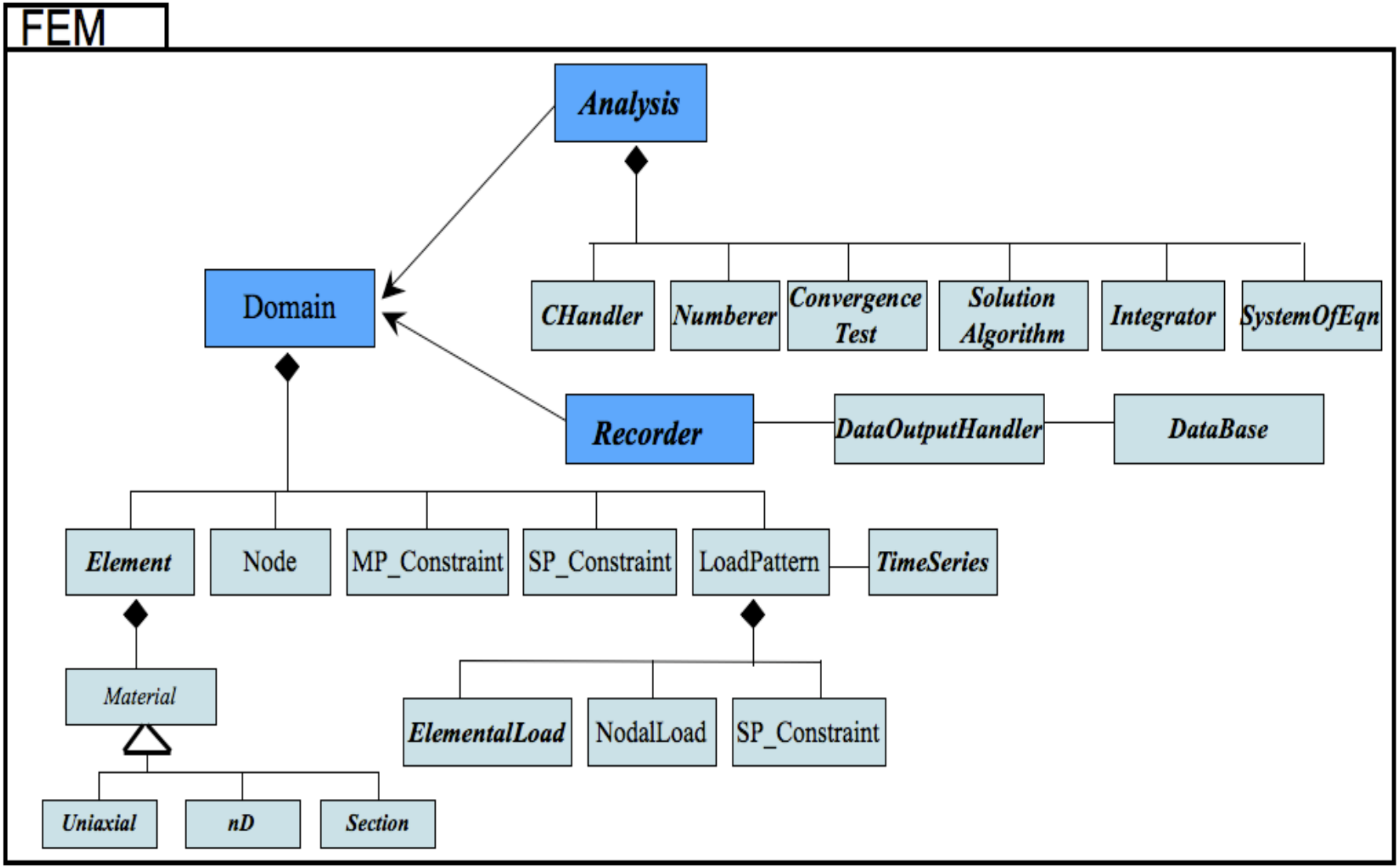
- Introduction to OpenSees Framework
- OpenSees & Tcl Interpreters
- OpenSees & Output
- Modeling in OpenSees
- Nonlinear Analysis in OpenSees
- Basic Examples
- Parallel & Distributed Processing
- OpenSees on NEEShub
- Hands on Exercise
- Adding Your Code to OpenSees
- Advanced Model Development
- Conclusion

Traditional Finite Element Software

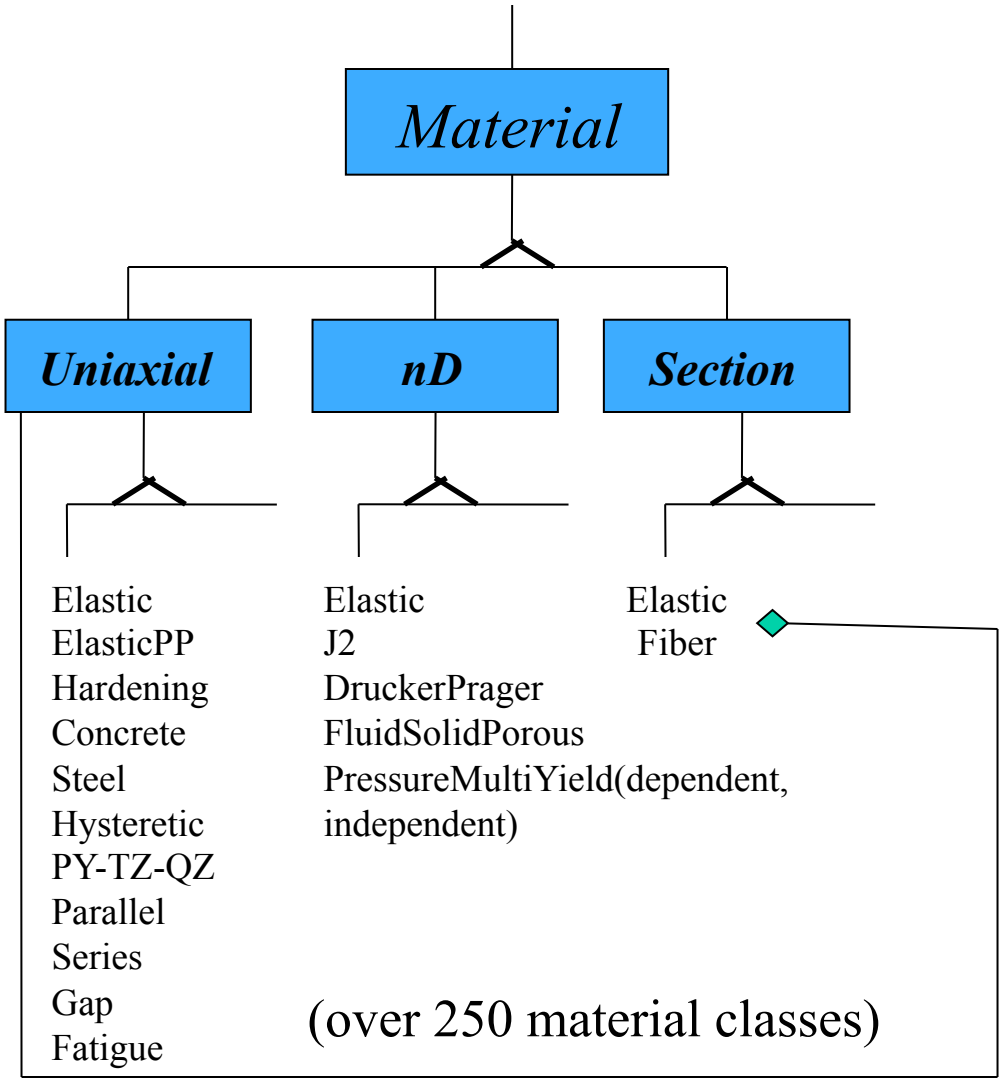
- Commercial fe codes **are large complex** software containing over one million lines of code. They **are closed-source** but **do allow new element and material routines to be added**. (at least the serious ones do)
- They are **slow to change** as hardware changes and they do **not allow researchers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

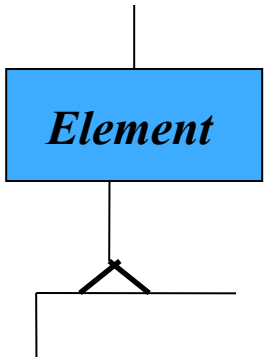
THE **ADVANTAGE** OF A **SOFTWARE FRAMEWORK** SUCH AS OPENSEES IS THAT **YOU DON'T HAVE TO UNDERSTAND ALL OF IT** TO BUILD APPLICATIONS OR MAKE CONTRIBUTIONS **YOU JUST NEED TO UNDERSTAND THAT PART THAT CONCERNS YOU**

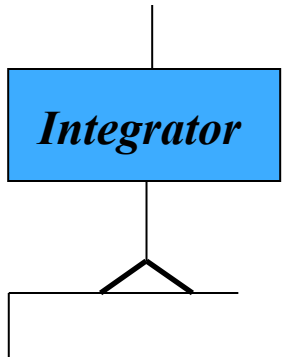
OPENSEES FEM ABSTRACT CLASSES



Framework Contains both Abstract Classes & Concrete Subclasses

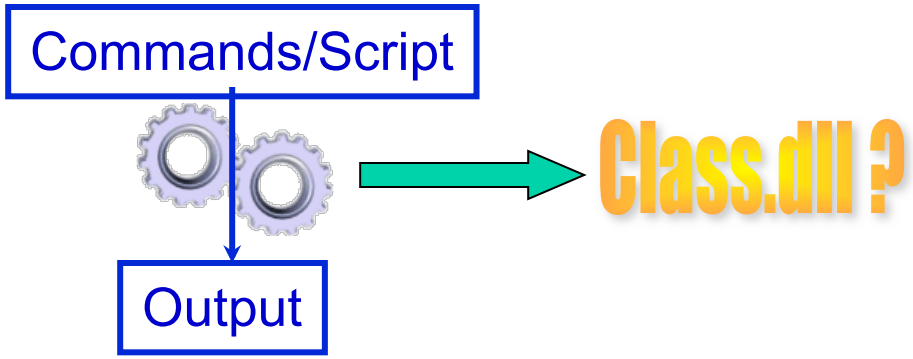


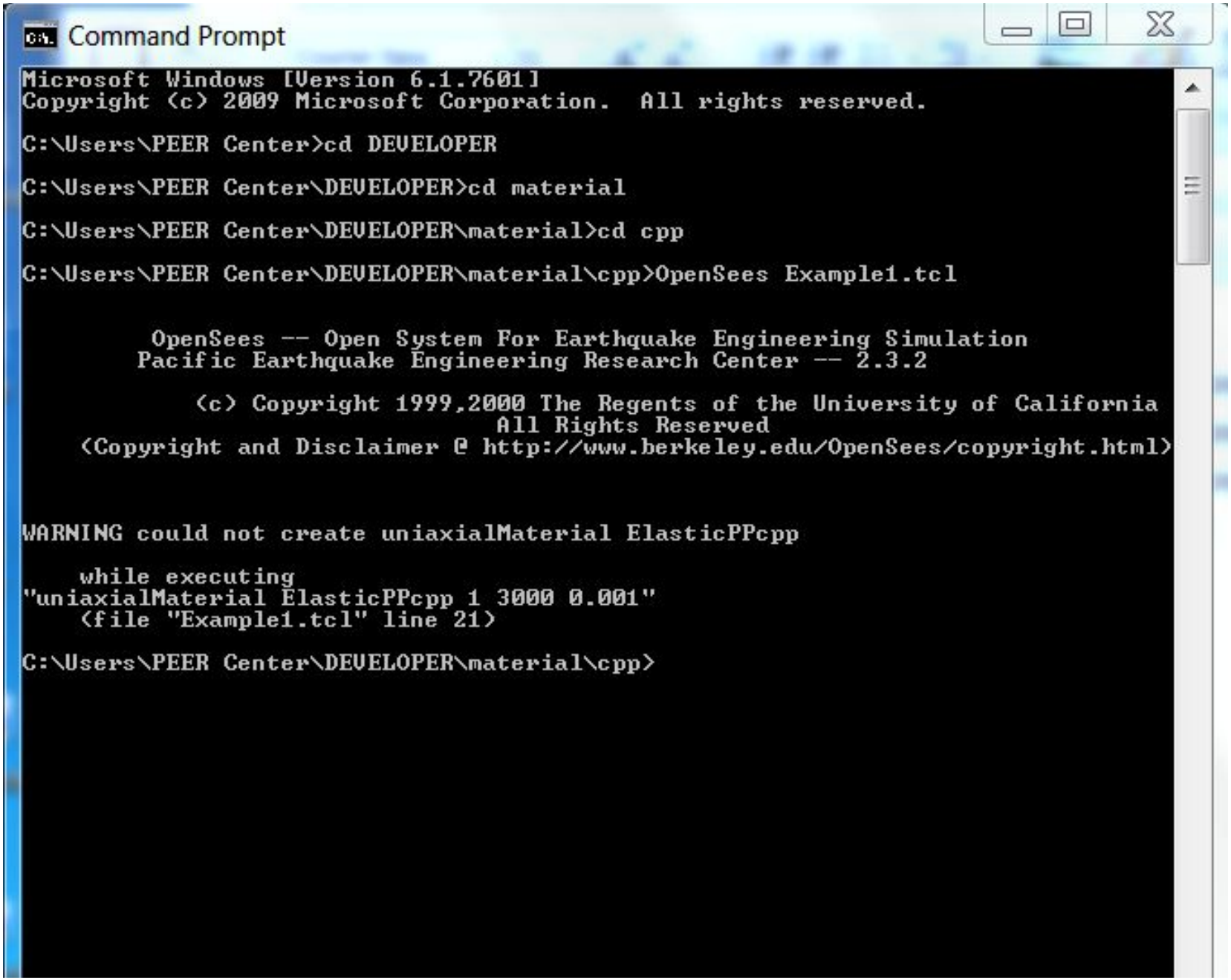




Unknown Class Type

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

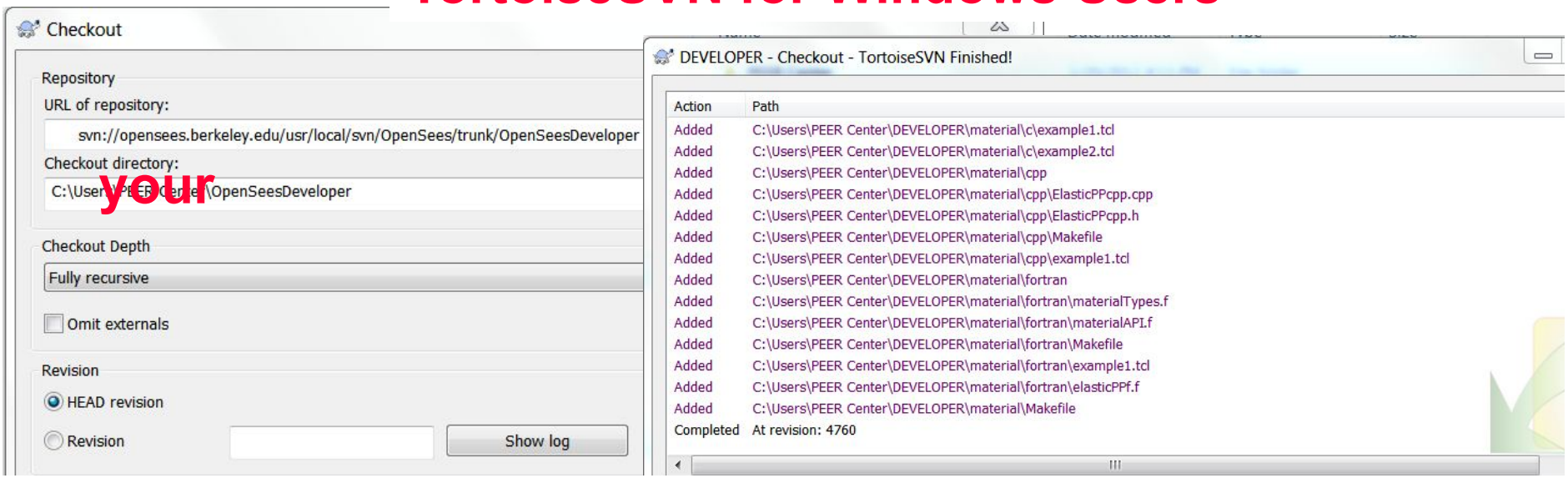
Adding New Code

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

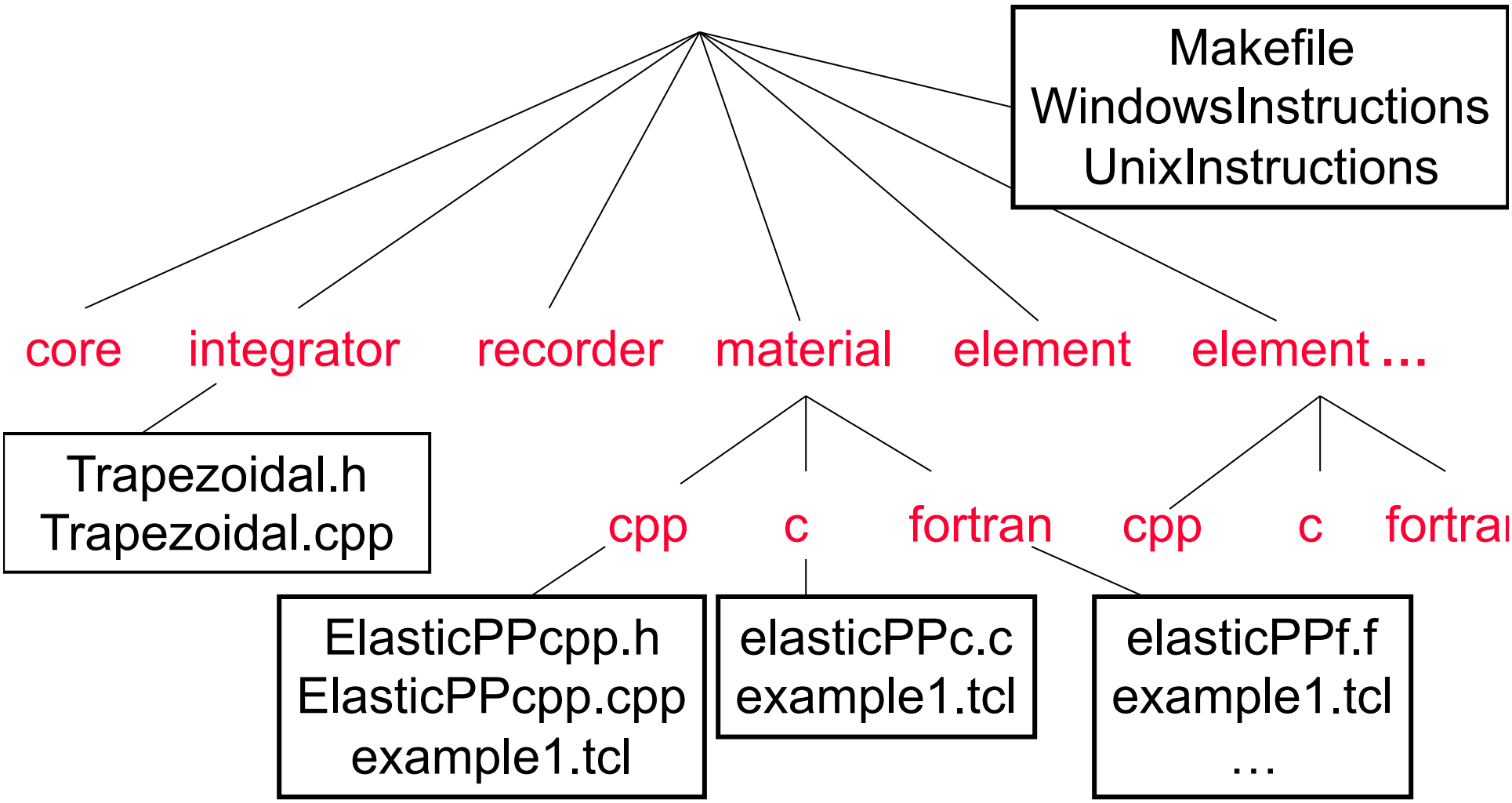
We provide C++, C and Fortran examples on-line using svn

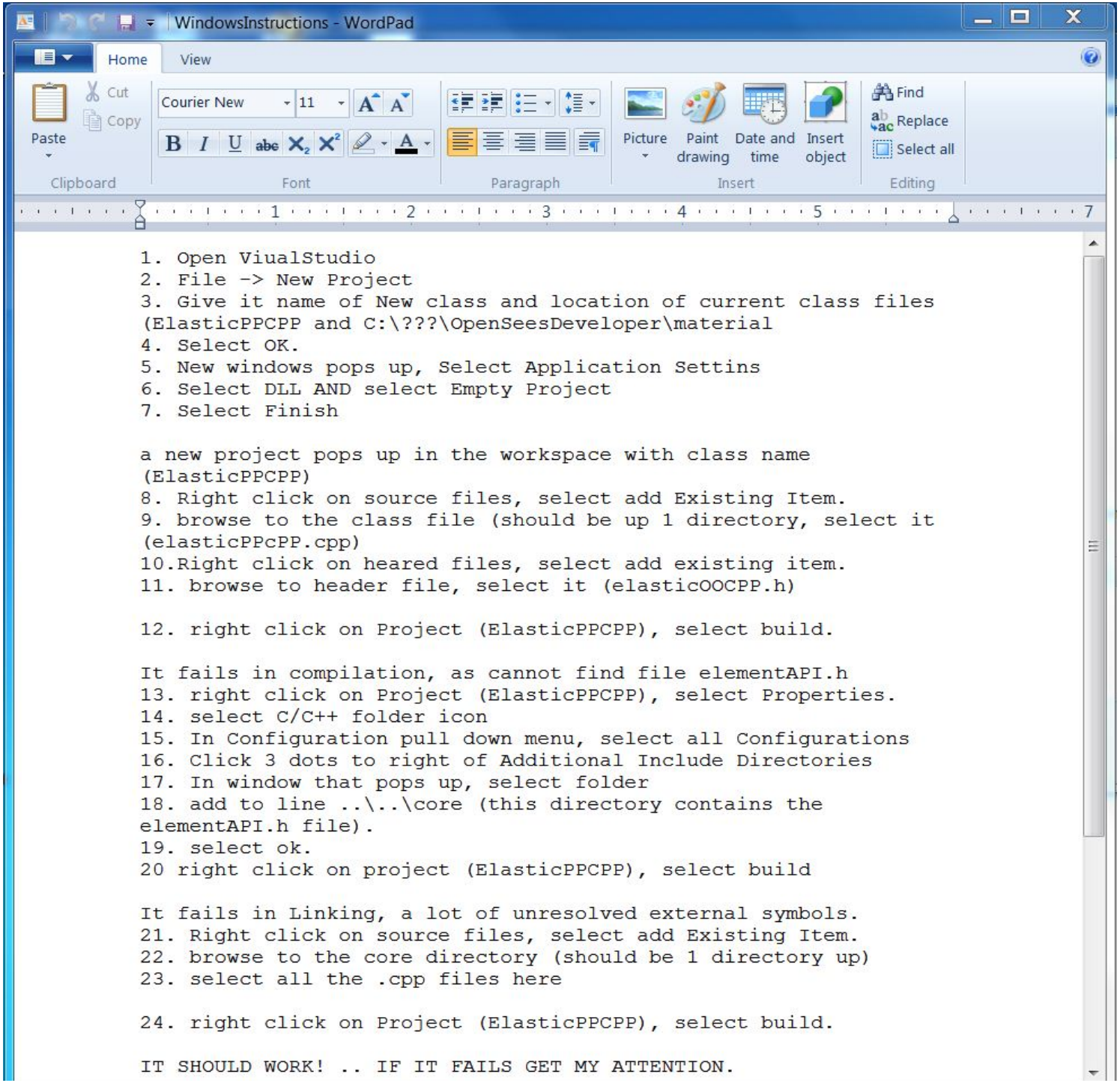
svn://opensees.berkeley.edu/usr/local/svn/OpenSees/trunk/DEVELOPER

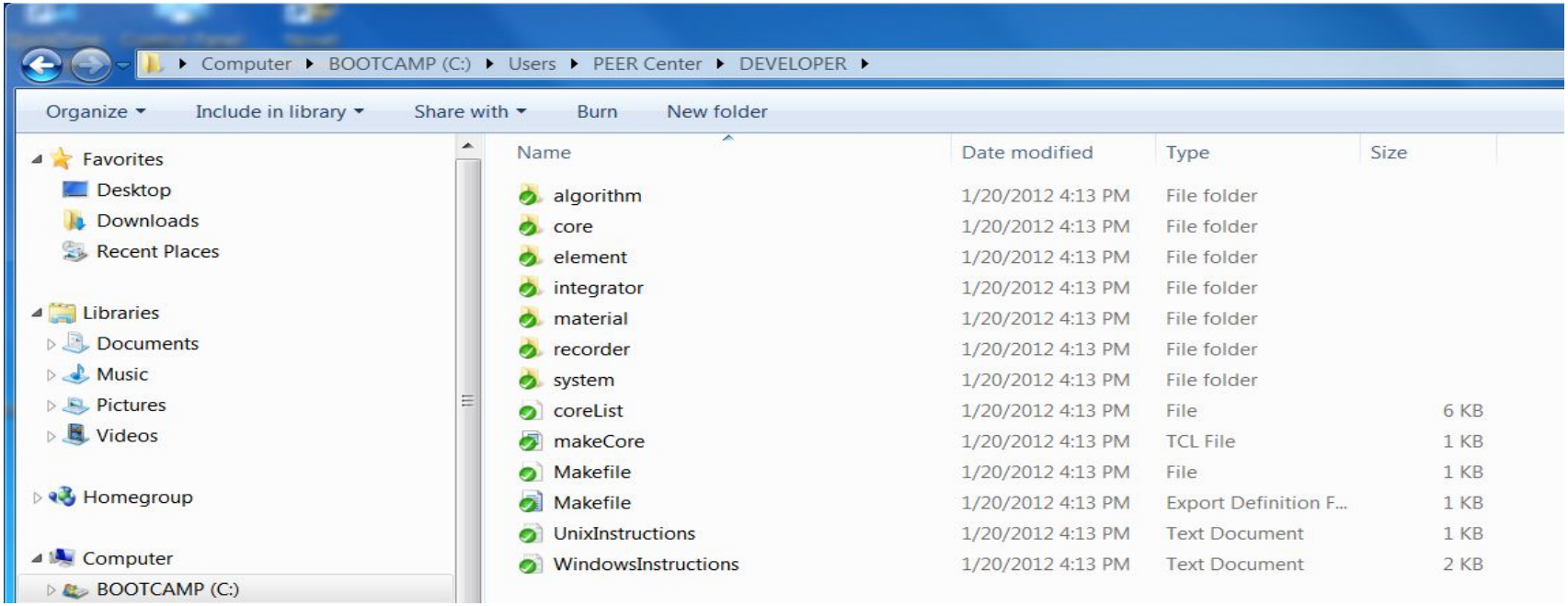
TortoiseSVN for Windows Users

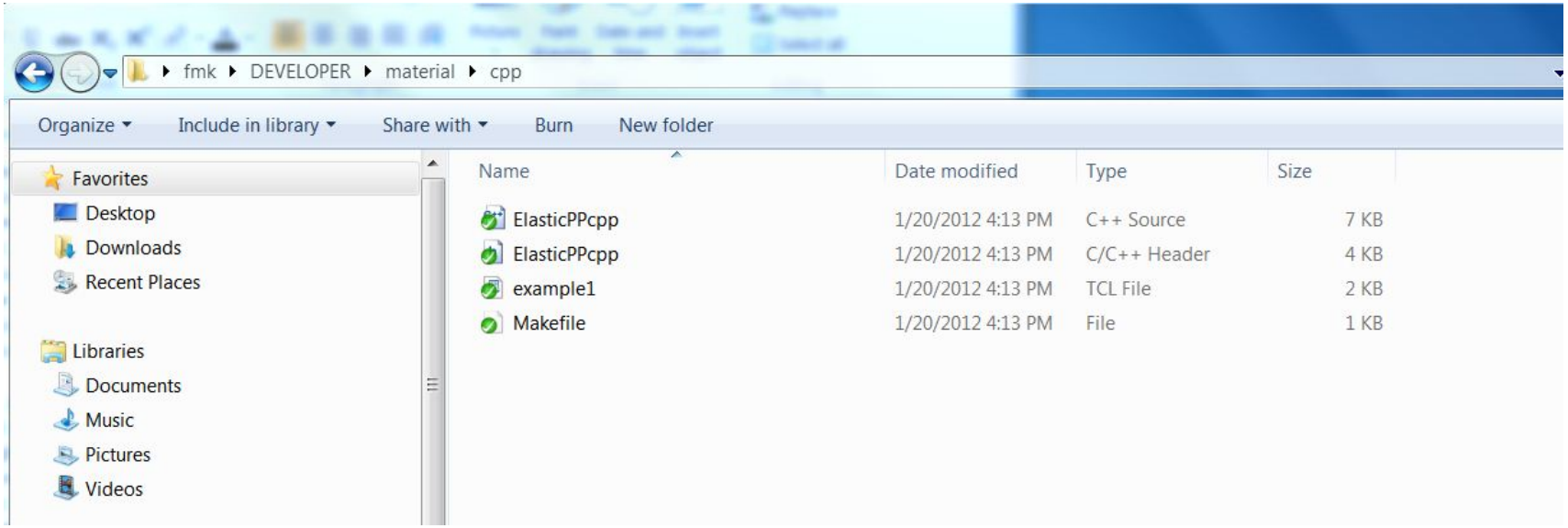


Source Code Tree in DEVELOPER




WindowsInstructions - WordPad
Home
View
Cut
Copy
Paste
Courier New
11
Clipboard
Font
Paragraph
Picture
Paint drawing
Date and time
Insert object
Insert
Find
Replace
Select all
Editing
1. Open ViualStudio
2. File -> New Project
3. Give it name of New class and location of current class files
(ElasticPPCPP and C:\???\OpenSeesDeveloper\material
4. Select OK.
5. New windows pops up, Select Application Settins
6. Select DLL AND select Empty Project
7. Select Finish
a new project pops up in the workspace with class name
(ElasticPPCPP)
8. Right click on source files, select add Existing Item.
9. browse to the class file (should be up 1 directory, select it
(elasticPPcPP.cpp)
10.Right click on heared files, select add existing item.
11. browse to header file, select it (elasticOOCPP.h)
12. right click on Project (ElasticPPCPP), select build.
It fails in compilation, as cannot find file elementAPI.h
13. right click on Project (ElasticPPCPP), select Properties.
14. select C/C++ folder icon
15. In Configuration pull down menu, select all Configurations
16. Click 3 dots to right of Additional Include Directories
17. In window that pops up, select folder
18. add to line ..\..\core (this directory contains the
elementAPI.h file).
19. select ok.
20 right click on project (ElasticPPCPP), select build
It fails in Linking, a lot of unresolved external symbols.
21. Right click on source files, select add Existing Item.
22. browse to the core directory (should be 1 directory up)
23. select all the .cpp files here
24. right click on Project (ElasticPPCPP), select build.
IT SHOULD WORK! .. IF IT FAILS GET MY ATTENTION.


Computer ▸ BOOTCAMP (C:) ▸ Users ▸ PEER Center ▸ DEVELOPER ▸
Organize
Include in library
Share with
Burn
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
BOOTCAMP (C:)
Name
Date modified
Type
Size
algorithm 1/20/2012 4:13 PM File folder
core 1/20/2012 4:13 PM File folder
element 1/20/2012 4:13 PM File folder
integrator 1/20/2012 4:13 PM File folder
material 1/20/2012 4:13 PM File folder
recorder 1/20/2012 4:13 PM File folder
system 1/20/2012 4:13 PM File folder
coreList 1/20/2012 4:13 PM File 6 KB
makeCore 1/20/2012 4:13 PM TCL File 1 KB
Makefile 1/20/2012 4:13 PM File 1 KB
Makefile 1/20/2012 4:13 PM Export Definition F... 1 KB
UnixInstructions 1/20/2012 4:13 PM Text Document 1 KB
WindowsInstructions 1/20/2012 4:13 PM Text Document 2 KB


fmk ▸ DEVELOPER ▸ material ▸ cpp
Organize
Include in library
Share with
Burn
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Name
Date modified
Type
Size
ElasticPPcpp 1/20/2012 4:13 PM C++ Source 7 KB
ElasticPPcpp 1/20/2012 4:13 PM C/C++ Header 4 KB
example1 1/20/2012 4:13 PM TCL File 2 KB
Makefile 1/20/2012 4:13 PM File 1 KB

Interface (.h file)

```
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 last commit
        double trialStrain, trialStress, trialTangent;
        double commitStrain, commitStress, commitTangent;
};
```

Material Name

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),
 commitStrein(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;
}
```

OpenSees Interpreter (.cpp file)

Interpreter looking for this function in lib

```
OPS_ElasticPPcpp()
{
  if (numElasticPPcpp == 0) {
    opserr << "ElasticPPcpp unaxial m
    numElasticPPcpp =1;
  }
  // Pointer to a uniaxial material that will be returned
  UniaxialMaterial *theMaterial = 0;
  int    iData[1];
  double dData[2];
  int numData;

  numData = 1;
  if (OPS_GetIntInput(&numData, iData) != 0) {
    opserr << "WARNING invalid uniaxialMaterial ElasticPP tag" << endln;
    return 0;
  }
  numData = 2;
  if (OPS_GetDoubleInput(&numData, dData) != 0) {
    opserr << "WARNING invalid E & ep\n";
    return 0;
  }

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

  return theMaterial;
}
```

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

parse the script for three material parameters

Function returns new material

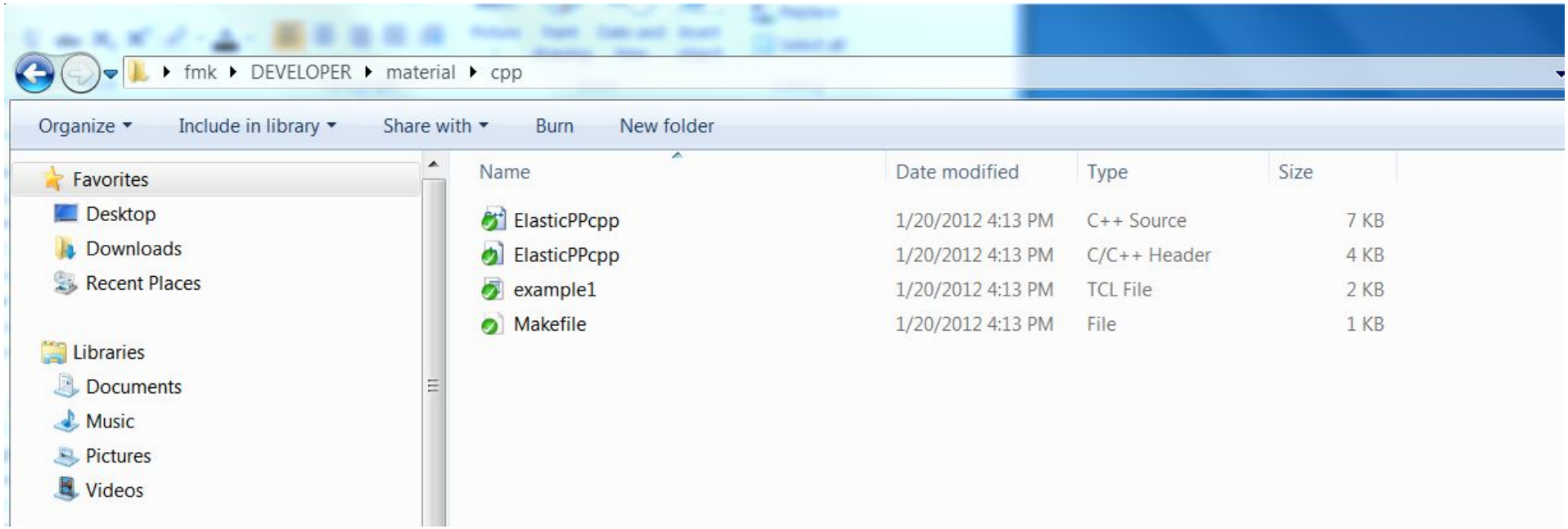
C & Fortran Procedural Languages Can Also Be Used

```
OPS_Export void
elasticPPc (matObj *thisObj,
            modelState *model,
            double *strain,
            double *tang,
            double *stress,
            int *isw,
            int *result)
{
  *result = 0;

  if (*isw == ISW_INIT) {

    double dData[2];
    int    iData[1];

 /* get the input data  - tag? E? e
    int numData = 1;
    OPS_GetIntInput(&numData, i
    numData = 2;
    OPS_GetDoubleInput(&numD

    /* Allocate the element state */
```

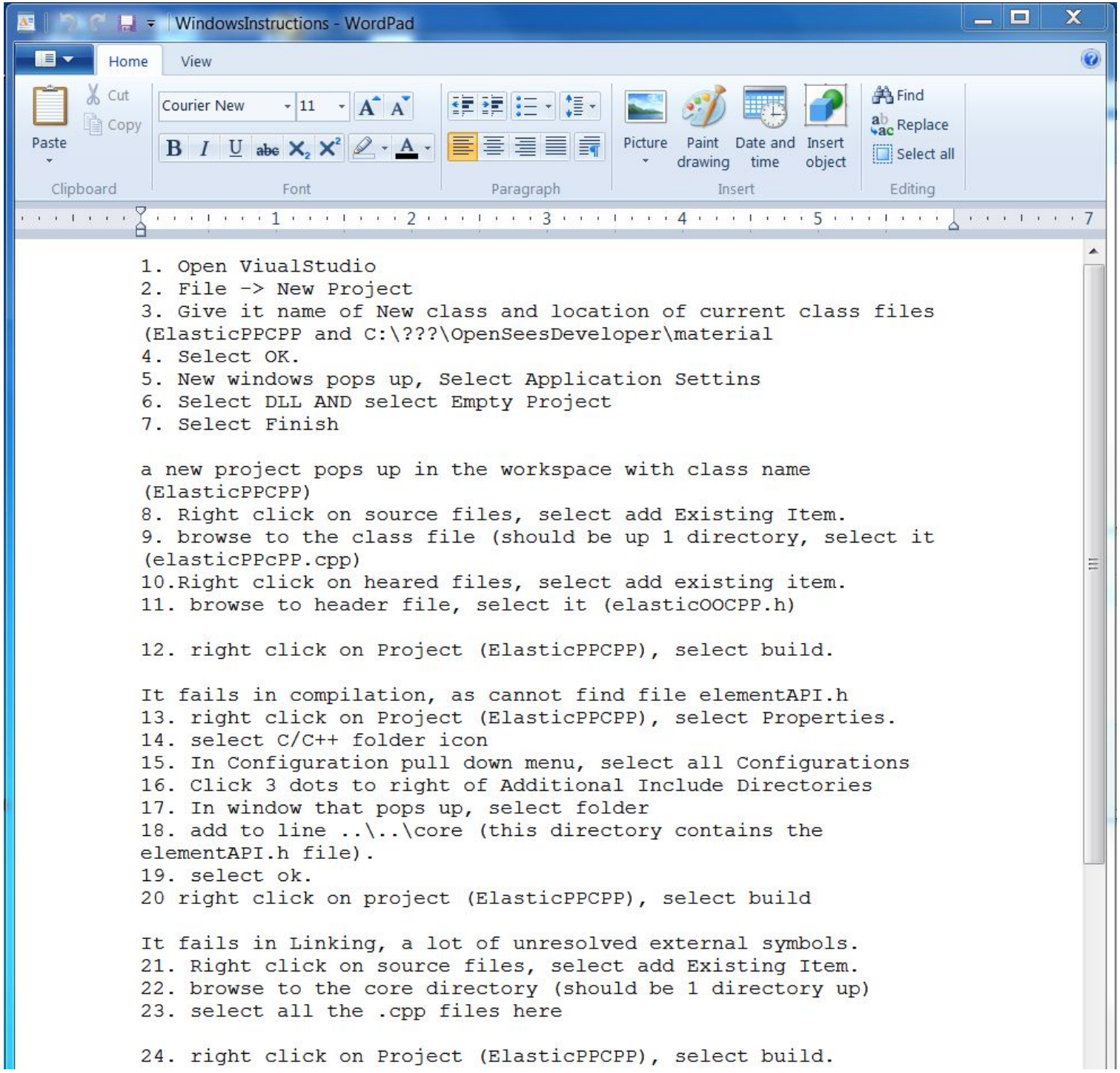
```
 SUBROUTINE ELASTICPPF(matObj,model,strain,tang,stress,is

!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

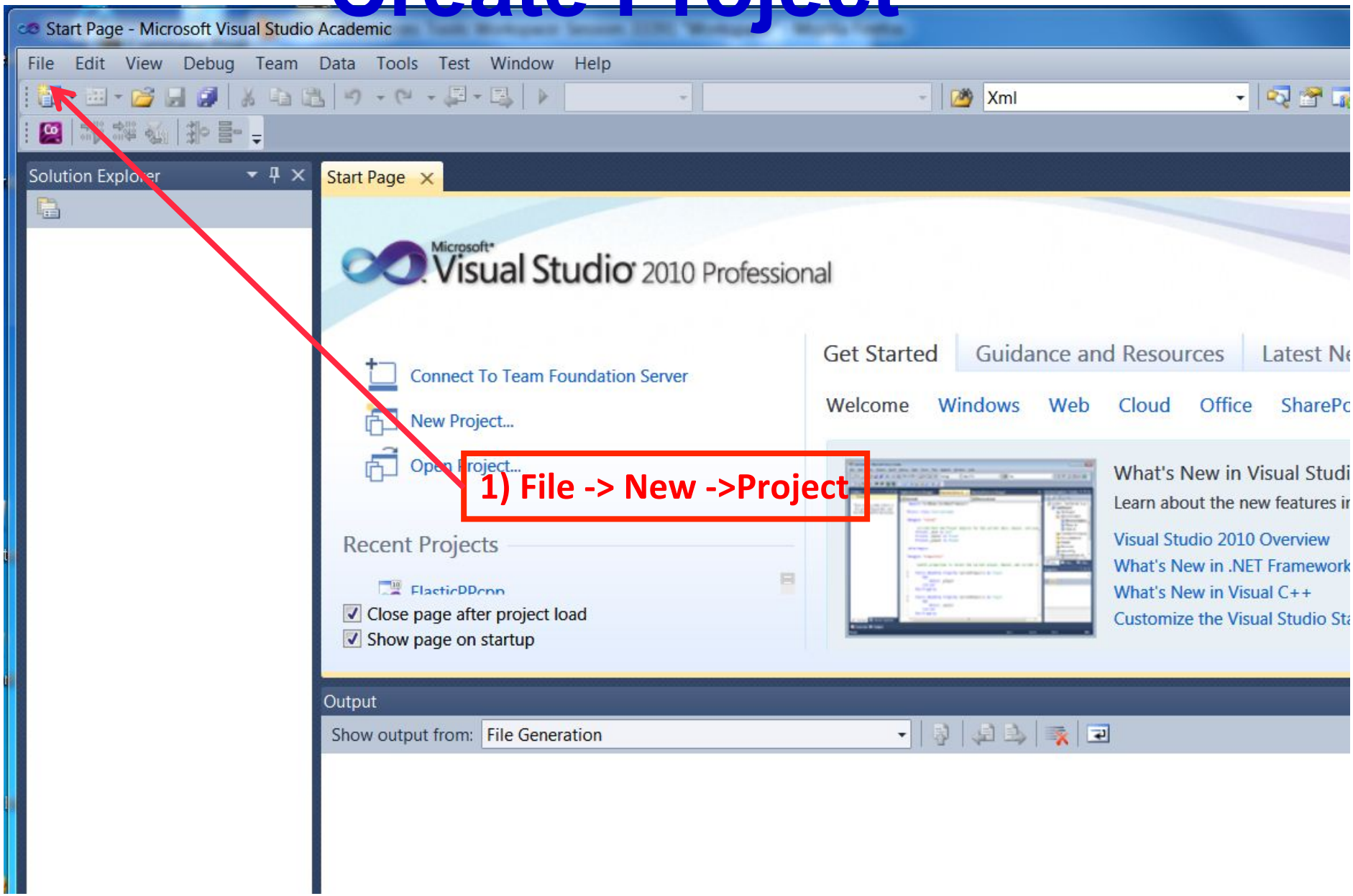
We Will Build the ElasticPPcpp.dll

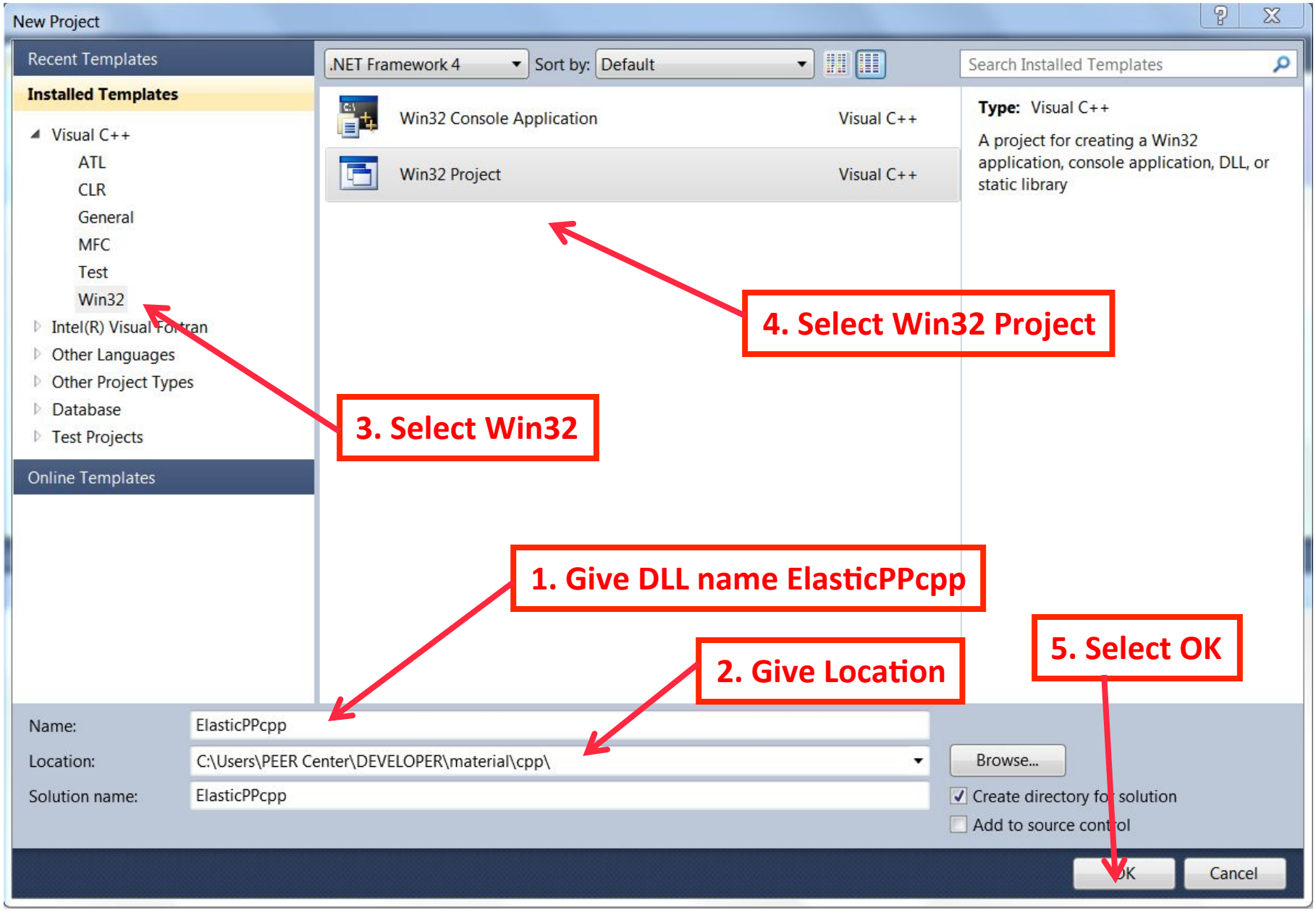


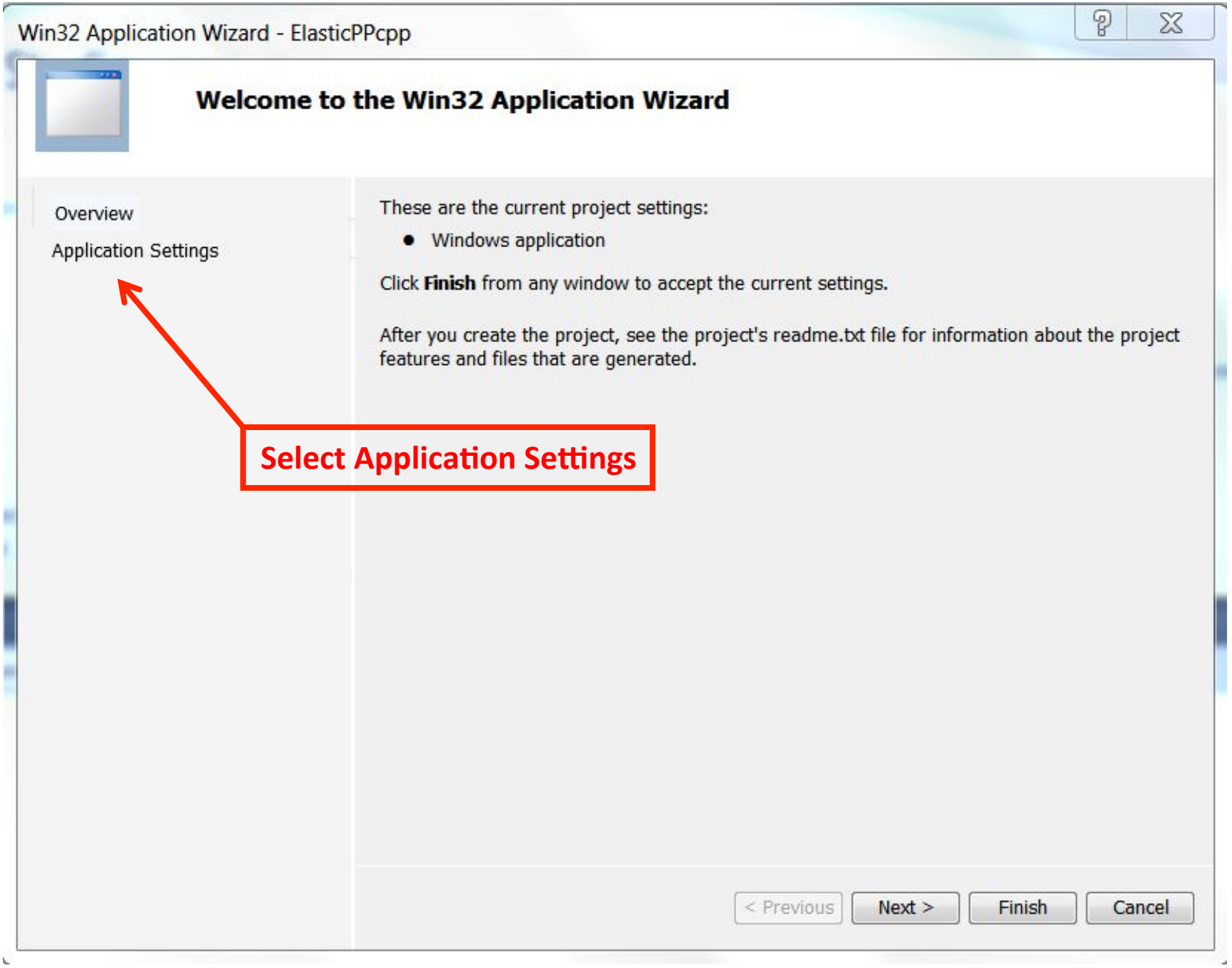
Source code and example are in /DEVELOPER/material/cpp

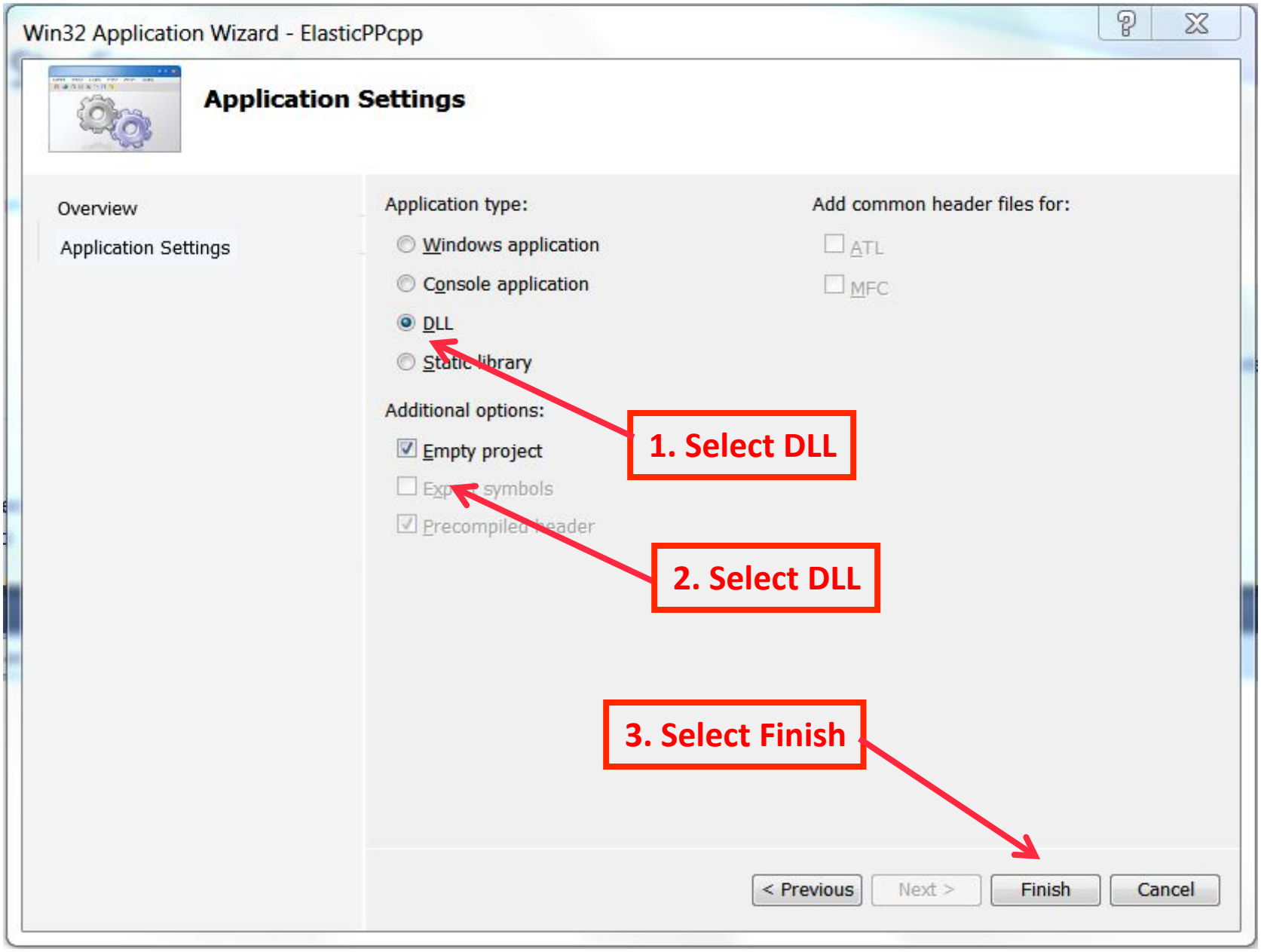

WindowsInstructions - WordPad
Home
View
Cut
Copy
Paste
Courier New
11
Clipboard
Font
Paragraph
Picture
Paint drawing
Date and time
Insert object
Insert
Find
Replace
Select all
Editing
1. Open ViualStudio
2. File -> New Project
3. Give it name of New class and location of current class files
(ElasticPPCPP and C:\???\OpenSeesDeveloper\material
4. Select OK.
5. New windows pops up, Select Application Settins
6. Select DLL AND select Empty Project
7. Select Finish
a new project pops up in the workspace with class name
(ElasticPPCPP)
8. Right click on source files, select add Existing Item.
9. browse to the class file (should be up 1 directory, select it
(elasticPPcPP.cpp)
10.Right click on heared files, select add existing item.
11. browse to header file, select it (elasticOOCPP.h)
12. right click on Project (ElasticPPCPP), select build.
It fails in compilation, as cannot find file elementAPI.h
13. right click on Project (ElasticPPCPP), select Properties.
14. select C/C++ folder icon
15. In Configuration pull down menu, select all Configurations
16. Click 3 dots to right of Additional Include Directories
17. In window that pops up, select folder
18. add to line ..\..\core (this directory contains the
elementAPI.h file).
19. select ok.
20 right click on project (ElasticPPCPP), select build
It fails in Linking, a lot of unresolved external symbols.
21. Right click on source files, select add Existing Item.
22. browse to the core directory (should be 1 directory up)
23. select all the .cpp files here
24. right click on Project (ElasticPPCPP), select build.

Create Project

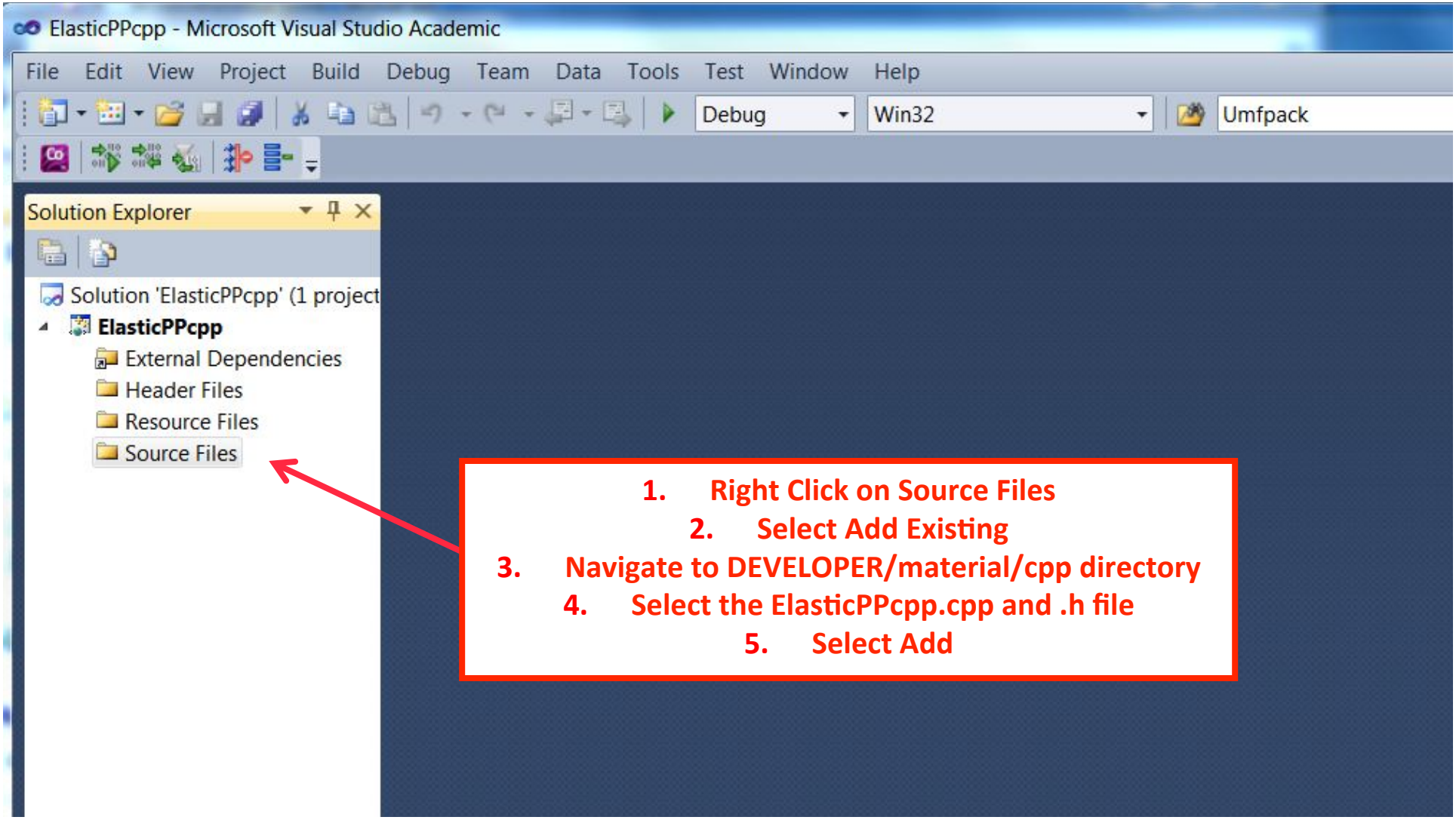


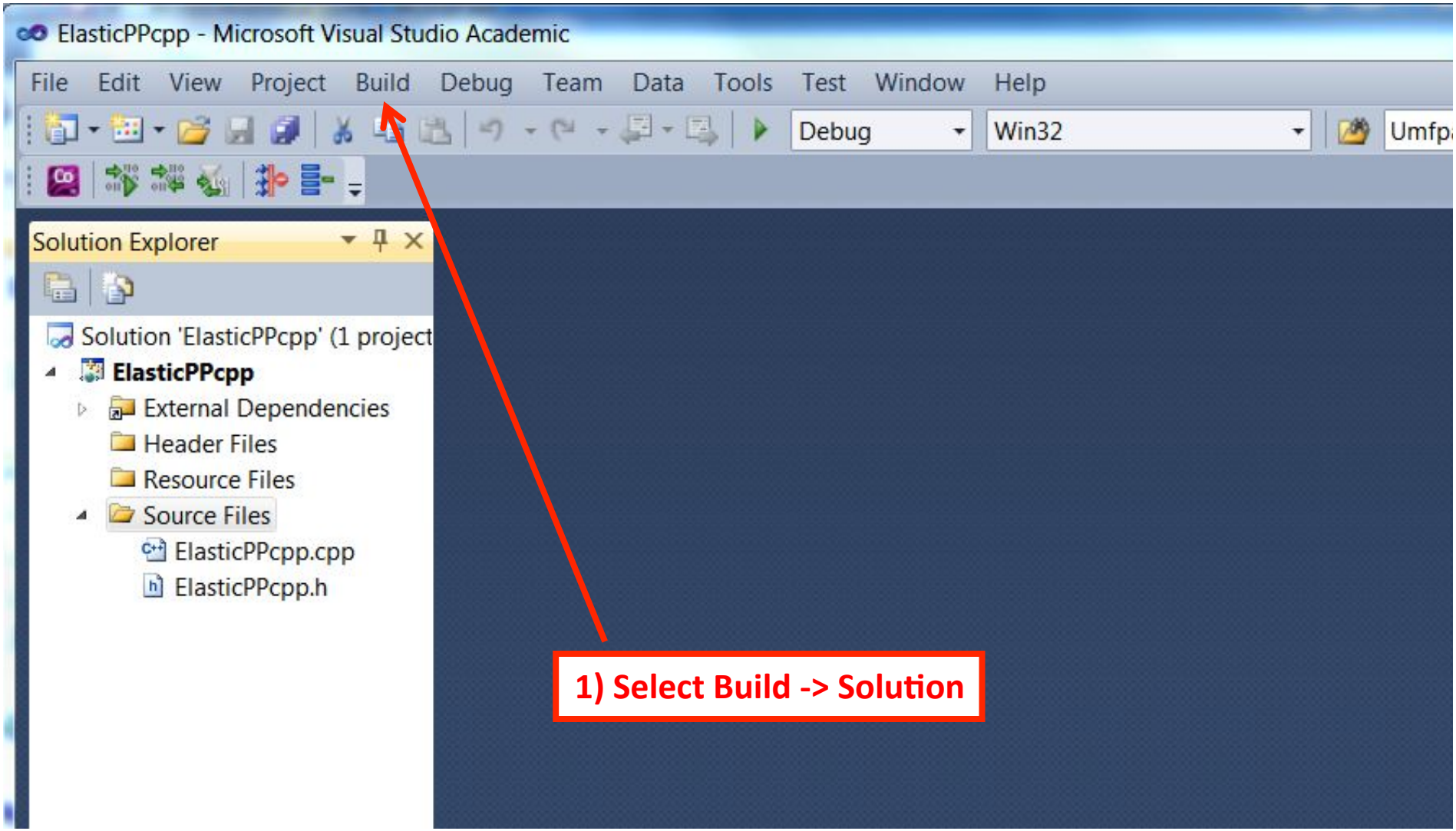

New Project
Recent Templates
Installed Templates
Visual C++
ATL
CLR
General
MFC
Test
Win32
Intel(R) Visual Fortran
Other Languages
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by:
Default
Search Installed Templates
Win32 Console Application
Visual C++
Win32 Project
Visual C++
Type: Visual C++
A project for creating a Win32 application, console application, DLL, or static library
4. Select Win32 Project
3. Select Win32
1. Give DLL name ElasticPPcpp
2. Give Location
5. Select OK
Name:
ElasticPPcpp
Location:
C:\Users\PEER Center\DEVELOPER\material\cpp\
Browse...
Solution name:
ElasticPPcpp
Create directory for solution
Add to source control
OK
Cancel

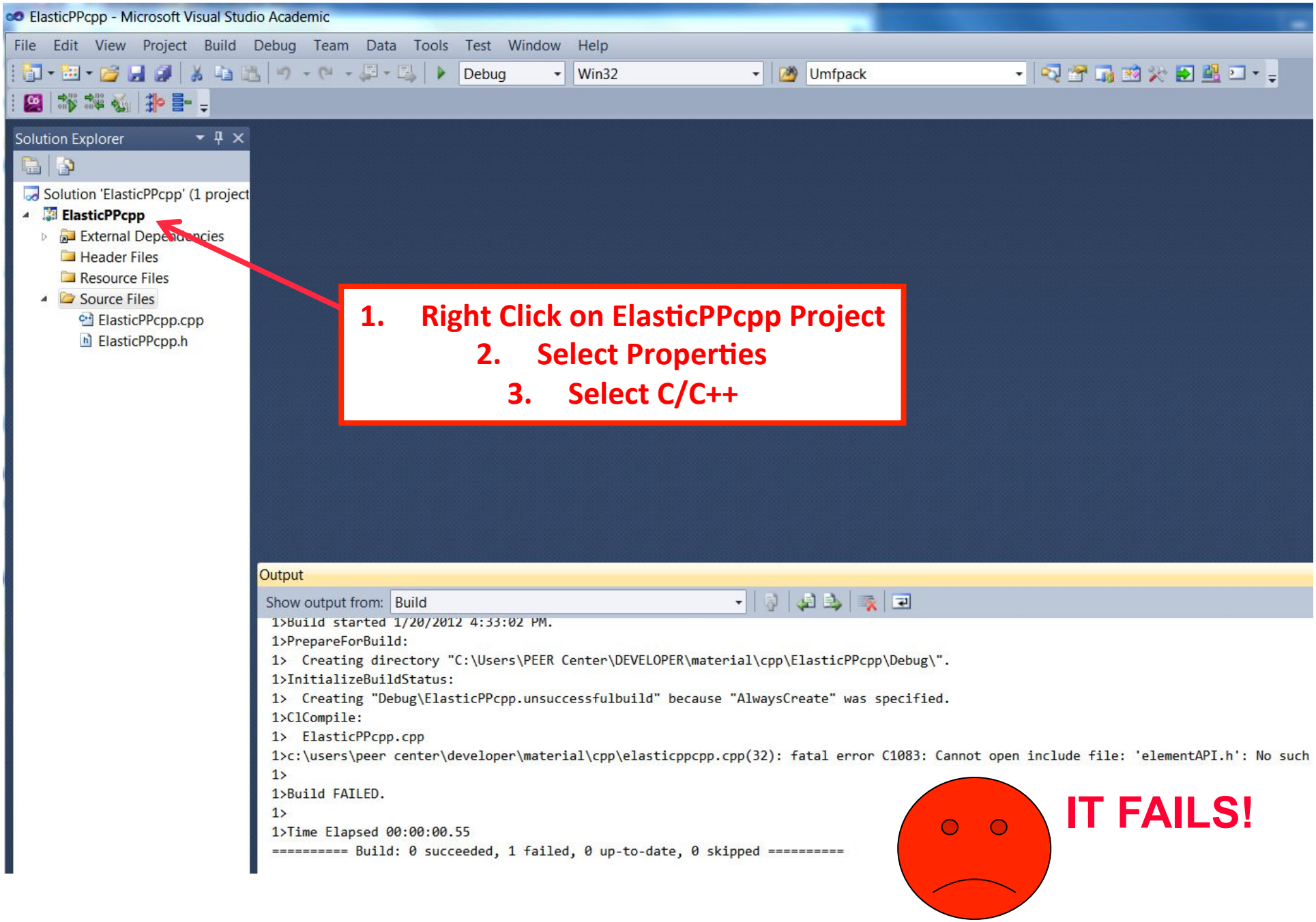

Win32 Application Wizard - ElasticPPcpp
Welcome to the Win32 Application Wizard
Overview
Application Settings
These are the current project settings:
Windows application
Click Finish from any window to accept the current settings.
After you create the project, see the project's readme.txt file for information about the project features and files that are generated.
Select Application Settings
< Previous
Next >
Finish
Cancel

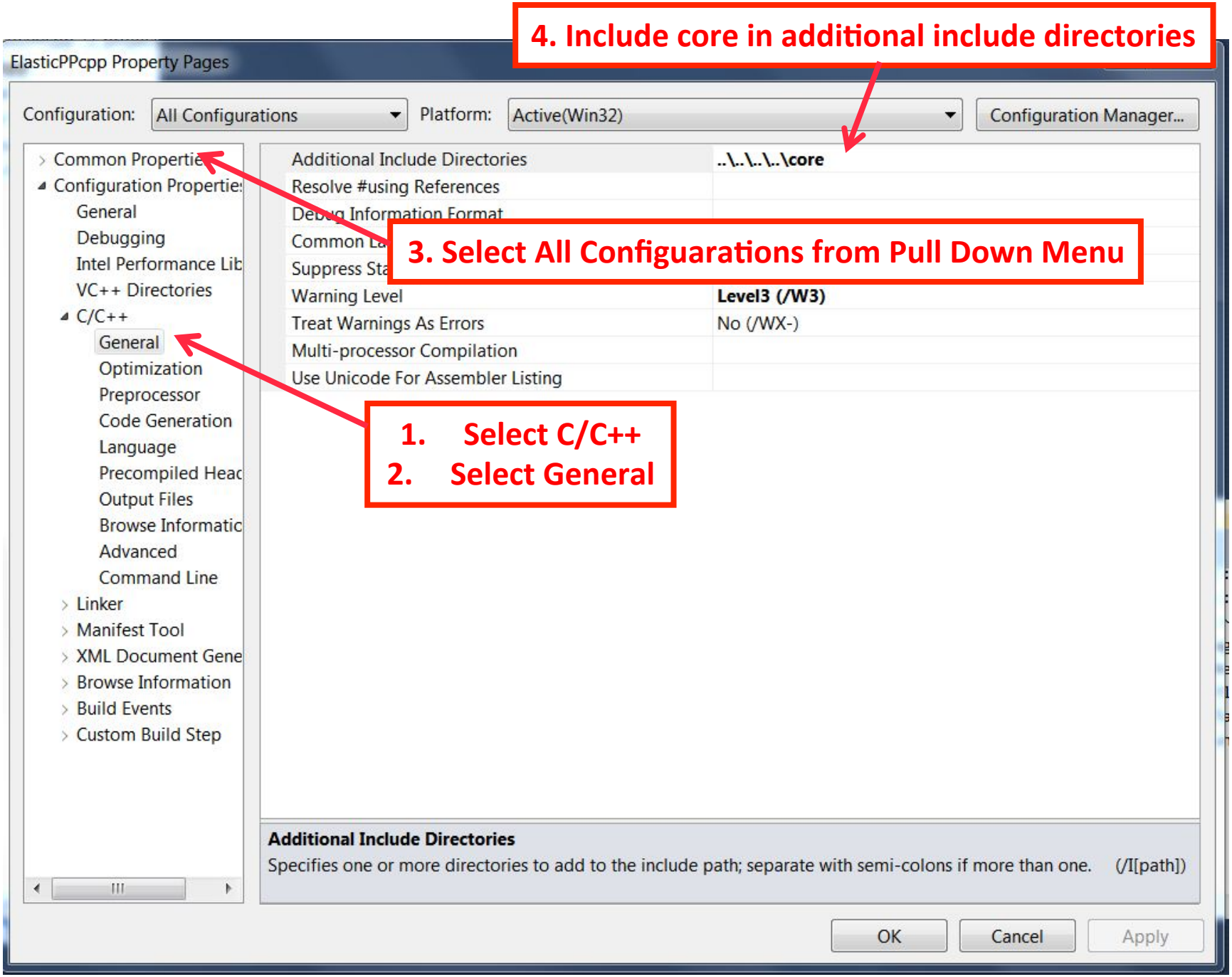

Win32 Application Wizard - ElasticPPcpp
Application Settings
Overview
Application Settings
Application type:
Windows application
Console application
DLL
Static library
Additional options:
Empty project
Precompiled header
Add common header files for:
ATL
MFC
1. Select DLL
2. Select DLL
3. Select Finish
< Previous
Next >
Finish
Cancel

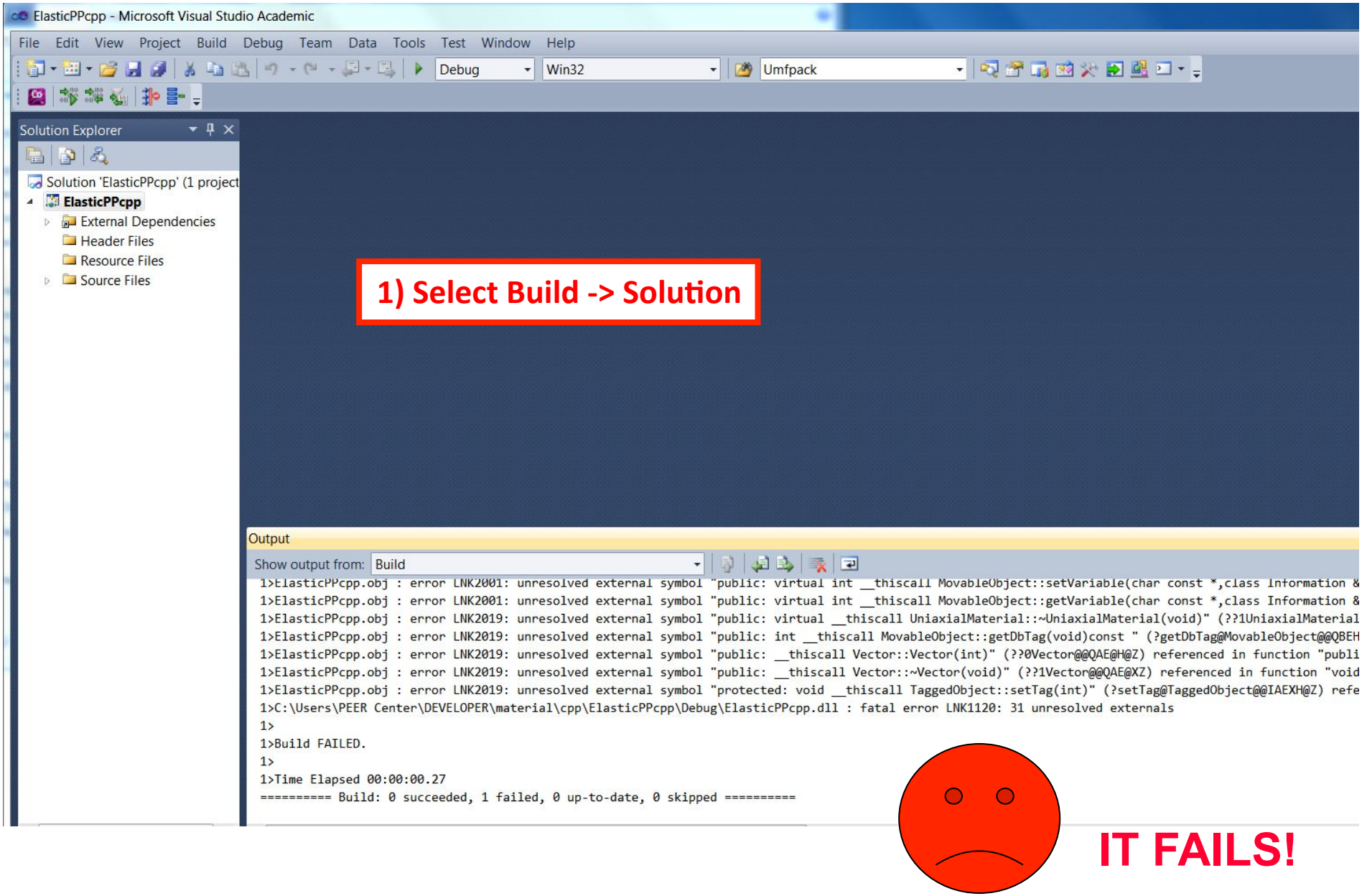
Add Files To Project

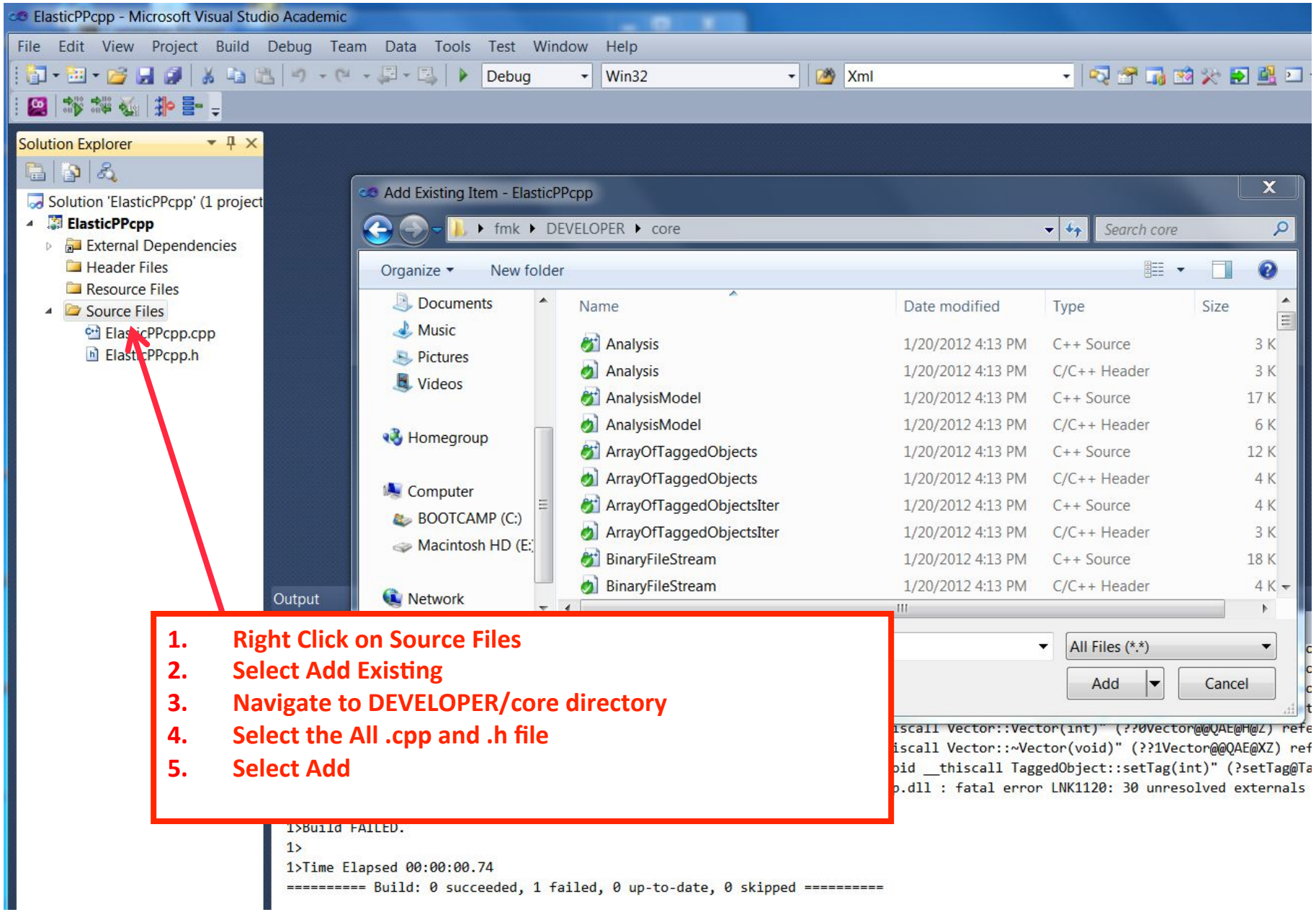


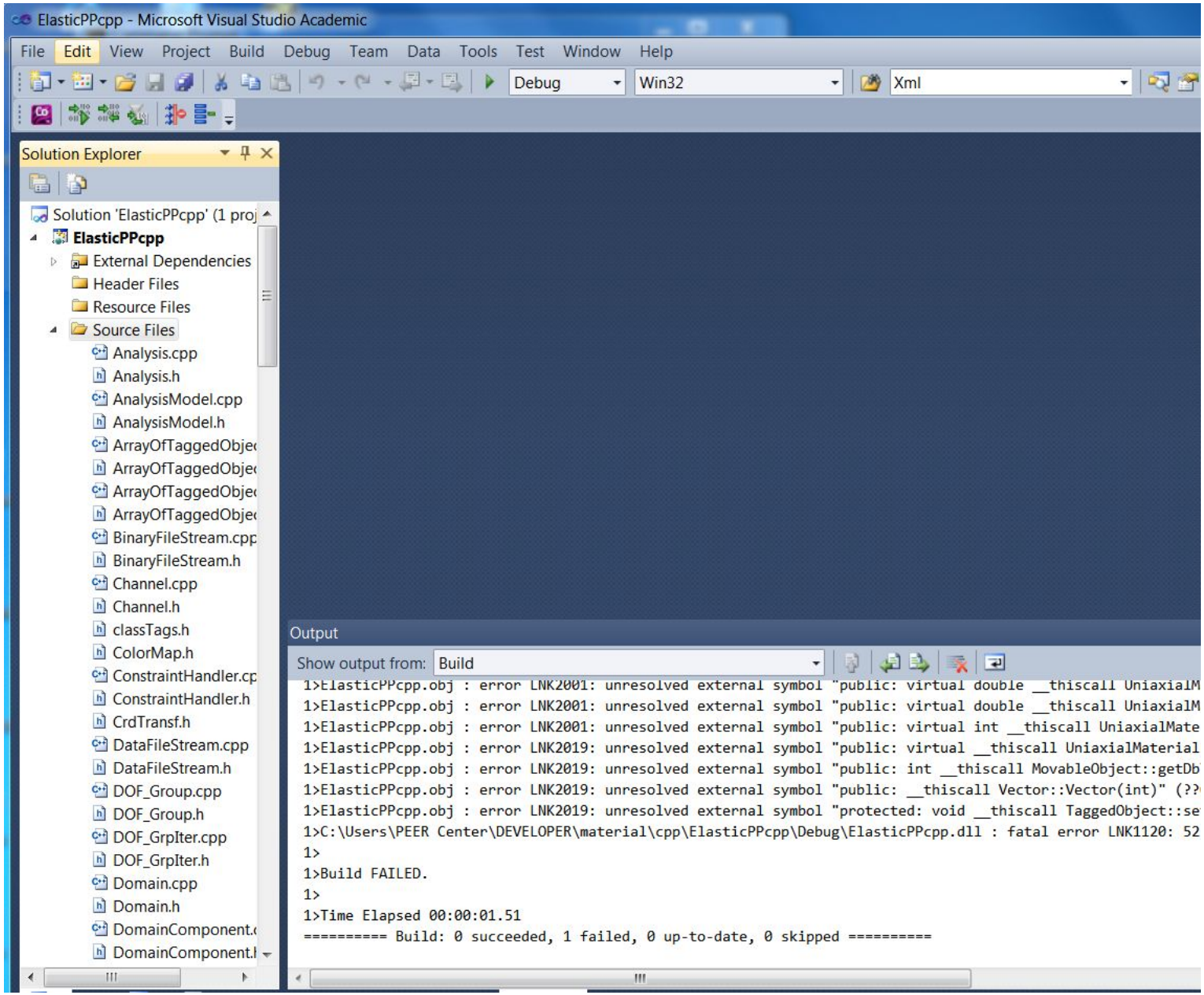

ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Solution Explorer
Solution 'ElasticPPcpp' (1 project
ElasticPPcpp
External Dependencies
Header Files
Resource Files
Source Files
ElasticPPcpp.cpp
ElasticPPcpp.h
1) Select Build -> Solution

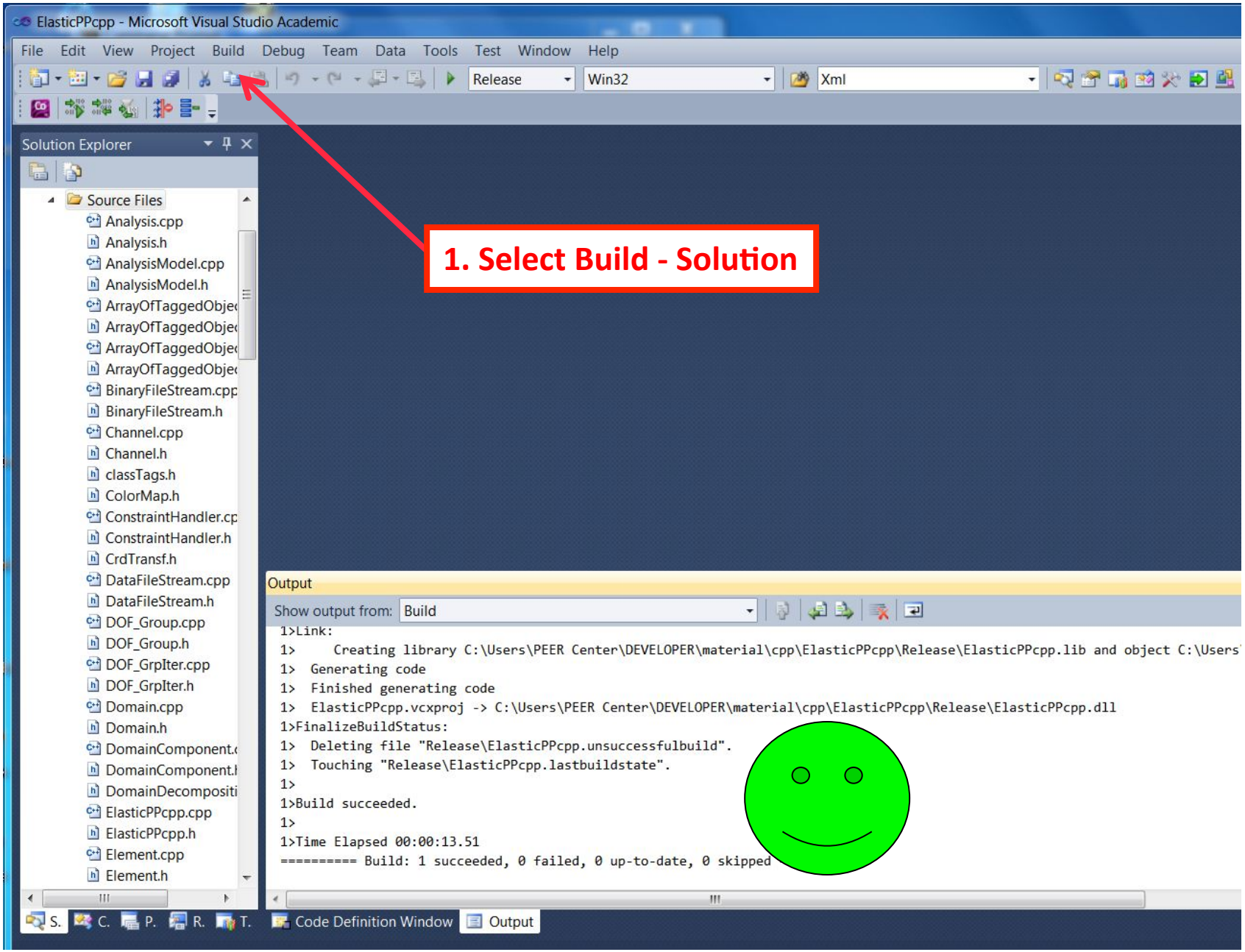

ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Umfpack
Solution Explorer
Solution 'ElasticPPcpp' (1 project
ElasticPPcpp
External Dependencies
Header Files
Resource Files
Source Files
ElasticPPcpp.cpp
ElasticPPcpp.h
1. Right 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!


4. Include core in additional include directories
ElasticPPcpp Property Pages
Configuration: All Configurations
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
Intel Performance Lib
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Head
Output Files
Browse Informatic
Advanced
Command Line
Linker
Manifest Tool
XML Document Gene
Browse Information
Build Events
Custom Build Step
Additional Include Directories
..\..\..\..\core
Resolve #using References
Debug Information Format
Common La
Suppress Sta
Warning Level
Level3 (/W3)
Treat Warnings As Errors
No (/WX-)
Multi-processor Compilation
Use Unicode For Assembler Listing
3. Select All Configuarations from Pull Down Menu
1. Select C/C++
2. Select General
Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])
OK
Cancel
Apply

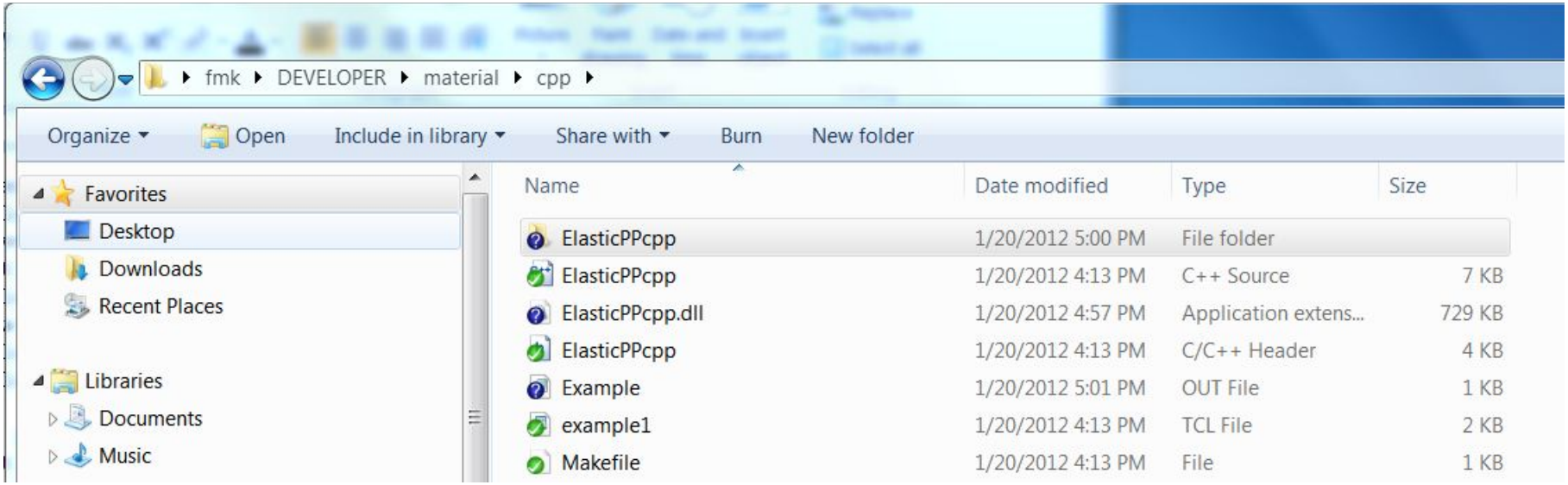

ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Umfpack
Solution Explorer
Solution 'ElasticPPcpp' (1 project
ElasticPPcpp
External Dependencies
Header Files
Resource Files
Source Files
1) Select Build -> Solution
Output
Show 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!


ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Xml
Solution Explorer
Solution 'ElasticPPcpp' (1 project
ElasticPPcpp
External Dependencies
Header Files
Resource Files
Source Files
ElasticPPcpp.cpp
ElasticPPcpp.h
Add Existing Item - ElasticPPcpp
fmk ▸ DEVELOPER ▸ core
Search core
Organize
New folder
Documents
Music
Pictures
Videos
Homegroup
Computer
BOOTCAMP (C:)
Macintosh HD (E:)
Network
Name
Date modified
Type
Size
Analysis 1/20/2012 4:13 PM C++ Source 3 K
Analysis 1/20/2012 4:13 PM C/C++ Header 3 K
AnalysisModel 1/20/2012 4:13 PM C++ Source 17 K
AnalysisModel 1/20/2012 4:13 PM C/C++ Header 6 K
ArrayOfTaggedObjects 1/20/2012 4:13 PM C++ Source 12 K
ArrayOfTaggedObjects 1/20/2012 4:13 PM C/C++ Header 4 K
ArrayOfTaggedObjectsIter 1/20/2012 4:13 PM C++ Source 4 K
ArrayOfTaggedObjectsIter 1/20/2012 4:13 PM C/C++ Header 3 K
BinaryFileStream 1/20/2012 4:13 PM C++ Source 18 K
BinaryFileStream 1/20/2012 4:13 PM C/C++ Header 4 K
All Files (*.*)
Add
Cancel
Output
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 ==========

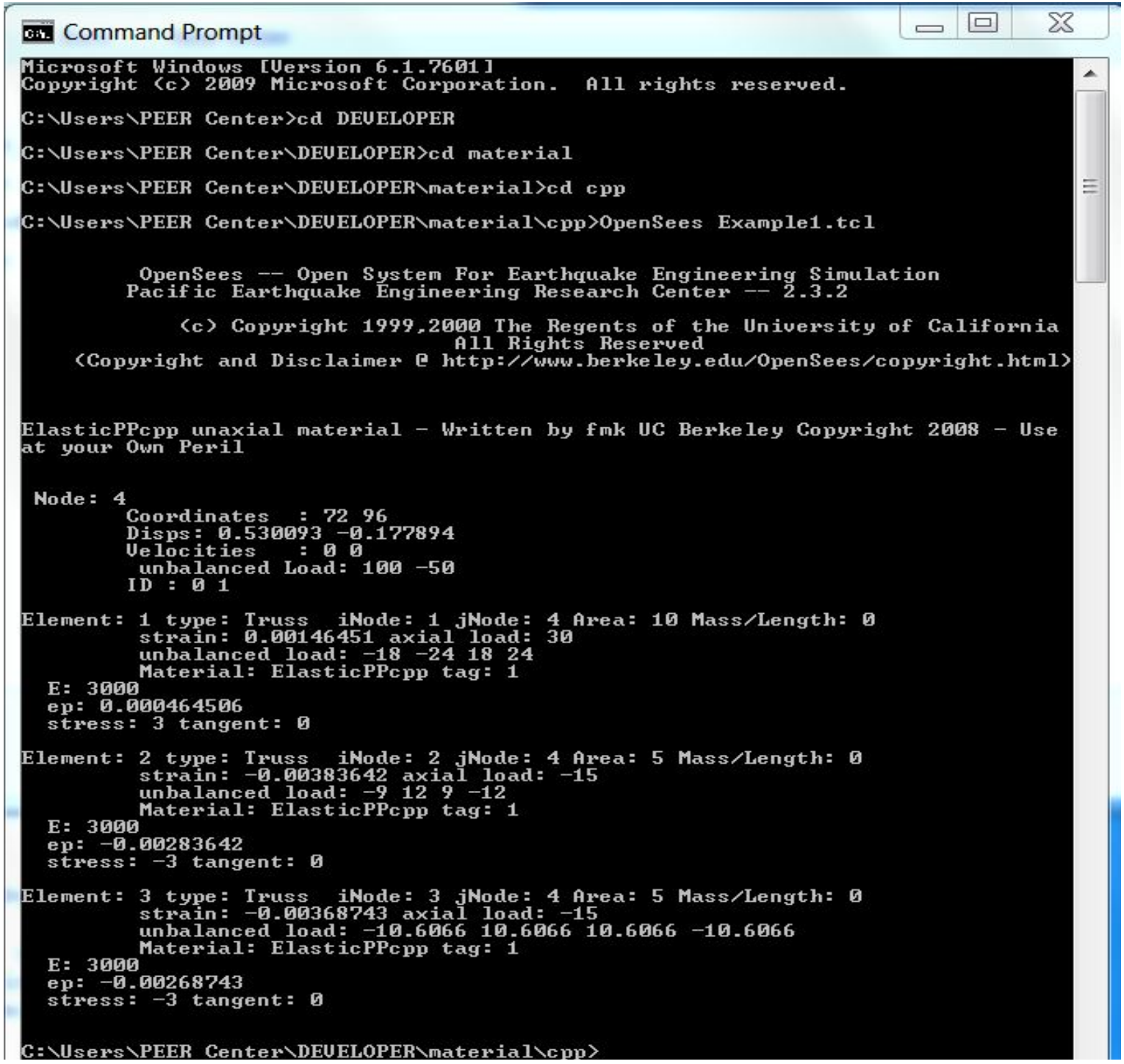

ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Xml
Solution Explorer
Solution 'ElasticPPcpp' (1 proj
ElasticPPcpp
External Dependencies
Header Files
Resource Files
Source Files
Analysis.cpp
Analysis.h
AnalysisModel.cpp
AnalysisModel.h
ArrayOfTaggedObje
ArrayOfTaggedObje
ArrayOfTaggedObje
ArrayOfTaggedObje
BinaryFileStream.cpp
BinaryFileStream.h
Channel.cpp
Channel.h
classTags.h
ColorMap.h
ConstraintHandler.cp
ConstraintHandler.h
CrdTransf.h
DataFileStream.cpp
DataFileStream.h
DOF_Group.cpp
DOF_Group.h
DOF_GrpIter.cpp
DOF_GrpIter.h
Domain.cpp
Domain.h
DomainComponent.c
DomainComponent.h
Output
Show output from: Build
1>ElasticPPcpp.obj : error LNK2001: unresolved external symbol "public: virtual double __thiscall UniaxialM
1>ElasticPPcpp.obj : error LNK2001: unresolved external symbol "public: virtual double __thiscall UniaxialM
1>ElasticPPcpp.obj : error LNK2001: unresolved external symbol "public: virtual int __thiscall UniaxialMate
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: virtual __thiscall UniaxialMaterial
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: int __thiscall MovableObject::getDb
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "public: __thiscall Vector::Vector(int)" (??
1>ElasticPPcpp.obj : error LNK2019: unresolved external symbol "protected: void __thiscall TaggedObject::se
1>C:\Users\PEER Center\DEVELOPER\material\cpp\ElasticPPcpp\Debug\ElasticPPcpp.dll : fatal error LNK1120: 52
1>
1>Build FAILED.
1>
1>Time Elapsed 00:00:01.51
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========


ElasticPPcpp - Microsoft Visual Studio Academic
File Edit View Project Build Debug Team Data Tools Test Window Help
Release
Win32
Xml
1. Select Build - Solution
Solution Explorer
Source Files
Analysis.cpp
Analysis.h
AnalysisModel.cpp
AnalysisModel.h
BinaryFileStream.cpp
BinaryFileStream.h
Channel.cpp
Channel.h
classTags.h
ColorMap.h
ConstraintHandler.h
CrdTransf.h
DataFileStream.cpp
DataFileStream.h
DOF_Group.cpp
DOF_Group.h
DOF_GrpIter.cpp
DOF_GrpIter.h
Domain.cpp
Domain.h
ElasticPPcpp.cpp
ElasticPPcpp.h
Element.cpp
Element.h
Output
Show output from: Build
1>Link:
1> Creating library C:\Users\PEER Center\DEVELOPER\material\cpp\ElasticPPcpp\Release\ElasticPPcpp.lib and object C:\Users
1> Generating code
1> Finished generating code
1> ElasticPPcpp.vcxproj -> C:\Users\PEER Center\DEVELOPER\material\cpp\ElasticPPcpp\Release\ElasticPPcpp.dll
1>FinalizeBuildStatus:
1> Deleting file "Release\ElasticPPcpp.unsuccessfulbuild".
1> Touching "Release\ElasticPPcpp.lastbuildstate".
1>
1>Build succeeded.
1>
1>Time Elapsed 00:00:13.51
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped
Code Definition Window
Output

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

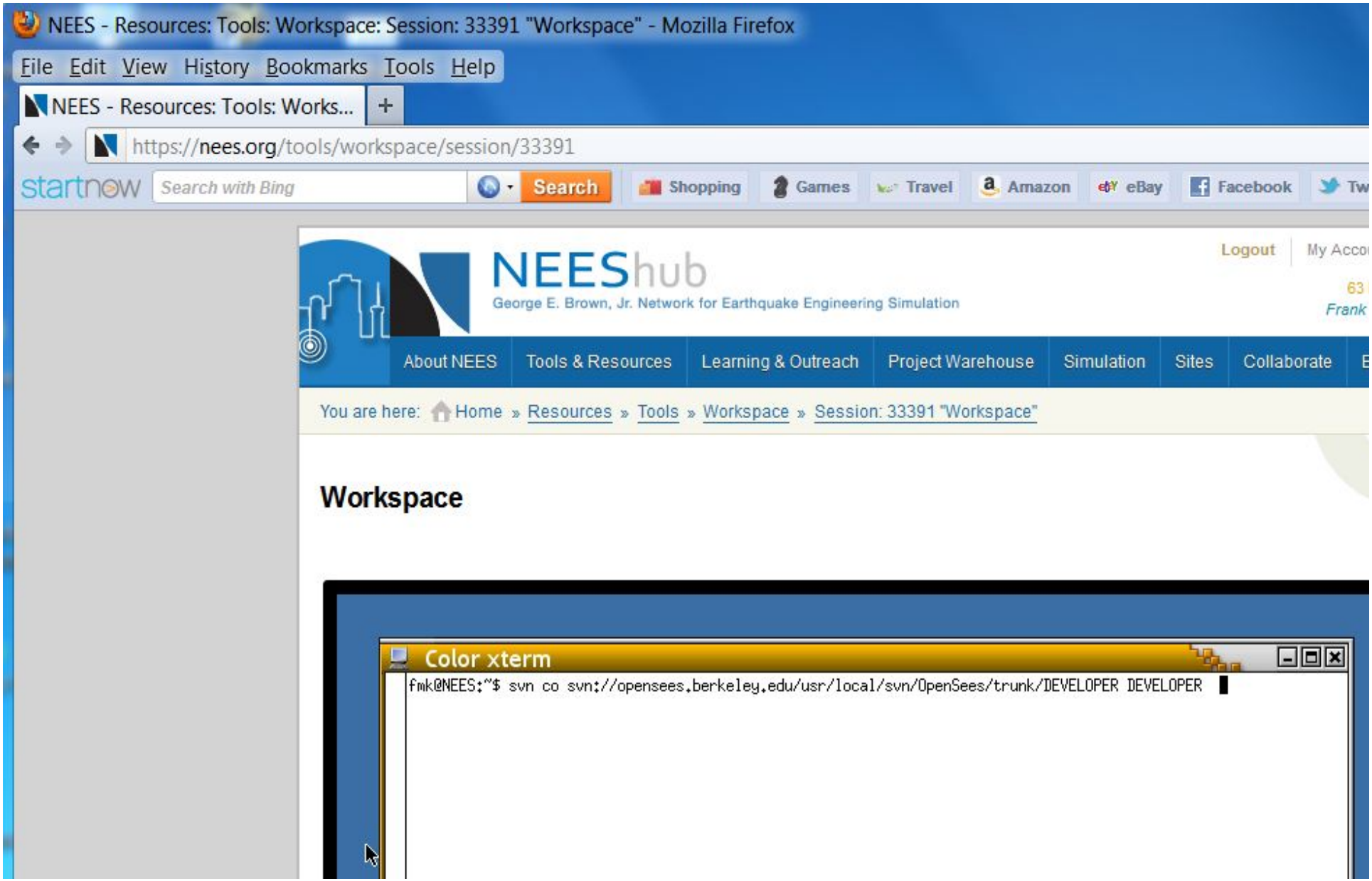
3. type make

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

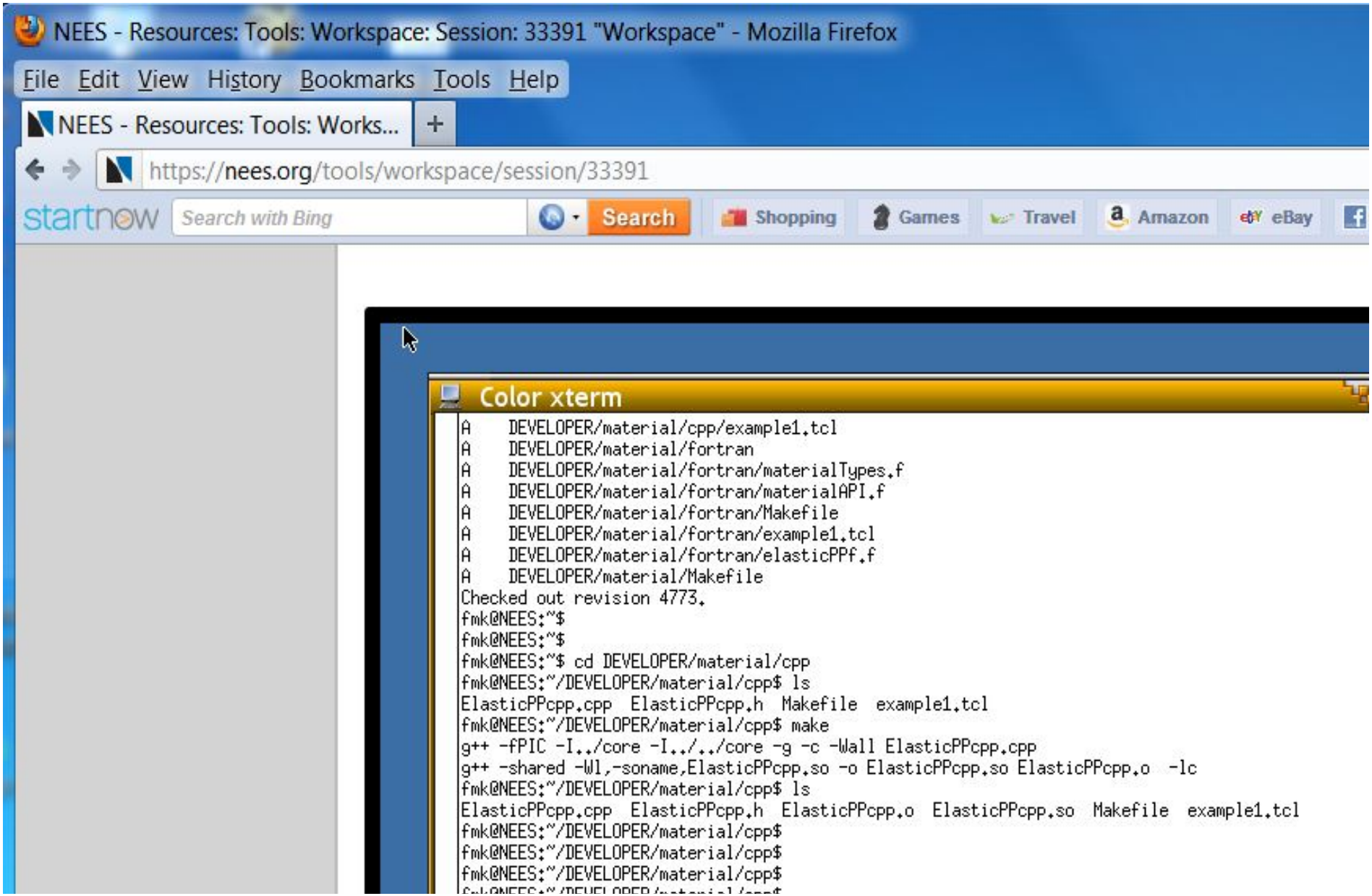
Can test it using >OpenSees example1.tcl

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



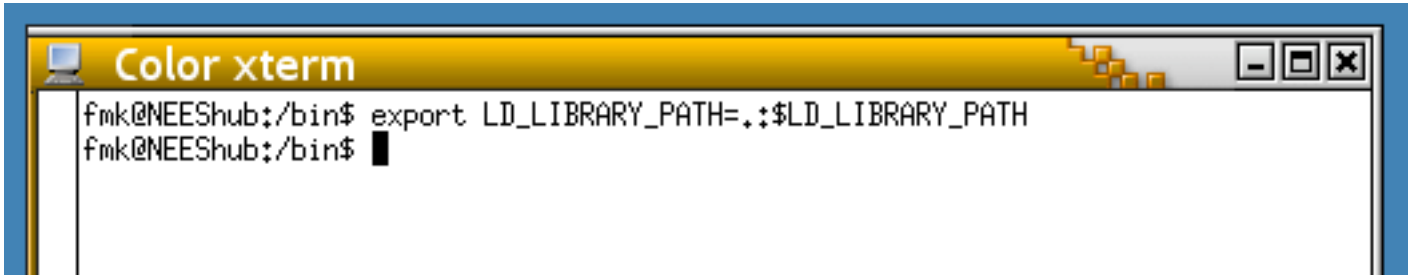
2/3: cd to Directory & Type make



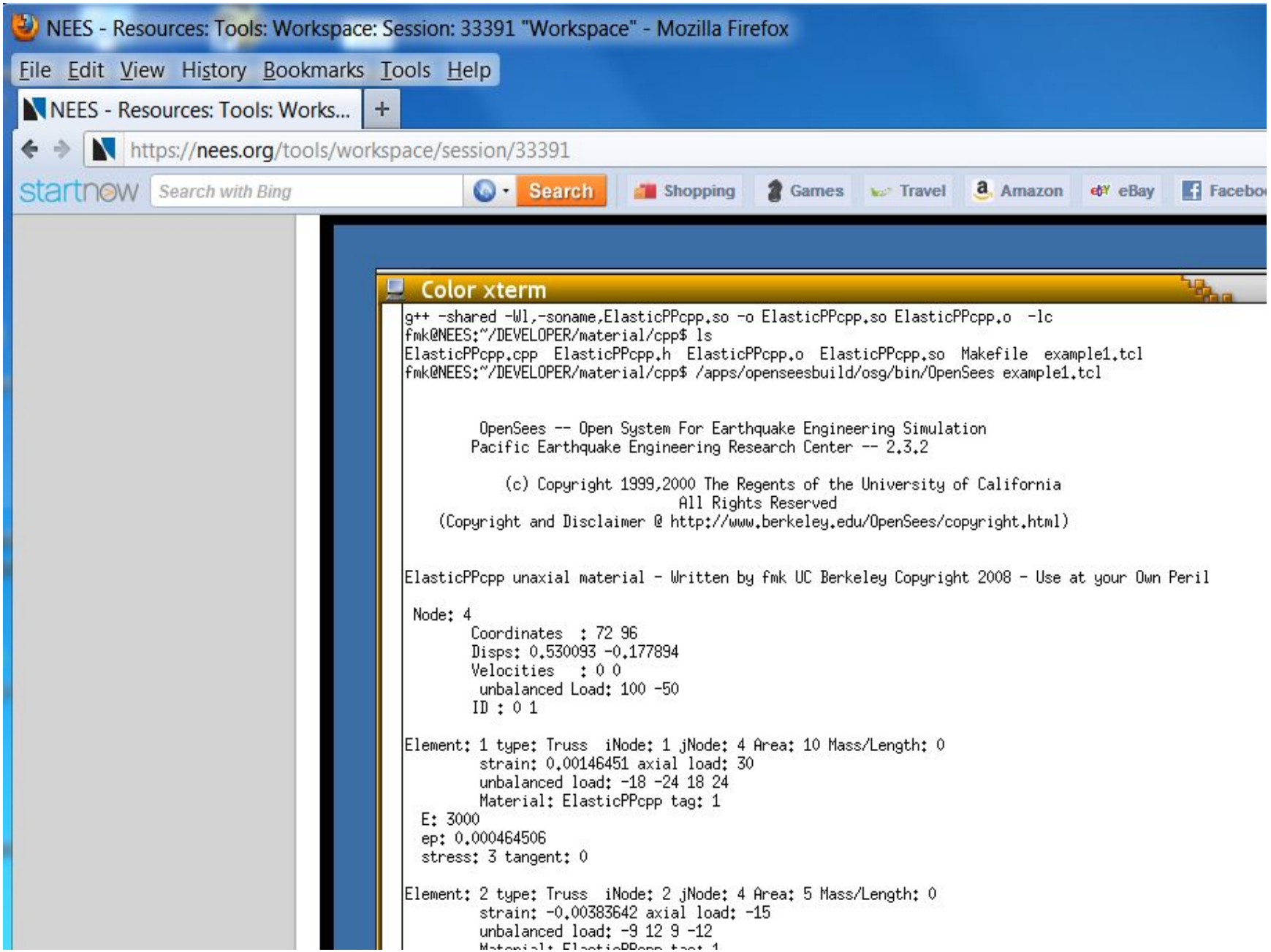
Outline of Workshop

- 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



Run It


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
Color xterm
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 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

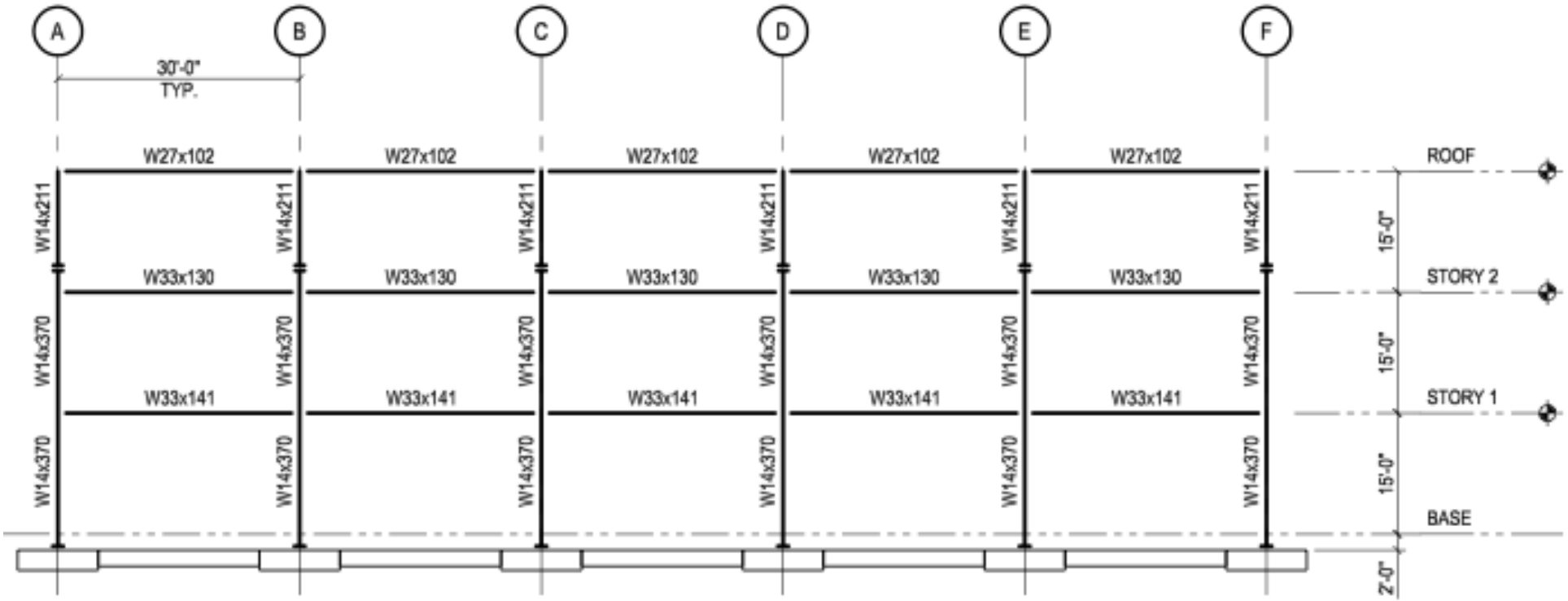
Outline of Workshop

- 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

- “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**”



MF ELEVATION ON LINE 1 (LINE 5 SIM.)

1"=20'

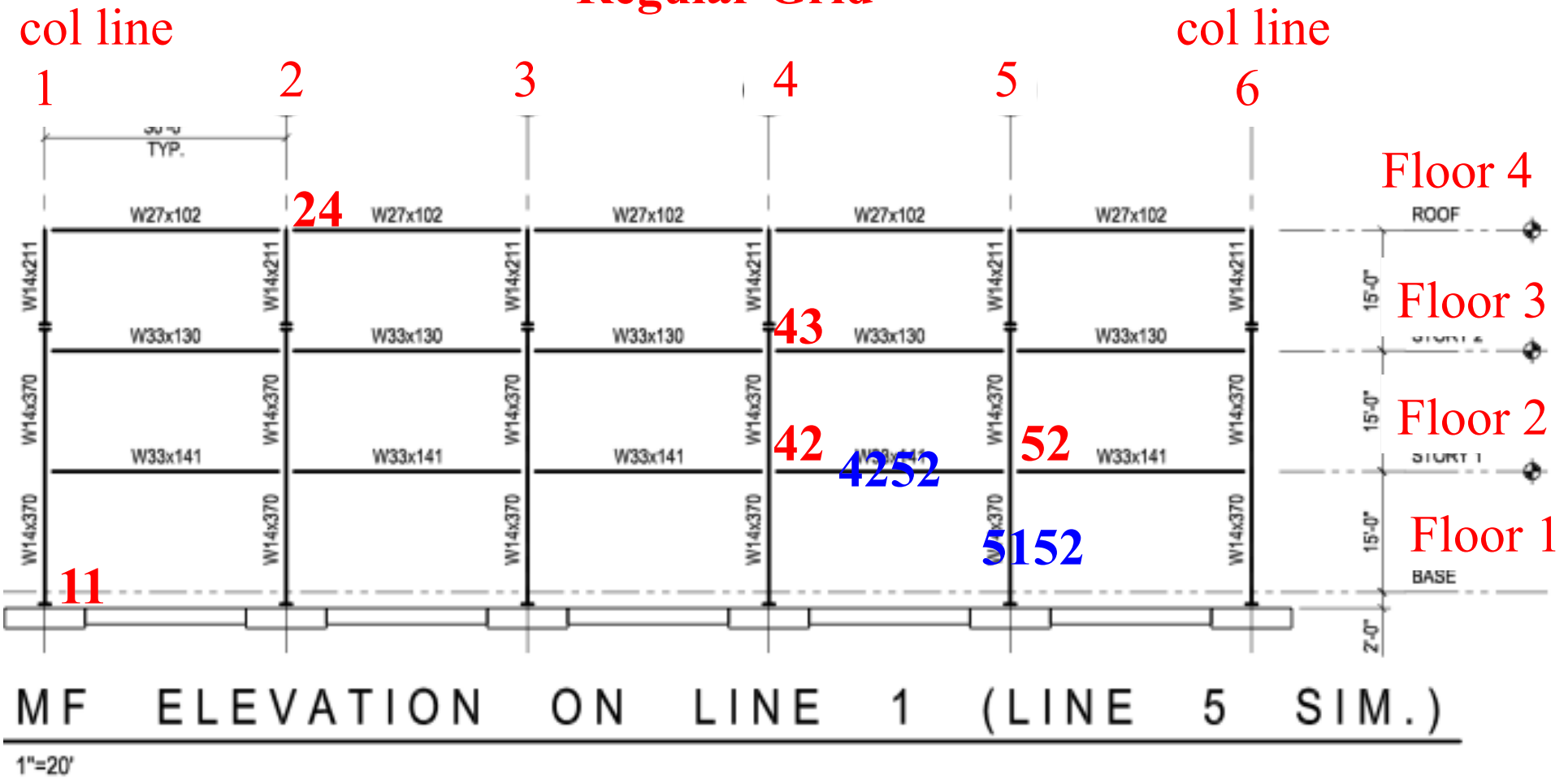
3 Story 5 Bay Moment Frame Original Code >350 lines

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

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

SO START PROGRAMMING!

Steel W Sections & Regular Grid



Nodes # colfloor

Elements # $iNode$jNode

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

MRF1.tcl

```
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 $cr1 $cr2
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
    }
}
```

Same model in 35 lines!

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
      }
   }
}
```

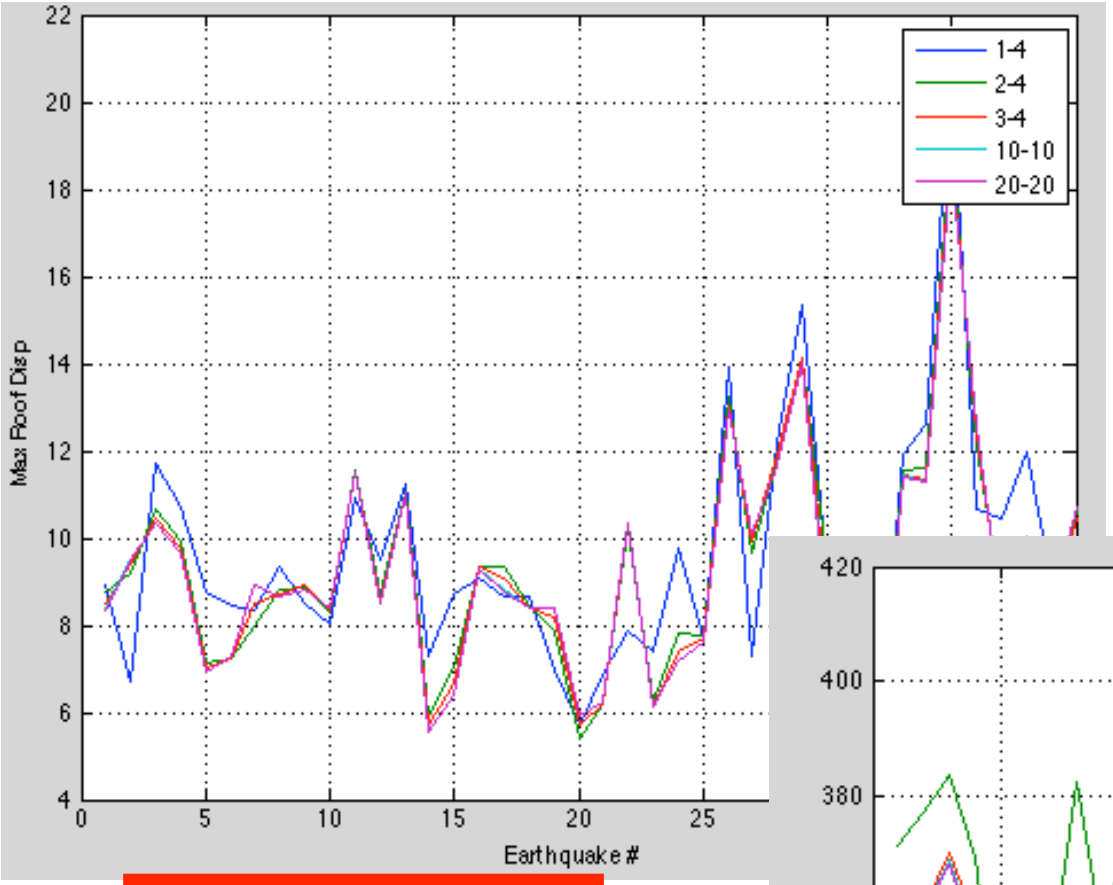
```
proc ForceBeamWSection2d {eleTag iNode jNode sectType matTag transfTag args} {

   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
}
```

Now that you don't have to spend a lot of time creating our models **you should spend time studying the effects of your modeling and analysis choices!**

Now We Can Do Fun Stuff
How Many Fibers?

MRF2.tcl

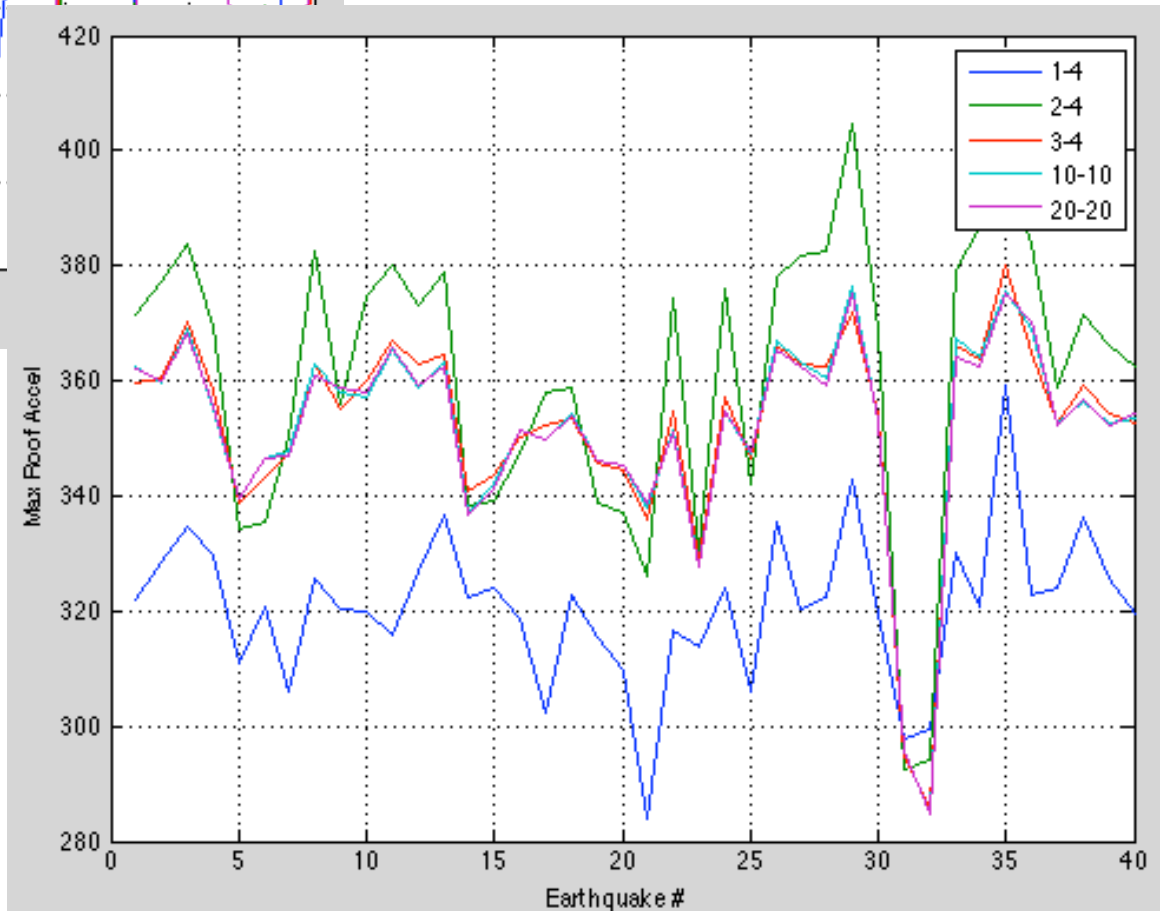
```
foreach nFlange {1 2 3 10 10 20 35} nWeb {4 4 4 5 10 20 30} {
    puts ”nFlange: $nFlange nWeb: $nWeb"

    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
    }
   }
}
```

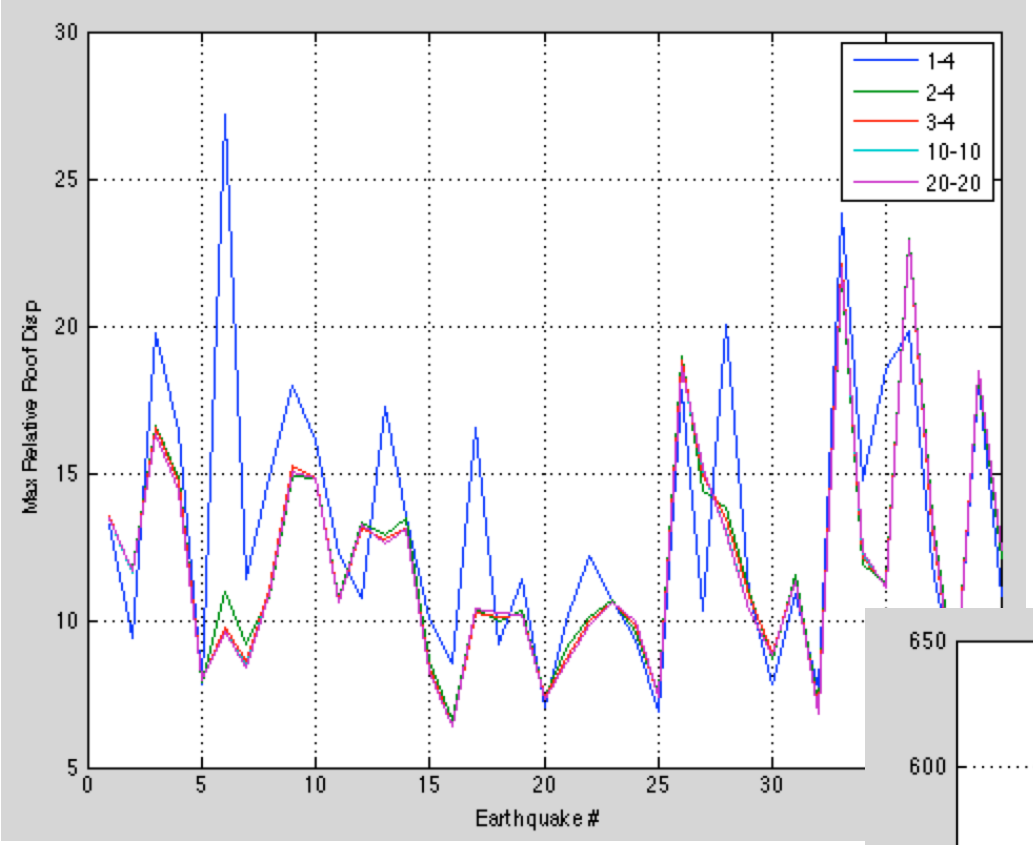


fibers flange- # fibers web

Results 10 % in 50year

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

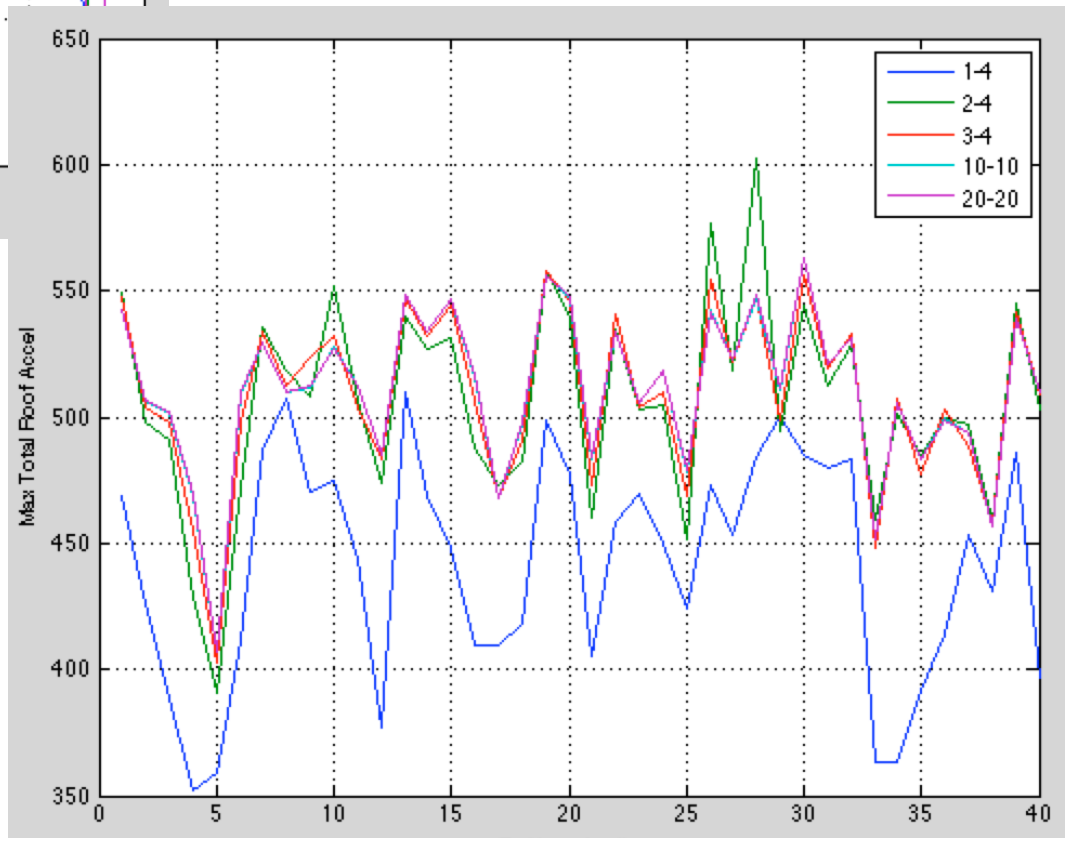




fibers flange- # fibers web

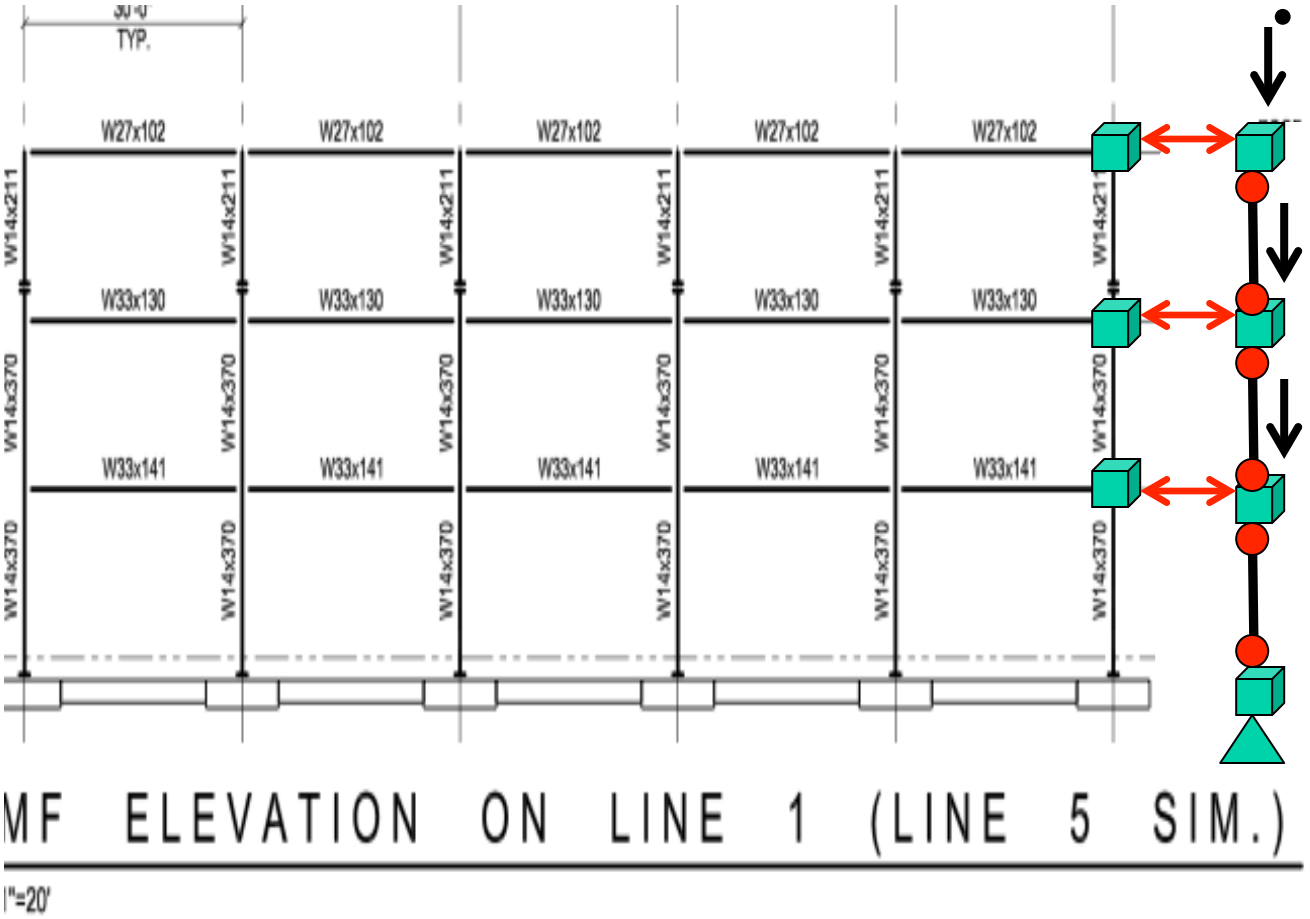
Results 2% in 50year

	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

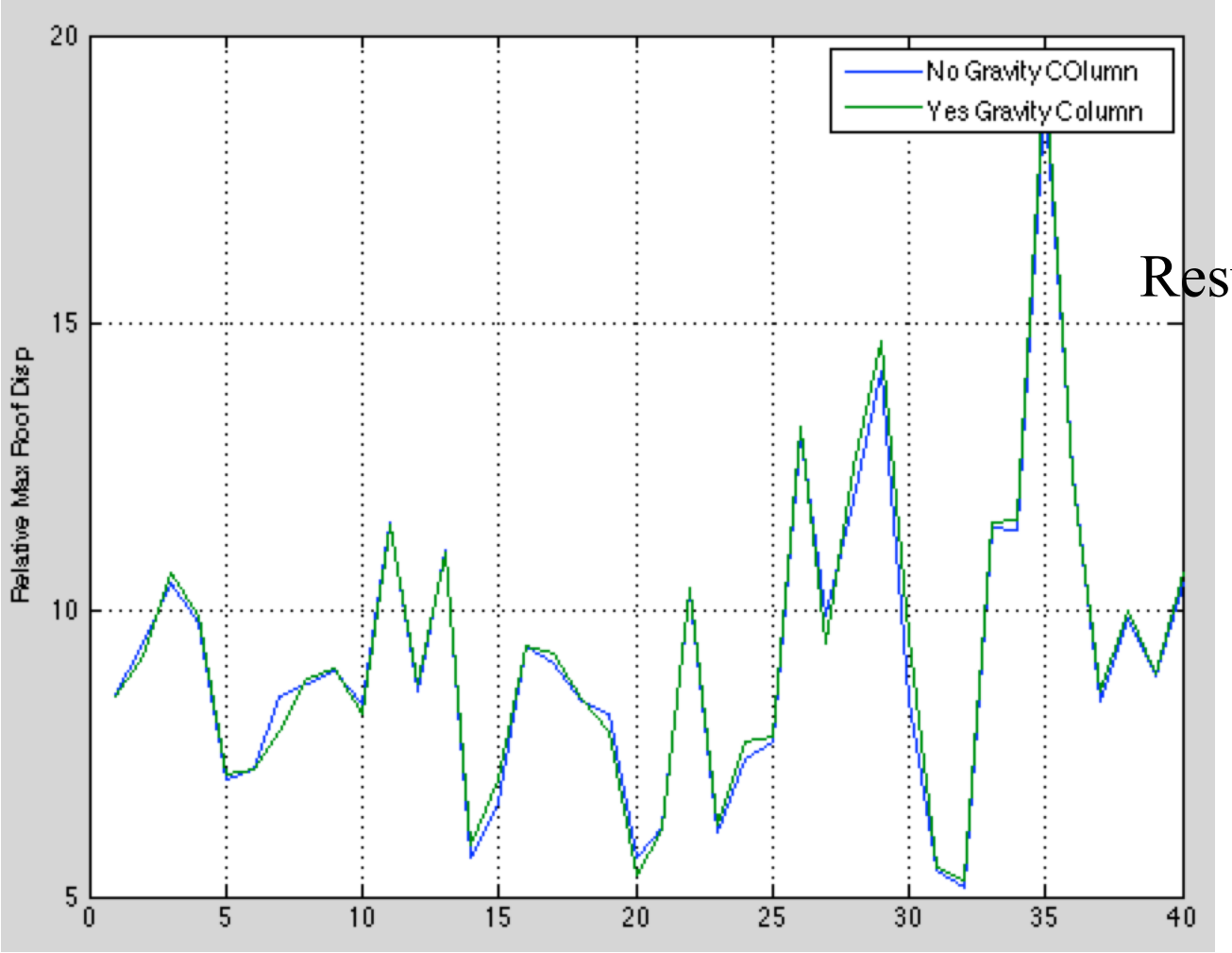
- Pinned base
- Gravity Column Loads
- Truss with Pdelta
- Tied to lateral system with MP Constraint

- 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}
 }

#elements
 foreach floor1 {1 2 3} floor2 { 2 3 4} {
      set A [lindex $gravityColsArea [expr $floor1-1]]
      element elasticBeamColumn 7$floor17$floor2 7$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
}

# 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
}
```

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

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



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)

MRF1.tcl

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

# set some lists containing floor and column line loca
set floorLocs {0. 204. 384. 564.};                  # floc
set colLocs {0. 360. 720. 1080. 1440. 1800.};  #colu
set massesX {0. 0.419 0.419 0.430};                  # ma
set massesY {0. 0.105 0.105 0.096} ;                 #   “

# add nodes at each floor at each column line location
foreach floor {1 2 3 4} floorLoc $floorLocs m
    foreach colLine {1 2 3 4 5 6} colLoc $coll
        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 $cr1 $cr2
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 a
set beamSizes {W33X141 W33X130 W27X102};  #beams sizes floor 1, 2,

# 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]]; # obtai
        ForceBeamWSection2d $colLine$floor1$colLine$floc
    }
}
#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
    }
}
```

I did not like having to calculate floor and col locations

While putting numbers in lists, e.g. 1 2 3 4 are easy to explain, changing model for some other configuation involves more work than necessary

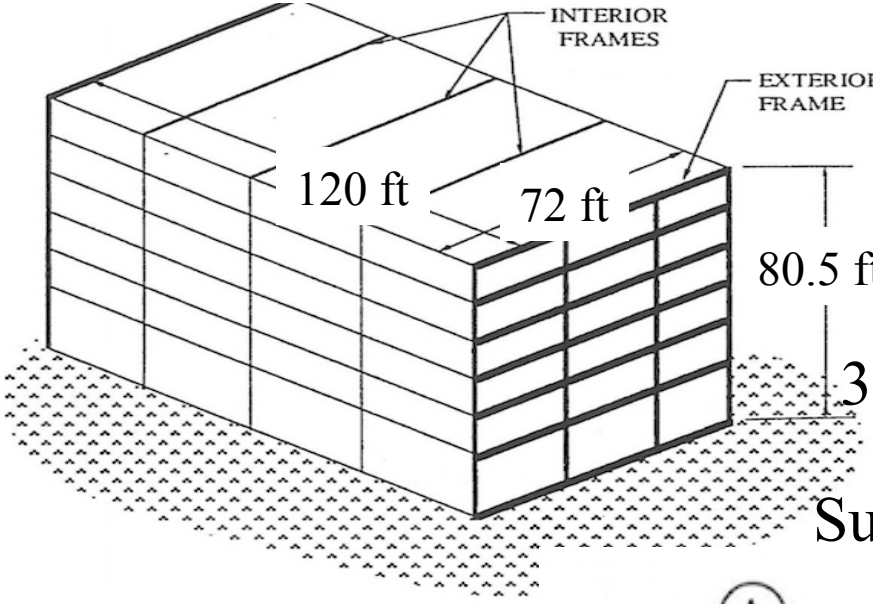
MRF3.tcl

```
source Steel2d.tcl
source ReadRecord.tcl;
# set some variables
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 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} ;
# set some calculated variables
set numFloor [expr [llength $floorOffsets]+1]
set numCline [expr [llength $colOffsets]+1]
set roofFloor [llength $numFloor]

wipe;
model BasicBuilder -ndm 2 -ndf 3
# build the nodes
for {set floor 1; set floorLoc 0} {$floor <= $numFloor} {incr floor 1} {
   set massX [lindex $massesX [expr $floor-1]]
   set massY [lindex $massesY [expr $floor-1]]
   for {set colLine 1; set colLoc 0;} {$colLine <= $numCline} {incr colLine 1} {
      node $colLine$floor $colLoc $floorLoc -mass $massX $massY 0.
      if {$floor == 1} {fix $colLine$floor 1 1 1}
      if {$colLine < $numCline} { set colLoc [expr $colLoc + [lindex $colOffsets [expr $colLine-1]]]}
   }
   if {$floor < $numFloor} { set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $floor-1]]]}
}
```

MRF3.tcl

```
# define material
#uniaxialMaterial Steel02 $tag $Fy $E $b $R0 $cr1 $cr2
uniaxialMaterial Steel02 1 50.0 29000 0.003 20 0.925 0.15

# build the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
    for {set floor1 1; set floor2 2} {$floor1 < $numFloor} {incr floor1 1; incr floor2 1} {
        set theSection [lindex $colSizes [expr $floor1 -1]]
        ForceBeamWSection2d $colLine$floor1$colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1 1
    }
}
#build the beams
geomTransf Linear 2
for {set colLine1  1; set colLine2 2} {$colLine1 < $numCline} {incr colLine1; incr colLine2 11} {
        for {set floor 2} {$floor <= $numFloor} {incr floor 1} {
          set theSection [lindex $beamSizes [expr $floor -2]]
         ForceBeamWSection2d $colLine1$floor$colLine2$floor $colLine1$floor $colLine2$floor $theSection 1
    }
}
```

Same model in 48 lines!



Max Roof Displacement = 5.6in

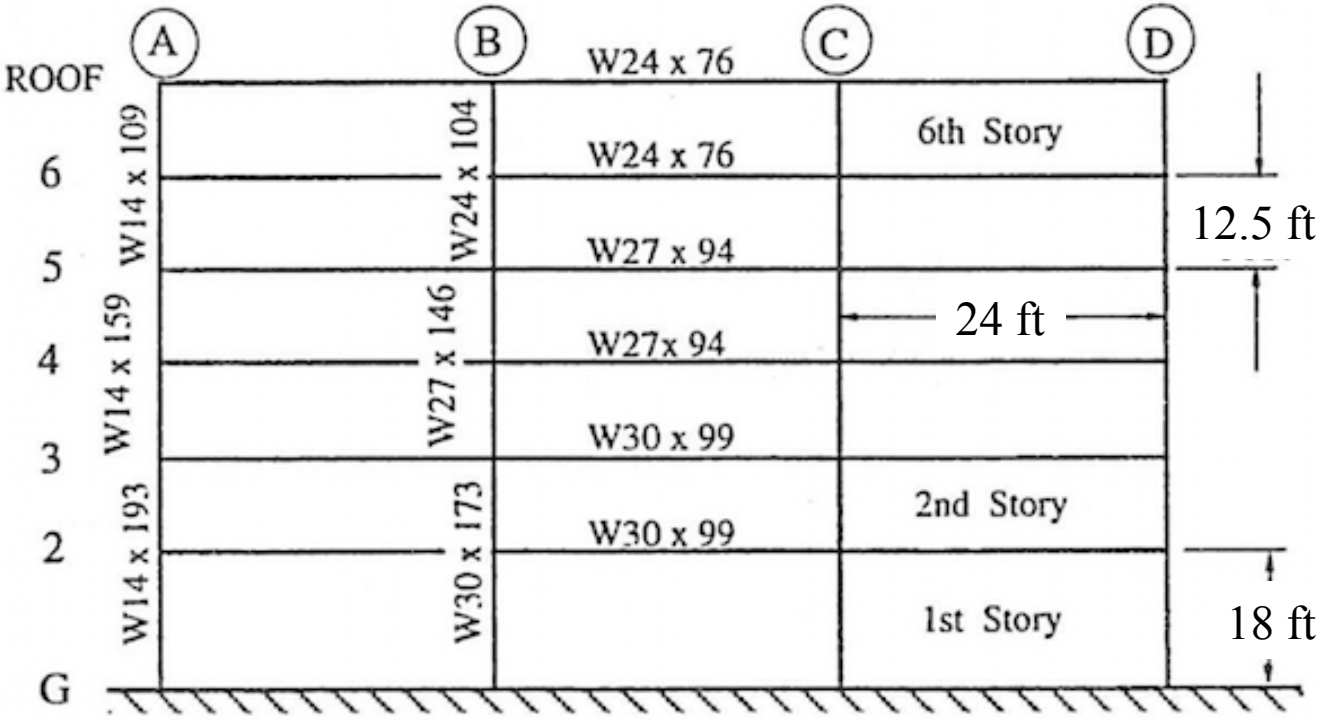
Dead Load: 95psf typical, roof 80psf

E=29,000, Fy=50.0, b =0.003

3% Rayleigh Damping 1st and 3rd Modes

Subjected to el_centro 1940 N-S (Peknold)

	A	Ixx
W14X193	56.8	2400
W14X159	46.7	1900
W14X109	32.0	1240
W30X173	51.0	8230
W27X146	43.1	5660
W24X104	30.6	3100
W30X99	29.1	3990
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"

Solution2014.tcl

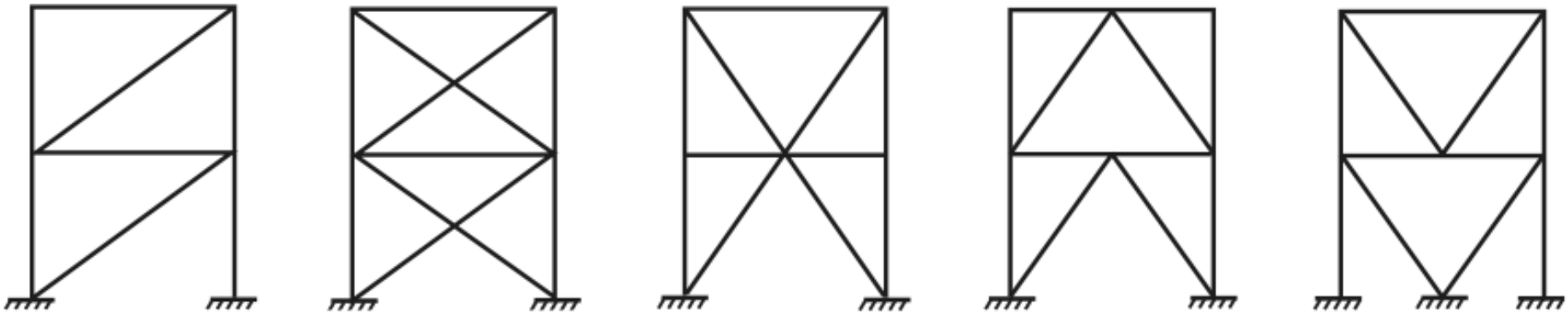
```
source Steel2d.tcl
source ReadRecord.tcl;

# set some variables
set motion  el_centro; set Fy 50.0; set E 29000; set b 0.003
set in 1.0;
set g 386.4;
set roofWeight  [expr 80*120.*72./1000.]; # 80psf * 120 long * 72 wide in kips
set floorWeight [expr 95*120.*72./1000.]; # 95psf * 120 long * 72 wide in kips
set numFrameResisting 2.0; # number lateral load resisting frames
set percentLoadFrame [expr 15./120.]; # percent load carried by lateral frame
# set up my lists
set floorOffsets  {216. 150. 150. 150. 150. 150.}
set colOffsets    {288. 288. 288.}
set colSizes      {W30X173 W30X173 W27X146 W27X146 W24X104 W24X104};
set colExtSizes  {W14X193 W14X193 W14X159 W14X159 W14X109 W14X109};
set beamSizes    {W30X99 W30X99 W27X94 W27X94 W24X76 W24X76};
#calculated properties
set numFloor [expr [llength $floorOffsets]+1]
set numCline [expr [llength $colOffsets]+1]
for {set i 0; set width 0;} {$i < [expr $numCline-1]} {incr i 1} {
   set width [expr $width + [lindex $colOffsets $i]
}
set massAtFloorNode [expr $floorWeight/($g*$numFrameResisting*$numCline*1.0)]
set massAtRoofNode [expr $roofWeight/($g*$numFrameResisting*$numCline*1.0)]
set uniformRoofLoad  [expr $roofWeight*$percentLoadFrame/$width]
set uniformFloorLoad [expr $floorWeight*$percentLoadFrame/$width]
```

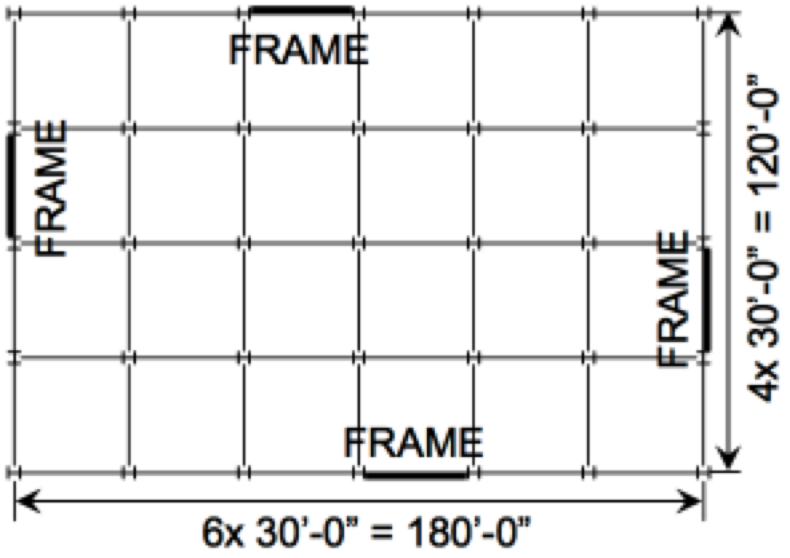
DON”T TOUCH BELOW LINE

Solution2014.tcl

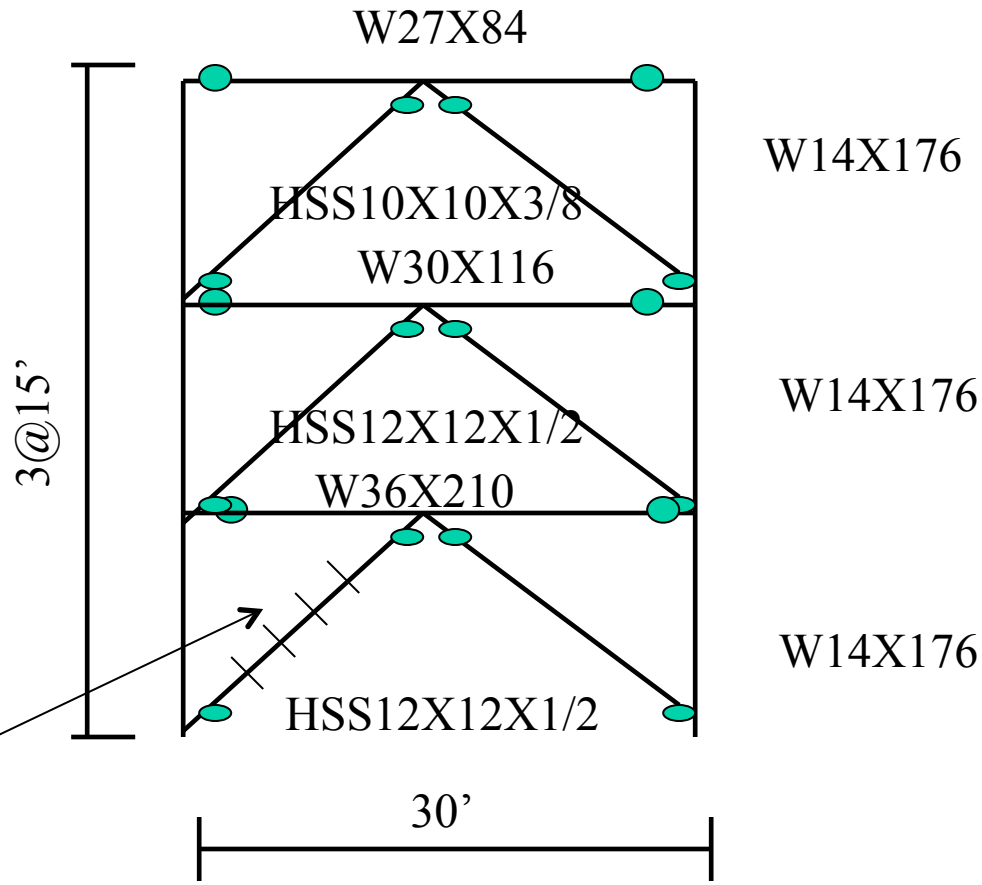
```
wipe;
model BasicBuilder -ndm 2 -ndf 3;
# build the nodes
for {set floor 1; set floorLoc 0} {$floor <= $numFloor} {incr floor 1} {
   if {$floor == $numFloor} {
      set mass $massAtRoofNode
   } else {
      set mass $massAtFloorNode
   }
   for {set colLine 1; set colLoc 0;} {$colLine <= $numCline} {incr colLine 1} {
      node $colLine$floor $colLoc $floorLoc -mass $mass $mass 0.
      if {$floor == 1} {fix $colLine$floor 1 1 1}
      if {$colLine < $numCline} {set colLoc [expr $colLoc + [lindex $colOffsets [expr $colLine-1]]]}
   }
   if {$floor < $numFloor} {set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $floor-1]]]}
}
# define material
uniaxialMaterial Steel02 1 $Fy $E $b 20 0.925 0.15
# build the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 1} {
   for {set floor1 1; set floor2 2} {$floor1 < $numFloor} {incr floor1 1; incr floor2 1} {
      if {$colLine == 1 || $colLine == $numCline} {
         set theSection [lindex $colExtSizes [expr $floor1 -1]]
      } else {
         set theSection [lindex $colSizes [expr $floor1 -1]]
      }
      ForceBeamWSection2d $colLine$floor1$colLine$floor2 $colLine$floor1 $colLine$floor2 $theSection 1
   }
}
```

Some Braced Frame EXAMPLES: Typical Configurations





- Beams pinned at column

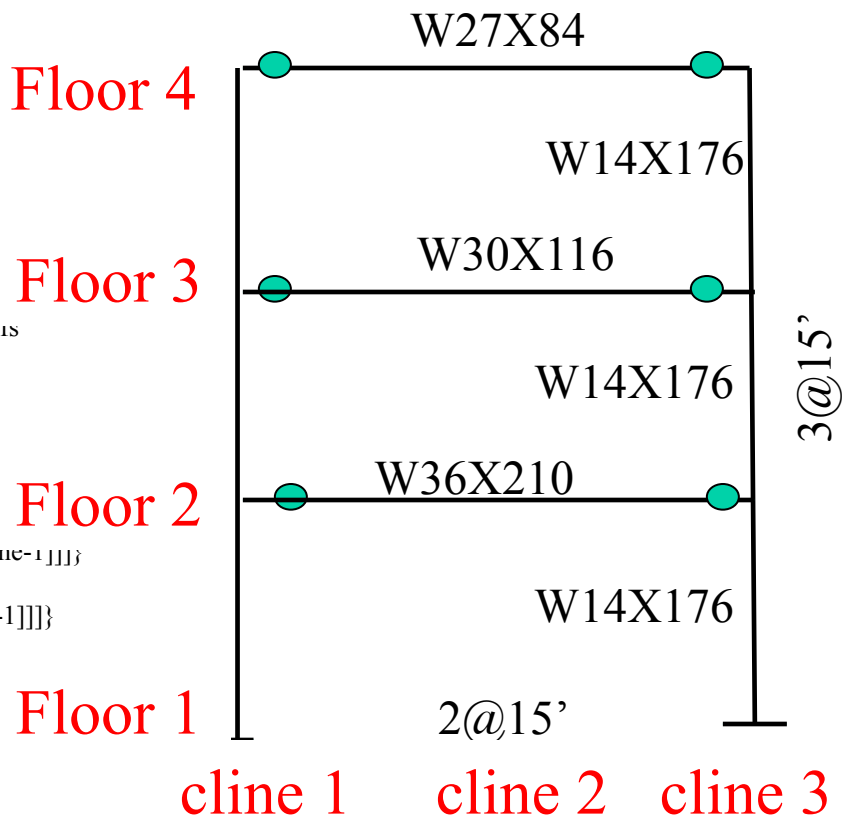


#number of beam elements pinned at either end
Imperfection so will buckle

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

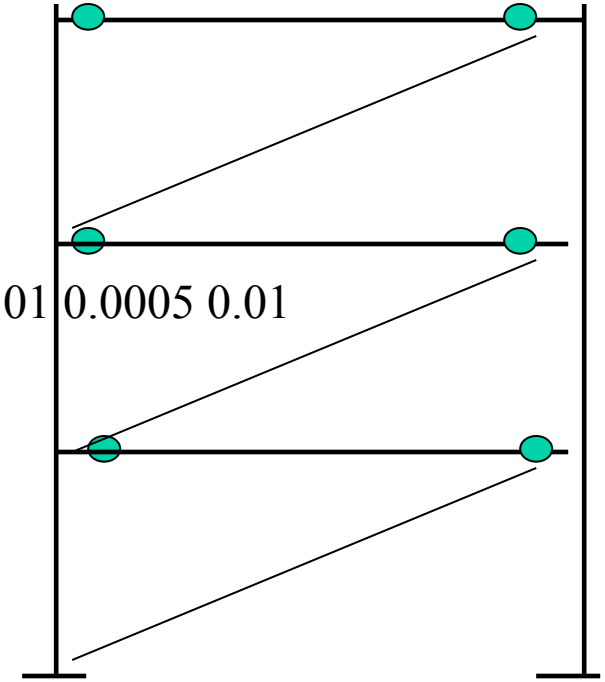
CBFbase.tcl

```
source Steel2d.tcl
# set up my structure
set in 1
set floorOffsets {180. 180. 180.}
set colOffsets   {180. 180.}
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}
# build colLocations and floorLocations & set some variables
set numFloor [expr [llength $floorOffsets]+1]
set numStory [expr $numFloor-1]
set numCline [expr [llength $colOffsets]+1]
wipe;
model BasicBuilder -ndm 2 -ndf 3;  # Define the model builder, ndm = #dimension, ndf = #dofs
# Build the Nodes
for {set floor 1; set floorLoc 0.} {$floor <= $numFloor} {incr floor 1} {
   set mass [lindex $masses [expr $floor-1]]
   for {set colLine 1; set colLoc 0.} {$colLine <= $numCline} {incr colLine 1} {
      node $colLine$floor $colLoc $floorLoc -mass $mass $mass 0.
      if {$floor == 1} { fix $colLine$floor 1 1 1}
      if {$colLine < $numCline} {set colLoc [expr $colLoc + [lindex $colOffsets [expr $colLine-1]]]}
   }
   if {$floor < $numFloor} {set floorLoc [expr $floorLoc + [lindex $floorOffsets [expr $floor-1]]]}
}
uniaxialMaterial Steel02 1 $Fy $Es $b 20 0.925 0.15 0.0005 0.01 0.0005 0.01
# add the columns
geomTransf PDelta 1
for {set colLine 1} {$colLine <= $numCline} {incr colLine 2} {
   for {set floor1 1; set floor2 2} {$floor1 < $numFloor} {incr floor1 1; incr floor2 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 -release1
   DispBeamWSection2d $colLine2$floor$colLine3$floor $colLine2$floor $colLine3$floor $theSection 1 2 -release2
}
```



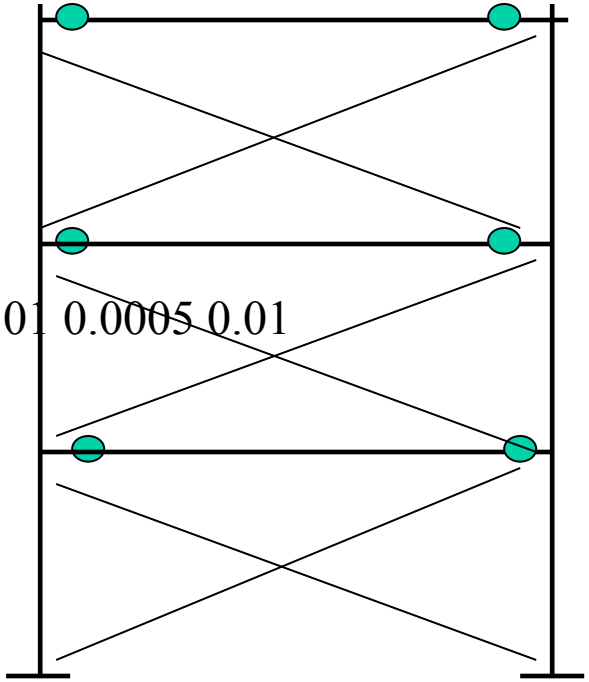
}

CBF1.tcl
(Diagonal bracing)



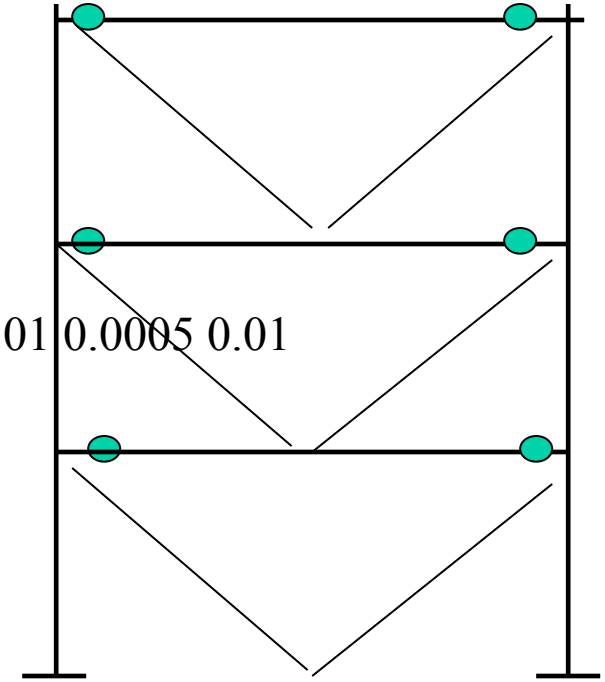
```
source CBFbase.tcl

# 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 CBFbase.tcl

# 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 CBFbase.tcl

# 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.