Development of Modeling Tools for OpenSees

Silvia Mazzoni
University of California, Berkeley

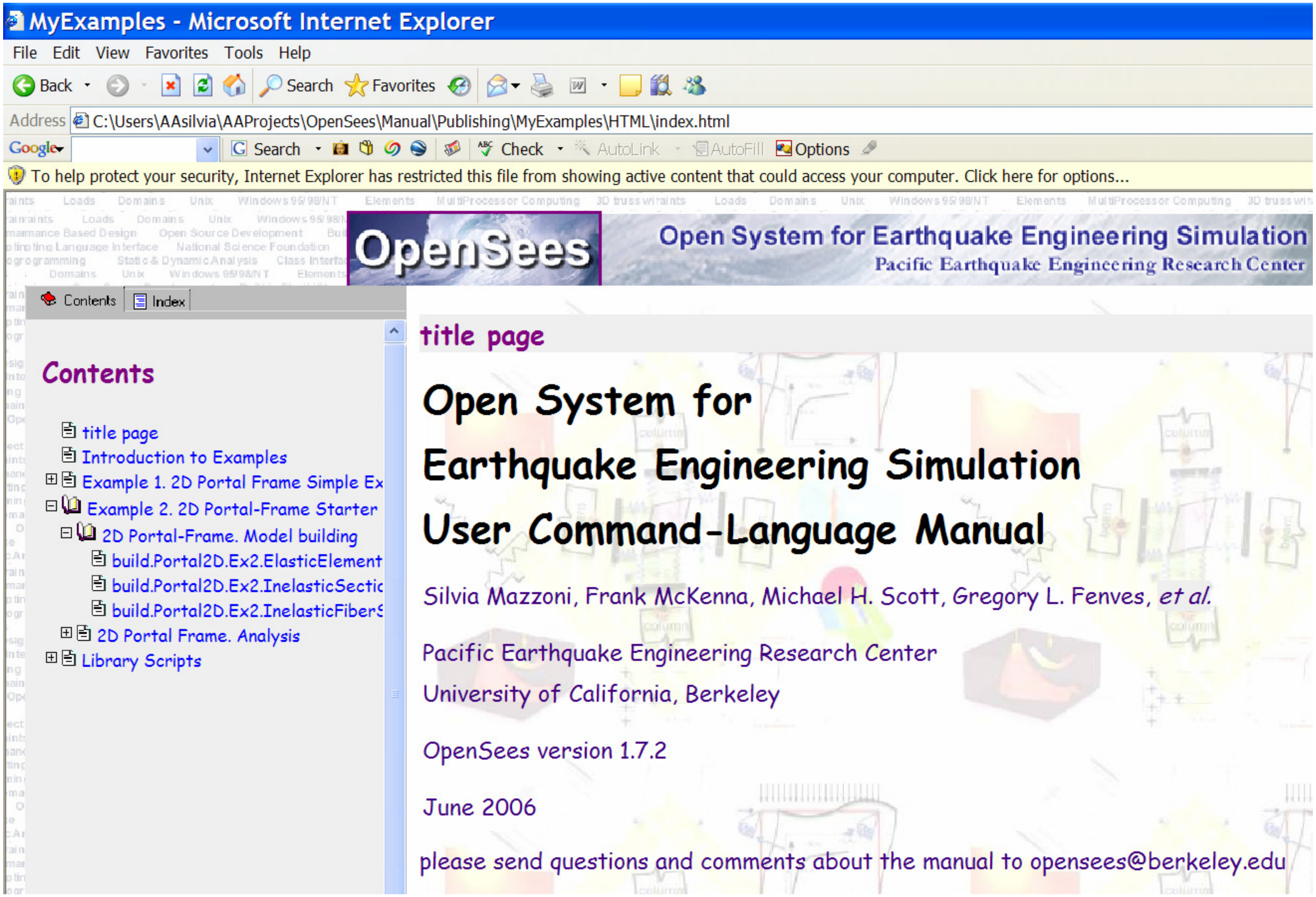
OpenSees Developer Symposium

16 August 2006

User-Support Activities

- Direct User Support (email + forum)
- Annual User/Developer Workshops
- Maintain Command-Language Manual
- Develop Examples Manual
- Develop Scripting Tools
- Comparison of OpenSees Models

Examples Manual (in progress)



Example -Model Building

- Elastic Element
- Inelastic Section
- Inelastic Fiber Section

Example -Analyses

- Static Pushover
- Static Reversed Cyclic
- Dynamic Unidirectional Uniform Excitation
  - Sine wave
  - EQ ground motion
- Dynamic Multiple-Support Excitation
  - Sine wave
  - EQ ground motion
- Dynamic Bidirectional Uniform Excitation
  - EQ ground motion

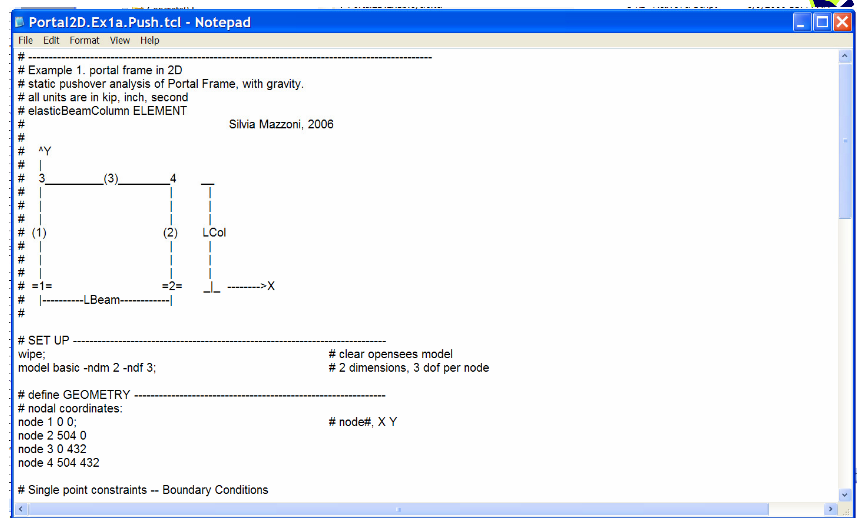
Example -Utility Scripts

Examples Manual (contents)

Example 1. 2D Portal Frame Simple Examples
- Portal2D.Ex1a.Push.tcl
- Portal2D.Ex1b.cycle.tcl
- Portal2D.Ex1c.EQ.tcl

Example 2. 2D Portal-Frame Starter Examples
- 2D Portal-Frame. Model building
  - build.Portal2D.Ex2.ElasticElement.tcl
  - build.Portal2D.Ex2.InelasticSection.tcl
  - build.Portal2D.Ex2.InelasticFiberSection.tcl
- 2D Portal Frame. Analysis
  - analyze.Portal2D.Ex2.Static.Push.tcl
  - analyze.Portal2D.Ex2.Static.Cycle.tcl
  - analyze.Dynamic.EQ.tcl
  - analyze.Dynamic.sine.Uniform.tcl
  - analyze.Portal2D.Ex2.Dynamic.EQ.multipleSupport.tcl
  - analyze.Portal2D.Ex2.Dynamic.sine.multipleSupport.tcl
  - analyze.Portal2D.Ex2.Dynamic.EQ.bidirect.tcl

Example 3. 2D Structural-Steel Frame Examples
- 2D Steel Frame. Model building
  - build.Frame2D.ex3.ElasticSection.tcl
  - build.Frame2D.ex3.InelasticFiberSection.tcl
- 2D Steel Frame. Analysis
  - analyze.Frame2D.ex3.Static.Push.tcl
  - analyze.Frame2D.ex3.Static.Cycle.tcl
  - analyze.Dynamic.EQ.tcl
  - analyze.Frame2D.ex3.Dynamic.EQ.multipleSupport.tcl……………………


Portal2D.Ex1a.Push.tcl - Notepad

```
# ----------------------------------------------------------------------------------------
# Example 1. portal frame in 2D
# static pushover analysis of Portal Frame, with gravity.
# all units are in kip, inch, second
# elasticBeamColumn ELEMENT
#                                         Silvia Mazzoni, 2006
#
#    ^Y
#    |
#    3_________(3)________4       __
#    |                    |        |
#    |                    |        |
#    |                    |        |
#   (1)                  (2)      LCol
#    |                    |        |
#    |                    |        |
#    |                    |        |
#   =1=                  =2=      _|_  -------->X
#    |----------LBeam------------|
#

# SET UP ----------------------------------------------------------------------------
wipe;                                        # clear opensees model
model basic -ndm 2 -ndf 3;                   # 2 dimensions, 3 dof per node

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

# Single point constraints -- Boundary Conditions
```

Utility Scripts
- LibReadSMDFile.tcl
- LibGenPeaks.tcl
- LibDisplay.tcl

Example - Model Building

```
# --------------------------------------------------------------------------------------------------
# buildPortal2D.tcl: generates nodes/materials/sections/elements
# elasticBeamColumn element
#                                                          Silvia Mazzoni, 2006
#
#    ^Y
#    |
#    3_________(3)________4      __
#    |                    |      |
#    |                    |      |
#    |                    |      |
#  (1)                  (2)    LCol
#    |                    |      |
#    |                    |      |
#    |                    |      |
#  =1=                  =2=    _|_  -------->X
#    |----------LBeam------------|
#
# SET UP ----------------------------------------------------------------------------
wipe;                                        # clear memory of all past model definitions
model BasicBuilder -ndm 2 -ndf 3;            # Define the model builder, ndm=#dimension, ndf=#dofs
………………..
# define UNITS ----------------------------------------------------------------------------
set in 1.;                                   # define basic units -- output units
set kip 1.;                                  # define basic units -- output units
set sec 1.;                                  # define basic units -- output units
set ft [expr 12.*$in];                       # define engineering units
……………………..
# define GEOMETRY -------------------------------------------------------------
set LCol [expr 36*$ft];                      # column length
set LBeam [expr 42*$ft];                     # beam length
…………………..
# calculated parameters ------------------------------------------------------------
set PCol [expr $Weight/2];                   # nodal dead-load weight per column
set Mass [expr $PCol/$g];                    # nodal mass
……….
# nodal coordinates: ------------------------------------------------------------
node 1 0 0;                                  # node#, X, Y
node 2 $LBeam 0
……………………..
```

Static Analysis

```
# --------------------------------    perform Static Pushover Analysis
set Nsteps [expr int($Dmax/$Dincr)];      # number of pushover analysis steps
set ok [analyze $Nsteps];            # this will return zero if no convergence problems
     were encountered
```

```
# --------------------------------    perform Static Cyclic Displacements Analysis
source LibGenPeaks.tcl;           # source in a proc to gen. disp. increments for a specified peak
      foreach Dmax $iDmax {
        set iDstep [procGenPeaks $Dmax $Dincr $CycleType $Fact];      # set up peak
        for {set i 1} {$i <= $Ncycles} {incr i 1} {
            set zeroD 0
            set D0 0.0
            foreach Dstep $iDstep {
                  set D1 $Dstep
                  set Dincr [expr $D1 - $D0]
                  integrator DisplacementControl  $IDctrlNode   $IDctrlDOF $Dincr; # -------------
---------------------first analyze command----------------------
                  set ok [analyze 1]
```

Dynamic Uniform Excitation Analyses

Start load-pattern defintion

```
#  ---------------------------------    perform Dynamic Ground-Motion Analysis
    # the following commands are unique to the Uniform Earthquake excitation
    set IDloadTag 400;  # for uniformSupport excitation
    # read a PEER strong motion database file, extracts dt from the header and converts the file
    # to the format OpenSees expects for Uniform/multiple-support ground motions
    source LibReadSMDFile.tcl;        # read in procedure Multinition
```

unidirectional

```
# Uniform EXCITATION: acceleration input
    set inFile $GMdir$GMfile.at2
    set outFile $GMdir$GMfile.g3;
    procReadSMDFile $inFile $outFile dt;          # call procedure to convert the ground-motion file
    set GMfatt [expr $g*$GMfact];                 # data in input file is in g Unifts -- ACCELERATION TH
    set AccelSeries "Series -dt $dt -filePath $outFile -factor  $GMfatt";          # time series information
    pattern UniformExcitation  $IDloadTag  $GMdirection -accel  $AccelSeries  ;    # load pattern
```

multidirectional

```
# Uniform EXCITATION: acceleration input
foreach GMdirection $iGMdirection GMfile $iGMfile GMfact $iGMfact {
    incr IDloadTag;
    set inFile $GMdir$GMfile.at2
    set outFile $GMdir$GMfile.g3;
    procReadSMDFile $inFile $outFile dt;          # call procedure to convert the ground-motion file
    set GMfatt [expr $g*$GMfact];                 # data in input file is in g Unifts -- ACCELERATION TH
    set AccelSeries "Series -dt $dt -filePath $outFile -factor  $GMfatt";          # time series information
    pattern UniformExcitation  $IDloadTag  $GMdirection -accel  $AccelSeries  ;    # load pattern
}
```

analyze

```
set Nsteps [expr int($TmaxAnalysis/$DtAnalysis)];
set ok [analyze $Nsteps $DtAnalysis];
```

MultipleSupport Excitation

```
#  -------------------------------    perform Dynamic Ground-Motion Analysis
      # the following commands are unique to the Multiple-Support Earthquake excitation
      set IDloadTag 400;
      set IDgmSeries 500;                # for multipleSupport Excitation
      # read a PEER strong motion database file, extracts dt from the header and converts the
      # file to the format OpenSees expects for Uniform/multiple-support ground motions
      source LibReadSMDFile.tcl;         # read in procedure Definition
# multiple-support excitation: displacement input at individual nodes
foreach SupportNode $iSupportNode GMfile $iGMfile GMfact $iGMfact GMdirection $iGMdirection {
      incr IDloadTag;
      incr IDgmSeries;
      set inFile $GMdir$GMfile.dt2
      set outFile $GMdir$GMfile.g3;
      procReadSMDFile $inFile $outFile dt;          # call procedure to convert the ground-motion file
      set GMfatt [expr $cm*$GMfact];                # data in input file is in cm Units -- DISPLACEMENT TH
      set DispSeries "Series -dt $dt -filePath $outFile -factor  $GMfatt";    # time series information
      pattern MultipleSupport $IDloadTag  {
                groundMotion $IDgmSeries Plain -disp  $DispSeries
                imposedSupportMotion $SupportNode  $GMdirection $IDgmSeries
      };        # end pattern
}
```

OpenSees Script Library

Utilities procs

- **LibGlobalVariables**: Global Variables for procs
- **LibUnits**: System of Units
- **LibAnalysisStatic**: Static-Analysis Procs
- **LibAnalysisDynamic**: Dynamic-Analysis Procs
- **LibDisplay**: Display Procs
- **LibModelBuilding**: Model-Building procs
- **LibMomCurv**: Moment-Curvature Procs
- **LibsectionRC**: RC-Section procs
- **LibsectionW**: W-Section proc
- **LibGeneralProcs**: Useful general-purpose procs
- **LibGMfiles**: Ground-Motion Filenames

Materials Definitions

- **LibMaterialsRC**: baseline RC materials
- **LibMaterialsRCVariations**: variations on RC materials
- **LibMaterialsPinching4**: pinching4 material
- **LibMaterialsSS**: structural-steel materials

Libraries.tcl

```
##############################################################################
# Libraries.tcl -- set up script-tools libraries
#              Silvia Mazzoni, 2006
##############################################################################
##############################################################################

variable LibDir C:/Users/AAsilvia/AAProjects/OpenSees/_TclScriptLibrary

source $LibDir/LibGlobalVariables.tcl;          # Define Global Variables for Analysis etc.

source $LibDir/LibUnits.tcl;                    # Define System of Units
source $LibDir/LibAnalysisStatic.tcl;           # Static-Analysis Procs
source $LibDir/LibAnalysisDynamic.tcl;          # Dynamic-Analysis Procs
source $LibDir/LibDisplay.tcl;                  # Display Procs
source $LibDir/LibGMfiles.tcl;                  # Ground-Motion Filenames
source $LibDir/LibModelBuilding.tcl;            # Model-Building procs
source $LibDir/LibMomCurv.tcl;                  # Moment-Curvature Procs
source $LibDir/LibsectionRC.tcl;                # RC-Section procs
source $LibDir/LibsectionW.tcl;                 # W-Section proc
source $LibDir/LibGeneralProcs.tcl;             # Useful general-purpose procs

# other library files, to be exectuted within the input script (after model is built)
#source $LibDir/LibMaterialsRC.tcl;             # Materials Definitions -- baseline RC materials
#source $LibDir/LibMaterialsRCVariations.tcl;   # Materials Definitions -- variations on baseline RC materials
#source $LibDir/LibMaterialsPinching4.tcl;      # Materials Definitions -- pinching4 material
#source $LibDir/LibMaterialsSS.tcl;             # Materials Definitions -- structural-steel materials
```

LibGlobalVariables.tcl

```
# These are global variables which can be referenced from within a proc by typing: global VariableName

# some general information:
  variable problemSize Large;          # option, Large or Small (less then 10 nodes)
  variable LeaningColumn yes;          # options  yes                    no
  variable DLType Concentrated;        # Dead Load: " Distributed " along elements, or  " Concentrated " at nodes
  variable SaveDatabase "off";         # save to database trigger "on"  saves at the end of each cycle

# define UNITS (LibUnits.tcl)
# The user can specify the basic units ($BasicUnitType) using three components:
# set BasicUnitType FLT;  # define Force, Length, and Time units using Funit,Lunit & Tunit, respectively.
#    (default, $BasicUnitType does not need to be specified)
# set BasicUnitType MLT;  # define Mass, Length, and Time units using Funit,Lunit & Tunit, respectively.
# Available Basic Units:
# Length, Lunits: in, inch, ft, meter, m, cm
# Force, Funits: lbf, kip, kgf, N, Newton, kg, lb  --   kg=kgf and lb=lbf when forces are the basic units (FLT)
# Mass, Munits: lbm, kgm, kg, lb  --  kg=kgm and lb=lbm when masses are the basic units (MLT)
# Time, Tunits: sec
  variable BasicUnitType FLT;          # define unit system by Force or Mass, Length and Time
  variable Lunit in;                   # set Length units
  variable Funit kip;                  # set Force units
  variable Tunit sec;                  # set Time units

# LOAD-TAG variables
  variable loadIDgravity 100;          # gravity load ID tag
  variable loadIDstatic 200;           # static load ID tag
  variable IDloadTagGMA 310;           # ground-motion load ID tag
  variable IDloadTagGMB 320;           # ground-motion load ID tag
  variable IDloadTagGMC 330;           # ground-motion load ID tag
  variable IDgmSeries 350;             # ground-motion series ID tag
```

LibGlobalVariables.tcl (cont.)

```
# CONSTRAINTS handler -- Determines how the constraint equations are enforced in the analysis
    (http://opensees.berkeley.edu/OpenSees/manuals/usermanual/617.htm)
#       Plain Constraints -- Removes constrained degrees of freedom from the system of equations
#       Lagrange Multipliers -- Uses the method of Lagrange multipliers to enforce constraints
#       Penalty Method -- Uses penalty numbers to enforce constraints
#       Transformation Method -- Performs a condensation of constrained degrees of freedom
    variable constraintsTypeGravity Plain;              # options: Plain, Penalty, Lagrange, Transformation
    variable constraintsTypeStatic Plain;               # options: Plain, Penalty, Lagrange, Transformation
    variable constraintsTypeDynamic Transformation;     # options: Plain, Penalty, Lagrange, Transformation
    variable alphaSP 1e6 ; # Penalty/Lagrange constraints -- factor adding the single-point constraint into the SOE
    variable alphaMP 1e6 ;# Penalty/Lagrange constraints -- factor adding the multi-point constraint into the SOE

# DOF NUMBERER (number the degrees of freedom in the domain):
    (http://opensees.berkeley.edu/OpenSees/manuals/usermanual/366.htm)
#   determines the mapping between equation numbers and degrees-of-freedom
#       Plain -- Uses the numbering provided by the user
#       RCM -- Renumbers the DOF to minimize the matrix band-width using the Reverse Cuthill-McKee algorithm
    variable numbererType Plain;    # options: Plain, RCM

# Solution ALGORITHM: -- Iterate from the last time step to the current
    (http://opensees.berkeley.edu/OpenSees/manuals/usermanual/682.htm)
#       Linear -- Uses the solution at the first iteration and continues
#       Newton -- Uses the tangent at the current iteration to iterate to convergence
#       ModifiedNewton -- Uses the tangent at the first iteration to iterate to convergence
#       NewtonLineSearch -- #      KrylovNewton -- #       BFGS -- #       Broyden --
    variable algorithmTypeGravity Newton;
    variable algorithmTypeStatic Newton;
    variable algorithmTypeDynamic Newton;
    variable NewtonLineSearchRatio 0.8;        # Algorithm: NewtonLineSearch, limiting ratio between the residuals
    before and after the incremental update (between 0.5 and 0.8)
    variable algorithmCount 5;      # Algorithm: BFGS/Broyden, number of iterations within a time step until a new
    tangent is formed
```

LibGlobalVariables.tcl (cont.)

```
…

# DISPLAY variables (0=upper left-most corner)
 variable xPixels 600;                          # height of graphical window in pixels
 variable yPixels 400;                          # height of graphical window in pixels
 variable xLoc1 10;                             # horizontal location of graphical window
 variable yLoc1 10;                             # vertical location of graphical window
 variable xLoc2 $xLoc1;                         # horizontal location of graphical window
 variable yLoc2 [expr $yLoc1+$yPixels];         # vertical location of graphical window
 variable xLoc3 [expr $xLoc1+$xPixels];         # horizontal location of graphical window
 variable yLoc3 $yLoc1;                         # vertical location of graphical window

…
```

LibUnits.tcl

```
# ########################################################################################
# LibUnits.tcl -- define system of units used in the tcl script
# OpenSees output will be in the basic units
#                                                        by Silvia Mazzoni, 2006
#
# The user can specify the basic units ($BasicUnitType) using three components:
# set BasicUnitType FLT;  # define Force, Length, and Time units using Funit,Lunit & Tunit, respectively.
#    (This is the default case and $BasicUnitType does not need to be specified)
# set BasicUnitType MLT;  # define Mass, Length, and Time units using Funit,Lunit & Tunit, respectively.
# Available Basic Units: -- these are global variables which can be referenced from within a proc by typing: global name
# Length, Lunits: in, inch, ft, meter, m, cm
# Force, Funits: lbf, kip, kgf, N, Newton, kg, lb  --   kg and lb default to kgf and lbf when forces are the basic units (FLT)
# Mass, Munits: lbm, kgm, kg, lb  --  kg and lb default to kgm and lbm when masses are the basic units (MLT)
# Time, Tunits: sec
# Additional Units Defined within, which can be used in the scripts:
# Constants: g, PI, pi, Ubig Usmall
# mixed: psi, ksi, Pa, MPa, pcf, psf, in2, in4, cm2, m2
###example input:
# 1. Define Mass-Length-Time in Imperial units
# set BasicUnitType MLT;             # define unit system by Force or Mass, Length and Time
# set Lunit in;       # set Length units
# set Munit lbm;   # set Mass units
# set Tunit sec;     # set Time units
# source $LibDir/LibUnits.tcl;
# 2. Define Mass-Length-Time in SI units
# set BasicUnitType MLT;             # define unit system by Force or Mass, Length and Time
# set Lunit meter; # set Length units
# set Munit kgm;   # set Mass units
# set Tunit sec;     # set Time units
# source $LibDir/LibUnits.tcl;
# 3. Define Force-Length-Time in Imperial units (default, so you don't need to specify BasicUnitType)
# set BasicUnitType FLT;             # define unit system by Force or Mass, Length and Time
# set Lunit in;       # set Length units
# set Funit kip;     # set Force units
# set Tunit sec;     # set Time units
# source $LibDir/LibUnits.tcl;
# 2. Define Force-Length-Time in SI units (default, so you don't need to specify BasicUnitType)
# set BasicUnitType FLT;             # define unit system by Force or Mass, Length and Time
# set Lunit cm;       # set Length units
# set Funit kgf;     # set Force units
# set Tunit sec;     # set Time units
# source $LibDir/LibUnits.tcl;
# --------
# example in proc:
# proc sample {} {
#       global lbf;                    # load global units variable
#       puts $lbf;
# }
```

LibUnits.tcl

```
if {$Lunit=="in" | $Lunit=="inch" } {
	variable in  1.;				# define basic units -- length
	variable LunitTXT "inch";			# define text for screen output
	variable ft  [expr 12.*$in];			# define engineering units
	variable cm  [expr $in/2.54];			# scaling factor for SI centimeter unit
	variable meter [expr $cm*100];
} elseif {$Lunit=="ft" } {
	variable ft  1.;				# define basic units -- length
	variable LunitTXT "ft";
	variable in  [expr $ft/12.];
	variable cm  [expr $in/2.54];
	variable meter [expr $cm*100];		…………
}
# additional units
variable m     $meter;				# another name for it
variable inch $in;				# another name for it
variable Newton $N;			# another name for it
variable Pa   [expr $N/$meter/$meter];			# pascals
variable MPa  [expr 1.e6*$Pa];			# megaPascals
variable ksi  [expr $kip/pow($in,2)];
variable psi  [expr $ksi/1000.];
variable pcf  [expr $lbf/pow($ft,3)];
variable psf  [expr $lbf/pow($ft,2)];
variable in2  [expr $in*$in];
variable in4  [expr $in*$in*$in*$in];
variable m2   [expr $m*$m];
variable cm2 [expr $cm*$cm];
variable PI   [expr 2*asin(1.0)];			# define constants
variable pi   [expr 2*asin(1.0)];
variable Ubig  1.e8;				# a really large number
variable Usmall  [expr 1/$Ubig];		# a really small number
```

LibAnalysisStatic.tcl

```
####################################################################
# LibAnalysisStatic.tcl: static gravity, lateral pushover or cyclic procedures
# Silvia Mazzoni, 2006
# NOTE: all global variables have been defined in LibGlobalVariables.tcl
####################################################################
# procs included in this file:
#:# proc procApplyGravity { } {
	# apply gravity load, set it constant and reset time to zero.
#:#proc procMakeLoadPattern {IDctrlDOF  iIDpushNode iPushNodeLoad ScaleFact LoadIDstatic} {
	# create load pattern for static pushover loads
#:#proc procFlateral {iLcol iFloorWeight  } {
	# calculate distribution of lateral load based on mass/weight distributions along building height
#:#proc procGenPeaks {Dmax {DincrStatic 0.01} {CycleType "Full"} {Fact 1} } {;	# generate incremental disps for Dmax
	# generate incremental disps for Dmax
#:#proc  procConvergeStatic { Tol } {
	# if analysis fails, we try some other stuff
#:#proc procAnalysisStatic { iDmax IDctrlNode IDctrlDOF {DincrStatic  0.01}  {CycleType "Full"}  {Ncycles 1} {Fact 1}   }  {
	# perform displacement-controlled static analysis (pushover or cyclic)
#:#proc procLoadCtrlStaticAnalysis {} {
	# perform force-controlled static analysis -- modify this script as needed
####################################################################
```

procApplyGravity

```
#########################################################################
# procApplyGravity.tcl -- apply gravity load, set it constant and reset time to zero.
# Silvia Mazzoni, 2006
#########################################################################
proc procApplyGravity { } {
	# apply gravity load, set it constant and reset time to zero.
	global constraintsTypeGravity alphaSP alphaMP;
	   global numbererType systemTypeGravity algorithmTypeGravity algorithmCount;
	   global testTypeGravity TolGravity maxNumIterGravity printFlagGravity;
	global NstepGravity;			# number of steps to apply gravity
	if {$constraintsTypeGravity == "Plain" | $constraintsTypeGravity == "Transformation"} {
	  constraints $constraintsTypeGravity ;
	} else {
	  constraints $constraintsTypeGravity $alphaSP $alphaMP	}
	if {$systemTypeGravity  == "SparseGeneralPivot"} {
	  system SparseGeneral  -piv;	# optional pivoting for SparseGeneral system
	} else {
	  system $systemTypeGravity	}
	numberer $numbererType;
	test $testTypeGravity  $TolGravity $maxNumIterGravity  ;
	if {$algorithmTypeGravity == "BFGS" | $algorithmTypeGravity == "Broyden" } {
		algorithm $algorithmTypeGravity $algorithmCount ;
	} elseif  {$algorithmTypeGravity == "NewtonLineSearch"} {
		algorithm $algorithmTypeGravity $NewtonLineSearchRatio;
	} else {
		algorithm $algorithmTypeGravity	}
	set DGravity [expr 1./$NstepGravity]; # first load increment;
	integrator LoadControl $DGravity
	   analysis Static
	   analyze $NstepGravity
	   loadConst  -time 0.0
};
```

LibAnalysisDynamic.tcl

```
####################################################
# LibAnalysisDynamic.tcl: procs for dynamic analysis
# Silvia Mazzoni, 2006
####################################################
#:#proc procGetTperiod {Neigen {PrintScreen "off"}} {
    #  perform eigenvalue analysis to determine fundamental periods
#:#proc procGetKsdof {Mass {PrintScreen "off"}} {
    #  perform eigenvalue analysis to determine determine sdof stiffness
#:#proc procGetOmega {{Neigen 1} {PrintScreen "off"} } {
    #  perform eigenvalue analysis to determine determine fundamental frequency
#:#proc procApplyDamping { xDamp {nEigenI 1} {nEigenJ 0} } {
    # apply Rayleigh DAMPING from $xDamp   -- from $omegaI & $omegaJ
    (modes 1&3 recomm. for mdof)
#:#proc procReadSMDFile {inFilename outFilename dt} {
    # read gm input format
#:#proc procConvergeDyna {DtAnalysis TmaxAnalysis} {
    # if analysis fails, we try some other stuff
#:#proc procAnalysisDynamic {LoadPatternType Tol DtAnalysis DtGround
    TmaxAnalysis GMfact GMFileNameA  IDdofA GMscaleA {GMFileNameB
    "nothing" } {IDdofB 0} {GMscaleB 1.0} {GMFileNameC "nothing" }  {IDdofC 0}
    {GMscaleC 1.0}   } {
    # perform dynamic (Transient) ground-motion analysis
####################################################
```

procGetTperiod

```
##################################################
# procGetTperiod $Neigen $PrintScreen
##################################################
#       Silvia Mazzoni, 2006 (mazzoni@berkeley_NO_SPAM_.edu)
#
proc procGetTperiod {Neigen {PrintScreen "off"}} {
    #  perform eigenvalue analysis to determine fundamental periods
    set fmt1 "Mode=%.1i: Tperiod=%.3f %s"
    global PI TunitTXT ;                         # load global unit variable
    set iTperiod ""
    set lambdaN [eigen $Neigen]
    for {set i 1} {$i <= $Neigen} {incr i 1} {;                # zero to one
            set lambda [lindex $lambdaN [expr $i-1]];
            set omega [expr pow($lambda,0.5)]
            set Tperiod [expr 2*$PI/$omega];                   # period (sec.)
            lappend iTperiod $Tperiod
            if {$PrintScreen == "on" } {
                    puts [format $fmt1 $i $Tperiod $TunitTXT]
            };
    }
    return $iTperiod
};
##################################################
```

LibModelBuilding.tcl

```
######################################################
# LibModelBuilding.tcl
#   Silvia Mazzoni, 2006
######################################################
#:#proc procAddFrameNodes2D {iLcol iLbeam {BoundaryConditions Free} {N0 0} }
    {
    # define nodes and boundary conditions of a 2-D frame, adding N0 to the node
    number
#:#proc procAddFrameColumns2D {iLcol iLbeam N0 IDTransf {np 5}
    {iIDSectionEXT 1} {iIDSectionINT 0}  } {
    # define column elements of a 2-D frame
#:#proc procAddFrameBeams2D {iLcol iLbeam N0 IDTransf {np 5}
    {iIDSectionEXT 1} {iIDSectionINT 0} {M0 0} } {
    # define beam elements of a 2-D frame
#:#proc procRotSpring2D {eleID nodeR nodeC matID} {
    # Create the zero length element
#:#proc procAddFrameJoints2D {iLcol iLbeam N0 iIDMaterialEXT
    {iIDMaterialINT  0} {M0 0} } {
    # define nodes and boundary conditions of a 2-D frame, adding N0 to the node
    number
#:#proc procElement2D {eleType eleTag iNode jNode arguments} {
    # define an element in 2D, simplify input
######################################################
```

procElement2D

```
########################################
## procElement2D $eleType $eleTag $iNode $jNode $arguments
########################################
# define an element in 2D, simplify input, put arg's in a list!
# by Silvia Mazzoni, 2006
#
proc procElement2D {eleType eleTag iNode jNode arguments} {
    # define an element in 2D, simplify input
    if {$eleType == "elasticBeamColumn"} {
        set A [lindex $arguments [set icount 0]];
        set E [lindex $arguments [incr icount 1]];
        set Iz [lindex $arguments [incr icount 1]];
        set transfTag [lindex $arguments [incr icount 1]];
        element elasticBeamColumn $eleTag $iNode $jNode $A $E $Iz
    $transfTag
    } elseif {$eleType == "nonlinearBeamColumn"} {
        set numIntgrPts [lindex $arguments [set icount 0]];
        set secTag [lindex $arguments [incr icount 1]];
        set transfTag [lindex $arguments [incr icount 1]];
......
```

LibMaterialsRC.tcl

```
# --------------------------------------------------------------------------------
# LibMaterialsRC.tcl:  define a library of Reinforced-Concrete materials
#                                  by Silvia Mazzoni, 2005

# General Material parameters
set G $Ubig
set J 1.0
set GJ [expr $G*$J];

############## confined and unconfined CONCRETE
#
#
#
#                                    |
#                                    |                                                  (-)
#                                    |
#                               fc1  |                      .(eps1,fc1)
#                                    |                   __/  \__
#                                    |                __/       \__
#                                    |   /         __/             \__
#                                    |  /       __/                   \_
#                               fc2|  Ec _/                            \________________________
#                                    |  /  _/                            (eps2,fc2)
#                                    | /_/
#                      epstu         | /
#   eps<------------------------.---------|------------------------------------------------------------
#            (epst0,ft0)        \      /|                          eps1          eps2
#                              Ets  / |
#                                \ /  |
#                                 \/  |
#                               (ft) |
#                                    |
#                                    |
#         (+)                        |
#                                   <fc
#
#
```

LibMaterialsRC.tcl (cont.)

```
# nominal concrete compressive strength
set fc [expr -4.0*$ksi];                # CONCRETE Compressive Strength, ksi   (+Tension, -Compression)
set Ec [expr 57*$ksi*sqrt(-$fc/$psi)];          # Concrete Elastic Modulus
set Kfc 1.3;                            # ratio of CONFINED to unconfined concrete strength
set fc1C [expr $Kfc*$fc];               # confined concrete (mander model), maximum stress
set eps1C [expr 2.*$fc1C/$Ec];          # strain at maximum stress
set fc2C [expr 0.2*$fc1C];              # ultimate stress
set eps2C [expr 5*$eps1C];              # strain at ultimate stress
set fc1U     $fc;                       # UNCONFINED concrete (todeschini parabolic model), maximum stress
set eps1U    -0.003;                    # strain at maximum strength of unconfined concrete
set fc2U      [expr 0.2*$fc1U];         # ultimate stress
set eps2U    -0.01;                     # strain at ultimate stress
set lambda 0.1;                         # ratio between unloading slope at $eps2 and initial slope $Ec

# tensile-strength properties
set ftC [expr -0.14*$fc1C];                     # tensile strength +tension
set ftU [expr -0.14*$fc1U];                     # tensile strength +tension
set Ets [expr $ftU/0.002];                      # tension softening stiffness

# set up library of materials
if {  [info exists imat ] != 1} {set imat 0};                   # set value only if it has not been defined previously.
uniaxialMaterial Elastic [set IDElastConc [incr imat 1]] $Ec;                   # elastic concrete material
uniaxialMaterial Elastic [set IDelasticUnity [incr imat 1]] 1.0;                # elastic material
uniaxialMaterial Elastic [set IDelasticRigid [incr imat 1]] [expr $Ec*$Ubig];           # elastic material
uniaxialMaterial Concrete01 [set IDconcCore01 [incr imat 1]] $fc1C $eps1C $fc2C $eps2C;           # Core concrete
uniaxialMaterial Concrete01 [set IDconcCover01 [incr imat 1]] $fc1U $eps1U $fc2U $eps2U;          # Cover concrete
uniaxialMaterial Concrete02 [set IDconcCore02 [incr imat 1]] $fc1C $eps1C $fc2C $eps2C $lambda $ftC $Ets;     # Core
uniaxialMaterial Concrete02 [set IDconcCover02 [incr imat 1]] $fc1U $eps1U $fc2U $eps2U $lambda $ftU $Ets;  # Cover
```

LibDisplay.tcl

```
##################################################
#LibDisplay.tcl
##################################################
#   Silvia Mazzoni, 2006
##################################################
#:#proc procDisplayPlane {ShapeType dAmp viewPlane {nEigen 0}  {quadrant 0}} {
    ## setup display parameters for specified viewPlane
#:#proc procDisplayShape3D { ShapeType {dAmp 5}  {xLoc 10} {yLoc 10} {xPixels 750} {yPixels 600} {nEigen 1} } {
    ## display Node Numbers, Deformed or Mode Shape in all 3 planes
#:#proc procDisplayShape2D { ShapeType {dAmp 5}  {xLoc 10} {yLoc 10} {xPixels 750} {yPixels 600} {nEigen 0} } {
    ## display Node Numbers, Deformed or Mode Shape in 2D problem
#:#proc procDisplayAll { {dAmp 5} } {
    ## display Node Numbers, Deformed AND Mode Shape using default values.
#:#proc procDisplayDeformedShape { {dAmp 5} } {
    ## display Deformed Shape using default values.
#:#proc procDisplayNodeNumbers {  } {
    ## display Node Numbers using default values.
##################################################
```

procDisplayDeformedShape

```
####################################################################
## procDisplayDeformedShape $dAmp
####################################################################
proc procDisplayDeformedShape { {dAmp 5} } {
	## display Deformed Shape using default values.
	# view model -- node numbers
	set xPixels 800;
	set yPixels 600;
	set xLoc1 10;
	set yLoc1 10;

#	set  dAmp 5;		# relative amplification factor for deformations
	procDisplayShape2D DeformedShape $dAmp $xLoc1 $yLoc1  800 600
};										#
####################################################################


####################################################################
## procDisplayNodeNumbers $dAmp
####################################################################
proc procDisplayNodeNumbers {  } {
#
	## display Node Numbers using default values.
	# view model -- node numbers
	set xPixels 800;
	set yPixels 600;
	set xLoc1 10;
	set yLoc1 10;

	procDisplayShape2D NodeNumbers 1 $xLoc1 $yLoc1  $xPixels $yPixels
};										#
####################################################################
```

LibGeneralProcs.tcl

```
#################################################################
# LibGeneralProcs.tcl -- define general-purpose procs
#                                       by Silvia Mazzoni, 2006
#:#proc procGetDb {BarSize} {
    # standard reinforcing bar nominal diameter: sizes #3, #4, #5, #6, #7, #8, #9, #10, #11, #14, #18
#:#proc procGetAb {BarSize} {
    # standard reinforcing bar nominal area: sizes #3, #4, #5, #6, #7, #8, #9, #10, #11, #14, #18
#:#proc procMax {vetto} {
    # find max of a list
#:#proc procSign {xx}            {
    # find sign of a variable
#################################################################

#################################################################
proc procGetDb {BarSize} {
    # standard reinforcing bar nominal diameter: sizes #3, #4, #5, #6, #7, #8, #9, #10, #11, #14, #18
    #       Silvia Mazzoni, 2006
    global in
    set keyDbNominal " #3 [expr 0.375*$in]  #4 [expr 0.50*$in]   #5  [expr 0.625*$in] #6  [expr
    0.75*$in] #7 [expr 0.875*$in] #8 [expr 1.0*$in]   #9 [expr 1.128*$in] #10  [expr 1.27*$in]  #11
    [expr 1.41*$in] #14  [expr 1.693*$in] #18  [expr 2.257*$in] ";
    set DbNomBar [string map $keyDbNominal $BarSize]
    return $DbNomBar
}
#################################################################
….
```

Example:

```
set AnalysisTypeTXT Push
source buildModel.tcl
procDisplayAll $DisplayFactor
source analysisStatic.tcl
#
set AnalysisTypeTXT Cycl
source buildModel.tcl
procDisplayAll $DisplayFactor
source analysisStatic.tcl
#
set AnalysisTypeTXT Dyna
source buildModel.tcl
procDisplayAll $DisplayFactor
source analysisGM.tcl
```

Static Pushover

Static Cyclic

Dynamic EQ

BuildFrame2d.tcl

```
# ------------------------------------------------------------------------------------------------------------------------------------------
# buildFrame2D.tcl: generates frame nodes/materials/sections/elements
#                                                  by Silvia Mazzoni, 2005
#
#  41______(241)___42_______(242)______43_________(243)______44______(244)________45-o------(344)-----o-46          _
#  |           BS4        |          BS4          |             BS4             |            BS4               |          BS5        o             |
#  |(141)                 |(142)                  |(143)                        |(144)                         |(145)                |(136)  LColTop
#  | CS6                  | CS4                   | CS4                         | CS4                          | CS6                 o CS7        |
#  31______(231)___32______(232)_______33_________(233)______34______(234)_________35-o-----(334)-----o-36          _
#  |           BS3        |          BS3          |             BS3             |            BS3               |          BS5        o             |
#  |(131)                 |(132)                  |(133)                        |(134)                         |(135)                |(126)  LColTop
#  | CS6                  | CS4                   | CS4                         | CS4                          | CS6                 o CS7        |
#  21______(221)___22______(222)_______23_________(223)______24______(224)_________25-o-----(324)-----o-26          _
#  |           BS2        |          BS2          |             BS2             |            BS2               |          BS5        o             |
#  |(121)                 |(122)                  |(123)                        |(124)                         |(125)                |(116)  LColTop
#  | CS5                  | CS3                   | CS3                         | CS3                          | CS5                 o CS7        |
#  11______(211)___12______(212)_______13_________(213)______14______(214)_________15-o-----(314)-----o-16          _
#  |           BS1        |          BS1          |             BS1             |            BS1               |          BS5        o             |
#  |                      |                       |                             |                              |                     |             |
#  |(111)                 |(112)                  |(113)                        |(114)                         |(115)                |(106)  LColBot
#  | CS1                  | CS2                   | CS2                         | CS2                          | CS1                 | CS7        |
#  |                      |                       |                             |                              |                     o             |
#  === 1                  === 2                   === 3                         === 4                          === 5                 === 6         -
#
#  |-----------LBeam------|----------LBeam-----------|----------LBeam--------------|----------LBeam--------------|--------LBeam-------|
#
#

wipe;                                  # clear memory of all past model definitions
# Define the model builder
model BasicBuilder -ndm 2 -ndf 3
variable problemSize Large;            # option, Large or Small (less then 10 nodes) -- used to make decisions
source Libraries.tcl;                  # set up all variables, procs and utilities libraries
.......
# REINFORCED-CONCRETE material properties
source $LibDir/LibMaterialsRC.tcl;     # baseline RC materials
```

buildFrame2d.tcl

```
…
if {$ColElemType == "Elastic" } {
	set ColArgs "$Ubig $Ec $Iz $IDcolTrans"
} else {
	set ColArgs "$np $secID $IDcolTrans"
}
procElement2D $ColElemType 1 1 2 $ColArgs
procElement2D $ColElemType 2 2 3 $ColArgs
procElement2D $ColElemType 3 3 4 $ColArgs
procElement2D $ColElemType 4 4 5 $ColArgs
procElement2D $ColElemType 5 5 6 $ColArgs
procElement2D $ColElemType 6 6 7 $ColArgs
…
# apply Rayleigh DAMPING from $xDamp
set xDamp 0.02 ;	# damping ratio
set modeI 1;		# modes to be used for mass and stiffness-proportional damping
set modeJ 3;
procApplyDamping $xDamp $modeI $modeJ

procGetTperiod 5 on;		# get Natural Periods and print to screen (on)
…
procApplyGravity;	# apply gravity load, set it constant and reset time to zero.
…
procDisplayNodeNumbers
puts FrameBuilt!!
```

AnalysisStatic.tcl

```
# ----------------------------------------------------------------
# AnalysisStatic.tcl
# Silvia Mazzoni, 2006
#

if {$AnalysisTypeTXT=="Push" } {
	set iDmax [expr 0.11*$Ldrift ]
	set Fact 1.0
	set CycleType Push
	set NCycles 1
} elseif {$AnalysisTypeTXT=="Cycl" } {
	set iDmax "0.001  0.005   0.01   0.015  0.025   0.05  0.075   0.09  0.11 "
	set Fact $Ldrift
	set CycleType Full
	set NCycles 2
} else {
	puts "No Analysis Type Specified"
	return
}

# calculate distribution of lateral load based on weight distributions and define load pattern
set iFj [procFlateral $iLCol $iFloorWeight];
# create load pattern for lateral loads
procMakeLoadPattern $IDctrlDOF $iIDpushNode $ iFj 1.0 $ loadIDstatic

procAnalysisStatic $iDmax $IDctrlNode $IDctrlDOF $DincrStatic  $CycleType $NCycles $Fact
```

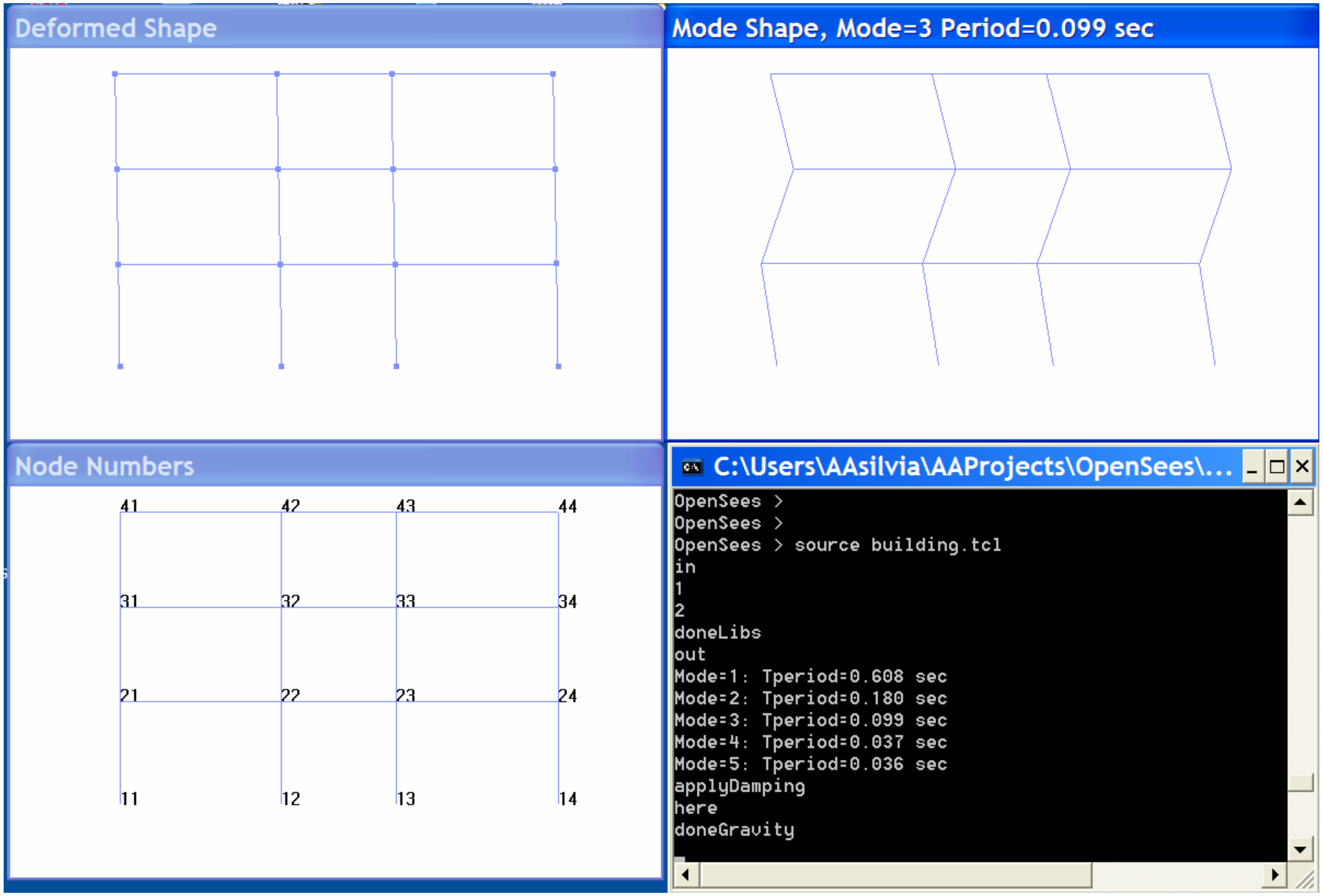
AnalysisGMot.tcl

```
# --------------------------------------------------------------------
# AnalysisGMot: dynamic ground-motion analysis
# Silvia Mazzoni, Febr 2005
#

# ------------------------------
# Define earthquake excitation
# ----------------------------

# set analysis parameters
set LoadPatternType Uniform;          # options: "UniformSupport" (default) "MultipleSupport"
set xDamp    0.02;                    # modal damping ratio
set Tol 1e-6;                         # convergence tolerance
#set omega  [procGetOmega];           # fundamental modal frequency -- calculated before gravity loads
set DtAnalysis [expr 0.01*$sec];      # time-step Dt for lateral analysis (remove *$sec if no units are defined)
set DtGround [expr 0.02*$sec];        # time-step Dt for input grond motion
set TmaxAnalysis [expr 50. *$sec];    # maximum duration of ground-motion analysis -- should be 50*$sec
set GMfact $g;                        # ground-motion input-units factor (acceleration: $g) (displacement $cm)
set GMdir "GMfiles/";                 # directory where ground motions are
set GMFileType "PEER";                # ground-motion file type
set GMFileNameA $GroundFile.at2;      # ground-motion filename for input A
set IDdofA 1;                         # lateral dof for ground motion input A
set GMscaleA 2.0;                     # scaling of ground motion for input A

procDynamicAnalysis $LoadPatternType $xDamp $omega $Tol $DtAnalysis $DtGround $TmaxAnalysis $GMfact
     $GMdir $GMFileType $GMFileNameA $IDdofA $GMscaleA
```

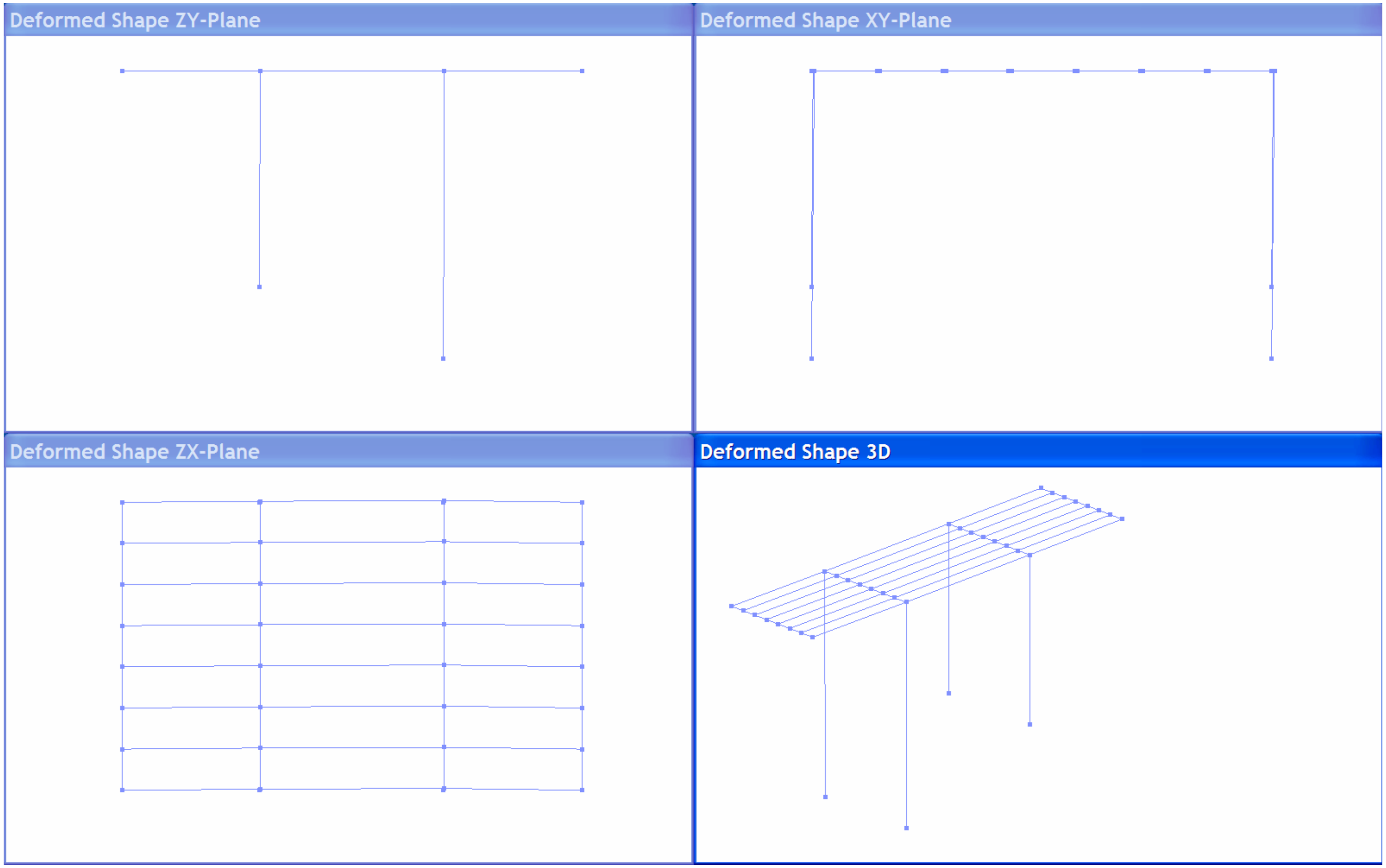
Example Runs

- Building.tcl – 2D
  - Eq ground motion
- Bridge.tcl – 3D
  - Eq ground motion
- Frame – 2D
  - Multiple-support excitation
- Portal Frame 2D
  - inelastic section vs. fiber section, static analysis

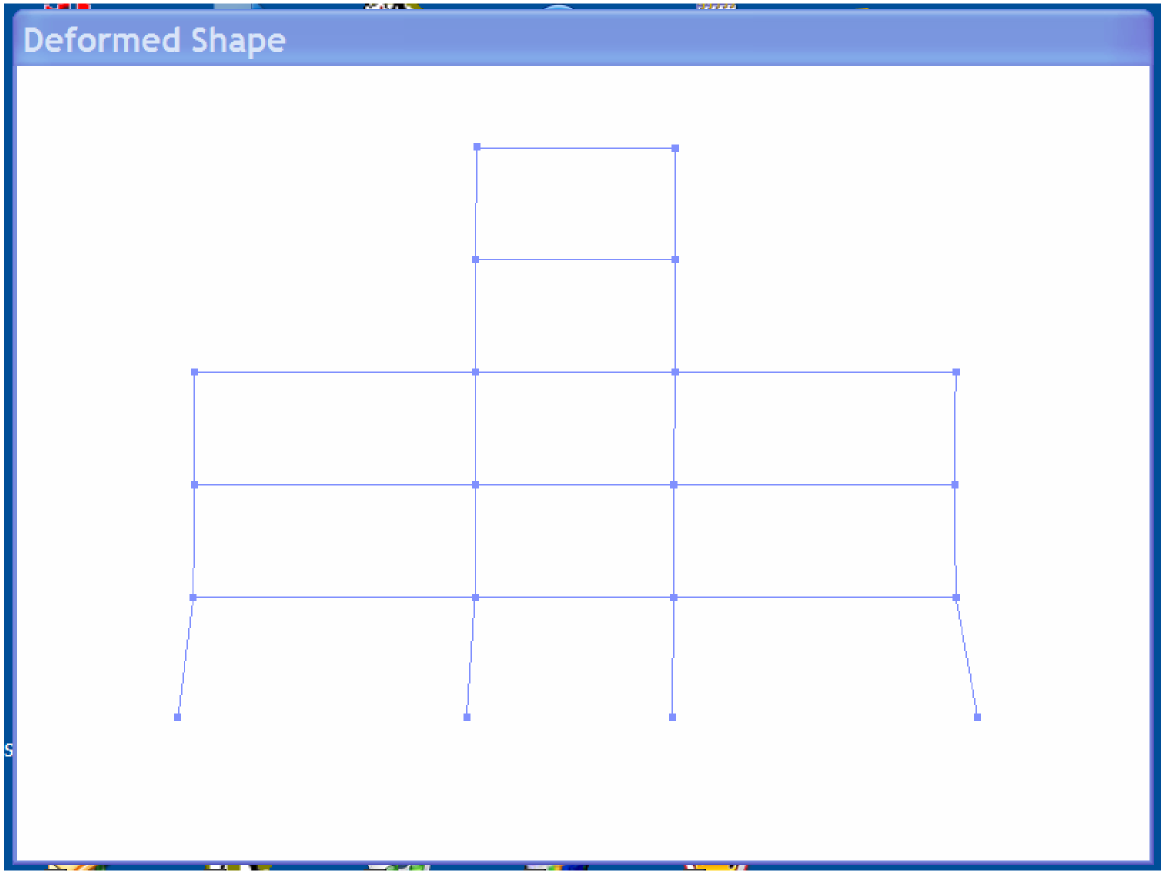
Building.tcl - 2D model

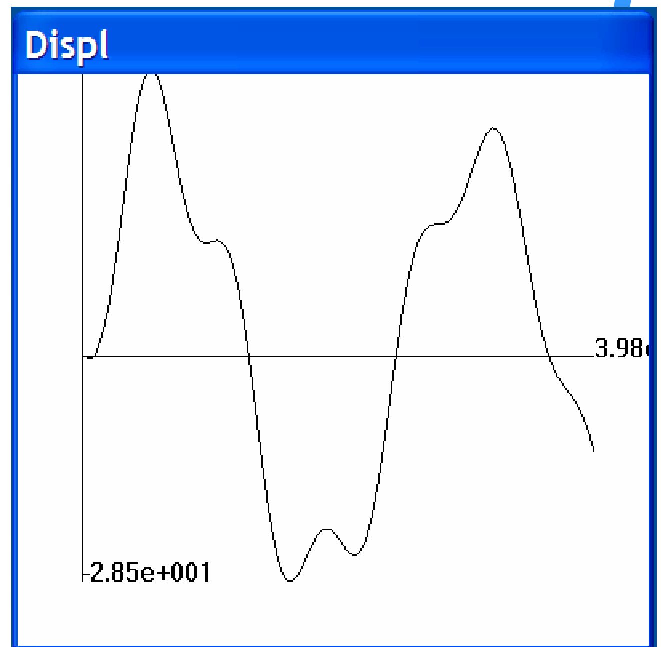


Bridge.tcl - 3D model, 2D EQ

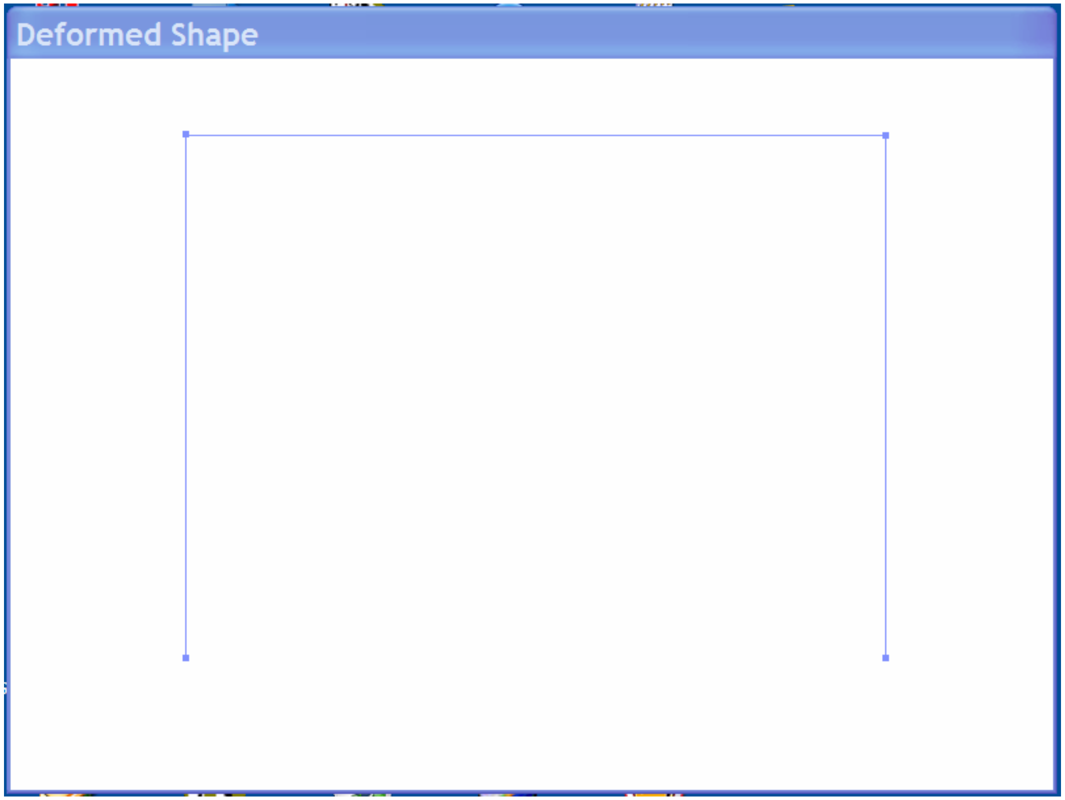


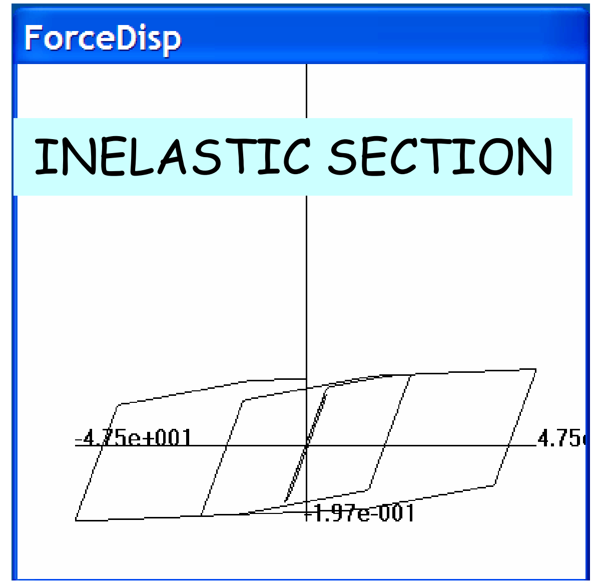
MultiSupport Building

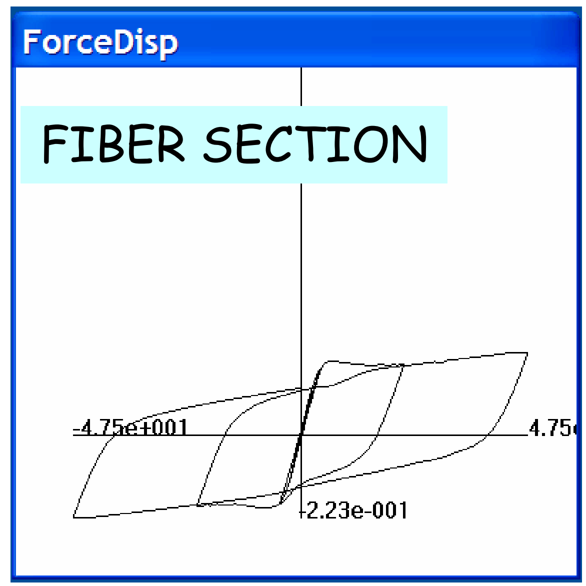




Portal Frame - 2D

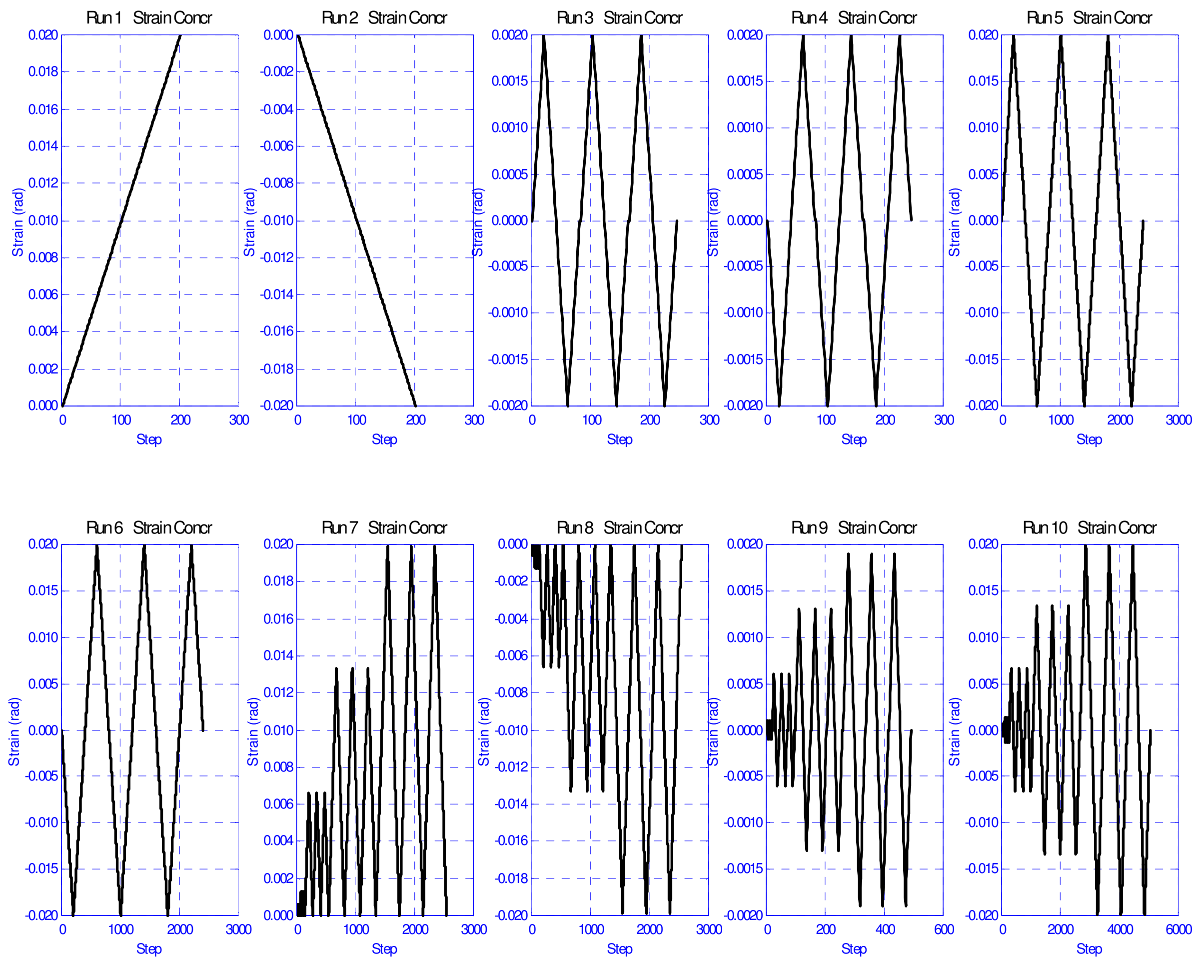




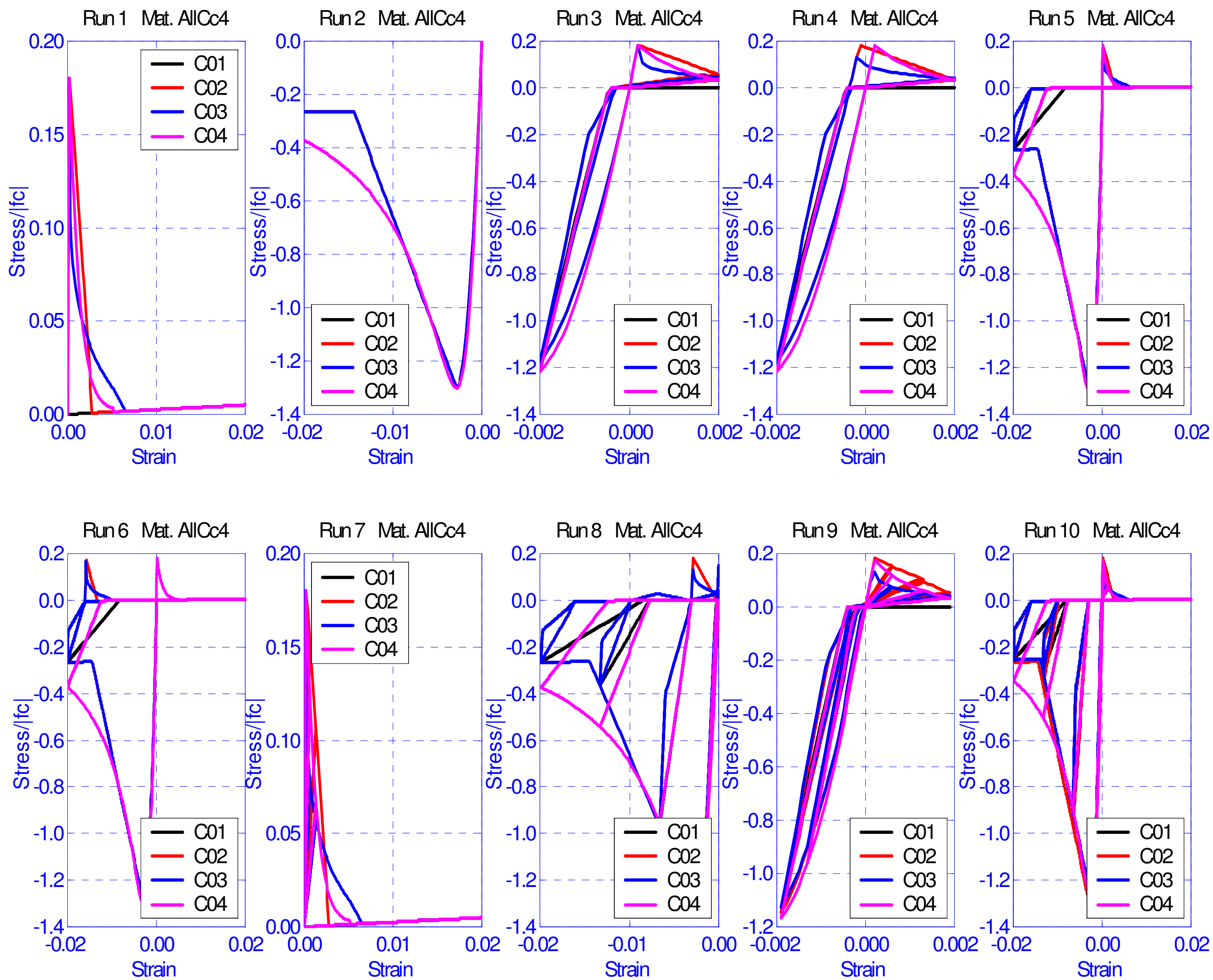


Comparison of OS Models

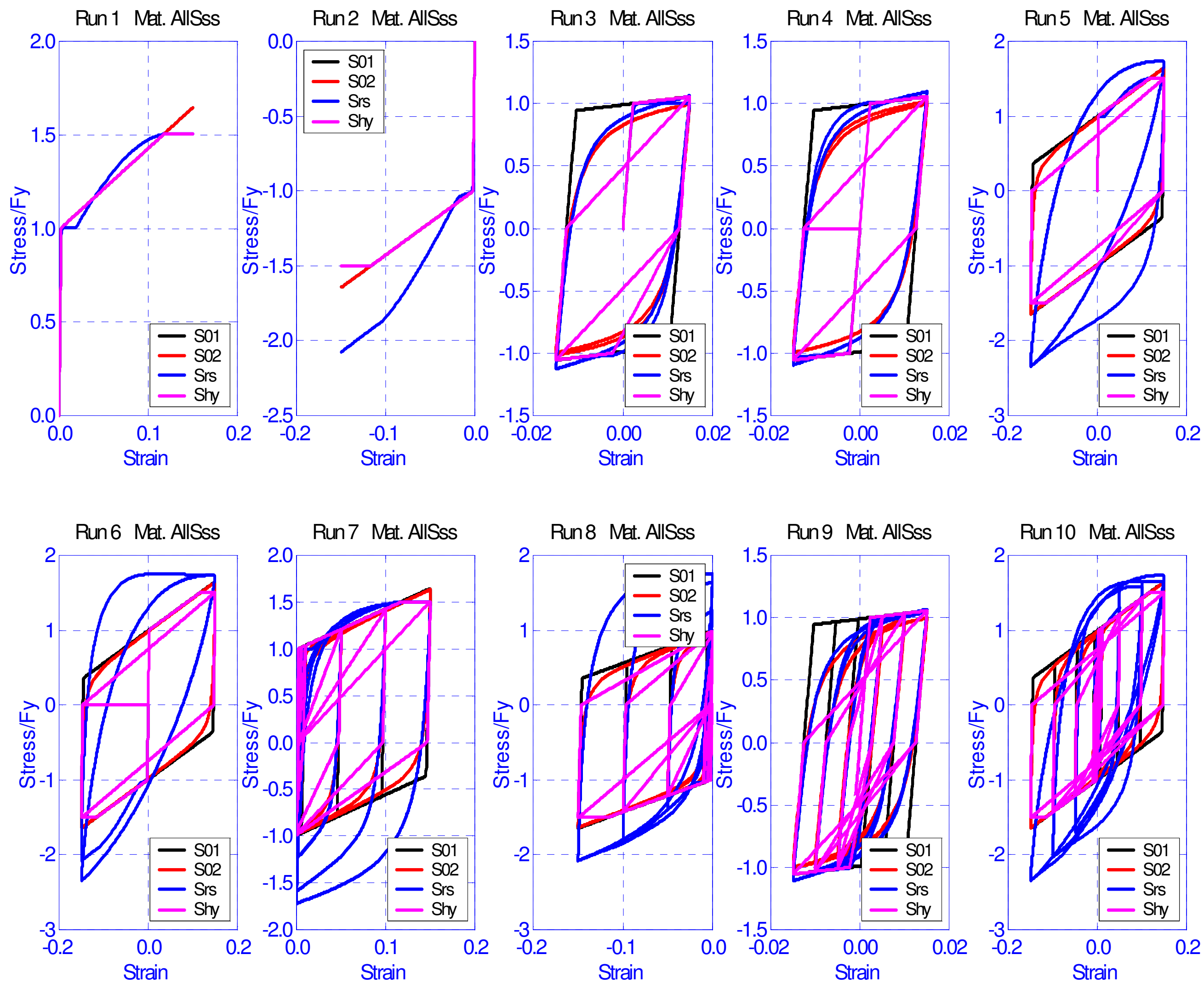
- Level 1. Material Models



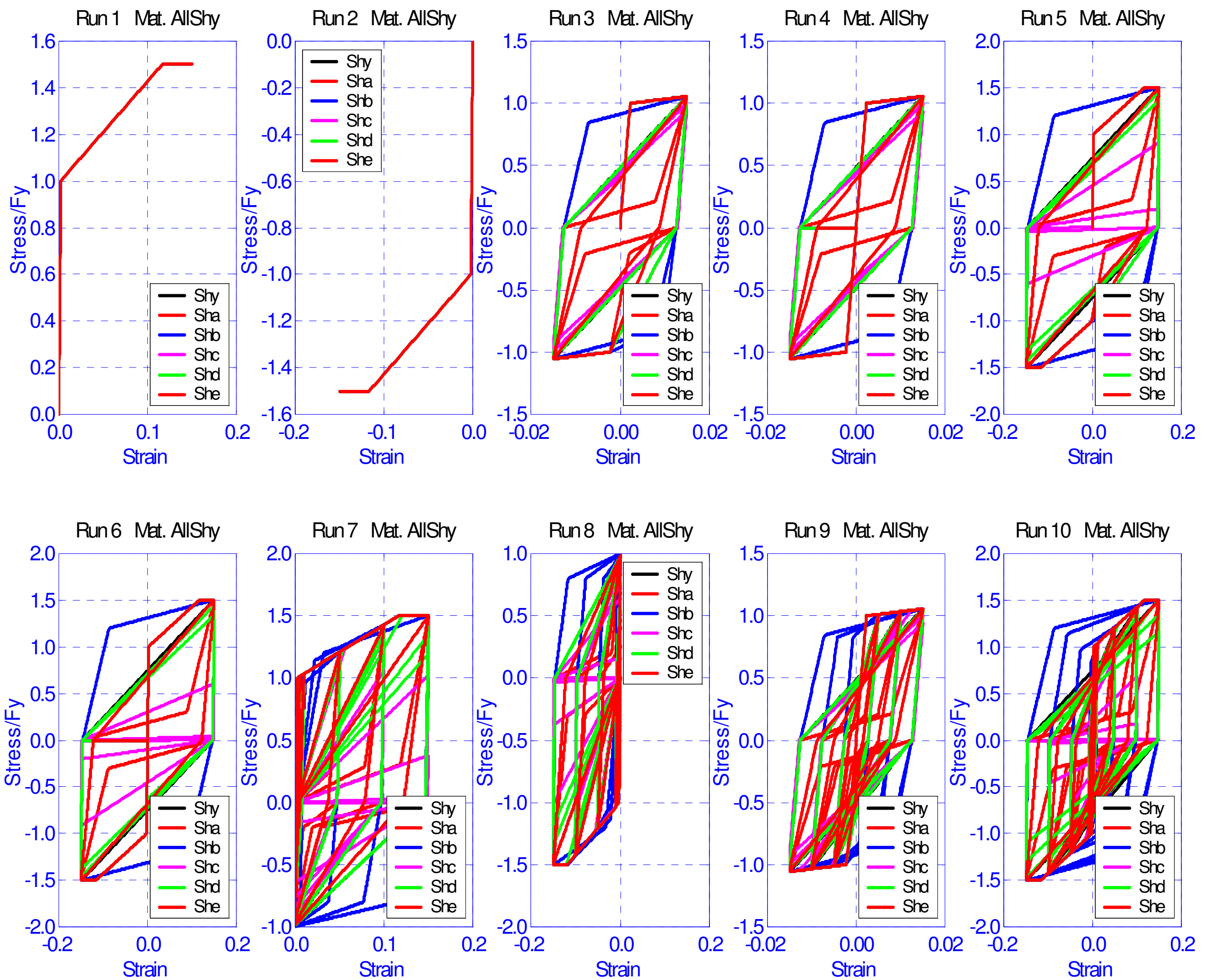
Concrete Material Models



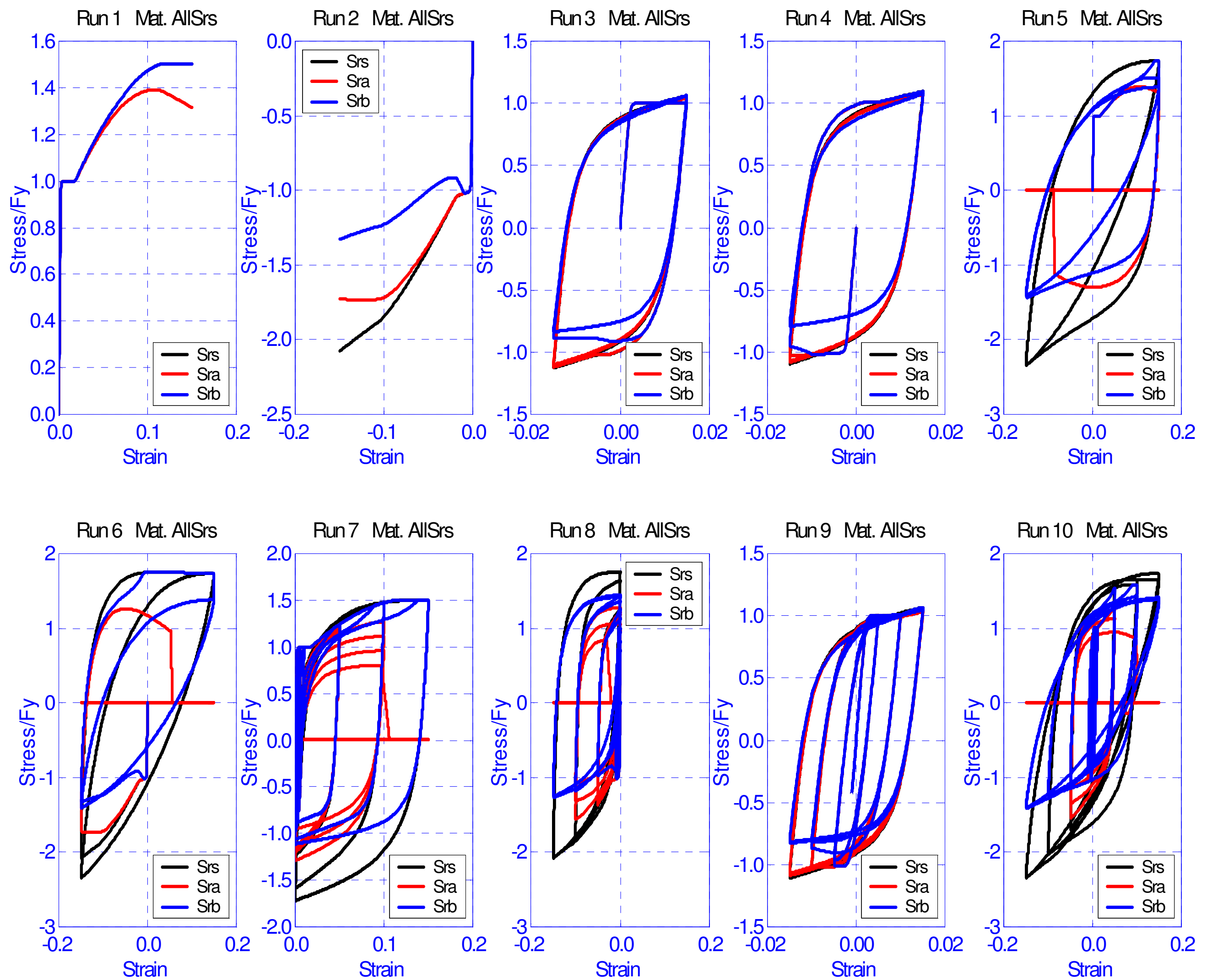
Steel Material Models



Hysteretic-Material Models



ReinforcingSteel Material Models



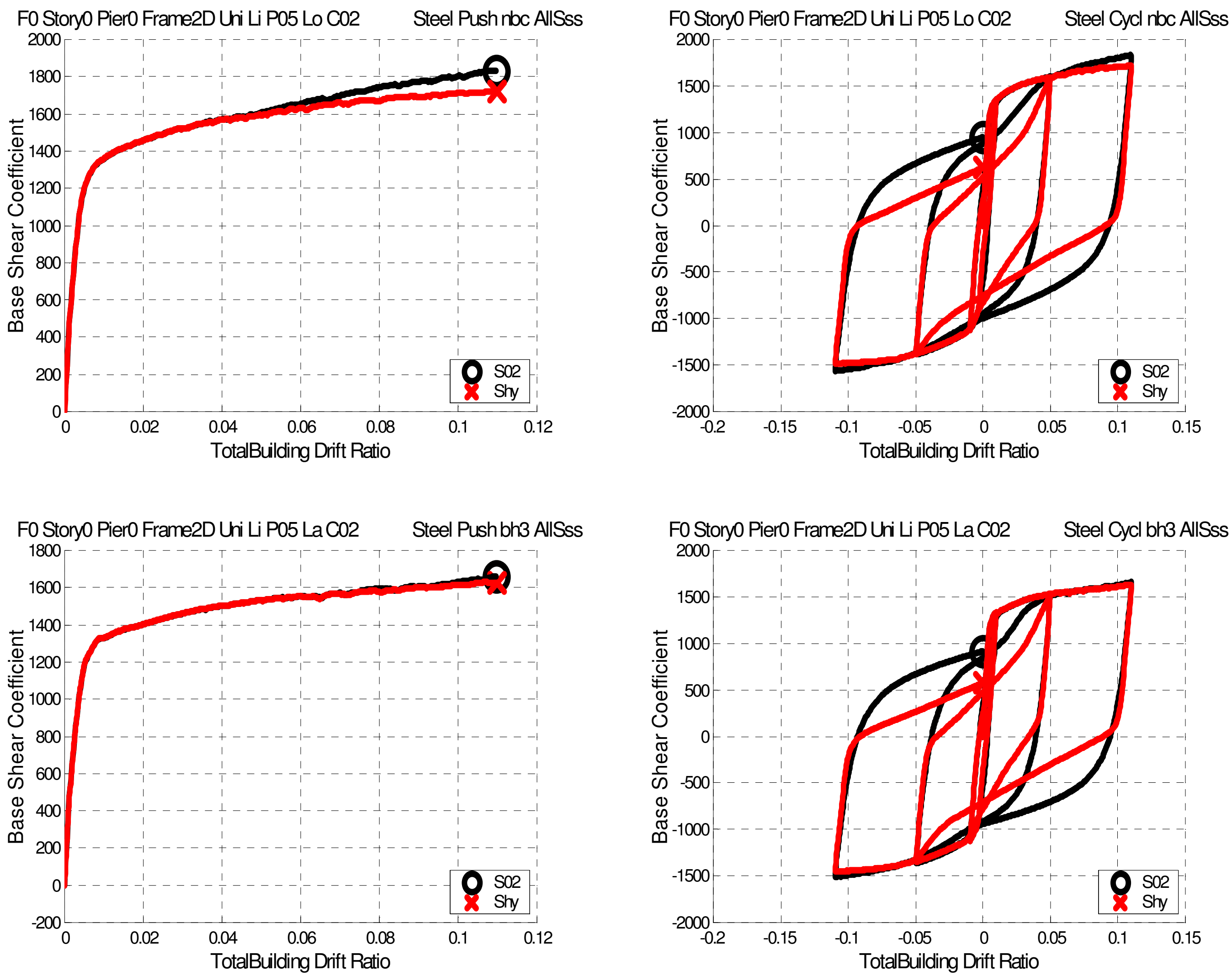
Level 2. Building Frame

- Model Quantities
  - Material: Concrete & Steel
  - Element
    - NonlinearBeamColumn Element
    - BeamWithHinges
  - Hinge Length in bwh
  - PDelta Effect
- Analysis
  - Static Pushover
  - Static Reversed Cyclic
- Response Quantities
  - Global Shear/Drift Response
  - Interstory-Drift Distribution
  - Local Moment-Curvature Response
  - Material Stress/Strain Response

Steel Materials - Shear vs. Drift

nonlinearBeamColumn

beamWithHinges



pushover

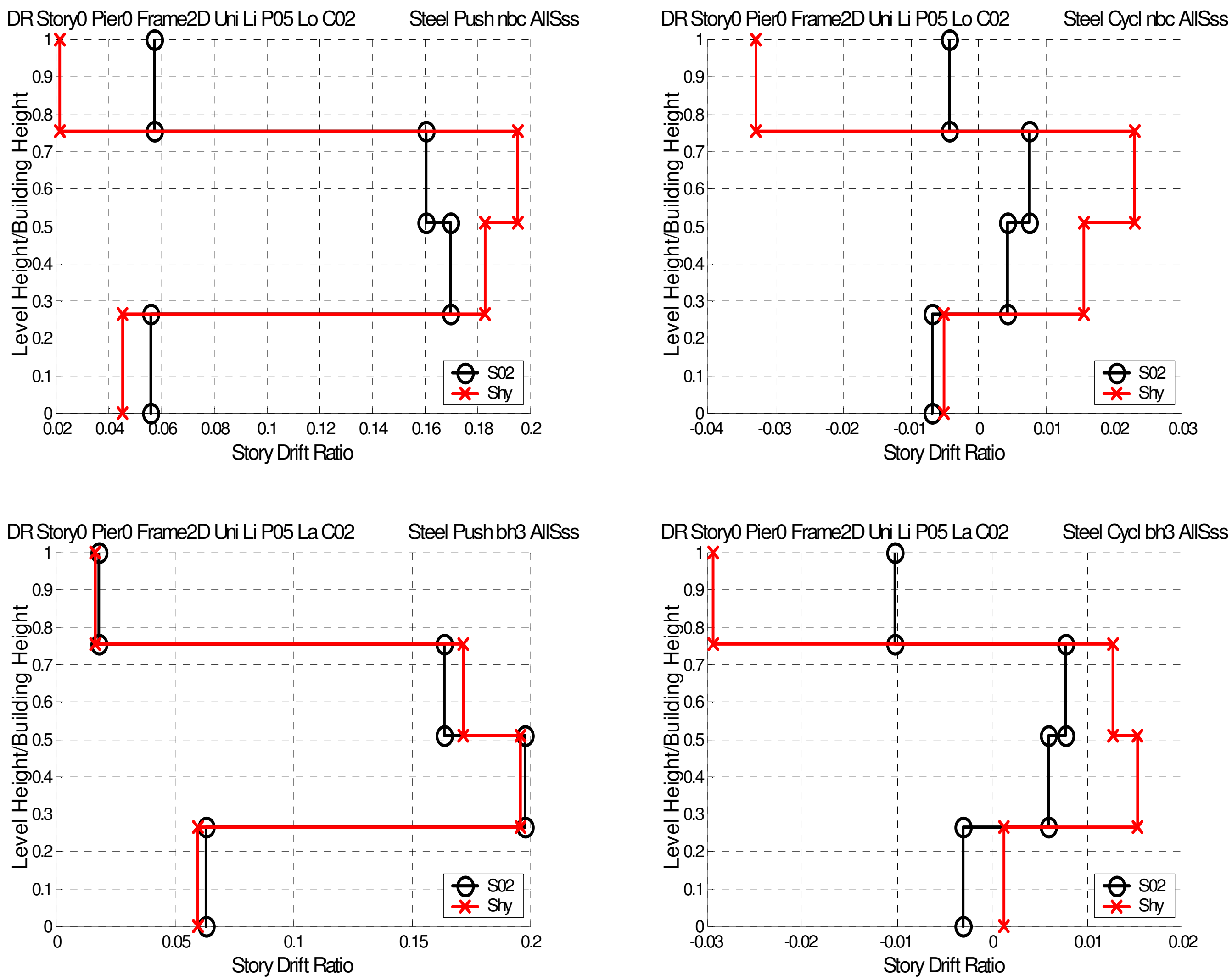
cyclic

Silvia Mazzoni OpenSees Days 2006

Steel Materials - Interstory Drift Distribution

nonlinearBeamColumn

beamWithHinges



pushover

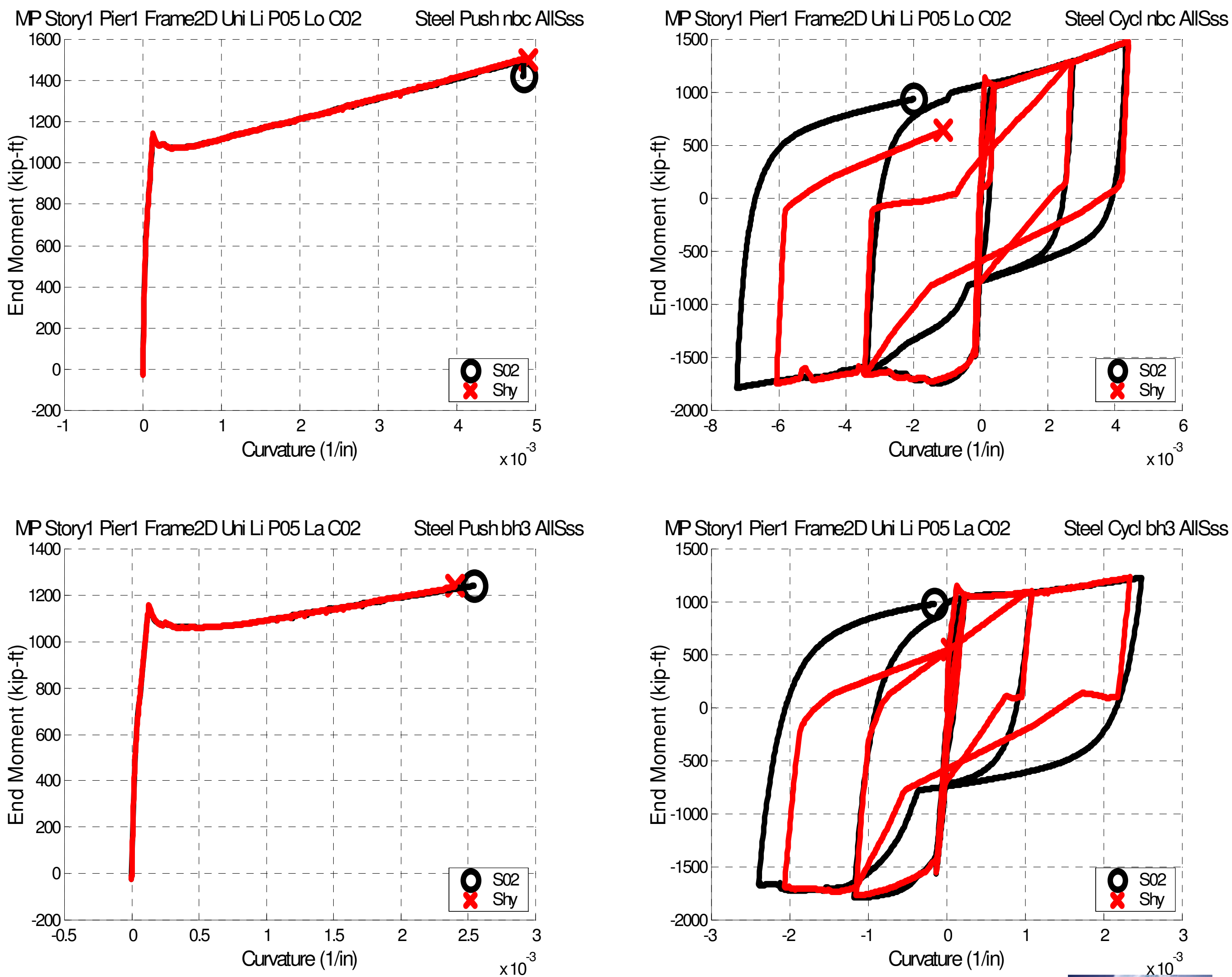
cyclic

Silvia Mazzoni OpenSees Days 2006

Steel Materials - Moment-Curvature in Base Column

nonlinearBeamColumn

beamWithHinges



pushover

cyclic

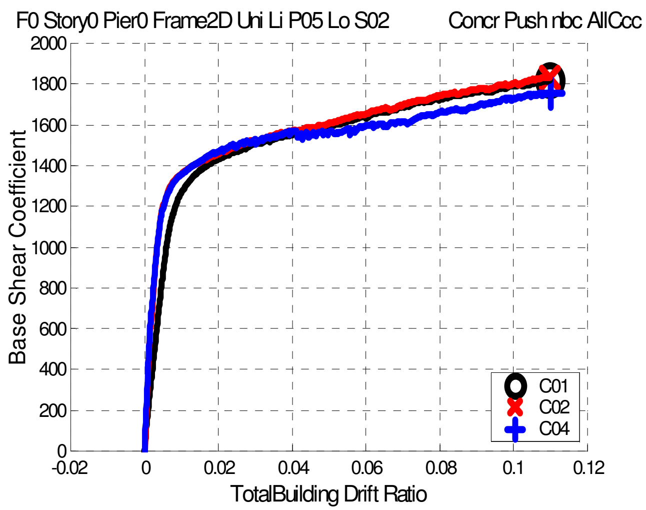
Silvia Mazzoni OpenSees Days 2006

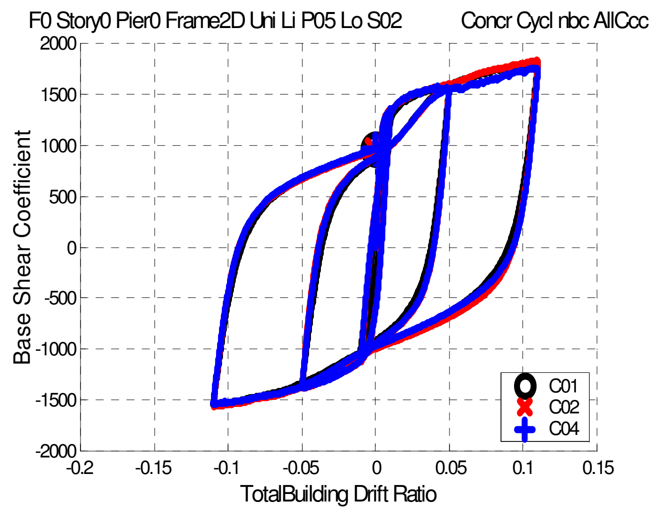
OpenSees

OpenSees NEESit

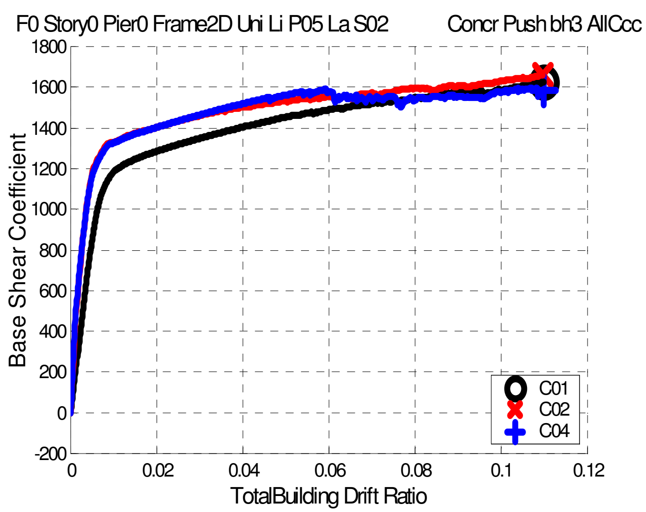
Concrete Materials - Shear vs. Drift

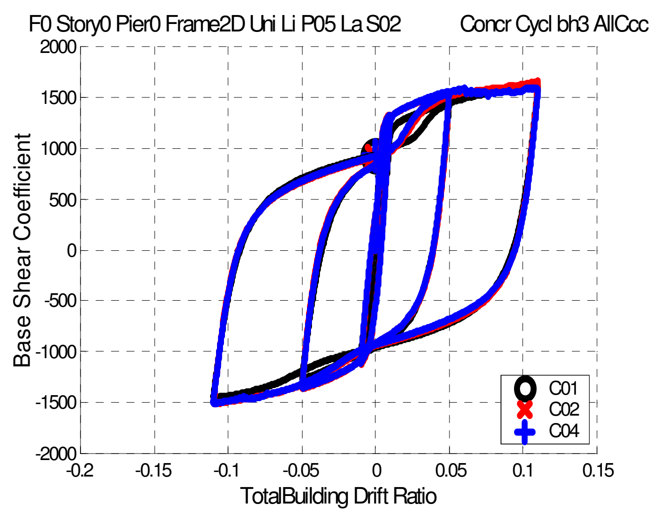
nonlinearBeamColumn





beamWithHinges

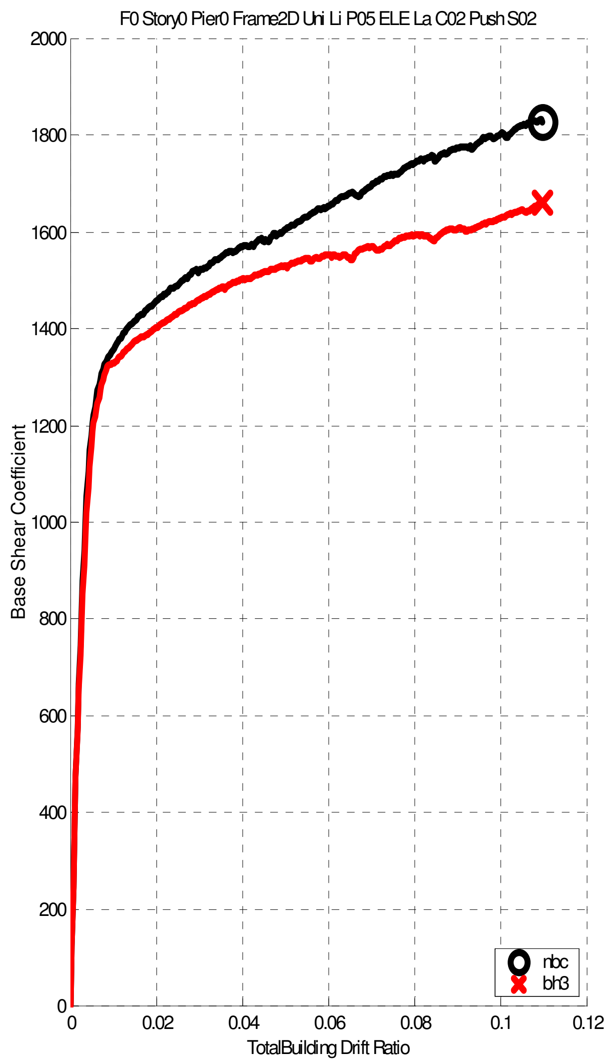




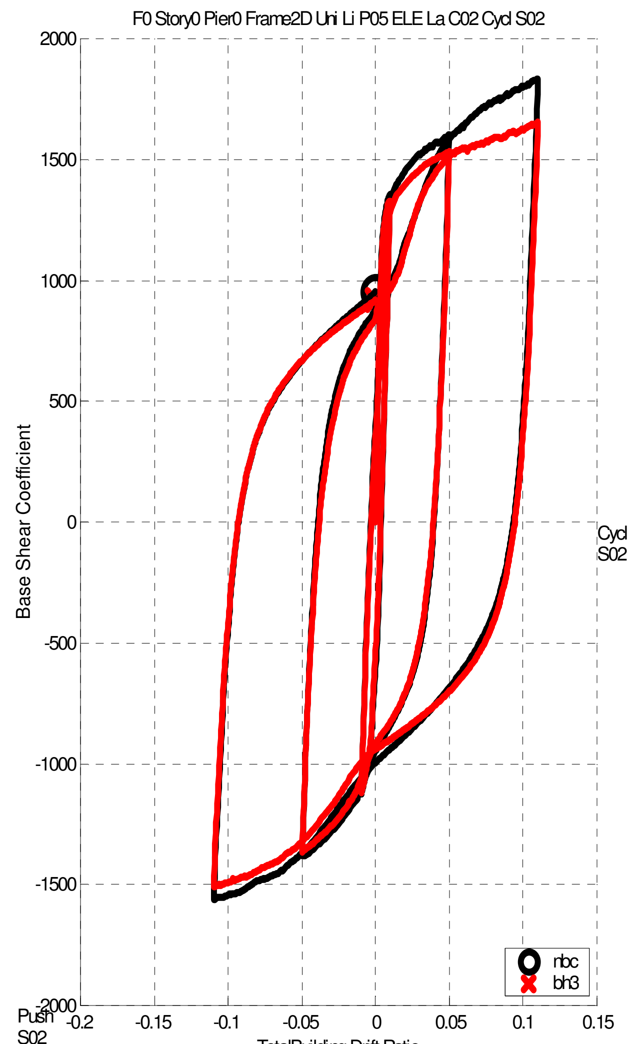
pushover

cyclic

Element Type - Shear vs. Drift



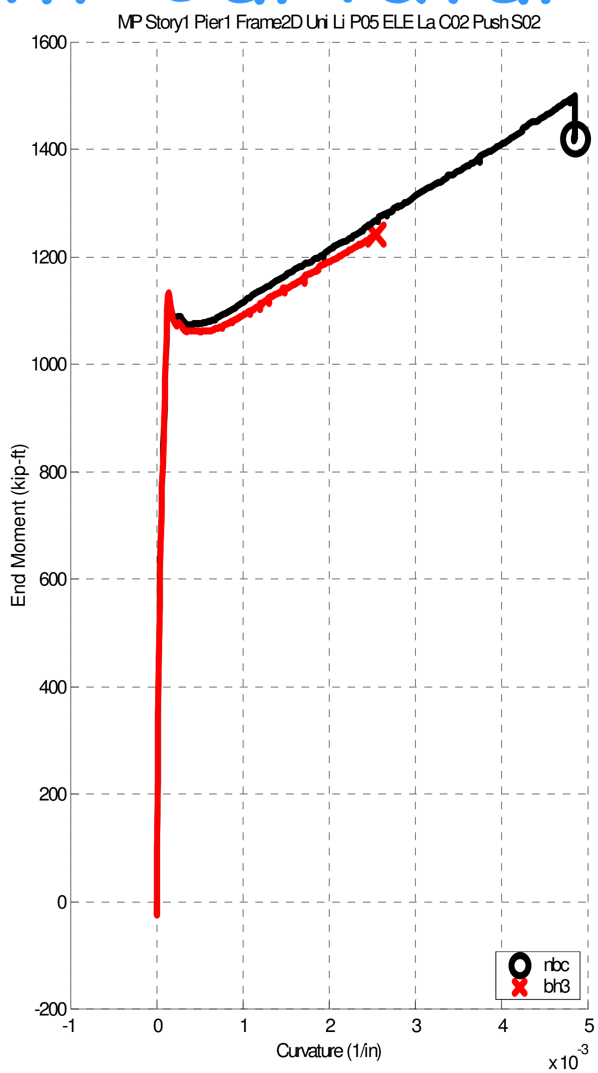
pushover



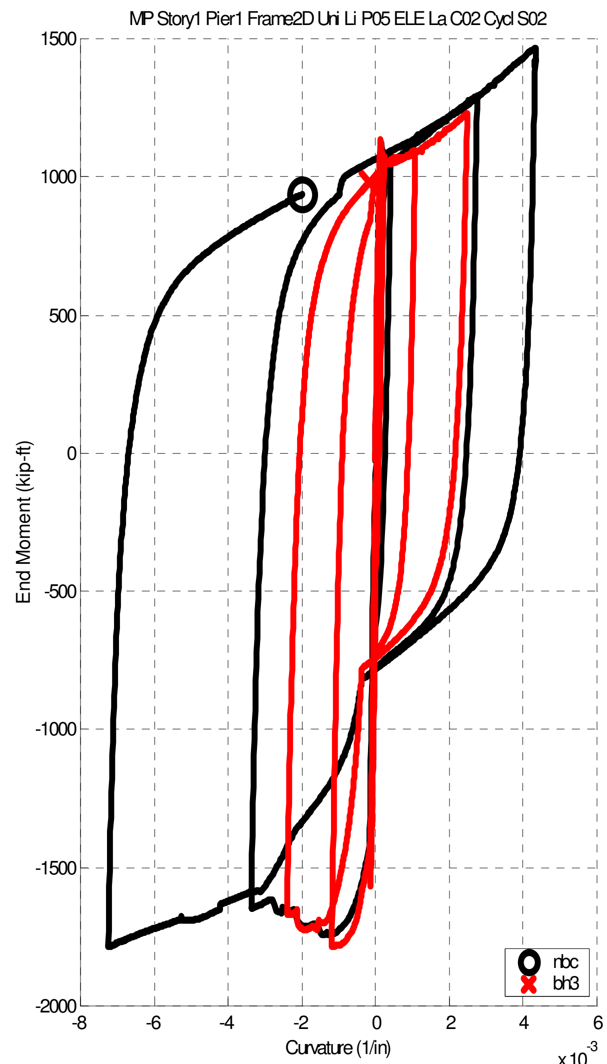
cyclic

Silvia Mazzoni OpenSees Days 2006

Element Type - Moment-Curvature

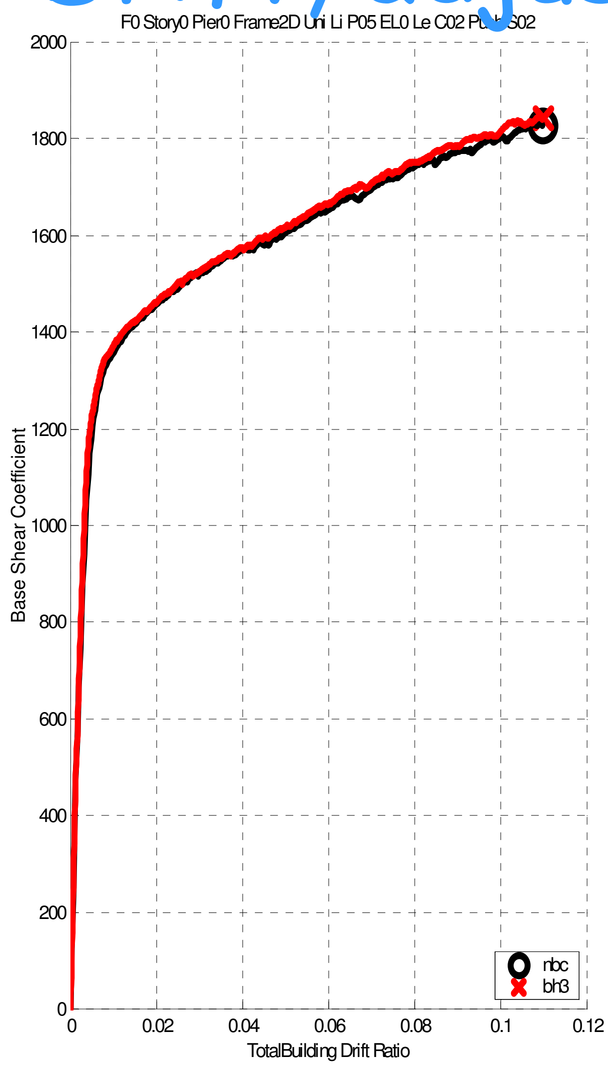


pushover

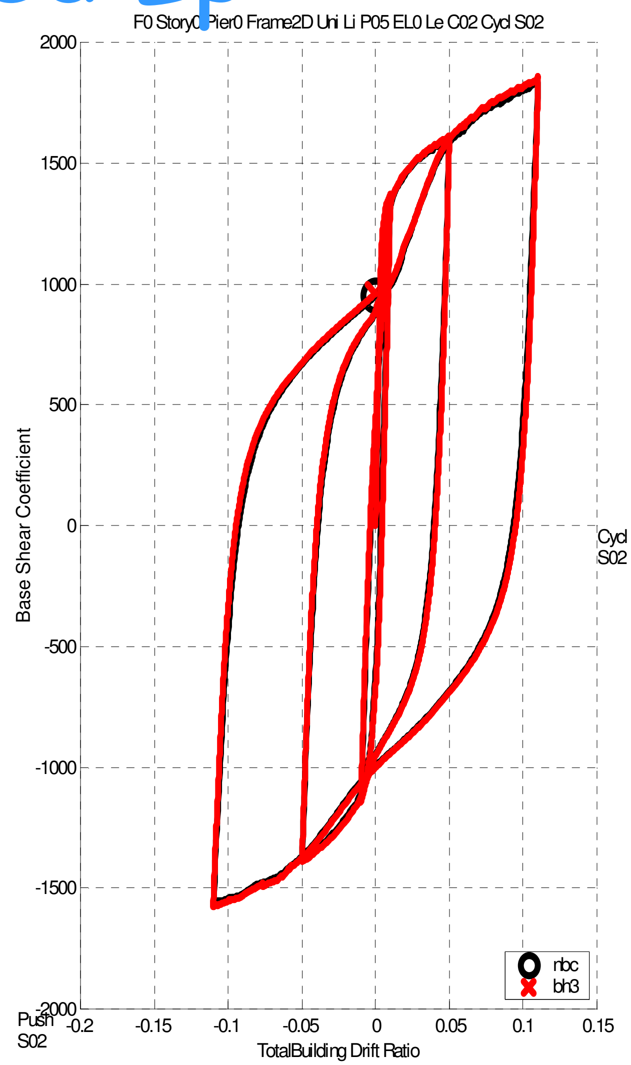


cyclic

Element Type - Force-Drift, adjusted Lp



pushover

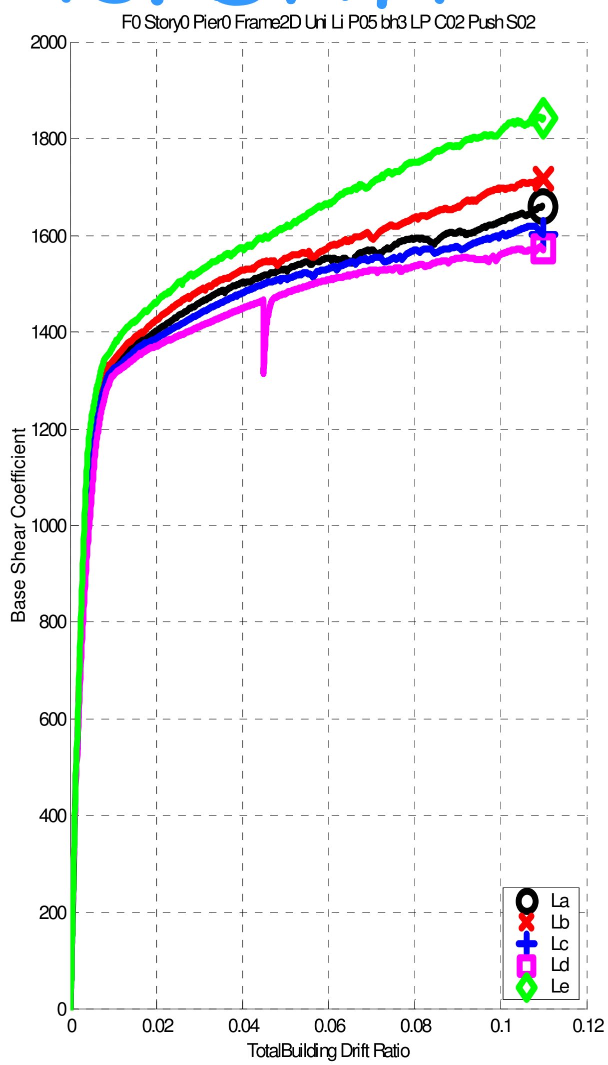


cyclic

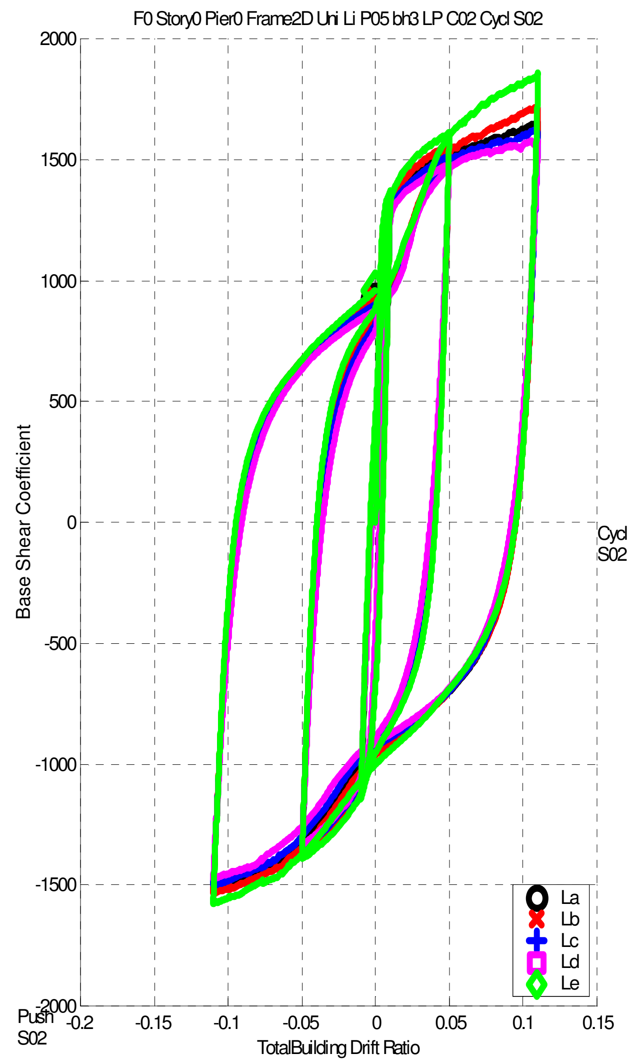
Silvia Mazzoni OpenSees Days 2006

Plastic-Hinge Length
Shear vs. Drift

beamWithHinges



pushover

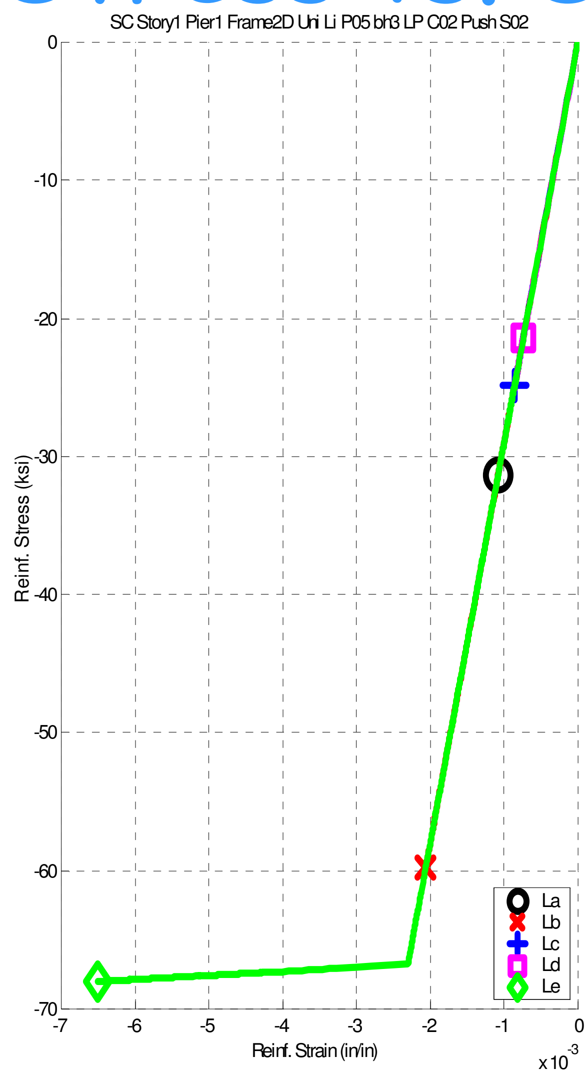


cyclic

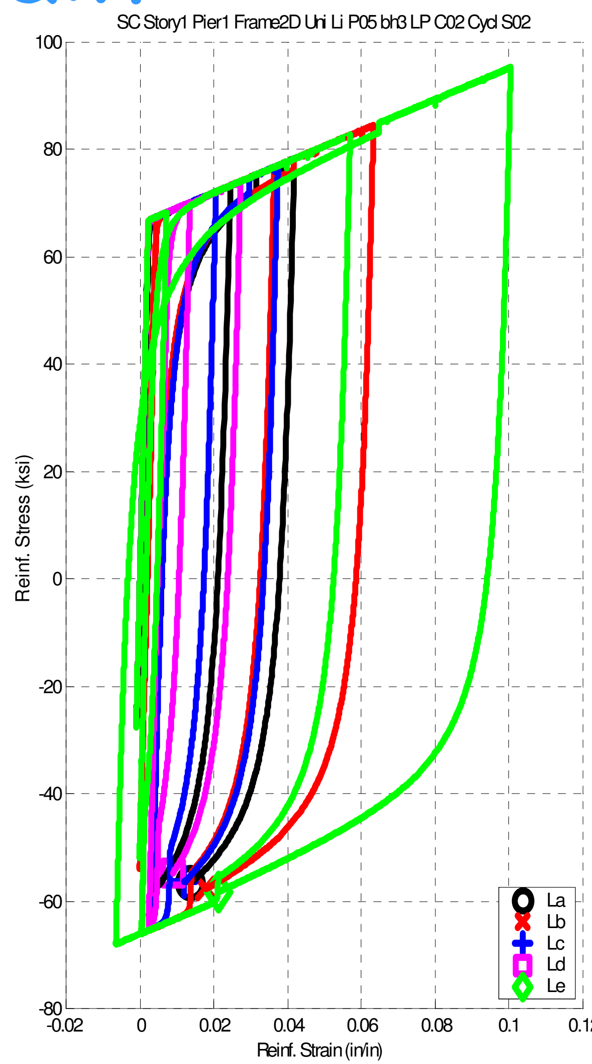
Silvia Mazzoni OpenSees Days 2006

Plastic-Hinge Length
Steel Stress vs. Strain

beamWithHinges



pushover



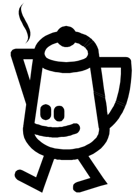
cyclic

Silvia Mazzoni OpenSees Days 2006

OpenSees

OpenSees NEESit

Summary



- Direct User Support (email + for
- Annual User/Developer Workshops
- Maintain Command-Language Manual
- Develop Examples Manual
- Develop Scripting Tools
- Comparison of OpenSees Models