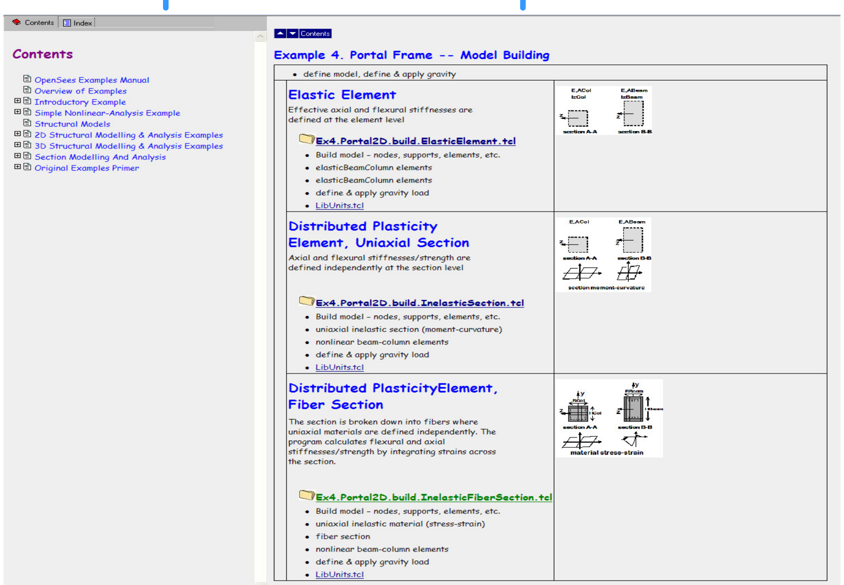
Structural Example – Reinforced-Concrete Frame: Building the Model

Silvia Mazzoni
OpenSees User-Support Manager
University of California, Berkeley

OpenSees User Workshop
8 September 2008

Example 4 in examples manual


Contents | Index

Contents

Contents

Example 4. Portal Frame -- Model Building

- define model, define & apply gravity

Elastic Element
Effective axial and flexural stiffnesses are defined at the element level

Ex4.Portal2D.build.ElasticElement.tcl
- Build model – nodes, supports, elements, etc.
- elasticBeamColumn elements
- elasticBeamColumn elements
- define & apply gravity load
- LibUnits.tcl

Distributed Plasticity Element, Uniaxial Section
Axial and flexural stiffnesses/strength are defined independently at the section level

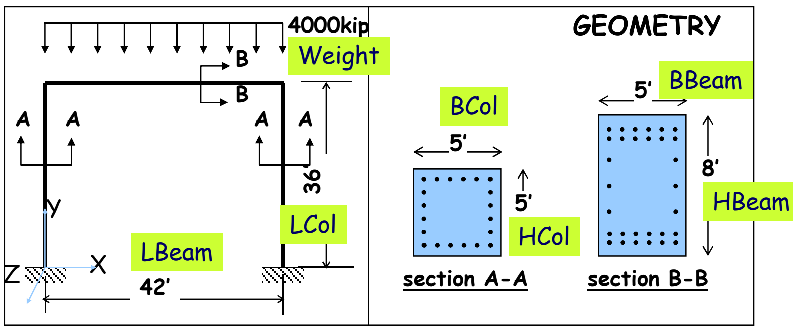
Ex4.Portal2D.build.InelasticSection.tcl
- Build model – nodes, supports, elements, etc.
- uniaxial inelastic section (moment-curvature)
- nonlinear beam-column elements
- define & apply gravity load
- LibUnits.tcl

Distributed PlasticityElement, Fiber Section
The section is broken down into fibers where uniaxial materials are defined independently. The program calculates flexural and axial stiffnesses/strength by integrating strains across the section.

Ex4.Portal2D.build.InelasticFiberSection.tcl
- Build model – nodes, supports, elements, etc.
- uniaxial inelastic material (stress-strain)
- fiber section
- nonlinear beam-column elements
- define & apply gravity load
- LibUnits.tcl

problem statement

- Reinforced-Concrete Portal Frame
- start with ALL elastic elements (At a more advanced level, these elements can be replaced by more refined element models)
- use kip, inch and sec as basic units



Silvia Mazzoni, OpenSees User Workshop, 2008

.....remember what I told you about Tcl?

- Tcl is a string based scripting language
- enables variables and variable substitution (use variables to define units!!!)
- Expression evaluation
- Array management
- Basic control structures (if , while, for, foreach)
- Procedures
- File manipulation

Silvia Mazzoni, OpenSees User Workshop, 2008

Ex4.Portal2D.build.InelasticFiberSection.tcl

```
# --------------------------------------------------------------------------------------------------
# Example4. 2D Portal Frame--  Build Model
# nonlinearBeamColumn element, inelastic fiber section
#     Silvia Mazzoni & Frank McKenna, 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
set dataDir Data;                       # set up name of data directory
file mkdir $dataDir;                    # create data directory
set GMdir "GMfiles";                    # ground-motion file directory
source LibUnits.tcl;                    # define basic and system units
```


LibUnits.tcl

```
# --------------------------------------------------------------------------
# LibUnits.tcl -- define system of units
#     Silvia Mazzoni & Frank McKenna, 2006
#

# 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 LunitTXT "inch";                    # define basic-unit text for output
set FunitTXT "kip";                     # define basic-unit text for output
set TunitTXT "sec";                     # define basic-unit text for output
set ft [expr 12.*$in];                  # define engineering units
set ksi [expr $kip/pow($in,2)];
set psi [expr $ksi/1000.];
set lbf [expr $psi*$in*$in];            # pounds force
set pcf [expr $lbf/pow($ft,3)];         # pounds per cubic foot
set psf [expr $lbf/pow($ft,3)];         # pounds per square foot
set in2 [expr $in*$in];                 # inch^2
set in4 [expr $in*$in*$in*$in];         # inch^4
set cm [expr $in/2.54];                 # centimeter
set PI [expr 2*asin(1.0)];              # define constants
set g [expr 32.2*$ft/pow($sec,2)];      # gravitational acceleration
set Ubig 1.e10;                         # a really large number
set Usmall [expr 1/$Ubig];              # a really small number
```

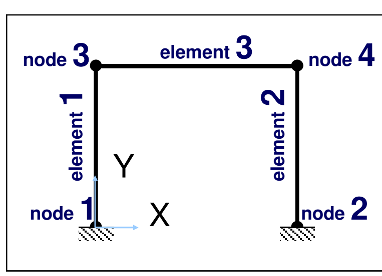
Silvia Mazzoni, OpenSees User Workshop, 2008

Geometry, Weight, Mass

```
# define GEOMETRY -------------------------------------------------------------
set LCol [expr 36*$ft];          # column length
set LBeam [expr 42*$ft];         # beam length
# define section geometry
set HCol [expr 5.*$ft];          # Column Depth
set BCol [expr 5.*$ft];          # Column Width
set HBeam [expr 8.*$ft];         # Beam Depth
set BBeam [expr 5.*$ft];         # Beam Width
# superstructure weight
set Weight [expr 2000.*$kip];

# calculated parameters
set PCol [expr $Weight/2];                                # nodal dead-load weight per column
set Mass [expr $PCol/$g];                                 # nodal mass
set MCol [expr 1./12.*($Weight/$LBeam)*pow($LBeam,2)];    # beam-end moment due to distributed load.
# calculated geometry parameters
set ACol [expr $BCol*$HCol];                              # cross-sectional area
set ABeam [expr $BBeam*$HBeam];
set IzCol [expr 1./12.*$BCol*pow($HCol,3)];               # Column moment of inertia
set IzBeam [expr 1./12.*$BBeam*pow($HBeam,3)];            # Beam moment of inertia
```


Nodes: Coords, BC, Mass

```
# nodal coordinates:
node 1 0 0;          # node#, X, Y
node 2 $LBeam 0
node 3 0 $LCol
node 4 $LBeam $LCol
# Single point constraints -- Boundary Conditions
fix 1 1 1 0;         # node DX DY RZ
fix 2 1 1 0;         # node DX DY RZ
fix 3 0 0 0
fix 4 0 0 0
# nodal masses:
mass 3 $Mass 0. 0.;       # node#, Mx My Mz, Mass=Weight/g, neglect rotational inertia at nodes
mass 4 $Mass 0. 0.
```



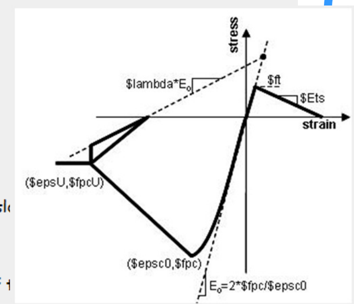
Concrete Material: Concrete02

Concrete02 Material -- Linear Tension Softening

This command is used to construct a uniaxial concrete material object with tensile strength and linear tension softening. T

uniaxialMaterial Concrete02 $matTag $fpc $epsc0 $fpcu $epsU $lambda $ft $Ets

$matTag	unique material object integer tag
$fpc	compressive strength*
$epsc0	strain at compressive strength*
$fpcu	crushing strength*
$epsU	strain at crushing strength*
$lambda	ratio between unloading slope at $epscu and initial sl
$ft	tensile strength
$Ets	tension softening stiffness (absolute value) (slope of t tension softening branch)



Concrete02 Material -- Material parameters

***NOTE:** Compressive concrete parameters should be input as negative values.

The initial slope for this model is (2*$fpc/$epsc0)

Steel Material: Steel02

Steel02 Material -- Giuffré-Menegotto-Pinto Model with Isotropic Strain Hardening

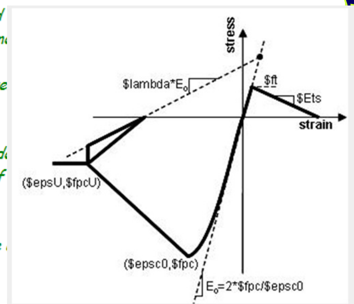
This command is used to construct a uniaxial Giuffre-Menegotto-Pinto steel material object with isotropic strain hardening.

uniaxialMaterial Steel02 $matTag $Fy $E $b $R0 $cR1 $cR2 <$a1 $a2 $a3 $a4 $sigInit>

$matTag	unique material object integer tag
$Fy	yield strength
$E	initial elastic tangent
$b	strain-hardening ratio (ratio between post-yield tangent and initial elastic tangent)
$R0, $cR1, $cR2	control the transition from elastic to plastic branches. Recommended values: $R0=between 10 and 20, $cR1=0.925, $cR2=0.15
$a1, $a2, $a3, $a4	isotropic hardening parameters: (optional, default: no isotropic hardening)

Materials

```
# MATERIAL parameters -------------------------------------------------------------------
set IDconcU 1;                          # material ID tag -- unconfined
set IDreinf 2;                          # material ID tag -- reinforcem
# nominal concrete compressive strength
set fc [expr -4.0*$ksi];                # CONCRETE Compressive Stre
set Ec [expr 57*$ksi*sqrt(-$fc/$psi)];  # Concrete Elastic Modulus
# unconfined concrete
set fc1U $fc;                           # UNCONFINED concrete (tod
set eps1U -0.003;                       # strain at maximum strength of
set fc2U [expr 0.2*$fc1U];              # ultimate stress
set eps2U -0.05;                        # strain at ultimate stress
set lambda 0.1;                         # ratio between unloading slope
# tensile-strength properties
set ftU [expr -0.14*$fc1U];             # tensile strength +tension
set Ets [expr $ftU/0.002];              # tension softening stiffness
# ----------
set Fy [expr 66.8*$ksi];                # STEEL yield stress
set Es [expr 29000.*$ksi];              # modulus of steel
set Bs 0.01;                            # strain-hardening ratio
set R0 18;                              # control the transition from elastic to plastic branches
set cR1 0.925;                          # control the transition from elastic to plastic branches
set cR2 0.15;                           # control the transition from elastic to plastic branches

uniaxialMaterial Concrete02 $IDconcU $fc1U $eps1U $fc2U $eps2U $lambda $ftU $Ets; # build cover concrete
uniaxialMaterial Steel02 $IDreinf $Fy $Es $Bs $R0 $cR1 $cR2;         # build reinforcement material
```



Concrete02 Material -- Material parameters

Section Command

section Command

This command is used to construct a SectionForceDeformation object, hereto referred to as Section, which represents force-de

What is a section?

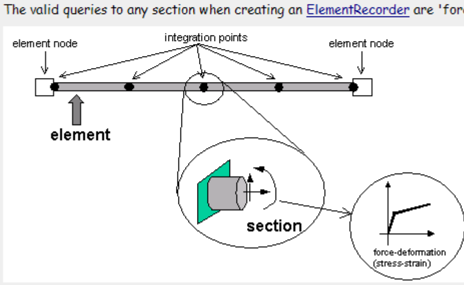
- A section defines the stress resultant force-deformation response at a cross section of a beam-column or plate element
- Types of sections:

• **Elastic** – defined by material and geometric constants

• **Resultant** – general nonlinear description of force-deformation response, e.g. moment-curvature

• **Fiber** – section is discretized into smaller regions for which the material stress-strain response is integrated to give resultant b

The valid queries to any section when creating an ElementRecorder are 'force' and 'deformation.'



- Elastic Section
- Uniaxial Section
- Fiber Section
- Section Aggregator
- Elastic Membrane Plate Section
- Plate Fiber Section
- Bidirectional Section
- Isolator2spring Section: Model to include buckling behavior of an elastomeric bearing

Beam Section: Elastic

Elastic Section

This command is used to construct an ElasticSection object.

```
section Elastic $secTag $E $A $Iz <$Iy $G $J>
```

$secTag	unique section object tag
$E	Young's Modulus
$A	cross-sectional area of section
$Iz	second moment of area about the local z-axis
$Iy	second moment of area about the local y-axis (optional, used for 3D analysis)
$G	Shear Modulus (optional, used for 3D analysis)
$J	torsional moment of inertia of section (optional, used for 3D analysis)

Column Section: Fiber Section

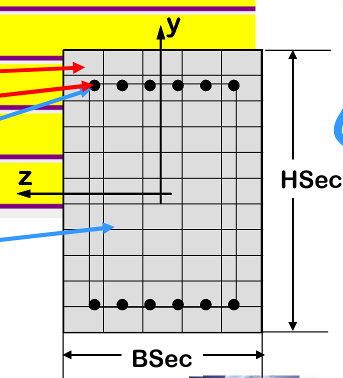
Fiber Section

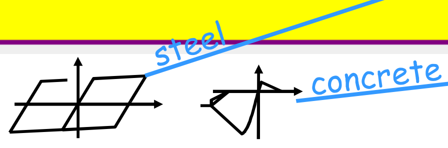
The FiberSection object is composed of Fiber objects.

A fiber section has a general geometric configuration formed by subregions of simpler, regular shapes (e.g. qu patch and layer (Circular Layer Command, Straight Layer Command) are used to define the discretization of th associated with uniaxialMaterial objects, which enforce Bernoulli beam assumptions.

The geometric parameters are defined with respect to a planar local coordinate system (y,z). See figures.

```
section Fiber $secTag {
fiber <fiber arguments>
patch <patch arguments>
layer <layer arguments>
}
```





material stress-strain

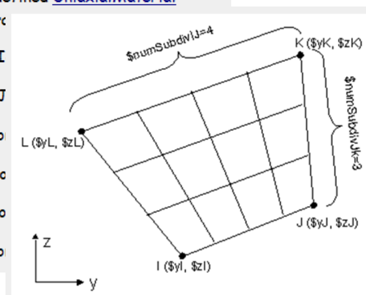
Patch Command: Concrete

Quadrilateral Patch Command

This command is used to construct a Patch object with a quadrilateral shape. The geometry of the pat
the four vertices is specified in sequence -- counter-clockwise.

patch quad $matTag $numSubdivIJ $numSubdivJK $yI $zI $yJ $zJ $yK $zK $yL $zL

$matTag		material integer tag of the previously-defined UniaxialMaterial object used to represent the stress-str
$numSubdivIJ		number of subdivisions (fibers) in the I
$numSubdivJK		number of subdivisions (fibers) in the J
$yI	**$zI**	y & z-coordinates of vertex I (local coo
$yJ	**$zJ**	y & z-coordinates of vertex J (local coo
$yK	**$zK**	y & z-coordinates of vertex K (local coo
$yL	**$zL**	y & z-coordinates of vertex L (local coo



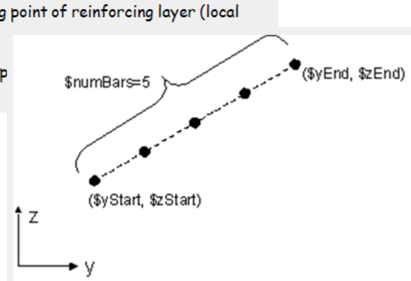
Layer Command: Steel

Straight Layer Command

This command is used to construct a straight layer of reinforcing bars.

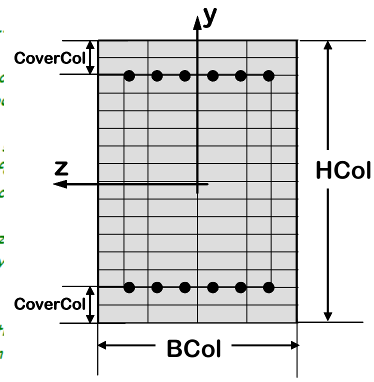
layer straight $matTag $numBars $areaBar $yStart $zStart $yEnd $zEnd

$matTag		material integer tag of the previously-defined UniaxialMaterial object used to represent the stress-strain for the area of the fiber
$numBars		number of reinforcing bars along layer
$areaBar		area of individual reinforcing bar
$yStart	**$zStart**	y and z-coordinates of starting point of reinforcing layer (local coordinate system)
$yEnd	**$zEnd**	y and z-coordinates of ending p coordinate system)



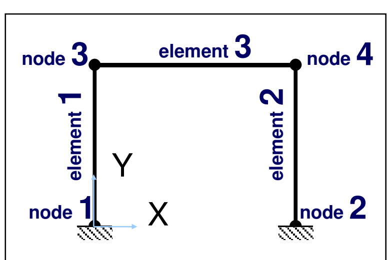
Column & Beam Sections

```
# Define ELEMENTS & SECTIONS -----------------------------
# symmetric section
set ColSecTag 1;                # assign a tag number to the co
set BeamSecTag 2;               # assign a tag number to the be
# define section geometry
set coverCol [expr 6.*$in];     # Column cover to reinforcing s
set numBarsCol 10;              # number of longitudinal-reinf
set barAreaCol [expr 2.25*$in2];  # area of longitudinal-reinforc
# RC section:
   set coverY [expr $HCol/2.0];  # The distance from the section z
   set coverZ [expr $BCol/2.0];  # The distance from the section y
   set coreY [expr $coverY-$coverCol]
   set coreZ [expr $coverZ-$coverCol]
   set nfY 16;           # number of fibers for concrete in y-directi
   set nfZ 4;         # number of fibers for concrete in z-direction
   section fiberSec $ColSecTag  {;    # Define the fiber section
    patch quadr $IDconcU $nfZ $nfY -$coverY $coverZ -$coverY -$coverZ $coverY -$coverZ $coverY $coverZ;
    layer straight $IDreinf $numBarsCol $barAreaCol -$coreY $coreZ -$coreY -$coreZ;   # top layer reinfocement
    layer straight $IDreinf $numBarsCol $barAreaCol  $coreY $coreZ  $coreY -$coreZ; # bottom layer reinforcement
    };  # end of fibersection definition

# BEAM section:
section Elastic $BeamSecTag   $Ec $ABeam $IzBeam;     # elastic beam section
```



Transformations and Elements

```
# define geometric transformation: performs a linear geometric transformation of
#     beam stiffness and resisting force from the basic system to the global-coordinate system
set ColTransfTag 1;          # associate a tag to column transformation
set BeamTransfTag 2;           # associate a tag to beam transformation (good practice to keep col and beam separate)
set ColTransfType Linear ;          # options, Linear PDelta Corotational
geomTransf $ColTransfType $ColTransfTag ;    # only columns can have PDelta effects (gravity effects)
geomTransf Linear $BeamTransfTag  ;

# element connectivity:
set numIntgrPts 5;                     # number of integration points for force-based element
element nonlinearBeamColumn 1 1 3 $numIntgrPts $ColSecTag $ColTransfTag;  # self-explanatory when using variables
element nonlinearBeamColumn 2 2 4 $numIntgrPts $ColSecTag $ColTransfTag;
element nonlinearBeamColumn 3 3 4 $numIntgrPts $BeamSecTag $BeamTransfTag;
```

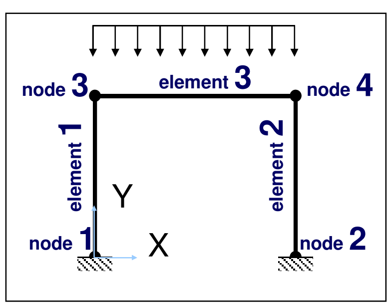


Recorders

```
# Define RECORDERS -------------------------------------------------------------
recorder Node -file $dataDir/DFree.out -time -node 3 4 -dof 1 2 3 disp;        # displacements of free nod
recorder Node -file $dataDir/DBase.out -time -node 1 2 -dof 1 2 3 disp;     # displacements of support noa
recorder Node -file $dataDir/RBase.out -time -node 1 2 -dof 1 2 3 reaction;     # support reaction
recorder Drift -file $dataDir/Drift.out -time -iNode 1 2 -jNode 3 4 -dof 1  -perpDirn 2 ;     # lateral drift
recorder Element -file $dataDir/FCol.out -time -ele 1 2 globalForce;     # element forces -- column
recorder Element -file $dataDir/FBeam.out -time -ele 3 globalForce;     # element forces -- beam
recorder Element -file $dataDir/ForceColSec1.out -time -ele 1 2 section 1 force;     # Column section forces, c
recorder Element -file $dataDir/DefoColSec1.out -time -ele 1 2 section 1 deformation;     # section deformc
recorder Element -file $dataDir/ForceColSec$numIntgrPts.out -time -ele 1 2 section $numIntgrPts force;     # se
recorder Element -file $dataDir/DefoColSec$numIntgrPts.out -time -ele 1 2 section $numIntgrPts deformation;
recorder Element -file $dataDir/ForceBeamSec1.out -time -ele 3 section 1 force;     # Beam section forces,
recorder Element -file $dataDir/DefoBeamSec1.out -time -ele 3 section 1 deformation;     # section deformc
recorder Element -file $dataDir/ForceBeamSec$numIntgrPts.out -time -ele 3 section $numIntgrPts force;     # se
recorder Element -file $dataDir/DefoBeamSec$numIntgrPts.out -time -ele 3 section $numIntgrPts deformation;   #
```


Gravity Load

```
# define GRAVITY -------------------------------------------------------------
set WzBeam [expr $Weight/$LBeam];
pattern Plain 1 Linear {
   eleLoad -ele 3 -type -beamUniform -$WzBeam ; # distributed superstructure-weight on beam
}
```

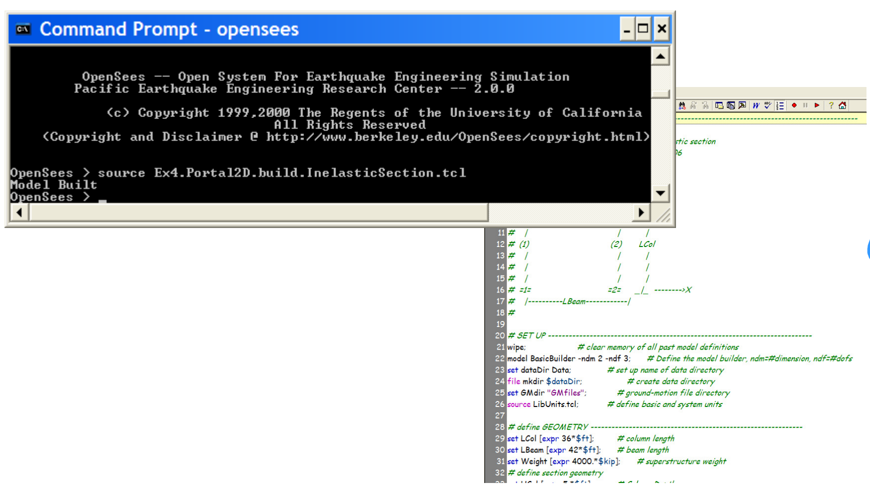


Gravity Analysis

```
# Gravity-analysis parameters -- load-controlled static analysis
set Tol 1.0e-8;          # convergence tolerance for test
constraints Plain;            # how it handles boundary conditions
numberer Plain;          # renumber dof's to minimize band-width (optimization), if you want to
system BandGeneral;      # how to store and solve the system of equations in the analysis
test NormDispIncr $Tol 6 ;       # determine if convergence has been achieved at the end of an iteration step
algorithm Newton;             # use Newton's solution algorithm: updates tangent stiffness at every iteration
set NstepGravity 10;       # apply gravity in 10 steps
set DGravity [expr 1./$NstepGravity];  # first load increment;
integrator LoadControl $DGravity;    # determine the next time step for an analysis
analysis Static;         # define type of analysis static or transient
analyze $NstepGravity;       # apply gravity
# ------------------------------------------------ maintain constant gravity loads and reset time to zero
loadConst -time 0.0

puts "Model Built"
```


Run OpenSees

source Ex4.Portal2D.build.InelasticSection.tcl



Silvia Mazzoni, OpenSees User Workshop, 2008