Advanced Modeling Techniques in OpenSees

Silvia Mazzoni
University of California, Berkeley

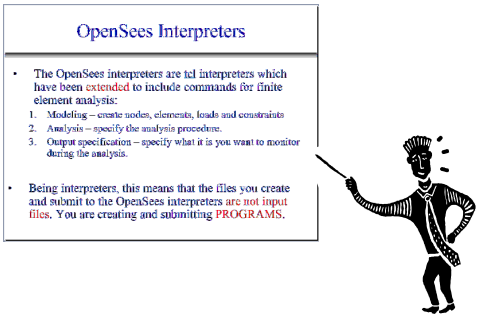
OpenSees Modeling Workshop

9 September 2008

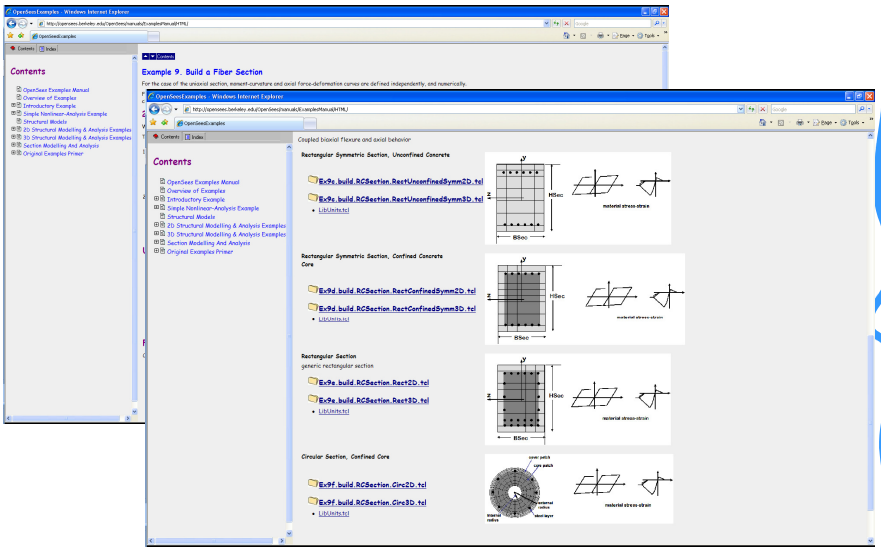
as Frank mentioned yesterday…

- In OpenSees you are:

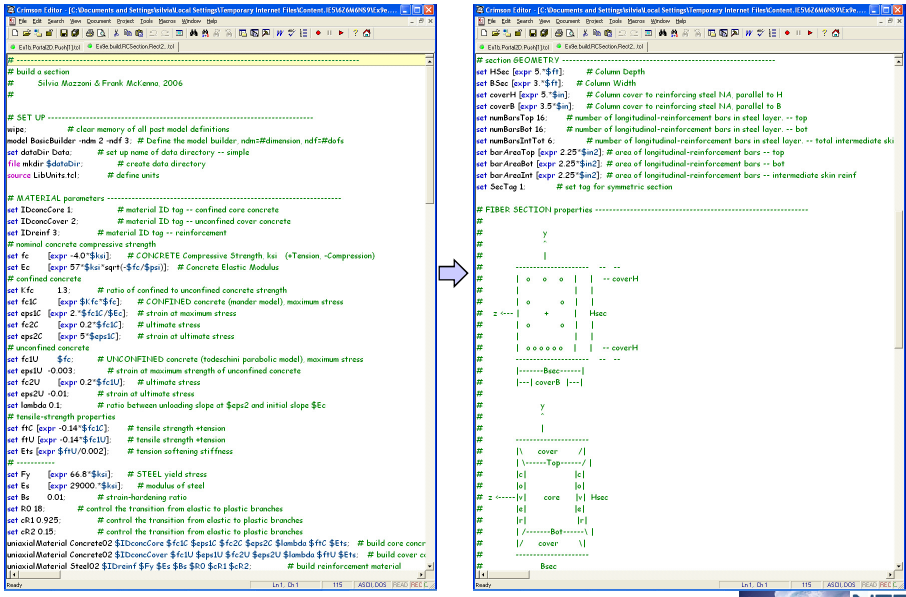
  RUNNING a program: a series of commands!



Section Analysis:Example 9



Ex9e.build.RCSection.Rect2D.tcl

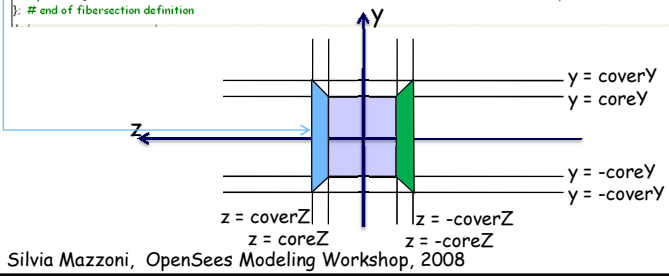


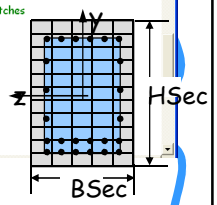
```
# build a section
#       Silvia Mazzoni & Frank McKenna, 2006
#
# SET UP ---------------------------------------------------------------------
wipe;                # clear memory of all past model definitions
model BasicBuilder -ndm 2 -ndf 3;  # Define the model builder, ndm=#dimension, ndf=#dofs
set dataDir Data;    # set up name of data directory -- simple
file mkdir $dataDir;         # create data directory
source LibUnits.tcl;         # define units

# MATERIAL parameters --------------------------------------------------------
set IDconcCore 1;            # material ID tag -- confined core concrete
set IDconcCover 2;           # material ID tag -- unconfined cover concrete
set IDreinf 3;               # material ID tag -- reinforcement
# nominal concrete compressive strength
set fc      [expr -40*$ksi];     # CONCRETE Compressive Strength, ksi   (+Tension, -Compression)
set Ec      [expr 57*$ksi*sqrt(-$fc/$psi)];  # Concrete Elastic Modulus
# confined concrete
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
# unconfined concrete
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 $ftU/0.002];    # tension softening stiffness
# ----------
set Fy      [expr 66.8*$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 $IDconcCore $fc1C $eps1C $fc2C $eps2C $lambda $ftC $Ets;  # build core concr
uniaxialMaterial Concrete02 $IDconcCover $fc1U $eps1U $fc2U $eps2U $lambda $ftU $Ets;  # build cover co
uniaxialMaterial Steel02 $IDreinf $Fy $Es $Bs $R0 $cR1 $cR2;        # build reinforcement material
```

```
# section GEOMETRY -----------------------------------------------------------
set HSec [expr 5.*$ft];      # Column Depth
set BSec [expr 3.*$ft];      # Column Width
set coverH [expr 5.*$in];    # Column cover to reinforcing steel NA, parallel to H
set coverB [expr 3.5*$in];   # Column cover to reinforcing steel NA, parallel to B
set numBarsTop 16;           # number of longitudinal-reinforcement bars in steel layer: -- top
set numBarsBot 16;           # number of longitudinal-reinforcement bars in steel layer: -- bot
set numBarsIntTot 6;         # number of longitudinal-reinforcement bars in steel layer: -- total intermediate ski
set barAreaTop [expr 2.25*$in2];  # area of longitudinal-reinforcement bars -- top
set barAreaBot [expr 2.25*$in2];  # area of longitudinal-reinforcement bars -- bot
set barAreaInt [expr 2.25*$in2];  # area of longitudinal-reinforcement bars -- intermediate skin reinf
set SecTag 1;                # set tag for symmetric section

# FIBER SECTION properties ---------------------------------------------------
#
#                y
#                ^
#                |
#  -------------------   --  --
#  | o    o    o  |   |    -- coverH
#  |                 |   |
#  | o              o |   |
#  |                 |   |
#  | o              o |   |  HSec
#  |                 |   |
#  | o  o  o  o  o  o |   |    -- coverH
#  -------------------   --  --
#  |-------Bsec------|
#  |---| coverB |---|
#
#                y
#                ^
#                |
#  ------------------
#  |\    cover    /|
#  | \-----Top----/ |
#  |c|            |c|
#  |o|            |o|
#  |v|    core    |v|  HSec
#  |e|            |e|
#  |r|            |r|
#  | /-------Bot------\ |
#  |/    cover      \|
#  ------------------
#
#                Bsec
```

Ex9e.build.RCSection.Rect2D.tcl

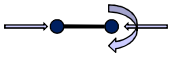
```
# Notes
#    The core concrete ends at the NA of the reinforcement
#    The center of the section is at (0,0) in the local axis system

set coverY [expr $HSec/2.0];   # The distance from the section z-axis to the edge of the cover concrete -- outer edge of cover concrete
set coverZ [expr $BSec/2.0];   # The distance from the section y-axis to the edge of the cover concrete -- outer edge of cover concrete
set coreY [expr $coverY-$coverH];   # The distance from the section z-axis to the edge of the core concrete --  edge of the core concrete/inner edge of cover concrete
set coreZ [expr $coverZ-$coverB];   # The distance from the section y-axis to the edge of the core concrete --  edge of the core concrete/inner edge of cover concrete;
set nfY 16;           # number of fibers for concrete in y-direction
set nfZ 4;            # number of fibers for concrete in z-direction
set numBarsInt [expr $numBarsIntTot/2]; # number of intermediate bars per side
section fiberSec $SecTag     {;  # Define the fiber section
    patch quadr $IDconcCore $nfZ $nfY -$coreY $coreZ -$coreY -$coreZ $coreY -$coreZ $coreY $coreZ;   # Define the core patch
    patch quadr $IDconcCover 1 $nfY -$coverY $coverZ -$coreY $coreZ $coreY $coreZ $coverY $coverZ;    # Define the four cover patches
    patch quadr $IDconcCover 1 $nfY -$coreY -$coreZ -$coverY -$coverZ $coverY -$coverZ $coreY -$coreZ
    patch quadr $IDconcCover $nfZ 1 -$coverY $coverZ -$coverY -$coverZ -$coreY -$coreZ -$coreY $coreZ
    patch quadr $IDconcCover $nfZ 1 $coreY $coreZ $coreY -$coreZ $coverY -$coverZ $coverY $coverZ
    layer straight $IDreinf $numBarsInt $barAreaInt  -$coreY $coreZ $coreY $coreZ;  # intermediate skin reinf. +z
    layer straight $IDreinf $numBarsInt $barAreaInt  -$coreY -$coreZ $coreY -$coreZ;   # intermediate skin reinf. -z
    layer straight $IDreinf $numBarsTop $barAreaTop $coreY $coreZ $coreY -$coreZ; # top layer reinfocement
    layer straight $IDreinf $numBarsBot $barAreaBot  -$coreY $coreZ  -$coreY -$coreZ; # bottom layer reinforcement
}; # end of fibersection definition
```





proc MomentCurvature2D

```
proc MomentCurvature2D { secTag axialLoad maxK {numIncr 100} } {
    ###########################################################
    # A procedure for performing section analysis (only does
    # moment-curvature, but can be easily modified to do any mode
    # of section reponse.)
    # MHS
    # October 2000
    # modified to 2D and to improve convergence by Silvia Mazzoni, 2006
    #
    # Arguments
    # secTag -- tag identifying section to be analyzed
    # axialLoad -- axial load applied to section (negative is compression)
    # maxK -- maximum curvature reached during analysis
    # numIncr -- number of increments used to reach maxK (default 100)
    #
    # Sets up a recorder which writes moment-curvature results to file
    # section$secTag.out ... the moment is in column 1, and curvature in column 2

    # Define two nodes at (0,0)
    node 1001 0.0 0.0
    node 1002 0.0 0.0
    # Fix all degrees of freedom except axial and bending
    fix 1001 1 1 1
    fix 1002 0 1 0
    # Define element
    #                tag ndI ndJ  secTag
    element zeroLengthSection  2001  1001  1002  $secTag
    # Create recorder
    recorder Node -file data/Mphi.out -time -node 1002 -dof 3 disp;    # output moment (col 1) & cur

    # Define constant axial load
    pattern Plain 3001 "Constant" {
        load 1002 $axialLoad 0.0 0.0
    }
    # Define analysis parameters
    integrator LoadControl 0 1 0 0
    system SparseGeneral -piv;    # Overkill, but may need the pivoting!
    test EnergyIncr  1.0e-9 10
    numberer Plain
    constraints Plain
    algorithm Newton
    analysis Static
    # Do one analysis for constant axial load
    analyze 1
```



```
    # Define reference moment
    pattern Plain 3002 "Linear" {
        load 1002 0.0 0.0 1.0
    }
    # Compute curvature increment
    set dK [expr $maxK/$numIncr]
    # Use displacement control at node 1002 for section analysis, dof 3
    integrator DisplacementControl 1002 3 $dK 1 $dK $dK
    # Do the section analysis
    set ok [analyze $numIncr]
    # ------------------------------------------------if convergence failure-------------------------
    set IDctrlNode 1002
    set IDctrlDOF 3
    set Dmax $maxK
    set Dincr $dK
    set TolStatic 1.e-9;
    set testTypeStatic EnergyIncr
    set maxNumIterStatic 6
    set algorithmTypeStatic Newton
    if {$ok != 0} {
        # if analysis fails, we try some other stuff, performance is slower inside this loop
        set Dstep 0.0;
        set ok 0
        while {$Dstep <= 1.0 && $ok == 0} {
            set controlDisp [nodeDisp $IDctrlNode $IDctrlDOF ]
            set Dstep [expr $controlDisp/$Dmax]
            set ok [analyze 1];                # this will return zero if no convergence problems were encountered
            if {$ok != 0} {;            # reduce step size if still fails to converge
                set Nk 4;          # reduce step size
                set DincrReduced [expr $Dincr/$Nk];
                integrator DisplacementControl  $IDctrlNode $IDctrlDOF $DincrReduced
                for {set ik 1} {$ik <=$Nk} {incr ik 1} {
                    set ok [analyze 1];                # this will return zero if no convergence problems were encountered
                    if {$ok != 0} {
                        # if analysis fails, we try some other stuff
                        # performance is slower inside this loop global maxNumIterStatic;     # max no. of iterations per
                        puts "Trying Newton with Initial Tangent .."
                        test NormDispIncr   $TolStatic       2000 0
                        algorithm Newton -initial
                        set ok [analyze 1]
                        test $testTypeStatic $TolStatic       $maxNumIterStatic    0
                        algorithm $algorithmTypeStatic
                    }
```

proc MomentCurvature2D

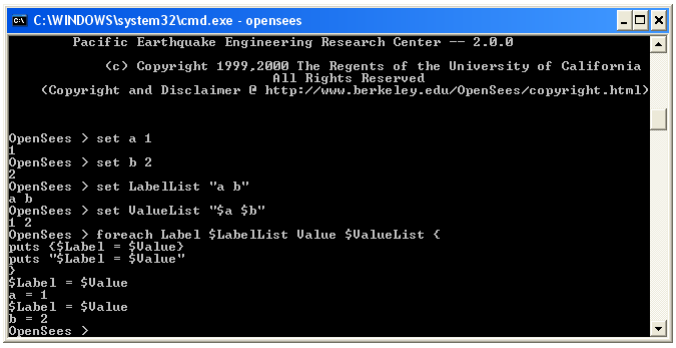
```
            if {$ok != 0} {
               puts "Trying Broyden .."
               algorithm Broyden 8
               set ok [analyze 1 ]
               algorithm $algorithmTypeStatic
            }
            if {$ok != 0} {
               puts "Trying NewtonWithLineSearch .."
               algorithm NewtonLineSearch 0.8
               set ok [analyze 1]
               algorithm $algorithmTypeStatic
            }
            if {$ok != 0} {;            # stop if still fails to converge
               puts [format $fmt1 "PROBLEM" $IDctrlNode $IDctrlDOF [nodeDisp $IDctrlNode $IDctrlDOF] $LunitTXT]
               return -1
            }; # end if
         }; # end for
         integrator DisplacementControl  $IDctrlNode $IDctrlDOF $Dincr;    # bring back to original increment
      }; # end if
   }; # end while loop
};     # end if ok !0
# ------------------------------------------------------------------------------------
global LunitTXT;                # load time-unit text
if {  [info exists LunitTXT] != 1} {set LunitTXT "Length"};      # set blank if it has not been defined previously.

set fmt1 "%s Pushover analysis: CtrlNode %.3i, dof %.1i, Curv=%.4f /%s";   # format for screen/file output of DONE/PROBLE
if {$ok != 0 } {
   puts [format $fmt1 "PROBLEM" $IDctrlNode $IDctrlDOF [nodeDisp $IDctrlNode $IDctrlDOF] $LunitTXT]
} else {
   puts [format $fmt1 "DONE"  $IDctrlNode $IDctrlDOF [nodeDisp $IDctrlNode $IDctrlDOF] $LunitTXT]
}
```


Ex9.analyze.MomentCurvature2D.tcl

```
# ------------------------------------------------------------------------------------
# Moment-Curvature analysis of section
#     Silvia Mazzoni & Frank McKenna, 2006
#

# define procedure
source MomentCurvature2D.tcl

# set AXIAL LOAD -----------------------------------------------------
set P [expr -1800*$kip];   # + Tension, - Compression

# set maximum Curvature:
set Ku [expr 0.01/$in];
set numIncr 100;   # Number of analysis increments to maximum curvature (default=100)
# Call the section analysis procedure
MomentCurvature2D $SecTag $P $Ku $numIncr
```

foreach Tcl command

- Implements a loop where the loop variable(s) take on values from one or more lists



Tcl list

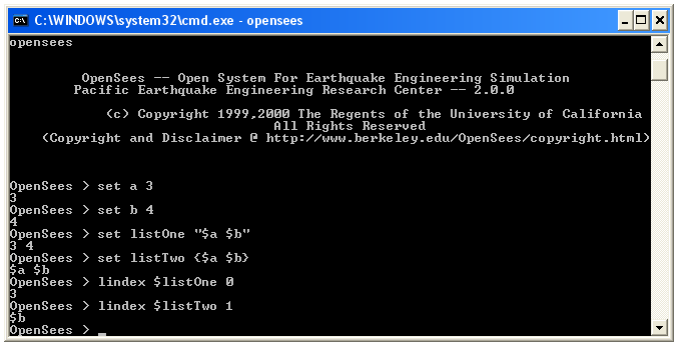
- Similar to a vector
- It is a "list" of elements
- Can manage with commands such as *lappend, lindex, linsert, llength, lrange, lrepeat, lreplace, lsearch, lset, lsort*
- *Very flexible size management (can have lists within lists)*
- *INDICE NUMBERING STARTS AT ZERO!*
- *Example:*

  set myList "item1 item2 item3 item4"

  set myList " item1 {item2a item2b} item3"

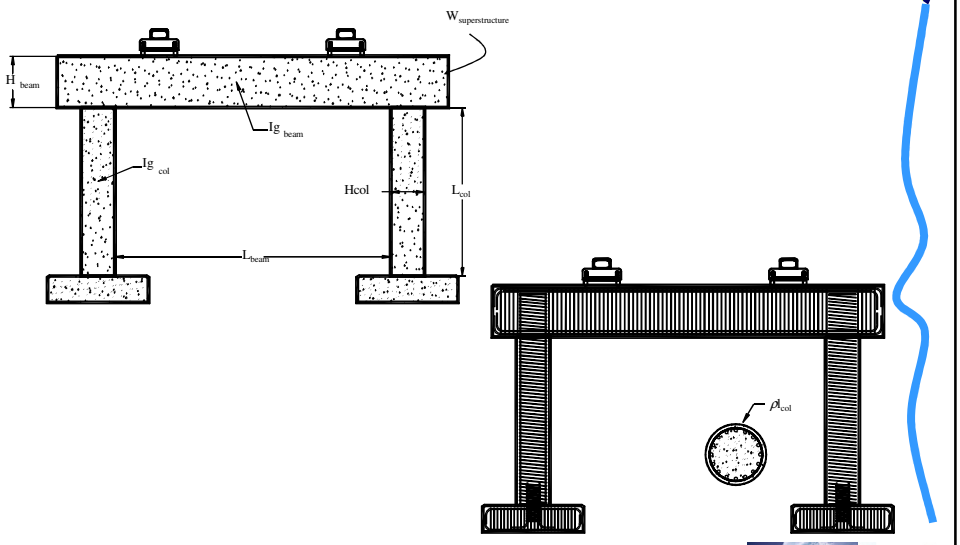
Tcl "" vs. {}

- Variables within "" are evaluated
- Variables within {} are NOT evaluated



Parameter Studies using OpenSees

application I:
parametric study of bridge-bent model



parameterList.tcl

```
set iBaseCase   {"PinBase" "FixBase" };

set iXframe     "1                  2";
set iHcol       "[expr 5*$ft]       [expr 6*$ft]";
set iLcol       "[expr 32*$ft]      [expr 36*$ft]";
set iHbeam      "[expr 6.*$ft]      [expr 8.*$ft]";
set iLbeam      "[expr 36.*$ft]     [expr 42.*$ft]";
set iGrhoCol    "0.0125             0.0175";
set iWeight     "[expr 1500.*$kip]  [expr 3000.*$kip]";

set iGMfact     "1.0 1.5 2.0";
```

boundary conditions

frame param's

ground-motion scaling

RUN.tcl (type 1)

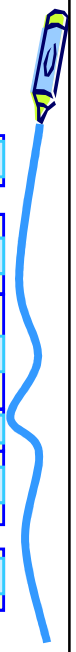
```
1. source Units.tcl;                    # define units
2. source parameterList.tcl;            # load up parameter values
3. source GMFiles.tcl;                  # load up ground-motion filenames
4. foreach   BaseCase $iBaseCase {
     foreach Xframe $iXframe Hcol $iHcol Lcol $iLcol Lbeam $iLbeam GrhoCol $iGrhoCol
             Weight $iWeight GMfact $iGMfact Hbeam $iHbeam Bbeam $iBbeam {      FRAME
       set ANALYSIS "Static";                                                   STATIC
       source Analysis.tcl
       set ANALYSIS "Dynamic";                                                  DYNAMIC
       foreach GroundFile $iGroundFile
         source Analysis.tcl                                                    GROUND MOTION
       }
     }
   }
```


RUN.tcl (type 2)

```
source Units.tcl;      source ParamList.tcl;  source GMFiles.tcl;

foreach BaseCase $iBaseCase {
  set Xframe 0;
  foreach Hcol $iHcol {
   foreach Lcol $iLcol {
    foreach Hbeam $iHbeam {
     foreach Lbeam $iLbeam {
      foreach GrhoCol $iGrhoCol {
       foreach Weight $iWeight {
        foreach GMfact $iGMfact {
          set Xframe [expr $Xframe+1];

          set ANALYSIS "Static";
          source singleAnalysis.tcl

          set ANALYSIS "Dynamic";
            foreach GroundFile $iGroundFile
              source singleAnalysis.tcl
            }
  }; }; }; }; }; }; };};};
```

singleAnalysis.tcl

```
1.  model basic -ndm 3 -ndf 6                    <- create model builder

2.  source  units.tcl;
3.  source  parameters.tcl;                      <- set up parameters and variables

4.  source  nodalmesh.tcl;
5.  source  materials.tcl;                       <- set up structural model
6.  source  elements.tcl;

7.  source  output.tcl;                          <- specify data output

8.  source  gravity.tcl;
9.  source  lateral.tcl;                         <- apply loading

10. wipeanalysis                                 <- clear memory
```



parameters.tcl

GEOMETRY

```
1.  set Rcol      [expr $Hcol/2];                 # COLUMN radius                    <- column
2.  set Acol      [expr $PI*pow($Rcol,2)];        # column cross-sectional area
3.  set cover     [expr $Hcol/15];                # column cover width
4.  set IgCol     [expr $PI*pow($Rcol,4)/4];      # column gross moment of inertia, uncracked
5.  set IyCol     $IgCol;                         # elastic-column properties
6.  set IzCol     $IgCol;                         # elastic-column properties
7.  set IzBeam    [expr $GIbIc*$IgCol];           # BEAM gross moment of inertia -   <- beam
8.  set Hbeam     [expr 8*$ft];                   # beam depth, not really used
9.  set Bbeam     [expr $IzBeam*12/pow($Hbeam,3)];# beam width not used
10. set IyBeam    [expr $Hbeam*pow($Bbeam,3)/12]; # beam gross moment of inertia--vert Y
11. set Abeam     [expr $Hbeam*$Bbeam*10000];     # beam cross-sectional area
12. set GLbLc     [expr $Lbeam/$Lcol];            # beam-to-column length ratio
```

output.tcl

```
# Record nodal displacements -NODAL DISPLACEMENTS
set filename0 "data/$BaseCase/"
set filename1 DStatFrame[expr $Xframe]
set filename2 GM$GroundFile
set iNode "3 4";

foreach xNode $iNode {
        set filename3 Node$xNode
        set filename $filename0$filename3$filename1$filename2

        recorder Node $filename.out disp -node $xNode -dof 1 2  6;

};      # end of xNode
```

directory

frame ID

ground motion

node no.

example filename: data/Pinbase/Node3DStatFrame1ElCentro.out

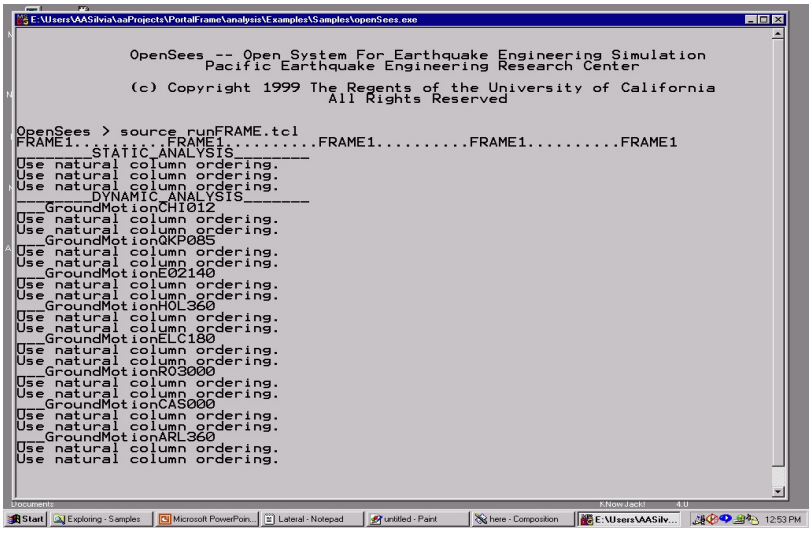
generating matlab input

```
set datadir "Data/"
# Open output file for writing
set outFileID [open Data/DataFrame$Xframe.m w]

puts $outFileID  "Xframe($Xframe) = $Xframe;";  # frame ID
puts $outFileID  "Hcol($Xframe) = $Hcol;"       ;# column diameter
puts $outFileID  "Lcol($Xframe) = $Lcol;" ;     # column length
puts $outFileID  "Lbeam($Xframe) = $Lbeam;";    # beam length
puts $outFileID  "Hbeam($Xframe) = $Hbeam;";    # beam depth
puts $outFileID  "Bbeam($Xframe) = $Bbeam;";    # beam width
puts $outFileID  "GrhoCol($Xframe) = $GrhoCol;"; # column long.-steel ratio
puts $outFileID  "GPcol($Xframe) = $GPcol;";    # Col.axial load:strength
puts $outFileID  "GMfact($Xframe) = $GMfact;"; # ground-mot. scaling fact
puts $outFileID  "Acol($Xframe) = $Acol;";      # column cross-sect. area
puts $outFileID  "Weight($Xframe) = $Weight;";  # superstructure weight
```

dataframe3.m

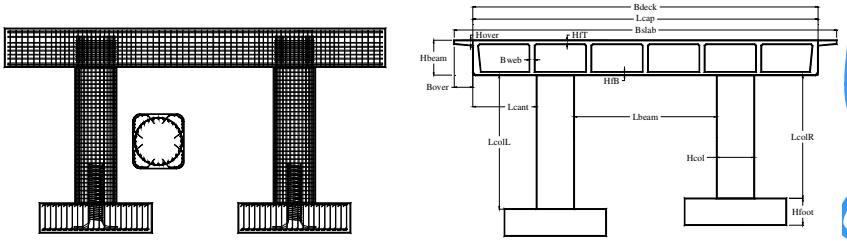
- Xframe(3) = 3;
- Hcol(3) = 78.0;
- Lcol(3) = 432.0;
- Lbeam(3) = 432.0;
- Hbeam(3) = 96.0;
- Bbeam(3) = 78.0;
- GrhoCol(3) = 0.0125;
- GPcol(3) = 0.0570754682058;
- GMfact(3) = 1.5;
- Acol(3) = 4778.36242611;
- Weight(3) = 3000.0;

analyses running

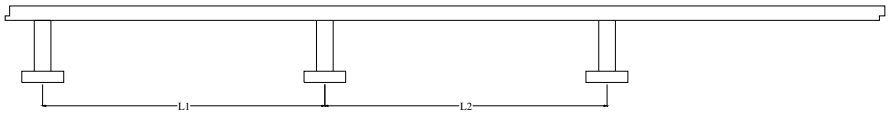


application II:
3D model of bridge frame

Cypress-Street Viaduct Replacement Structure



bent detail



frame detail

nodalmesh.tcl – a number of bents

```
set iNode0   "100   200   300"
set iZbent "[expr 0*$ft] [expr 200*$ft] [expr 400*$ft]"
```

define nodes of each bent

```
foreach Node0 $iNode0 Zoffset $iZbent {
      # Define nodes  ------ frame is in X-Y plane (X-horizontal, Y-vertical)
      #     tag          X              Y            Z     <-mass MX MY MZ RX  RY  RZ> (nodal masses)
      node [expr $Node0 + 1]    [expr -$Lbeam/2.]   [expr -$LcolL]  $Zoffset
      node [expr $Node0 +  2]   [expr  $Lbeam/2.]   [expr -$LcolR]  $Zoffset
      node [expr $Node0 +  3]   [expr -$Lbeam/2.]   0              $Zoffset  -mass $Mnode 0.0 0.0 0.0 0.0 0.0
      node [expr $Node0 +  4]   [expr  $Lbeam/2.]   0              $Zoffset  -mass $Mnode 0.0 0.0 0.0 0.0 0.0
      node [expr $Node0 +  5]   [expr -$Lbeam/2.- $Lcant]    0     $Zoffset;  #overhang
      node [expr $Node0 +  6]   [expr  $Lbeam/2.+$Lcant]     0     $Zoffset;
      node [expr $Node0 +  13]  [expr -$Lbeam/2.]   0              $Zoffset;
      node [expr $Node0 +  14]  [expr  $Lbeam/2.]   0              $Zoffset
      node [expr $Node0 +  23]  [expr -$Lbeam/2.]   0              $Zoffset;
      node [expr $Node0 +  24]  [expr  $Lbeam/2.]   0              $Zoffset


      # Boundary conditions    # node DX DY DZ RX RY RZ     # 1: fixed, 0: released
      fix  [expr $Node0 + 1]          1  1  1  0  1  0;          # pin support
.....
      #            5------4------3------------3------------4------5------6
      #                          |                          |
      #                          |                          |
      #                        --1--                      --2--                ^ y
      #                          |                          |                  |
      #                          |                          |                  |
      #                          1                          2                  ----------> X
}
```

elements.tcl

elem. #

node i node j

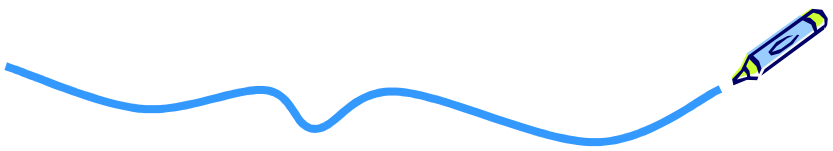
define elements of each bent

```
set iNode0   "100  200   300";
set iElem0   "100  200   300"

foreach Node0 $iNode0 Elem0 $iElem0 {
    # Define COLUMNS
    element nonlinearBeamColumn [expr $Elem0 + 1]  [expr $Node0 + 1]  [expr $Node0 +3]  $np   $IDcolSec   $ZZ;
    element nonlinearBeamColumn [expr $Elem0 + 2]  [expr $Node0 + 2]  [expr $Node0 +4]  $np   $IDcolSec   $ZZ;

    #IDphtMat  has been defined in parameters.tcl and materials.tcl  -- TOP PLASTIC HINGE, connects column to joint
    rotSpringDOF6 [expr $Elem0 + 11]  [expr $Node0 +3]  [expr $Node0 +13]   $IDphtMat   [expr $Node0 +3];
    rotSpringDOF6 [expr $Elem0 + 12]  [expr $Node0 +4]  [expr $Node0 +14]   $IDphtMat   [expr $Node0 +4]

    #IDjointMat  has been defined in parameters.tcl and materials.tcl -- JOINT, connects hinge to beam
    rotSpringDOF6 [expr $Elem0 + 21]  [expr $Node0 +13] [expr $Node0 +23]   $IDjointMat  [expr $Node0 +3];
    rotSpringDOF6 [expr $Elem0 + 22]  [expr $Node0 +14] [expr $Node0 +24]   $IDjointMat  [expr $Node0 +4]

    # Define BEAM
    element elasticBeamColumn [expr $Elem0 + 3]  [expr $Node0 +23] [expr $Node0 +24] $Abeam $Ec $GJ 1.0 $IyBeam $IzBeam $ZZ
    element elasticBeamColumn [expr $Elem0 + 4]  [expr $Node0 +5]  [expr $Node0 +23] $Abeam $Ec $GJ 1.0 $IyBeam $IzBeam $ZZ
    element elasticBeamColumn [expr $Elem0 + 5]  [expr $Node0 +24] [expr $Node0 +6]  $Abeam $Ec $GJ 1.0 $IyBeam $IzBeam $ZZ

};  # end foreach node and element in a bent
```

connect bents

```
element elasticBeamColumn   501   [expr $Node0Bent1 +3]   [expr $Node0Bent2 +3]   $Abeam $Ec $GJ 1.0 $U $U $YY
element elasticBeamColumn   502   [expr $Node0Bent2 +3]   [expr $Node0Bent3 +3]   $Abeam $Ec $GJ 1.0 $U $U $YY
element elasticBeamColumn   503   [expr $Node0Bent1 +4]   [expr $Node0Bent2 +4]   $Abeam $Ec $GJ 1.0 $U $U $YY
element elasticBeamColumn   504   [expr $Node0Bent2 +4]   [expr $Node0Bent3 +4]   $Abeam $Ec $GJ 1.0 $U $U $YY
```


2D Frame



Build Frame 2D

```
# ------------------------------------------------------------------------------------------------------------------------
# buildFrame2D.tcl: generates frame nodes/materials/sections/elements
#          by Silvia Mazzoni, 2005
#
#  41_____(241)___42_____(242)_____43_____(243)_____44_____(244)_____45          _
#  |         BS4    |         BS4    |         BS4    |         BS4    |          |
#  |(141)           |(142)           |(143)           |(144)           |(145)     LColTop
#  | CS6            | CS4            | CS4            | CS4            | CS6      |
#  31_____(231)___32_____(232)_____33_____(233)_____34_____(234)_____35          _
#  |         BS3    |         BS3    |         BS3    |         BS3    |          |
#  |(131)           |(132)           |(133)           |(134)           |(135)     LColTop
#  | CS6            | CS4            | CS4            | CS4            | CS6      |
#  21_____(221)___22_____(222)_____23_____(223)_____24_____(224)_____25          _
#  |         BS2    |         BS2    |         BS2    |         BS2    |          |
#  |(121)           |(122)           |(123)           |(124)           |(125)  LColTop
#  | CS5            | CS3            | CS3            | CS3            | CS5      |
#  11_____(211)___12_____(212)_____13_____(213)_____14_____(214)_____15          _
#  |         BS1    |         BS1    |         BS1    |         BS1    |          |
#  |                |                |                |                |          |
#  |(111)           |(112)           |(113)           |(114)           |(115)  LColBot
#  | CS1            | CS2            | CS2            | CS2            | CS1      |
#  |                |                |                |                |          |
# === 1           === 2            === 3            === 4            === 5      -
#
#  |----------LBeam------|----------LBeam----------|---------LBeam--------------|---------LBeam--------------|
#
#
```


Define Geometry

```
model BasicBuilder -ndm 2 -ndf 3
variable problemSize Large;     # option, Large or Small (less then 10 nodes) -- used to
source Libraries.tcl;  # set up all variables, procs and utilities libraries

# define frame geometry
set LColBot [expr 14*$ft];
set LColTop [expr 13*$ft];
set LBeam [expr 30*$ft];
set typicalStoryHeight $LColTop;
set DincrStatic [expr 0.2*$in];    # displacement increment for static pushover/cycli
set DincrStatic [expr 0.1*$in];    # displacement increment for static pushover/cycli
set DincrStatic [expr 0.05*$in];   # displacement increment for static pushover/cy
set DincrStatic [expr 0.01*$in];   # displacement increment for static pushover/cy

set iLCol "$LColBot $LColTop $LColTop $LColTop "
set iLBeam "$LBeam $LBeam $LBeam $LBeam"
set Nstory [llength $iLCol];
set Nbay [llength $iLBeam];
set Npier [expr $Nbay+1];

set iYlevel ""
set Ylevel 0.0
for {set j 1} {$j <= $Nstory} {incr j 1} {
   set Ylevel [expr $Ylevel + [lindex $iLCol [expr $j-1]]];
   lappend iYlevel $Ylevel;          # height of level i from ground
}

# Define nodes
set iSupNode "";   # define support nodes for multiple-support excitation or reactions
for {set i 1} {$i <= [expr $Nbay+1]} {incr i 1} {
   set Xcoord [expr ($i-1)*$LBeam]
   node $i        $Xcoord   0.
   lappend iSupNode $i;
   for {set j 1} {$j <= $Nstory} {incr j 1} {
      set jcount [expr $j-1]
      set Ylevel [lindex $iYlevel $jcount]
      node [expr $i+10*$j]   $Xcoord   $Ylevel
   }
}
```

```
set iIDpushNode "";     # define nodes where to apply lateral load for static pushover int
for {set j 1} {$j <= $Nstory} {incr j 1} {
   set nodeID [expr 1+10*$j]
   lappend iIDpushNode $nodeID
}

set IDbaseNode 1;    # define displacement base node
set IDctrlNode [lrange $iIDpushNode  end end];   # define control node for pushover an
set Ldrift [lrange $iYlevel end end];     # for displacement-control static pushover analy
set IDctrlDOF 1;  # lateral dof

# Single point constraints -- Boundary Conditions
for {set i 1} {$i <= [expr $Nbay+1]} {incr i 1} {
   fix  $i   1 1 1
}

# REINFORCED-CONCRETE material properties
source $LibDir/LibMaterialsRC.tcl;     # baseline RC materials
source $LibDir/LibMaterialsRCVariations.tcl;   # variations on the baseline RC materials

# Select material types for parameter study
set IDconcCore [string map $keyIDconcCore   $ConcType  ];
set IDconcCover [string map $keyIDconcCover $ConcType  ];
set IDreinf [string map $keyIDreinf $SteelType];

# REINFORCED-CONCRETE section properties
# numbers of fiber
set nfCoreY 16;    # number of fibers in the core patch in the y direction
set nfCoreZ 2;     # number of fibers in the core patch in the z direction
set nfCoverY 16;      # number of fibers in the cover patches with long sides in the y dire
set nfCoverZ 2;       # number of fibers in the cover patches with long sides in the z dire
```

Define Column Sections

```
# columns: CS1, CS2, CS3, CS4, CS5, CS6
set IDColSec0 100;      # starting ID number for column sections
set CS1 [set icount [expr $IDColSec0 +1]];
set CS2 [incr icount 1]
set CS3 [incr icount 1]
set CS4 [incr icount 1]
set CS5 [incr icount 1]
set CS6 [incr icount 1]
set coverCol [expr 2.6*$in];     # cover from face to centroid of longitudinal reinforcement.
set iIDsecCol " $CS1 $CS2 $CS3 $CS4 $CS5 $CS6"
set iHCol "[expr 30*$in] [expr 40*$in] [expr 33*$in] [expr 34*$in] [expr 30*$in] [expr 28*$in] "
set iBCol "[expr 30*$in] [expr 30*$in] [expr 30*$in] [expr 30*$in] [expr 30*$in] [expr 24*$in] "
set ikBarCol    "#9 #10 #10 #9 #9 #8 "; # bar size
set iNbarBotCol " 6 6 5 5 5 5 ";   # number of bottom longitudinal reinforcing bars in section
set iNbarTopCol " 6 6 5 5 5 5 ";   # number of top longitudinal reinforcing bars in section
set iNbarIntCol " 6 6 6 6 6 6 ";   # number of intermediate-reinforcing bars in section (2 bars per layer) -- different from Curt in specification
set iDbCol ""
set iAbCol ""
foreach kBarCol $ikBarCol {
   lappend iDbCol [procGetDb $kBarCol]
   lappend iAbCol [procGetAb $kBarCol]
}
set iIzCol ""
set iPo ""
foreach HCol $iHCol BCol $iBCol AbCol $iAbCol NbarBotCol $iNbarBotCol NbarTopCol $iNbarTopCol NbarIntCol $iNbarIntCol IDsecCol $iIDsecCol {
   set AgCol [expr $HCol*$BCol];
   set AsBotCol [expr $NbarBotCol*$AbCol];
   set AsTopCol [expr $NbarTopCol*$AbCol];
   set IzCol [expr 1./12*$BCol*pow($HCol,3) + $Es/$Ec*($AsBotCol+$AsTopCol)*pow(($HCol/2-$coverCol),2)];
   set PoCol [expr -$AgCol*$fc];
   lappend iIzCol $IzCol
   lappend iPoCol  $PoCol
   procRCrectSection $IDsecCol $HCol $BCol $coverCol $coverCol $IDconcCore  $IDconcCover $IDreinf $NbarBotCol $AbCol $NbarTopCol $AbCol \
      $NbarIntCol $AbCol $nfCoreY $nfCoreZ $nfCoverY $nfCoverZ
}
set columnSets "$CS1 $CS2 $CS2 $CS2 $CS2 $CS1
$CS5 $CS3 $CS3 $CS3 $CS5
$CS6 $CS4 $CS4 $CS4 $CS6
$CS6 $CS4 $CS4 $CS4 $CS6";      # where each column goes in the frame.
```

Define Beam Sections

```
# ------------------------------------------------------------------------
# beams: BS1, BS2, BS3, BS4
set IDBeamSec0 200;      # starting ID number for beam sections
set BS1 [set icount [expr $IDBeamSec0+1] ];
set BS2 [incr icount 1]
set BS3 [incr icount 1]
set BS4 [incr icount 1]
set coverBeam [expr 2.6*$in];      # cover from face to centroid of longitudinal reinforcement.
set iIDsecBeam " $BS1 $BS2 $BS3 $BS4"
set iHBeam "[expr 42*$in] [expr 36*$in] [expr 32*$in] [expr 32*$in]"
set iBBeam "[expr 24*$in] [expr 24*$in] [expr 24*$in] [expr 24*$in]"
set ikBarLongBeam "#9 #9 #9 #9 ";   # bar size for longitudinal reinforcement
set ikBarIntBeam "#4 #4 #4 #4 ";    # bar size for intermediate layers
set iNbarBotBeam " 8 6 5 4 ";  # number of bottom longitudinal reinforcing bars in section
set iNbarTopBeam " 10 8 7 6 "; # number of top longitudinal reinforcing bars in section
set iNbarIntBeam " 4 4 2 2 ";  # number of intermediate-reinforcing bars in section (2 bars per layer) -- 2x Curt
set iDbLongBeam "";set iAbLongBeam "";set iDbIntBeam "";set iAbIntBeam ""; # zero these values, as we append after
foreach kBarLongBeam $ikBarLongBeam kBarIntBeam $ikBarIntBeam {
   lappend iDbLongBeam [procGetDb $kBarLongBeam]
   lappend iAbLongBeam [procGetAb $kBarLongBeam]
   lappend iDbIntBeam [procGetDb $kBarIntBeam]
   lappend iAbIntBeam [procGetAb $kBarIntBeam]
}
set iIzBeam ""
set iPoBeam ""
foreach HBeam $iHBeam BBeam $iBBeam AbLongBeam $iAbLongBeam AbIntBeam $iAbIntBeam NbarBotBeam $iNbarBotBeam \
         NbarTopBeam $iNbarTopBeam NbarIntBeam $iNbarIntBeam IDsecBeam $iIDsecBeam {
   set AgBeam [expr $HBeam*$BBeam];
   set AsBotBeam [expr $NbarBotBeam*$AbLongBeam];
   set AsTopBeam [expr $NbarTopBeam*$AbLongBeam];
   set IzBeam [expr 1./12*$BBeam*pow($HBeam,3) + $Es/$Ec*($AsBotBeam+$AsTopBeam)*pow(($HBeam/2-$coverBeam),2)];
   set PoBeam [expr -$AgBeam*$fc];
   lappend iIzBeam $IzBeam
   lappend iPoBeam $PoBeam
   procRCrectSection $IDsecBeam $HBeam $BBeam $coverBeam $coverBeam $IDconcCore  $IDconcCover $IDreinf $NbarBotBeam \
         $AbLongBeam $NbarTopBeam $AbLongBeam $NbarIntBeam $AbIntBeam $nfCoreY $nfCoreZ $nfCoverY $nfCoverZ
}

set BeamSets "$BS1 $BS1 $BS1 $BS1
$BS2 $BS2 $BS2 $BS2
$BS3 $BS3 $BS3 $BS3
$BS4 $BS4 $BS4 $BS4 ";      # where each beam goes in the frame, each level and bay
```

Define Elements

```
# ------------------------------------------------------------------
# Define ELEMENTS
set np 5;           # number of integration points for nonlinearBeamColumn element
set IDcolTrans [set icount 1];       # associate a tag to column transformation
set IDbeamTrans [incr icount 1];     # associate a tag to beam transformation
geomTransf $Gtransf $IDcolTrans ;    # COLUMN! transformation
geomTransf Linear $IDbeamTrans ;     # BEAM! Linear transformation: no second-order effects

set IDColElt0 100;
set IDBeamElt0 200;

# columns
set ColElemType $ElemType;
for {set j 1} {$j <= $Nstory} {incr j 1} {
   for {set i 1} {$i <= [expr $Npier]} {incr i 1} {
      set elID [expr $IDColElt0+$i+($j)*10];
      set nodeI [expr $i+($j-1)*10];
      set nodeJ [expr $i+$j*10];
      set NN [expr $Npier]
      set secIDindex [expr ($j-1)*$NN+$i];
      set secID [lindex $columnSets [expr $secIDindex-1]];

      element nonlinearBeamColumn $elID $nodeI $nodeJ $np $secID $IDcolTrans ; # actually build element
   }; # end for i= row of columns
}; # end for j= column of columns


# ---------------------------------
# build beams
for {set j 1} {$j <= $Nstory} {incr j 1} {
   for {set i 1} {$i <= $Nbay} {incr i 1} {
      set elID [expr $IDBeamElt0+$i+$j*10];
      set nodeI [expr $i+$j*10];
      set nodeJ [expr $i+$j*10+1];
      set NN $Nbay;

      set keyBeamArgs {elasticBeamColumn $BeamArgumentsEBC  nonlinearBeamColumn $BeamArgumentsNBC}
      set BeamArguments [string map $keyBeamArgs $BeamElemType]

      element nonlinearBeamColumn $elID $nodeI $nodeJ $np $secID $IDbeamTrans ; # actually build element
   }; # end for i= row of beams
}; # end for j= column of beams
```

..... and so on.....

conclusions

- advantages of Tcl scripting language:
  - simplify parameter studies
  - simplify error check
  - generate a new input file while using components of a previously-generated and tested input file
  - improve collaboration