School of Architecture and Civil Engineering
厦门大学 建筑与土木工程学院

OpenSees-Snopt: Framework for Finite Element Based Optimization

by

Quan Gu	(Xiamen University)
Joel P. Conte	(UCSD)
Michele Barbato	(LSU)
Philip E. Gill	(UCSD)
Frank McKenna	(UCB)

OpenSees day 2012, Aug 15-16, UC Berkeley

Outline

1. Introduction to OpenSees-SNOPT framework
2. Application examples
3. TCL commands
4. How to record responses and modify model parameters

OpenSees in China

Major Research Program of NSFC: Disaster Evolution of Civil Infrastructure under Strong Earthquake and Wind (DECISEW)

（2008.1-2015.12，30 million UCD for 85 projects，
＋ 4 China-Japan Collaboration Projects 0.5 million each
＋ 6 China-US Collaboration Projects 0.5 million each ）

Investigations on damage evolution and collapse mechanism of civil infrastructures under extreme loadings, i.e. earthquakes and typhoons

Director: Jinping Ou(欧进萍)
Dalian University of Technology (DUT)

National Science Foundation (China) International Collaborative Project
(Key Research Program: Disaster evolution of large-scale projects under dynamic loadings)

Supporting project (2013-2017), 0.5 million USD.

Joint development of OpenSees and its applications in earthquake-induced disaster evolution of civil infrastructures

Purpose: Making OpenSees as one of main simulation/integration platforms

China team
PI: Prof. Jingping Ou (DUT)

- **Li Mingchu Li** (DUT)
- Li Chen (Tsinghua Univ.)
- Quan Gu (Xiamen Univ.)
- Yang Yang (DUT)

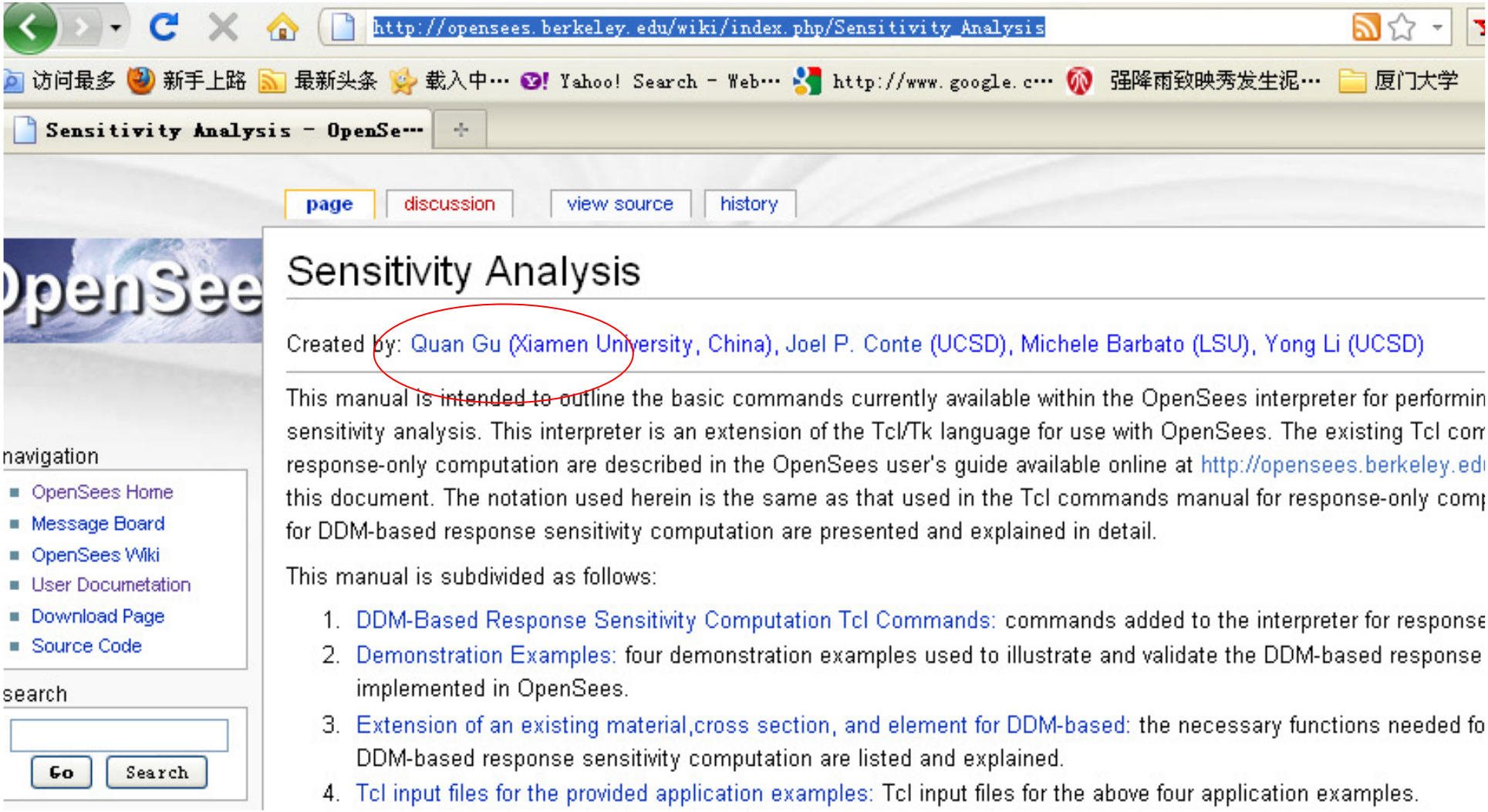
US collaborators
PI: Prof. Stephen Mahin (UCB)

- Joel Conte (UCSD)
- Frank McKenna (UCB)
- Gilberto Mosqueda (UCSD)
- Juan Caicedo (U south carolina)

Http://OpenSees.berkeley.edu

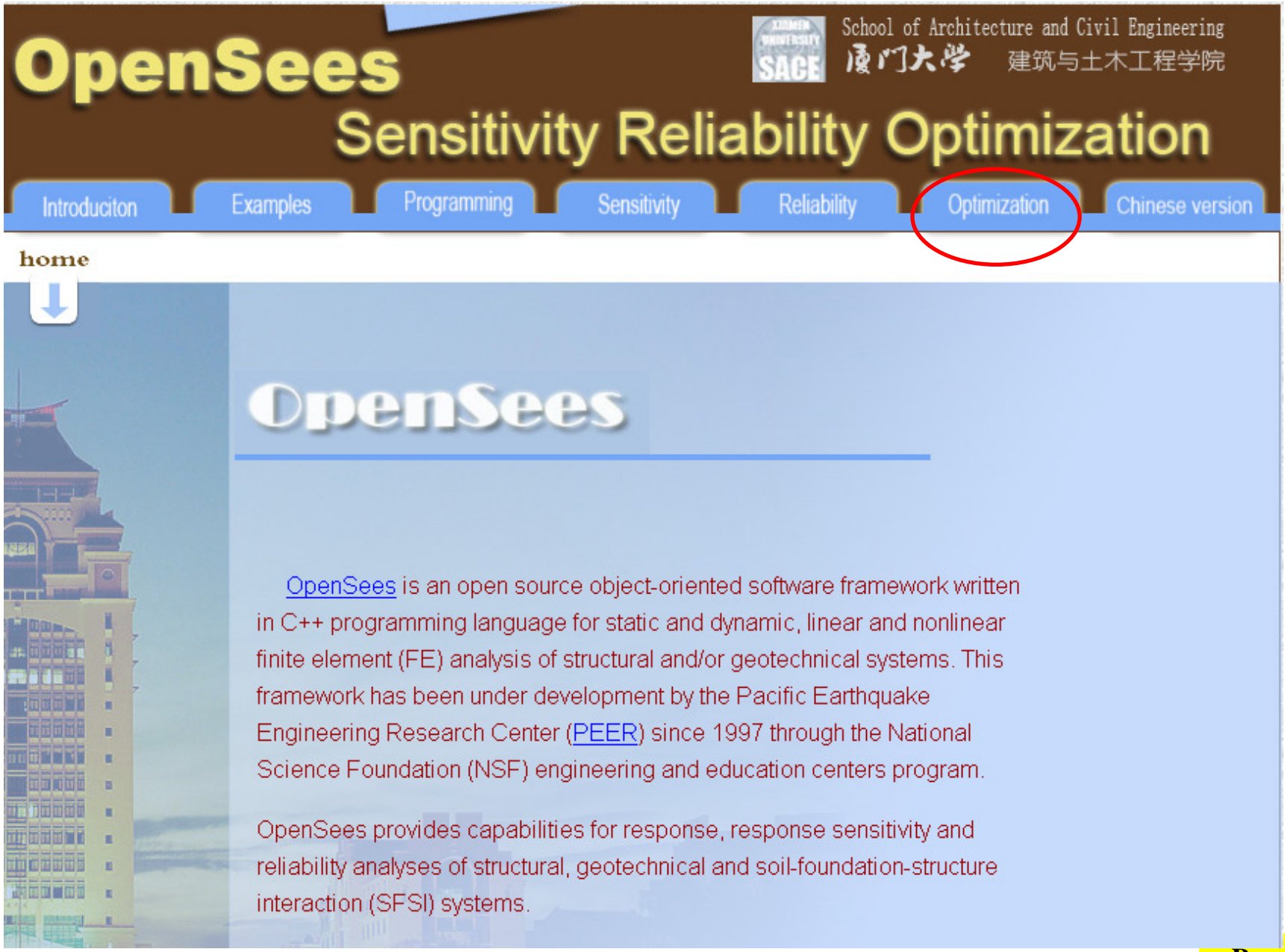
Sensitivity Analysis Manual

http://opensees.berkeley.edu/wiki/index.php/Sensitivity_Analysis



OpenSees Sensitivity, reliability and Optimization

http://archt.xmu.edu.cn/opensees/opensees.html



Outline

1. Introduction to OpenSees-SNOPT framework
2. Application examples
3. TCL commands
4. How to record responses and modify model parameters

Needs for Integrating Optimization Software Package with OpenSees

- Finite element based optimization
  - Model updating & calibration
  - System identification
  - Structural optimization
- Reliability analysis
- Reliability-based optimization
  - Reliability-based seismic design optimization

Needs & Challenges

- Optimization needed for different purposes at various levels:
  - Optimization is called for deterministic structural optimization and finite element model updating problems
  - Optimization for design point search is called inside the reliability module
  - Optimization needs to be called at two different levels for reliability-based optimization
- Interface (objective function: OF, constraint functions: CFs, design variables: DVs) must be sufficiently flexible to accommodate users' needs:
  - Any input for OpenSees can be used as DVs
  - Any output from OpenSees can be used in OF and/or CFs
- Need to integrate DDM-based sensitivity, reliability module and parameterization framework with optimization framework:
  - Need to couple Optimization with DDM in order to use analytical gradient computation
  - Reliability module needs to be compatible with parameterization framework
  - Need to develop and maintain up-to-date user's manuals for optimization, sensitivity and reliability analyses
  - Need for reliable demonstration examples

SNOPT

➢ What is SNOPT ?

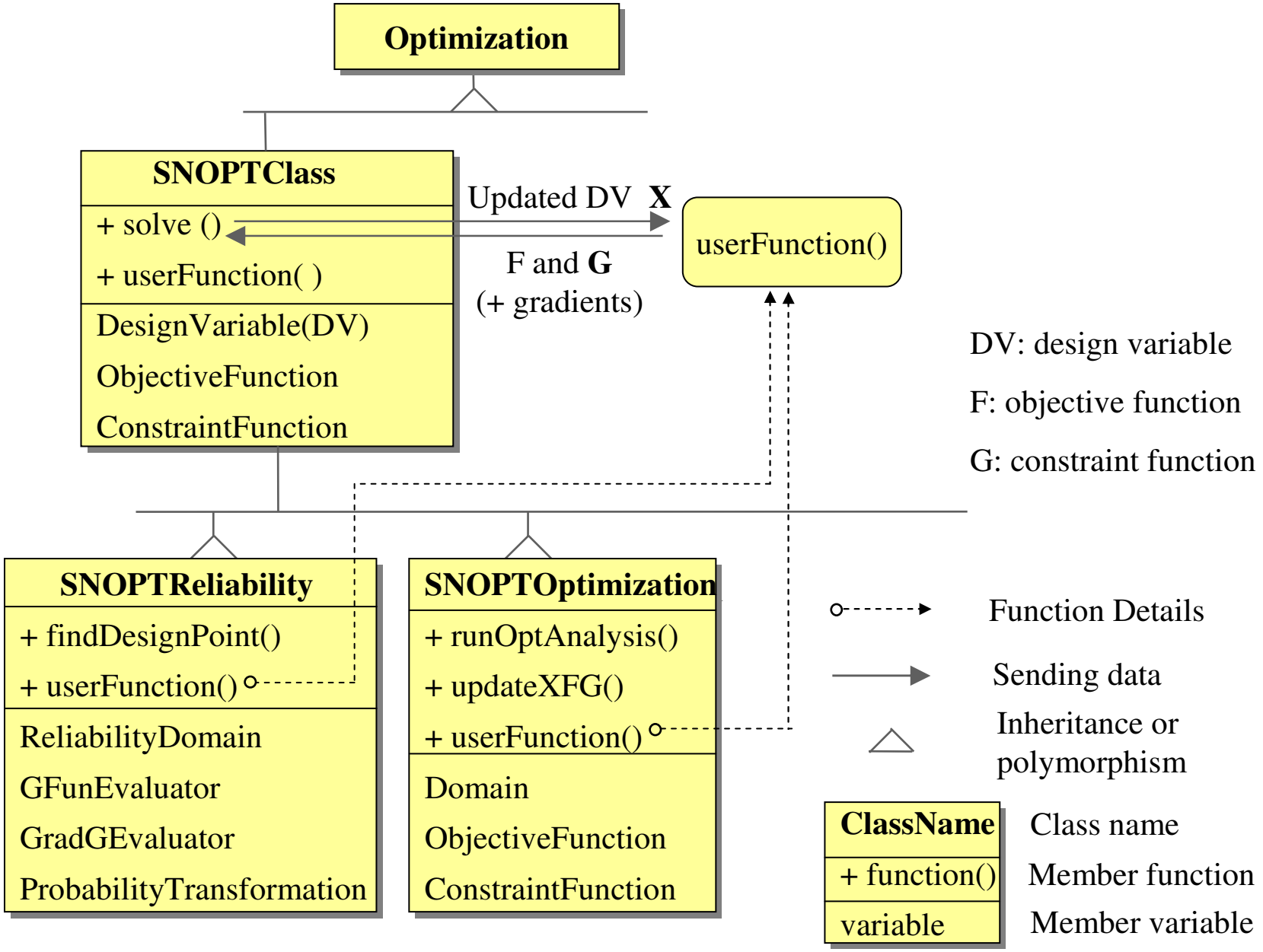
A Fortran software package for solving large-scale nonlinear optimization problems (by Philip Gill, Walter Murray and Michael Saunders)

➢ Why SNOPT ?

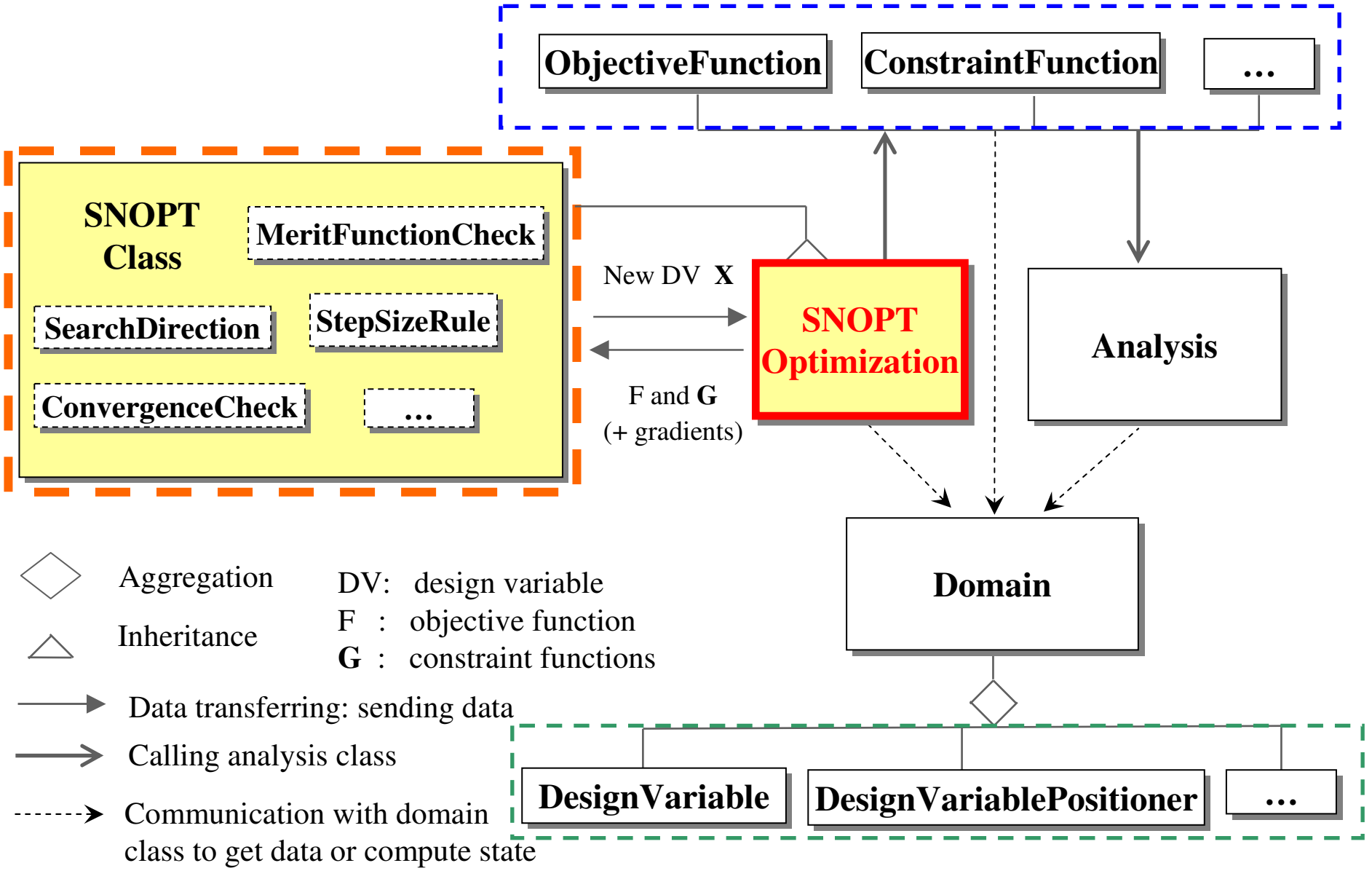
- **Suitable for large-scale applications**
- **Requiring relatively few evaluations of OF and CFs and their gradients**
- **Able to tolerate discontinuities in the gradients of OF and CFs**
- **Able to by-pass points where the OF and CFs cannot be evaluated numerically**
- **Very flexible and easy to customize**
- **Free for academic purposes**

Software Architecture

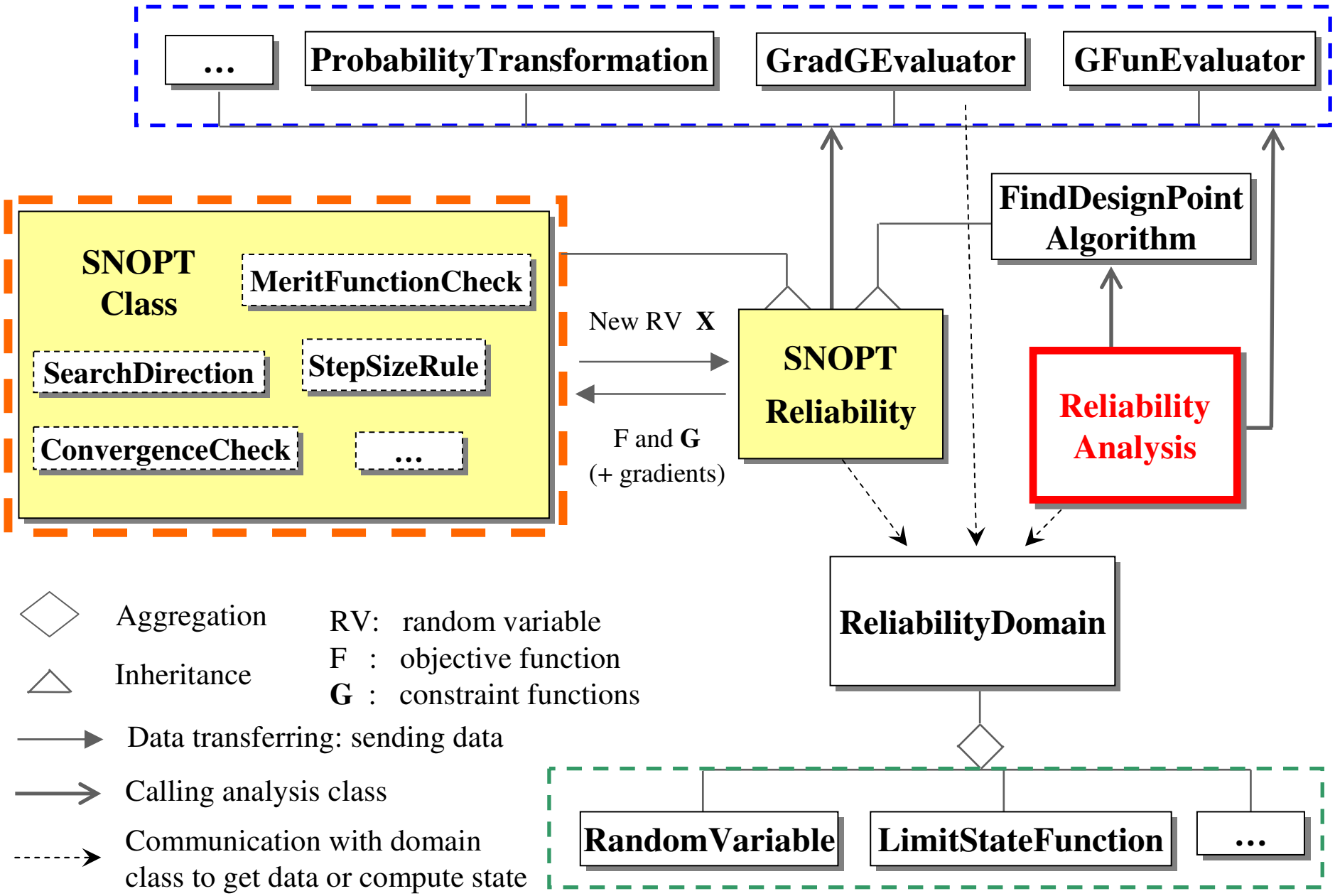
Optimization framework in OpenSees



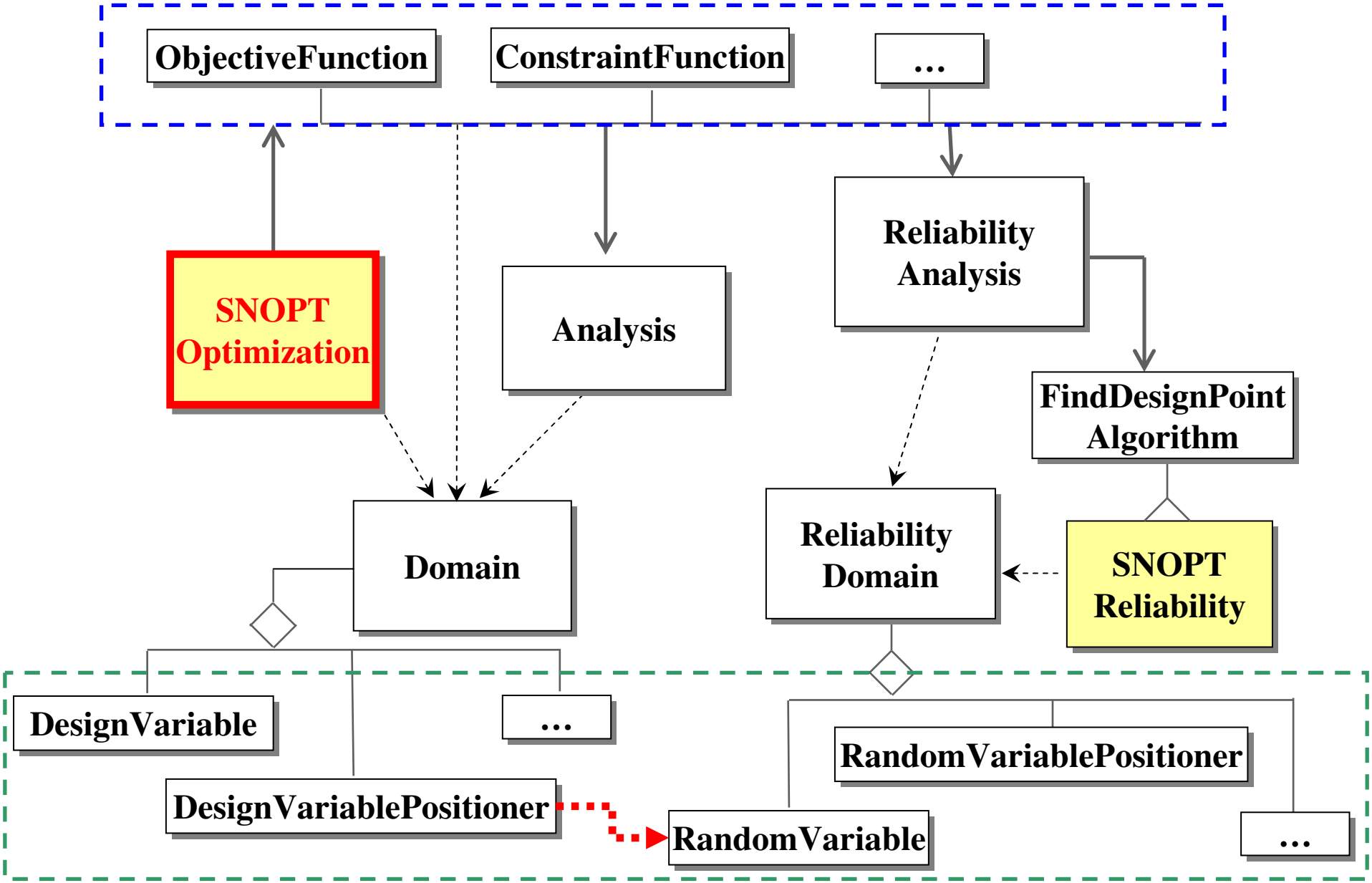
Model updating or structural optimization/identification



Structural reliability analysis in OpenSees



Reliability-based optimization

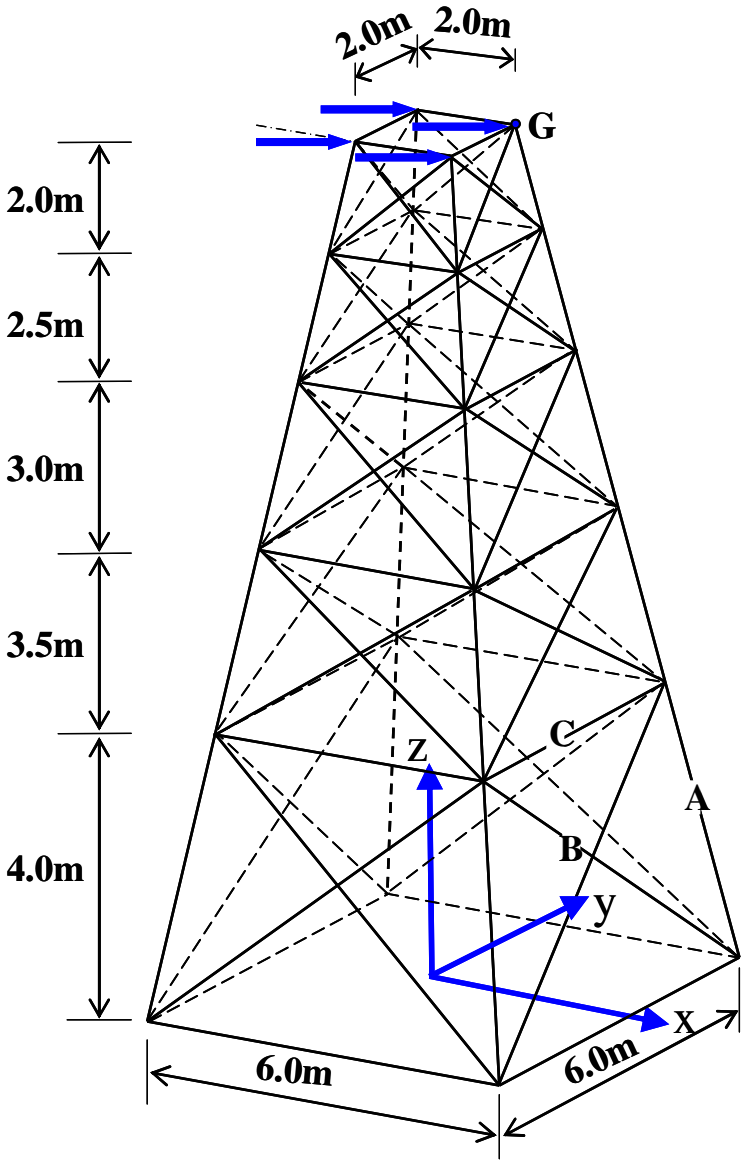


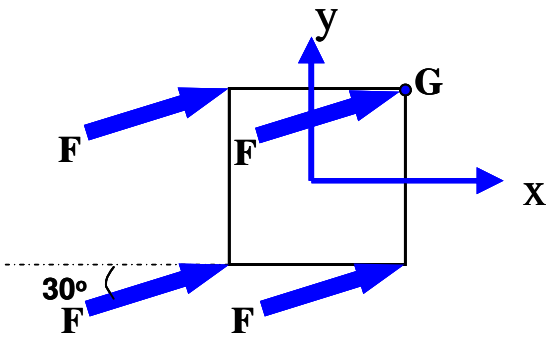
Outline

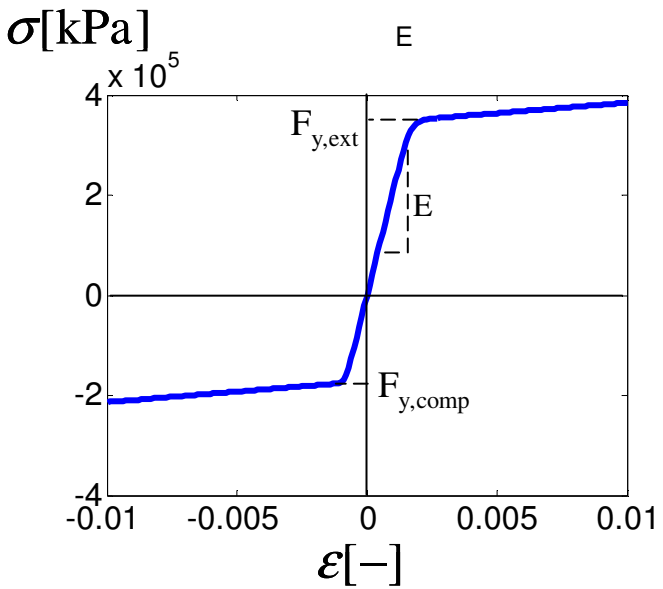
1. Introduction to OpenSees-SNOPT framework
2. Application examples
3. TCL commands
4. How to record responses and modify model parameters

Application example 1: Nonlinear Structural Optimization

Refer to: **Gu Q.**, Michele B., Conte J.P, Gill P.E., and McKenna F. OpenSees-SNOPT Framework for Finite Element-Based Optimization of Structural and Geotechnical Systems. ***Journal of Structural Engineering***, Volume 138, Issue 6 (June 2012)


2.0m
2.0m
G
2.0m
2.5m
3.0m
3.5m
4.0m
C
Z
A
B
y
x
6.0m
6.0m


y
G
F
F
x
30°
F
F


σ[kPa]
E
x 10^5
4
2
0
-2
-4
F y,ext
E
F y,comp
-0.01
-0.005
0
0.005
0.01
ε[−]

➢ Design variables

(1) Cross-section Area A: in range [8.0e-4, 1.6e-2] m^2 , initial 8e-3 m^2

(2) Cross-section Area B: in range [3.0e-4, 1.6e-3] m^2 , initial 3e-3 m^2

(3) Cross-section Area C: in range [2.0e-4, 4.0e-3] m^2 , initial 2e-3 m^2

➢ Minimize the total cost (or volume) such that

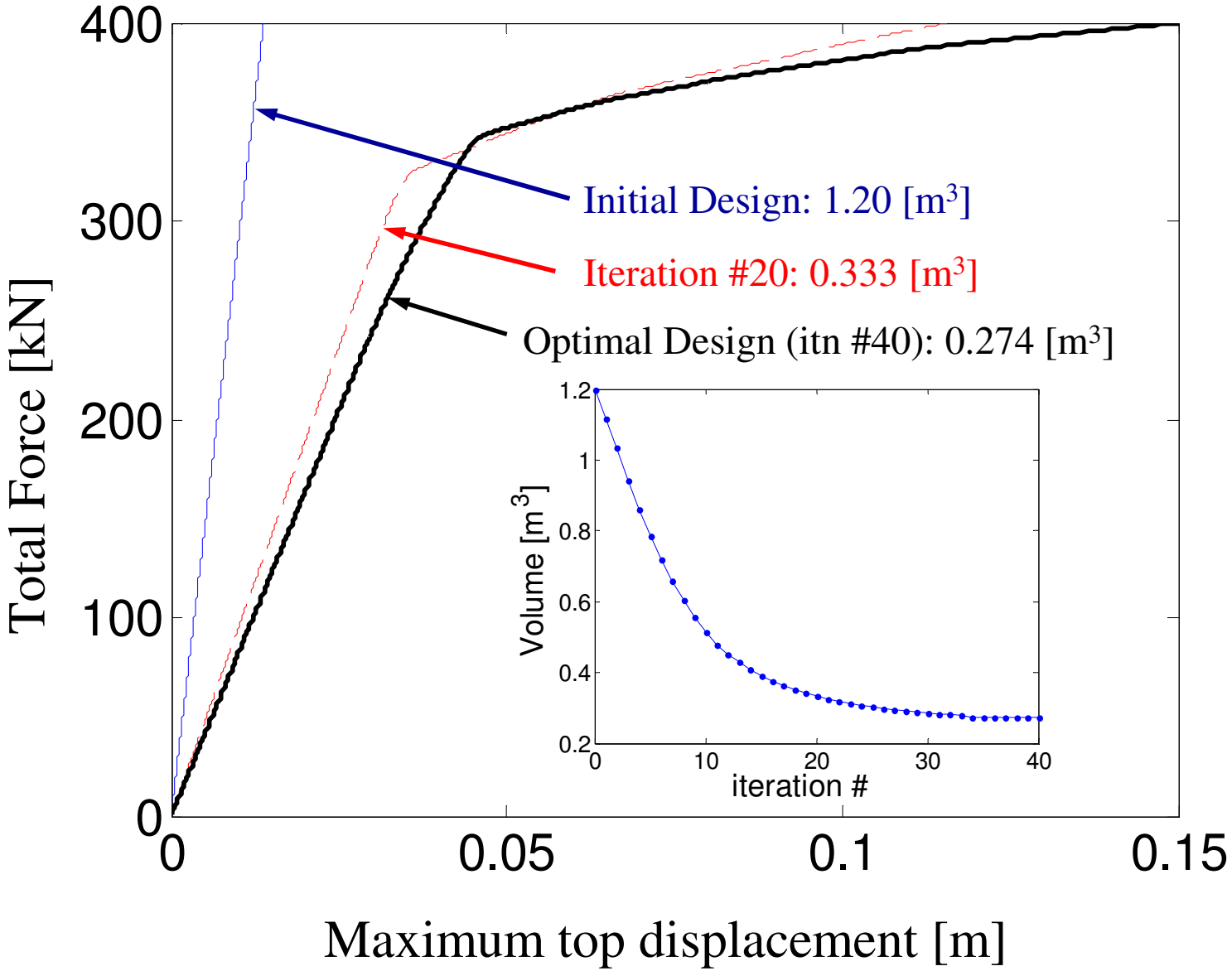
(1) when F = 25 kN, u_{max} < 1.50cm (at the top of the tower)

(2) when F = 100 kN, u_{max} < 15.0cm (at the top of the tower)

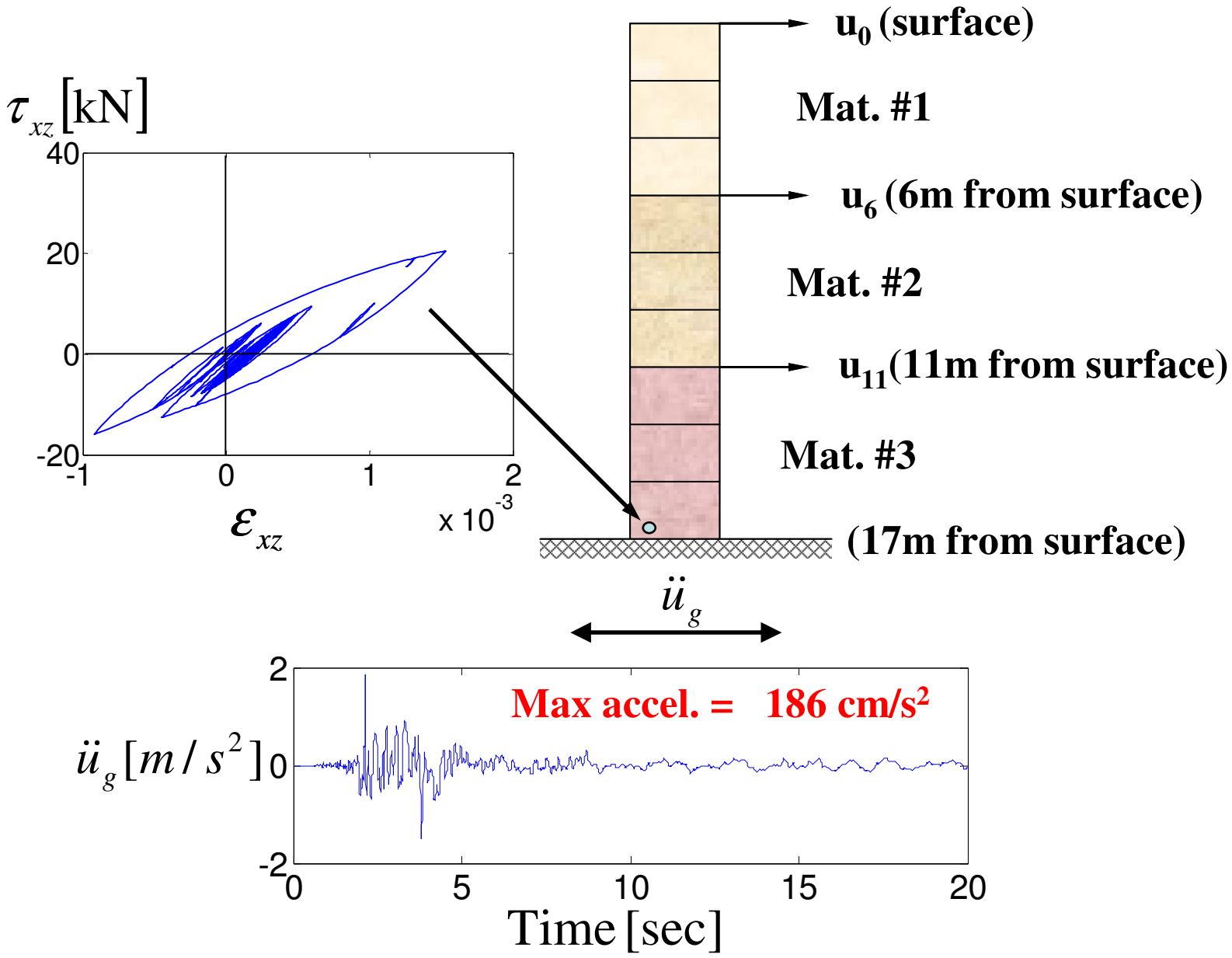
➢ Optimal design

(1) A = 3.17e-3 m^2, B = 3.51e-4 m^2, C = 2.00e-4 m^2

(2) Total volume = 0.274 m^3 (compared with initial volume of 1.20 m^3)


Initial Design: 1.20 [m3]
Iteration #20: 0.333 [m3]
Optimal Design (itn #40): 0.274 [m3]
Total Force [kN]
Maximum top displacement [m]
Volume [m3]
iteration #
400
300
200
100
0
0.05
0.1
0.15
1.2
1
0.8
0.6
0.4
0.2
10
20
30
40

Application example 2:
Nonlinear FE Model Updating


τ_{xz} [kN]
40
20
0
-20
-1
0
1
2
x 10^{-3}
ε_{xz}
u_0 (surface)
Mat. #1
u_6 (6m from surface)
Mat. #2
u_{11}(11m from surface)
Mat. #3
(17m from surface)
$\ddot{u}_g$
Max accel. = 186 cm/s^2
$\ddot{u}_g$ [m/s^2]
2
0
-2
0
5
10
15
20
Time [sec]

- Design variables, their true (and initial) values [kPa]:

  (1) Material #1: G_1, τ_1. 28800 (init: 30000), 31 (init:30)

  (2) Material #2: G_2, τ_2 . 39200 (init: 30000), 33 (init:30)

  (3) Material #3: G_3, τ_3 . 57800 (init: 30000), 34 (init:30)

  Range: $20000 < G_i < \infty \quad 20 < \tau_i < \infty \quad (\text{i} = 1, 2, 3)$

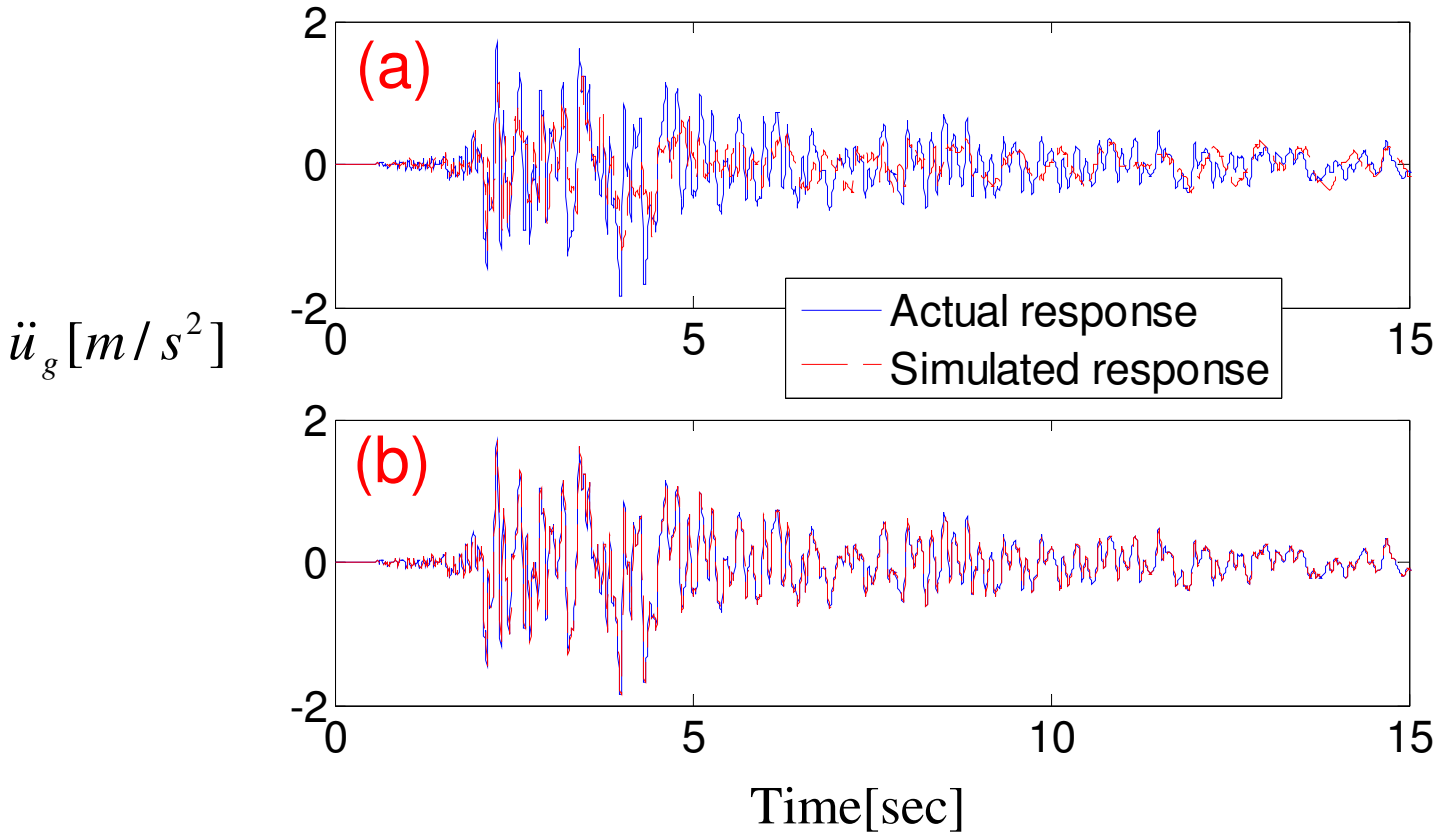
- Objective function and its gradients:

$$F = \frac{1}{2} \sum_{\text{j=1}}^{\text{\#stations}} \left(\sum_{\text{n=1}}^{\text{time step}} \left(\ddot{u}_j(t_n) - \ddot{u}_j^{Exp}(t_n) \right)^2 \right)$$

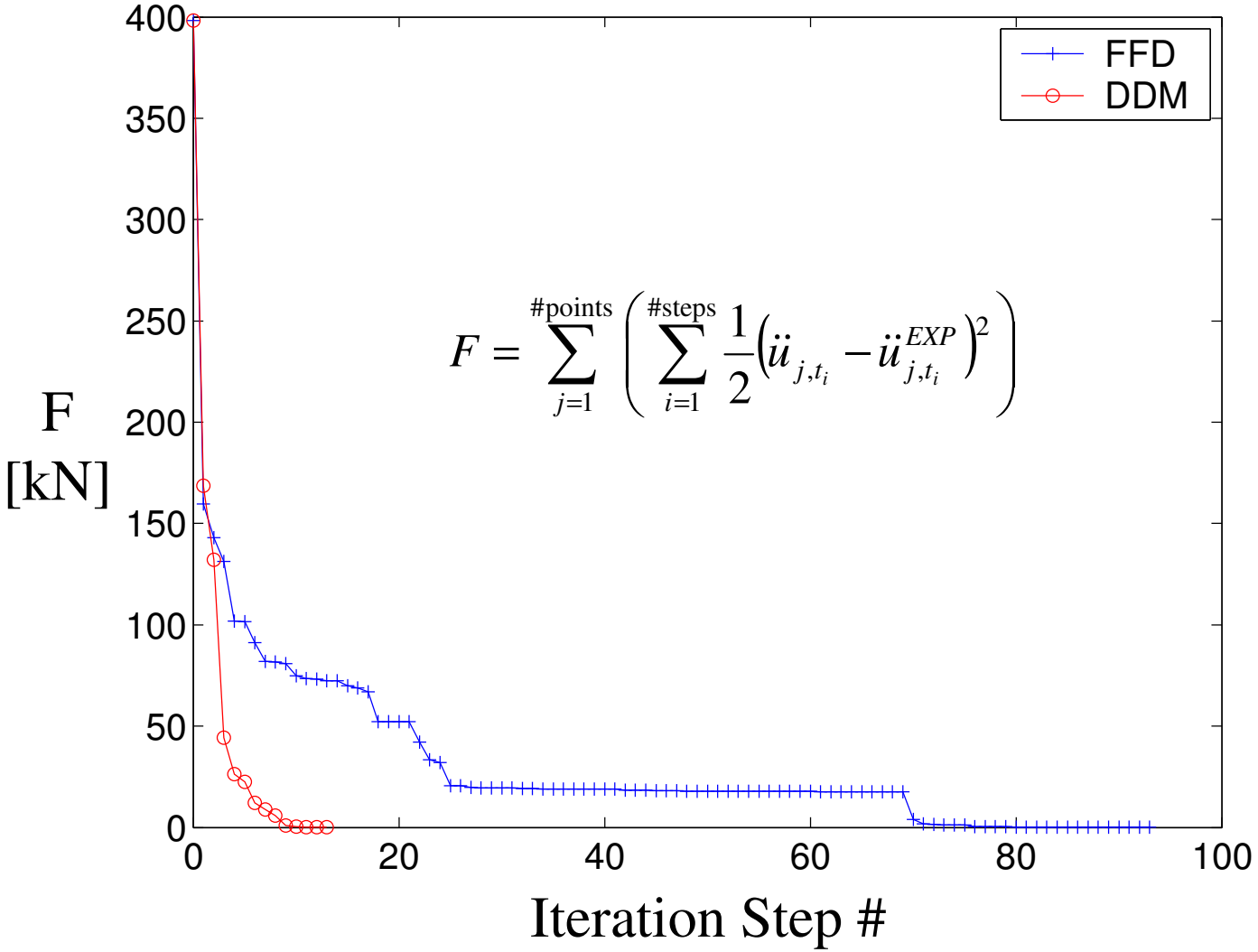
$$\frac{\partial F}{\partial \theta} = \sum_{\text{j=1}}^{\text{\#stations}} \left(\sum_{\text{n=1}}^{\text{time step}} \left(\ddot{u}_j(t_n) - \ddot{u}_j^{Exp}(t_n) \right) \frac{\partial \ddot{u}_j(t_n)}{\partial \theta} \right)$$

- Parameters obtained by SNOPT: true values

- Comparison between measured (actual) and predicted ground surface accelerations:
  (a) before model updating
  (b) after model updating

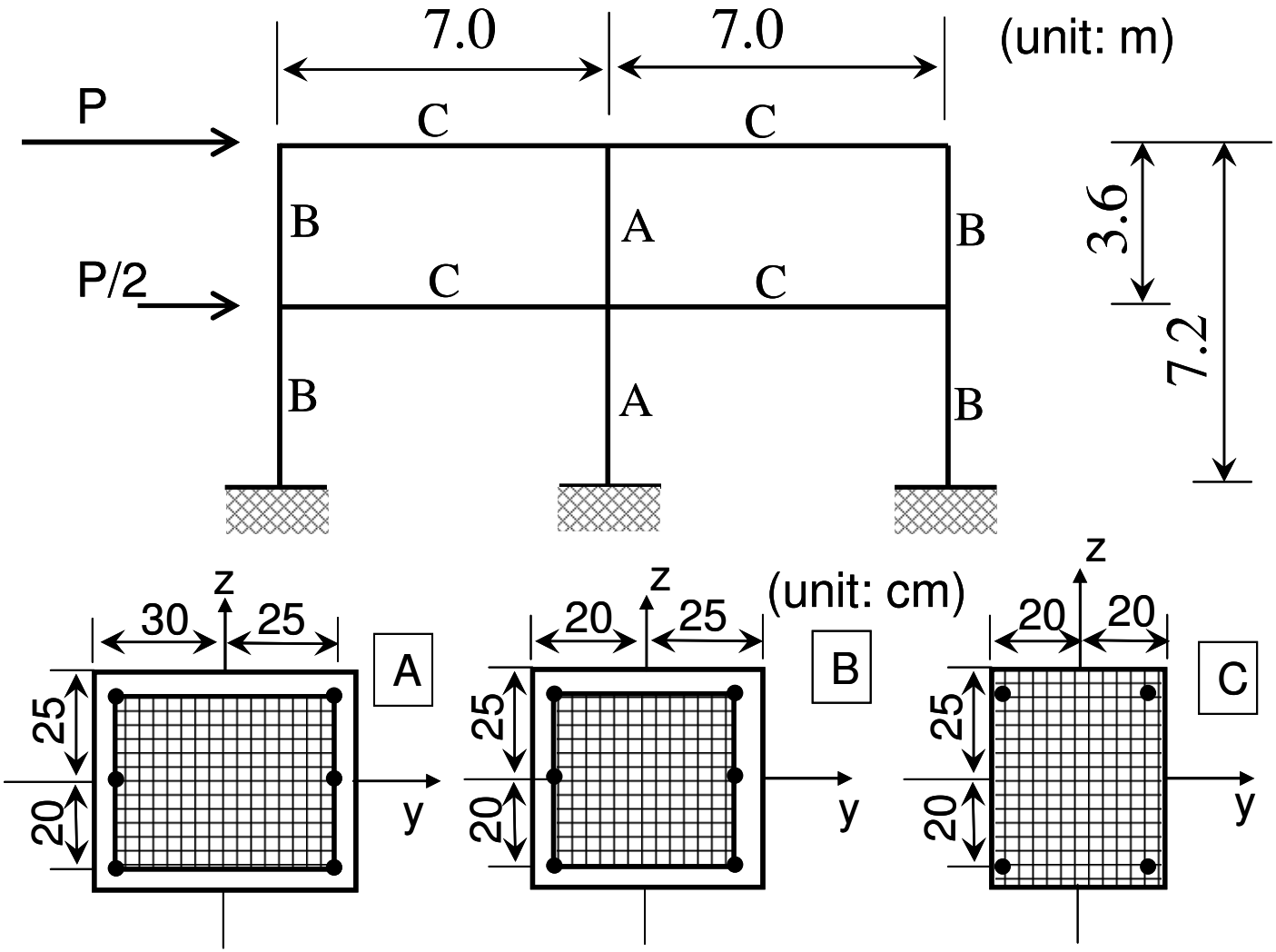


Convergence process (FFD vs DDM)



Algorithm using DDM converges MUCH FASTER than FFD !

Application example 3: Structural Reliability Analysis


7.0
7.0
(unit: m)
P
P/2
C
C
B
A
B
C
C
B
A
B
3.6
7.2
z
30
25
A
25
20
y
z
20
25
B
25
20
y
z
20
20
C
25
20
y
(unit: cm)

➢ Objective function and constraint

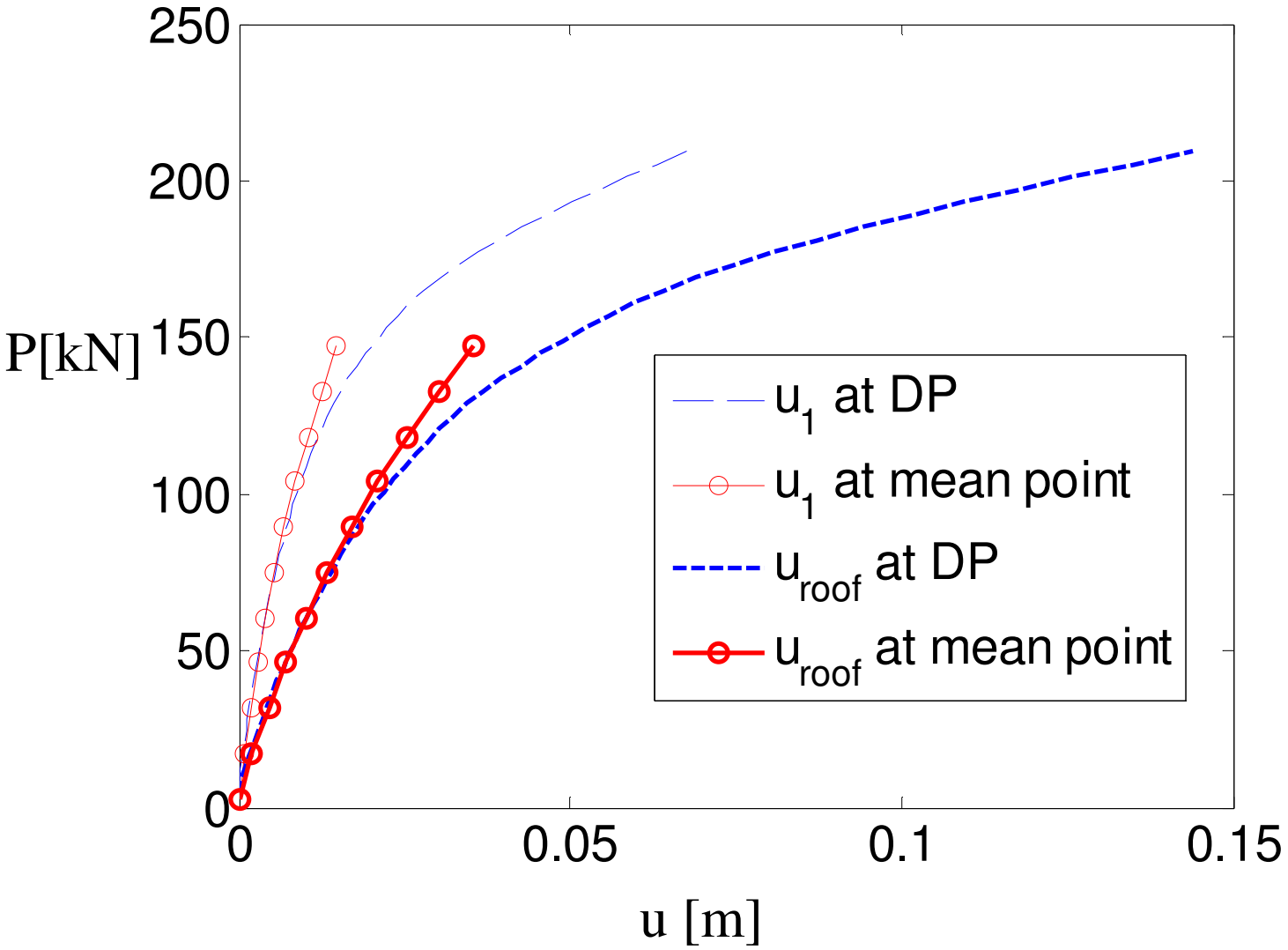
$$F = \frac{1}{2}\mathbf{y}^T\mathbf{y} \qquad G = 0.144\text{m} - u_{\text{top}} > 0$$

➢ Marginal PDFs, mean, c.o.v. and DP values of RVs

RV [unit]	Distribution	Mean	c.o.v.	DP
$f_{c,cover}$ [kPa]	lognormal	2.759e4	0.20	2.569e4
$\varepsilon_{c,cover}$ [-]	lognormal	2.0e-3	0.20	1.959e-3
$\varepsilon_{cu,cover}$ [-]	lognormal	8.0e-3	0.20	7.791e-3
$f_{c,core}$ [kPa]	lognormal	3.449e4	0.20	3.255e4
$f_{cu,core}$ [kPa]	lognormal	2.069e4	0.20	1.974e4
$\varepsilon_{c,core}$ [-]	lognormal	4.0e-3	0.20	3.924e-3
$\varepsilon_{cu,core}$ [-]	lognormal	1.4e-2	0.20	1.367e-2
f_y [kPa]	lognormal	2.482e5	0.20	2.229e5
E [kPa]	lognormal	2.1e8	0.20	2.092e8
b [-]	lognormal	2.0e-2	0.20	1.790e-2
P [kPa]	lognormal	1.5e2	0.20	2.092e2

➢ Reliability: $P_{f,\text{FORM}} = 0.0181 \quad \beta_{\text{FORM}} = 2.094$

- Base shear-horizontal floor displacements with RVs at mean point and at the DP

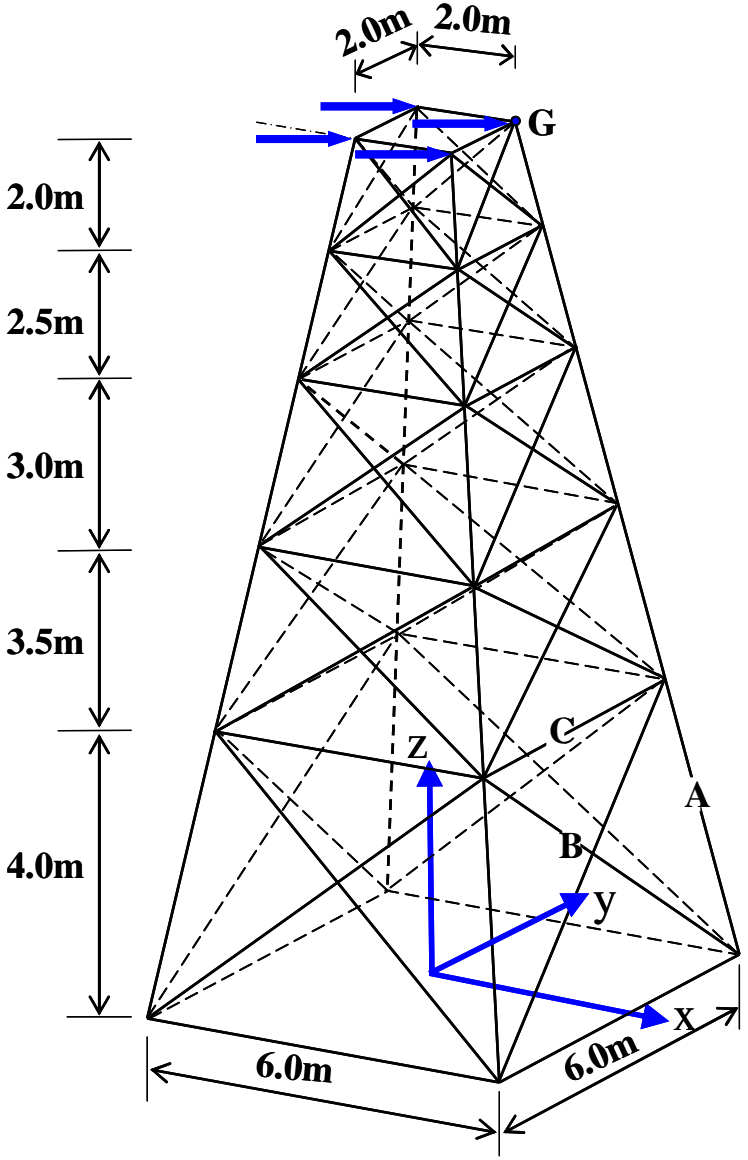


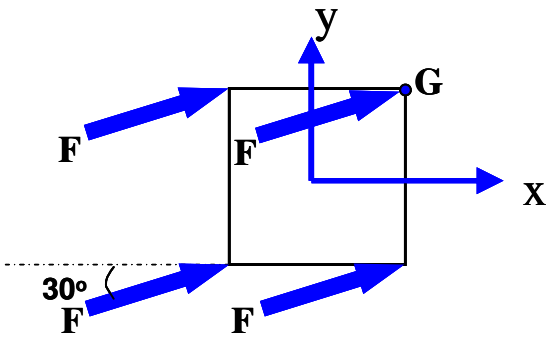
Outline

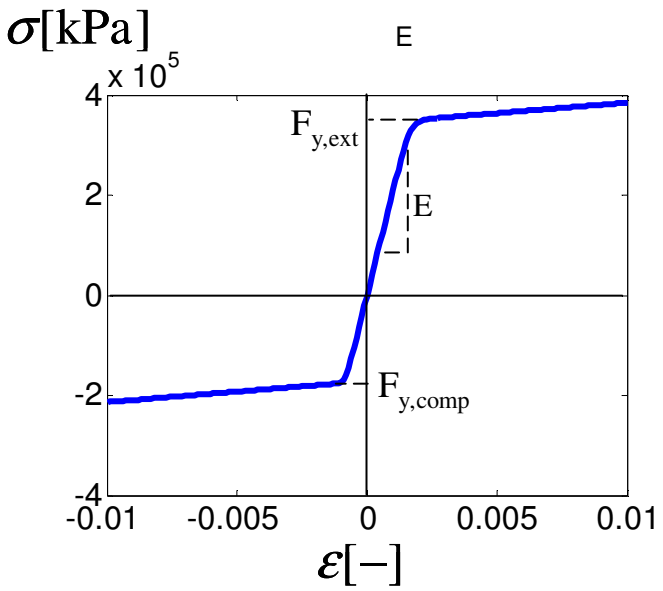
1. Introduction to OpenSees-SNOPT framework
2. Application examples
3. TCL commands
4. How to record responses and modify model parameters

```
source model.tcl   ;# FE model

Optimization

designVariable  1 -name DV_E   -startPt 1.8e8     -lowerBound 1.0E8 -upperBound  1.e20
designVariable  2 -name DV_fy  -startPt 270000  -lowerBound 1.0E5 -upperBound 1.e20

designVariablePositioner 1  -dvNum 1   -element 1  -material  E
designVariablePositioner 2  -dvNum 1   -element 3  -material  E

objectiveFunction  1 -name F -tclFile F.tcl  -lowerBound -1.e20 -upperBound 1.e20
    -gradientName gradF

array set uBound {1 4.0 2 5.0}
array set lBound {1 -1e20 2 -1e20}

constraintFunction 1  -name G -tclFile G.tcl -upperBound $uBound -lowerBound $lBound
     -gradientName gradG

runSNOPTAnalysis -maxNumIter 50 -printOptPointX OptX.out -printFlag 1
      -tclFileToRun tclFileToRun.tcl
```

Example 1.


2.0m
2.0m
G
2.0m
2.5m
3.0m
3.5m
4.0m
C
Z
A
B
y
x
6.0m
6.0m


y
G
F
F
x
30°
F
F


σ[kPa]
E
x 10^5
4
2
0
-2
-4
F y,ext
E
F y,comp
-0.01
-0.005
0
0.005
0.01
ε[−]

➢ Design variables (define variables DV_A, DV_B, DV_C)

(1) Cross-section Area A: in range [8.0e-4, 1.6e-2] m^2 , initial 8e-3 m^2

(2) Cross-section Area B: in range [3.0e-4, 1.6e-3] m^2 , initial 3e-3 m^2

(3) Cross-section Area C: in range [2.0e-4, 4.0e-3] m^2 , initial 2e-3 m^2

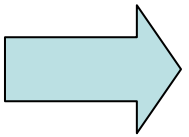
➢ Objective Function: total volume(define F in F.tcl)

$$F = L_A * DV\_A + L_B * DV\_B + L_C * DV\_C$$

➢ Constraint Functions(define G in G.tcl)

(1) when Force = 25 kN, the top $u_{max} < 1.50cm$

(2) when Force = 100 kN, the top $u_{max} < 15.0cm$



Go to source files

➢ Analysis (in tclfiletorun.tcl)

Store the requested respones by F and G (and their gradient)

Main file

```
....
# ---------------------------------- optimization part ----------------
optimization

designVariable  1 -name DV_A   -startPt 8.0e-3    -lowerBound 8.0e-4   -
upperBound  1.6e-2

designVariable  2 -name DV_B   -startPt 3.0e-3    -lowerBound 3.0e-4   -
upperBound  6.0e-3

designVariable  3 -name DV_C   -startPt 2.0e-3    -lowerBound 2.0e-4   -
upperBound  4.0e-3

for { set i 1} { $i <= 20 } { incr i} {
   designVariablePositioner $i  -dvNum 1   -element $i  A
}
for { set i 21} { $i <= 60 } { incr i} {
          designVariablePositioner $i  -dvNum 2   -element $i  A
}
for { set i 61} { $i <= 80 } { incr i} {
          designVariablePositioner $i  -dvNum 3   -element $i  A
}
```

To be continued

Main file

```
objectiveFunction  1 -name F -tclFile F.tcl  -lowerBound 0.0 -upperBound
1.e20

array set uBound { 1  0.015  2   0.15 }
array set lBound { 1  0.0    2   0.00 }

constraintFunction 1  -name G    -tclFile G.tcl -upperBound uBound -
lowerBound lBound

runSNOPTAnalysis -maxNumIter 100 -printOptPointX OptX.out -printFlag 1  -
tclFileToRun tclFileToRun.tcl
```

Tclfiletorun.tcl

```
remove loadPattern 1

pattern Plain 1 "Constant" {
      load 21 21.6506 12.5 0.0
      load 22 21.6506 12.5 0.0
      load 23 21.6506 12.5 0.0
      load 24 21.6506 12.5 0.0
}

constraints Transformation
numberer RCM
test NormDispIncr 1.E-12 25
integrator LoadControl 1 1 1 1
algorithm Newton
system BandSPD
analysis Static

analyze 1

set temp1a [nodeDisp 23 1]
set temp2a [nodeDisp 23 2]
```

To be continued

Tclfiletorun.tcl

```
# ====================================

reset

remove loadPattern 1

# ----- LOADING ----------------------

pattern Plain 1 "Constant" {
      load 21 86.6024 50.0 0.0
      load 22 86.6024 50.0 0.0
      load 23 86.6024 50.0 0.0
      load 24 86.6024 50.0 0.0

}

analyze 1

set temp1b [nodeDisp 23 1]
set temp2b [nodeDisp 23 2]
```

F.tcl

```
# F = L1*A1 +L2*A2+L3*A3

set F [expr $DV_A * 61.05735 + $DV_B * 192.3565 + $DV_C * 66.66667]
```

G.tcl

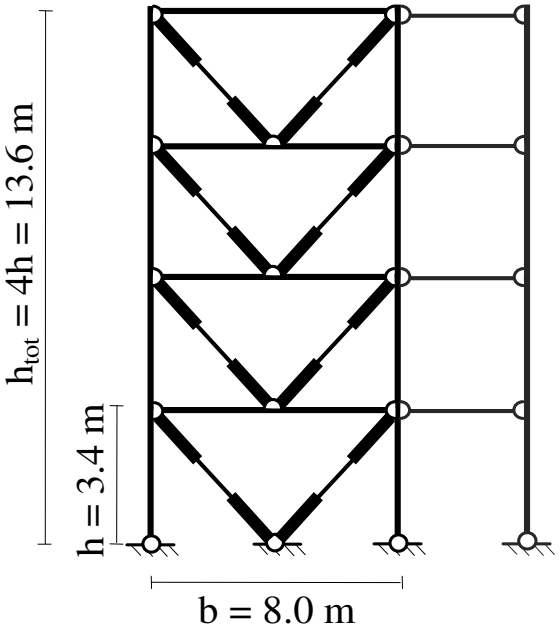
```
set G(1) [expr sqrt($temp1a*$temp1a+$temp2a*$temp2a)]
set G(2) [expr sqrt($temp1b*$temp1b+$temp2b*$temp2b)]
```

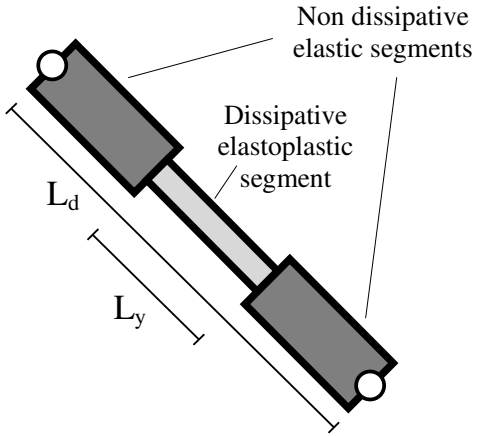
- **Only if you want to modify default setting of SNOPT:**

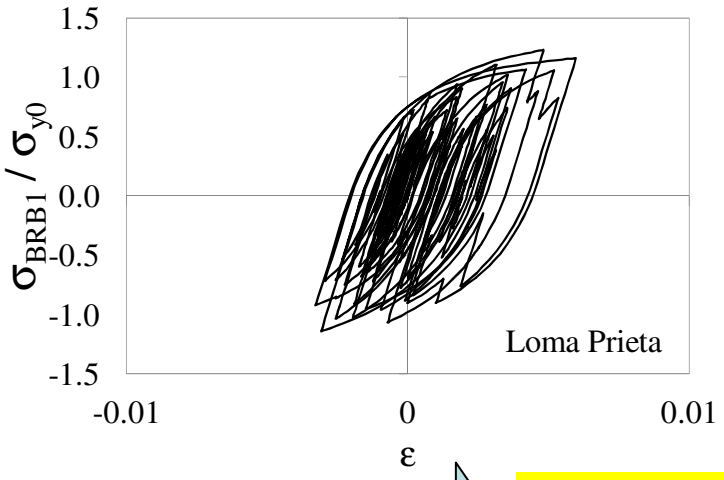
sntoya.spc

```
Begin  Toy NLP problem
   Solution                      yes
   Major feasibility tolerance  1.0e-6 * target nonlinear constraint violation
   Major optimality tolerance  1.0e-6 * target complementarity gap
   Minor feasibility tolerance  1.0e-6
   Hessian            full memory
End   Toy NLP problem
```

Example 2. Nonlinear model updating







$$F = \frac{1}{2} \sum_{n=1}^{\text{time step}} \left(\ddot{u}_j(t_n) - \ddot{u}_j^{Exp}(t_n) \right)^2$$

Go to source files

Reference: Zona A, Ragni L, Dall'Asta A. Sensitivity-based study of the influence of brace over-strength distributions on the seismic response of steel frames with BRBs. Engineering Structures 2012; 37(1):179-192.

Main file

```
optimization

designVariable 1-name sigmaYC -startPt 454.75 -lowerBound 453.7 –upperBound
535.4
designVariable  2 -name alphaT    -startPt 0.75  -lowerBound 0.6   -upperBound  0.9
designVariable  3 -name alphaC    -startPt 0.65 -lowerBound 0.4   -upperBound  0.9
designVariable  4 -name deltaT    -startPt 0.4   -lowerBound 0.1    -upperBound  0.8
designVariable  5 -name deltaC    -startPt 0.45  -lowerBound 0.1  -upperBound  0.8


objectiveFunction  1 -name F   -lowerBound -1.0e20  -upperBound 1.e20  -tclFile
F.tcl  ;# -gradientName gradF

runSNOPTAnalysis -maxNumIter 100 -printOptPointX OptX.out -printFlag 1  -
tclFileToRun tclFileToRun.tcl
```

Tclfiletorun.tcl

```
Wipe

....

 uniaxialMaterial SteelBRB 11   [expr 2.1e5*$A1]  [expr 275.0*$A1]  [expr
453.75*$A1]   $alphaT   0.01   $deltaT   [expr $sigmaYC*$A1]   $alphaC
0.015   $deltaC  1.0e-14

 uniaxialMaterial SteelBRB 12   [expr 2.1e5*$A2]  [expr 275.0*$A2]  [expr
453.75*$A2]   $alphaT   0.01   $deltaT   [expr $sigmaYC*$A2]   $alphaC
0.015   $deltaC  1.0e-14

 uniaxialMaterial SteelBRB 13   [expr 2.1e5*$A3]  [expr 275.0*$A3]  [expr
453.75*$A3]   $alphaT   0.01   $deltaT   [expr $sigmaYC*$A3]   $alphaC
0.015   $deltaC  1.0e-14

 uniaxialMaterial SteelBRB 14   [expr 2.1e5*$A4]  [expr 275.0*$A4]  [expr
453.75*$A4]   $alphaT   0.01   $deltaT   [expr $sigmaYC*$A4]   $alphaC
0.015   $deltaC  1.0e-14
....
recorder Node    -file disp_f4.out       -time   -node  5         -dof 1   disp;
...
```

F.tcl

```
# F = (u1-u1_0)^2+(u2-u2_0)^2+....

set fileId_1 [open "disp_f4_stand.out" "r"]
set fileId1 [open "disp_f4.out" "r"]

set F 0

  while {[gets $fileId_1    tmp_1] >= 1} {

  scan $tmp_1  "%f  %e   " time  u_1
  if {[gets $fileId1    tmp1] >= 1} {
     scan $tmp1 "%f  %e " time  u1
  }    else { puts "can not run F.tcl"   }

  set  F [expr $F+($u_1-$u1)*($u_1-$u1)/2.0]

}; #wihle

close $fileId_1
close $fileId1
```

Outline

1. Introduction to OpenSees-SNOPT framework
2. Application examples
3. TCL commands
4. How to record responses and modify model parameters

Recording responses and modifying model parameters

- ✓ **Element Recorder commands**

  recording any computed global or local responses

- ✓ **SetParameter and UpdateParameter**

  changing any model parameters at any time

Recorder Command

Command_Manual

This command is used to generate a recorder object which is to monitor what is happening during the analysis and generate output for the user.

recorder recorderType? arg1? ...

RETURNS

>0 an integer tag that can be used as a handle on the recorder for the remove recorder commmand.

-1 recorder command failed if integer **-1** returned.

The type of recorder created and the additional arguments required depends on the **recorderType?** provided in the command.

The following contain information about matType? and the args required for each of the available material types:

- Node
  - Node Recorder
  - Node Envelope Recorder
  - Drift Recorder
- Element/Section/Fiber
  - Element Recorder
  - ElementEnvelopeRecorder
- Graphics
  - Plot Recorder

Question: How to recode the local responses in various elements, sections or materials?

Recorder command is interpreted by using TclCreateRecorder() function inside TclRecorderCommands.cpp:

```
TclCreateRecorder(ClientData clientData, Tcl_Interp *interp, int argc,
                  TCL_Char **argv, Domain &theDomain, Recorder **theRecorder) {
  ……
 if ((strcmp(argv[1],"Element") == 0) || (strcmp(argv[1],"EnvelopeElement") == 0)
          || (strcmp(argv[1],"ElementEnvelope") == 0)) {……}
  if ((strcmp(argv[loc],"-ele") == 0) ||
              (strcmp(argv[loc],"-eles") == 0) ||
              (strcmp(argv[loc],"-element") == 0)) {……}
……
}
```

This command specifies the elements to be recorded, and the detailed information belonging to this element (such as sections, materials, etc.), such as:

"Recorder Element -file $fileName -time –ele 2 ……"

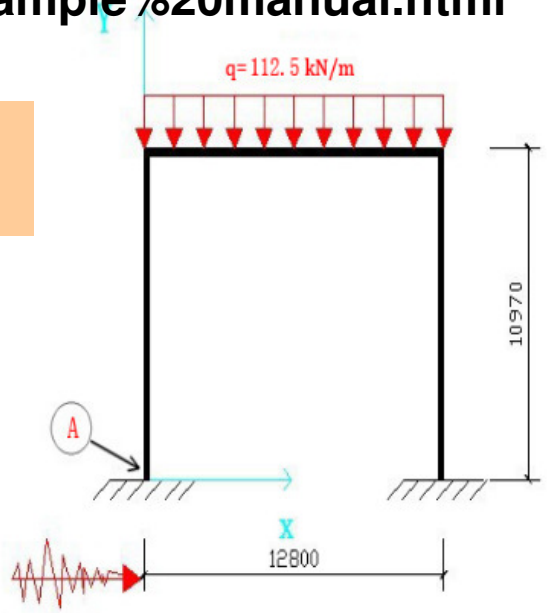
Question: how to write these detailed commands?

Difficulty: the commands are different for different element/section/material ! Need to check the source C++ code!

Example

http://archt.xmu.edu.cn/opensees/examples%20manual/example%20manual.html

Ex 3_2 Seismic response analysis of a nonlinear 2D frame modeled using fiber section



✓ In the following, using Ex3_2.tcl as example:

✓**Element:**

element dispBeamColumn …

Need to refer to the file:
DispBeamColumn2d.cpp

✓ **Section:**

section Fiber 1 { …}

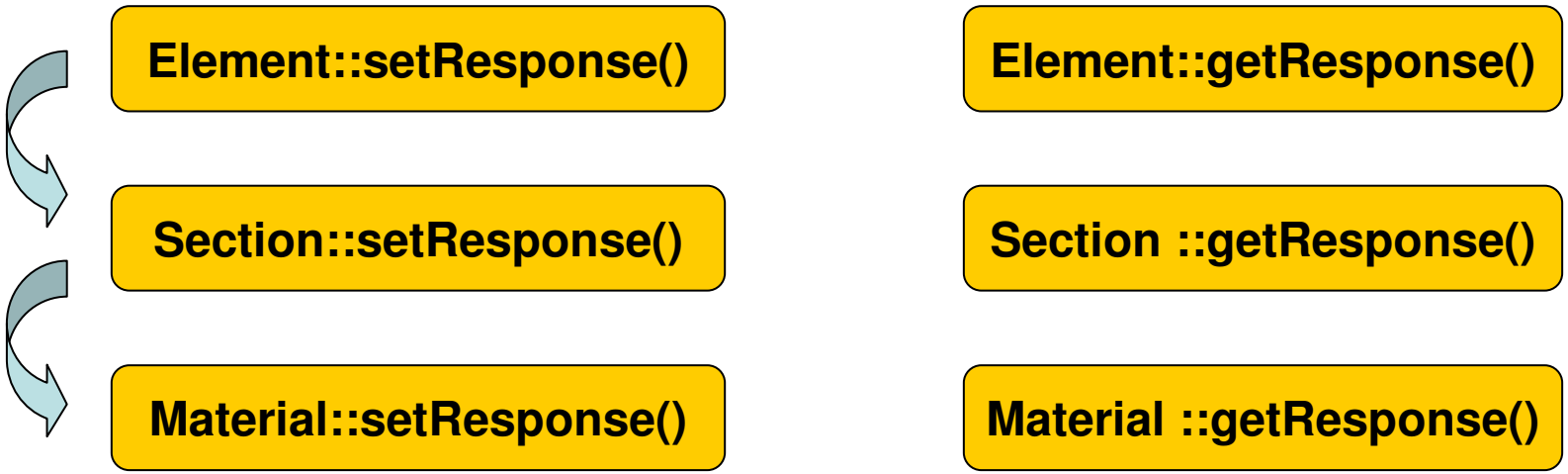
Need to refer to the file:
FiberSection2d.cpp and its base class SectionForceDeformation.cpp

✓ **Material:**

uniaxialMaterial Concrete01 ….

Need to refer the file:
Concrete01 and its base class UniaxialMaterial.cpp

✓ **In each element/section/material C++ files, the following two functions are used to specify the information to be recorded and perform the required recording:**



setResponse () : what information in this element/section/material is to be recorded (what keywords are used in the 'recorder' command)

getResponse() : obtain and return the required responses through recorder.

When and how is getResponse() called? It is called by Recorder::record() in Domain::commit() after node and element are committed.

Example: Three possible recorders

http://archt.xmu.edu.cn/opensees/examples%20manual/example%20manual.html

Ex 3_2 Seismic response analysis of a nonlinear 2D frame modeled using fiber section

Ex3_2.tcl

- ✓ recorder Element -file output/foce_2.out -time -ele 2 globalForce
- ✓ recorder Element -file output/defobeamsec_1.out -time -ele 2 section 1 force

- ✓ recorder Element -file output/defobeamsec_1.out -time -ele 2 section 1 fiber 0.22 0.22 3 stress

✓ “recorder Element -file output/foce_2.out -time -ele 2 globalForce”

```
DispBeamColumn2d::setResponse(const char **argv, int argc,
                              OPS_Stream &output)
{
  ……
  if (strcmp(argv[0],"forces") == 0 || strcmp(argv[0],"force") == 0
      || strcmp(argv[0],"globalForce") == 0 || strcmp(argv[0],"globalForces") == 0)
       { ……}
  else if (strcmp(argv[0],"localForce") == 0 || strcmp(argv[0],"localForces") == 0)
       { ……}
  else if (strcmp(argv[0],"basicForce") == 0 || strcmp(argv[0],"basicForces") == 0)
  ….
}
```

✓"recorder Element -file output/defobeamsec_1.out -time -ele 3 section 1 force"

The 1st Gauss point

```
DispBeamColumn2d::setResponse(const char **argv, int argc,
                              OPS_Stream &output) { ……
  if (strcmp(argv[0],"forces") == 0 || strcmp(argv[0],"force") == 0
    || strcmp(argv[0],"globalForce") == 0 || strcmp(argv[0],"globalForces") == 0)
    { ……}
  else if (strcmp(argv[0],"localForce") == 0 || strcmp(argv[0],"localForces") == 0)
    { ……}
  ….
  else if (strstr(argv[0],"section") != 0) {
    …
    int sectionNum = atoi(argv[1]);
    if (sectionNum > 0 && sectionNum <= numSections && argc > 2) {
        ……
      theResponse = theSections[sectionNum-1]->setResponse(&argv[2], argc-2,
output);
        output.endTag();
    …."
```

Here: argv = "section 1 force",argc=3

Here: argv[2] = "force", argc-2=1

Call function in FiberSection2d, move the pointer rightwards by 2, reduce the number of parameters by 2, thus the passed command is 'force'

- Then, since FiberSection2d does not have setResponse( ), it calls its base class SectionForceDeformation ::setResponse( ) to record

```
Response* SectionForceDeformation::setResponse(const char **argv, int
   argc, OPS_Stream &output)
 {……
      else if (strcmp(argv[0],"forces") == 0 || strcmp(argv[0],"force") == 0)
        {……}
      ……
}
```

- ✓ "recorder Element -file output/defobeamsec_1.out -time -ele 3 section 1 force"

✓“recorder Element –file output/defobeamsec_1.out -time -ele 2 section 1 fiber 0.22 0.22 3 stress”

```
DispBeamColumn2d::setResponse(co
nst char **argv, int argc,

OPS_Stream &output)
{
……
else if (strstr(argv[0],"section") != 0)
{……
theResponse =
theSections[sectionNum-1]-
>setResponse(&argv[2], argc-2,
output);
……}
……
}
```

Call FiberSection2d, pass “fiber 0.22 0.22 3 stress”

```
Response*FiberSection2d::setResponse(co
nst char **argv, int argc, OPS_Stream
&output)
{
……

else {
   if (argc > 2 || strcmp(argv[0],"fiber") == 0)
{……
passarg = 4;……
theResponse =  theMaterials[key]-
>setResponse(&argv[passarg], argc-
passarg, output); }……
}
```

Find the specified fiber with mat 3 (steel01 material) and at specified position. Then call steel01, pass “stress”

Steel01 call its base class UniaxialMaterial::setResponse()

```
Response*
UniaxialMaterial::setResponse(const char **argv,
int argc,   OPS_Stream &theOutput)
{  ……
if (strcmp(argv[0],"stress") == 0) {……}
……}
```

perform recording at material level

- Advanced topic: How to record the information that is not provided by current code in OpenSees? -- adding new recorders

"recorder" commands can be extended to record any responses by OpenSees (such as mass, stiffness, local forces, deformations, max strain, plastic strain, energy, or quantities computed using these responses).

- If we want to record the maxStrain in element 2, we need to add a recorder command in Steel01.cpp:
  "recorder Element -file output/CMaxStrain_1.out -time -ele 2 section 1 fiber 0.22 0.22 3 maxStrain".

```
Response *
Steel01::setResponse (const char **argv, int argc, OPS_Stream
&theOutput){……
   if (strcmp(argv[0],"maxStrain") == 0) {
      theOutput.tag("ResponseType", "maxsigma");
      theResponse =  new MaterialResponse(this, 6, this->getStrain());
   }
   else  return UniaxialMaterial::setResponse (argv, argc, theOutput);
};

int Steel01::getResponse (int responseID, Information &matInfo){
            switch (responseID) {……
   case 6:
        matInfo.setDouble(this-> CmaxStrain);
        return 0;
default:
   UniaxialMaterial::getResponse (responseID, matInfo);   }
```

Number "6" should be unique here.

How to modifying model parameters

SetParameter and UpdateParameter

Method to update the model parameters

In OpenSees, any model material parameter can be modified at any time step, by using the interface setParameter() and updateParameter().

Reference: Scott, M. and Haukaas, T. (2008). "Software Framework for Parameter Updating and Finite-Element Response Sensitivity Analysis." J. Comput. Civ. Eng., 22(5), 281–291.

Parameter Command

Sensitivity_Command_Manual

In DDM-based FE response sensitivity analysis, the sensitivity parameters can be material, geometry or discrete loading parameters. Each parameter should be defined as:

parameter $tag <specific object arguments>

$tag	integer tag identifying the parameter.
<specific object arguments>	depend on the object in the FE model encaps

Note: Each parameter must be unique in the FE domain, and all parameter tags mus

Question: How to identify the 'object' parameter? Such as elastic modulus of the element…

EXAMPLE

- To identify the elastic modulus, E, of the material 1 at section 3 of element

```
parameter 1 element 4 section 3 material 1 E
```

- To identify the elastic modulus, E, of elastic section 3 of element 4 (for elastic section, no specific material need to be defined), the <specific object arguments> string becomes:

```
parameter 1 element 4 section 3 E
```

- To parameterize E for element 4 with material 1 (no section need to be defined), the <specific object arguments> string simplifies as:

```
parameter 1 element 4 material 1 E
```

Notice that the format of the <specific object arguments> is different for each considered element/section/material. The user is referred to the corresponding section of this manual for the specific set of parameters and the relative <specific object arguments> format.

UpdateParameter Command

Sensitivity_Command_Manual

Once the parameters in FE model are defined, their value can be updated:

updateParameter $tag $newValue

$tag	integer tag identifying an ex
<specific object arguments>	the updated value to which th

update the parameter specified by $tag in previous 'parameter' command

Reference:

Scott M.H., Haukaas T. (2008). "Software framework for parameter updating and finite element response sensitivity analysis." Journal of Computing in Civil Engineering, 22(5):281-291.

Parameter command is interpreted by using TclModelBuilderParameterCommand() function inside TclParameterCommands.cpp :

```
int
TclModelBuilderParameterCommand(…) {  ……
   Parameter *theParameter = theTclDomain->getParameter(paramTag);……
   if (strcmp(argv[0],"parameter") == 0 || strcmp(argv[0],"addToParameter")
       == 0) { ……
      if (strstr(argv[2],"element") != 0) {……
          theObject =  theTclDomain->getElement(eleTag);……  }
    ……
}
```

This command specifies the parameter, and the parameter belonging to different objects (such as elements, sections, materials, etc).

Question: how to write these detailed commands?

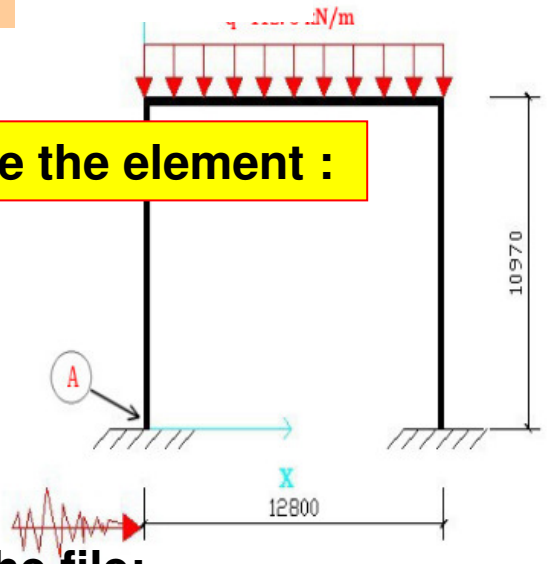
Difficulty: the commands are different for different element/section/material !

Example

http://archt.xmu.edu.cn/opensees/examples%20manual/example%20manual.html

Ex 3_2 Seismic response analysis of a nonlinear 2D frame modeled using fiber section

Ex3_2.tcl



✓ In the following, using Ex3_2.tcl as example, but change the element :

Example: parameter 1 element 3 ....

updateParameter 1 $value

✓Element:

element 2/3 .....

Need to refer to the file:
TclParameterCommands.cpp

✓ Section:

section 1/2 .....

Need to refer to the file:
DispBeamColumn2dWithSensitivity.cpp

✓ Material:

Material 3 .....

Need to refer the file:
FiberScetion2d.cpp

Example:

Caution: here the number after 'section' is the section Number 2 defined in TCL file, NOT the 2nd Gauss point (not like 'recorder' command) ! Section 2 is elastic section herein.

- ✓ **parameter 1 element 3 section 2 E**
- ✓ **updateParameter 1 2.0E8**

- ✓ **parameter 2 element 2 section 1 material 3 fc**
- ✓ **updateParameter 2 2.6E4**

✓ “parameter 1 element 3 section 2 E”

```
Int ElasticSection2d::setParameter(const char **argv, int argc,
Parameter &param) {
  …

  if (strcmp(argv[0],"E") == 0)
    return param.addObject(1, this);
  …
}
```

✓ updateParameter 1 2.0E8

```
Int ElasticSection2d::updateParameter(int paramID, Information &info)
{
  if (paramID == 1)
    E = info.theDouble;
…
  return 0;
}
```

Advance topic: how to add a parameter that is not provided by current OpenSees ? – adding new parameters

Any parameter (such as material parameters, or even CminStrain, CmaxStrain , etc) of element/section/material can be set or updated by extending the setParameter or updateParameter commands.

Example: If we want to set and update the parameter 'CminStrain' of material in Steel01, now add the commands in the Steel01.cpp as following:

- ✓**parameter 3 element 2 section 1 material 3 CminStrain**
- ✓**updateParameter 3 0.005**

```
Steel01::setParameter(const char **argv, int argc, Parameter &param)
{  ......
    if (strcmp(argv[0],"a4") == 0)
        return param.addObject(7, this);
    if(strcmp(argv[0],"CminStrain")==0
        return param.addObject(8,this);
    ......}
```

Add this command to set a new parameter 'CminStrain'

```
Steel01::updateParameter(int parameterID, Information &info)
{
        switch (parameterID) {......
        case 8:
                this->CminStrain=info.theDouble;
         ......} ......
}
```

Update the parameter 'CminStrain'

How to implement the command " parameter 3 ……"? Please look at page 22.

More information? http://archt.xmu.edu.cn

Thank you!