Reliability and Sensitivity Analysis in OpenSees

Prof. Michael H. Scott (OSU) and
Prof. Kevin R. Mackie (UCF)

OpenSees Days 2011
Richmond Field Station
Richmond, CA

August 23, 2011

Outline

Changes in Reliability Modules

- Reliability modules of OpenSees have been going through extensive changes in the last few years
  - More flexible and extensible approach to parameterizing a finite element model
  - Give user more control of input/output by taking advantage of native Tcl commands and introducing more low-level OpenSees/Tcl commands
  - Do away with sequential tagging of reliability objects
  - Less reliance on fixed-format output files for results of uncertainty analysis
- Some modules "broken" by this re-design

Parameter Objects

- FE model defined by uncertain properties and loading, aka "parameters"
- Identify and potentially update parameters during an FE analysis
- Characterize uncertainty by mapping probability distributions to parameters
- Compute sensitivity with respect to parameters
- Need script-level mechanisms for identifying, updating, and mapping parameters to PDFs in order to support
  - FORM, SORM, etc.
  - Importance Sampling
  - Monte Carlo Simulation
  - Optimization

The Parameter Command

- With an OpenSees model defined, use the `parameter` command to identify uncertain element, material, load, and node properties

```
parameter $tag <specific object arguments>
```

- Just like objects of the FE domain ...
  - each Parameter has a unique tag
  - Parameter tags do not have to be sequential or start at 1

Section/Material Parameters of Nonlinear Frame Elements

- See the *setParameter()* method in frame element, section, and material classes for valid arguments of `parameter` command, e.g.,
- Map parameter 5 to yield strength at all sections of element 3

```
parameter 5 element 3 fy
```

- Map parameter 18 to yield strength at only section 2 (from end I to J: 1, ... N_p) of element 6

```
parameter 18 element 6 section 2 fy
```

- Map parameter 9 to compressive strength of only the fibers with material tag 5 for all sections of element 23

```
parameter 9 element 23 material 5 fc
```

- Note: section dimension parameters only for "canned" section models, not generic fiber sections defined by patches and layers

Load Parameters

- Nodal and member loads contained in load patterns
- Map parameter 14 to horizontal load and node 12 contained in load pattern 3

  `parameter 14 loadPattern 3 loadAtNode 12 1`

  Last argument is global DOF, e.g., in 2D P_X=1, P_Y=2, M_Z=3
- Map parameter 28 to uniform member load on frame element 13 in load pattern 5

  `parameter 28 loadPattern 5 elementLoad 13 wy`

  Last argument: uniform loads = wx or wy; point loads Px, Py, or xOverL in local coordinate system of element

Node Parameters

- Nodal coordinates defined using the `node` command
- Map parameter 43 to X-coordinate of node 5

  `parameter 43 node 5 coord 1`

  Last argument: Global coordinate direction (1=X, 2=Y, 3=Z)
- Nodal mass defined using the mass command
- Map parameter 34 to X-direction `mass` of node 12

  `parameter 34 node 12 mass 1`

  Last argument: Global coordinate direction (1=X, 2=Y, 3=Z)
- Can also use XY and XYZ for all translational DOFs

  `parameter 34 node 12 mass XY`

Adding More Objects to a Parameter

- One-to-many relationship between a Parameter object and objects of an FE model
- Invoke the `addToParameter` command on an existing parameter

```
addToParameter $tag <specific object arguments>
```

- Specific object arguments follow the same syntax shown on previous slides
- Map parameter 5 yield strength of elements 3 and 4

```
parameter 5 element 3 fy
addToParameter 5 element 4 fy
```

- Can also create *null* parameter, the add to it

```
parameter 5
addToParameter 5 element 3 fy
addToParameter 5 element 4 fy
```

Updating Parameter Values

- Often necessary to update parameter values during an analysis
  - Staged or sequential analysis
  - Monte Carlo simulation
  - Finite difference calculations
- Use the `updateParameter` command

  ```
  updateParameter $tag $newValue
  ```
- Example:

  ```
  updateParameter 5 61.5
  ```
- Can also remove parameters from an analysis

  ```
  removeParameter 5
  ```

Query Parameter Values

- Command added to OpenSees/Tcl interpreter

  `getParamValue $tag`
- Use a parameter as an r-value of a Tcl expression [be sure to use square brackets!]
  - Put to standard output stream

    `puts [getParamValue 5]`
  - Put to file output stream

    `puts $filehandle [getParamValue 5]`
  - Store in a Tcl variable

    `set x [getParamValue 5]`
- **Note**: This command returns 0.0 if invoked before `updateParameter` (will be fixed)

Tcl List of Parameter Tags

- Command `getParamTags` returns a list of parameter tags currently defined in OpenSees model
- Iterate over all parameters with a `foreach` loop

```
foreach p [getParamTags] {
   puts [getParamValue $p]
}
```

- `getParamTags` returns an unordered list, so use inctrinsic Tcl function to sort

```
foreach p [lsort -integer [getParamTags]] {
   puts [getParamValue $p]
}
```

- List length is the total number of currently defined parameters

```
set Nparam [llength [getParamTags]]
```

Stand Alone Sensitivity Analysis

- Compute "raw" FE response sensitivity with respect to parameters
  - Direct differentiation method (DDM) operations invoked
  - Useful for assessing change in response due to changes in uncertain parameters
- Issue the following commands so that OpenSees will compute DDM sensitivity

```
reliability
sensitivityIntegrator -static ;# or -transient
sensitivityAlgorithm -computeByCommand -parameters
```

Computing the Gradients

- With sensitivtity integrator and algorithm defined, use the `computeGradients` command
- After a specified number of analysis steps

```
analyze 20
computeGradients
```

- Or inside an analysis loop

```
for {set i 1} {$i <= 20} {incr i} {
   analyze 1
   computeGradients
}
```

- Use `sensNodeDisp` and `sensSectionForce` commands to obtain response sensitivity values

Nodal Response Sensitivity

- Use the `sensNodeDisp` command

  ```
  sensNodeDisp $node $dof $param
  ```

  - Analogous to `nodeDisp` command with node tag and dof
  - Last argument is parameter wrt which sensitivity is reported

- For example, output sensitivity of node 2 horizontal displacement wrt all parameters

```
for {set i 1} {$i <= 20} {incr i} {
   analyze 1
   computeGradients
   foreach p [lsort -integer [getParamTags]] {
      puts "Parameter $p [sensNodeDisp 2 1 $p]"
   }
}
```

Section Response Sensitivity

- For nonlinear frame elements, use `sensSectionForce`
  sensSectionForce $ele $sec $dof $param
  - Analogous to `sectionForce` command with element tag, section number ($1,\ldots,N_p$ from I to J), and section dof (depends on section model)
  - Last argument is parameter tag wrt which sensitivity is computed
- For example, output sensitivity of bending moment at section 1 of element 3 wrt all parameters

```
for {set i 1} {$i <= 20} {incr i} {
   analyze 1
   computeGradients
   foreach p [lsort -integer [getParamTags]] {
    puts "Parameter $p [sensSectionForce 3 1 2 $p]"
   }
}
```

Random Variable Definition

- Associate parameters with random variables (probability distribution functions) in order to perform probabilistic finite element analysis

  ```
  randomVariable $tag type <arguments>
  ```

- Similar to parameters and FE objects
  - Each RV has a unique integer tag
  - RV tags do not have to be sequential or start at one
- RV types: normal, lognormal, poisson, etc.
- Trailing command arguments are distribution parameters, e.g., mean and stdev

  ```
  randomVariable 1 normal 60.0 6.0
  randomVariable 5 lognormal 5.0 1.0
  ```

Random Variable Utilities

- Probability density function, $f_X(x)$

  `getPDF $tag $x`

- Cumulative density function, $F_X(x)$

  `getCDF $tag $x`

- Inverse CDF, $F_X^{-1}(p)$

  `getInverseCDF $tag $p`

- List of all currently defined RV tags (returns unordered Tcl list)

  `getRVtags`

- Remove a random variable

  `remove randomVariable $tag`

Random Variable Positioners

- Map each random variable to a parameter

```
randomVariablePositioner $tag -rvTag $rvTag
   -parameter $pTag
```

- RVs and parameters do not have to have same tags
- Can use a Tcl array and `foreach` loop to construct RVPs

```
parameter 12 element 1 ...
parameter 4 loadPattern 2 ...
randomVariable 7 normal ...; set param(7) 12
randomVariable 15 lognormal ...; set param(15) 4
foreach rvTag [array names param] {
   randomVariablePositioner $rvTag -rvTag $rvTag
      -parameter $param($rvTag)
}
```

Random Variable Positioner Utilities

- Return a list of `(rvTag, pTag)` pairs for all RVPs

  getRVPositioners

  - Facilitates Monte Carlo analysis and defining gradients of performance functions

```
puts [getRVPositioners]
7 12 15 4
```

- Remove a random variable positioner

```
remove randomVariablePositioner $tag
```

Performance Functions

- Also known as "limit state functions" or "g-functions," performance functions define failure of a system
- Typically expressed in terms of nodal and element response
  - Exceeding a displacement limit
  - Exceeding member capacity, etc.
- For most PBEE applications, performance can usually be expressed as a simple algebraic function of capacity minus demand, e.g.,

$$g_1(\mathbf{x}) = 0.15 - U$$

$$g_2(\mathbf{x}) = 1500.0 - M$$

Performance Function Definition

- Use a Tcl string to express a performance function

  performanceFunction $tag "expression"

  - Arbitrary, non-sequential tags
  - Expression contains regular Tcl syntax and OpenSees commands

```
performanceFunction 76 "0.15 - \[nodeDisp 2 1\]"
performanceFunction 23 "1500 - \[sectionForce 5 3
2\]"
```

- The "slash bracket" syntax \[\] defers evaluation of the command to run-time instead of at the time of definition when the FE response is zero

Performance Function Gradients

- In some uncertainty analyses, the gradient of the performance function is required in order to search for a solution

$$\frac{\partial g_i}{\partial x_j} = \frac{\partial g_i}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x_j} + \frac{\partial g_i}{\partial \mathbf{P}} \frac{\partial \mathbf{P}}{\partial x_j}$$

- $\partial \mathbf{U} / \partial x_j$ from gradient of the FE response
  - Finite differences or direct differentiation
- $\partial g_i / \partial \mathbf{U}$ from gradient of performance function
  - Automatic finite differences, or
  - Analytic user-defined expressions

Analytic Performance Function Gradients

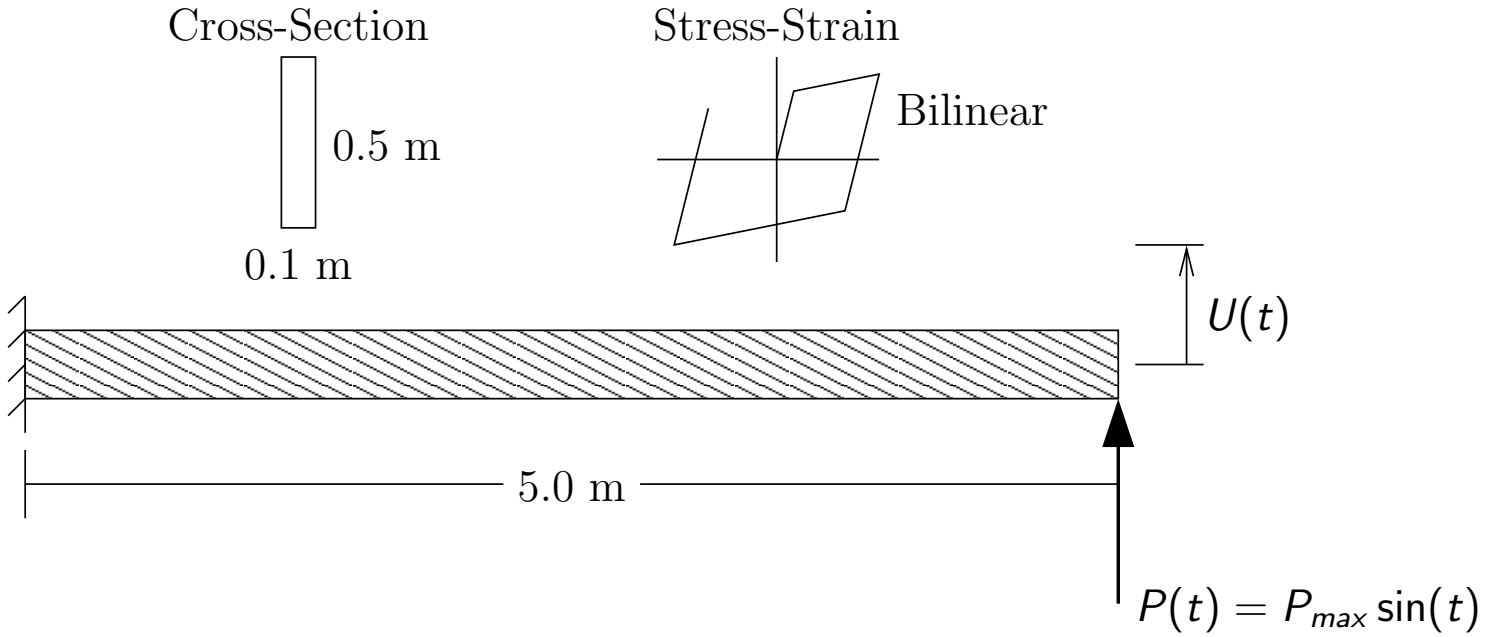
- User-defined Tcl expressions for gradient of each performance function wrt random variables

```
gradPerformanceFunction $pfTag $rvTag "expression"
```

- Define using (rvTag, pTag) pairs from RVPs in conjuction with the `sensNodeDisp` and `sensSectionForce` commands

```
foreach {rvTag pTag} [getRVPositioners] {
   gradPerformanceFunction 76 $rvTag
     "-\[sensNodeDisp 2 1 $pTag\]"
   gradPerformanceFunction 23 $rvTag
     "-\[sensSectionForce 5 3 2 $pTag\]"
}
```

Example – Material and Geometric Nonlinear Cantilever



- Discretize with five frame finite elements (linear curvature, constant axial deformation approximation)
- Corotational formulation for large displacements
- Numerically integrate stress-strain response over cross-section
- E = 2.0e8 kPa, σ_y = 4.1e5 kPa, 2% kinematic strain-hardening
- One load cycle with P_{max} = 1710 kN (5 times yield load)

Model Definition

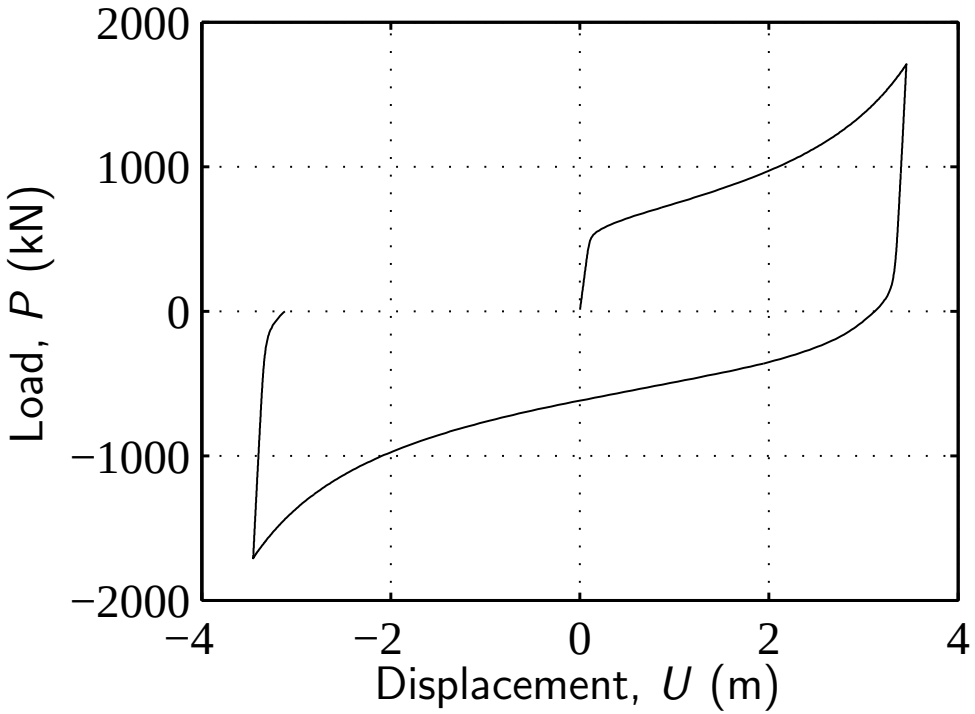
- Material definition

```
set b 0.1; set d 0.5
set E 2.0e8; set fy 4.1e5
uniaxialMaterial Hardening 4 $E $fy 0 [expr
0.02/(1-0.02)*$E]
section WFSection2d 2 4 $d $b $b 0 20 0
```

- Element definition

```
geomTransf Corotational 1
set L 5.0; set Nele 5; set dL [expr $L/$Nele]
node 0 0.0 0.0; fix 0 1 1 1
for {set e 1} {$e <= $Nele} {incr e} {
   node $e [expr $e*$dL] 0.0
   element dispBeamColumn $e [expr $e-1] $e 1
Legendre $secTag 2
}
```

Mean Load-Displacement Response



- Material yield at about 400 kN load
- Tension stiffening at about 1 m displacement
- Elastic unloading and reverse cyclic yielding

Parameter Definition

- Create blank parameters for element, then populate in loop

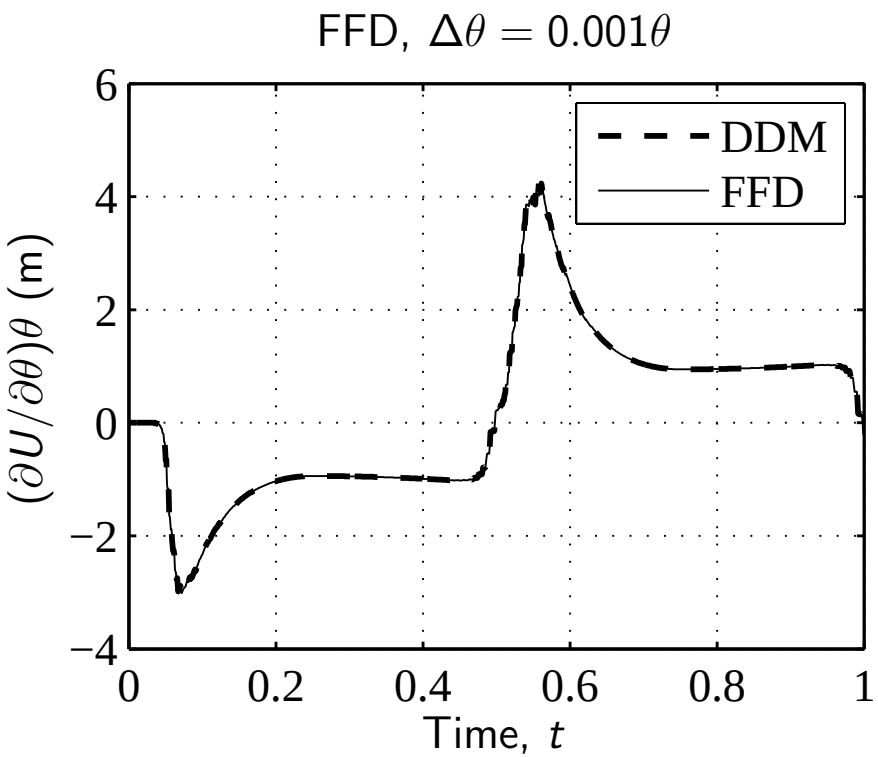
```
parameter 1
parameter 4
parameter 10
for {set e 1} {$e <= $Nele} {incr e} {
   addToParameter 1 element $e fy
   addToParameter 4 element $e E
   addToParameter 10 element $e d
}
```

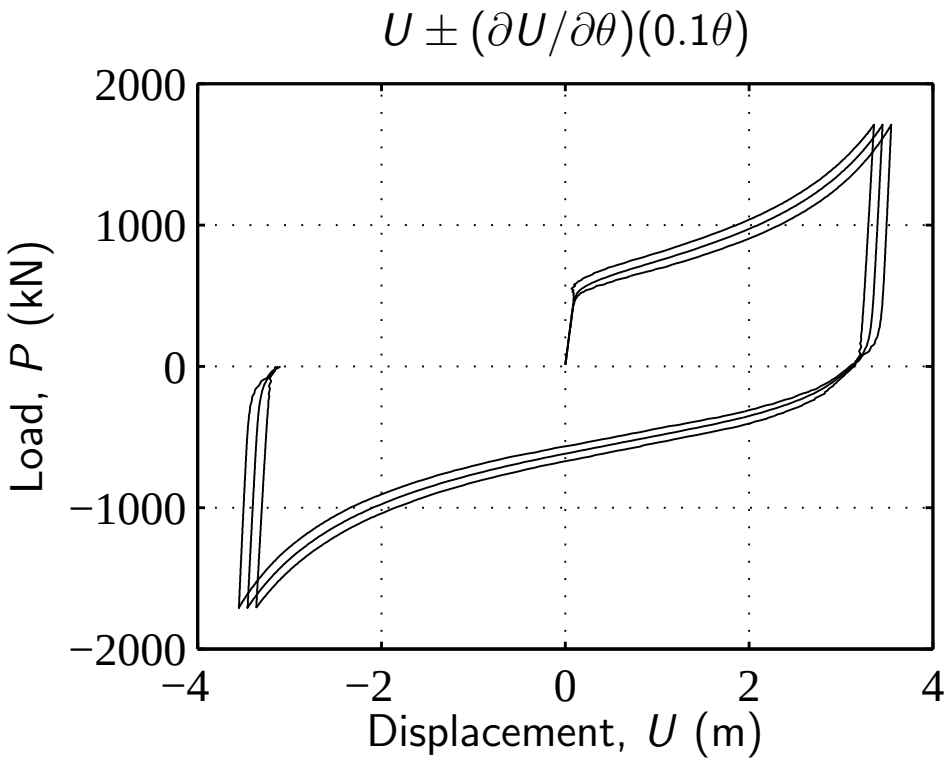
- Parameters for nodal coordinate and load

```
parameter 5 node $Nele coord 1
parameter 8 loadPattern 1 loadAtNode $Nele 2
```

Sensitivity to Yield Stress

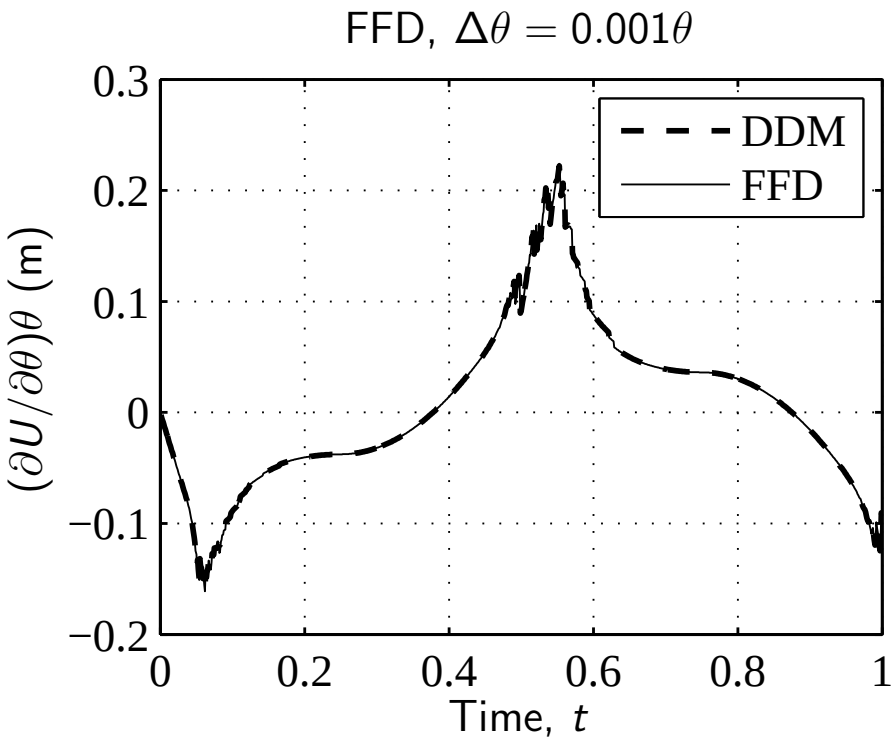
DDM verification and response envelope for strength parameter

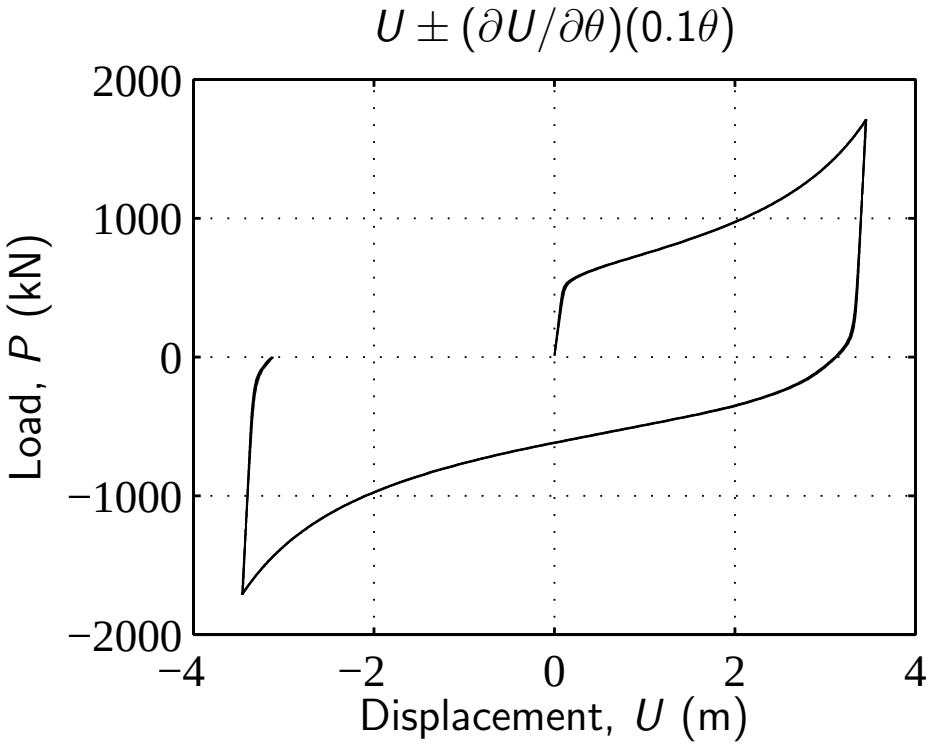




Sensitivity to Elastic Modulus

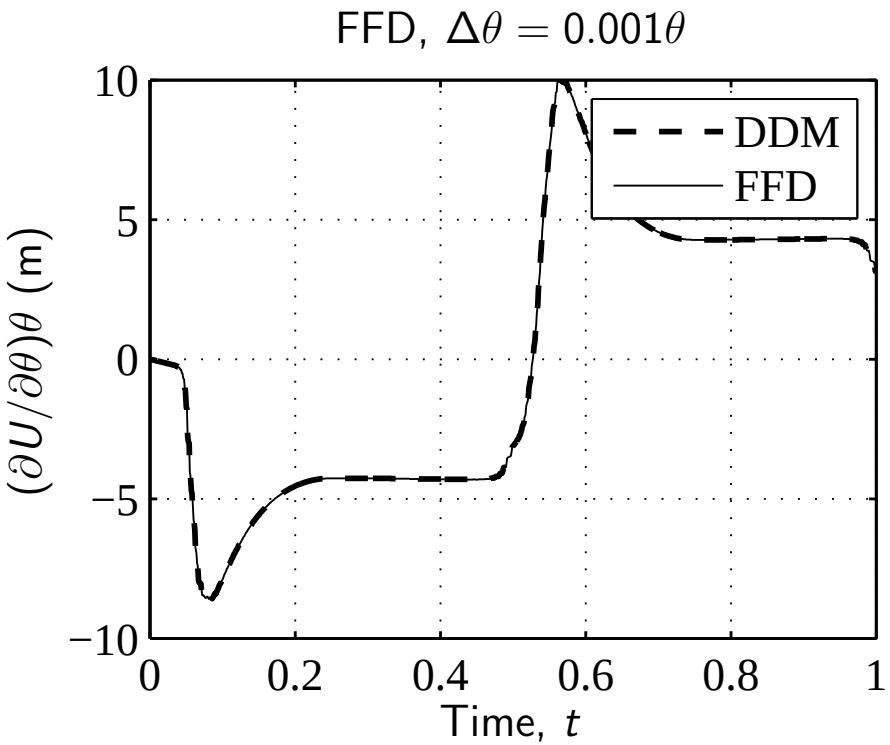
DDM verification and response envelope for stiffness parameter

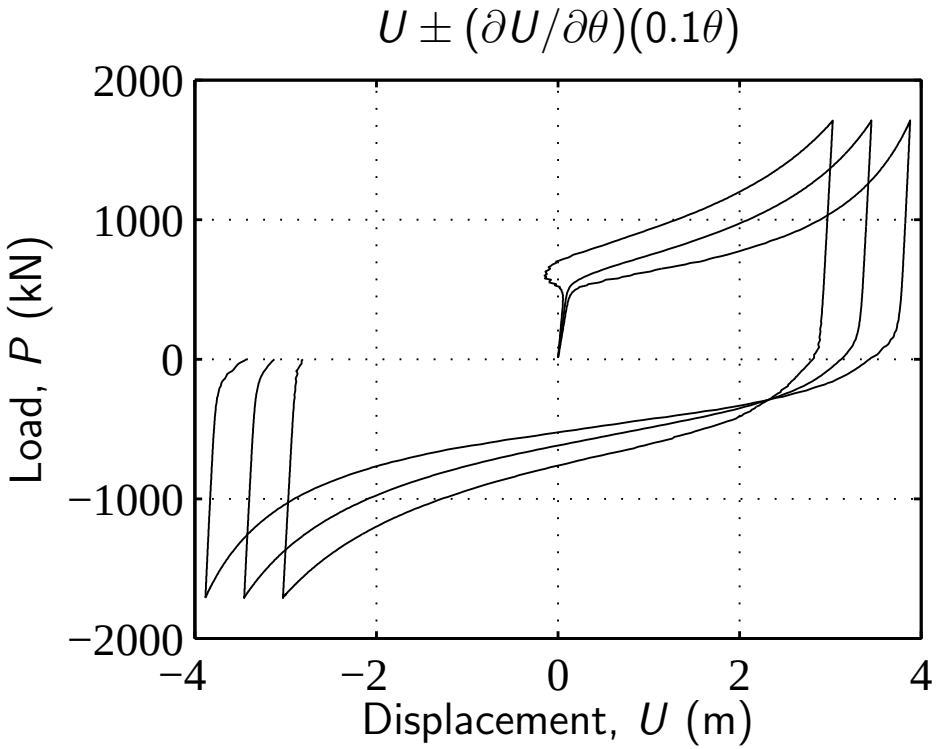




Sensitivity to Cross-Section Depth

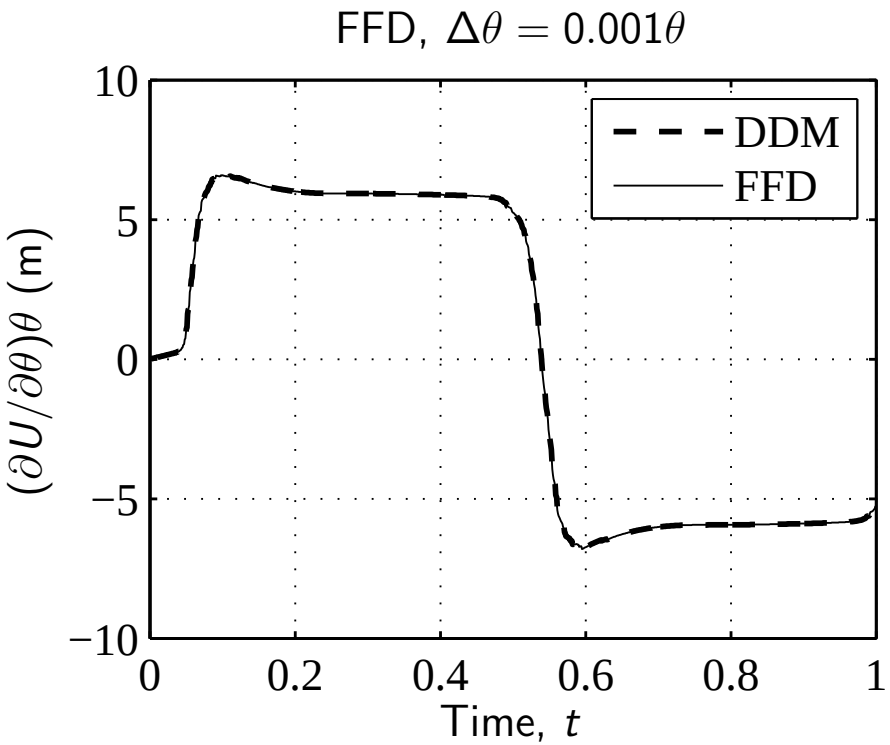
DDM verification and response envelope for local geometric parameter

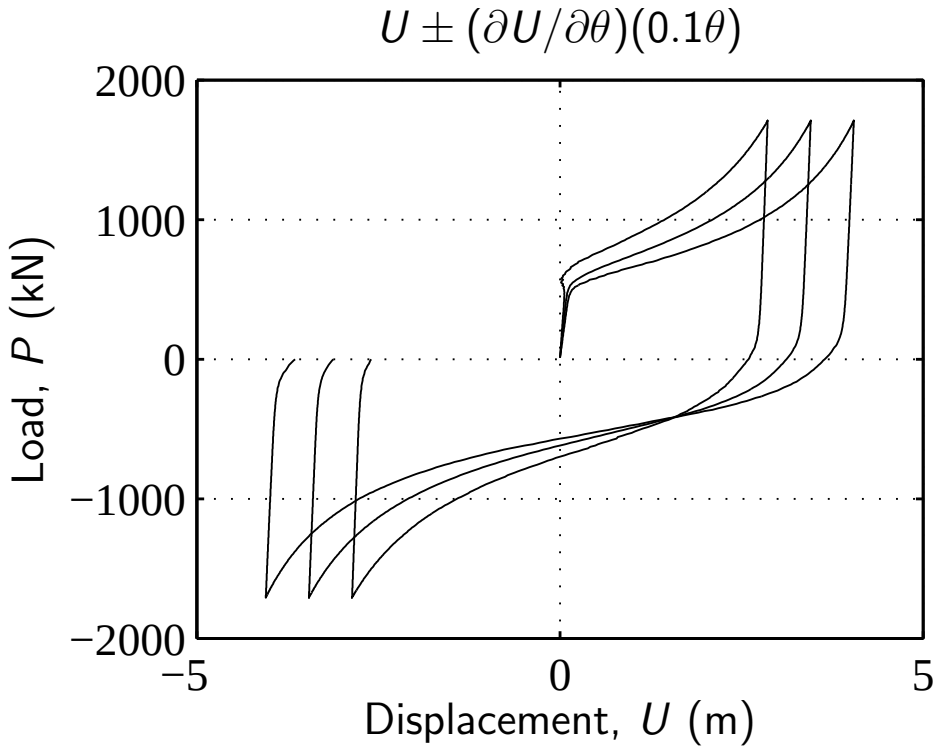




Sensitivity to Cantilever Length

DDM verification and response envelope for global geometric parameter



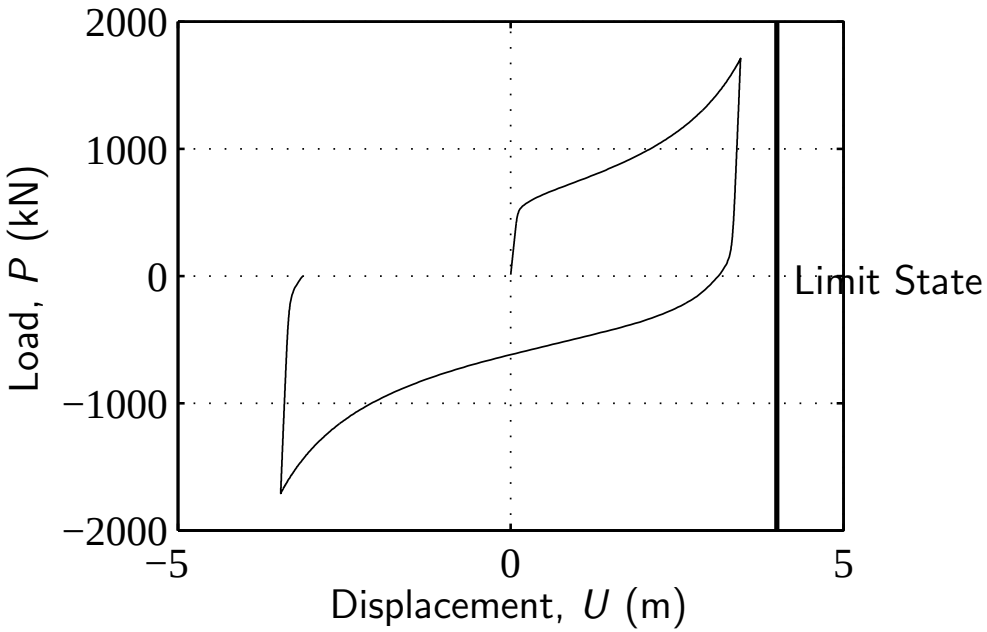


Uncertainty Analysis

Determine probability that peak displacement exceeds 4.0 m

$$g(\mathbf{x}) = 4.0 - U$$

$X_1 \sim LN(4.1e5, 0.05)$ MPa maps to yield stress, σ_y
$X_2 \sim LN(2.0e8, 0.1)$ MPa maps to elastic modulus, E
$X_3 \sim N(1710, 0.15)$ kN maps to applied load, P_{max}
$X_4 \sim N(5.0, 0.02)$ m maps to cantilever length, L
$X_5 \sim N(0.5, 0.04)$ m maps to cross-section depth, d



Random Variable Definition

- Assign distribution functions

```
randomVariable 1 lognormal $E [expr 0.05*$E]; set
param(1) 4
randomVariable 2 lognormal $fy [expr 0.1*$fy]; set
param(2) 2
randomVariable 3 normal $Pmax [expr 0.15*$Pmax];
set param(3) 8
randomVariable 4 normal $L [expr 0.02*$L]; set
param(4) 5
randomVariable 5 normal $d [expr 0.04*$d]; set
param(5) 10
```

- Then create RVPs through `foreach` loop shown on slide 19
- Performance Function

```
performanceFunction 1 "4.0-\[nodeDisp $Nele 2\]"
```

Monte Carlo Analysis

- Use the RVP pairs list, Tcl's random number generator, inverse CDF, and updateParameter commands

```
set Ntrials 10000; set Nfail 0
expr srand([clock clicks])
for {set i 1} {$i <= $Ntrials} {incr i} {
   reset
   foreach {rvTag pTag} [getRVPositioners] {
      set p [expr rand()]
      updateParameter [getInverseCDF $rvTag $p]
   }
   analyze
   if {[performanceFunction 1] <= 0.0} {incr Nfail}
}
puts ``pf = [expr double($Nfail)/$Ntrials]''
```

- Avoid integer division in the pf calculation!!!

Gradients of Performance Function

- Provide directions in search for design point
- Use built-in Tcl OpenSees commands

```
foreach {rvTag paramTag} [getRVPositioners] {
  gradPerformanceFunction 1 $rvTag "-\[sensNodeDisp
$Nele 2 $paramTag\]"
}
```

FORM Analysis Options

```
randomNumberGenerator CStdLib
probabilityTransformation Nataf -print 3
reliabilityConvergenceCheck Standard -e1 1.0e-2 -e2
1.0e-2 -print 1
gFunEvaluator Tcl "analyze $Nsteps"
gradGEvaluator OpenSees
searchDirection iHLRF
meritFunctionCheck AdkZhang -multi 2.0 -add 50 -factor
0.5
stepSizeRule Armijo -maxNum 10 -base 0.5 -initial 0.3 5
startPoint Mean
findDesignPoint StepSearch -maxNumIter 30
```

FORM Results

HLRF converges in 7 iterations

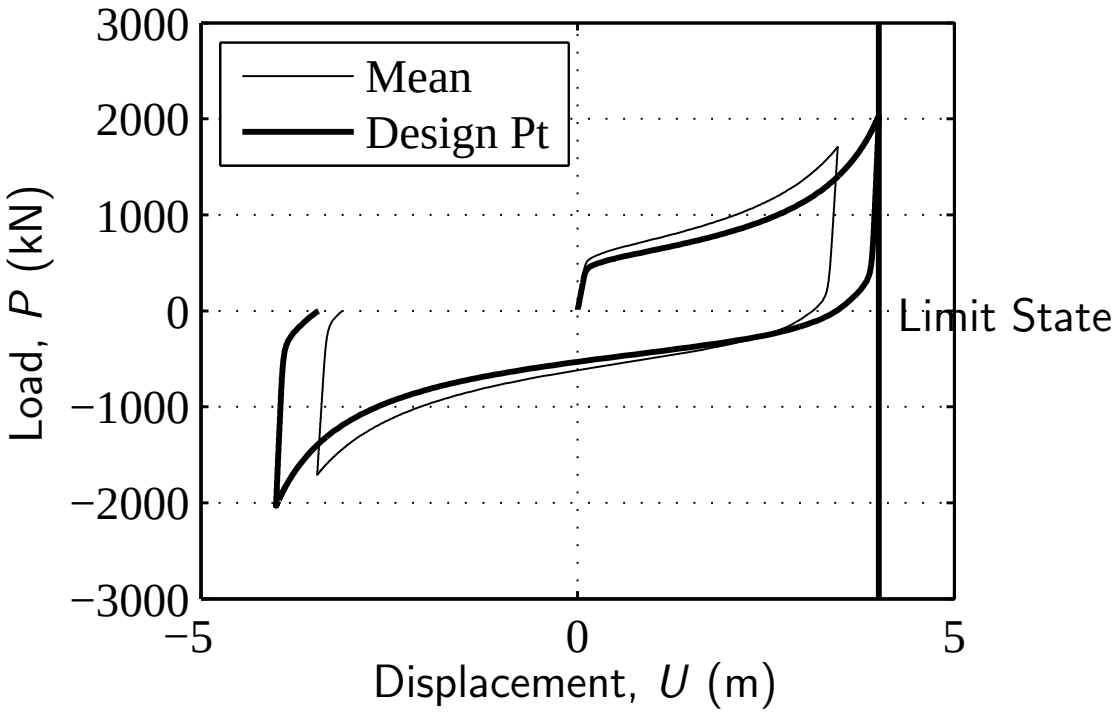
Reliability index, $\beta = 1.983$

Probability of failure, $p_f = \Phi(-1.983) = 2.368\%$

Random Variable, i		μ_i	X_i^*	α_i
1	σ_y	4.1e5 kPa	3.897e5 kPa	-0.2317
2	E	2.0e8 kPa	1.996e8 kPa	-0.005413
3	P_{max}	1710 kN	2032 kN	0.6393
4	L	5.0 m	5.100 m	0.5008
5	d	0.5 m	0.4787 m	-0.5356

Mean and Design Point Response

Cantilever load-displacement response at realization of random variables in $\mathbf{x}^*$



Concluding Remarks

- Continual process of extending OpenSees uncertainty modeling capabilities to wider range of applications
  - Bridge engineering
  - Soil-structure interaction
  - Fluid-structure interaction
- Many changes to underlying software architecture underway, but will be transparent to users with minimal deprecation
- Documentation and updated examples on OpenSees wiki page

Questions/Comments?

michael.scott@oregonstate.edu