

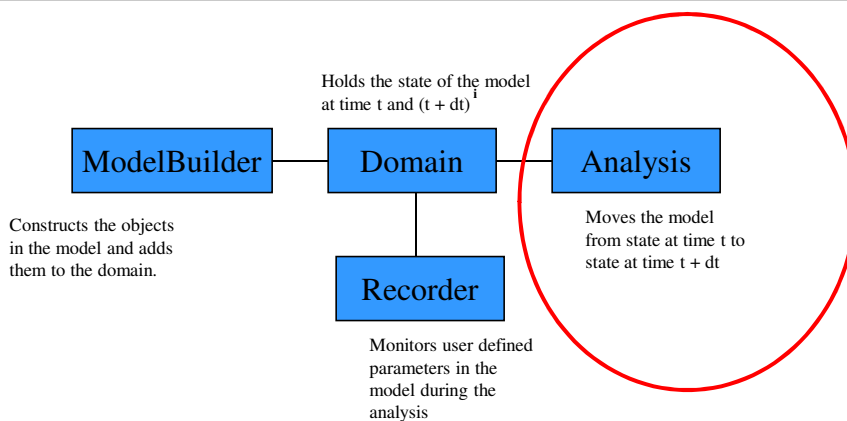
Introduction to Analysis Commands

Frank McKenna
UC Berkeley

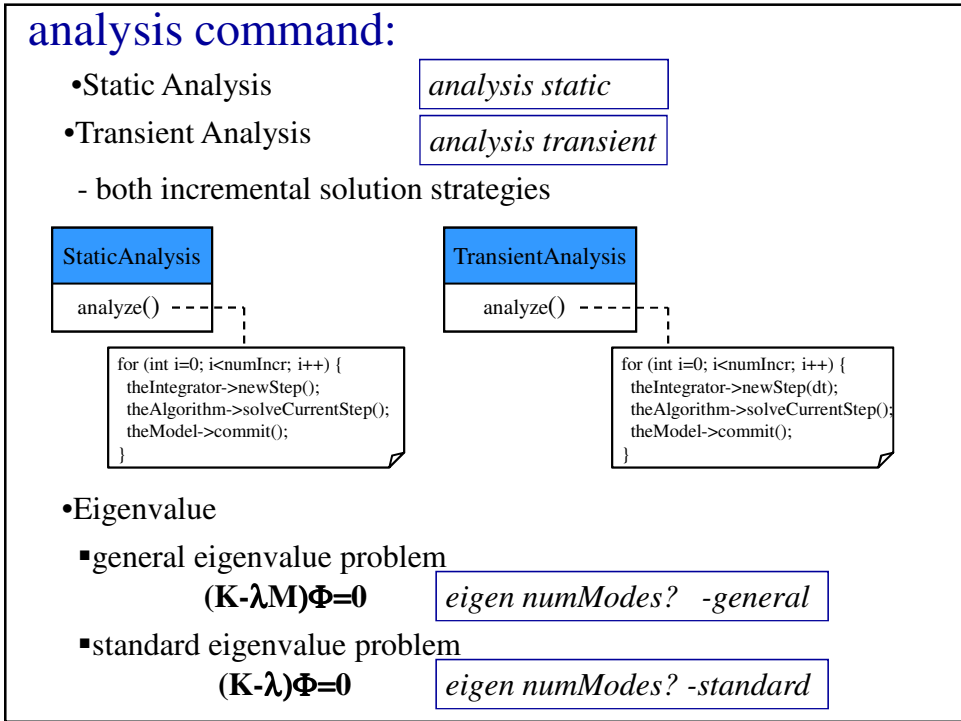
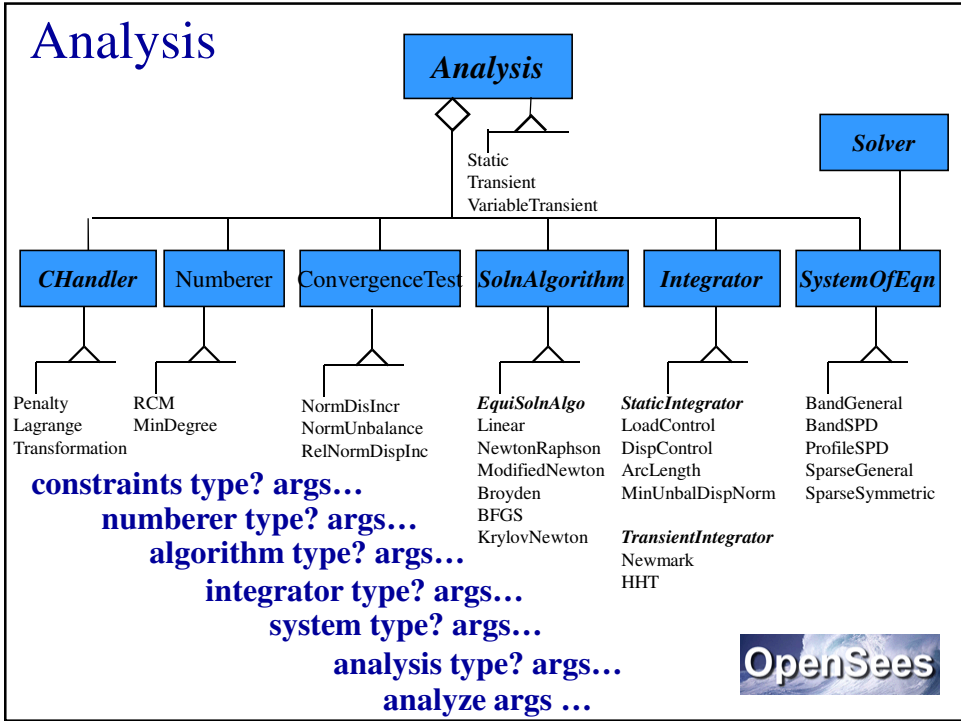
OpenSees User Workshop
September 8, 2008



Main Abstractions in OpenSees



In this presentation we focus on **ANALYSIS GENERATION**



integrator command:

- determines the predictive step for time $t+\delta t$
- specifies the tangent matrix and residual vector at any iteration
- determines the corrective step based on ΔU

•Transient Integrator for Use in Transient Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \dot{\mathbf{U}}, \ddot{\mathbf{U}}) = \mathbf{P}(t) - \mathbf{F}_I(\ddot{\mathbf{U}}) - \mathbf{F}_R(\mathbf{U}, \dot{\mathbf{U}})$$

▪Newmark Method

integrator Newmark $\gamma\beta$

▪Hilbert-Hughes-Taylor Method

integrator Newmark α

•Static Integrators for Use in Static Analysis

Nonlinear equation of the form:

$$\mathbf{R}(\mathbf{U}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{P}^* - \mathbf{F}\mathbf{R}(\mathbf{U})$$

▪Load Control

$$\boldsymbol{\lambda}_n = \boldsymbol{\lambda}_{n-1} + \Delta\boldsymbol{\lambda}$$

integrator LoadControl $\Delta\boldsymbol{\lambda}$

- *does not require a reference load, i.e. loads in load patterns with Linear series and all other loads constant.

▪Displacement Control

$$\mathbf{U}_{j_n} = \mathbf{U}_{j_{n-1}} + \Delta\mathbf{U}_j$$

integrator DisplacementControl node dof $\Delta\boldsymbol{\lambda}$

▪Arc Length

$$\Delta\mathbf{U}_n^T \Delta\mathbf{U}_n + \alpha^2 \Delta\boldsymbol{\lambda}_n^T \Delta\boldsymbol{\lambda}_n = \Delta s^2$$

integrator LoadControl $\alpha \Delta s$

▪Minimum Unbalance Displacement Norm

$$\frac{d(\Delta\mathbf{U}_n^T \Delta\mathbf{U}_n)}{d\Delta\boldsymbol{\lambda}} = \mathbf{0}$$

integrator LoadControl $\Delta\boldsymbol{\lambda}$

algorithm command:

- to specify the steps taken to solve the nonlinear equation

•Linear Algorithm

```
theIntegrator->formUnbalance();
theIntegrator->formTangent();
theSOE->solve()
theIntegrator->update(theSOE->getX());
```

algorithm Linear

•Newton-Raphson Algorithm

```
theIntegrator->formUnbalance();
do {
  theIntegrator->formTangent();
  theSOE->solve()
  theIntegrator->update(theSOE->getX());
  theIntegrator->formUnbalance();
} while (theTest->test() == fail)
```

algorithm Newton

•Modified Newton Algorithm

algorithm ModifiedNewton <-initial>

•Accelerated Modified Newton Algorithm

algorithm KrylovNewton <-initial>

constraints command:

- to specify how the constraints are enforced

$$\mathbf{U}_c = \mathbf{C}_r \mathbf{C}_c \mathbf{U}_r$$

$$\mathbf{C} \mathbf{U} = \mathbf{0}$$

$$\mathbf{T} \mathbf{U}_r = [\mathbf{U}_r \ \mathbf{U}_c]^T$$

$$[\mathbf{C}_r \ \mathbf{C}_c]^T [\mathbf{U}_r \ \mathbf{U}_c] = \mathbf{0}$$

•Transformation Handler

$$\mathbf{K}^* \mathbf{U}_r = \mathbf{R}^* \quad \begin{aligned} \mathbf{K}^* &= \mathbf{T}^T \mathbf{K} \mathbf{T} \\ \mathbf{R}^* &= \mathbf{T}^T \mathbf{R} \end{aligned}$$

constraints Transformation

in OpenSees currently don't allow retained node in one constraint to be a constrained node in another constraint

•Lagrange Handler

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{R} \\ \mathbf{Q} \end{bmatrix}$$

constraints Lagrange

•Penalty Handler

$$[\mathbf{K} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{C}] \mathbf{U} = [\mathbf{R} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{Q}] \quad \textit{constraints Penalty } \alpha_{sp} \text{? } \alpha_{mp} \text{?}$$

system command:

- to specify how matrix equation $KU = R$ is stored and solved

- Profile Symmetric Positive Definite (SPD)



system ProfileSPD

- Banded Symmetric Positive Definite



system BandSPD

- Sparse Symmetric Positive Definite



system SparseSPD

- Banded General



system BandGeneral

- Sparse Symmetric



system SparseGeneral

system Umfpack

numberer command:

- to specify how the degrees of freedom are numbered

- Plain Numberer

nodes are assigned dof arbitrarily

numberer Plain

- Plain Numberer

nodes are assigned dof using the
Reverse Cuthill-McKee algorithm

numberer RCM

test command:

- to specify when convergence has been achieved

all look at system: $\mathbf{KU} = \mathbf{R}$

- Norm Unbalance

$$\sqrt{\mathbf{R}^T \mathbf{R}} < \text{tol} \quad \text{test NormUnbalance tol? numIter? <flag?>}$$

- Norm Displacement Increment

$$\sqrt{\mathbf{U}^T \mathbf{U}} < \text{tol} \quad \text{test NormDispIncr tol? numIter? <flag?>}$$

- Norm Energy Increment

$$\frac{1}{2} (\mathbf{U}^T \mathbf{R}) < \text{tol} \quad \text{test NormEnergyIncr tol? numIter? <flag?>}$$

- Relative Tests

test RelativeNormUnbalance tol? numIter? <flag?>

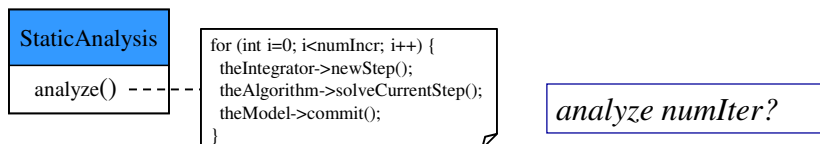
test RelativeNormDispIncr tol? numIter? <flag?>

test RelativeNormEnergyIncr tol? numIter? <flag?>

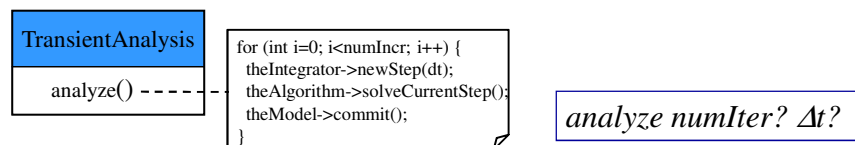
analyze command:

- to perform the static/transient analysis

- Static Analysis



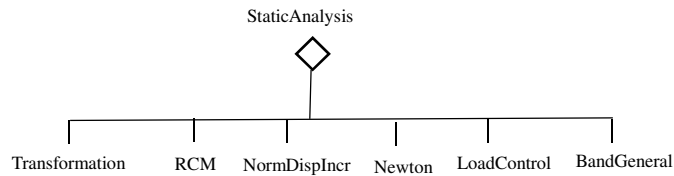
- Transient Analysis



Example Static Analysis:

- Static Nonlinear Analysis with LoadControl

```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator LoadControl 0.1
analysis Static
analyze 10
```



Example Dynamic Analysis:

- Transient Nonlinear Analysis with Newmark

```
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
analyze 2000 0.01
```

