

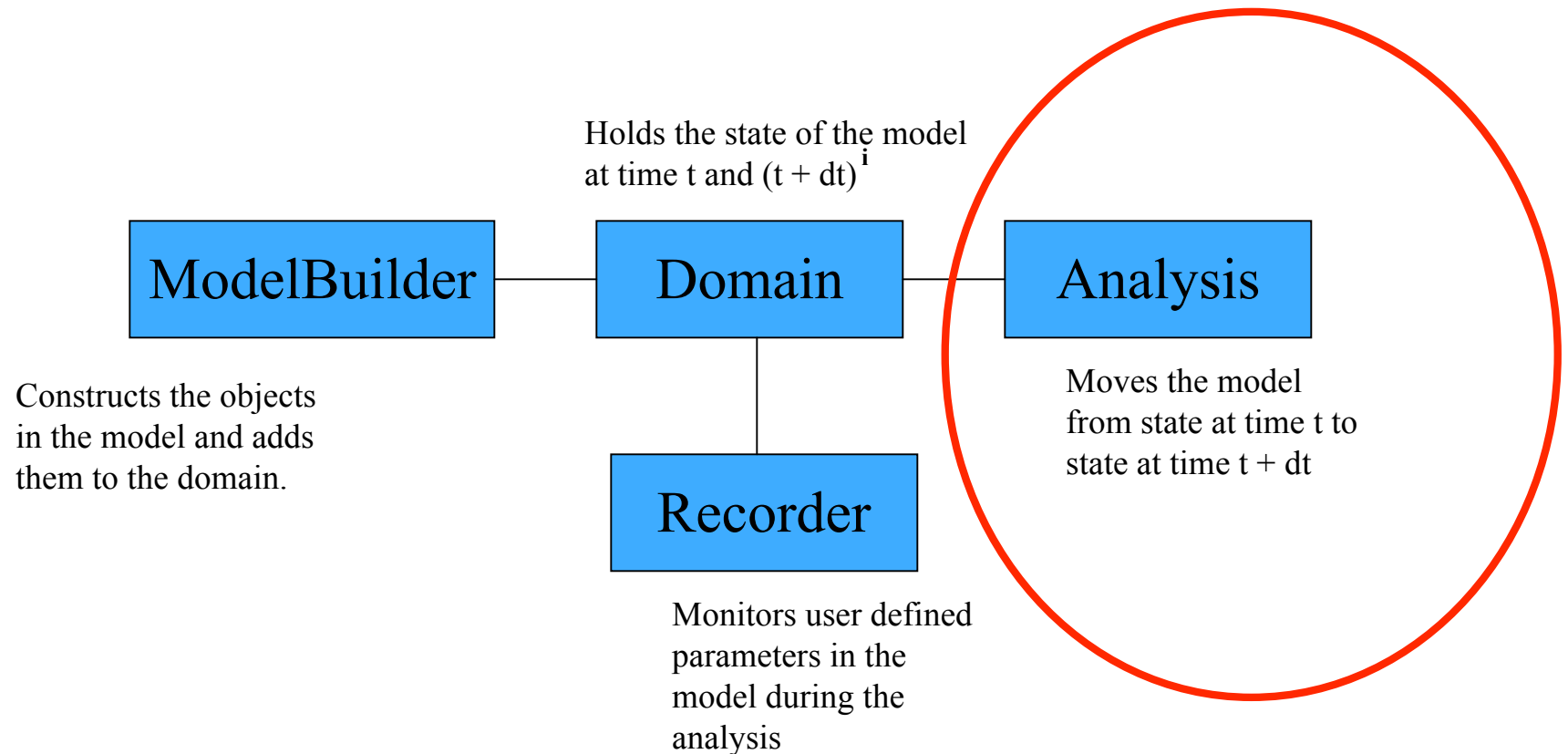
OpenSees: Analysis

Frank McKenna

NEES Hybrid Simulation Workshop
Berkeley, CA

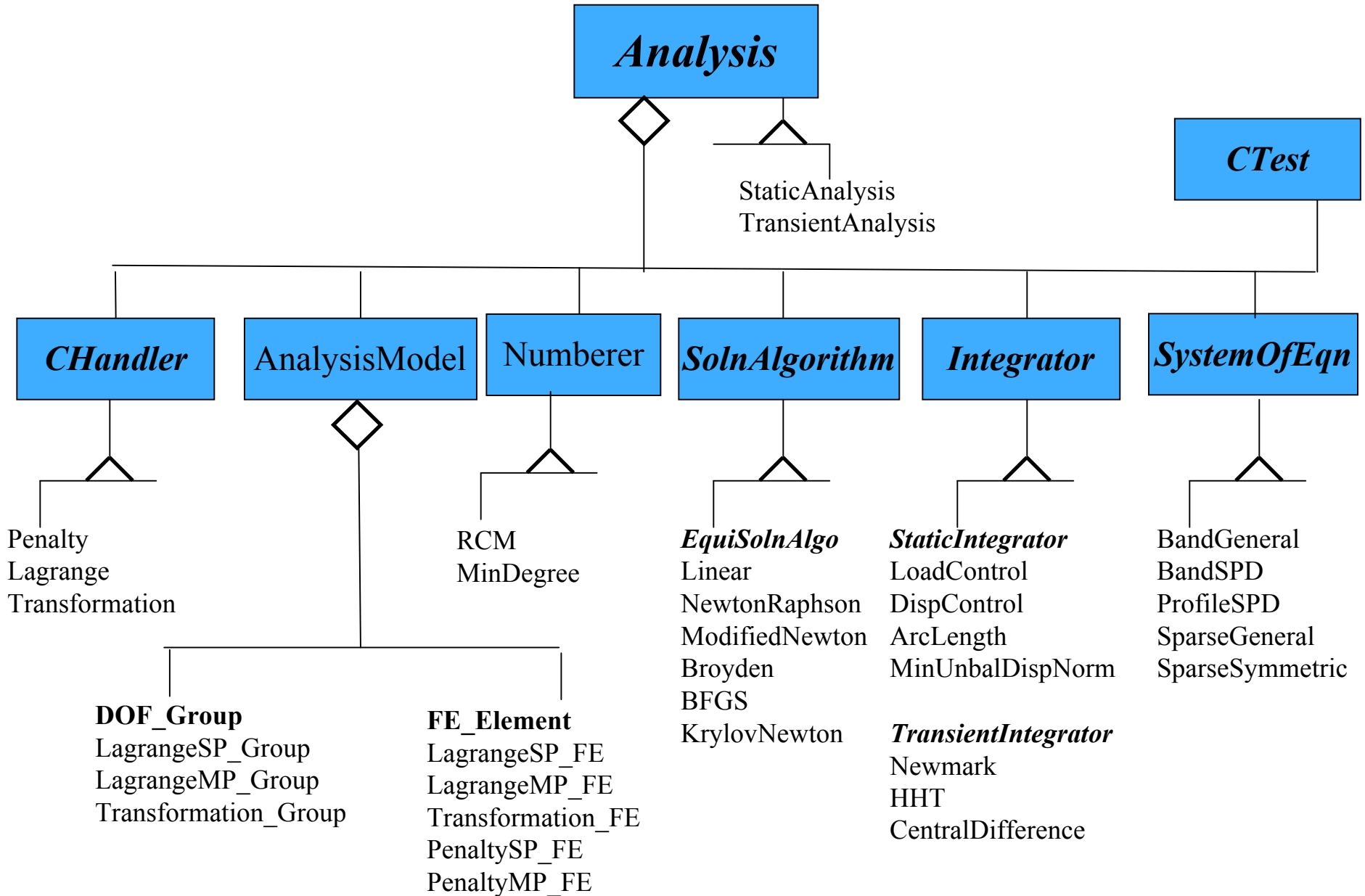


Main Abstractions in OpenSees



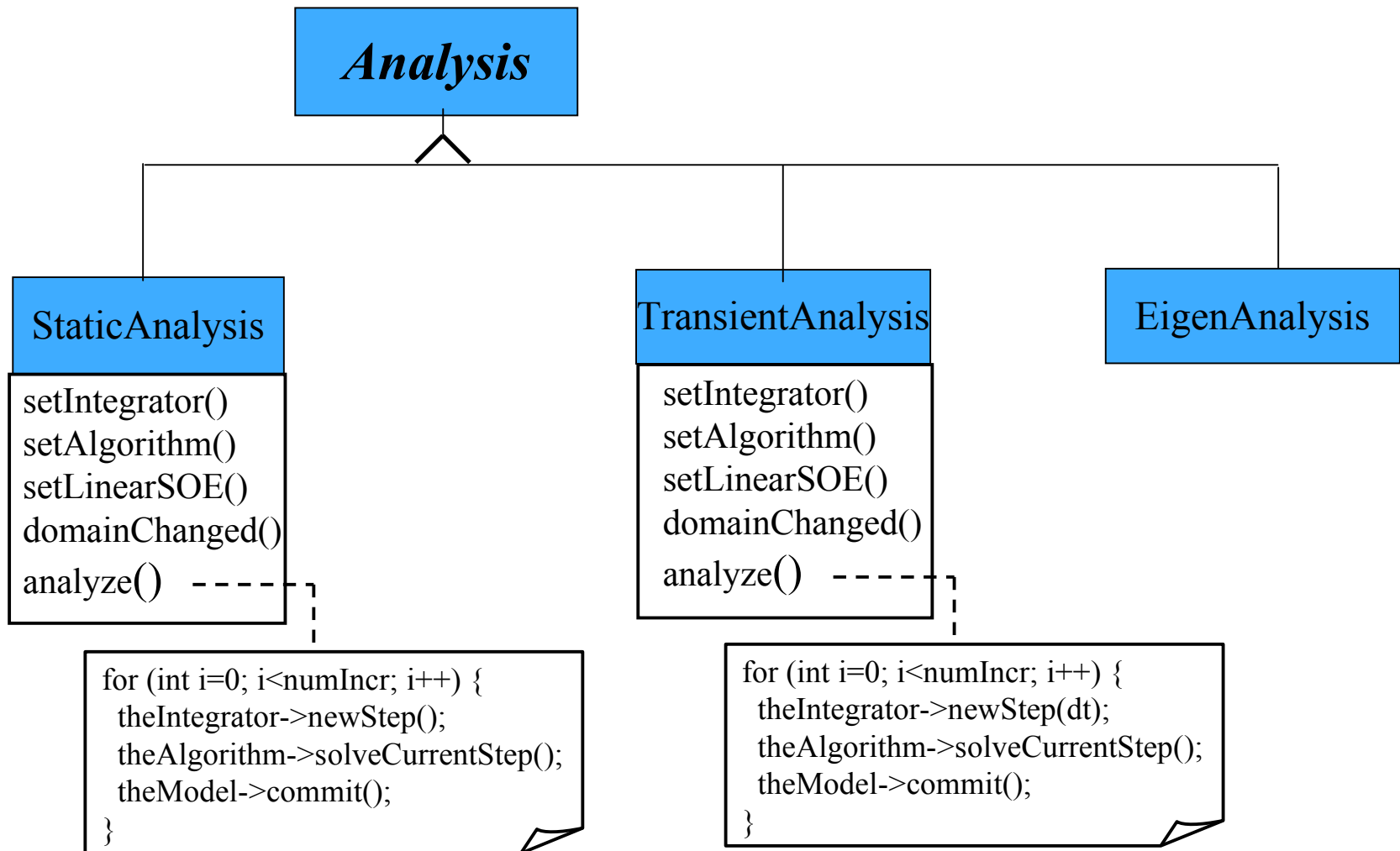
In this presentation we focus on the ANALYSIS

OpenSees Analysis



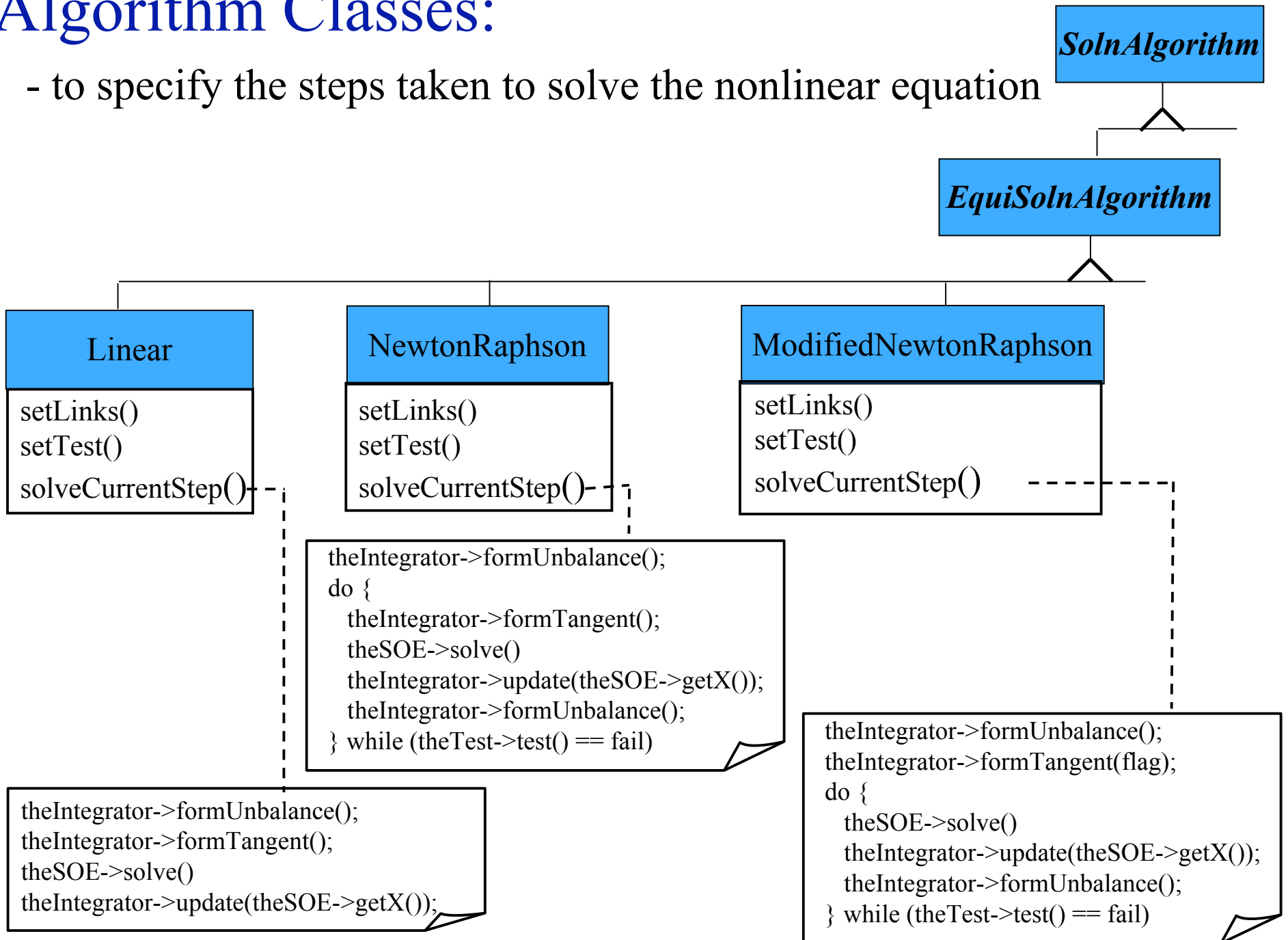
Analysis Classes:

- to update the state of the Domain



Algorithm Classes:

- to specify the steps taken to solve the nonlinear equation



Integrator Classes:

- 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:

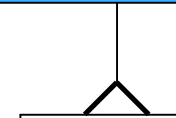
$$\mathbf{R}(\mathbf{U}, \dot{\mathbf{U}}, \ddot{\mathbf{U}}) = \mathbf{P}(t) - \mathbf{F}_I(\ddot{\mathbf{U}}) - \mathbf{F}_R(\mathbf{U}, \dot{\mathbf{U}})$$

•Static Integrators for Use in Static Analysis

Nonlinear equation of the form:

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

- Load Control $\lambda_n = \lambda_{n-1} + \Delta \lambda$
- Displacement Control $U_{jn} = U_{j, n-1} + \Delta U_j$
- Arc Length $\Delta U_n^T \Delta U_n + \alpha^2 \Delta \lambda_n^2 = \Delta s^2$



StaticIntegrator

LoadControl

DispControl

ArcLength

MinUnbalDispNorm

TransientIntegrator

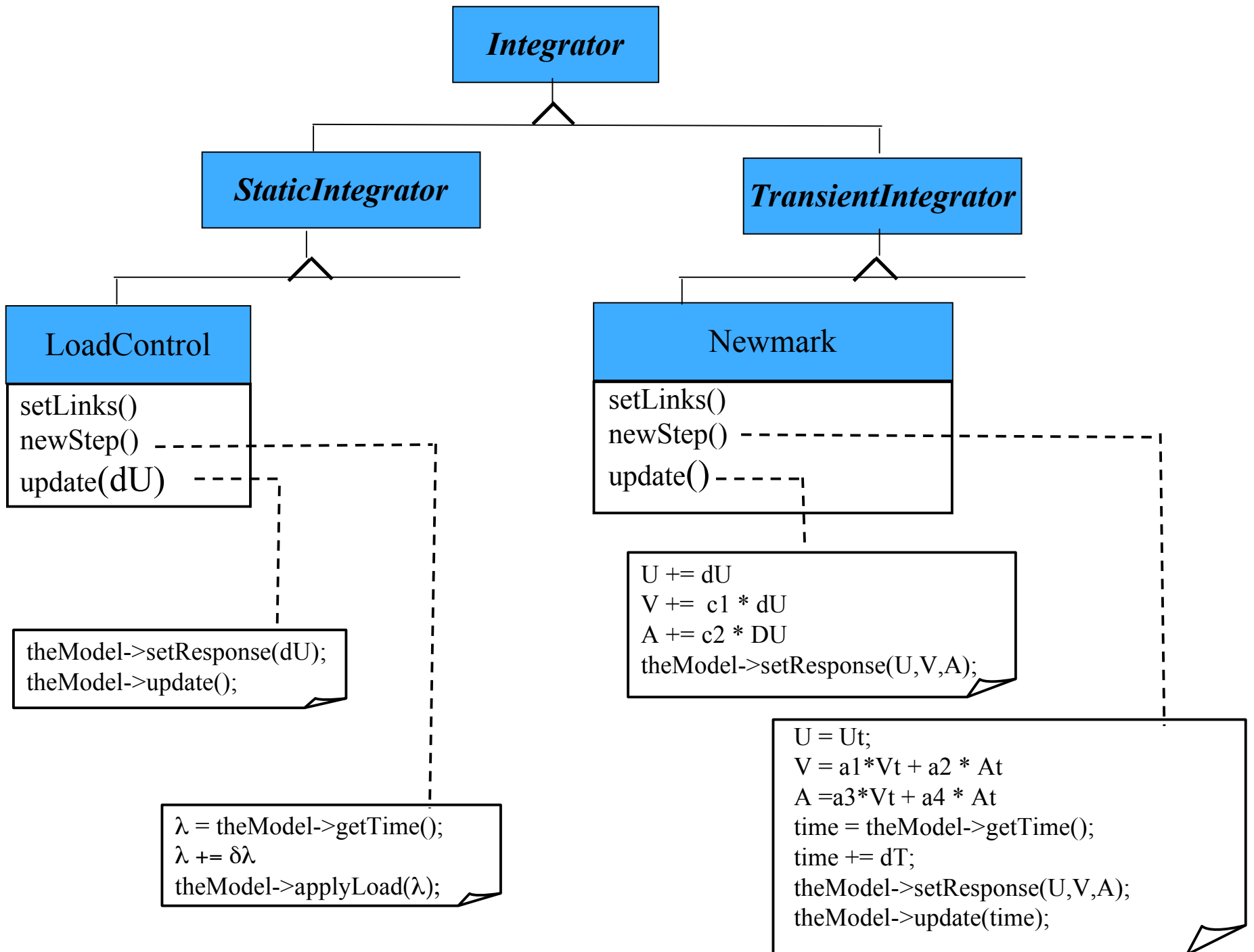
Newmark

HHT

CentralDifference (X3)

AlphaOS

NewmarkHybrid



ConstraintHandler

$$U_c = C_{rc} U_r$$

- to specify how the constraints are enforced $C U = 0$

$$T U_r = [U_r \ U_c]^T$$

ConstraintHandler

Transformation

handle()

forall nodes in Domain
if not constrained
construct and add DOF_Group
else
construct and add TDOF_Group
forall elements in Domain
if a node not constrained
construct and add FE_Element
else
construct and add TFE_Element

Lagrange

handle()

forall nodes in Domain
construct and add DOF_Group
forall elements in Domain
construct and add FE_Element
forall SP_Constraints in Domain
construct and add LagrangeSP_FE and
LagrangeDOF_Group
forall MP_Constraints in Domain
construct and add LagrangeMP{ }_FE
and a LagrangeDOF_Group

Penalty

handle()

forall nodes in Domain
construct and add DOF_Group
forall elements in Domain
construct and add FE_Element
forall SP_Constraints in Domain
construct and add PenaltySP_FE
forall MP_Constraints in Domain
construct and add PenaltyMP{ }_FE

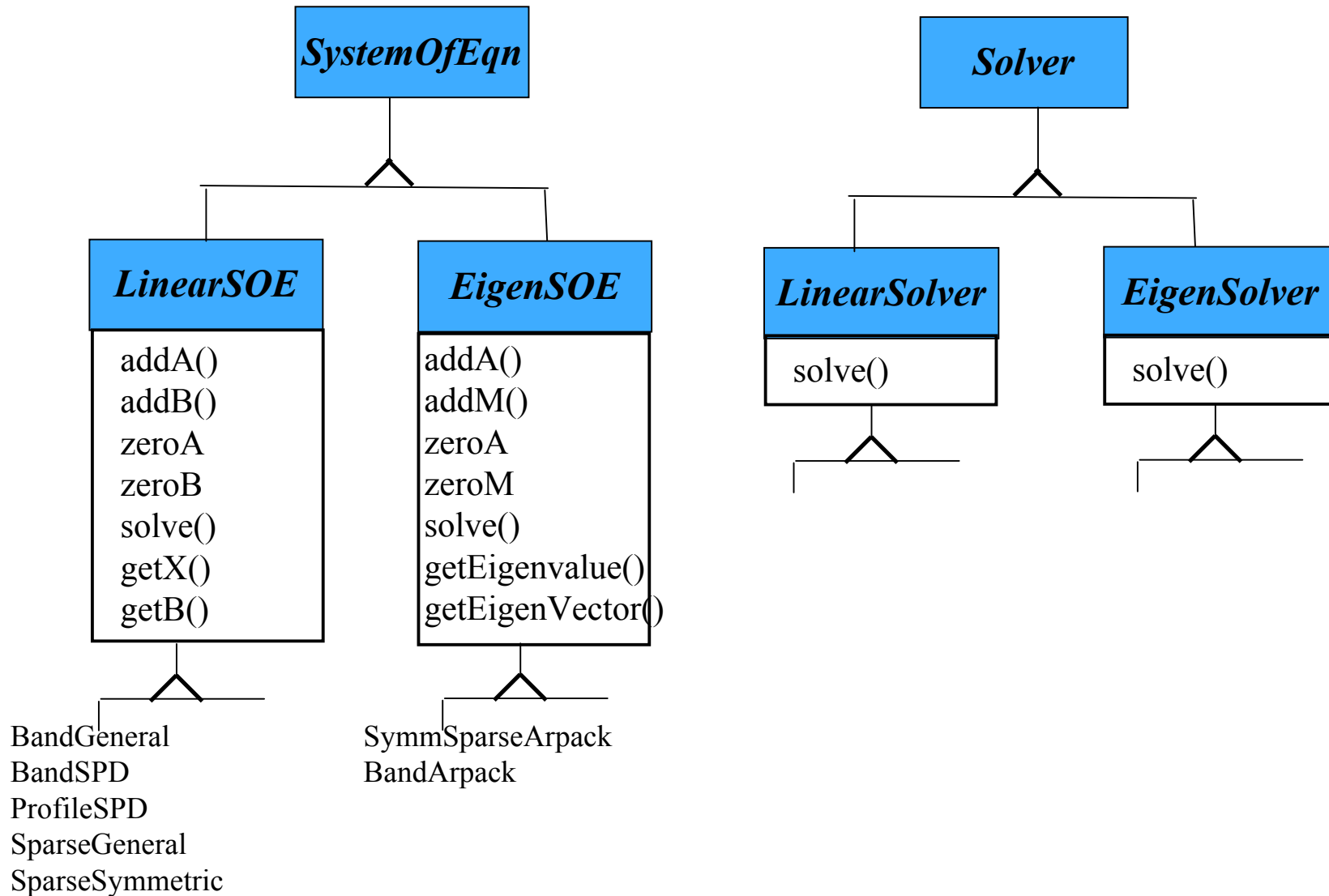
$$K^* U_r = R^*$$

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

$$[K + C^T \alpha C] U = [R + C^T \alpha Q]$$

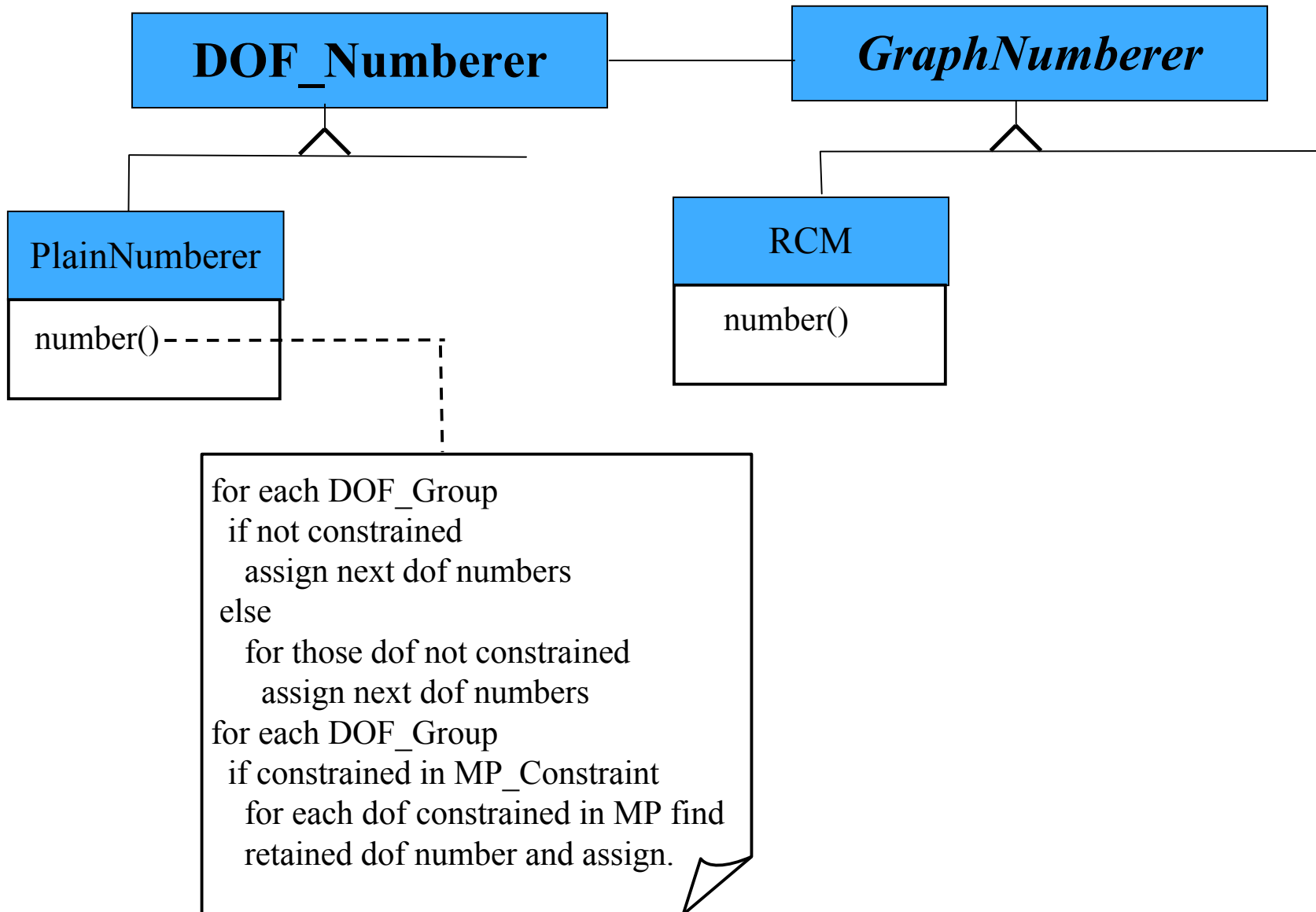
SystemOfEqn and Solver Classes:

- the SystemOfEqn classes store the matrix equations
- The Solver classes work on the SystemOfEqn classes to solve the eqn.



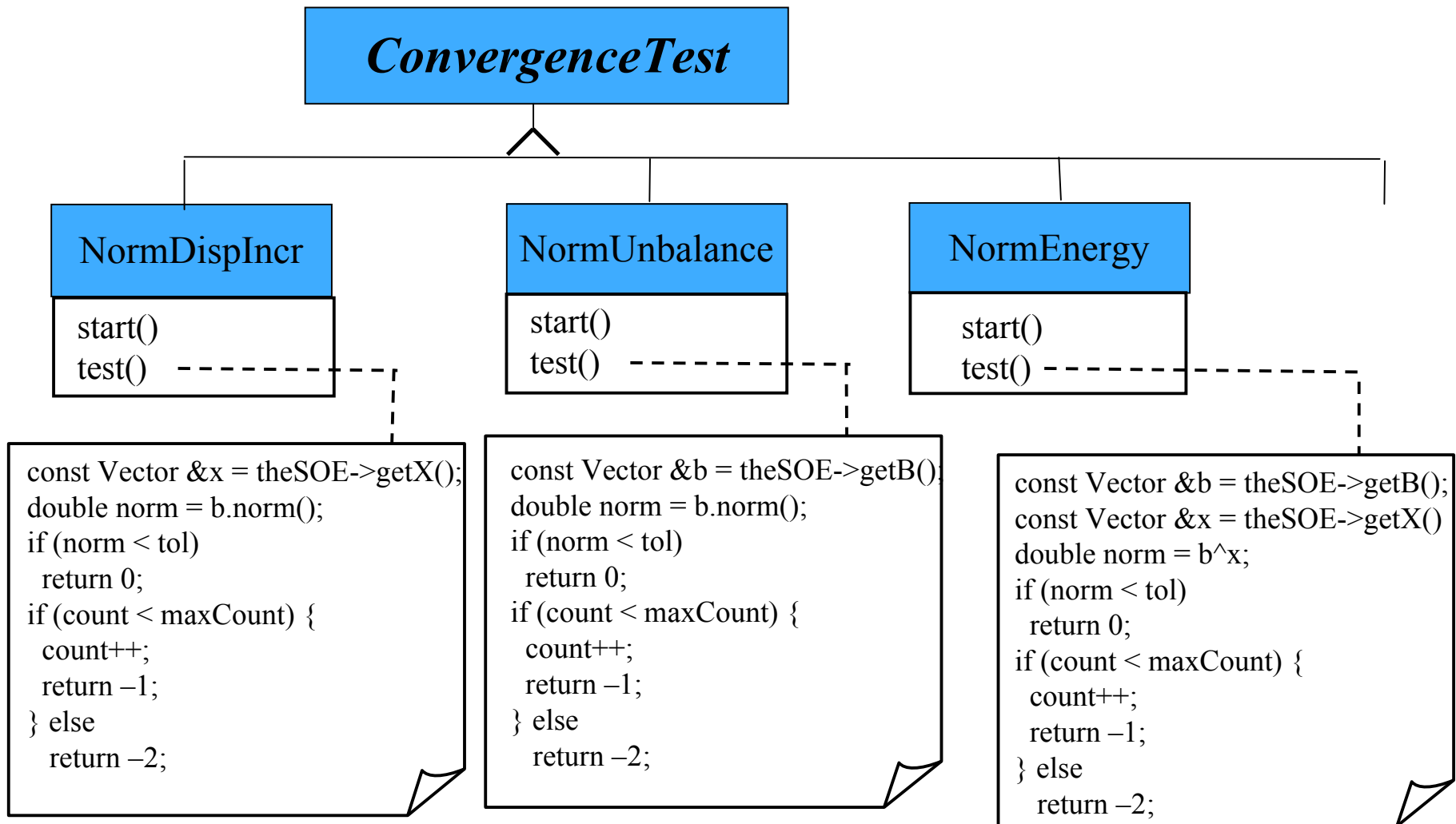
Numberer command:

- to specify how the degrees of freedom are numbered

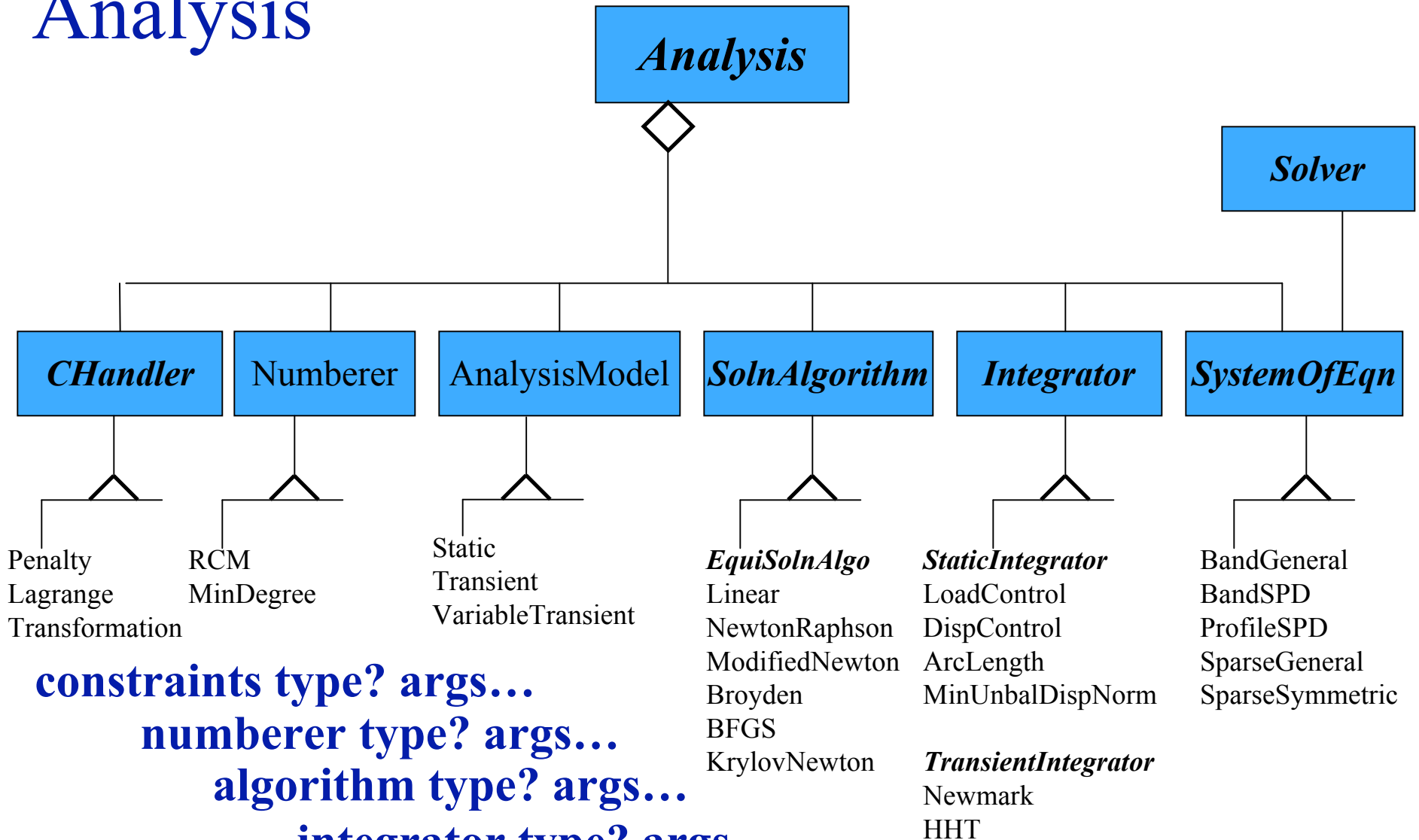


ConvergenceTest Classes:

- to determine if convergence has been achieved



Analysis



constraints type? args...

numberer type? args...

algorithm type? args...

integrator type? args...

system type? args...

analysis type? args...

analyze args ...



analysis command:

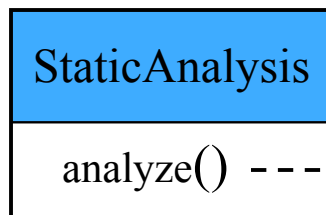
- Static Analysis

analysis static

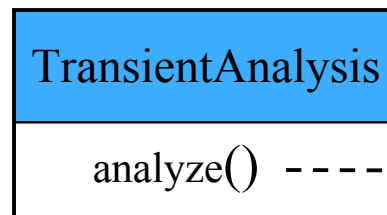
- Transient Analysis

analysis transient

- both incremental solution strategies



```
for (int i=0; i<numIncr; i++) {  
  theIntegrator->newStep();  
  theAlgorithm->solveCurrentStep();  
  theModel->commit();  
}
```



```
for (int i=0; i<numIncr; i++) {  
  theIntegrator->newStep(dt);  
  theAlgorithm->solveCurrentStep();  
  theModel->commit();  
}
```

- Eigenvalue

- general eigenvalue problem

$$(\mathbf{K}-\lambda\mathbf{M})\Phi=0$$

eigen numModes? -general

- standard eigenvalue problem

$$(\mathbf{K}-\lambda)\Phi=0$$

eigen numModes? -standard

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 HHT α

▪CentralDifference

integrator CentralDifference

▪Alpha Operator Splitting Method

Integrator AlphaOS $\alpha \beta \gamma$

•Static Integrators for Use in Static Analysis

Nonlinear equation of the form:

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

▪Load Control

$$\lambda_n = \lambda_{n-1} + \Delta\lambda$$

integrator LoadControl $\Delta\lambda$

*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 $\Delta\lambda$

▪Arc Length

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

integrator LoadControl $\alpha \Delta s$

▪Minimum Unbalance Displacement Norm

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

integrator LoadControl $\Delta\lambda$

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

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

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

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

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

•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

$$[\mathbf{K} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{C}] \mathbf{U} = [\mathbf{R} + \mathbf{C}^T \boldsymbol{\alpha} \mathbf{Q}]$$

constraints Penalty α_{sp} ? α_{mp} ?

system command:

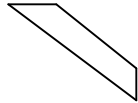
- to specify how matrix equation $KU = R$ is stored and solved

- Profile Symmetric Positive Definite (SPD)



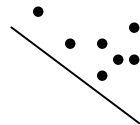
system ProfileSPD

- Banded Symmetric Positive Definite



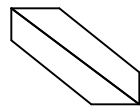
system BandSPD

- Sparse Symmetric Positive Definite



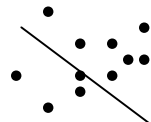
system SparseSPD

- Banded General



system BandGeneral

- Sparse Symmetric



system 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

$$\sqrt{\mathbf{R}^T \mathbf{R}} < \mathbf{tol}$$

test NormUnbalance tol? numIter? <flag?>

- Norm Displacement Increment

$$\sqrt{\mathbf{U}^T \mathbf{U}} < \mathbf{tol}$$

test NormDispIncr tol? numIter? <flag?>

- Norm Energy Increment

$$\frac{1}{2} (\mathbf{U}^T \mathbf{R}) < \mathbf{tol}$$

test NormEnergyIncr tol? numIter? <flag?>

- Relative Tests

test RelativeNormUnbalance tol? numIter? <flag?>

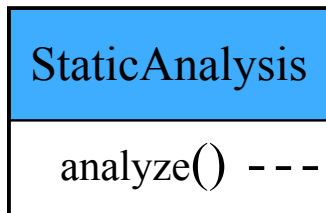
test RelativeNormDispIncr tol? numIter? <flag?>

test RelativeNormEnergyIncr tol? numIter? <flag?>

analyze command:

- to perform the static/transient analysis

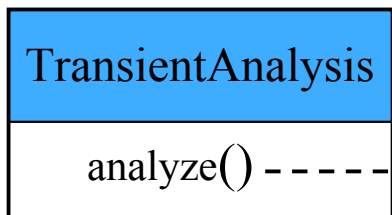
•Static Analysis



```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep();  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```

analyze numIter?

•Transient Analysis



```
for (int i=0; i<numIncr; i++) {  
    theIntegrator->newStep(dt);  
    theAlgorithm->solveCurrentStep();  
    theModel->commit();  
}
```

analyze numIter? Δt ?

Example Analysis: remember these!

- 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
```

- 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
```



Commands that Return Values

- analyze command

The analyze command returns 0 if successful.
It returns a negative number if not

```
set ok [analyze numIter <ΔT>]
```

- getTime command

The getTime command returns pseudo time in Domain.

```
set currentTime [getTime]
```

- nodeDisp command

The nodeDisp command returns a nodal displacement.

```
set disp [nodeDisp node dof]
```

Example Usage – Displacement Control

```
set maxU 15.0; set dU 0.1
constraints transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator DispControl 3 1 $dU
analysis Static
set ok 0
set currentDisp 0.0
while {$ok == 0 && $currentDisp < $maxU} {
    set ok [analyze 1]
    if {$ok != 0} {
        test NormDispIncr 1.0e-6 1000 1
        algorithm ModifiedNewton -initial
        set ok [analyze 1]
        test NormDispIncr 1.0e-6 6 2
        algorithm Newton
    }
    set currentDisp [nodeDisp 3 1]
}
}
```

Example Usage – Transient Analysis

```
set tFinal 15.0; set dT 0.01;
constraints Transformation
numberer RCM
system BandGeneral
test NormDispIncr 1.0e-6 6 2
algorithm Newton
integrator Newmark 0.5 0.25
analysis Transient
set ok 0
set currentTime 0.0
while {$ok == 0 && $currentTime < $tFinal} {
    set ok [analyze 1 $dT]
    if {$ok != 0} {
        test NormDispIncr 1.0e-6 1000 1
        algorithm ModifiedNewton -initial
        set ok [analyze 1 $dT]
        test NormDispIncr 1.0e-6 6 2
        algorithm Newton
    }
    set currentTime [getTime]
}
}
```