

MERCURY

Optimized Software for (Single Site) Hybrid Simulation
From Pseudo Dynamics to Real Time Hybrid Simulation

Theory Manual

Ver. 1

Dae-Hung Kang
Gary Haussmann
Victor E. Saouma

Fast Hybrid Testing Laboratory

<http://fht.colorado.edu>

Department of Civil Environmental and Architectural Engineering
University of Colorado, Boulder
Boulder, Colorado 80309-0428

Contents

1	ELEMENT FORMULATION	9
1.1	Truss Element	9
1.1.1	Formulation	9
1.1.2	Coordinate system for 2D truss element	10
1.1.3	State determination	11
1.2	Beam-Column Element	11
1.2.1	Stiffness-based 2D beam-column element	11
1.2.1.1	Formulation	11
1.2.1.2	Coordinate system for stiffness-based 2D beam-column element	13
1.2.1.3	State determination	13
1.2.1.4	Nonlinear analysis using stiffness-based formulation	17
1.2.2	Flexibility-based 2D beam-column element	21
1.2.2.1	Formulation	22
1.2.2.2	Coordinate system for flexibility-based 2D beam-column element	24
1.2.2.3	State determination	25
1.2.2.3.1	With element iterations	25
1.2.2.3.2	Without element iterations	32
1.2.2.4	Nonlinear analysis using flexibility-based method	34
1.2.2.4.1	With element iterations	34
1.2.2.4.2	Without element iterations	40
1.3	Layer/Fiber Section	45
1.4	Zero-Length 2D Element Formulation	47
1.4.1	Formulation	48
1.4.2	Coordinate system in zero-length 2D element	50
1.4.3	Element state determination	50
1.5	Zero-Length Section Element Formulation	52
1.5.1	Formulation	53
1.5.2	Coordinate system in zero-length 2D section element	54
1.5.3	Element state determination	54
1.6	Element Force	56
1.7	Summary	58
2	CONSTITUTIVE MODELS	59
2.1	Steel Models	59
2.1.1	“Exact” Models	60
2.1.1.1	Isotropic hardening model	60
2.1.1.2	Combined isotropic and kinematic hardening model	63
2.1.1.3	Determination for isotropic and kinematic harding parameters	66
2.1.2	Heuristic Models	66
2.1.2.1	Simplified Bilinear model with isotropic hardening	66
2.1.2.1.1	Stress-strain relation	66
2.1.2.1.2	Determination for simplified bilinear model	66
2.1.2.2	Giuffre-Menegotto-Pinto Model Modified by Filippou et al.	66
2.1.2.2.1	Stress-strain relationship	66
2.1.2.2.2	Determination for modified Giuffre-Menegotto-Pinto Model	72
2.2	Concrete Models	72
2.2.1	Heuristic Models	72
2.2.1.1	Modified Kent And Park Model	72

2.2.1.1.1	Stress-strain relation	72
2.2.1.1.2	Data preparation for modified Kent and Park model	81
2.2.1.1.3	Determination for modified Kent and Park model	82
2.2.2	“Exact” Models	82
2.2.2.1	Anisotropic damage model with effective damage and stiffness recovery	82
2.2.2.1.1	Constitutive model	82
2.2.2.1.2	Uniaxial multi-fiber formulation	85
2.2.2.1.3	Determination for anisotropic damage model	86
2.2.2.1.3.1	Tension loading:	86
2.2.2.1.3.2	Compressive loading:	87
2.2.2.2	Anisotropic damage model with permanent strains	90
2.2.2.2.1	Uniaxial numerical scheme	93
2.3	Bond-Slip	94
2.4	Lumped Plasticity Model	96
2.5	Constitutive models for zero length and zero length section elements	96
2.5.1	Zero Length Element	96
2.5.2	Zero length section element	98
2.5.3	Summary	99
2.6	Summary	99
3	Structural Modeling	101
3.1	Truss, Material Nonlinearity, Static and Dynamic	101
3.2	Uniaxial Concrete Nonlinear Element, Cyclic Displacement	105
3.3	Uniaxial Steel Nonlinear Element, Cyclic Displacement	108
3.4	Steel Beam-columns, Fiber Section, Static, Transient (HHT and Shing)	112
3.5	Zero-length and Beam Column, Nonlinear Steel Element, load control	116
3.6	Zero-length Section, and Beam Column, Fiber, Nonlinear Steel Element, load control	117
3.7	Beam-column, Fiber Section, Nonlinear Material, multi-d.o.f.s displacement control, Pushover Analysis	120
3.8	Reinforced Concrete Beam-Column, Fiber Section, Transient Analysis	123
3.9	Ex29: Full Reinforced Concrete Frame, HHT, Shing	130
3.9.0.1	Frame discretization and properties	130
4	NONLINEAR PUSHOVER ANALYSIS	141
4.1	pushover	141
4.2	Shakedown	141
4.3	Control Method	141
4.4	Classical Pushover	141
4.5	Improved Pushover Analysis Procedure	141
4.5.1	Load Control	143
4.5.2	Displacement Control	143
5	DYNAMIC ANALYSIS	147
5.1	Preliminaries	147
5.1.1	Variational Formulation	147
5.1.2	Mass Representation	148
5.1.2.1	Lumped mass	148
5.1.2.2	Consistent mass	148
5.1.3	Damping Representation	149
5.1.4	Euler Methods	150
5.2	Time Integration Methods	150
5.2.1	Newmark β method	151
5.2.1.1	Newmark β implicit method	152
5.2.2	Hilber-Hughes-Taylor Method	153
5.2.2.1	General Formulation	153
5.3	Free Vibration	156
6	Notation	159

List of Figures

1.1	Internal forces and corresponding displacements, 2D truss element	10
1.2	Internal forces and corresponding displacements, stiffness-based 2D element	13
1.3	State determination for stiffness-based method	15
1.4	State determination procedure for stiffness-based method	16
1.5	Flow chart of nonlinear analysis using stiffness-based method (1)	18
1.6	Flow chart of nonlinear analysis using stiffness-based method (2)	19
1.7	Flow chart of nonlinear analysis using stiffness-based method (3)	20
1.8	2D beam-column element without rigid body modes	22
1.9	Sign convention on section forces	23
1.10	Internal forces and corresponding displacement,flexibility-based 2D element	24
1.11	The relationship between rigid body modes and no rigid body modes	25
1.12	State determination procedure for flexibility-based method with Newton-Raphson iteration in the element	26
1.13	State determination procedure for flexibility-based method without Newton-Raphson iteration in the element	27
1.14	State determination for flexibility-based method with element iteration (?)	28
1.15	Element and section state determinations	29
1.16	State determination for flexibility-based method without element iteration	33
1.17	Flow chart of nonlinear analysis, flexibility-based w/ element iteration(1)	35
1.18	Flow chart of nonlinear analysis, flexibility-based w/ element iteration(2))	36
1.19	Flow chart of nonlinear analysis, flexibility-based w/ element iteration(3)	37
1.20	Flow chart of nonlinear analysis, flexibility-based w/o element iteration(1)	41
1.21	Flow chart of nonlinear analysis, flexibility-based w/o element iteration(2)	42
1.22	Flow chart of nonlinear analysis, flexibility-based w/o element iteration(3)	43
1.23	Stress and strain distribution of nonlinear material	46
1.24	Layer/fiber section	47
1.25	Layer/fiber section state determination	48
1.26	Zero-length 2D element(1)	49
1.27	Zero-length 2D element(2)	50
1.28	Flow chart of zero-length 2D element for element state determination	51
1.29	Zero-length 2D section element(1)	53
1.30	Zero-length 2D section element(2)	54
1.31	Flow chart of zero-length 2D section element for element state determination	55
1.32	Forces of an element	56
1.33	Nodal equivalent forces of an element	57
1.34	Example with element distributed forces and nodal displacement	58
2.1	Uniaxial stress-strain curve for steel	59
2.2	Idealized stress-strain response hardening material	60
2.3	The tangent modulus for isotropic hardening model	61
2.4	The evolution of elastic domain in isotropic hardening model	61
2.5	Isotropic and kinematic hardening plasticity	63
2.6	The evolution of elastic domain in isotropic and kinematic hardening model	65
2.7	Determination for isotropic and kinematic hardening model	67
2.8	Bilinear model	68
2.9	Determination for simplified bilinear model	69
2.10	Menegotto-Pinto steel model	70
2.11	Definition of curvature parameter R in Menegotto-Pinto steel model	70
2.12	Determination (1) for modified Giuffre-Menegotto-Pinto Model	72
2.13	Determination (2) for modified Giuffre-Menegotto-Pinto Model	73
2.14	Determination (3) for modified Giuffre-Menegotto-Pinto Model	74

2.15	Determination (4) for modified Giuffre-Menegotto-Pinto Model	75
2.16	Concrete material model in compression	76
2.17	Concrete material model under cyclic loading in compression	78
2.18	Concrete material model under cyclic loading in tension	80
2.19	Determination (1) for modified Kent and Park model	83
2.20	Determination (2) for modified Kent and Park model	84
2.21	Cracks induced by compression, tension, tension and compression	85
2.22	Strain-stress curve for anisotropic damage model without permanent strain	89
2.23	Determination (1) for the anisotropic damage model	90
2.24	Determination (2) for the anisotropic damage model	91
2.25	Determination (3) for the anisotropic damage model	92
2.26	Strain-stress curve for anisotropic damage model with permanent strain	94
2.27	Longitudinal bar bond stress, steel stress, and strain profiles, (?)	95
2.28	Stress slip curve	96
2.29	Hysteretic model with strength and stiffness deterioration, (?)	97
2.30	Compatibility of NA between beam-column and zero length fiber section	99
2.31	Zero length element and section	99
3.1	Ex10: Truss, ModGMP material, load control, HHT	101
3.2	Cyclic wave without dimension	106
3.3	Examples 11- 12	106
3.4	Examples 16- 19	108
3.5	Examples 20- 22	112
3.6	Examples 23	116
3.7	Examples 24	118
3.8	Examples 25- 26	120
3.9	Beam-column elements for Ex27 and Ex28	123
3.10	Bar slip zero-length fiber section element (?)	124
3.11	Numerical model for real-time hybrid simulation	131
3.12	Shake table test of reinforce concrete frame, ?	132
3.13	Section properties of Ex29	132
3.14	Seismic excitation for Ex29	132
4.1	Pushover analysis	142
4.2	Determination of pushover displacements magnitudes	142
4.3	Pushover Analysis of a Frame	142
4.4	Implementation for displacement control	145
5.1	Rayleigh damping	149
5.2	Geometric interpretation of Euler's method	151
5.3	Flow chart (1) using the Newmark β implicit method	154
5.4	Flow chart (2) using the Newmark β implicit method	155
5.5	Flow chart using the HHT implicit method	157

List of Tables

3.1	Features of Example Problems	102
3.2	Material properties of concrete for Ex27 (unit: kips, in)	124
3.3	Material properties of concrete for Ex28 (unit: kips, in)	124
3.4	Material properties of reinforcement for Ex27 and Ex28 (unit: kips, in)	124
3.5	Property of shear spring in zero-length element for Ex27 and Ex28 (unit: kips, in)	124
3.6	Concrete material properties of concrete for Ex29 (unit:kips, in)	130
3.7	Material properties of reinforcement for Ex29 (unit: kips, in)	131
3.8	Property of shear spring for Ex29 (unit: kips, in)	131
3.9	Properties of rigid elements (unit: kips, in)	132
3.10	Mass properties in first floor column nodes (unit: $kips \cdot sec^2/in$)	133
3.11	Mass properties in column nodes except first floor columns (unit: $kips \cdot sec^2/in$)	133
3.12	Mass properties in beam nodes (unit: $kips \cdot sec^2/in$)	133
5.1	Properties of the Newmark β method	152

Chapter 1

ELEMENT FORMULATION

This chapter will describe the formulation of the elements in Mercury. It will start with the simplest one, truss and then proceed with beam-column of increasing levels of complexity. Finally zero length element and section element will be addressed.

1.1 Truss Element

The truss element has only one degree of freedom associated with each node in local reference whether in 2D or 3D.

1.1.1 Formulation

As with all finite elements, stiffness matrix derivation hinges on three requirements.

Compatibility of:

Displacement Section displacements are determined from the element nodal displacements through the shape function. Since the truss element has only one d.o.f per node, the generalized relationship between section displacement vector $\mathbf{d}_s(x)$ and element nodal displacement vector $\bar{\mathbf{d}}_e$ is expressed through the displacement interpolation functions (shape functions), $\mathbf{N}_d(x)$ as

$$\begin{aligned}\mathbf{d}_s(x) &= \{ u(x) \} \\ &= \underbrace{\left[-\frac{x}{L} + 1 \quad \frac{x}{L} \right]}_{\mathbf{N}_d(x)} \cdot \underbrace{\begin{Bmatrix} \bar{u}_{x1} \\ \bar{u}_{x2} \end{Bmatrix}}_{\bar{\mathbf{d}}_e}\end{aligned}$$

Deformation of displacements: Under the assumption that displacements are small, the section deformation vector $\boldsymbol{\varepsilon}_s(x)$ is related to the element nodal displacement vector by

$$\begin{aligned}\boldsymbol{\varepsilon}_s(x) &= \{ \varepsilon_x(x) \} \\ &= \underbrace{\left[-\frac{1}{L} \quad \frac{1}{L} \right]}_{\mathbf{B}_d(x)} \cdot \bar{\mathbf{d}}_e\end{aligned}\tag{1.1}$$

where $\mathbf{B}_d(x)$ is the matrix which relates displacement to strain through the derivatives of $\mathbf{N}_d(x)$.

Constitutive law is expressed as

$$\underbrace{\{ N_x(x) \}}_{\boldsymbol{\sigma}_s(x)} = \mathbf{k}_s(x) \cdot \boldsymbol{\varepsilon}_s(x)\tag{1.2}$$

where $\boldsymbol{\sigma}_s(x)$ is the section¹ force vector, and $\mathbf{k}_s(x)$ is the section stiffness matrix.

For linear elastic analysis $\mathbf{k}_s(x)$ is simply a scalar equal to

$$\mathbf{k}_s(x) = [E(x) \cdot A(x)]$$

where, $E(x)$ and $A(x)$ are elastic modulus and cross sectional area.

Equilibrium (weak form) through the principle of virtual work (displacement) which is expressed as

$$\underbrace{\delta \bar{\mathbf{d}}_e^T \cdot \bar{\mathbf{f}}_e}_{\text{External}} = \underbrace{\int_0^{L_e} \delta \boldsymbol{\varepsilon}_s(x)^T \boldsymbol{\sigma}_s(x) dx}_{\text{Internal}}\tag{1.3}$$

¹The notion of section is not essential to understand the formulation of the truss element stiffness matrix. It is nevertheless introduced to be consistent with the subsequent formulation of beam-column (Sec. 1.2)

Substitution of Eq. 1.1 and 1.2 into Eq. 1.3 and since the latter must hold for any arbitrary $\delta \bar{\mathbf{d}}_e$ leads to

$$\begin{aligned} \delta \bar{\mathbf{d}}_e^T \cdot \bar{\mathbf{f}}_e &= \int_0^{L_e} \delta \bar{\mathbf{d}}_e^T \cdot \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \boldsymbol{\varepsilon}_s(x) dx \\ \bar{\mathbf{f}}_e &= \int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \boldsymbol{\varepsilon}_s(x) dx \\ &= \underbrace{\int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \mathbf{B}_d(x) dx}_{\bar{\mathbf{k}}_e} \cdot \bar{\mathbf{d}}_e \\ \bar{\mathbf{f}}_e &= \bar{\mathbf{k}}_e \cdot \bar{\mathbf{d}}_e \end{aligned}$$

The element stiffness matrix in local reference is thus given by

$$\bar{\mathbf{k}}_e = \int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \mathbf{B}_d(x) dx$$

1.1.2 Coordinate system for 2D truss element

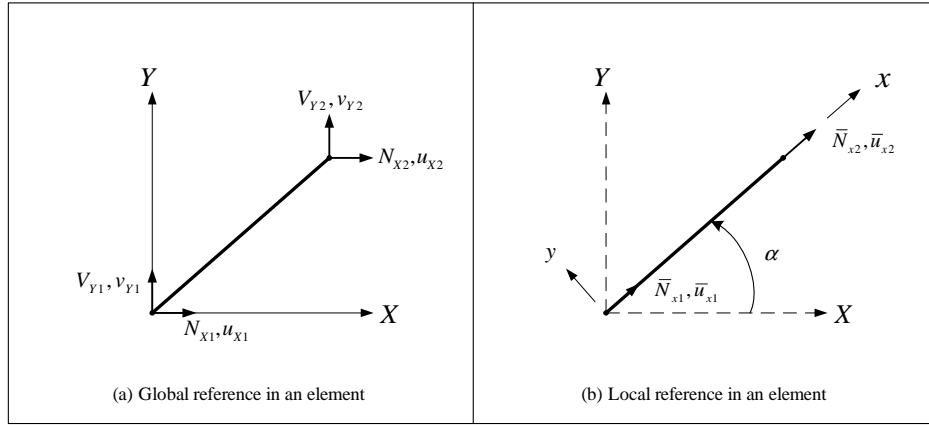


Figure 1.1: Internal forces and corresponding displacements in global and local coordinate system for the 2D truss element

The element nodal forces and displacements are expressed with respect to the global reference, Fig. 1.1(a) as:

$$\begin{aligned} \mathbf{F}_e &= [N_{X1}, V_{Y1}, N_{X2}, V_{Y2}]^T \\ \boldsymbol{\delta}_e &= [u_{X1}, v_{Y1}, u_{X2}, v_{Y2}]^T \end{aligned}$$

Furthermore, the element nodal forces and displacements can also be expressed with respect to the local reference, Fig. 1.1(b)

$$\begin{aligned} \bar{\mathbf{f}}_e &= [\bar{N}_{x1}, \bar{N}_{x2}]^T \\ \bar{\mathbf{d}}_e &= [\bar{u}_{x1}, \bar{u}_{x2}]^T \end{aligned}$$

The rotation matrix which transform global reference, Fig. 1.1(a), to local reference, Fig. 1.1(b), is given by $\boldsymbol{\Gamma}_e$ such that

$$\begin{aligned} \bar{\mathbf{f}}_e &= \boldsymbol{\Gamma}_e \cdot \mathbf{F}_e \\ \bar{\mathbf{d}}_e &= \boldsymbol{\Gamma}_e \cdot \boldsymbol{\delta}_e \\ \mathbf{K}_e &= \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{k}}_e \cdot \boldsymbol{\Gamma}_e \end{aligned}$$

where, \mathbf{K}_e is the element stiffness matrix in global reference and the rotation matrix is

$$\mathbf{\Gamma}_e = \begin{bmatrix} \frac{N_{x1}}{N_{x2}} & N_{X1} & V_{Y1} & N_{X2} & V_{Y2} \\ \cos \alpha & \sin \alpha & 0 & 0 & 0 \\ 0 & 0 & \cos \alpha & \sin \alpha & 0 \end{bmatrix}$$

1.1.3 State determination

In a nonlinear structural analysis, state determination for 2D truss element is identical to the stiffness-based 2D beam-column element. It will be discussed in Sec. 1.2.1.3 and 1.2.1.4.

1.2 Beam-Column Element

The formulation of nonlinear frame structure using elements with layer/fiber sections can be achieved using the stiffness-based (displacement) method or the flexibility-based (force) method. Both formulations will be presented, with the assumption that deformations are small and plane sections remain plane during the loading history. The stiffness-based method assumes displacement interpolation functions along the element, while the flexibility-based one has force interpolation functions along the element. As will be shown later, a smaller number of flexibility based elements (albeit with a greater number of integration points) than stiffness based ones is often needed to achieve the same level of accuracy.

However, the element formulation in the flexibility-based method is more complex than that of the stiffness-based one as we need to perform an additional loop of iterations at the element level in order to obtain ε from material constitutive models which are usually expressed as $\sigma = \sigma(\varepsilon)$ (?). On the other hand, stiffness based elements are much simpler to formulate.

Both formulations are described next.

1.2.1 Stiffness-based 2D beam-column element

We next formulate the stiffness matrix of the beam-column elements (also known as “frame” element) by considering the “classical” stiffness-based formulation first. The formulation is based on linear polynomial interpolation for the axial displacements, and cubic for the transverse ones.

1.2.1.1 Formulation

As for the truss element we consider each of the three requirements which must be met.

Compatibility of

Displacement Section displacements are determined from the element nodal displacements through the shape functions. The generalized relationship between section displacement vector $\mathbf{d}_s(x)$ and the element nodal displacement vector $\bar{\mathbf{d}}_e$ can be expressed as

$$\mathbf{d}_s(x) = \begin{Bmatrix} u(x) \\ v(x) \end{Bmatrix} = \mathbf{N}_d(x) \cdot \underbrace{\left[\bar{u}_{x1}, \bar{v}_{y1}, \bar{\theta}_{z1}, \bar{u}_{x2}, \bar{v}_{y2}, \bar{\theta}_{z2} \right]^T}_{\bar{\mathbf{d}}_e}$$

where $\mathbf{N}_d(x)$ is the matrix of displacement interpolation functions which can be expressed as

$$\mathbf{N}_d(x) = \begin{bmatrix} \psi_1(x) & 0 & 0 & \psi_2(x) & 0 & 0 \\ 0 & \phi_1(x) & \phi_2(x) & 0 & \phi_3(x) & \phi_4(x) \end{bmatrix}$$

where ψ_1 , ψ_2 , ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 are the interpolation functions for axial and transverse displacements respectively and are given by

$$\begin{aligned} \psi_1(x) &= -\frac{x}{L_e} + 1 & \psi_2(x) &= \frac{x}{L_e} \\ \phi_1(x) &= 2\frac{x^3}{L_e^3} - 3\frac{x^2}{L_e^2} + 1 & \phi_2(x) &= \frac{x^3}{L_e^2} - 2\frac{x^2}{L_e} + x \\ \phi_3(x) &= -2\frac{x^3}{L_e^3} + 3\frac{x^2}{L_e^2} & \phi_4(x) &= \frac{x^3}{L_e^2} - \frac{x^2}{L_e} \end{aligned}$$

We note the uncoupling between axial and transverse displacements since geometric nonlinearity is ignored.

Deformation Under the assumptions of small displacements and plane sections remaining plane (Euler Bernoulli as opposed to Timoshenko), the section deformation vector $\boldsymbol{\varepsilon}_s(x)$ (axial strain $\varepsilon_x(x)$ and curvature $\phi_z(x)$) is related to the element nodal displacement vector

$$\begin{aligned}\boldsymbol{\varepsilon}_s(x) &= \begin{Bmatrix} \varepsilon_x(x) \\ \phi_z(x) \end{Bmatrix} \\ &= \mathbf{B}_d(x) \cdot \bar{\mathbf{d}}_e\end{aligned}\tag{1.4}$$

where $\mathbf{B}_d(x)$ is the matrix obtained from the appropriate derivatives of the displacement interpolation functions.

$$\mathbf{B}_d(x) = \begin{bmatrix} \psi_1'(x) & 0 & 0 & \psi_2'(x) & 0 & 0 \\ 0 & \phi_1''(x) & \phi_2''(x) & 0 & \phi_3''(x) & \phi_4''(x) \end{bmatrix}$$

with

$$\begin{aligned}\psi_1'(x) &= -\frac{1}{L_e} & \psi_2'(x) &= \frac{1}{L_e} \\ \phi_1''(x) &= \frac{12x}{L_e^3} - \frac{6}{L_e^2} & \phi_2''(x) &= \frac{6x}{L_e^2} - \frac{4}{L_e} \\ \phi_3''(x) &= -\frac{12x}{L_e^3} + \frac{6}{L_e^2} & \phi_4''(x) &= \frac{6x}{L_e^2} - \frac{2}{L_e}\end{aligned}$$

Constitutive law

Section constitutive law relates axial strain and curvature to axial force and moment

$$\underbrace{\begin{Bmatrix} N_x(x) \\ M_z(x) \end{Bmatrix}}_{\boldsymbol{\sigma}_s(x)} = \mathbf{k}_s(x) \boldsymbol{\varepsilon}_s(x)\tag{1.5}$$

where $\boldsymbol{\sigma}_s(x)$ is the section force vector, and $\mathbf{k}_s(x)$ is the section stiffness matrix.

If $\mathbf{k}_s(x)$ is not derived from layer/fiber discretization of the cross section, and for linear elastic case $\mathbf{k}_s(x)$ is simply equal to

$$\mathbf{k}_s(x) = \begin{bmatrix} E(x) \cdot A(x) & 0 \\ 0 & E(x) \cdot I_z(x) \end{bmatrix}\tag{1.6}$$

where, $E(x)$, $A(x)$, and $I_z(x)$ are elastic modulus, cross sectional area, and section moment of inertia.

Equilibrium will be satisfied only in the weak sense through the principle of virtual displacement expressed as

$$\underbrace{\delta \bar{\mathbf{d}}_e^T \cdot \bar{\mathbf{f}}_e}_{\text{External}} = \underbrace{\int_0^{L_e} \delta \boldsymbol{\varepsilon}_s(x)^T \cdot \boldsymbol{\sigma}_s(x) dx}_{\text{Internal}}\tag{1.7}$$

Substituting Eq. 1.4 and Eq. 1.5 into Eq. 1.7 and since the latter must hold for any arbitrary $\delta \bar{\mathbf{d}}_e$, the principle of virtual work leads to

$$\begin{aligned}\delta \bar{\mathbf{d}}_e^T \cdot \bar{\mathbf{f}}_e &= \int_0^{L_e} \delta \bar{\mathbf{d}}_e^T \cdot \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \boldsymbol{\varepsilon}_s(x) dx \\ \bar{\mathbf{f}}_e &= \int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \boldsymbol{\varepsilon}_s(x) dx \\ &= \underbrace{\int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \mathbf{B}_d(x) dx}_{\bar{\mathbf{k}}_e} \cdot \bar{\mathbf{d}}_e \\ \bar{\mathbf{f}}_e &= \bar{\mathbf{k}}_e \cdot \bar{\mathbf{d}}_e\end{aligned}$$

The element stiffness matrix in local reference is thus given by

$$\bar{\mathbf{k}}_e = \int_0^{L_e} \mathbf{B}_d(x)^T \cdot \mathbf{k}_s(x) \cdot \mathbf{B}_d(x) dx \quad (1.8)$$

1.2.1.2 Coordinate system for stiffness-based 2D beam-column with Bernoulli beam theory

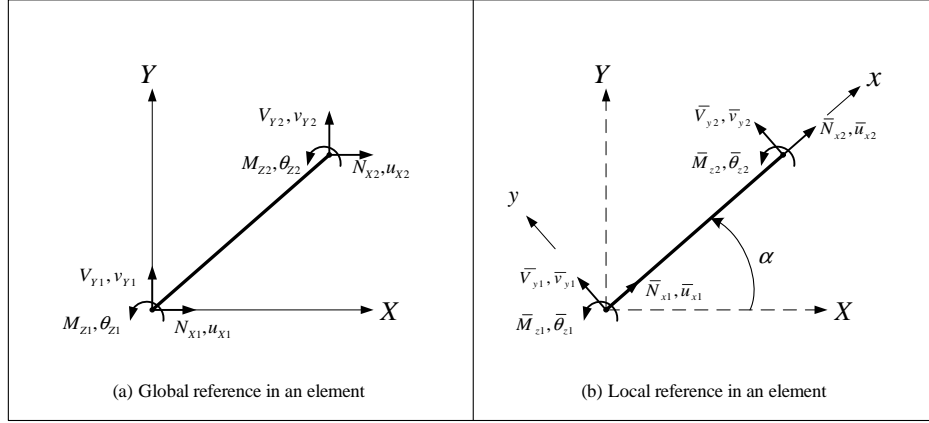


Figure 1.2: Internal forces and corresponding displacements in global and local coordinate system for the stiffness-based 2D beam-column element

The element nodal forces and displacements are expressed with respect to the global reference, Fig. 1.2(a) as

$$\mathbf{F}_e = [N_{X1}, V_{Y1}, M_{Z1}, N_{X2}, V_{Y2}, M_{Z2}]^T$$

$$\boldsymbol{\delta}_e = [u_{X1}, v_{Y1}, \theta_{Z1}, u_{X2}, v_{Y2}, \theta_{Z2}]^T$$

Again, the element nodal forces and displacements of the element can be expressed with respect to the local reference as shown in Fig. 1.2(b)

$$\bar{\mathbf{f}}_e = [\bar{N}_{x1}, \bar{V}_{y1}, \bar{M}_{z1}, \bar{N}_{x2}, \bar{V}_{y2}, \bar{M}_{z2}]^T$$

$$\bar{\mathbf{d}}_e = [\bar{u}_{x1}, \bar{v}_{y1}, \bar{\theta}_{z1}, \bar{u}_{x2}, \bar{v}_{y2}, \bar{\theta}_{z2}]^T$$

As for the truss element, the rotation matrix which transforms from global reference, (Fig. 1.2(a)) to local reference (Fig. 1.2(b)) is given by $\boldsymbol{\Gamma}_e$ such that

$$\bar{\mathbf{f}}_e = \boldsymbol{\Gamma}_e \cdot \mathbf{F}_e$$

$$\bar{\mathbf{d}}_e = \boldsymbol{\Gamma}_e \cdot \boldsymbol{\delta}_e \quad (1.9)$$

$$\mathbf{K}_e = \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{k}}_e \cdot \boldsymbol{\Gamma}_e$$

where, \mathbf{K}_e is the element stiffness matrix in global reference.

The rotation matrix is then given by

$$\boldsymbol{\Gamma}_e = \begin{bmatrix} & N_{X1} & V_{Y1} & M_{Z1} & N_{X2} & V_{Y2} & M_{Z2} \\ \bar{N}_{x1} & \cos \alpha & \sin \alpha & 0 & 0 & 0 & 0 \\ \bar{V}_{y1} & -\sin \alpha & \cos \alpha & 0 & 0 & 0 & 0 \\ \bar{M}_{z1} & 0 & 0 & 1 & 0 & 0 & 0 \\ \bar{N}_{x2} & 0 & 0 & 0 & \cos \alpha & \sin \alpha & 0 \\ \bar{V}_{y2} & 0 & 0 & 0 & -\sin \alpha & \cos \alpha & 0 \\ \bar{M}_{z2} & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

1.2.1.3 State determination

It is important to note that we do operate at three different levels in the structural analysis: a) structure level, b) element level, and c) section level. Fig. 1.3 shows the interaction of those three levels, and accompanying state

determinations.

In an incremental nonlinear analysis, to each increment of force corresponds an increment of displacement from which the total displacement is then determined. Those nodal displacements are in turn used to determine internal section forces.

State determination is the process of determining internal forces and tangent stiffness matrix corresponding to element nodal displacements. It is composed of three parts, Fig. 1.3:

1. **Structure** state determination, Fig. 1.3(a). The element tangent stiffness matrices and internal element force vector of all the elements are assembled to form the (augmented) tangent stiffness matrix \mathbf{K}_S^{tan} and internal nodal force vector \mathbf{P}_S^{int} ($\mathbf{P}_S^{int} = \mathbf{P}_t^{int} + \mathbf{P}_u^{int}$) of the structure. Subscript t and u refer to free and constrained degrees of freedom respectively (that is along the natural and essential boundaries).
2. **Element** state determination, Fig. 1.3(b). The element tangent stiffness matrices and internal element nodal forces of each element are determined from the internal section forces for each element which are in turn computed from section deformations
3. **Section** state determination, Fig. 1.3(c), where internal section forces are computed from section deformations which are in turn determined from element nodal displacements.

Once the structure state determination is complete, the internal nodal force vector ($\mathbf{P}_{t,n}^{int}$) is compared with the total applied external nodal force vector ($\mathbf{P}_{t,n}^{ext}$) and the difference ($\mathbf{P}_{t,n}^R$), is the residual nodal force vector which is then reapplied to the structure in an iterative solution process until convergence (equilibrium) is satisfied.

The state determination procedure is straightforward for a stiffness-based 2D beam-column element. The section deformation vectors $\boldsymbol{\varepsilon}_{s,e}(x)$ are determined from the element nodal displacement vector $\bar{\mathbf{d}}_e$ as shown in Eq. 1.4. The corresponding section tangent stiffness matrices $\mathbf{k}_{s,e}^{tan}(x)$ and the internal section force vectors $\boldsymbol{\sigma}_{s,e}^{int}(x)$ are determined from the section constitutive law, Eq. 1.5. The element tangent stiffness matrices $\bar{\mathbf{k}}_e^{tan}$ are obtained from Eq. 1.8, while the internal element nodal force vectors $\bar{\mathbf{f}}_e^{int}$ are determined from the principle of virtual work

$$\bar{\mathbf{f}}_e^{int} = \int_0^{L_e} \mathbf{B}_{d,e}(x)^T \cdot \boldsymbol{\sigma}_{s,e}^{int}(x) dx \quad (1.10)$$

It is important to note that this method leads to an erroneous element response in the nonlinear case. This problem is again illustrated by Fig. 1.3 which shows the evolution of the structure, element and section states during one incremental external force vector at free degrees of freedom $\Delta\mathbf{P}_{t,n}$ that requires several iterations k , (?). Subscript n refers to external force step, superscript k the k^{th} iteration within the external force step n .

The Newton-Raphson iteration method operates in the global coordinate (structural level) system. At each Newton-Raphson iteration, Fig. 1.4, the incremental nodal displacement vector at the structure level is determined from the nodal incremental force vector, and then the total element nodal displacement vectors (②) are determined next.

At the k^{th} Newton-Raphson iteration, the element nodal displacement vectors in local reference $\bar{\mathbf{d}}_{e,n}$ (③) of the nodal displacement vector $\mathbf{u}_{S,n}^k$ ($\mathbf{u}_{S,n}^k = \mathbf{u}_{t,n}^k + \mathbf{u}_{u,n}^k$) of degrees of freedom at the structure level are determined for each element.

Using Eq. 1.5 the section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^k(x)$ (④) for each section are computed. This is the first approximation of the element state determination, since $\mathbf{B}_{d,e}(x)$ is exact only in the linear elastic case of a prismatic member.

Assuming that the section constitutive law is explicitly known, the section tangent stiffness matrices $\mathbf{k}_{s,e,n}^{tan,k}(x)$ and the internal section force vectors $\boldsymbol{\sigma}_{s,e,n}^{int,k}(x)$ are readily determined from $\boldsymbol{\varepsilon}_{s,e,n}^k(x)$ (⑤). Then, using Eq. 1.8 and Eq. 1.10 the element stiffness matrices $\bar{\mathbf{k}}_{e,n}^k$ in local reference and the internal element nodal force vectors in local reference $\bar{\mathbf{f}}_{e,n}^{int,k}$ are determined next (⑥).

Finally, through assembly of the global stiffness matrix, the structures' tangent stiffness matrix and vector of nodal internal forces are determined (⑦), before the residual is computed (⑧) for convergence assessment.

Since $\mathbf{B}_{d,e}(x)$ is only approximate² (since we are approximating the displacement field), the integrals for the element tangent stiffness matrix in local reference and the internal element nodal force vector in local reference will also yield approximate results. The approximation of $\mathbf{B}_{d,e}(x)$ leads to stiffer solution, Fig. 1.3 (a) and (b). Note that the curve labeled "Exact solution" is only exact within the assumptions of the section constitutive law and the kinematic approximations that deformations are small and plane sections remain plane.

To overcome numerical errors that arise from the approximation of $\mathbf{B}_{d,e}(x)$, analysts resort to fine mesh discretization of the structure, especially, in frame regions that undergo highly nonlinear behaviors, such as the member ends. Even so, numerical convergence problems persist. This is precisely why the flexibility based element (as will be shown later) yields better results.

²We shall see later that the corresponding term in the flexibility based element is not approximate.

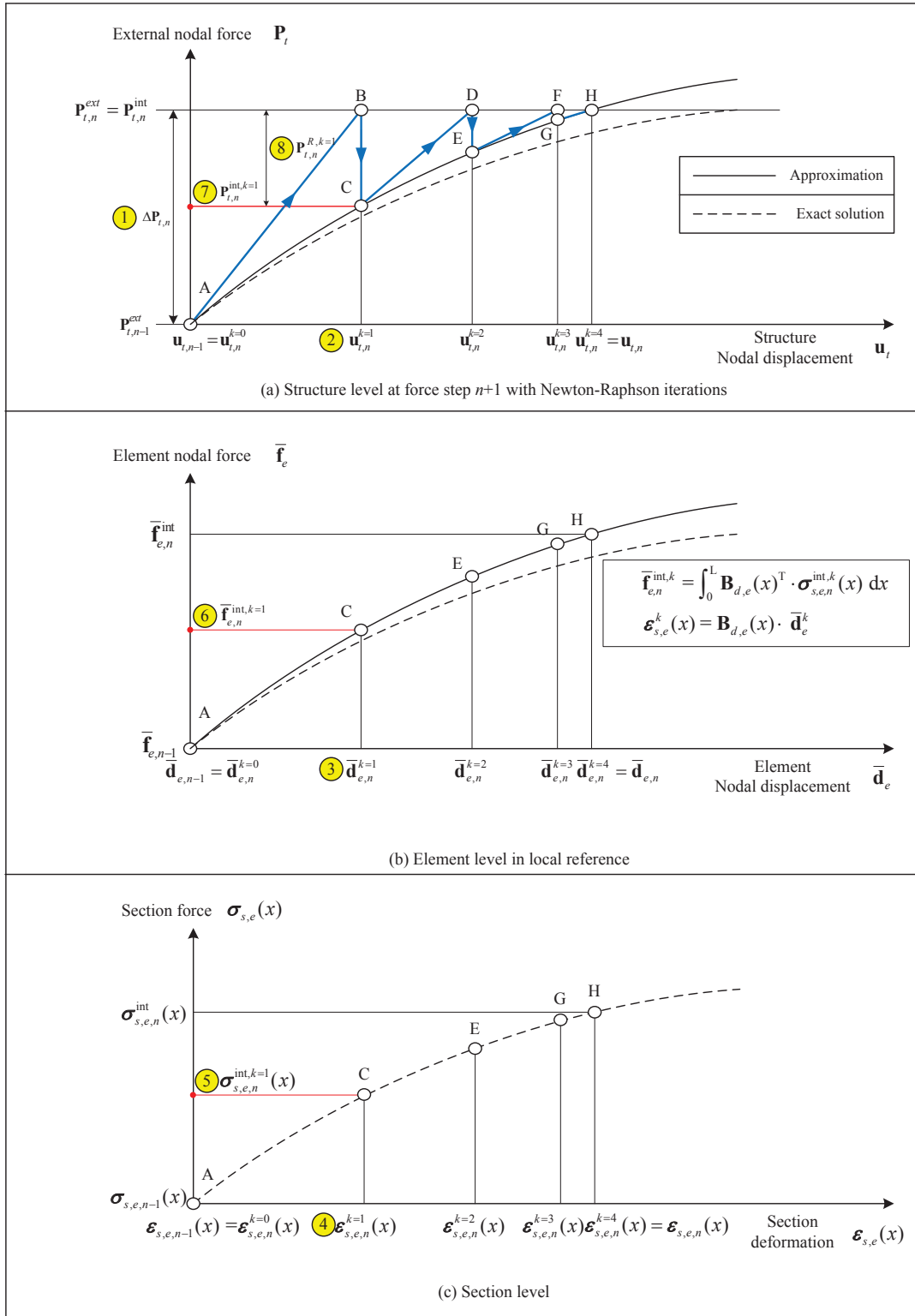
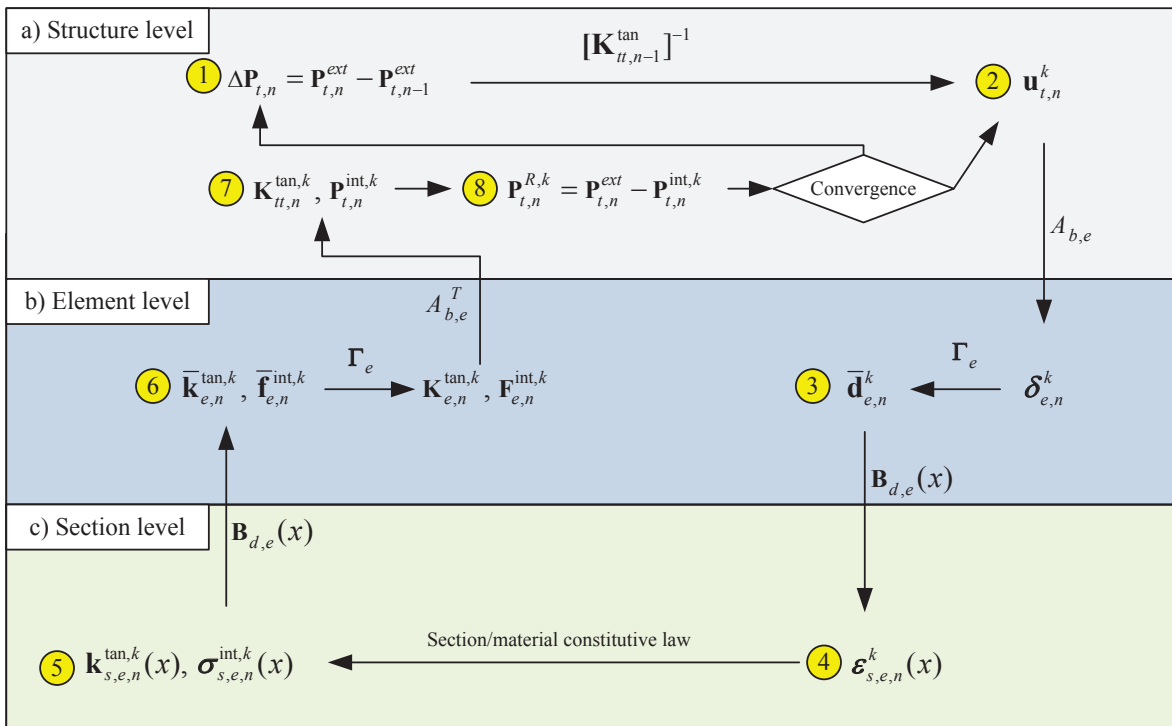
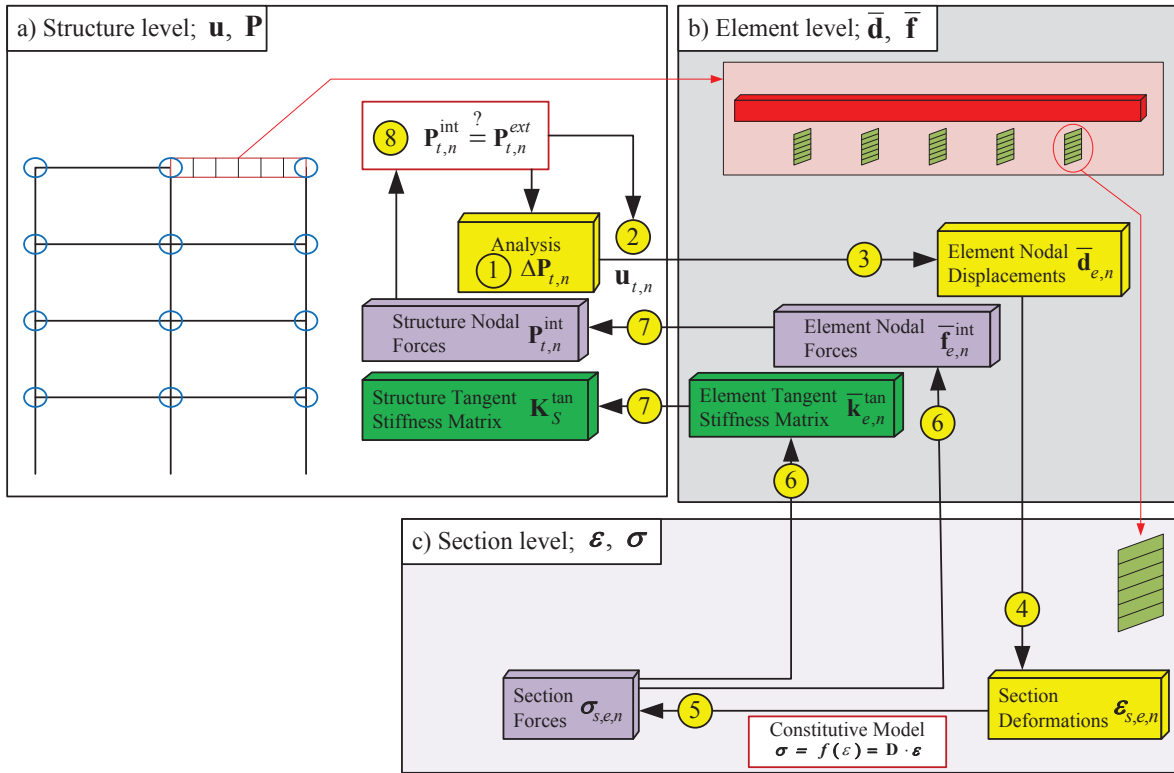


Figure 1.3: State determination for stiffness-based method



where, $\mathbf{A}_{b,e}$ and $\mathbf{A}_{b,e}^T$ are the displacement extracting operator and the force assembling operator.

Figure 1.4: State determination procedure for stiffness-based method

1.2.1.4 Nonlinear analysis using stiffness-based formulation

With reference to Fig. 1.4 to 1.7, we will examine one single step of the Newton-Raphson method for the nonlinear analysis with section constitutive law. Force and displacement control, mentioned here, are discussed in Sec. 4.4 and 4.5. Layer or fiber section procedure in Fig. 1.7 will be described in Sec. 1.3.

Again step numbers refer to those in Fig. 1.3.

- ① Compute the incremental nodal force vector $\Delta \mathbf{P}_{t,n}^{ext}$.

$$\Delta \mathbf{P}_{t,n}^{ext} = \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n-1}^{ext}$$

- ② Expressing the augmented stiffness matrix \mathbf{K}_S as

$$\begin{Bmatrix} \mathbf{P}_t \\ \mathbf{P}_u \end{Bmatrix} = \begin{bmatrix} \mathbf{K}_{tt} & \mathbf{K}_{tu} \\ \mathbf{K}_{ut} & \mathbf{K}_{uu} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_t \\ \mathbf{u}_u \end{Bmatrix} \quad (1.11)$$

where subscript t and u are associated with free end constrained degrees of freedom respectively, compute the incremental nodal displacement vector $\delta \mathbf{u}_{t,n}$ and corresponding total nodal displacement vector $\mathbf{u}_{t,n}$ in the structure level. Initially, iteration starts from Eq. 1.12 with $k = 1$.

If $k = 1$,

$$\begin{aligned} \delta \mathbf{u}_{u,n}^k &= \mathbf{u}_{u,n} - \mathbf{u}_{u,n-1}, \quad \mathbf{u}_{u,n} = \mathbf{u}_{u,n-1} + \delta \mathbf{u}_{u,n}^k \\ \delta \mathbf{u}_{t,n}^k &= [\mathbf{K}_{tt}^{tan,k-1}]^{-1} \cdot [\Delta \mathbf{P}_{t,n}^{ext} - \mathbf{K}_{tu,n}^{tan,k-1} \delta \mathbf{u}_{u,n}^k] \\ \mathbf{u}_{t,n}^k &= \mathbf{u}_{t,n}^{k-1} + \delta \mathbf{u}_{t,n}^k \end{aligned} \quad (1.12)$$

If $k \neq 1$,

$$\begin{aligned} \mathbf{P}_{t,n}^{R,k} &= \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n}^{int,k} \\ \delta \mathbf{u}_{t,n}^{k+1} &= [\mathbf{K}_{tt}^{tan,k}]^{-1} \cdot \mathbf{P}_{t,n}^{R,k} \\ \mathbf{u}_{t,n}^{k+1} &= \mathbf{u}_{t,n-1} + \Delta \mathbf{u}_{t,n}^{k+1} = \mathbf{u}_{t,n}^k + \delta \mathbf{u}_{t,n}^{k+1} \end{aligned} \quad (1.13)$$

where, superscript k is the iteration counter, \mathbf{P}_t^R the residual nodal force vector in structural level, $\Delta \mathbf{u}_{t,n}^{k+1}$ the total incremental displacement vector from the last converged step, and $\delta \mathbf{u}_{t,n}^{k+1}$ the last incremental displacement vector.

- ③ Loop over all the elements and determine their state.

- Determine the element nodal displacement vector in global reference.

If $k = 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k + \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{u,n}^k$$

If $k \neq 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k$$

where, $\mathcal{A}_{b,e}$ is a displacement extracting operator.

$$\boldsymbol{\delta}_{e,n}^k = \boldsymbol{\delta}_{e,n}^{k-1} + \delta \boldsymbol{\delta}_{e,n}^k$$

and, $\boldsymbol{\delta}_{e,n}^0$ corresponds to $\boldsymbol{\delta}_{e,n-1}$.

- Determine the element nodal displacement vector in local reference

$$\begin{aligned} \delta \bar{\mathbf{d}}_{e,n}^k &= \boldsymbol{\Gamma}_e \cdot \delta \boldsymbol{\delta}_{e,n}^k \\ \bar{\mathbf{d}}_{e,n}^k &= \bar{\mathbf{d}}_{e,n}^{k-1} + \delta \bar{\mathbf{d}}_{e,n}^k \end{aligned}$$

and, $\bar{\mathbf{d}}_{e,n}^0$ corresponds to $\bar{\mathbf{d}}_{e,n-1}$.

- ④ Start the section state determination by looping over the element's sections. The total number of section may vary from element to element. Therefore, the total number of sections in an element is $nI p_e$ and it depends on the number of integration points in the Gauss Legendre quadrature rule.

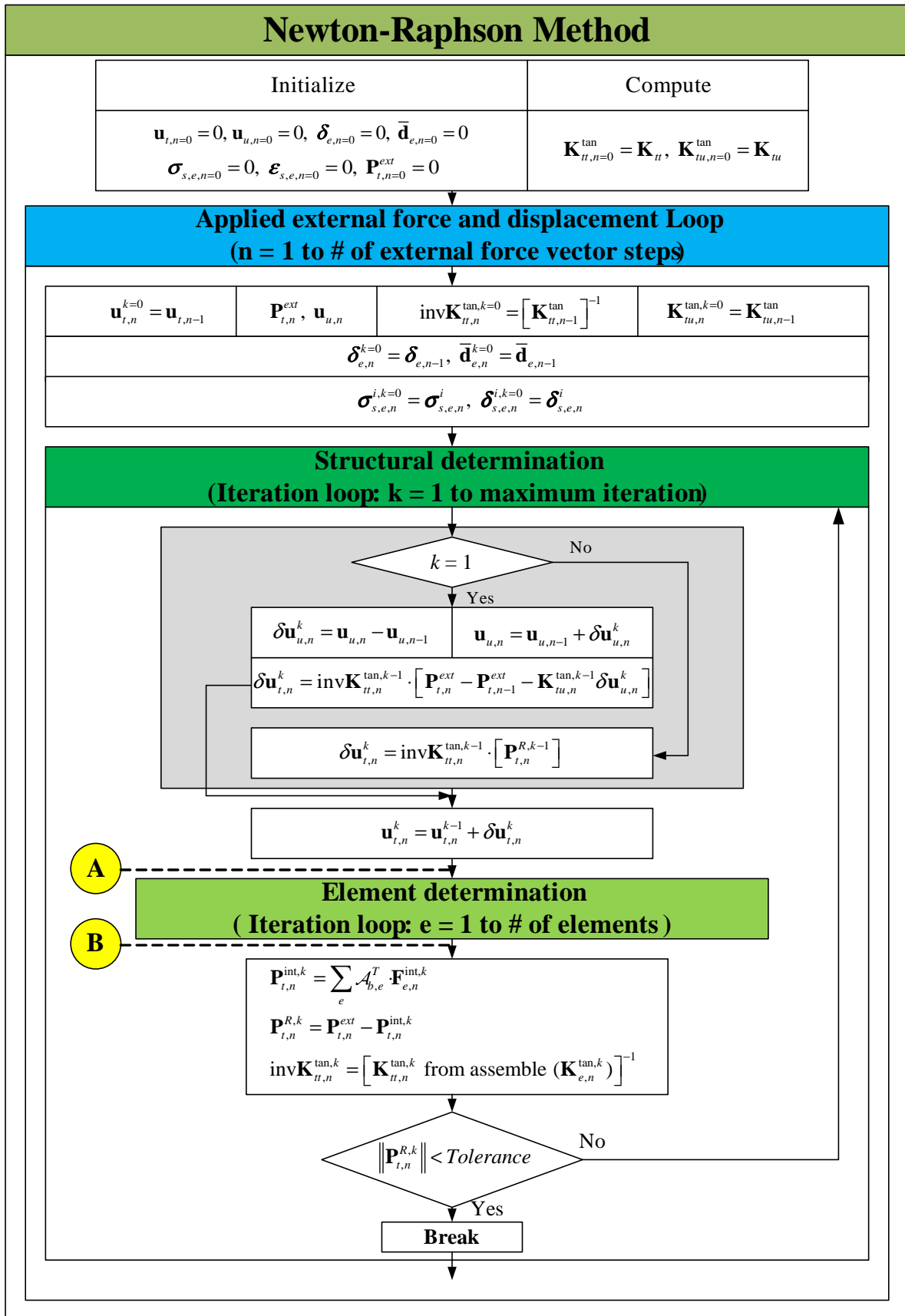


Figure 1.5: Flow chart of nonlinear analysis using stiffness-based method (1)

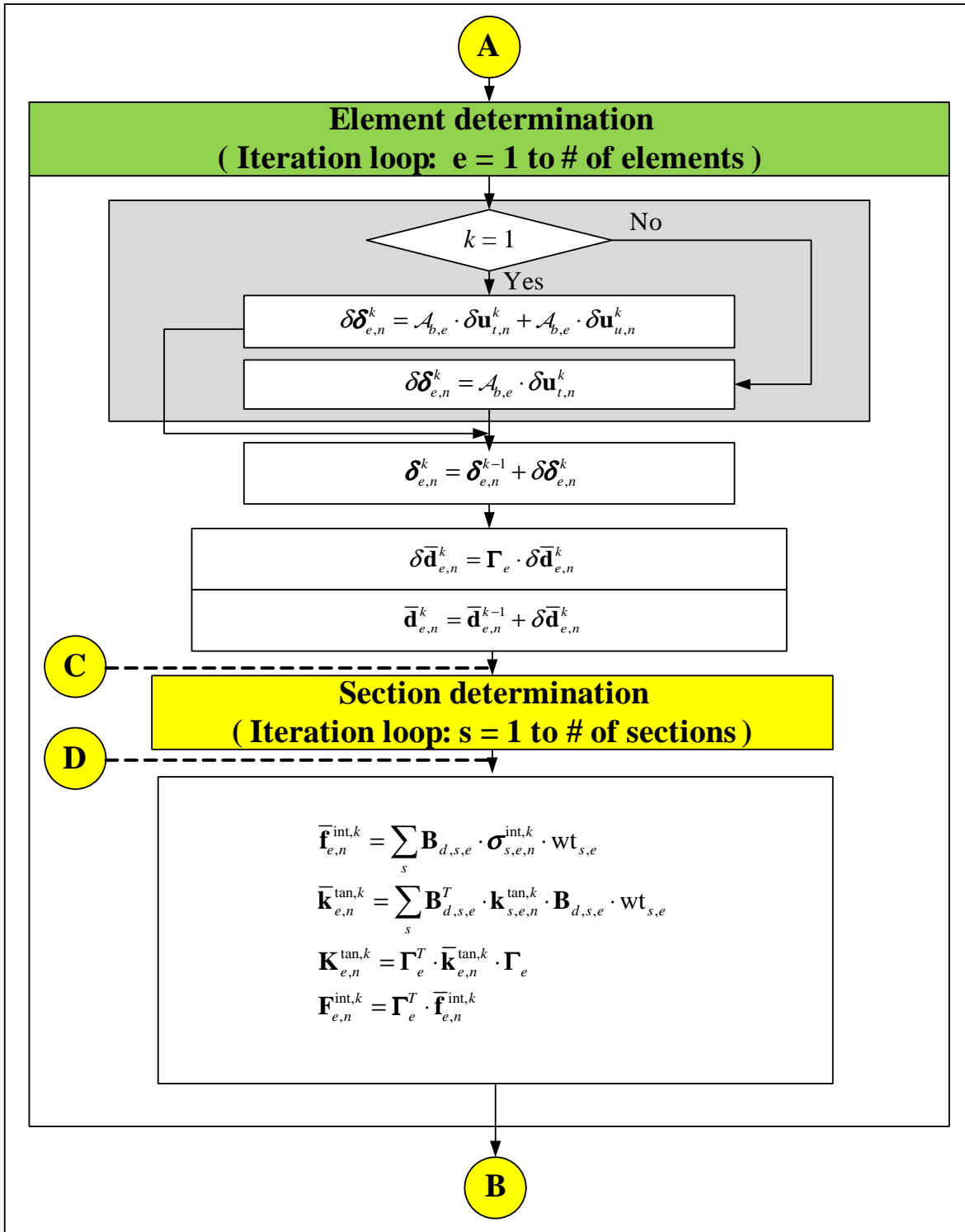


Figure 1.6: Flow chart of nonlinear analysis using stiffness-based method (2)

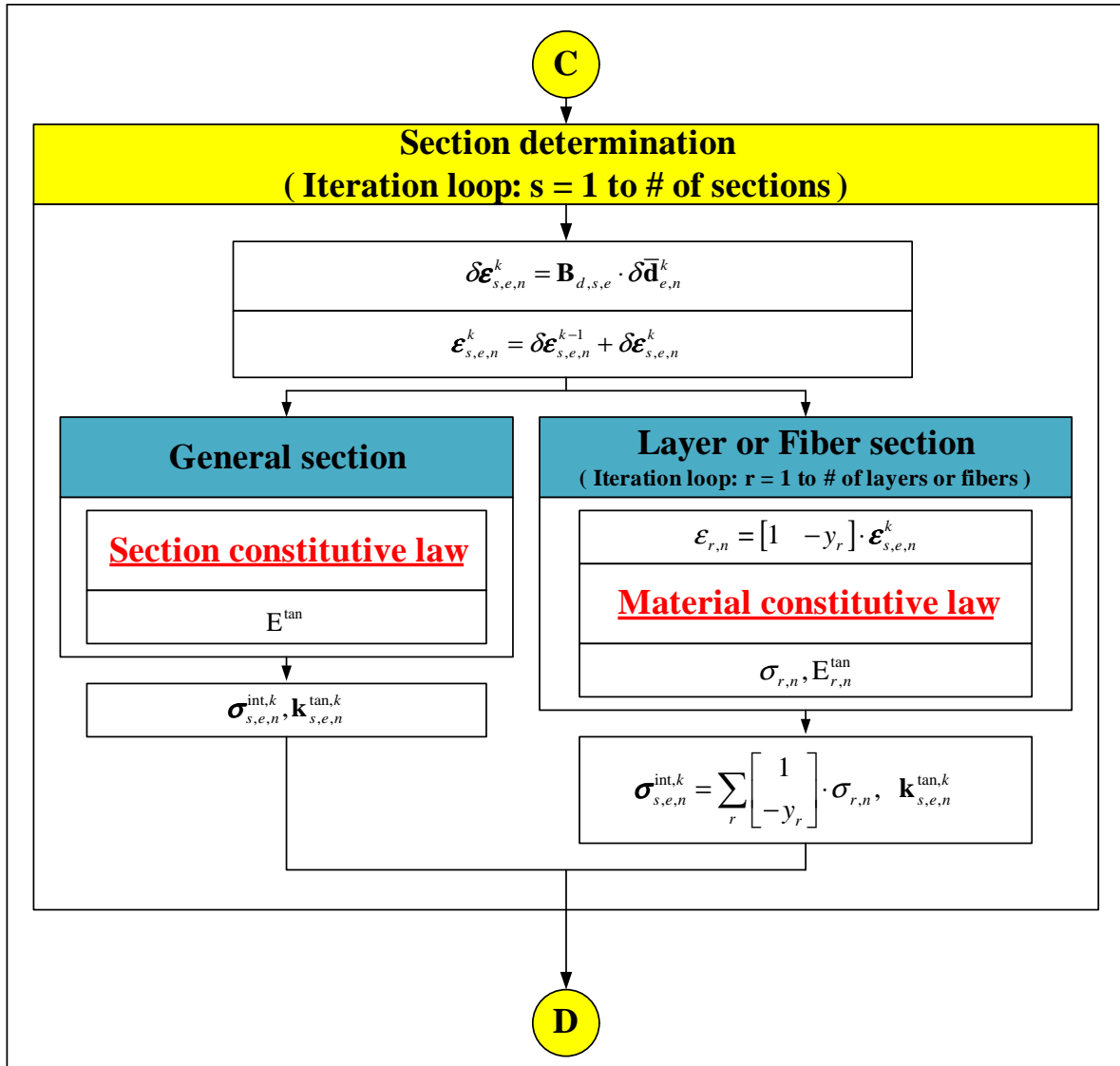


Figure 1.7: Flow chart of nonlinear analysis using stiffness-based method (3)

For the section state determination, we first need to distinguish between general section and fiber section as described later in Sec. 1.3. Presently, we only consider general section. The state determination of each section is computed within loop s which refers to the s^{th} section.

$$\delta \boldsymbol{\varepsilon}_{s,e,n}^k = \mathbf{B}_{d,s,e} \cdot \delta \bar{\mathbf{d}}_{e,n}^k$$

where, $\mathbf{B}_{d,s,e}$ is the matrix derived (derivatives) from the displacement interpolation functions at s^{th} section of e^{th} element.

$$\boldsymbol{\varepsilon}_{s,e,n}^k = \boldsymbol{\varepsilon}_{s,e,n}^{k-1} + \delta \boldsymbol{\varepsilon}_{s,e,n}^k$$

and, $\boldsymbol{\varepsilon}_{s,e,n}^0$ corresponds to $\boldsymbol{\varepsilon}_{s,e,n-1}$.

- ⑤ Determine the section tangent stiffness matrix and the internal section force vector. If the section constitutive law is explicitly known, then $\mathbf{k}_{s,e,n}^{tan,k}$ and $\boldsymbol{\sigma}_{s,e,n}^{int,k}$ are determined from $\boldsymbol{\varepsilon}_{s,e,n}^k$. For linear elastic section, we need not to recompute $\mathbf{k}_{s,e,n}^{tan,k}$ as it is identical to the initial section stiffness matrix $\mathbf{k}_{s,e}$.

Thus, for elastic sections:

$$\begin{aligned} \mathbf{k}_{s,e,n}^{tan,k} &= \mathbf{k}_{s,e} \\ \boldsymbol{\sigma}_{s,e,n}^{int,k} &= \mathbf{k}_{s,e,n}^{tan,k} \cdot \boldsymbol{\varepsilon}_{s,e,n}^k \end{aligned}$$

where, $\mathbf{k}_{s,e,n}^{tan,k}$ is the section tangent stiffness matrix at k^{th} iteration.

- ⑥ Determine first the internal element nodal force vector and the element tangent stiffness matrix, then from Eq. 1.10 and Eq. 1.8, determine $\bar{\mathbf{f}}_{e,n}^{int,k}$ and $\bar{\mathbf{K}}_{e,n}^{tan,k}$.

$$\bar{\mathbf{f}}_{e,n}^{int,k} = \sum_s \mathbf{B}_{d,s,e}^T \cdot \boldsymbol{\sigma}_{s,e,n}^{int,k} \text{ wt}_{s,e}$$

where, $\text{wt}_{s,e}$ is the weight coefficient associated with the Jacobian at the s^{th} section of the e^{th} element.

$$\bar{\mathbf{K}}_{e,n}^{tan,k} = \sum_s \mathbf{B}_{d,s,e}^T \cdot \mathbf{k}_{s,e,n}^{tan,k} \cdot \mathbf{B}_{d,s,e} \cdot \text{wt}_{s,e}$$

where, $\bar{\mathbf{K}}_{e,n}^{tan,k}$ is the element tangent stiffness matrix in local reference.

Finally, from Eq. 1.9, determine $\mathbf{F}_{e,n}^{int,k}$ and $\mathbf{K}_{e,n}^{tan,k}$.

$$\begin{aligned} \mathbf{F}_{e,n}^{int,k} &= \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{f}}_{e,n}^{int,k} \\ \mathbf{K}_{e,n}^{tan,k} &= \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{K}}_{e,n}^{tan,k} \cdot \boldsymbol{\Gamma}_e \end{aligned}$$

- ⑦ Determine the internal nodal force vector and the augmented tangent stiffness matrix.

$$\begin{aligned} \mathbf{P}_{t,n}^{int,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{F}_{e,n}^{int,k} \\ \mathbf{K}_{S,n}^{tan,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{K}_{e,n}^{tan,k} \cdot \mathcal{A}_{b,e} \end{aligned}$$

where, $\mathcal{A}_{b,e}^T$ is a force assembling operator, and $\mathbf{K}_{S,n}^{tan,k}$ encompasses the four submatrices, $\mathbf{K}_{tt,n}^{tan,k}$, $\mathbf{K}_{tu,n}^{tan,k}$, $\mathbf{K}_{ut,n}^{tan,k}$, and $\mathbf{K}_{uu,n}^{tan,k}$ as shown Eq. 1.11.

- ⑧ Compute the residual nodal force vector at the structural level from Eq. 1.13 and check for convergence:

- If $\mathbf{P}_{t,n}^{R,k}$ is within the specified tolerance, go to next force increment.
- If $\mathbf{P}_{t,n}^{R,k}$ is not within the specified tolerance, k is updated to $k+1$ and the next Newton-Raphson iteration starts. Eq. 1.13 in ② through ⑧ are repeated until convergence occurs at the structure level.

1.2.2 Flexibility-based 2D beam-column element

Flexibility-based 2D beam-column elements are “nonconformist” finite elements since they yield the element flexibility matrix rather than the classical stiffness matrix, and are based on the equations of equilibrium rather than on assumed displacement field. Nevertheless, they do offer some important advantages which will be highlighted later.

1.2.2.1 Formulation

In this section we will derive the element flexibility matrix $\tilde{\mathbf{c}}_e$ without rigid body modes and then invert it to obtain the corresponding element stiffness matrix $\tilde{\mathbf{k}}_e$ (again without rigid body modes). The retained degrees of freedom are the axial force at node 2, and the two end moments.

This is particularly interesting in those instances where stiffness-based method formulations are approximate and flexibility-based method formulations are exact such as a section varying along the element and elements with material nonlinearity.

Whereas we have used the principle of virtual work (displacement) for the derivation of the stiffness based element, we shall now use the principle of complementary virtual work (force) through the usual three steps.

Equilibrium will now be strongly enforced (whereas it was satisfied in the weak sense previously), Fig. 1.8

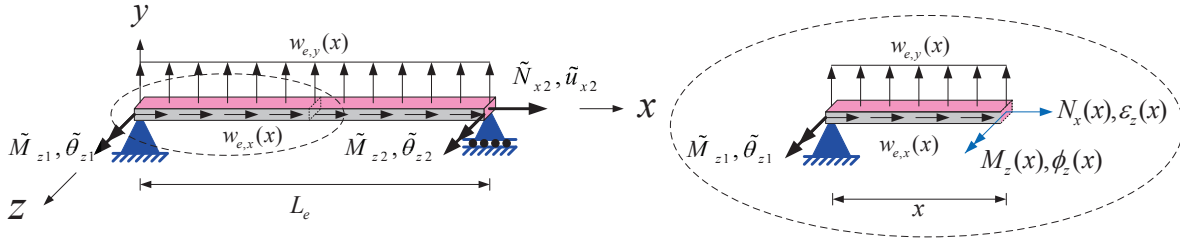


Figure 1.8: 2D beam-column element without rigid body modes

$$\underbrace{\mathbf{w}_e(x)}_{\text{External}} + \underbrace{\mathcal{L}_f \cdot \boldsymbol{\sigma}_s(x)}_{\text{Internal}} = \mathbf{0} \quad (1.14)$$

$$\begin{Bmatrix} w_x^{(e)}(x) \\ w_y^{(e)}(x) \end{Bmatrix} + \begin{bmatrix} \frac{d}{dx} & 0 \\ 0 & \frac{d^2}{dx^2} \end{bmatrix} \begin{Bmatrix} N_x(x) \\ M_z(x) \end{Bmatrix} = \mathbf{0}$$

where, $\mathbf{w}_e(x)$ is the external element traction vector and \mathcal{L}_f is the force differential operator which enforces equilibrium.

By analogy, in the stiffness formulation, the compatibility was “strongly” enforced, and it has led to the interpolation functions for displacements (also known as shape functions in that particular case).

In here, we will be expressing the equilibrium of sectional stresses in terms of the nodal forces through equilibrium. We first assume that there are no external element traction³, that is $\mathbf{w}_e(x) = \mathbf{0}$. Whereas we previously used displacement interpolation functions, we now need force interpolation functions, $\mathbf{N}_f(x)$ in order to exactly satisfy equilibrium (Eq. 1.14) along the element

$$\begin{bmatrix} \frac{d}{dx} & 0 \\ 0 & \frac{d^2}{dx^2} \end{bmatrix} \begin{Bmatrix} N_x(x) \\ M_z(x) \end{Bmatrix} = \mathbf{0}$$

will yield

$$\begin{aligned} \frac{dN_x(x)}{dx} &= 0 \\ \frac{d^2 M_z(x)}{dx^2} &= 0 \end{aligned}$$

Integrating these equations, we obtain

$$\begin{aligned} N_x(x) &= c_3 \\ M_z(x) &= c_1 x + c_2 \end{aligned} \quad (1.15)$$

Applying the natural boundary condition from Fig. 1.8 with the sign convention (on section force) shown in Fig. 1.9,

³This restriction will be later removed.

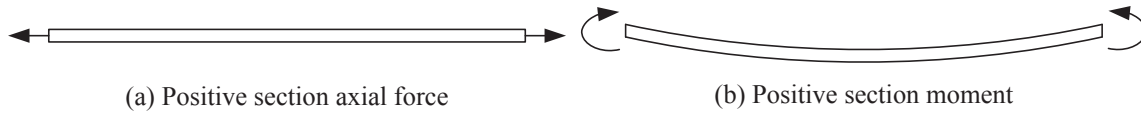


Figure 1.9: Sign convention on section forces

$$\begin{aligned} N_x(L) &= N_{x2} \\ M_z(0) &= -M_{z1} \\ M_z(L) &= M_{z2} \end{aligned}$$

we obtain

$$\begin{aligned} c_1 &= \frac{M_{z1} + M_{z2}}{L_e} \\ c_2 &= -M_{z1} \\ c_3 &= N_{x2} \end{aligned} \tag{1.16}$$

Substituting Eq. 1.16 into Eq. 1.15,

$$\begin{aligned} N_x(x) &= N_{x2} \\ M_z(x) &= \left(\frac{x}{L_e} - 1 \right) M_{z1} + \frac{x}{L_e} M_{z2} \end{aligned}$$

or

$$\underbrace{\begin{Bmatrix} N_x(x) \\ M_z(x) \end{Bmatrix}}_{\boldsymbol{\sigma}_s(x)} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ \frac{x}{L_e} - 1 & \frac{x}{L_e} & 0 \end{bmatrix}}_{\mathbf{N}_f(x)} \underbrace{\begin{Bmatrix} M_{z1} \\ M_{z2} \\ N_{x2} \end{Bmatrix}}_{\tilde{\mathbf{f}}_e} \tag{1.17}$$

where, $\tilde{\mathbf{f}}_e$ is the element nodal force vector without rigid body modes.

Finally, from Eq. 1.17, we extract the matrix of force interpolation functions.

$$\mathbf{N}_f(x) = \begin{bmatrix} 0 & 0 & 1 \\ \frac{x}{L_e} - 1 & \frac{x}{L_e} & 0 \end{bmatrix}$$

It should be noted that these shape functions enforce equilibrium at any section along the element.

Constitutive law : Whereas we have previously expressed section forces in terms of section deformations (Eq. 1.5), we now need to express section deformations in terms of section forces

$$\boldsymbol{\varepsilon}_s(x) = \mathbf{c}_s(x) \cdot \boldsymbol{\sigma}_s(x) \tag{1.18}$$

where, $\mathbf{c}_s(x)$ is the section flexibility matrix. If $\mathbf{c}_s(x)$ is not derived from fiber section (which will be discussed in Sec. 1.3), then for linear elastic analysis $\mathbf{c}_s(x)$ is simply.

$$\mathbf{c}_s(x) = \begin{bmatrix} \frac{1}{E(x) \cdot A(x)} & 0 \\ 0 & \frac{1}{E(x) \cdot I_z(x)} \end{bmatrix}$$

which is simply the inverse of Eq. 1.6. We note that shear deformations (but not shear forces) are absent in this classical Euler-Bernoulli formulation, whereas they are included in the so-called Timoshenko formulation.

Compatibility of displacements:

This requirement will be enforced only in a weak form through the principle of complementary virtual work (as

opposed to the principle of virtual work for the stiffness-based method in Eq. 1.7).

$$\underbrace{\delta \tilde{\mathbf{f}}_e^T \tilde{\mathbf{d}}_e}_{\text{External}} = \underbrace{\int_0^{L_e} \delta \boldsymbol{\sigma}_s(x)^T \cdot \boldsymbol{\varepsilon}_s(x) dx}_{\text{Internal}} \quad (1.19)$$

where $\tilde{\mathbf{d}}_e$ is the element nodal displacement vector without rigid body modes.

Substituting Eq. 1.17 and Eq. 1.18 into Eq. 1.19 and since the latter must hold for any arbitrary $\delta \tilde{\mathbf{f}}_e$, we obtain

$$\begin{aligned} \delta \tilde{\mathbf{f}}_e^T \tilde{\mathbf{d}}_e &= \int_0^{L_e} \delta \tilde{\mathbf{f}}_e^T \cdot \mathbf{N}_f(x)^T \cdot \mathbf{c}_s(x) \cdot \boldsymbol{\sigma}_s(x) dx \\ \tilde{\mathbf{d}}_e &= \int_0^{L_e} \mathbf{N}_f(x)^T \cdot \mathbf{c}_s(x) \cdot \boldsymbol{\sigma}_s(x) dx = \underbrace{\int_0^{L_e} \mathbf{N}_f(x)^T \cdot \mathbf{c}_s(x) \cdot \mathbf{N}_f(x) dx}_{\tilde{\mathbf{c}}_e} \tilde{\mathbf{f}}_e \\ \tilde{\mathbf{d}}_e &= \tilde{\mathbf{c}}_e \cdot \tilde{\mathbf{f}}_e \end{aligned}$$

The element flexibility matrix without rigid body modes in local reference is thus given by

$$\tilde{\mathbf{c}}_e = \int_0^{L_e} \mathbf{N}_f(x)^T \cdot \mathbf{c}_s(x) \cdot \mathbf{N}_f(x) dx \quad (1.20)$$

which is the counterpart of Eq. 1.8, and the corresponding element stiffness matrix without rigid body modes in local reference is simply

$$\tilde{\mathbf{k}}_e = [\tilde{\mathbf{c}}_e]^{-1}$$

1.2.2.2 Coordinate system for flexibility-based 2D beam-column element with Bernoulli beam theory

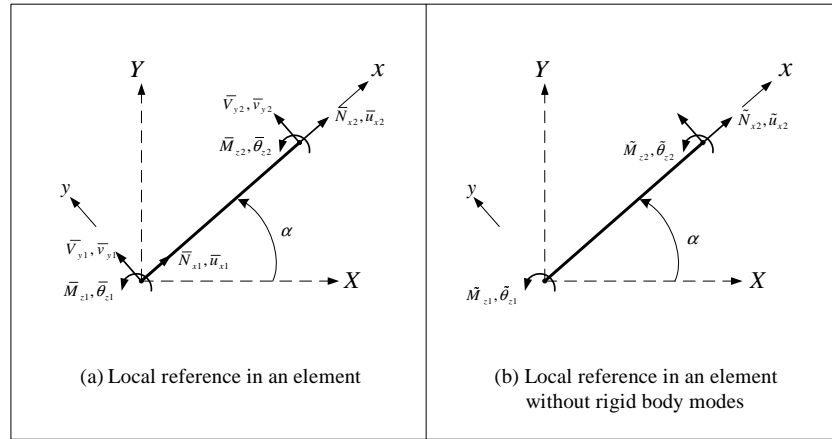


Figure 1.10: Internal forces and corresponding displacements in local coordinate system with or without rigid body for the flexibility-based 2D beam-column element

Contrarily to the reference system of the stiffness-based method in Fig. 1.2, we need to consider forces and displacements in local reference with and without rigid body modes as shown in Fig. 1.10.

Element nodal force vector without rigid body modes in local reference are (arbitrarily) selected as

$$\tilde{\mathbf{f}}_e = [\tilde{M}_{z1}, \tilde{M}_{z2}, \tilde{N}_{x2}]^T$$

and the corresponding element nodal displacement vector without rigid body modes in local reference are given by

$$\tilde{\mathbf{d}}_e = [\tilde{\theta}_{z1}, \tilde{\theta}_{z2}, \tilde{u}_{x2}]^T$$

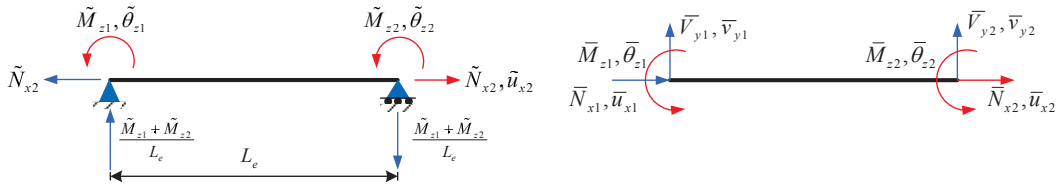


Figure 1.11: The relationship between rigid body modes and no rigid body modes

The relationship between rigid body modes and no rigid body modes is obtained through equilibrium, Fig. 1.11.

$$\underbrace{\begin{Bmatrix} \bar{N}_{x1} \\ \bar{V}_{y1} \\ \bar{M}_{z1} \\ \bar{N}_{x2} \\ \bar{V}_{y2} \\ \bar{M}_{z2} \end{Bmatrix}}_{\bar{\mathbf{f}}_e} = \underbrace{\begin{bmatrix} 0 & 0 & -1 \\ \frac{1}{L_e} & \frac{1}{L_e} & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{L_e} & -\frac{1}{L_e} & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\bar{\mathbf{\Gamma}}_e^T} \underbrace{\begin{Bmatrix} \tilde{M}_{z1} \\ \tilde{M}_{z2} \\ \tilde{N}_{x2} \end{Bmatrix}}_{\tilde{\mathbf{f}}_e} \quad (1.21)$$

$$\begin{aligned} \bar{\mathbf{f}}_e &= \bar{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{f}}_e \\ \bar{\mathbf{d}}_e &= \bar{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{d}}_e \\ \mathbf{K}_e &= \bar{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{k}}_e \cdot \bar{\mathbf{\Gamma}}_e \end{aligned} \quad (1.22)$$

which is the counterpart of Eq. 1.9.

Finally, the derivation of the stiffness matrix from the flexibility one and the equations of equilibrium parallels the one which yielded Eq. ??

$$[\mathbf{K}] = \left[\begin{array}{c|c} [\mathbf{d}]^{-1} & [\mathbf{d}]^{-1}[\mathbf{B}]^T \\ \hline [\mathbf{B}][\mathbf{d}]^{-1} & [\mathbf{B}][\mathbf{d}]^{-1}[\mathbf{B}]^T \end{array} \right] \quad (1.23)$$

1.2.2.3 State determination

In their early and pioneering publication ? did not provide a clear and consistent method for calculating the internal element force vectors from element deformations. Possibly because the finite element formulation is based on the complementary principle of virtual work and the corresponding flexibility-based 2D beam-column elements do not have shape functions that relate deformation field inside the element with element nodal displacement vector.

Since the global finite element formulation is based on the stiffness (displacement) formulation, the flexibility (force) based element formulation will have to be reconciled with the global formulation. Furthermore, internal element force vectors of all elements should be determined during the phase of state determination. Hence, the state determination is obtained from the mixed stiffness-based and flexibility-based methods.

The nonlinear algorithm for the mixed stiffness-based and flexibility-based methods will be divided into two methods (a) with Newton-Raphson iteration in the element level to determine element state, (b) without iteration in the element level to determine element state. The former is based on the formulation of (?) and the later on the one of (?).

1.2.2.3.1 With element iterations As with the state determination of stiffness-based 2D beam-column in a nonlinear structural analysis the state determination process for the mixed stiffness-based and flexibility-based methods with Newton-Raphson iteration loop in the element level is made up of three parts as shown in Fig. 1.14 (which is the counterpart of Fig. 1.3).

1. Section state determination, Fig. 1.14(c)
2. Element state determination, Fig. 1.14(b)
3. Structure state determination, Fig. 1.14(a)

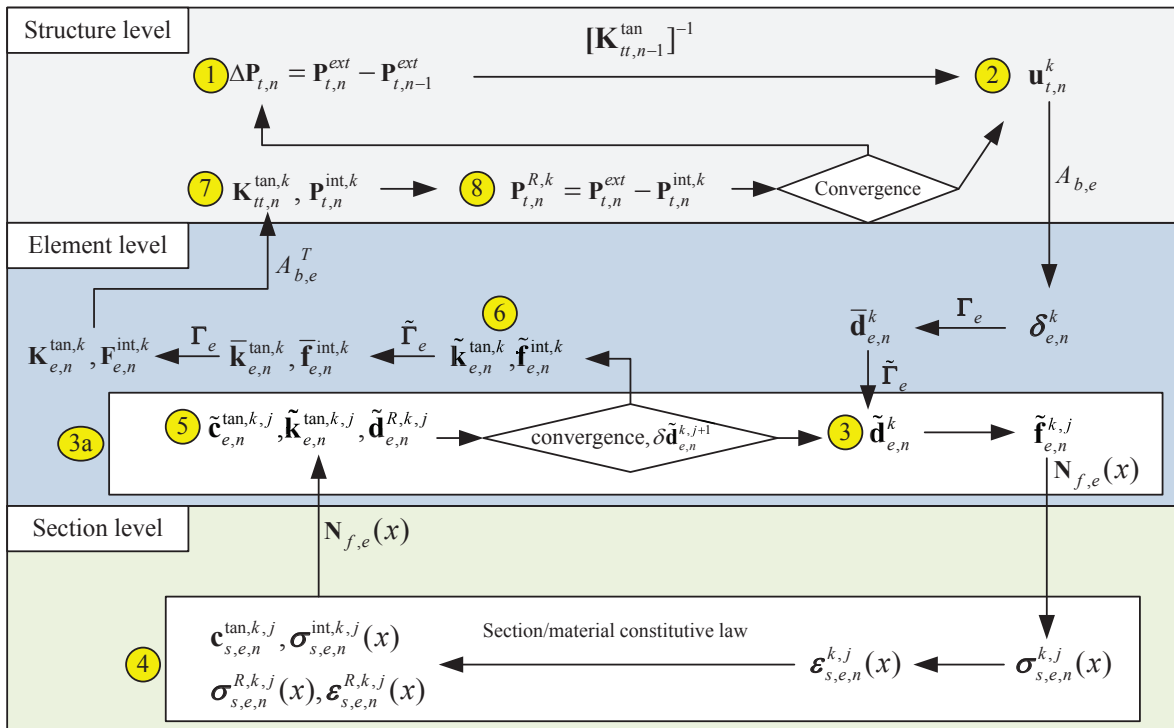
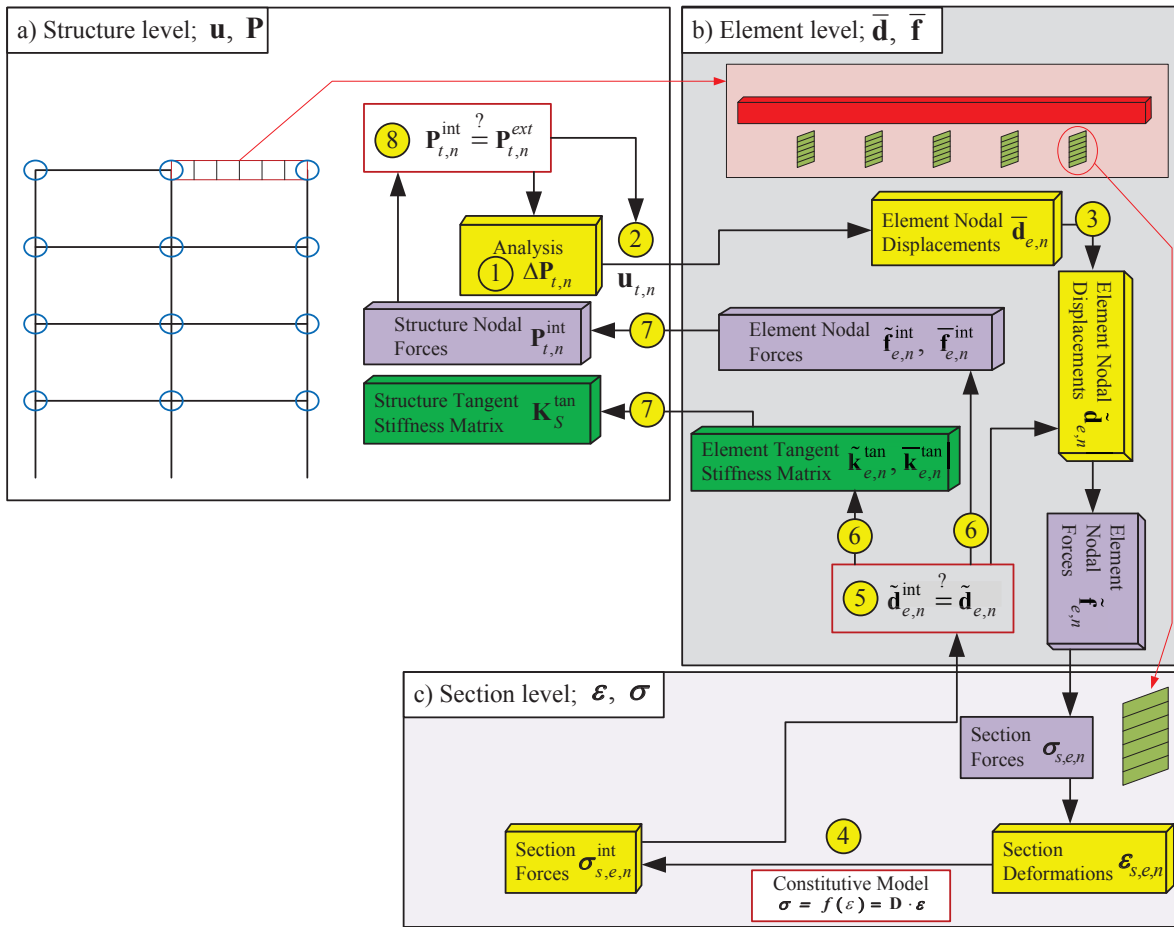


Figure 1.12: State determination procedure for flexibility-based method with Newton-Raphson iteration in the element

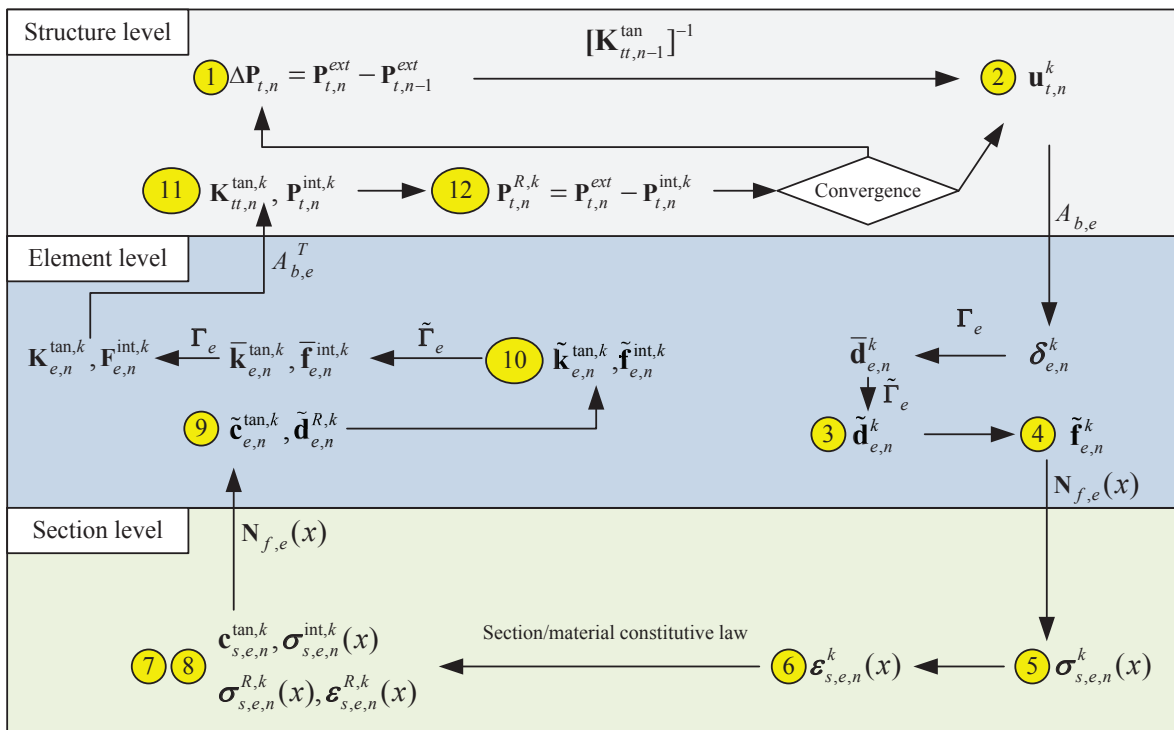
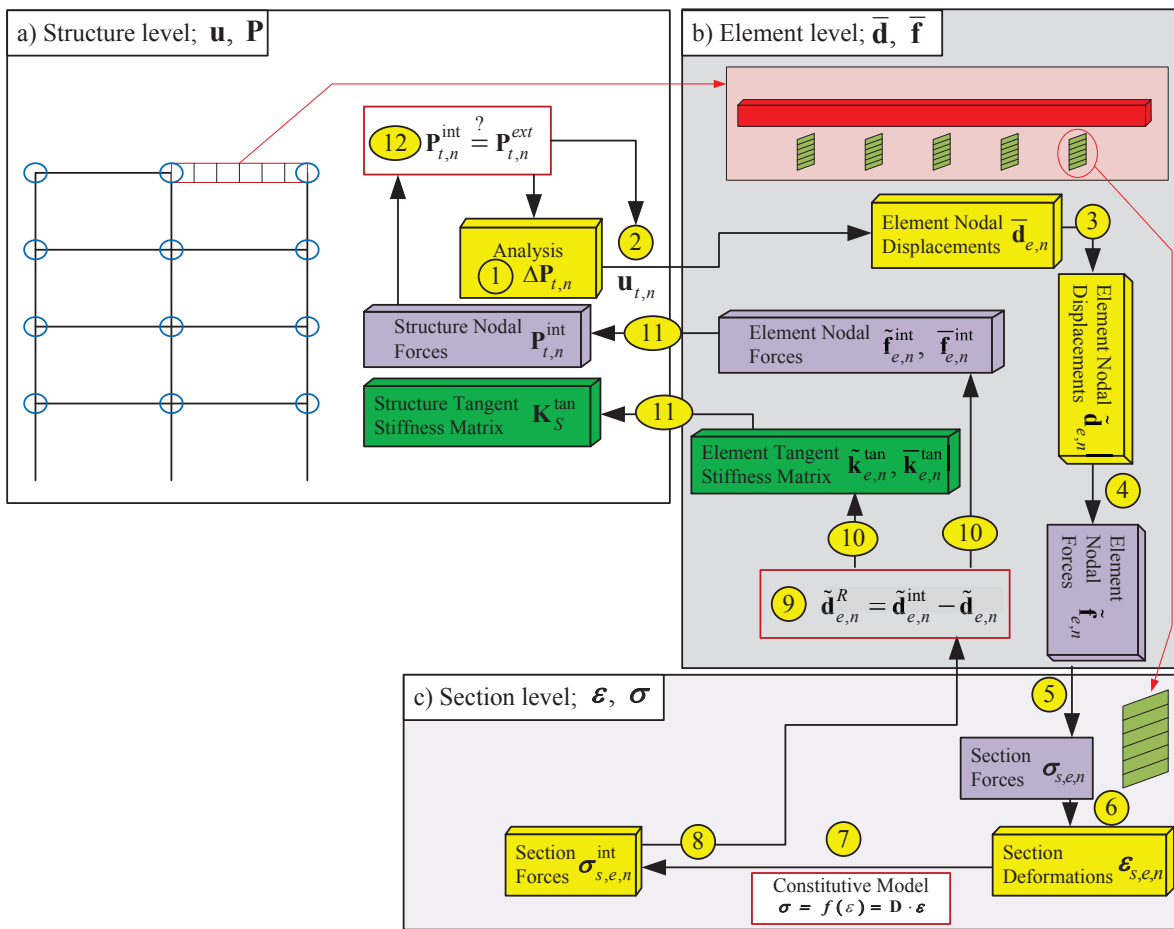


Figure 1.13: State determination procedure for flexibility-based method without Newton-Raphson iteration in the element

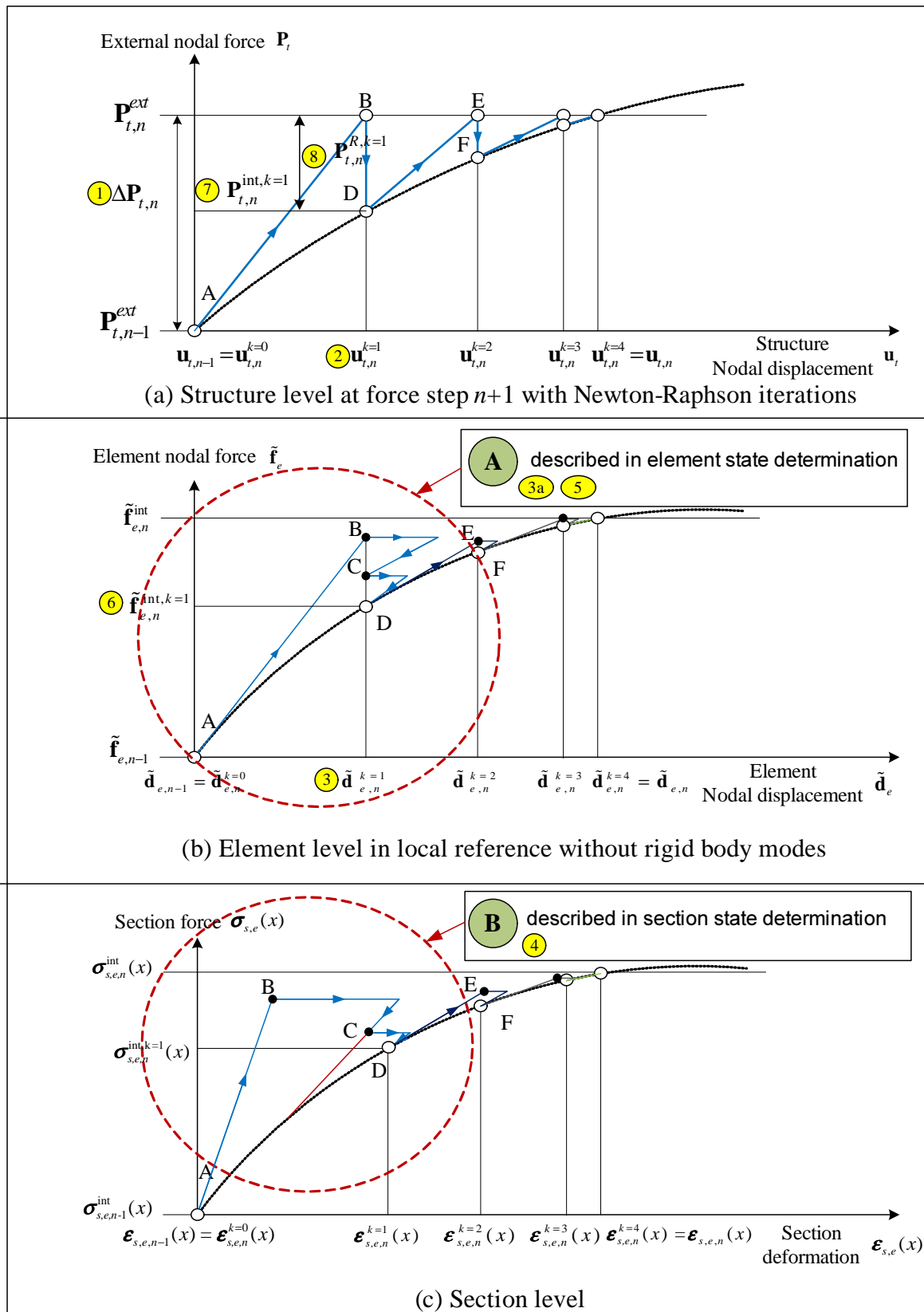
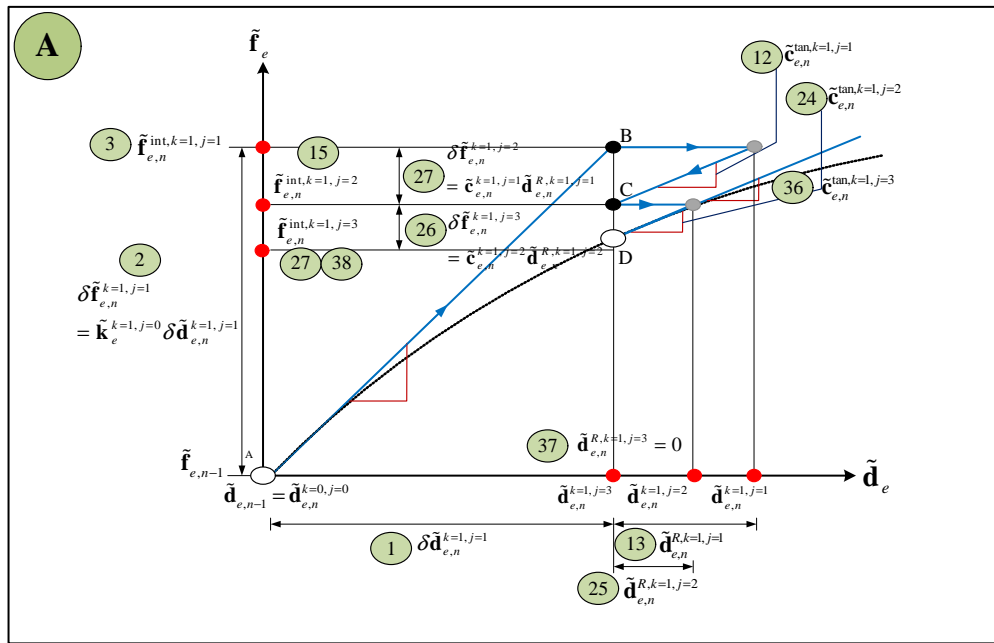
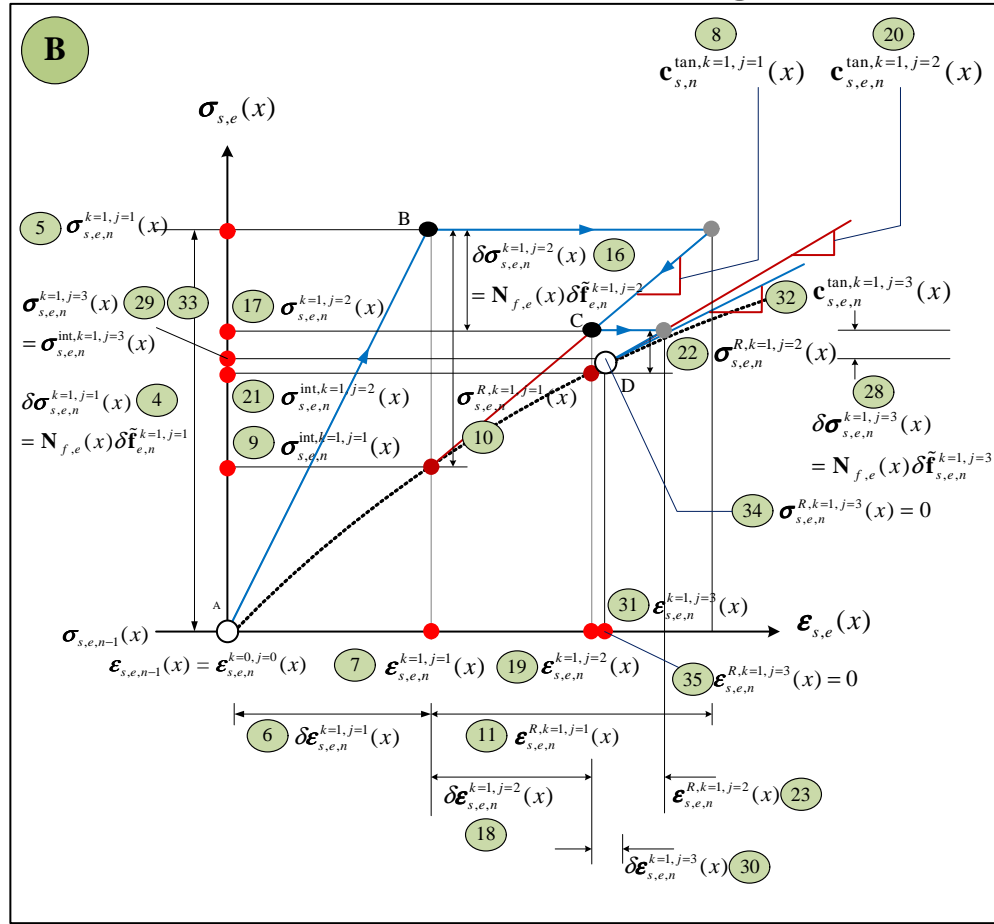


Figure 1.14: State determination for flexibility-based method with element iteration (?)



(a) Element state determination 3a 5



(b) Section state determination 4

Figure 1.15: Element and section state determinations for flexibility-based 2D beam-column element with Newton-Raphson iteration loop in the element level

Where, superscript j in Fig. 1.14(b) denotes the iteration scheme at the element level in the element state determination process. This iteration loop is necessary for the determination of the internal element nodal force vector $\tilde{\mathbf{f}}_e^{int,k}$ that correspond to the element nodal displacement vector $\tilde{\mathbf{d}}_e^k$ at the k^{th} Newton-Raphson iteration.

It is again assumed that the Newton-Raphson is used for the iteration process in the structural level and that this does not affect the iterative scheme in the element level (which determines the internal element nodal force vector $\tilde{\mathbf{f}}_e^{int}$ for the element nodal displacement vector $\tilde{\mathbf{d}}_e$).

Broadly speaking, this formulation can be decomposed as follows:

1. First, determine the element nodal force vector $\tilde{\mathbf{f}}_{e,n}^{k,j}$ (⑥) from the current element nodal displacement vector using the element tangent stiffness matrix $\tilde{\mathbf{k}}_{e,n}^{tan,k,j-1}$ (③) of the previous iteration.
2. Through the force interpolation functions $\mathbf{N}_{f,e}(x)$ determine the section force vectors $\boldsymbol{\sigma}_{s,e,n}^{k,j}(x)$ along the element. These are two complications in this procedure.
 - a) The determination of the section deformation vectors $\boldsymbol{\epsilon}_{s,e,n}^{k,j}(x)$ from section force vectors since the nonlinear section force-deformation relation is commonly expressed as an explicit function of section deformation vector (④).
 - b) Changes in the section tangent stiffness matrices $\mathbf{k}_{s,e,n}^{tan}(x)$ produce a new element tangent stiffness matrix which, in turn, changes the element nodal force vector for the given element nodal displacement vector (⑥).

These problems are solved through a nonlinear approach which first determines residual element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^{R,k,j}$ at each iteration. Then, compatibility of displacement at the structural level requires that this residual element nodal displacement vector be corrected.

This is accomplished at the element level by applying corrective element nodal force vector based on the current element tangent stiffness matrix. The corresponding section force vectors are then determined from the force interpolation functions so that equilibrium will always be satisfied along the element. These section force vectors will not change during the section state determination in order to maintain equilibrium along the element.

Finally, the linear approximation of the section force-deformation relation about the present state results in residual section deformation vectors $\boldsymbol{\sigma}_{s,e,n}^{R,k,j}(x)$. These are then integrated along the element to obtain new residual element nodal displacement vector (⑤) and the whole process is repeated until convergence occurs. It is important to stress that compatibility of element nodal displacement vector and equilibrium along the element are always satisfied in this process.

The nonlinear solution procedure for the element and section state determinations in Fig. 1.14 are illustrated in details in Fig. 1.15 for one Newton-Raphson iteration k . For illustrative purposes, convergence in loop j is reached in three iterations in Fig. 1.15.

The goal of the Newton-Raphson iteration loop in the element level is to determine the internal element nodal force vector (⑥) for the current element nodal displacement vector at the k^{th} Newton-Raphson iteration, hence

$$\tilde{\mathbf{d}}_{e,n}^k = \tilde{\mathbf{d}}_{e,n}^{k-1} + \delta\tilde{\mathbf{d}}_{e,n}^k$$

An iterative process in the element level denoted through the superscript j will be introduced inside the k^{th} Newton-Raphson iteration, and the first iteration corresponds to $j = 1$.

The initial state of the element, represented by the point \mathbf{A} , and $j = 0$ and $k = 0$ in Fig. 1.15, corresponds to the state at the end of the last convergence in structural level. With the initial element tangent stiffness matrix given by

$$\tilde{\mathbf{c}}_{e,n}^{tan,k=1,j=0} = \tilde{\mathbf{c}}_{e,n-1}^{tan}$$

and the given incremental element nodal displacement vector

$$\delta\tilde{\mathbf{d}}_{e,n}^{k=1,j=1} = \delta\tilde{\mathbf{d}}_{e,n}^{k=1}$$

hence, the corresponding incremental element nodal force vector is

$$\delta\tilde{\mathbf{f}}_{e,n}^{k=1,j=1} = \left[\tilde{\mathbf{c}}_{e,n}^{tan,k=1,j=0} \right]^{-1} \cdot \delta\tilde{\mathbf{d}}_{e,n}^{k=1,j=1} = \tilde{\mathbf{k}}_{e,n}^{tan,k=1,j=0} \cdot \delta\tilde{\mathbf{d}}_{e,n}^{k=1,j=1}$$

The incremental section force vectors can now be determined from the force interpolation functions

$$\delta\boldsymbol{\sigma}_{s,e,n}^{k=1,j=1}(x) = \mathbf{N}_{f,e}(x) \cdot \delta\tilde{\mathbf{f}}_{e,n}^{k=1,j=1}$$

With the section tangent flexibility matrices at end of the last convergence in structural level given by

$$\mathbf{c}_{s,e,n}^{tan,k=1,j=0}(x) = \mathbf{c}_{s,e,n-1}^{tan}(x)$$

the linearization of the section force-deformation relation yields the incremental section deformation vectors.

$$\delta \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=1}(x) = \mathbf{c}_{s,e,n}^{tan,k=1,j=0}(x) \cdot \delta \boldsymbol{\sigma}_{s,e,n}^{k=1,j=1}(x)$$

The section deformation vectors are updated to the state that corresponds to point **B** in Fig. 1.15(b), and the updated section deformation vector (④) will be given by

$$\boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=1}(x) = \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=0}(x) + \delta \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=1}(x)$$

For the sake of simplicity we will assume that the section force-deformation relation is explicitly known, then the section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=1}(x)$ will correspond to internal section force vectors $\boldsymbol{\sigma}_{s,e,n}^{int,k=1,j=1}(x)$ and updated section tangent flexibility matrices $\mathbf{c}_{s,e,n}^{tan,k=1,j=1}(x)$ in Fig. 1.15(b) can be defined.

The residual section force vectors are then determined

$$\boldsymbol{\sigma}_{s,e,n}^{R,k=1,j=1}(x) = \boldsymbol{\sigma}_{s,e,n}^{k=1,j=1}(x) - \boldsymbol{\sigma}_{s,e,n}^{int,k=1,j=1}(x)$$

and are transformed into residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=1}(x)$

$$\boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=1}(x) = \mathbf{c}_{s,e,n}^{tan,k=1,j=1}(x) \cdot \boldsymbol{\sigma}_{s,e,n}^{R,k=1,j=1}(x)$$

The residual section deformation vectors are thus the linear approximation of the deformation error made in the linearization of the section force-deformation relation (Fig. 1.15(b)). While any suitable section flexibility matrix can be used to calculate the residual section deformation vector, the section tangent flexibility matrices offer the fastest convergence rate.

The residual section deformation vectors are integrated along the element using the complimentary principle of virtual work to obtain the residual element nodal displacement vector (⑤).

$$\tilde{\mathbf{d}}_{e,n}^{R,k=1,j=1} = \int_0^{L_e} \mathbf{N}_{f,e}(x)^T \cdot \boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=1}(x) dx$$

At this stage the first iteration ($j = 1$) is completed. The final element and section states for $j = 1$ correspond to point **B** in Fig. 1.15. The residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=1}(x)$ and the residual element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^{R,k=1,j=1}$ were determined in the first iteration, but the corresponding element nodal displacement vector have not yet been updated. Instead, they constitute the starting point of the remaining steps within iteration loop j .

The presence of residual element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^{R,k=1,j=1}$ will violate compatibility, since elements sharing a common node would now have different element nodal displacement vector. In order to restore the inter-element compatibility, corrective force vector $\delta \tilde{\mathbf{f}}_{e,n}^{k=1,j=2}$ must be applied at the ends of the element as follows

$$\begin{aligned} \delta \tilde{\mathbf{f}}_{e,n}^{k=1,j=2} &= - \left[\tilde{\mathbf{c}}_{e,n}^{k=1,j=1} \right]^{-1} \cdot \tilde{\mathbf{d}}_{e,n}^{R,k=1,j=1} \\ \tilde{\mathbf{c}}_{e,n}^{k=1,j=1} &= \int_0^{L_e} \mathbf{N}_{f,e}(x)^T \cdot \mathbf{c}_{s,e,n}^{tan,k=1,j=1}(x) \cdot \mathbf{N}_{f,e}(x) dx \end{aligned} \quad (1.24)$$

Thus, in the second iteration ($j = 2$), the element nodal force vector (⑥) is updated as

$$\tilde{\mathbf{f}}_{e,n}^{k=1,j=2} = \tilde{\mathbf{f}}_{e,n}^{k=1,j=1} + \delta \tilde{\mathbf{f}}_{e,n}^{k=1,j=2}$$

and the section force and deformation vectors are also updated to

$$\begin{aligned} \delta \boldsymbol{\sigma}_{s,e,n}^{k=1,j=2}(x) &= \mathbf{N}_{f,e}(x) \cdot \delta \tilde{\mathbf{f}}_{e,n}^{k=1,j=2} \\ \boldsymbol{\sigma}_{s,e,n}^{k=1,j=2}(x) &= \boldsymbol{\sigma}_{s,e,n}^{k=1,j=1}(x) + \delta \boldsymbol{\sigma}_{s,e,n}^{k=1,j=2}(x) \\ \delta \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=2}(x) &= \boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=1}(x) + \mathbf{c}_{s,e,n}^{tan,k=1,j=1}(x) \cdot \delta \boldsymbol{\sigma}_{s,e,n}^{k=1,j=2}(x) \\ \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=2}(x) &= \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=1}(x) + \delta \boldsymbol{\varepsilon}_{s,e,n}^{k=1,j=2}(x) \end{aligned}$$

The state of the element and sections within the element at the end of the second iteration $j = 2$ corresponds to point **C** in Fig. 1.15. It should be noted that the updated tangent flexibility matrices $\mathbf{c}_{s,e,n}^{tan,k=1,j=2}(x)$ and residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k=1,j=2}(x)$ are computed for all sections.

The residual section deformation vectors are then integrated to obtain the residual element nodal deformation vector $\tilde{\mathbf{d}}_{e,n}^{R,k=1,j=2}$ and the new element tangent flexibility matrix $\tilde{\mathbf{c}}_{e,n}^{k=1,j=2}$ is determined by integration of the section flexibility matrices $\mathbf{c}_{s,e,n}^{tan,k=1,j=2}(x)$ according to Eq. 1.24. This completes the second iteration within loop j .

The third and subsequent iterations follow exactly the same scheme. Convergence is achieved when the selected convergence criterion is satisfied. With the conclusion of iteration loop j the internal element nodal force vector for the given element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^{k-1}$ are established, as represented by point \mathbf{D} in Fig. 1.14 and Fig. 1.15. The Newton-Raphson iteration process can now proceed with step $k + 1$.

When incremental element nodal displacement vector $\delta\tilde{\mathbf{d}}_{e,n}^{k,j=1} = \delta\tilde{\mathbf{d}}_{e,n}^k$ is added to the element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^{k-1}$ at the end of the previous Newton-Raphson iteration, it is important to make sure that the element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^k$ do not change except in the first iteration $j = 1$ during iteration loop j

Equilibrium along the element is always strictly satisfied since section force vectors (④) are derived from element nodal force vector by the force interpolation functions.

$$\boldsymbol{\sigma}_{s,e,n}^k(x) = \mathbf{N}_{f,e}(x) \cdot \tilde{\mathbf{f}}_{e,n}^k \quad \text{and} \quad \delta\boldsymbol{\sigma}_{s,e,n}^k(x) = \mathbf{N}_{f,e}(x) \cdot \delta\tilde{\mathbf{f}}_{e,n}^k$$

Compatibility is also satisfied, not only at the element ends, but also along the element.

$$\begin{aligned} \delta\tilde{\mathbf{f}}_{e,n}^{k,j} &= - \left[\tilde{\mathbf{c}}_{e,n}^{k,j-1} \right]^{-1} \cdot \tilde{\mathbf{d}}_{e,n}^{R,k,j-1} \\ \delta\boldsymbol{\sigma}_{s,e,n}^{k,j}(x) &= \mathbf{N}_{f,e}(x) \cdot \delta\tilde{\mathbf{f}}_{e,n}^{k,j} \\ \delta\boldsymbol{\varepsilon}_{s,e,n}^{k,j}(x) &= \boldsymbol{\varepsilon}_{s,e,n}^{R,k,j-1}(x) + \mathbf{c}_{s,e,n}^{tan,k,j-1}(x) \cdot \delta\boldsymbol{\sigma}_{s,e,n}^{k,j}(x) \end{aligned} \quad (1.25)$$

The second term of Eq. 1.25 expresses the relation between section deformation vectors and element nodal displacement vector. However, it should be noted that residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k,j-1}(x)$ do not strictly satisfy this compatibility condition. This requirement can only be satisfied by integrating the residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k,j-1}(x)$ to obtain $\tilde{\mathbf{d}}_{e,n}^{R,k,j-1}$. Since this is rather inefficient from a computational standpoint, the small compatibility error in the calculation of residual section deformation vectors $\boldsymbol{\varepsilon}_{s,e,n}^{R,k,j-1}(x)$ will be neglected.

While equilibrium and compatibility are satisfied along the element during each iteration of loop j , the section force-deformation relation and the element force-deformation relation is only satisfied within a specified tolerance when convergence is achieved at point in Fig. 1.15.

1.2.2.3.2 Without element iterations This alternative method, first introduced by (?) has the added advantage (over the previous approach) of avoiding an inner element loop, at the expenses however of additional iterations at the structural level.

As with the state determination of stiffness-based 2D beam-column in a nonlinear structural analysis the state determination process for the mixed stiffness-based and flexibility-based method without iteration in the element level is made up of three phases as shown in Fig. 1.16.

1. Section state determination, Fig. 1.16(c)
2. Element state determination, Fig. 1.16(b)
3. Structure state determination, Fig. 1.16(a)

Starting with completion of the structure state determination, the internal nodal force vector at k^{th} iteration $\mathbf{P}_{t,n}^{int,k}$ is compared with the total applied external nodal force vector $\mathbf{P}_{t,n}^{ext}$ and the difference, if any, yields the residual nodal force vector $\mathbf{P}_{t,n}^{R,k}$ (①) which is then reapplied to the structure in an iterative solution process until the difference of total applied external force vector and internal nodal force vector satisfies within a specified tolerance.

The state determination procedure is straightforward for a flexibility-based 2D beam-column element without element iteration like stiffness-based 2D beam-column element. The element nodal displacement vector $\tilde{\mathbf{d}}_{e,n}^k$ (③) without rigid body modes at the k^{th} iteration is determined from Eq. 1.22

$$\begin{aligned} \delta\tilde{\mathbf{d}}_{e,n}^k &= \tilde{\Gamma}_e \cdot \delta\tilde{\mathbf{f}}_{e,n}^k \\ \tilde{\mathbf{d}}_{e,n}^k &= \tilde{\mathbf{d}}_{e,n}^{k-1} + \delta\tilde{\mathbf{d}}_{e,n}^k \end{aligned}$$

and the element nodal force vector $\tilde{\mathbf{f}}_{e,n}^k$ (④) without rigid body modes is determined from Eq. 1.20

$$\begin{aligned} \delta\tilde{\mathbf{f}}_{e,n}^k &= \tilde{\mathbf{k}}_{e,n}^{tan,k-1} \cdot \delta\tilde{\mathbf{d}}_{e,n}^k \\ \tilde{\mathbf{f}}_{e,n}^k &= \tilde{\mathbf{f}}_{e,n}^{k-1} + \delta\tilde{\mathbf{f}}_{e,n}^k \end{aligned} \quad (1.26)$$

The section force vectors at the k^{th} iteration $\boldsymbol{\sigma}_{s,e,n}^k(x)$ (⑤) are determined from the element nodal force vector $\tilde{\mathbf{f}}_{e,n}^k$, and then the section deformation vectors at k^{th} iteration $\boldsymbol{\varepsilon}_{s,e,n}^k(x)$ (⑥) are determined from the section force vectors

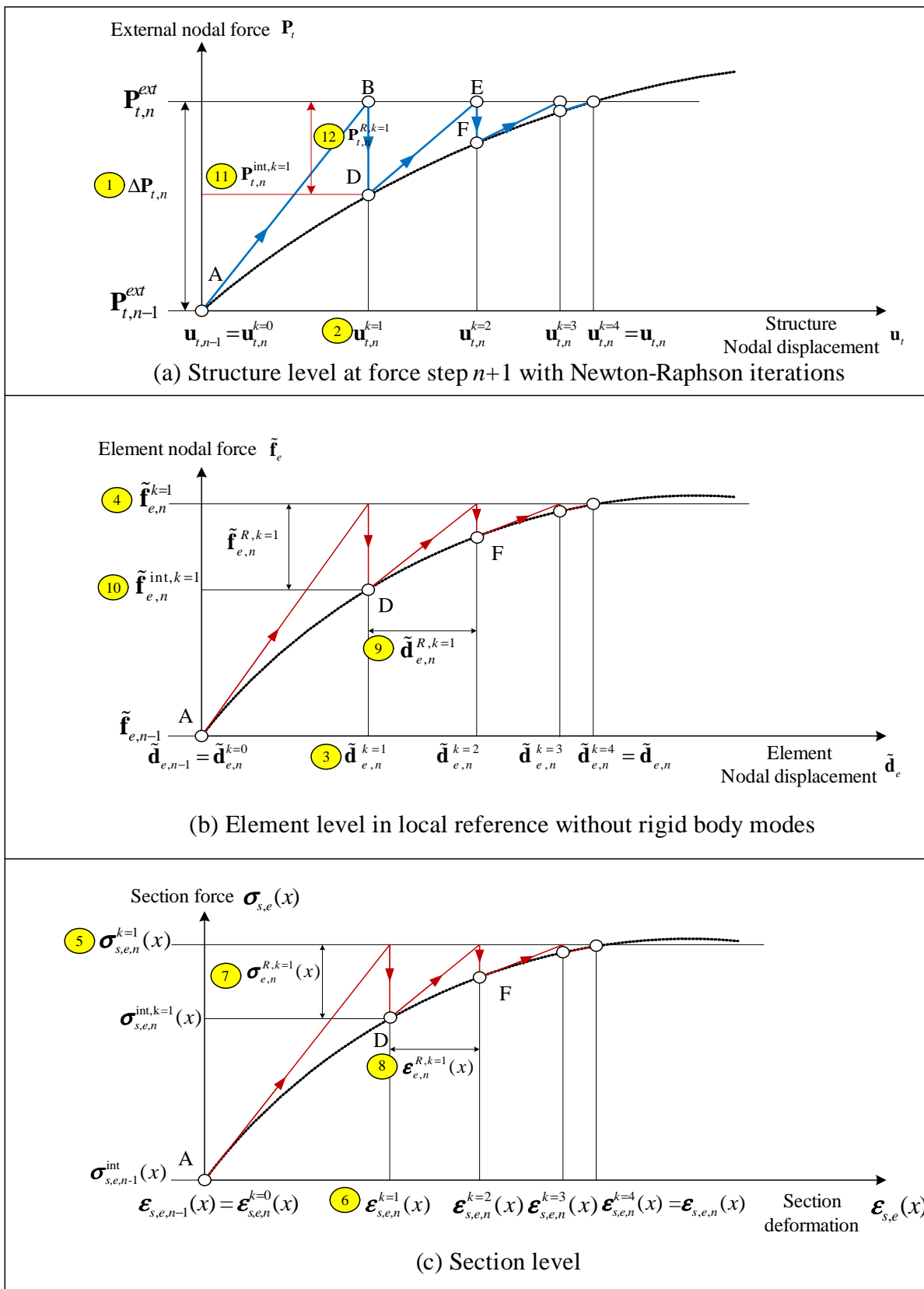


Figure 1.16: State determination for flexibility-based method without element iteration (?)

$\boldsymbol{\sigma}_{s,e,n}^k(x)$.

$$\begin{aligned}\delta\boldsymbol{\sigma}_{s,e,n}^k(x) &= \mathbf{N}_{f,e} \cdot \delta\tilde{\mathbf{f}}_{e,n}^k \\ \boldsymbol{\sigma}_{s,e,n}^k(x) &= \boldsymbol{\sigma}_{s,e,n}^{k-1}(x) + \delta\boldsymbol{\sigma}_{s,e,n}^k(x) \\ \delta\boldsymbol{\varepsilon}_{s,e,n}^k(x) &= \mathbf{c}_{s,e,n}^{tan,k-1}(x) \cdot \delta\boldsymbol{\sigma}_{s,e,n}^k(x) \\ \boldsymbol{\varepsilon}_{s,e,n}^k(x) &= \boldsymbol{\varepsilon}_{s,e,n}^{k-1}(x) + \delta\boldsymbol{\varepsilon}_{s,e,n}^k(x)\end{aligned}$$

Assuming that the section constitutive law is explicitly known, the section tangent flexibility matrices $\mathbf{c}_{s,e,n}^k(x)$ and the internal section force vectors $\boldsymbol{\sigma}_{s,e,n}^{int,k}(x)$ (⑦) are readily computed from $\boldsymbol{\varepsilon}_{s,e,n}^k(x)$ (⑧).

In this approach the residual section force and deformation vectors are first determined from the section tangent flexibility matrices $\mathbf{c}_{s,e,n}^k(x)$.

$$\begin{aligned}\boldsymbol{\sigma}_{s,e,n}^{R,k}(x) &= \boldsymbol{\sigma}_{s,e,n}^k(x) - \boldsymbol{\sigma}_{s,e,n}^{int,k}(x) \\ \boldsymbol{\varepsilon}_{s,e,n}^{R,k}(x) &= \mathbf{c}_{s,e,n}^k(x) \cdot \boldsymbol{\sigma}_{s,e,n}^{R,k}(x)\end{aligned}$$

The residual section deformation vectors are integrated along the element using the complimentary principle of virtual work to obtain the residual element nodal displacement vector (⑨).

$$\tilde{\mathbf{d}}_{e,n}^{R,k} = \int_0^{L_e} \mathbf{N}_{f,e}(x)^T \cdot \boldsymbol{\varepsilon}_{s,e,n}^{R,k}(x)$$

The element flexibility matrix without rigid body modes $\tilde{\mathbf{c}}_{e,n}^k$ is determined from Eq. 1.20, and then the residual element nodal force vector will be determined from the residual element nodal displacement vector.

$$\tilde{\mathbf{f}}_{e,n}^{R,k} = [\tilde{\mathbf{c}}_{e,n}^k]^{-1} \cdot \tilde{\mathbf{d}}_{e,n}^{R,k} \quad (1.27)$$

The internal element nodal force vector (⑩) without rigid body modes is determined from Eq. 1.26 and Eq. 1.27.

$$\tilde{\mathbf{f}}_{e,n}^{int,k} = \tilde{\mathbf{f}}_{e,n}^k - \tilde{\mathbf{f}}_{e,n}^{R,k}$$

Finally, the internal element nodal force vector is obtained from Eq. 1.22.

$$\tilde{\mathbf{f}}_{e,n}^{int,k} = \tilde{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{f}}_{e,n}^{int,k}$$

The flexibility-based 2D beam-column element without iteration in the element level to determine element state is straightforward method for the state determination procedure unlike the flexibility-based 2D beam-column element with Newton-Raphson iteration in the element level to do. This element formulation uses the force interpolation functions without any assumption unlike stiffness based element with displacement interpolation functions. Due to the exact character of the force interpolation functions, no intrinsic errors exist in the flexibility-based 2D beam-column element formulation without iteration in the element level to do. In addition, there are no limitations to the size of the elements, if the element has consistent section along longitudinal axis, for this reason (?)

1.2.2.4 Nonlinear analysis using flexibility-based method

With reference to Fig. 1.14 and 1.15, and Fig. 1.17 to Fig. 1.19 for the method with Newton-Raphson iteration loop in the element level and Fig. 1.20 to Fig. 1.22 for The method without iteration in the element level, we will examine one single step of the Newton-Raphson method in structural level for nonlinear analysis using the flexibility-based method with section constitutive law due to force and displacement control. Force and displacement control are discussed in Sec. ???. Layer/fiber section procedure in Fig. 1.7 will be described in Sec. 1.3.

1.2.2.4.1 With element iterations We first examine the nonlinear analysis of flexibility-based 2D beam-column element with element iterations with reference to Fig. 1.17 to 1.19.

Step numbers are shown in Fig. 1.14, and numbers in Fig. 1.15 indicate the procedure for element and section determination during iterations.

① Compute the incremental nodal force vector $\Delta\mathbf{P}_{i,n}^{ext}$.

$$\Delta\mathbf{P}_{i,n}^{ext} = \mathbf{P}_{i,n}^{ext} - \mathbf{P}_{i,n-1}^{ext}$$

② Compute the incremental nodal displacement vector $\delta\mathbf{u}_{i,n}$ and total nodal displacement vector $\mathbf{u}_{i,n}$ in structure level. Initially, iteration started from Eq. 1.28 and $k = 1$.

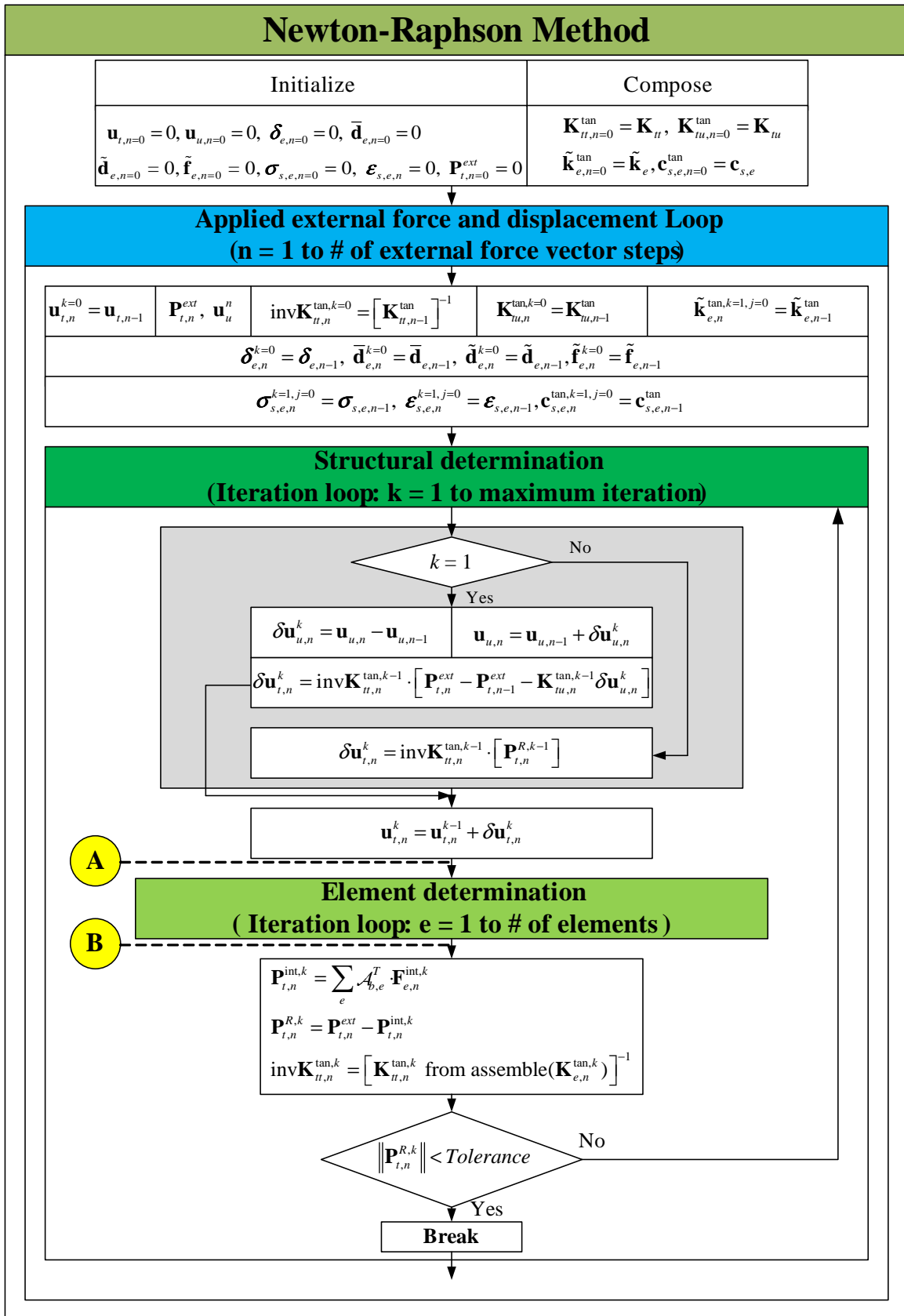


Figure 1.17: Flow chart of nonlinear analysis using flexibility-based method with Newton-Raphson iteration in the element level (1)

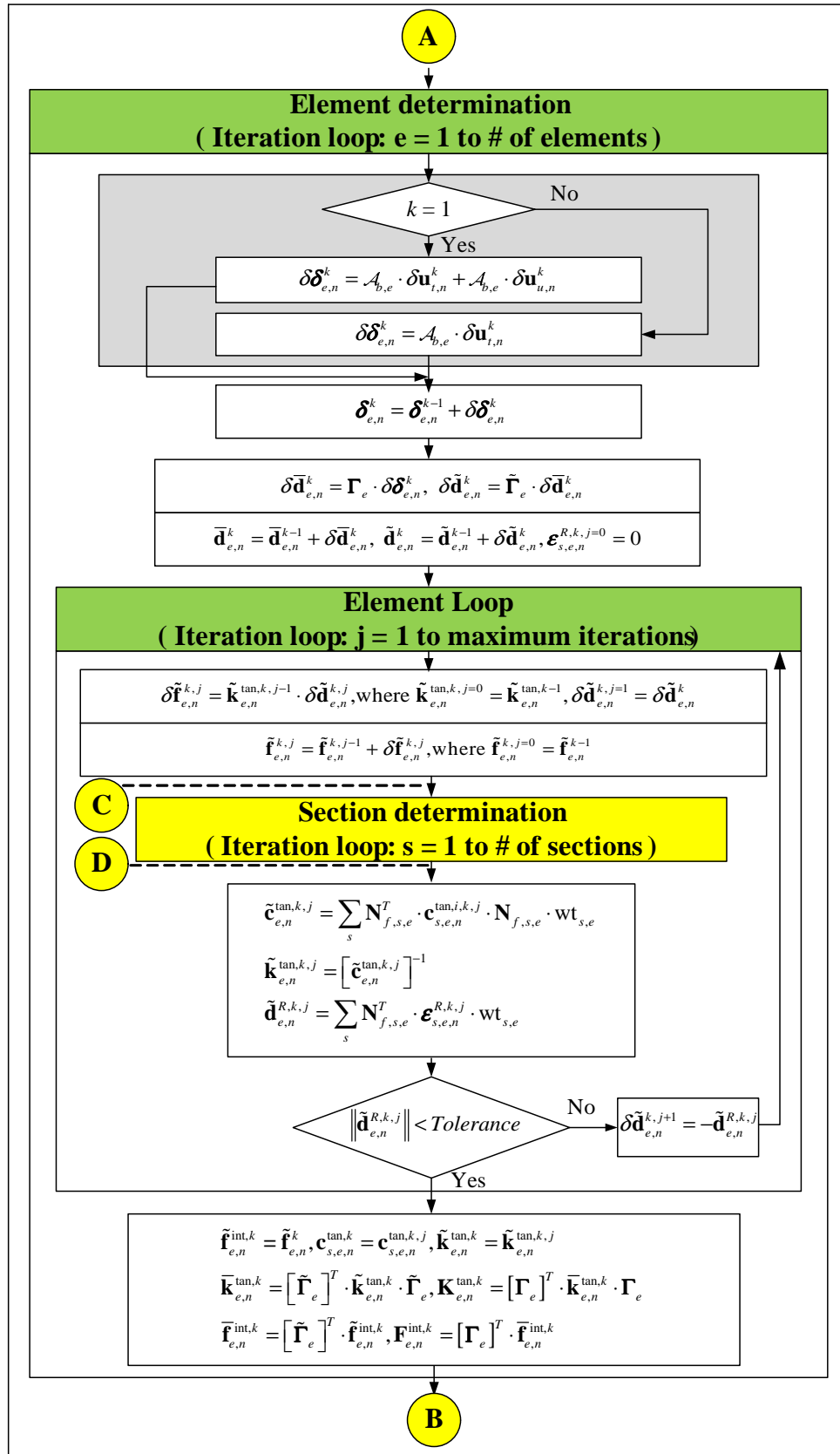


Figure 1.18: Flow chart of nonlinear analysis using flexibility-based method with Newton-Raphson iteration in the element level (2)

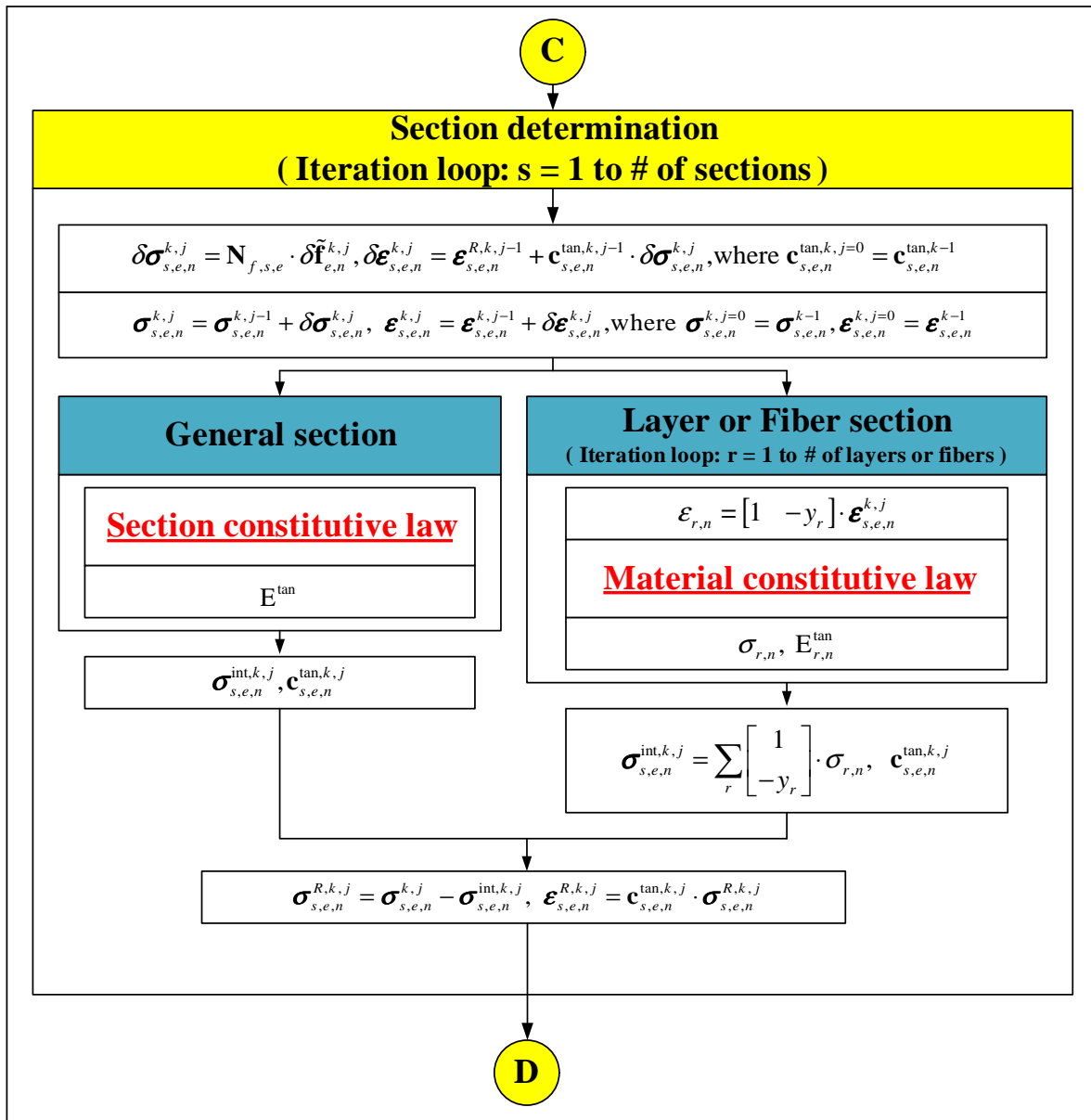


Figure 1.19: Flow chart of nonlinear analysis using flexibility-based method with Newton-Raphson iteration in the element level (3)

If $k = 1$,

$$\begin{aligned}\delta \mathbf{u}_{u,n}^k &= \mathbf{u}_{u,n} - \mathbf{u}_{u,n-1} \\ \mathbf{u}_{u,n} &= \mathbf{u}_{u,n-1} + \delta \mathbf{u}_{u,n}^k \\ \delta \mathbf{u}_{t,n}^k &= [\mathbf{K}_{tt,n}^{tan,k-1}]^{-1} \cdot [\mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n-1}^{ext} - \mathbf{K}_{tu,n}^{tan,k-1} \delta \mathbf{u}_{u,n}^k] \\ \mathbf{u}_{t,n}^k &= \mathbf{u}_{t,n}^{k-1} + \delta \mathbf{u}_{t,n}^k\end{aligned}\tag{1.28}$$

If $k \neq 1$,

$$\begin{aligned}\mathbf{P}_{t,n}^{R,k} &= \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n}^{int,k} \\ \delta \mathbf{u}_{t,n}^{k+1} &= [\mathbf{K}_{tt,n}^{tan,k}]^{-1} \cdot \mathbf{P}_{t,n}^{R,k} \\ \mathbf{u}_{t,n}^{k+1} &= \mathbf{u}_{t,n-1} + \Delta \mathbf{u}_{t,n}^{k+1} = \mathbf{u}_{t,n}^k + \delta \mathbf{u}_{t,n}^{k+1}\end{aligned}\tag{1.29}$$

where, superscript k is the iteration counter, \mathbf{P}_t^R the residual nodal force vector in structural level, $\Delta \mathbf{u}_{t,n}^{k+1}$ the total incremental displacement vector from the last converged step, and $\delta \mathbf{u}_n^{k+1}$ the last incremental displacement vector.

③ Loop over all the elements and determine their state.

- Determine the element nodal displacement vector in global reference.

If $k = 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k + \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{u,n}^k$$

If $k \neq 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k$$

where, $\mathcal{A}_{b,e}$ is displacement extracting operator.

$$\boldsymbol{\delta}_{e,n}^k = \boldsymbol{\delta}_{e,n}^{k-1} + \delta \boldsymbol{\delta}_{e,n}^k$$

and $\boldsymbol{\delta}_{e,n}^0$ corresponds to $\boldsymbol{\delta}_{e,n-1}$.

- Determine the element nodal displacement vector in local reference.

$$\begin{aligned}\delta \bar{\mathbf{d}}_{e,n}^k &= \boldsymbol{\Gamma}_e \cdot \delta \boldsymbol{\delta}_{e,n}^k \\ \bar{\mathbf{d}}_{e,n}^k &= \bar{\mathbf{d}}_{e,n}^{k-1} + \delta \bar{\mathbf{d}}_{e,n}^k\end{aligned}$$

and $\bar{\mathbf{d}}_{e,n}^0$ corresponds to $\bar{\mathbf{d}}_{e,n-1}$.

- Determine the element nodal displacement vector without rigid body modes in local reference.

$$\begin{aligned}\delta \tilde{\mathbf{d}}_{e,n}^k &= \tilde{\boldsymbol{\Gamma}}_e \cdot \delta \bar{\mathbf{d}}_{e,n}^k \\ \tilde{\mathbf{d}}_{e,n}^k &= \tilde{\mathbf{d}}_{e,n}^{k-1} + \delta \tilde{\mathbf{d}}_{e,n}^k\end{aligned}$$

and, $\tilde{\mathbf{d}}_{e,n}^0$ and $\bar{\mathbf{d}}_{e,n}^0$ correspond to $\tilde{\mathbf{d}}_{e,n-1}$ and $\bar{\mathbf{d}}_{e,n-1}$.

③a Start the element state determination. Loop over all elements in the structure. The state determination of each element is performed within loop j .

- Determine the element nodal force vector without rigid body modes

$$\begin{aligned}\delta \tilde{\mathbf{f}}_{e,n}^{k,j} &= \tilde{\mathbf{k}}_{e,n}^{tan,k,j-1} \cdot \delta \tilde{\mathbf{d}}_{e,n}^{k,j} \\ \tilde{\mathbf{f}}_{e,n}^{k,j} &= \tilde{\mathbf{f}}_{e,n}^{k,j-1} + \delta \tilde{\mathbf{f}}_{e,n}^{k,j}\end{aligned}$$

$\delta \tilde{\mathbf{d}}_{e,n}^{k,j=1}$, $\tilde{\mathbf{f}}_{e,n}^{k,j=0}$, and $\tilde{\mathbf{k}}_{e,n}^{tan,k,j=0}$ correspond to $\delta \tilde{\mathbf{d}}_e^k$, $\tilde{\mathbf{f}}_{e,n}^{k-1}$, and $\tilde{\mathbf{k}}_{e,n}^{tan,k-1}$.

④ Start the section state determination by looping over the element's sections. The total number of section may vary from element to element. Therefore, the total number of sections in an element is nIp_e and it depends on the number of integration points in the Gauss Lobatto quadrature rule.

- Determine the section force vector.

For the section state determination, we first need to differentiate between general section and layer/fiber section as described in Sec. 1.3. For the sake of this description we consider general section only. The state determination of each section is performed within loop s where the subscript corresponds to the s^{th} section.

$$\begin{aligned}\delta\sigma_{s,e,n}^{k,j} &= \mathbf{N}_{f,s,e} \cdot \delta\tilde{\mathbf{f}}_{e,n}^{k,j} \\ \sigma_{s,e,n}^{k,j} &= \sigma_{s,e,n}^{k,j-1} + \delta\sigma_{s,e,n}^{k,j}\end{aligned}$$

where, $\mathbf{N}_{f,s,e}$ is the matrix derived from the force interpolation functions at s^{th} section of e^{th} element, and $\sigma_{s,e,n}^{k,j=0}$ corresponds to $\sigma_{s,e,n}^{k-1}$.

- Determine section deformation vector.

$$\begin{aligned}\delta\epsilon_{s,e,n}^{k,j} &= \epsilon_{s,e,n}^{R,k,j-1} + \mathbf{c}_{s,e,n}^{tan,k,j-1} \cdot \delta\sigma_{s,e,n}^{k,j} \\ \epsilon_{s,e,n}^{k,j} &= \epsilon_{s,e,n}^{k,j-1} + \delta\epsilon_{s,e,n}^{k,j}\end{aligned}$$

and, $\epsilon_{s,e,n}^{k,j=0}$ and $\mathbf{c}_{s,e,n}^{tan,k,j=0}$ correspond to $\epsilon_{s,e,n}^{k-1}$ and $\mathbf{c}_{s,e,n}^{tan,k-1}$, and $\epsilon_{s,e,n}^{R,k,j=0} = 0$.

- Determine the section tangent flexibility matrix and the internal section force vector. If we assume that the section constitutive law is explicitly known, $\mathbf{c}_{s,e,n}^{tan,k,j}$ and $\sigma_{s,e,n}^{int,k,j}$ are both determined from $\epsilon_{s,e,n}^{k,j}$. However, in elastic section, we need not compute $\mathbf{c}_{s,e,n}^{tan,k,j}$ again as it is identical to the initial section flexibility matrix $\tilde{\mathbf{c}}_{s,e}$.
- Determine the residual section force and deformation vector.

$$\begin{aligned}\sigma_{s,e,n}^{R,k,j} &= \sigma_{s,e,n}^{k,j} - \sigma_{s,e,n}^{int,k,j} \\ \epsilon_{s,e,n}^{R,k,j} &= \mathbf{c}_{s,e,n}^{tan,k,j} \cdot \sigma_{s,e,n}^{R,k,j}\end{aligned}$$

- ⑤ Determine the updated element flexibility matrix and the residual element nodal displacement vector.

The residual section deformation vectors and the section tangent flexibility matrices are integrated along the element using the complimentary principle of virtual work to obtain the residual element nodal displacement vector and the element tangent flexibility matrix.

$$\begin{aligned}\tilde{\mathbf{d}}_{e,n}^{R,k,j} &= \sum_s \mathbf{N}_{f,s,e}^T \cdot \epsilon_{s,e,n}^{R,k,j} \cdot \text{wt}_{s,e} \\ \tilde{\mathbf{c}}_{e,n}^{tan,k,j} &= \sum_s \mathbf{N}_{f,s,e}^T \cdot \mathbf{c}_{s,e,n}^{tan,k,j} \cdot \mathbf{N}_{f,s,e} \cdot \text{wt}_{s,e} \\ \tilde{\mathbf{K}}_{e,n}^{tan,k,j} &= [\tilde{\mathbf{c}}_{e,n}^{tan,k,j}]^{-1}\end{aligned}$$

where, $\text{wt}_{s,e}$ is the weight coefficient associated with the Jacobian at the s^{th} section of the e^{th} element.

Convergence at the element level must be satisfied with the residual element nodal displacement vector.

- If $\tilde{\mathbf{d}}_{e,n}^{R,k,j}$ is within the specified tolerance, go to ⑥.
- If $\tilde{\mathbf{d}}_{e,n}^{R,k,j}$ is not within the specified tolerance, j and $\delta\tilde{\mathbf{d}}_{e,n}^{k,j+1}$ are updated to $j+1$ and $-\tilde{\mathbf{d}}_{e,n}^{R,k,j}$, and the next Newton-Raphson iteration initiates. We then repeat ③a through ⑤ until convergence occurs at the element level.

- ⑥ Determine the internal element nodal force vector and the element tangent stiffness matrix.

$$\begin{aligned}\tilde{\mathbf{f}}_{e,n}^{int,k} &= \tilde{\mathbf{f}}_{e,n}^{int,k,j} \\ \tilde{\mathbf{\Gamma}}_{e,n}^{int,k} &= \tilde{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{f}}_{e,n}^{int,k} \\ \mathbf{F}_{e,n}^{int,k} &= \mathbf{\Gamma}_e^T \cdot \tilde{\mathbf{f}}_{e,n}^{int,k}\end{aligned}$$

$$\begin{aligned}
\mathbf{c}_{s,e,n}^{tan,k} &= \mathbf{c}_{s,e,n}^{tan,k,j} \\
\tilde{\mathbf{k}}_{e,n}^{tan,k} &= \tilde{\mathbf{k}}_{e,n}^{tan,k,j} \\
\bar{\mathbf{k}}_{e,n}^{tan,k} &= \tilde{\Gamma}_e^T \cdot \tilde{\mathbf{k}}_{e,n}^{tan,k} \cdot \tilde{\Gamma}_e \\
\mathbf{K}_{e,n}^{tan,k} &= \Gamma_e^T \cdot \bar{\mathbf{k}}_{e,n}^{tan,k} \cdot \Gamma_e
\end{aligned}$$

⑦ Determine the internal nodal force vector and the augmented tangent stiffness matrix.

$$\begin{aligned}
\mathbf{P}_{t,n}^{int,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{F}_{e,n}^{int,k} \\
\mathbf{K}_{S,n}^{tan,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{K}_{e,n}^{tan,k} \cdot \mathcal{A}_{b,e}
\end{aligned}$$

where, $\mathcal{A}_{b,e}^T$ is force assembling operator, and $\mathbf{K}_{S,n}^{tan,k}$ is consist of $\mathbf{K}_{tt,n}^{tan,k}$, $\mathbf{K}_{tu,n}^{tan,k}$, $\mathbf{K}_{ut,n}^{tan,k}$, and $\mathbf{K}_{uu,n}^{tan,k}$.

⑧ Compute the residual nodal force vector at the structural level from Eq. 1.29. We then satisfy convergence with the residual nodal force vector.

- If $\mathbf{P}_{t,n}^{R,k}$ is within the specified tolerance, go to next force increment.
- If $\mathbf{P}_{t,n}^{R,k}$ is not within the specified tolerance, k is updated to $k + 1$ and the next Newton-Raphson iteration initiates. Eq. 1.29 in ② through ⑧ are repeated until convergence occurs at the structure level.

1.2.2.4.2 Without element iterations We then consider the methodology first proposed by (?) in which there are no element iteration with reference to Fig. 1.16 and Fig. 1.20 to 1.22.

① Compute the incremental nodal force vector $\Delta \mathbf{P}_{t,n}^{ext}$.

$$\Delta \mathbf{P}_{t,n}^{ext} = \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n-1}^{ext}$$

② Compute the incremental nodal displacement vector $\delta \mathbf{u}_{t,n}$ and total nodal displacement vector $\mathbf{u}_{t,n}$ in structure level. Initially, iteration started from Eq. 1.30 and $k = 1$.

If $k = 1$,

$$\begin{aligned}
\delta \mathbf{u}_{u,n}^k &= \mathbf{u}_{u,n} - \mathbf{u}_{u,n-1} \\
\mathbf{u}_{u,n} &= \mathbf{u}_{u,n-1} + \delta \mathbf{u}_{u,n}^k \\
\delta \mathbf{u}_{t,n}^k &= [\mathbf{K}_{tt,n}^{tan,k-1}]^{-1} \cdot [\mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n-1}^{ext} - \mathbf{K}_{tu,n}^{tan,k-1} \delta \mathbf{u}_{u,n}^k] \\
\mathbf{u}_{t,n}^k &= \mathbf{u}_{t,n}^{k-1} + \delta \mathbf{u}_{t,n}^k
\end{aligned} \tag{1.30}$$

If $k \neq 1$,

$$\begin{aligned}
\mathbf{P}_{t,n}^{R,k} &= \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n}^{int,k} \\
\delta \mathbf{u}_{t,n}^{k+1} &= [\mathbf{K}_{tt,n}^{tan,k}]^{-1} \cdot \mathbf{P}_{t,n}^{R,k} \\
\mathbf{u}_{t,n}^{k+1} &= \mathbf{u}_{t,n-1} + \Delta \mathbf{u}_{t,n}^{k+1} = \mathbf{u}_{t,n}^k + \delta \mathbf{u}_{t,n}^{k+1}
\end{aligned} \tag{1.31}$$

where, superscript k is the iteration counter, \mathbf{P}_t^R the residual nodal force vector in structural level, $\Delta \mathbf{u}_{t,n}^{k+1}$ the total incremental displacement vector from the last converged step, and $\delta \mathbf{u}_n^{k+1}$ the last incremental displacement vector.

③ Loop over all the elements and determine their state.

- Determine the element nodal displacement vector in global reference.

If $k = 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k + \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{u,n}^k$$

If $k \neq 1$,

$$\delta \boldsymbol{\delta}_{e,n}^k = \mathcal{A}_{b,e} \cdot \delta \mathbf{u}_{t,n}^k$$

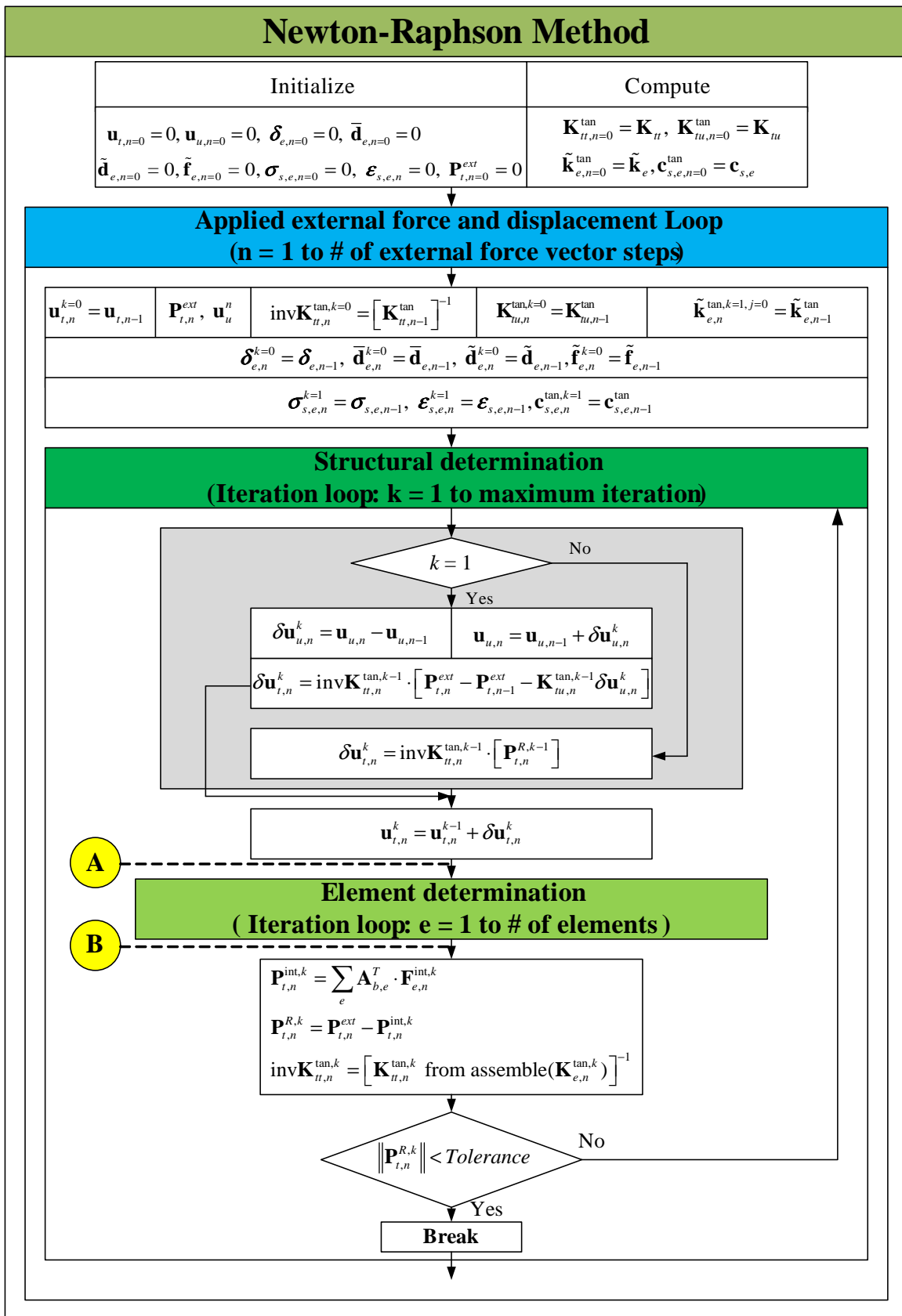


Figure 1.20: Flow chart of nonlinear analysis using flexibility-based method without iteration in the element level (1)

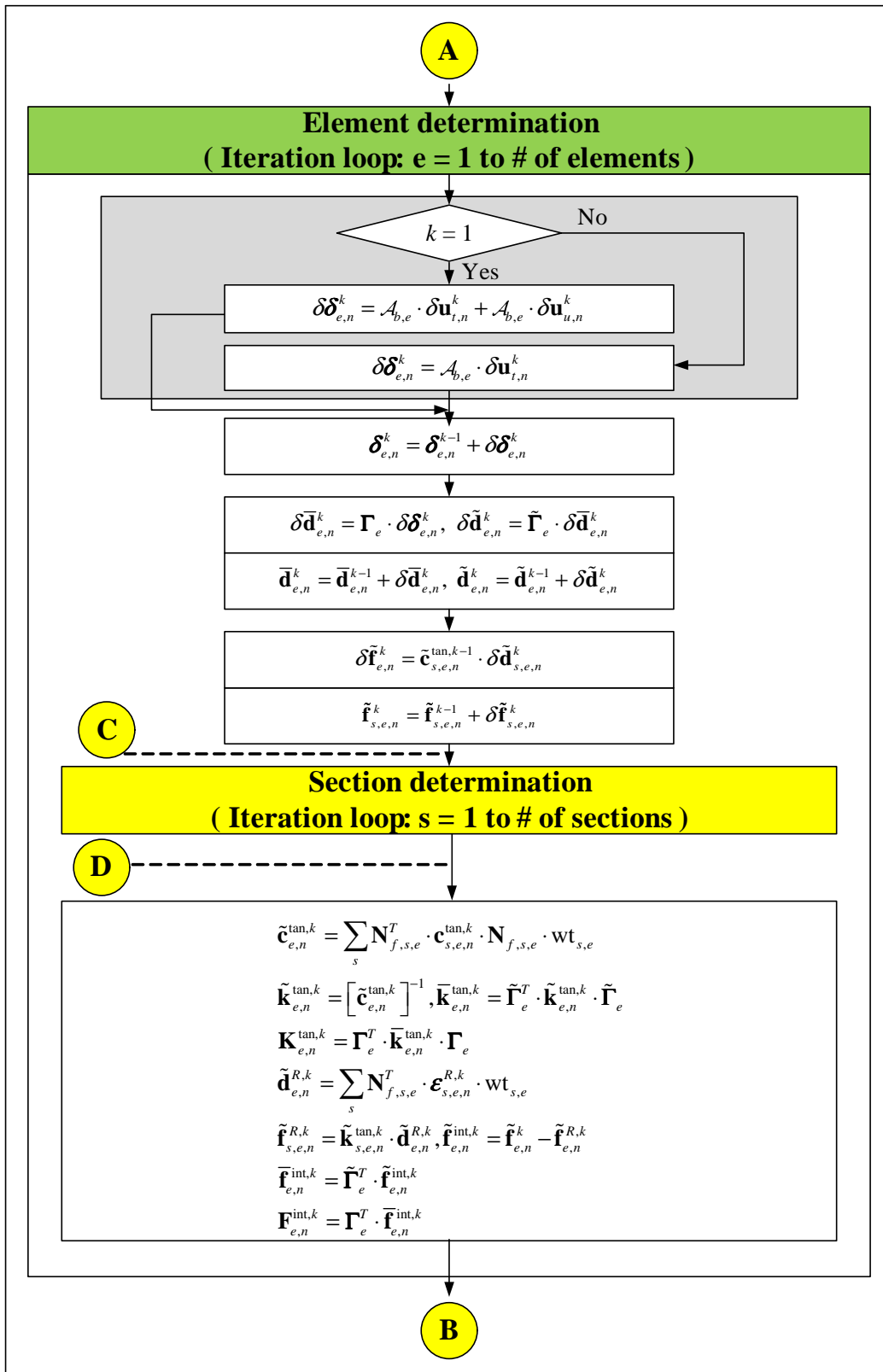


Figure 1.21: Flow chart of nonlinear analysis using flexibility-based method without iteration in the element level (2)

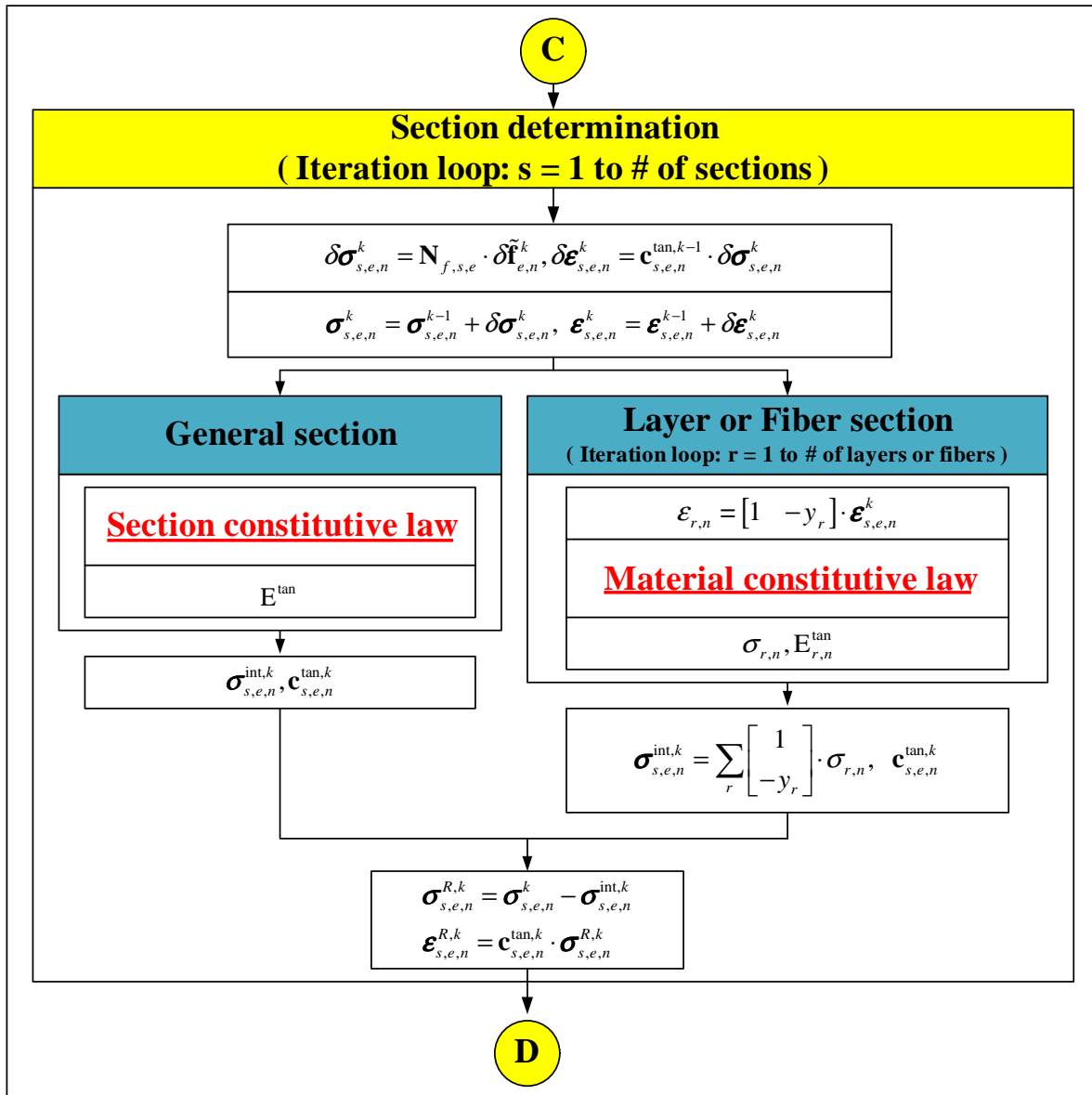


Figure 1.22: Flow chart of nonlinear analysis using flexibility-based method without iteration in the element level (3)

where, $\mathcal{A}_{b,e}$ is displacement extracting operator.

$$\delta_{e,n}^k = \delta_{e,n}^{k-1} + \delta\delta_{e,n}^k$$

and $\delta_{e,n}^0$ corresponds to $\delta_{e,n-1}$.

- Determine the element nodal displacement vector in local reference.

$$\begin{aligned}\delta\bar{\mathbf{d}}_{e,n}^k &= \mathbf{\Gamma}_e \cdot \delta\delta_{e,n}^k \\ \bar{\mathbf{d}}_{e,n}^k &= \bar{\mathbf{d}}_{e,n}^{k-1} + \delta\bar{\mathbf{d}}_{e,n}^k\end{aligned}$$

and $\bar{\mathbf{d}}_{e,n}^0$ corresponds to $\bar{\mathbf{d}}_{e,n-1}$.

- Determine the element nodal displacement vector without rigid body modes in local reference.

$$\begin{aligned}\delta\tilde{\mathbf{d}}_{e,n}^k &= \tilde{\mathbf{\Gamma}}_e \cdot \delta\tilde{\mathbf{d}}_{e,n}^k \\ \tilde{\mathbf{d}}_{e,n}^k &= \tilde{\mathbf{d}}_{e,n}^{k-1} + \delta\tilde{\mathbf{d}}_{e,n}^k\end{aligned}$$

and, $\tilde{\mathbf{d}}_{e,n}^0$ and $\tilde{\mathbf{d}}_{e,n}^0$ correspond to $\tilde{\mathbf{d}}_{e,n-1}$ and $\tilde{\mathbf{d}}_{e,n-1}$.

- ④ Determine the element nodal force vector without rigid body modes

$$\begin{aligned}\delta\tilde{\mathbf{f}}_{e,n}^k &= \tilde{\mathbf{k}}_{e,n}^{tan,k-1} \cdot \delta\tilde{\mathbf{d}}_{e,n}^k \\ \tilde{\mathbf{f}}_{e,n}^k &= \tilde{\mathbf{f}}_{e,n}^{k-1} + \delta\tilde{\mathbf{f}}_{e,n}^k\end{aligned}$$

and, $\tilde{\mathbf{f}}_{e,n}^{k=0}$ and $\tilde{\mathbf{k}}_{e,n}^{tan,k=0}$ correspond to $\tilde{\mathbf{f}}_{e,n-1}$, and $\tilde{\mathbf{k}}_{e,n-1}^{tan}$.

- ⑤ Start the section state determination by looping over the element's sections. The total number of section may vary from element to element. Therefore, the total number of sections in an element is nIp_e and it depends on the number of integration points in the Gauss Lobatto quadrature rule.

- Determine the section force vector.

$$\begin{aligned}\delta\sigma_{s,e,n}^k &= \mathbf{N}_{f,s,e} \cdot \delta\tilde{\mathbf{f}}_{e,n}^k \\ \sigma_{s,e,n}^k &= \sigma_{s,e,n}^{k-1} + \delta\sigma_{s,e,n}^k\end{aligned}$$

where, $\mathbf{N}_{f,s,e}$ is the matrix derived from the force interpolation functions at the s^{th} section of the e^{th} element, and $\sigma_{s,e,n}^{k=0}$ corresponds to $\sigma_{s,e,n-1}$.

- ⑥ Determine the section deformation vector. We again assume a general section. The state determination of each section is performed in loop s .

$$\begin{aligned}\delta\epsilon_{s,e,n}^k &= \mathbf{c}_{s,e,n}^{tan,k-1} \cdot \delta\sigma_{s,e,n}^k \\ \epsilon_{s,e,n}^k &= \epsilon_{s,e,n}^{k-1} + \delta\epsilon_{s,e,n}^k\end{aligned}$$

and, $\mathbf{c}_{s,e,n}^{tan,k=0}$ and $\epsilon_{s,e,n}^{k=0}$ correspond to $\tilde{\mathbf{c}}_{s,e,n-1}^{tan}$ and $\epsilon_{s,e,n-1}$.

- ⑦ Determine the section tangent flexibility matrix and the internal section force vector. If we assume that the section constitutive law is explicitly known, $\mathbf{c}_{s,e,n}^{tan,k}$ and $\sigma_{s,e,n}^{int,k}$ are both determined from $\epsilon_{s,e,n}^k$. However, in elastic section, we need not to compute $\mathbf{c}_{s,e,n}^{tan,k}$ again as it is identical to the initial section flexibility matrix $\tilde{\mathbf{c}}_{s,e}$.

- Determine the residual section force vector.

$$\sigma_{s,e,n}^{R,k} = \sigma_{s,e,n}^k - \sigma_{s,e,n}^{int,k}$$

- ⑧ Determine the residual section deformation vector.

$$\epsilon_{s,e,n}^{R,k} = \mathbf{c}_{s,e,n}^{tan,k} \cdot \sigma_{s,e,n}^{R,k}$$

- ⑨ Determine updated element flexibility matrix and the residual element nodal displacement vector.

The residual section deformation vectors and the section tangent flexibility matrices are integrated along the element using on the complimentary principle of virtual work to obtain the residual element nodal displacement

vector and the element tangent flexibility matrix.

$$\begin{aligned}\tilde{\mathbf{d}}_{e,n}^{R,k} &= \sum_s \mathbf{N}_{f,s,e}^T \cdot \boldsymbol{\varepsilon}_{s,e,n}^{R,k} \cdot \text{wt}_{s,e} \\ \tilde{\mathbf{c}}_{e,n}^{tan,k} &= \sum_s \mathbf{N}_{f,s,e}^T \cdot \mathbf{c}_{s,e,n}^{tan,k} \cdot \mathbf{N}_{f,s,e} \cdot \text{wt}_{s,e} \\ \tilde{\mathbf{K}}_{e,n}^{tan,k} &= [\tilde{\mathbf{c}}_{e,n}^{tan,k}]^{-1}\end{aligned}$$

where, $\text{wt}_{s,e}$ is the weight coefficient associated with the Jacobian at the s^{th} section of the e^{th} element.

⑩ Determine the internal element nodal force vector and the element tangent stiffness matrix.

$$\begin{aligned}\tilde{\mathbf{f}}_{e,n}^{R,k} &= \tilde{\mathbf{k}}_{e,n}^{tan,k} \cdot \tilde{\mathbf{d}}_{e,n}^{R,k} \\ \tilde{\mathbf{f}}_{e,n}^{int,k} &= \tilde{\mathbf{f}}_{e,n}^k - \tilde{\mathbf{f}}_{e,n}^{R,k} \\ \tilde{\mathbf{F}}_{e,n}^{int,k} &= \tilde{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{f}}_{e,n}^{int,k} \\ \mathbf{F}_{e,n}^{int,k} &= \mathbf{\Gamma}_e^T \cdot \tilde{\mathbf{F}}_{e,n}^{int,k} \\ \tilde{\mathbf{k}}_{e,n}^{tan,k} &= \tilde{\mathbf{\Gamma}}_e^T \cdot \tilde{\mathbf{k}}_{e,n}^{tan,k} \cdot \tilde{\mathbf{\Gamma}}_e \\ \mathbf{K}_{e,n}^{tan,k} &= \mathbf{\Gamma}_e^T \cdot \tilde{\mathbf{k}}_{e,n}^{tan,k} \cdot \mathbf{\Gamma}_e\end{aligned}$$

⑪ Determine the internal nodal force vector and the augmented tangent stiffness matrix.

$$\begin{aligned}\mathbf{P}_{t,n}^{int,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{F}_{e,n}^{int,k} \\ \mathbf{K}_{S,n}^{tan,k} &= \sum_e \mathcal{A}_{b,e}^T \cdot \mathbf{K}_{e,n}^{tan,k} \cdot \mathcal{A}_{b,e}\end{aligned}$$

where, $\mathcal{A}_{b,e}^T$ is force assembling operator.

⑫ Compute the residual nodal force vector at the structural level from Eq. 1.29. We then satisfy convergence with the residual nodal force vector.

Check convergence

- If $\mathbf{P}_{t,n}^{R,k}$ is within the specified tolerance, go to next force increment.
- If $\mathbf{P}_{t,n}^{R,k}$ is not within the specified tolerance, k is updated to $k + 1$ and the next Newton-Raphson iteration initiates. Eq. 1.31 in ② through ⑫ are repeated until convergence occurs at the structure level.

1.3 Layer/Fiber Section

So far, we have assumed that a section is characterized by a moment curvature relation, and that when the moment reaches the plastic/yield moment, it instantaneously plastifies. This is only an approximation, as in reality there is a gradual plastification starting from the outer fibers, and this plastification zone gradually spreads inward until the whole section ultimately becomes plastic. To capture this gradual spread one can either resort to continuum 2D/3D solid (finite) elements, which is computationally expensive/inefficient, or use layered elements. Hence, our objective is to derive $\mathbf{k}_{s,e}^{tan}(x)$ such that

$$\begin{Bmatrix} N(x) \\ M_z(x) \end{Bmatrix} = \mathbf{k}_{s,e}^{tan}(x) \begin{Bmatrix} \varepsilon(x) \\ \phi_z(x) \end{Bmatrix}$$

Ignoring the effects of transverse shear deformation (accounted for in the so-called Timoshenko beam), and thus assuming a linear strain distribution (Euler-Bernoulli beam), but a non linear stress-strain behavior, the stress distribution can be readily determined from

$$\sigma_t(x) = \frac{N_x(x)}{A(x)} \pm \frac{M_z(x)}{I_z(x)} y$$

At this point, from the nodal displacement, we can determine the section deformations (axial strain, $\varepsilon(x)$ and curvature, $\phi(x)$) (and thus the linear strain distribution), and since we have a nonlinear material, the exact location

of the neutral axis is not yet known, and at each fiber elevation we do have a different $E_r^{tan}(x)$. Hence, with reference to Fig. 1.23.

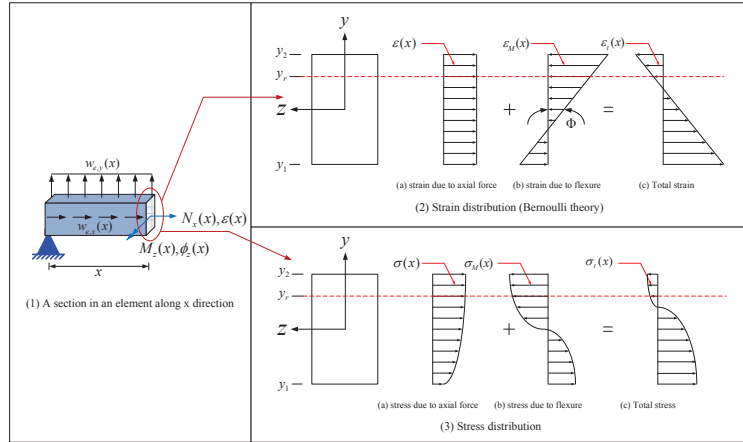


Figure 1.23: Stress and strain distribution of nonlinear material

Primary Terms are those due to pure axial and flexure:

Pure axial force due to $\sigma(x)$ is simply determined from

$$\begin{aligned} N_x(x) &= \int_{-y_1}^{y_2} \sigma(x) dA = \int_{-y_1}^{y_2} E_r^{tan}(x) \cdot \varepsilon(x) dA \\ &\simeq \sum_r E_r^{tan}(x) \cdot A_r(x) \cdot \varepsilon(x) \end{aligned} \quad (1.32)$$

Pure moment due to $\sigma_M(x)$ is considered next, and again we seek an expression of $(M(x))$ in terms of the curvature and $E_r^{tan}(x)$, and recalling that $I = \int y^2 dA$ and $\sigma_M(x) @_{y_r} = E_r^{tan}(x) \cdot \phi_z(x) \cdot y_r$

$$\begin{aligned} M_z(x) &= \int_{-y_1}^{y_2} \sigma_M(x) \cdot y dA \\ &= \int_{-y_1}^{y_2} E_r^{tan}(x) \cdot \phi_z(x) \cdot y \cdot y dA \\ &= \phi_z(x) \int_{-y_1}^{y_2} E_r^{tan}(x) \cdot \phi_z(x) \cdot y^2 dA \\ &\simeq \sum_r E_r^{tan}(x) \cdot A_r(x) \cdot y_r^2 \cdot \phi_z(x) \end{aligned} \quad (1.33)$$

Secondary Terms of axial force or moment are thus caused by curvature or axial strain.

Second axial force due to curvature

$$\begin{aligned} dN_x(x) &= -E_r^{tan}(x) \cdot \varepsilon_M(x) dA = -E_r^{tan}(x) \cdot \phi_z(x) \cdot y dA \\ N_x(x) &= - \int_{-y_1}^{y_2} E_r^{tan}(x) \cdot \phi_z(x) \cdot y dA \\ &\simeq - \sum_r E_r^{tan}(x) \cdot A_r(x) \cdot y_r \cdot \phi_z(x) \end{aligned} \quad (1.34)$$

where the strain ($\varepsilon_M(x)$) is obtained from the curvature ($\phi_z(x)$).

Secondary moment due to axial strain

$$\begin{aligned}
 dM_z(x) &= -E_r^{tan}(x) \cdot \varepsilon(x) \cdot y \cdot dA \\
 M_z(x) &= - \int_{-y_1}^{y_2} E_r^{tan}(x) \cdot \varepsilon(x) \cdot y dA \\
 &\simeq - \sum_r E_r^{tan}(x) \cdot A_r(x) \cdot y_r \cdot \varepsilon(x)
 \end{aligned} \tag{1.35}$$

Summing up from Eq. 1.32 to 1.35 within a matrix, $\mathbf{k}_{s,e}^{tan}(x)$ takes the form:

$$\begin{Bmatrix} N \\ M \end{Bmatrix} = \sum_r \begin{bmatrix} E_r^{tan}(x) \cdot A_r(x) & -E_r^{tan}(x) \cdot A_r(x) \cdot y_r \\ -E_r^{tan}(x) \cdot A_r(x) \cdot y_r & E_r^{tan}(x) \cdot A_r(x) \cdot y_r^2 \end{bmatrix} \begin{Bmatrix} \varepsilon(x) \\ \phi_z(x) \end{Bmatrix} \tag{1.36}$$

The implementation of this layer or fiber section will require an additional discretization of the cross section into layers or fibers as shown in Fig. 1.24.

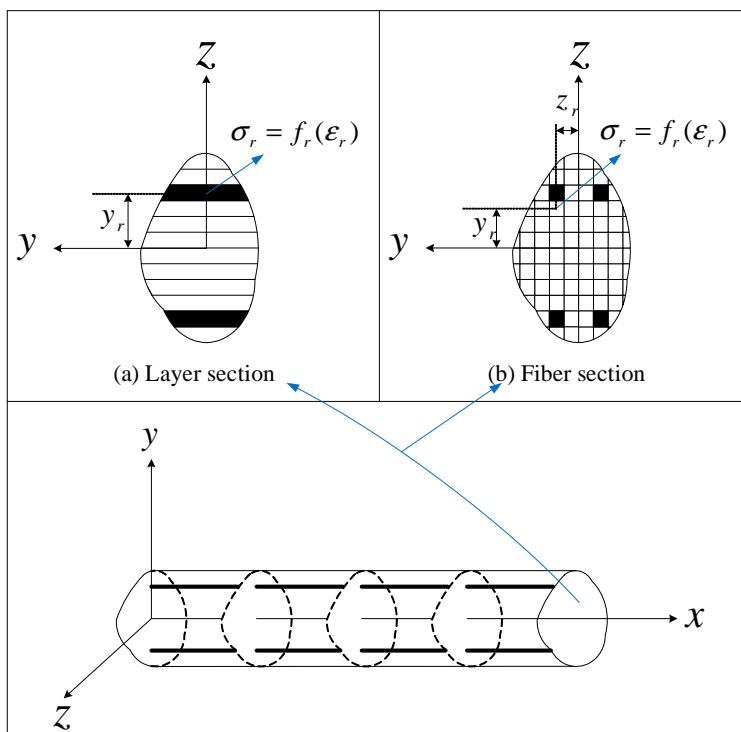


Figure 1.24: Layer/fiber sections-Distribution of control sections and section subdivision into layers/fibers

Noting that layer/fiber stress-strain relations are typically expressed as explicit functions of strain, state determination is shown in Fig. 1.25,

We note that this cross sectional definition allows us to easily specify longitudinal steel reinforcement. Shear reinforcement, on the other hand, can not be explicitly modeled, however, common practice is to assign modified properties to the confined concrete (?).

1.4 Zero-Length 2D Element Formulation

As discussed in Sec. ?? a methodology to account for nonlinearities in frame analysis is lumped plasticity. In this approach we consider that plastic hinges (or their formulation) contribute for the structural nonlinearity. Those hinges typically form at the end node regions. Therefore, zero-length element can be used at the end of beam or beam-column elements.

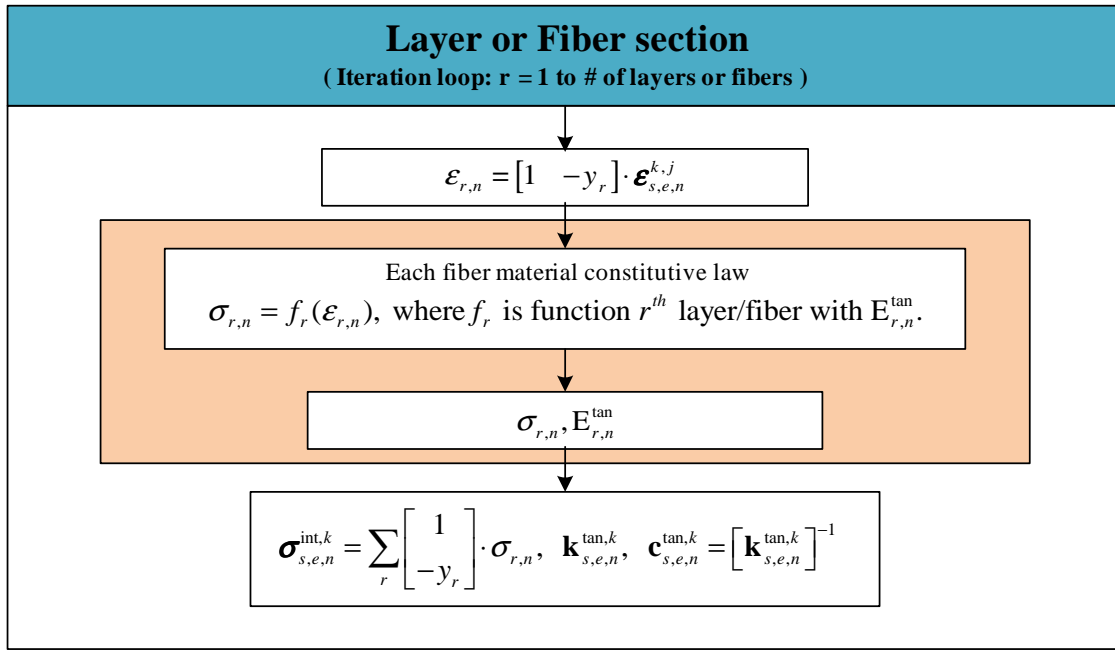


Figure 1.25: Layer/fiber section state determination

1.4.1 Formulation

The element end deformations in the reinforced concrete are composed of two types:

- flexural deformation that causes inelastic strains
- element end rotation which may be caused by the slip of longitudinal reinforcement in reinforced concrete or plastic hinges in steel members.

Fig. 1.26 describes zero-length 2D element and examples for usage. Its formulation does not account for coupling of the three possible degrees of freedom.

Constitutive law Section constitutive law is expressed as

$$\underbrace{\begin{Bmatrix} N_x \\ V_y \\ M_z \end{Bmatrix}}_{\boldsymbol{\sigma}_s} = \underbrace{\begin{bmatrix} [EA]^{\text{tan}} & 0 & 0 \\ 0 & [GA]^{\text{tan}} & 0 \\ 0 & 0 & [EI_z]^{\text{tan}} \end{bmatrix}}_{\mathbf{k}_s^{\text{tan}}} \underbrace{\begin{Bmatrix} \bar{u}_{x2} - \bar{u}_{x1} \\ \bar{v}_{y2} - \bar{v}_{y1} \\ \bar{\theta}_{z2} - \bar{\theta}_{z1} \end{Bmatrix}}_{\boldsymbol{\varepsilon}_s}$$

where, $[EA]^{\text{tan}}$, $[GA]^{\text{tan}}$ and $[EI_z]^{\text{tan}}$ are tangent stiffnesses associated with axial, shear and moment.

Equilibrium

Composing equilibrium equations between point A and point B in Fig. 1.27,

$$\begin{aligned} \bar{N}_{x1} &= [EA]^{\text{tan}} \cdot (\bar{u}_{x1} - \bar{u}_{x2}) \\ \bar{V}_{y1} &= [GA]^{\text{tan}} \cdot (\bar{v}_{y1} - \bar{v}_{y2}) \\ \bar{M}_{z1} &= [EI_z]^{\text{tan}} \cdot (\bar{\theta}_{z1} - \bar{\theta}_{z2}) \end{aligned} \tag{1.37}$$

Likewise between point B and point C,

$$\begin{aligned} \bar{N}_{x2} &= [EA]^{\text{tan}} \cdot (\bar{u}_{x2} - \bar{u}_{x1}) \\ \bar{V}_{y2} &= [GA]^{\text{tan}} \cdot (\bar{v}_{y2} - \bar{v}_{y1}) \\ \bar{M}_{z2} &= [EI_z]^{\text{tan}} \cdot (\bar{\theta}_{z2} - \bar{\theta}_{z1}) \end{aligned} \tag{1.38}$$

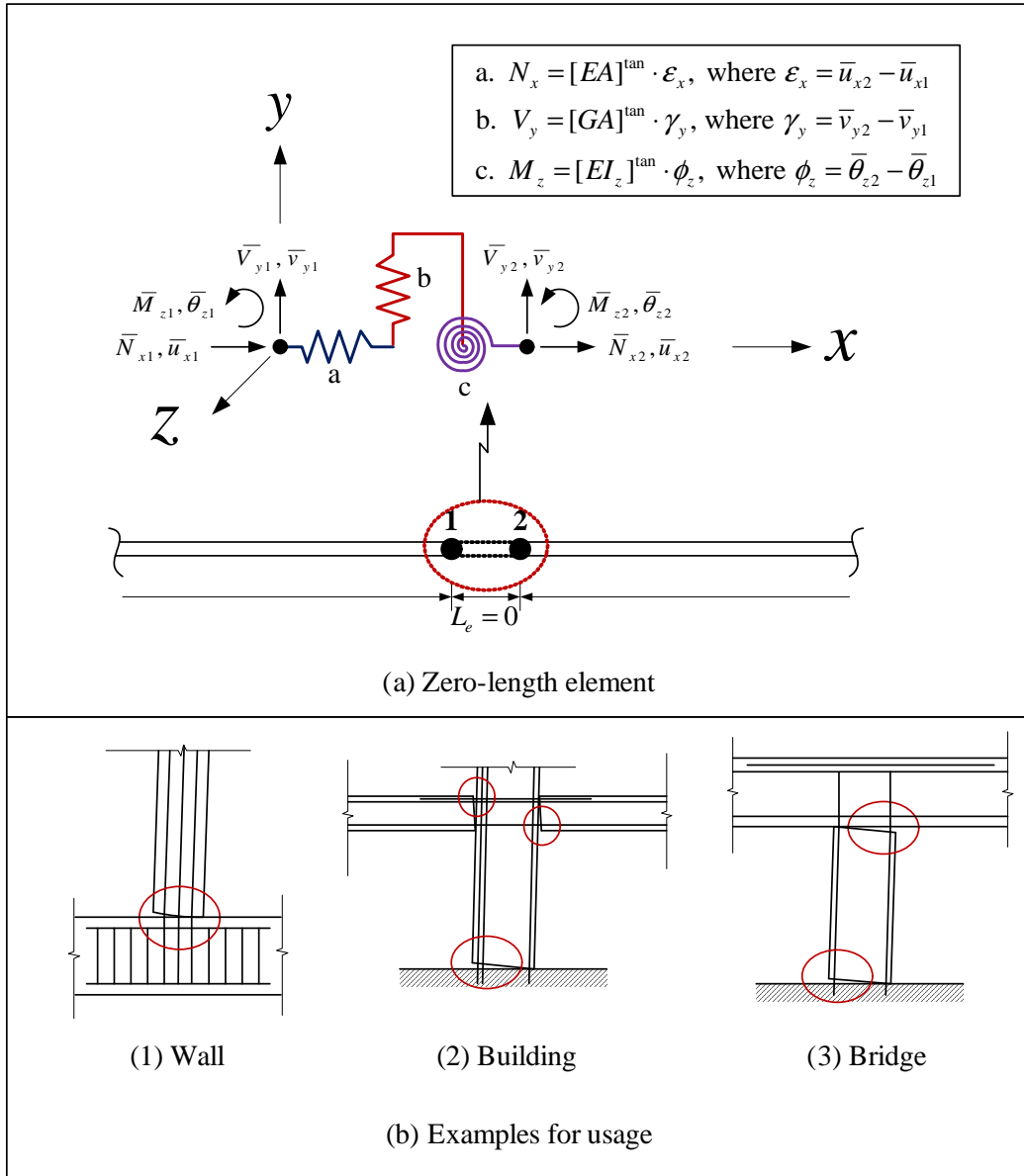


Figure 1.26: Zero-length 2D element(1)

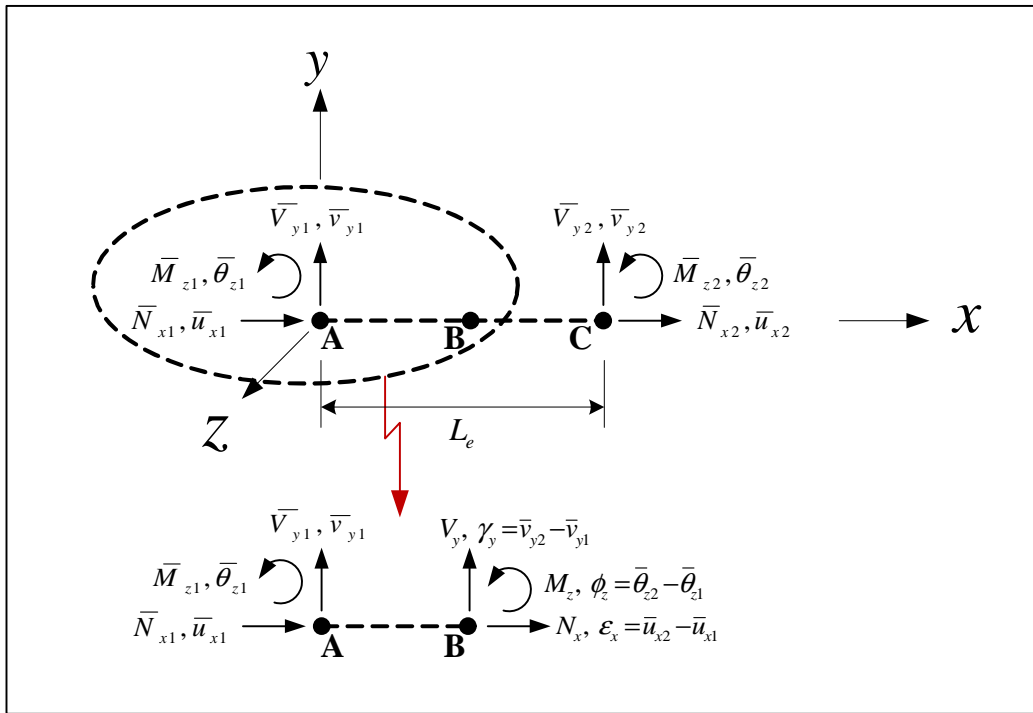


Figure 1.27: Zero-length 2D element(2)

Rewriting Eq. 1.37 and 1.38 to matrix form, the relationship between element nodal force and displacement vector is given by

$$\underbrace{\begin{Bmatrix} \bar{N}_{x1} \\ \bar{V}_{y1} \\ \bar{M}_{z1} \\ \bar{N}_{x2} \\ \bar{V}_{y2} \\ \bar{M}_{z2} \end{Bmatrix}}_{\bar{\mathbf{f}}_e} = \bar{\mathbf{k}}_e^{tan} \underbrace{\begin{Bmatrix} \bar{u}_{x1} \\ \bar{v}_{y1} \\ \bar{\theta}_{z1} \\ \bar{u}_{x2} \\ \bar{v}_{y2} \\ \bar{\theta}_{z2} \end{Bmatrix}}_{\bar{\mathbf{d}}_e}$$

where, $\bar{\mathbf{k}}_e^{tan}$ is the element stiffness matrix in local reference.

$$\bar{\mathbf{k}}_e^{tan} = \begin{bmatrix} [EA]^{tan} & 0 & 0 & -[EA]^{tan} & 0 & 0 \\ 0 & [GA]^{tan} & 0 & 0 & -[GA]^{tan} & 0 \\ 0 & 0 & [EI_z]^{tan} & 0 & 0 & -[EI_z]^{tan} \\ -[EA]^{tan} & 0 & 0 & [EA]^{tan} & 0 & 0 \\ 0 & -[GA]^{tan} & 0 & 0 & [GA]^{tan} & 0 \\ 0 & 0 & -[EI_z]^{tan} & 0 & 0 & [EI_z]^{tan} \end{bmatrix} \quad (1.39)$$

1.4.2 Coordinate system in zero-length 2D element

Coordinate system in zero-length 2D element is same as in Sec. 1.2.1.2, Fig. 1.2.

1.4.3 Element state determination with 2 Dimension and 3 degrees of freedom per node

With reference Fig. 1.28, we will examine one single step of zero-length 2D element for nonlinear analysis.

Step 1: Determine the section deformation vector, axial deformation, shear deformation and curvature.

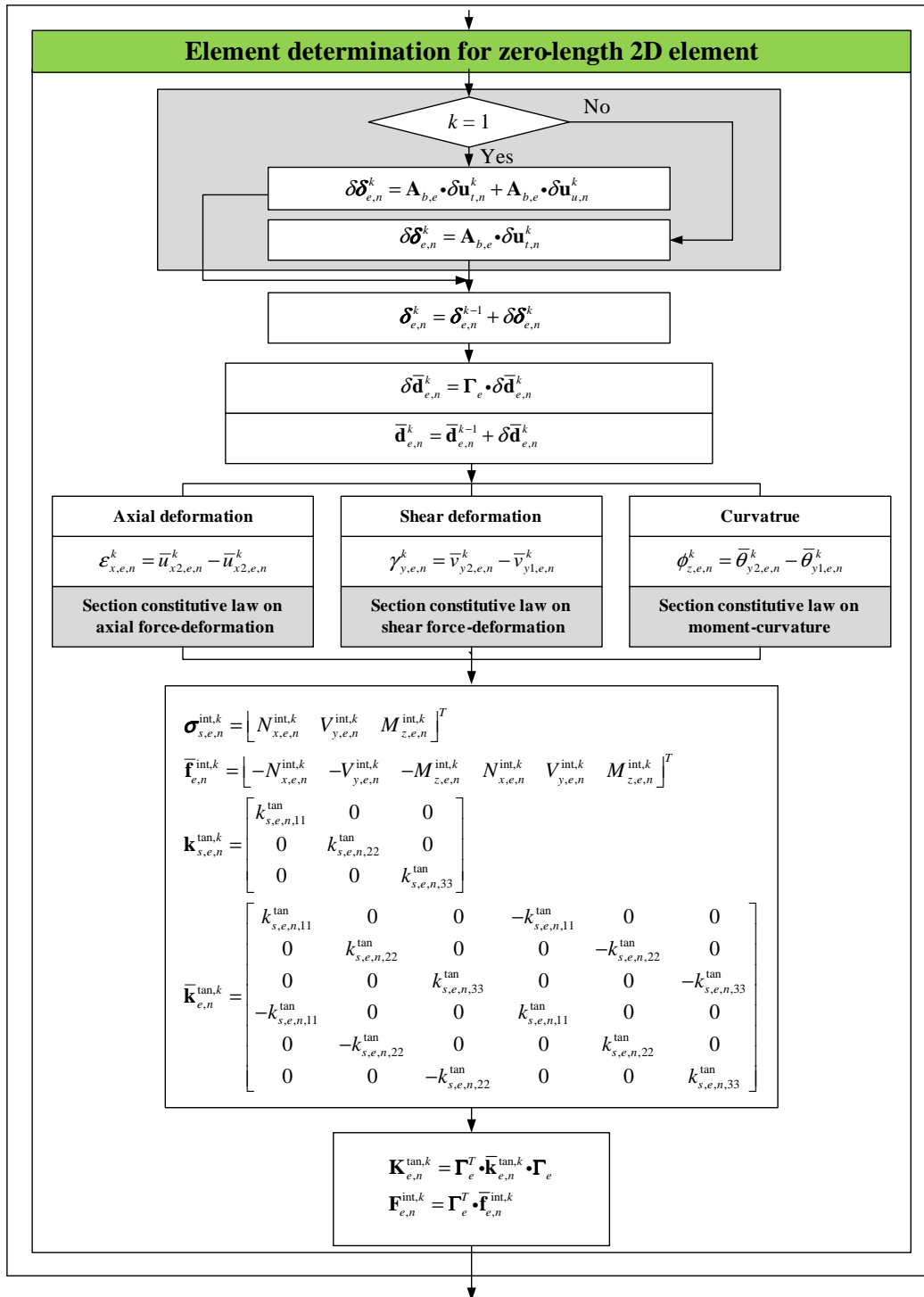


Figure 1.28: Flow chart of zero-length 2D element for element state determination

For each deformation, we extract the associated components from $\bar{\mathbf{d}}_{e,n}^k$.

$$\begin{aligned}\bar{\mathbf{d}}_{e,n}^k &= [\bar{u}_{x1,e,n}^k \ \bar{v}_{y1,e,n}^k \ \bar{\theta}_{z1,e,n}^k \ \bar{u}_{x2,e,n}^k \ \bar{v}_{y2,e,n}^k \ \bar{\theta}_{z2,e,n}^k]^T \\ \boldsymbol{\varepsilon}_{s,e,n}^k &= [\varepsilon_{x,e,n}^k \ \gamma_{y,e,n}^k \ \phi_{z,e,n}^k]^T \\ \varepsilon_{x,e,n}^k &= \bar{u}_{x2,e,n}^k - \bar{u}_{x1,e,n}^k \\ \gamma_{y,e,n}^k &= \bar{v}_{y2,e,n}^k - \bar{v}_{y1,e,n}^k \\ \phi_{z,e,n}^k &= \bar{\theta}_{z2,e,n}^k - \bar{\theta}_{z1,e,n}^k\end{aligned}\tag{1.40}$$

$$\tag{1.41}$$

$$\tag{1.42}$$

where, Eq. 1.40 defines axial section deformation, Eq. 1.41 the shear deformation, and Eq. 1.42 the curvature.

Step 2: Determine the section tangent stiffness associated with axial force-deformation, shear force-deformation, and moment-curvature in the section constitutive laws. Section constitutive laws modified with several variables in function of material constitutive law associated with uniaxial stress-strain relationship can be used (?). The internal section force vector is determined next. If we assume that the section constitutive law is explicitly known, $\mathbf{k}_{s,e,n}^{tan,k}$ and $\boldsymbol{\sigma}_{s,e,n}^{int,k}$ are determined from $\boldsymbol{\varepsilon}_{s,e,n}^k$. However, in elastic section, we need not to compute $\mathbf{k}_{s,e,n}^{tan,k}$ again as it is identical to the initial section stiffness matrix $\mathbf{k}_{s,e}$.

For an elastic section,

$$\begin{aligned}\mathbf{k}_{s,e,n}^{tan} &= \mathbf{k}_{s,e} \\ \underbrace{\begin{Bmatrix} N_{x,e,n}^{int,k} \\ V_{y,e,n}^{int,k} \\ M_{z,e,n}^{int,k} \end{Bmatrix}}_{\boldsymbol{\sigma}_{s,e,n}^{int,k}} &= \mathbf{k}_{s,e,n}^{tan} \underbrace{\begin{Bmatrix} \varepsilon_{x,e,n}^k \\ \gamma_{y,e,n}^k \\ \phi_{z,e,n}^k \end{Bmatrix}}_{\boldsymbol{\varepsilon}_{s,e,n}^k}\end{aligned}$$

where, $\mathbf{k}_{s,e,n}^{tan,k}$ is the section tangent stiffness matrix at k^{th} iteration.

Step 3: Determine the internal element nodal force vector and the element tangent stiffness matrix from Eq. 1.39.

$$\bar{\mathbf{f}}_{e,n}^{int,k} = [N_{x,e,n}^{int,k} \ V_{y,e,n}^{int,k} \ M_{z,e,n}^{int,k} \ -N_{x,e,n}^{int,k} \ -V_{y,e,n}^{int,k} \ -M_{z,e,n}^{int,k}]^T$$

$$\bar{\mathbf{k}}_e^{tan,k} = \begin{bmatrix} EA_{e,n}^{tan,k} & 0 & 0 & -EA_{e,n}^{tan,k} & 0 & 0 \\ 0 & GA_{e,n}^{tan,k} & 0 & 0 & -GA_{e,n}^{tan,k} & 0 \\ 0 & 0 & EI_{z,e,n}^{tan,k} & 0 & 0 & -EI_{z,e,n}^{tan,k} \\ -EA_{e,n}^{tan,k} & 0 & 0 & EA_{e,n}^{tan,k} & 0 & 0 \\ 0 & -GA_{e,n}^{tan,k} & 0 & 0 & GA_{e,n}^{tan,k} & 0 \\ 0 & 0 & -EI_{z,e,n}^{tan,k} & 0 & 0 & EI_{z,e,n}^{tan,k} \end{bmatrix}$$

where, $\bar{\mathbf{k}}_{e,n}^{tan,k}$ is the element tangent stiffness matrix in local reference.

We determine $\mathbf{F}_{e,n}^{int,k}$ and $\mathbf{K}_{e,n}^{tan,k}$.

$$\begin{aligned}\mathbf{F}_{e,n}^{int,k} &= \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{f}}_{e,n}^{int,k} \\ \mathbf{K}_{e,n}^{tan,k} &= \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{k}}_{e,n}^{tan,k} \cdot \boldsymbol{\Gamma}_e\end{aligned}$$

1.5 Zero-Length Section Element Formulation

Zero-length section element is analogous to the zero length element, however, it uses layer/fiber. This element enables us to model the shift in center of section rotation which may occur (in bar-slip for example). The element is formulated on the basis of coupled axial force and moment.

Fig. 1.29 describes zero-length 2D section element.

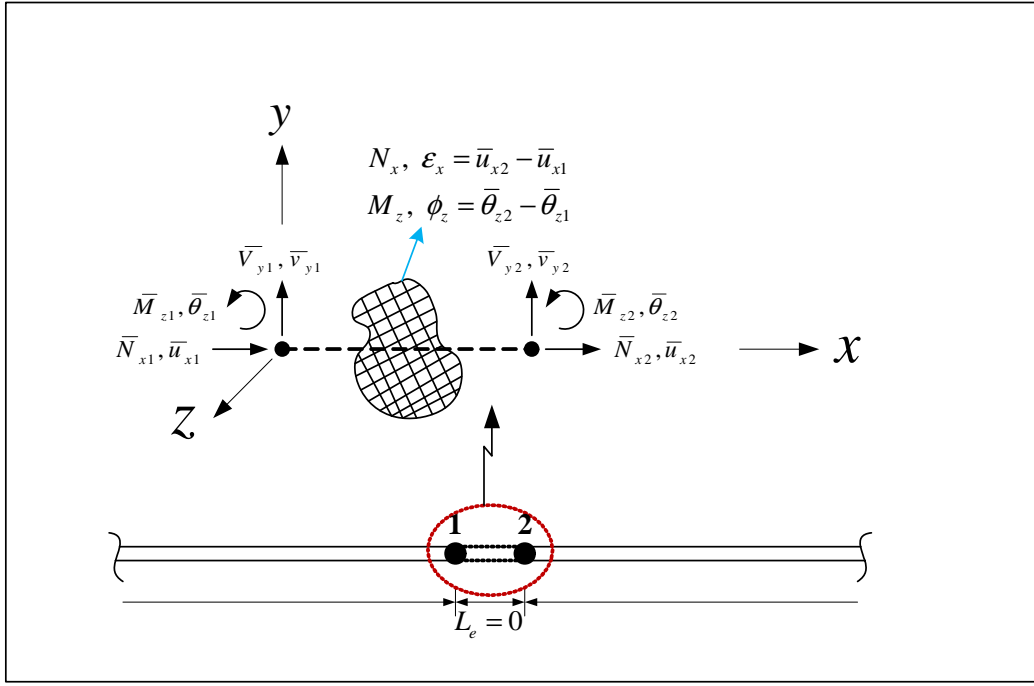


Figure 1.29: Zero-length 2D section element(1)

1.5.1 Formulation

Constitutive law Section constitutive law is expressed as

$$\underbrace{\begin{Bmatrix} N_x \\ M_z \end{Bmatrix}}_{\sigma_s} = \underbrace{\begin{bmatrix} k_{s,11}^{tan} & k_{s,12}^{tan} \\ k_{s,21}^{tan} & k_{s,22}^{tan} \end{bmatrix}}_{\mathbf{k}_s^{tan}} \cdot \underbrace{\begin{Bmatrix} \bar{u}_{x2} - \bar{u}_{x1} \\ \bar{\theta}_{z2} - \bar{\theta}_{z1} \end{Bmatrix}}_{\epsilon_s}$$

where, \mathbf{k}_s^{tan} is the section tangent stiffness matrix obtained from layer/fiber state determination.

Equilibrium Zero-length section element is based on Bernoulli beam theory.

Composing equilibrium equations between point A and point B in Fig. 1.30,

$$\begin{aligned} \bar{N}_{x1} &= k_{s,11}^{tan} \cdot (\bar{u}_{x1} - \bar{u}_{x2}) + k_{s,12}^{tan} \cdot (\bar{\theta}_{z1} - \bar{\theta}_{z2}) \\ \bar{M}_{z1} &= k_{s,21}^{tan} \cdot (\bar{u}_{x1} - \bar{u}_{x2}) + k_{s,22}^{tan} \cdot (\bar{\theta}_{z1} - \bar{\theta}_{z2}) \end{aligned} \quad (1.43)$$

Likewise between point B and point C,

$$\begin{aligned} \bar{N}_{x2} &= k_{s,11}^{tan} \cdot (\bar{u}_{x2} - \bar{u}_{x1}) + k_{s,12}^{tan} \cdot (\bar{\theta}_{z2} - \bar{\theta}_{z1}) \\ \bar{M}_{z2} &= k_{s,21}^{tan} \cdot (\bar{u}_{x2} - \bar{u}_{x1}) + k_{s,22}^{tan} \cdot (\bar{\theta}_{z2} - \bar{\theta}_{z1}) \end{aligned} \quad (1.44)$$

Rewriting Eq. 1.43 and 1.44 to matrix form, the relationship between element nodal force and displacement vector is given by

$$\underbrace{\begin{Bmatrix} \bar{N}_{x1} \\ 0 \\ \bar{M}_{z1} \\ \bar{N}_{x2} \\ 0 \\ \bar{M}_{z2} \end{Bmatrix}}_{\bar{\mathbf{f}}_e} = \mathbf{k}_e^{tan} \underbrace{\begin{Bmatrix} \bar{u}_{x1} \\ \bar{\theta}_{z1} \\ \bar{u}_{x2} \\ \bar{\theta}_{z2} \end{Bmatrix}}_{\bar{\mathbf{d}}_e}$$

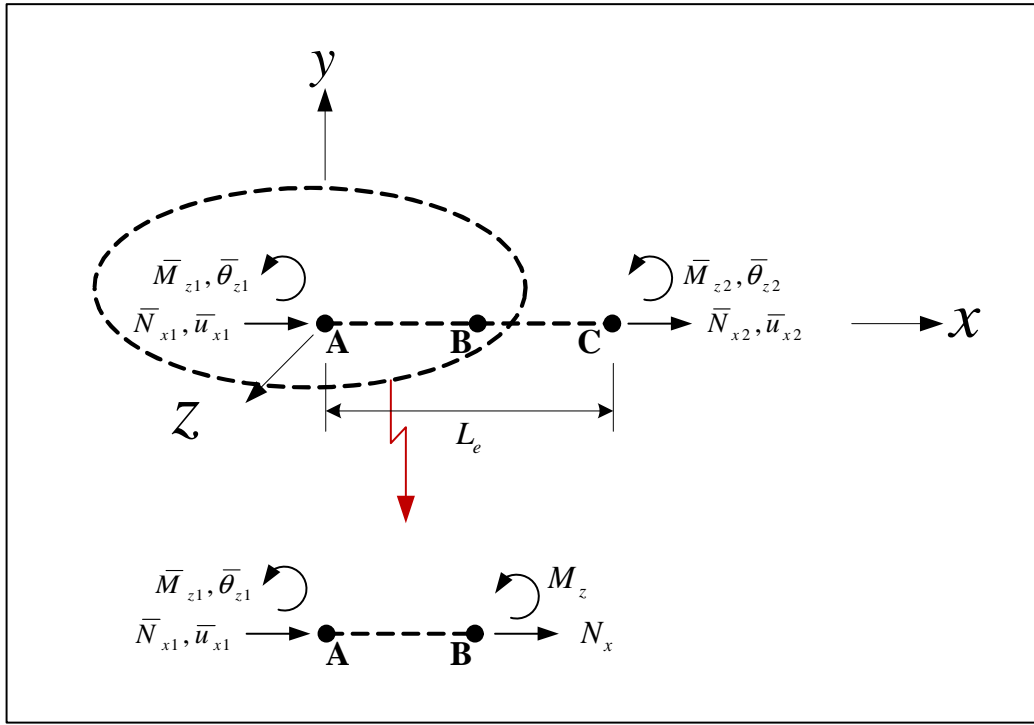


Figure 1.30: Zero-length 2D section element(2)

where, $\bar{\mathbf{k}}_e^{tan}$ is the element stiffness matrix in local reference.

$$\bar{\mathbf{k}}_e^{tan} = \begin{bmatrix} k_{s,11}^{tan} & 0 & k_{s,12}^{tan} & -k_{s,11}^{tan} & 0 & -k_{s,12}^{tan} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ k_{s,21}^{tan} & 0 & k_{s,22}^{tan} & -k_{s,21}^{tan} & 0 & -k_{s,22}^{tan} \\ -k_{s,11}^{tan} & 0 & -k_{s,12}^{tan} & k_{s,11}^{tan} & 0 & k_{s,12}^{tan} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -k_{s,21}^{tan} & 0 & -k_{s,22}^{tan} & k_{s,21}^{tan} & 0 & k_{s,22}^{tan} \end{bmatrix} \quad (1.45)$$

1.5.2 Coordinate system in zero-length 2D section element

Coordinate system in zero-length 2D element is same as in Sec. 1.2.1.2, Fig. 1.2.

1.5.3 Element state determination with 2 Dimension and 3 degrees of freedom per node

With reference Fig. 1.31, we will examine one single step of zero-length 2D section element for nonlinear analysis.

Step 1: Determine the section deformation vector, axial deformation and curvature.

For each deformation, we extract the associated components from $\bar{\mathbf{d}}_{e,n}^k$.

$$\begin{aligned} \bar{\mathbf{d}}_{e,n}^k &= [\bar{u}_{x1,e,n}^k \ 0 \ \bar{\theta}_{z1,e,n}^k \ \bar{u}_{x2,e,n}^k \ 0 \ \bar{\theta}_{z2,e,n}^k]^T \\ \boldsymbol{\epsilon}_{s,e,n}^k &= [\epsilon_{x,e,n}^k \ \phi_{z,e,n}^k]^T \\ \epsilon_{x,e,n}^k &= \bar{u}_{x2,e,n}^k - \bar{u}_{x1,e,n}^k \\ \phi_{z,e,n}^k &= \bar{\theta}_{z2,e,n}^k - \bar{\theta}_{z1,e,n}^k \end{aligned}$$

Step 2: Determine the section tangent stiffness associated with axial force-deformation and moment-curvature using layer/fiber state determination in Sec. 1.3. Determine next the internal section force vector. If we assume that the material constitutive law is explicitly known, $\mathbf{k}_{s,e,n}^{tan,k}$ and $\boldsymbol{\sigma}_{s,e,n}^{int,k}$ are determined from $\boldsymbol{\epsilon}_{s,e,n}^k$. However, in the section with elastic material, we need not to compute $\mathbf{k}_{s,e,n}^{tan,k}$ again as it is identical to the initial section stiffness matrix $\mathbf{k}_{s,e}$.

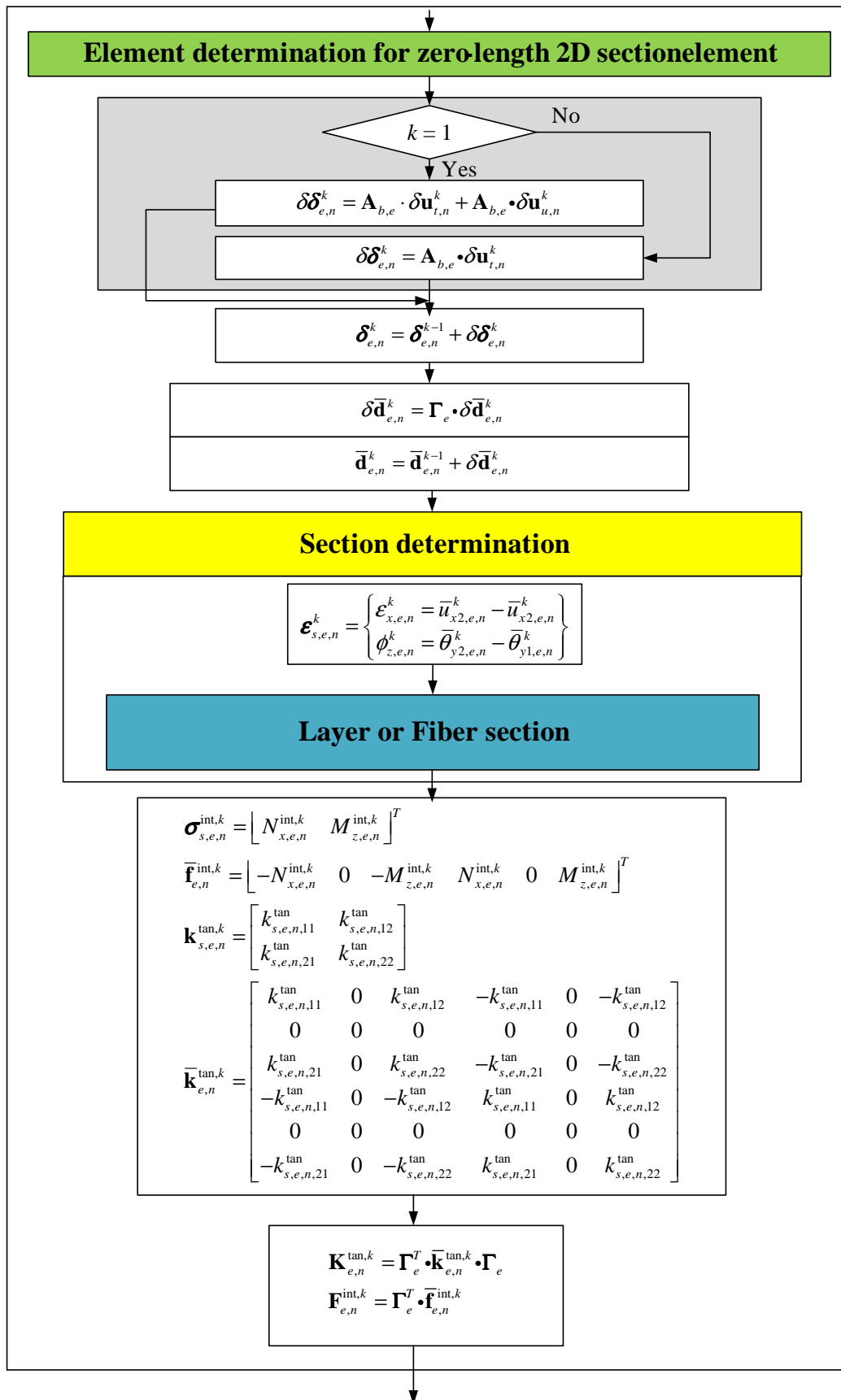


Figure 1.31: Flow chart of zero-length 2D section element for element state determination

If we have a section with elastic material, then

$$\mathbf{k}_{s,e,n}^{tan} = \mathbf{k}_{s,e}$$

$$\underbrace{\begin{Bmatrix} N_{x,e,n}^{int,k} \\ M_{z,e,n}^{int,k} \end{Bmatrix}}_{\boldsymbol{\sigma}_{s,e,n}^{int,k}} = \mathbf{k}_{s,e,n}^{tan} \underbrace{\begin{Bmatrix} \varepsilon_{x,e,n}^k \\ \phi_{z,e,n}^k \end{Bmatrix}}_{\boldsymbol{\varepsilon}_{s,e,n}^k}$$

where, $\mathbf{k}_{s,e,n}^{tan,k}$ is the section tangent stiffness matrix at k^{th} iteration.

Step 3: Determine the internal element nodal force vector and the element tangent stiffness matrix from Eq. 1.45.

$$\bar{\mathbf{f}}_{e,n}^{int,k} = [N_{x,e,n}^{int,k}, 0, M_{z,e,n}^{int,k}, -N_{x,e,n}^{int,k}, 0, -M_{z,e,n}^{int,k}]^T$$

$$\bar{\mathbf{K}}_{e,n}^{tan,k} = \begin{bmatrix} k_{s,e,n,11}^{tan,k} & 0 & k_{s,e,n,12}^{tan,k} & -k_{s,e,n,11}^{tan,k} & 0 & -k_{s,e,n,12}^{tan,k} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ k_{s,e,n,21}^{tan,k} & 0 & k_{s,e,n,22}^{tan,k} & -k_{s,e,n,21}^{tan,k} & 0 & -k_{s,e,n,22}^{tan,k} \\ -k_{s,e,n,11}^{tan,k} & 0 & -k_{s,12e,n}^{tan,k} & k_{s,e,n,11}^{tan,k} & 0 & k_{s,e,n,12}^{tan,k} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -k_{s,e,n,21}^{tan,k} & 0 & -k_{s,e,n,22}^{tan,k} & k_{s,e,n,21}^{tan,k} & 0 & k_{s,e,n,22}^{tan,k} \end{bmatrix}$$

where, $\bar{\mathbf{K}}_{e,n}^{tan,k}$ is the element tangent stiffness matrix in local reference.

We determine $\mathbf{F}_{e,n}^{int,k}$ and $\mathbf{K}_{e,n}^{tan,k}$.

$$\mathbf{F}_{e,n}^{int,k} = \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{f}}_{e,n}^{int,k}$$

$$\mathbf{K}_{e,n}^{tan,k} = \boldsymbol{\Gamma}_e^T \cdot \bar{\mathbf{K}}_{e,n}^{tan,k} \cdot \boldsymbol{\Gamma}_e$$

1.6 Element Force

Structures should resist applied external forces. Applicable external forces of an element basically are divided into three forces; element nodal forces, element distributed forces and element nodal displacements such as settlements or pushover tests due to displacement control. These external forces are shown in Fig. 1.32.

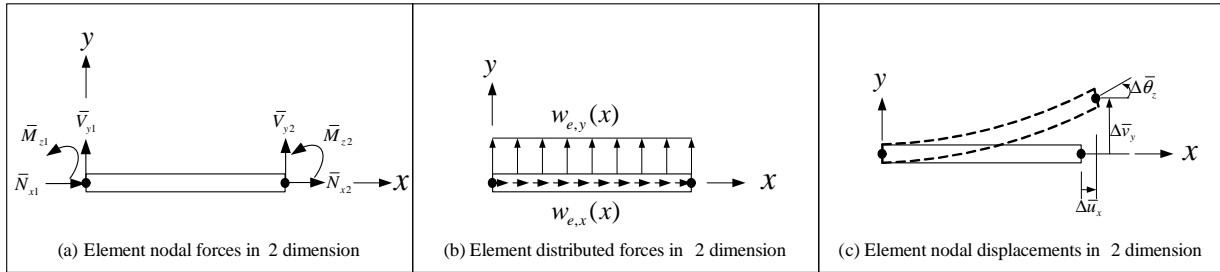


Figure 1.32: Forces of an element

Element nodal forces shown in Fig. 1.32 (a) are applied directly at the corresponding degrees of freedom. However, so far we have not accounted for the element forces $\mathbf{w}_e(x)$ (as defined in Eq. 1.14) shown in Fig. 1.32 (b). Those can be lumped as concentrated forces at suitably selected arbitrary nodes, and the degrees of freedom at these and the actual joints are treated as the unknowns. For element distributed forces, we should consider the nodal equivalent forces $\bar{\mathbf{f}}_e^{nef}$ of element distributed forces in structural level irrespective of whether we use the stiffness-based or the mixed stiffness-based and flexibility-based method. $\bar{\mathbf{f}}_e^{nef}$ on element distributed forces of an element is based on stiffness-based formulation.

$$\begin{aligned}
 \mathbf{N}_d(x) &= \begin{bmatrix} 1 - \frac{x}{L_e} & 0 & 0 & \frac{x}{L_e} & 0 & 0 \\ 0 & \frac{2x^3}{L_e^3} - \frac{3x^2}{L_e^2} + 1 & \frac{x^3}{L_e^2} - \frac{2x^2}{L_e} + x & 0 & \frac{3x^2}{L_e^2} - \frac{2x^3}{L_e^3} & \frac{x^3}{L_e^2} - \frac{x^2}{L_e} \end{bmatrix} \\
 \mathbf{w}_e(x) &= \begin{Bmatrix} w_{e,x}(x) \\ w_{e,y}(x) \end{Bmatrix} \\
 \bar{\mathbf{f}}_e^{nef} &= \int_0^{L_e} \mathbf{N}_d(x) \cdot \mathbf{w}_e(x) dx
 \end{aligned} \tag{1.46}$$

For a uniformly distributed force vector $\mathbf{w}_e(x) = \begin{Bmatrix} w_x \\ w_y \end{Bmatrix}$, then Eq. 1.46 reduces to the classical

$$\bar{\mathbf{f}}_e^{nef} = \left[\frac{w_x \cdot L_e}{2}, \frac{w_y \cdot L_e}{2}, \frac{w_y \cdot L_e^2}{12}, \frac{w_x \cdot L_e}{2}, \frac{w_y \cdot L_e}{2}, -\frac{w_y \cdot L_e^2}{12} \right]^T \tag{1.47}$$

which are shown in Fig. 1.33.

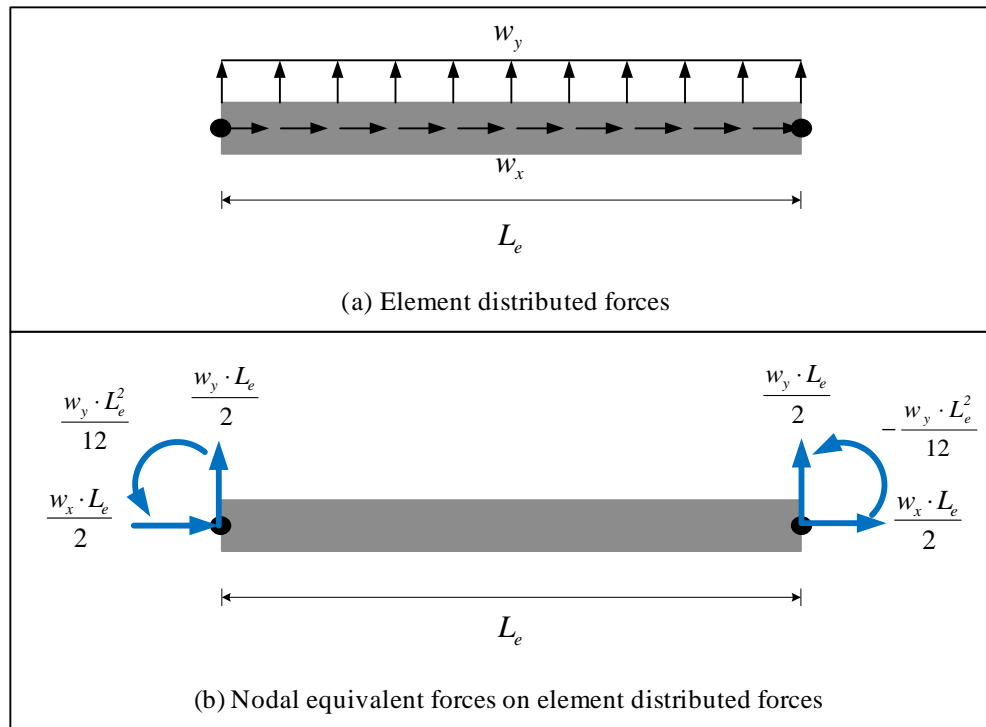


Figure 1.33: Nodal equivalent forces of an element

Element nodal displacements considered as external forces are shown in Fig. 1.32 (c). Those are applied as constraint degrees of freedom.

Let us consider the equilibrium equation considered by element nodal forces, element distributed forces, and element nodal displacements in an element shown in Fig. 1.34.

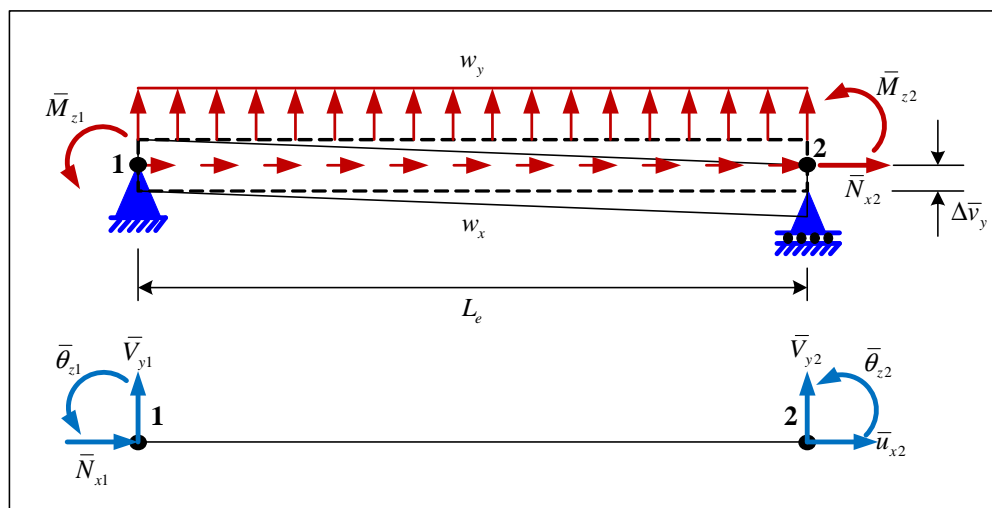


Figure 1.34: Example with element distributed forces and nodal displacement applied at constraint degree of freedom

If the element has elastic section, we obtain equilibrium equation, Eq. 1.48.

$$\underbrace{\begin{Bmatrix} \bar{N}_{x1}^? \\ \bar{V}_{y1}^? \\ \bar{M}_{z1}^? \\ \bar{N}_{x2}^? \\ \bar{V}_{y2}^? \\ \bar{M}_{z2}^? \end{Bmatrix}}_{\bar{\mathbf{f}}_e} = \underbrace{\begin{bmatrix} \frac{EA}{L_e} & 0 & 0 & -\frac{EA}{L_e} & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & \frac{6EI_z}{L^2} & 0 & -\frac{12EI_z}{L^3} & \frac{6EI_z}{L^2} \\ 0 & \frac{6EI_z}{L^2} & \frac{4EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & \frac{2EI_z}{L} \\ -\frac{EA}{L_e} & 0 & 0 & \frac{EA}{L_e} & 0 & 0 \\ 0 & -\frac{12EI_z}{L^3} & -\frac{6EI_z}{L^2} & 0 & \frac{12EI_z}{L^3} & -\frac{6EI_z}{L^2} \\ 0 & \frac{6EI_z}{L^2} & \frac{2EI_z}{L} & 0 & -\frac{6EI_z}{L^2} & \frac{4EI_z}{L} \end{bmatrix}}_{\bar{\mathbf{k}}_e} \cdot \underbrace{\begin{Bmatrix} 0^? \\ 0^? \\ \bar{\theta}_{z1}^? \\ \bar{u}_{x2}^? \\ \Delta \bar{V}_y^? \\ \bar{\theta}_{z2}^? \end{Bmatrix}}_{\bar{\mathbf{d}}_e} - \underbrace{\begin{Bmatrix} \frac{w_x \cdot L_e}{2} \\ \frac{w_y \cdot L_e}{2} \\ \frac{w_y \cdot L_e^2}{12} \\ \frac{w_x \cdot L_e}{2} \\ \frac{w_y \cdot L_e}{2} \\ -\frac{w_y \cdot L_e^2}{12} \end{Bmatrix}}_{\bar{\mathbf{f}}_e^{nef}} \quad (1.48)$$

where, superscript ? indicates unknown forces and displacements, and \checkmark known forces and displacements.

1.7 Summary

This chapter has presented the details of the element formulation coded in Mercury. The Mercury supports most of the elements (in particular flexibility based fiber elements, zero length and zero length section elements). These elements are widely used and tested and would enable the analysis of complex structure such as reinforced concrete frame.

Chapter 2

CONSTITUTIVE MODELS

The nonlinear analysis of structures hinges on the constitutive relation relating strains to stresses, better yet incremental strains to incremental stresses through E^{tan} . This chapter is devoted to the derivation of such relations for steel and concrete. In each case we will consider both analytically (or thermodynamically) derived models as well as heuristic ones which are found to be acceptable.

It should be noted that the constitutive models expressed in terms of stress-strain relationships are to be used in fiber/layered sections (though the simplified bilinear model can be used in zero-length elements).

2.1 Steel Models

Most metallic materials, when subjected to high stress level, exhibit plasticity behavior, i.e. when force is removed, the body does not return to its original shape, but has some permanent plastic deformation associated with it.

Materials such as steel or concrete, commonly used in structures, show yielding and plastic deformation. A typical uniaxial stress-strain curve for steel is shown in Fig. 2.1. Following a linear response, steel exhibits an abrupt change

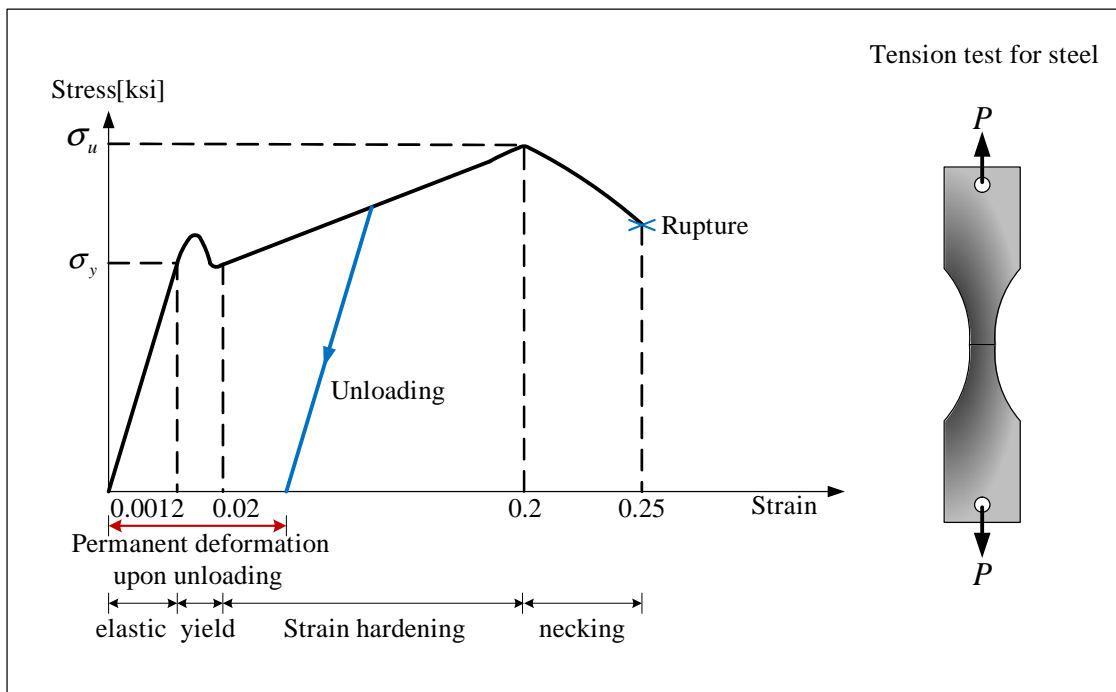


Figure 2.1: Uniaxial stress-strain curve for steel

in stiffness which occurs at the yield point. Beyond yielding, a permanent deformation is introduced upon unloading. This behavior can be idealized by a bilinear stress-strain relationship with two slopes, the second being called the tangent modulus E^{tan} . After reaching the yield point, the slope could be smaller, equal, or greater than zero as shown in Fig. 2.2(a).

Fig. 2.2(b) illustrates on elastic-perfectly plastic situation, that is after reaching the yield stress, the material starts flowing plastically without any further increase or decrease. The specimen is first loaded to point A, reaches its yield stress, and then flows plastically at point B, we unload the specimen until we reach a zero-load condition at point C. However, since we have loaded the specimen beyond the yield point, we observe that a permanent deformation or a permanent strain has been introduced in the material. We denote this permanent strain in one dimension the plastic strain ϵ^p which is permanent deformation.

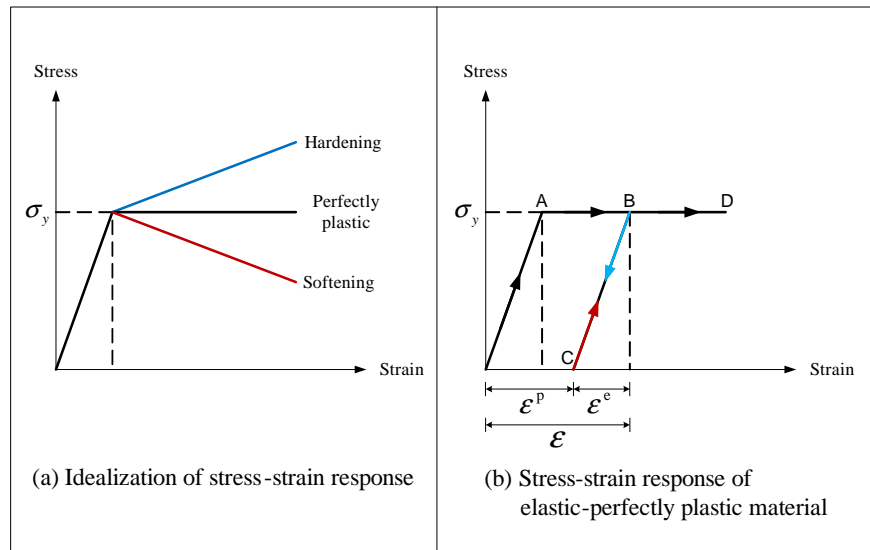


Figure 2.2: Idealized stress-strain response hardening material

The total strain ε can thus be decomposed into an elastic strain ε^e and a plastic one.

$$\begin{aligned}
 \varepsilon &= \varepsilon^e + \varepsilon^p \\
 \sigma &= E \cdot \varepsilon^e \\
 \sigma &= E \cdot (\varepsilon - \varepsilon^p)
 \end{aligned} \tag{2.1}$$

Eq. 2.1 describes the stress-strain relation with plastic behavior for the elastic-perfectly plastic uniaxial loading conditions.

Next we will present the mathematical structure of two classical phenomenological isotropic hardening model (?).

2.1.1 “Exact” Models

2.1.1.1 Isotropic hardening model

Formulation of this constitutive law is based on classical theory of plasticity and is thus developed as follows:

Helmholtz free energy potential The general format of associated dissipative models starts from the Helmholtz free energy expansion. Expanding the free energy into two terms, an elastic and a plastic one, we have

$$\rho \cdot \phi(\varepsilon^e, \xi) =: \underbrace{\frac{1}{2} \varepsilon^e \cdot E \cdot \varepsilon^e}_{\text{Elastic}} + \underbrace{\frac{1}{2} \cdot \xi \cdot H \cdot \xi}_{\text{Plastic}}$$

where ρ is density, $\phi(\varepsilon^e, \xi)$ the Helmholtz free energy potential function in terms of elastic strain ε^e and a strain-like internal variable ξ , E elastic modulus, and H plastic modulus.

The elastic and plastic moduli E and H are shown in Fig. 2.3. The elastic and plastic stress, σ and q respectively, are (by definition) given by

$$\text{Linear elasticity: } \sigma := \rho \cdot \frac{\partial \phi}{\partial \varepsilon^e} = E \cdot \varepsilon^e \tag{2.2}$$

$$\text{Plasticity: } q := \rho \cdot \frac{\partial \phi}{\partial \xi} = H \cdot \xi \tag{2.3}$$

where, q is the initial yield stress σ_y at $\xi = 0$.

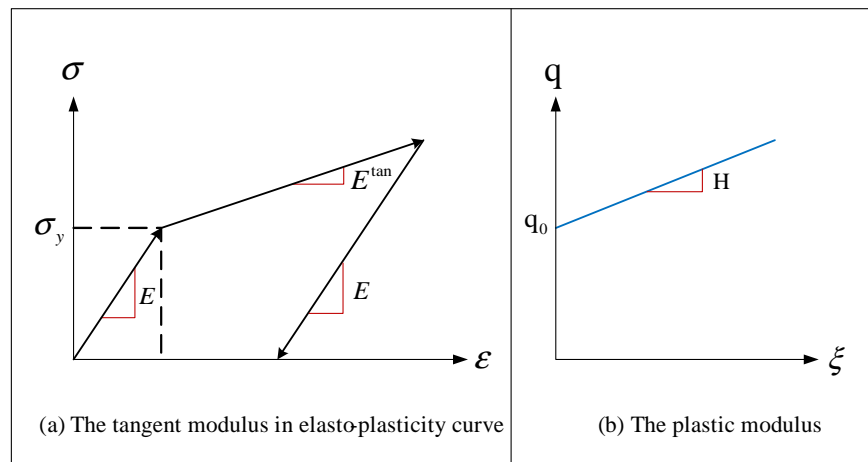


Figure 2.3: The tangent modulus for isotropic hardening model

Assuming E and H constant, we can rewrite Eq. 2.2 and Eq. 2.3 in rate form as

$$\dot{\sigma} = E \cdot \dot{\varepsilon}^e = E \cdot (\dot{\varepsilon} - \dot{\varepsilon}^p) \quad (2.4)$$

$$\dot{q} = H \cdot \dot{\xi} \quad (2.5)$$

Yield function The yield function (in 1D) is defined as

$$f(\sigma, q) := |\sigma| - q$$

Thus if $f(\sigma, q) < 0$, then the stress is within the elastic domain. Alternatively, if $f(\sigma, q) = 0$, the stress has reached its plastic limit. Fig. 2.4 shows the evolution of the elastic domain. First the strain reaches yielding

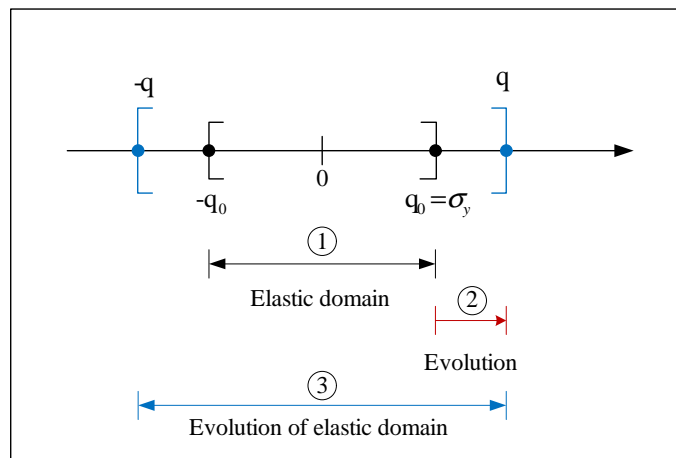


Figure 2.4: The evolution of elastic domain in isotropic hardening model

($q_0 = \sigma_y$), and then at that point further increase in strain results in an expansion of q this will in turn expand the elastic domain.

Evolution equation We must define two of them¹:

¹We have established a yield criterion. When the stress is inside the yield surface, it is elastic, Hooke's law is applicable, strains are recoverable, and there is no dissipation of energy. However, when the load on the structure pushes the stress tensor to be beyond the yield surface, the stress tensor locks up on the yield surface, and the structure deforms plastically (if the material exhibits

1. Plastic flow for $\dot{\epsilon}^P$ defined by

$$\dot{\epsilon}^P := \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \quad (2.6)$$

where, $\dot{\gamma}$ is a plastic multiplier or consistency parameter (as we shall see later, this term will ultimately drop out from E^{tan} , and g is the plastic potential function in terms of σ and q).

In isotropic hardening models, we assume $g(\sigma, q) = f(\sigma, q)$, thus $\frac{\partial g}{\partial \sigma} = \text{sign}(\sigma)$ where $\text{sign}(\sigma)$ is 1 if $\sigma \geq 0$ or -1 if $\sigma < 0$. Then, we rewrite Eq. 2.6 as

$$\dot{\epsilon}^P = \dot{\gamma} \cdot \text{sign}(\sigma)$$

or

$$\dot{\gamma} = |\dot{\epsilon}^P| \geq 0$$

2. The second plastic flow is associated with the strain-like internal state variable associated with plasticity.

$$\dot{\xi} := \dot{\gamma} \cdot h(\sigma, q)$$

where, $h(\sigma, q)$ is a dimensionless hardening function. For simplicity and under the assumption of elasto-plastic hardening material with bilinear curve,

$$h(\sigma, q) = - \frac{\partial g}{\partial q} = 1$$

Therefore, we can rewrite Eq. 2.4 and Eq. 2.5 as follows

$$\begin{aligned} \dot{\sigma} &= E \cdot \left(\dot{\epsilon} - \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \right) \\ &= E \cdot \left(\dot{\epsilon} - \dot{\gamma} \cdot \text{sign}(\sigma) \right) \end{aligned} \quad (2.7)$$

and

$$\begin{aligned} \dot{q} &= H \cdot \dot{\gamma} \cdot h(\sigma, q) \\ &= H \cdot \dot{\gamma} \end{aligned} \quad (2.8)$$

Kuhn-Tucker conditions and consistency condition In the elastic regime, the yield function f must remain negative and the plastic multiplier is zero. On the other hand, during plastic flow the yield function f must be zero while plastic multiplier is positive. This is succinctly expressed by the so-called Kuhn-Tucker condition,

$$\text{Elastic loading and unloading} \quad : f < 0, \quad \dot{\gamma} = 0 \quad (2.9)$$

$$\text{Plastic loading} \quad : f = 0, \quad \dot{\gamma} > 0 \quad (2.10)$$

Eq. 2.9 and Eq. 2.10 can be combined for the general case of $f \leq 0$ and $\dot{\gamma} \geq 0$, then

$$f \cdot \dot{\gamma} = 0 \quad (2.11)$$

Consistency Condition: During plastic loading the stress path is constrained to move along the yield surface, thus this consistency condition precludes us from going beyond the yield surface and is mathematically expressed as $\dot{f}(\sigma, q) = 0$, or

$$\begin{aligned} \dot{f}(\sigma, q) &= \frac{\partial f}{\partial \sigma} \cdot \frac{\partial \sigma}{\partial t} + \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial t} \\ &= \frac{\partial f}{\partial \sigma} \cdot \dot{\sigma} + \frac{\partial f}{\partial q} \cdot \dot{q} = 0 \end{aligned} \quad (2.12)$$

Tangent Modulus Finally, we need to evaluate tangent modulus E^{tan} such that $\dot{\sigma} = E^{tan} \dot{\epsilon}$. Substituting Eq. 2.7

hardening as opposed to elastic-perfectly plastic response, then the yield surface expands or moves with the stress point still on the yield surface). At this point, the crucial question is what will be direction of the plastic flow (that is the relative magnitude of the components of ϵ^P). This question is addressed by the **flow rule**, or **normality rule**

and Eq. 2.8 into Eq. 2.12, we obtain

$$\frac{\partial f}{\partial \sigma} \cdot E \cdot \left(\dot{\varepsilon} - \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \right) + \frac{\partial f}{\partial q} \cdot H \cdot \dot{\gamma} \cdot h(\sigma, q) = 0$$

Therefore,

$$\dot{\gamma} = \frac{\frac{\partial f}{\partial \sigma} \cdot E \cdot \dot{\varepsilon}}{\frac{\partial f}{\partial \sigma} \cdot E \cdot \frac{\partial g}{\partial \sigma} - \frac{\partial f}{\partial q} \cdot H \cdot h(\sigma, q)} \quad (2.13)$$

Substituting Eq. 2.13 into Eq. 2.7, we obtain an explicit expression for the incremental stress,

$$\begin{aligned} \dot{\sigma} &= E \cdot \left(\dot{\varepsilon} - \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \right) \\ &= \left(E - \frac{\frac{\partial f}{\partial \sigma} \cdot E^2 \cdot \frac{\partial g}{\partial \sigma}}{\frac{\partial f}{\partial \sigma} \cdot E \cdot \frac{\partial g}{\partial \sigma} - \frac{\partial f}{\partial q} \cdot H \cdot h(\sigma, q)} \right) \cdot \dot{\varepsilon} \\ &= E^{tan} \cdot \dot{\varepsilon} \end{aligned}$$

For elasto-plastic hardening material with bilinear curve in one dimension,

$$\dot{\gamma} = \frac{\text{sign}(\sigma) \cdot E \cdot \dot{\varepsilon}}{E + H}$$

and the tangent modulus reduces to

$$\begin{aligned} E^{tan} &= E - \frac{\text{sign}(\sigma) \cdot E^2 \cdot \text{sign}(\sigma)}{\text{sign}(\sigma) \cdot E \cdot \text{sign}(\sigma) - (-1) \cdot H \cdot (1)} \\ &= E - \frac{E^2}{E + H} \\ &= \frac{E \cdot H}{E + H} \end{aligned}$$

2.1.1.2 Combined isotropic and kinematic hardening model

We describe isotropic and kinematic hardening model. Finally, we summarize the major equations governing plasticity.

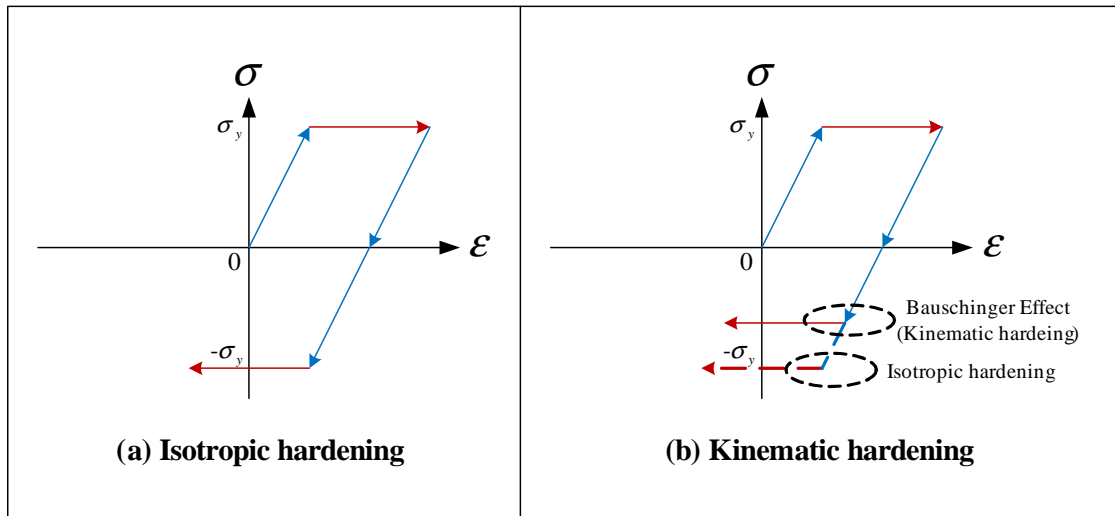


Figure 2.5: Isotropic and kinematic hardening plasticity

Yield surface of isotropic hardening plasticity is symmetric about the origin at $\sigma = 0$, Fig. 2.5(a), whereas for kinematic hardening plasticity the yield surface is unsymmetric, Fig. 2.5(b). This lack of symmetry is caused by the classical Bauschinger effect.

Helmholtz free energy potential The Helmholtz free energy function is now enriched by a set of strain-like internal state variables for combined isotropic and kinematic hardening (as opposed to a single value previously):

$$\rho \cdot \phi(\varepsilon^e, \boldsymbol{\xi}) = \underbrace{\frac{1}{2} \cdot \varepsilon^e \cdot E \cdot \varepsilon^e}_{\text{Elastic}} + \underbrace{\frac{1}{2} \cdot \boldsymbol{\xi}^T \cdot \mathbf{H} \cdot \boldsymbol{\xi}}_{\text{Plastic}}$$

where, $\boldsymbol{\xi}$ is a strain-like vector of internal state variables, and \mathbf{H} is the hardening matrix.

$$\boldsymbol{\xi} = [\xi^{iso}, \xi^{kin}]^T$$

$$\mathbf{H} = \begin{bmatrix} H^{iso} & 0 \\ 0 & H^{kin} \end{bmatrix}$$

where, ξ^{iso} is a strain-like internal state variable of isotropic plasticity, ξ^{kin} a strain-like internal state variable of kinematic plasticity, H^{iso} the isotropic plastic modulus, and H^{kin} the kinematic plastic modulus. The thermodynamically stress-like vector of internal state variables is thus given by

$$\mathbf{q}^\xi = \mathbf{H} \cdot \boldsymbol{\xi}$$

$$= [H^{iso} \cdot \xi^{iso}, H^{kin} \cdot \xi^{kin}]^T$$

$$= [q^{iso}, q^{kin}]^T$$

where, q^{iso} is an isotropic stress-like internal state variable and q^{kin} a kinematic stress-like internal state variable. Therefore, the rate form of stress-like vector of internal state variables becomes

$$\dot{\mathbf{q}}^\xi = \mathbf{H} \cdot \dot{\boldsymbol{\xi}}$$

$$= [H^{iso} \cdot \dot{\xi}^{iso}, H^{kin} \cdot \dot{\xi}^{kin}]^T \quad (2.14)$$

$$= [\dot{q}^{iso}, \dot{q}^{kin}]^T$$

Yield function Having stress-like internal state variable q^{iso} , we redefine the yield function as

$$f(\sigma, \mathbf{q}^\xi) := |\sigma - q^{kin}| - q^{iso}$$

As mentioned above, if $f(\sigma, \mathbf{q}^\xi) < 0$, the stress is in the elastic domain, otherwise, if $f(\sigma, \mathbf{q}^\xi) = 0$, the stress reaches plasticity.

Evolution Equations Again, we consider two of them:

1. Plastic flow of ε^p is again given by

$$\dot{\varepsilon}^p := \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \quad (2.15)$$

and we again assume that $g(\sigma, q^{iso})$ to be the same as $f(\sigma, q^{iso})$.

Fig. 2.6 depicts the evolution of the elastic domain for $H^{iso} = 0$ (isotropic perfectly plastic) and $H^{kin} > 0$. The size of elastic domains is the same after evolution.

2. The evolution of $\dot{\boldsymbol{\xi}}$ is defined by

$$\dot{\boldsymbol{\xi}} := \dot{\gamma} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi) \quad (2.16)$$

where, $\mathbf{h}(\sigma, \mathbf{q}^\xi)$ is a dimensionless hardening function.

$$\mathbf{h}(\sigma, \mathbf{q}^\xi) = -\frac{\partial g}{\partial \mathbf{q}^\xi}$$

$$= \left[-\frac{\partial g}{\partial q^{iso}}, -\frac{\partial g}{\partial q^{kin}} \right]^T$$

$$= [1, \text{sign}(\sigma - q^{kin})]^T$$

Combining Eq. 2.14 and Eq. 2.16,

$$\dot{\mathbf{q}}^\xi = \mathbf{H} \cdot \dot{\boldsymbol{\xi}} = \mathbf{H} \cdot \dot{\gamma} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi) \quad (2.17)$$

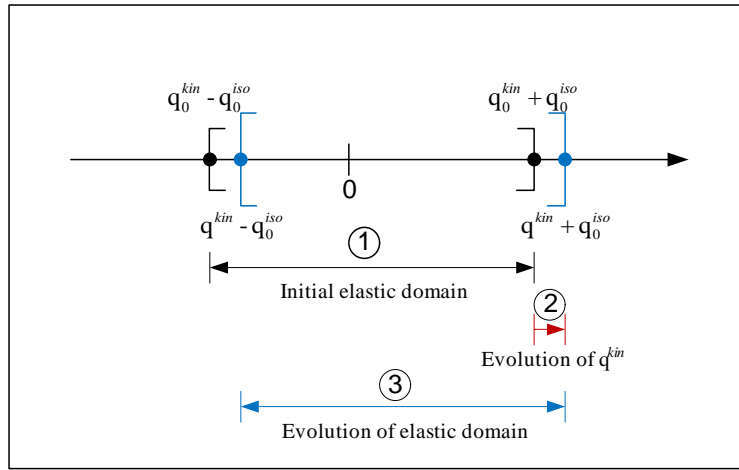


Figure 2.6: The evolution of elastic domain in isotropic and kinematic hardening model for $H^{iso} = 0$ (isotropic perfectly plastic) and $H^{kin} > 0$

Tangent Modulus To determine E^{tan} , we again invoke the consistency condition given by Eq. 2.12 and where $\dot{\mathbf{q}}$ has now been replaced by $\dot{\mathbf{q}}^\xi$

$$\dot{f}(\sigma, \mathbf{q}^\xi) = \frac{\partial f}{\partial \sigma} \cdot \dot{\sigma} + \frac{\partial f}{\partial \mathbf{q}^\xi} \cdot \dot{\mathbf{q}}^\xi = 0 \quad (2.18)$$

Substituting Eq. 2.7 and Eq. 2.17 into Eq. 2.18,

$$\frac{\partial f}{\partial \sigma} \cdot E \cdot \left(\dot{\varepsilon} - \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \right) + \frac{\partial f}{\partial \mathbf{q}^\xi} \cdot \mathbf{H} \cdot \dot{\gamma} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi) = 0 \quad (2.19)$$

$\dot{\gamma}$ is determined from Eq. 2.19

$$\dot{\gamma} = \frac{\frac{\partial f}{\partial \sigma} \cdot E \cdot \dot{\varepsilon}}{\frac{\partial f}{\partial \sigma} \cdot E \cdot \frac{\partial g}{\partial \sigma} - \frac{\partial f}{\partial \mathbf{q}^\xi} \cdot \mathbf{H} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi)} \quad (2.20)$$

From Eq. 2.7,

$$\begin{aligned} \dot{\sigma} &= E \cdot \left(\dot{\varepsilon} - \dot{\gamma} \cdot \frac{\partial g}{\partial \sigma} \right) \\ &= \left(E - \frac{\frac{\partial f}{\partial \sigma} \cdot E^2 \cdot \frac{\partial g}{\partial \sigma}}{\frac{\partial f}{\partial \sigma} \cdot E \cdot \frac{\partial g}{\partial \sigma} - \frac{\partial f}{\partial \mathbf{q}^\xi} \cdot \mathbf{H} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi)} \right) \cdot \dot{\varepsilon} \\ &= E^{tan} \cdot \dot{\varepsilon} \end{aligned}$$

Thus, for elasto-plastic hardening material with bilinear curve, Eq. 2.20 reduces to:

$$\dot{\gamma} = \frac{\text{sign}(\sigma - q^{kin}) \cdot E \cdot \dot{\varepsilon}}{E + H^{iso} + H^{kin}}$$

Finally, we determine the tangent modulus, E^{tan} ,

$$\begin{aligned}
 E^{tan} &= E - \frac{\frac{\partial f}{\partial \sigma} \cdot E^2 \cdot \frac{\partial q}{\partial \sigma}}{\frac{\partial f}{\partial \sigma} \cdot E \cdot \frac{\partial q}{\partial \sigma} - \frac{\partial f}{\partial \mathbf{q}^\xi} \cdot \mathbf{H} \cdot \mathbf{h}(\sigma, \mathbf{q}^\xi)} \\
 &= E - \frac{\text{sign}(\sigma - q^{kin})^2 \cdot E^2}{\text{sign}(\sigma - q^{kin})^2 \cdot E - [-1, -\text{sign}(\sigma - q^{kin})] \begin{bmatrix} H^{iso} & 0 \\ 0 & H^{kin} \end{bmatrix} \begin{Bmatrix} 1 \\ \text{sign}(\sigma - q^{kin}) \end{Bmatrix}} \\
 &= \frac{E \cdot (H^{iso} + H^{kin})}{E + H^{iso} + H^{kin}}
 \end{aligned}$$

2.1.1.3 Determination for isotropic and kinematic hardening parameters

Fig. 2.7 illustrates the procedure to determine uniaxial stress σ and tangent modulus E^{tan} from uniaxial strain ε .

2.1.2 Heuristic Models

2.1.2.1 Simplified Bilinear model with isotropic hardening

Whereas the proceeding model is rooted in plasticity theory, its implementation may be problematic. Alternatively, a phenomenologically similar model can be derived based on (?) and as implemented in (?).

2.1.2.1.1 Stress-strain relation Instead of determining the E^{tan} with H in Sec. 2.1.1.1, the simplified bilinear model computes it through a strain-hardening coefficient b which is the ratio of the post-yield tangent modulus E^{tan} and the initial elastic modulus E , and only considers isotropic hardening with Eq. 2.21 and Eq. 2.22 (?). Fig. 2.8 describes this simplified bilinear model.

$$E^{tan} = b \cdot E$$

To account for the evolution of elastic domain in isotropic hardening, a stress shift σ_Δ is determined as follow:

- If the incremental strain $\Delta\varepsilon$ changes a positive value into a negative one:

$$\begin{aligned}
 \Delta^N &= 1 + a_1 \cdot \left(\frac{\varepsilon^{max} - \varepsilon^{min}}{2 \cdot a_2 \cdot \varepsilon_y} \right)^{0.8} \\
 \sigma_\Delta &= \Delta^N \cdot \sigma_y \cdot (1 - b)
 \end{aligned} \tag{2.21}$$

- If the incremental strain $\Delta\varepsilon$ changes a negative value into a positive one:

$$\begin{aligned}
 \Delta^P &= 1 + a_3 \cdot \left(\frac{\varepsilon^{max} - \varepsilon^{min}}{2 \cdot a_4 \cdot \varepsilon_y} \right)^{0.8} \\
 \sigma_\Delta &= \Delta^P \cdot \sigma_y \cdot (1 - b)
 \end{aligned} \tag{2.22}$$

where, a_1 and a_3 are isotropic hardening parameter which reflect an increase of the compression yield envelope through a fraction of the yield strength after a plastic strain $a_2 \cdot \frac{\sigma_y}{E}$, and tension yield envelope as a fraction of the yield strength after a plastic strain of $a_4 \cdot \frac{\sigma_y}{E}$. a_2 and a_4 are isotropic hardening parameter with respect to a_1 and a_3 , and ε_{max} and ε_{min} are the strain at the maximum and minimum strain reversal point. Limiting factor of this model is that a_1 , a_2 , a_3 and a_4 must be determined through curve fitting of the model with experimental results. Default values are $a_1 = 0$, $a_2 = 55$, $a_3 = 0$, and $a_4 = 55$ in (?).

2.1.2.1.2 Determination for simplified bilinear model Fig. 2.9 shows the procedure to obtain uniaxial stress σ and tangent modulus E^{tan} from uniaxial strain ε .

2.1.2.2 Giuffre-Menegotto-Pinto Model Modified by Filippou et al.

This section is based on doctoral dissertation of ?.

2.1.2.2.1 Stress-strain relationship The reinforcing steel stress-strain behavior is described by the nonlinear model of ?, as modified by ?, to include isotropic strain hardening. This model introduces smooth curve to describe similar behavior to experimental one instead of bilinear curve.

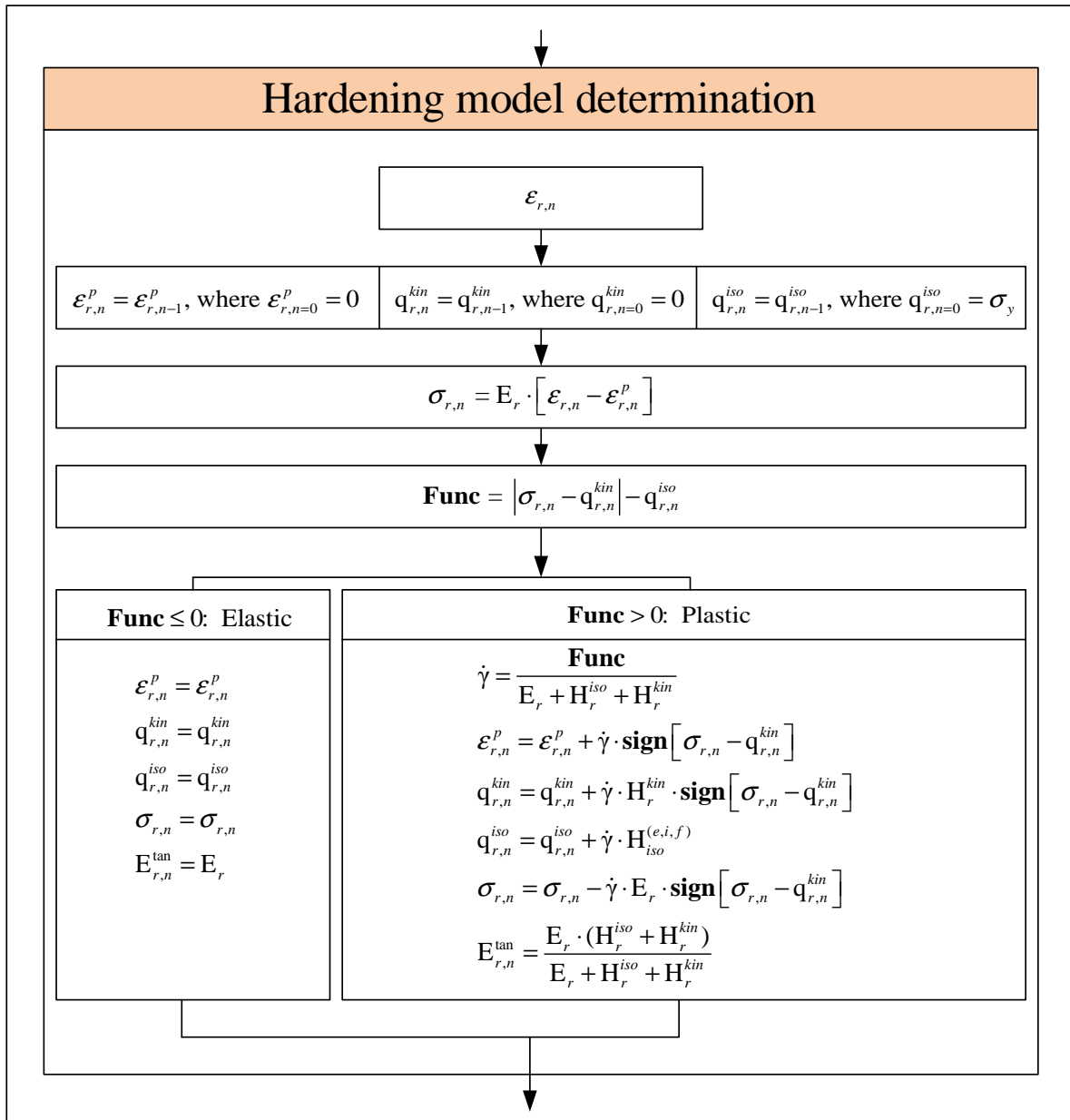


Figure 2.7: Determination for isotropic and kinematic hardening model

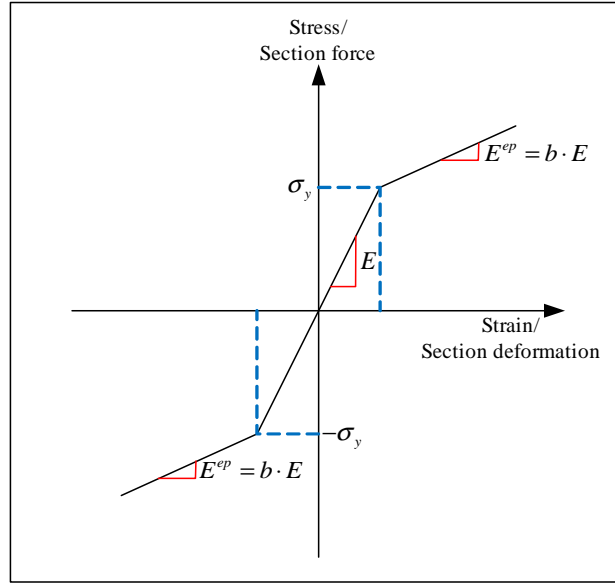


Figure 2.8: Bilinear model

The model presented in ? starts from the following empirical form of the stress-strain relation which is found to be close enough to experimentally determined ones.

$$\sigma^* = b \cdot \varepsilon^* + \frac{(1-b) \cdot \varepsilon^*}{(1 + \varepsilon^{*R})^{1/R}} \quad (2.23)$$

where,

$$\begin{aligned} \varepsilon^* &= \frac{\varepsilon - \varepsilon_{rev}}{\varepsilon_0 - \varepsilon_{rev}} \\ \sigma^* &= \frac{\sigma - \sigma_{rev}}{\sigma_0 - \sigma_{rev}} \end{aligned} \quad (2.24)$$

The tangent modulus E^{tan} is obtained by differentiating Eq. 2.23 and 2.24,

$$E^{tan} = \frac{d\sigma}{d\varepsilon} = \frac{\sigma_0 - \sigma_{rev}}{\varepsilon_0 - \varepsilon_{rev}} \cdot \frac{d\sigma^*}{d\varepsilon^*} \quad (2.25)$$

where,

$$\frac{d\sigma^*}{d\varepsilon^*} = b + \left[\frac{1-b}{(1 + \varepsilon^{*R})^{1/R}} \right] \cdot \left[1 - \frac{\varepsilon^{*R}}{1 + \varepsilon^{*R}} \right] \quad (2.26)$$

In Fig. 2.10, Eq. 2.24 represents a curved transition from a straight line asymptote with slope E (a) to another asymptote with slope E^{tan} (b), σ_{rev} and ε_{rev} are the stress and strain at the point of strain reversal (point A), which also forms the origin of the asymptote with slope E (a), and σ_0 and ε_0 are the stress and strain at the point of intersection of the two asymptotes (point B).

b is the strain hardening ratio between slope E^{tan} and E , and R is a parameter that influences the curvature of the transition curve between the two asymptotes and permits a good representation of the Bauschinger effect. As indicated in Fig. 2.10, σ_0 , ε_0 , σ_{rev} and ε_{rev} are updated after each strain reversal.

In Fig. 2.11, R is dependent on the absolute strain difference between the current asymptote intersection point (point B) and the previous maximum or minimum strain reversal point (point C) depending on whether the current strain is increasing or decreasing, respectively. There are two reported expression for $R(\xi)$:

- Menegotto-Pinto original model (?),

$$R(\xi) = R_0 - \frac{cR_1 \cdot \xi}{cR_2 + \xi} \quad (2.27)$$

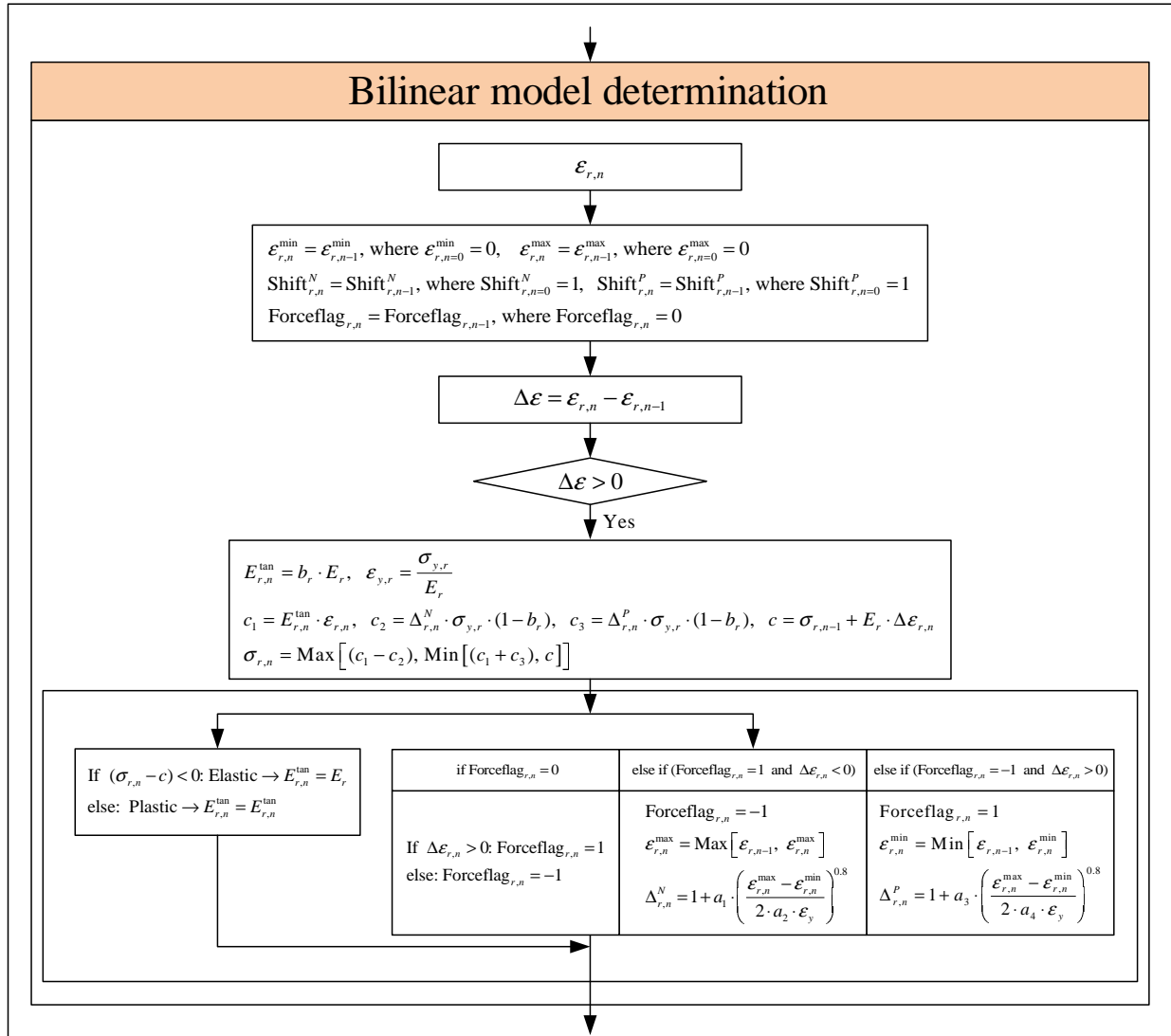


Figure 2.9: Determination for simplified bilinear model

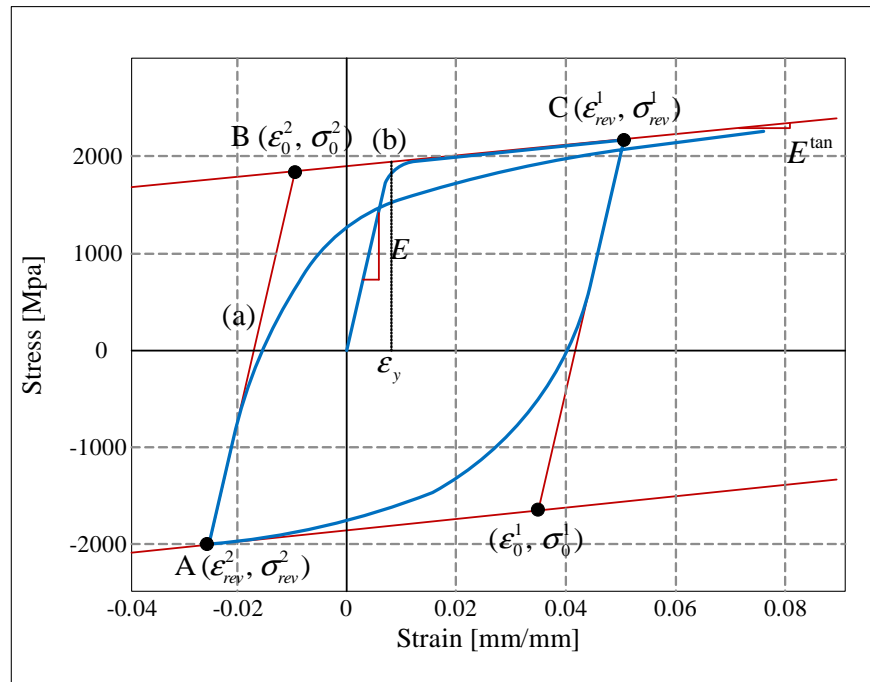


Figure 2.10: Menegotto-Pinto steel model
(?)

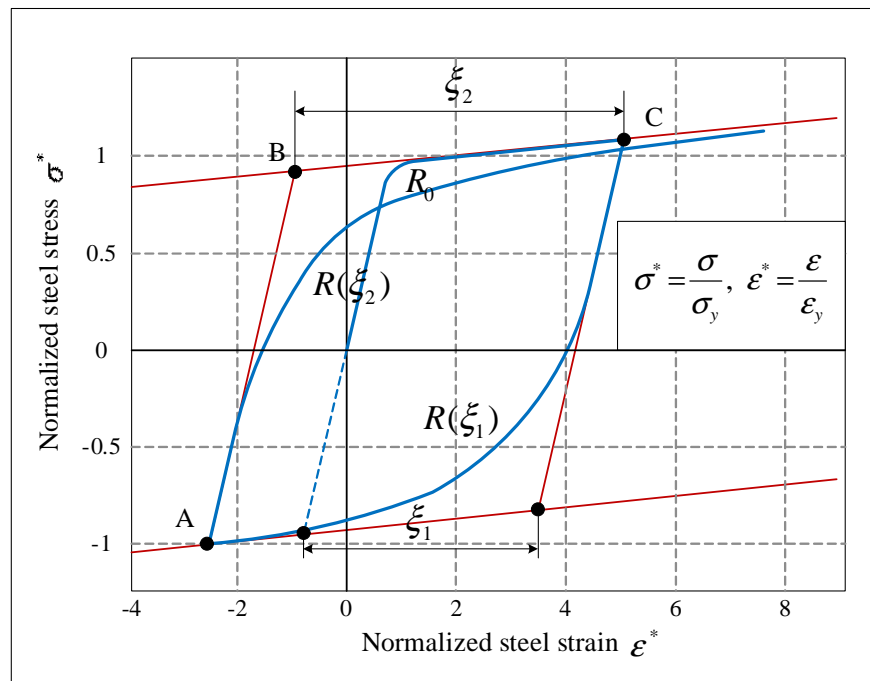


Figure 2.11: Definition of curvature parameter R in Menegotto-Pinto steel model
(?)

- Steel12 in OpenSees (?),

$$R(\xi) = R_0 \left(1 - \frac{cR_1 \cdot \xi}{cR_2 + \xi} \right) \quad (2.28)$$

where, R_0 is the value of the parameter R during first loading, and cR_1 and cR_2 are experimentally determined parameters to be defined together with R_0 . ξ can be expressed as

$$\xi = \left| \frac{\varepsilon^m - \varepsilon_0}{\varepsilon_y} \right| \quad (2.29)$$

where, ε^m is the strain at the previous maximum or minimum strain reversal point depending on whether the current strain is increasing or decreasing, respectively. ε_0 is the strain at the current intersection point of the two asymptotes.

As shown in Fig. 2.10, both ε^m and ε_0 lie along the same asymptote and ε_y is the initial yield strain. Fig. 2.11 shows how ξ is updated following a strain reversal.

Some clarification is needed in connection with the set of rules for unloading and reloading which are implied by Eq. 2.23 to 2.29, allowing for a generalized load history. If the analytical model had a memory extending over all previous branches of the stress-strain history, then it would allow for the resumption of the previous reloading branch, as soon as the new reloading curve reached it. However, this would require that the model store all necessary information to retrace all previous incomplete reloading curves, and this is clearly impractical from a computational standpoint. Memory of the past stress-strain history is therefore limited to a predefined number of parameters, which in the present model are:

1. stress and strain at the last state of the model
2. stress and strain at the last state reversal point
3. stress and strain at the last asymptote intersection point
4. flag indication whether the last branch is ascending or descending
5. strain at the previous minimum strain reversal point
6. strain at the previous maximum strain reversal point

As a result of these restrictions reloading following partial unloading does not hit the original curve following unloading started, but, instead, continues on the new reloading curve until reaching the monotonic envelope. However, the discrepancy between the analytical model and the actual behavior is typically very small, ?.

The above implementation of the model corresponds to its simplest form, as proposed by ?: elastic and yield asymptotes are assumed to be straight lines, the position of the limiting asymptotes corresponding to the yield surface is assumed to be fixed at all times and the slope E remains constant, Fig. 2.10.

In spite of the simplicity in formulation, the model is capable of reproducing well experimental results and its major drawback stems from its failure to allow for isotropic hardening. To account for this effect ? proposed a shift of σ_0 and ε_0 in the linearly yield asymptote as follows:

- If the incremental strain $\Delta\varepsilon$ changes a positive value to a negative value:

$$\begin{aligned} \Delta^N &= 1 + a_1 \cdot \left(\frac{\varepsilon^{max} - \varepsilon^{min}}{2 \cdot a_2 \cdot \varepsilon_y} \right)^{0.8} \\ \varepsilon_0 &= \frac{-\sigma_y \cdot \Delta^N + E^{tan} \cdot \varepsilon_y \cdot \Delta^N - \sigma_{rev} + E \cdot \varepsilon_{rev}}{E - E^{tan}} \\ \sigma_0 &= -\sigma_y \cdot \Delta^N + E^{tan} \cdot (\varepsilon_0 + \varepsilon_y \cdot \Delta^N) \end{aligned} \quad (2.30)$$

- If the incremental strain $\Delta\varepsilon$ changes a negative value to a positive value,

$$\begin{aligned} \Delta^P &= 1 + a_3 \cdot \left(\frac{\varepsilon^{max} - \varepsilon^{min}}{2 \cdot a_4 \cdot \varepsilon_y} \right)^{0.8} \\ \varepsilon_0 &= \frac{\sigma_y \cdot \Delta^P - E^{tan} \cdot \varepsilon_y \cdot \Delta^P - \sigma_{rev} + E \cdot \varepsilon_{rev}}{E - E^{tan}} \\ \sigma_0 &= \sigma_y \cdot \Delta^P + E^{tan} \cdot (\varepsilon_0 - \varepsilon_y \cdot \Delta^P) \end{aligned} \quad (2.31)$$

where, a_1 and a_3 are isotropic hardening parameter which reflect an increase of the compression yield envelope through a fraction of the yield strength after a plastic strain $a_2 \cdot \frac{\sigma_y}{E}$, and tension yield envelope as a fraction of the yield strength after a plastic strain of $a_4 \cdot \frac{\sigma_y}{E}$. a_2 and a_4 are isotropic hardening parameter with respect to a_1 and a_3 , and ε_{max} and ε_{min} are the strain at the maximum and minimum strain reversal point. Limiting factor of this model is that a_1 , a_2 , a_3 and a_4 must be determined through curve fitting of the model with experimental results. Default values are $a_1 = 0$, $a_2 = 55$, $a_3 = 0$, and $a_4 = 55$ in (?).

2.1.2.2.2 Determination for modified Giuffre-Menegotto-Pinto Model Fig. 2.12 to 2.15 show the procedure to obtain uniaxial stress σ and tangent modulus E^{tan} from uniaxial strain ε .

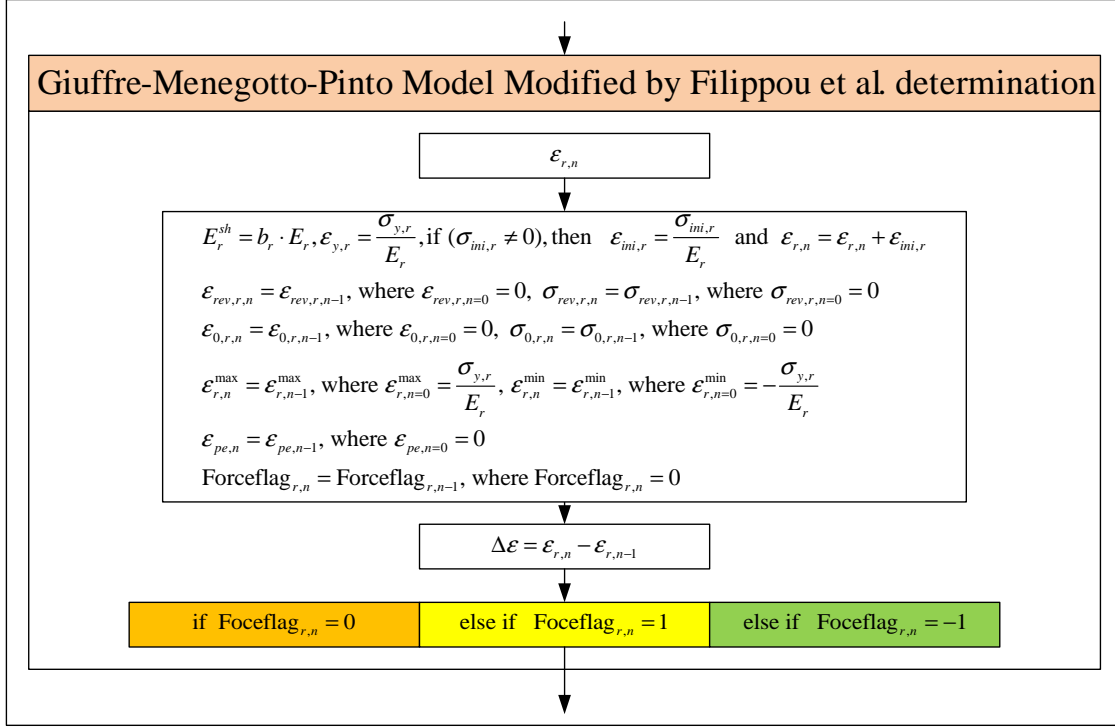


Figure 2.12: Determination (1) for Giuffre-Menegotto-Pinto Model Modified by Filippou et al.

2.2 Concrete Models

2.2.1 Heuristic Models

2.2.1.1 Modified Kent And Park Model

This section is based on doctoral dissertation of (?).

2.2.1.1.1 Stress-strain relation The concrete model describes the concrete stress-strain relation under an arbitrary cyclic strain history. In particular, the model implemented takes into account four important factors:

1. effect of concrete confinement on the monotonic envelope curve in compression
2. successive degradation of stiffness of both the unloading and reloading curves, for increasing values of compressive strain
3. effect of tension stiffening
4. hysteretic response under cyclic loading in compression

if Foceflag _{r,n} = 0		
if Δε = 0	else if Δε < 0	else if Δε > 0
$\varepsilon_{rev,r,n} = 0$	$\varepsilon_{rev,r,n} = 0$	$\varepsilon_{rev,r,n} = 0$
$\sigma_{rev,r,n} = 0$	$\sigma_{rev,r,n} = 0$	$\sigma_{rev,r,n} = 0$
$\varepsilon_{0,r,n} = 0$	$\varepsilon_{0,r,n} = \sigma_{y,r} / E_r$	$\varepsilon_{0,r,n} = -\sigma_{y,r} / E_r$
$\sigma_{0,r,n} = 0$	$\sigma_{0,r,n} = \sigma_{y,r}$	$\sigma_{0,r,n} = -\sigma_{y,r}$
$\varepsilon_{r,n}^{\max} = \sigma_{y,r} / E_r$	$\varepsilon_{r,n}^{\max} = \sigma_{y,r} / E_r$	$\varepsilon_{r,n}^{\max} = \sigma_{y,r} / E_r$
$\varepsilon_{r,n}^{\min} = -\sigma_{y,r} / E_r$	$\varepsilon_{r,n}^{\min} = -\sigma_{y,r} / E_r$	$\varepsilon_{r,n}^{\min} = -\sigma_{y,r} / E_r$
Forceflag _{r,n} = 0	$\varepsilon_{pe,n} = \varepsilon_{r,n}^{\min}$	$\varepsilon_{pe,n} = \varepsilon_{r,n}^{\max}$
$\sigma_{r,n} = \sigma_{ini,r}$	Forceflag _{r,n} = 1	Forceflag _{r,n} = -1
$E_{r,n}^{\tan} = E_r$	$\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $	$\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $
	$R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$	$R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$
	$\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$	$\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$
	$c_1 = 1 + \dot{\varepsilon} ^R$	$c_1 = 1 + \dot{\varepsilon} ^R$
	$c_2 = c_1^{1/R}$	$c_2 = c_1^{1/R}$
	$\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$	$\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$
	$\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$	$\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$
	$E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$	$E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$
	$E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$	$E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$

Figure 2.13: Determination (2) for Giuffre-Menegotto-Pinto Model Modified by Filippou et al.

else if Forceflag _{r,n} = 1	
if Δε < 0	else if Δε > 0
$\varepsilon_{rev,r,n}, \sigma_{rev,r,n}$ $\Delta^N = 1 + a_1 \cdot \left(\frac{\varepsilon_{r,n}^{\max} - \varepsilon_{r,n}^{\min}}{2 \cdot a_2 \cdot \varepsilon_y} \right)^{0.8}$ $\varepsilon_{0,r,n} = \frac{-\sigma_{y,r} \cdot \Delta^N + E_r^{sh} \cdot \varepsilon_{y,r} \cdot \Delta^N - \sigma_{rev,r,n} + E_r \cdot \varepsilon_{rev,r,n}}{E_r - E_r^{sh}}$ $\sigma_{0,r,n} = -\sigma_{y,r} \cdot \Delta^N + E_r^{sh} \cdot (\varepsilon_{0,r,n} + \varepsilon_{y,r} \cdot \Delta^N)$ $\varepsilon_{r,n}^{\max} = \text{Max}[\varepsilon_{r,n}^{\max}, \varepsilon_{r,n-1}]$ $\varepsilon_{r,n}^{\min} = \text{Min}[\varepsilon_{r,n}^{\min}, \varepsilon_{r,n-1}]$ $\varepsilon_{pe,n} = \varepsilon_{r,n}^{\min}$ $\text{Forceflag}_{r,n} = -1$ $\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $ $R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$ $\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$ $c_1 = 1 + \dot{\varepsilon} ^R$ $c_2 = c_1^{1/R}$ $\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$ $\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$ $E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$ $E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$	$\varepsilon_{rev,r,n}, \sigma_{rev,r,n}, \varepsilon_{0,r,n}, \sigma_{0,r,n}$ $\varepsilon_{r,n}^{\max} = \text{Max}[\varepsilon_{r,n}^{\max}, \varepsilon_{r,n-1}]$ $\varepsilon_{r,n}^{\min} = \text{Min}[\varepsilon_{r,n}^{\min}, \varepsilon_{r,n-1}]$ $\varepsilon_{pe,n} = \varepsilon_{r,n}^{\max}$ $\text{Forceflag}_{r,n} = 1$ $\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $ $R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$ $\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$ $c_1 = 1 + \dot{\varepsilon} ^R$ $c_2 = c_1^{1/R}$ $\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$ $\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$ $E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$ $E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$

Figure 2.14: Determination (3) for Giuffre-Menegotto-Pinto Model Modified by Filippou et al.

else if Foceflag _{r,n} = -1	
if Δε > 0	else if Δε < 0
$\varepsilon_{rev,r,n}, \sigma_{rev,r,n}$ $\Delta^p = 1 + a_3 \cdot \left(\frac{\varepsilon_{r,n}^{\max} - \varepsilon_{r,n}^{\min}}{2 \cdot a_4 \cdot \varepsilon_y} \right)^{0.8}$ $\varepsilon_{0,r,n} = \frac{\sigma_{y,r} \cdot \Delta^p - E_r^{sh} \cdot \varepsilon_{y,r} \cdot \Delta^p - \sigma_{rev,r,n} + E_r \cdot \varepsilon_{rev,r,n}}{E_r - E_r^{sh}}$ $\sigma_{0,r,n} = \sigma_{y,r} \cdot \Delta^p + E_r^{sh} \cdot (\varepsilon_{0,r,n} - \varepsilon_{y,r} \cdot \Delta^p)$ $\varepsilon_{r,n}^{\max} = \text{Max}[\varepsilon_{r,n}^{\max}, \varepsilon_{r,n-1}]$ $\varepsilon_{r,n}^{\min} = \text{Min}[\varepsilon_{r,n}^{\min}, \varepsilon_{r,n-1}]$ $\varepsilon_{pe,n} = \varepsilon_{r,n}^{\max}$ $\text{Forceflag}_{r,n} = 1$ $\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $ $R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$ $\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$ $c_1 = 1 + \dot{\varepsilon} ^R$ $c_2 = c_1^{1/R}$ $\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$ $\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$ $E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$ $E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$	$\varepsilon_{rev,r,n}, \sigma_{rev,r,n}, \varepsilon_{0,r,n}, \sigma_{0,r,n}$ $\varepsilon_{r,n}^{\max} = \text{Max}[\varepsilon_{r,n}^{\max}, \varepsilon_{r,n-1}]$ $\varepsilon_{r,n}^{\min} = \text{Min}[\varepsilon_{r,n}^{\min}, \varepsilon_{r,n-1}]$ $\varepsilon_{pe,n} = \varepsilon_{r,n}^{\min}$ $\text{Forceflag}_{r,n} = -1$ $\xi = \left \frac{\varepsilon_{pe,n} - \varepsilon_{0,r,n}}{\varepsilon_{y,r}} \right $ $R = R_0 - R_0^{nratio} \cdot \frac{cR_1 \cdot \xi}{cR_2 + \xi}$ $\dot{\varepsilon} = \frac{\varepsilon_{r,n} - \varepsilon_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$ $c_1 = 1 + \dot{\varepsilon} ^R$ $c_2 = c_1^{1/R}$ $\sigma_{r,n} = b_r \cdot \dot{\varepsilon} + (1 - b_r) \cdot \frac{\dot{\varepsilon}}{c_2}$ $\sigma_{r,n} = \sigma_{r,n} \cdot (\sigma_{0,r,n} - \sigma_{rev,r,n}) + \sigma_{rev,r,n}$ $E_{r,n}^{\tan} = b_r + \frac{(1 - b_r)}{c_1 \cdot c_2}$ $E_{r,n}^{\tan} = E_{r,n}^{\tan} \cdot \frac{\sigma_{0,r,n} - \sigma_{rev,r,n}}{\varepsilon_{0,r,n} - \varepsilon_{rev,r,n}}$

Figure 2.15: Determination (4) for Giuffre-Menegotto-Pinto Model Modified by Filippou et al.

The monotonic envelope curve of concrete in compression follows the original model of ? and extended by ?. Even though more accurate and complete models have been published since, the so-called modified Kent and Park model offers a good balance between simplicity and accuracy.

Tension stiffening is the ability of concrete between cracks to resist tensile stress and contribute to the flexural stiffness of the member. Due to the discrete nature of the cracks, concrete in between cracks remains bonded to the reinforcement and, thus, contributes to the stiffness of the member. However, as the magnitude of load increases, additional cracks form at closer intervals, hence reducing the tensile stress that can be developed in the concrete. Therefore tension stiffening is gradually reduced as load is increased in the post-cracking stage. Past investigators have taken tension stiffening into account by modifying the concrete stress-strain relation such that, after reaching the tensile strength (cracking), the tensile stress reduces gradually to zero as tensile strain is increased. The gradual reduction of tensile strength is often approximated as linear, multi-linear or exponential. A similar approach with a linear rate of reduction is adopted. However, the results indicate that the application of tension stiffening over all fibers of a member can lead to significant overestimation of the ultimate strength. Rather tension stiffening is a localized phenomenon that affects the concrete in the immediate vicinity of the reinforcement. In analytical studies, only the concrete fibers within an effective area around the reinforcements are assigned tension softening. Clearly, the size of the effective area has a significant effect on tension stiffening behavior of the member.

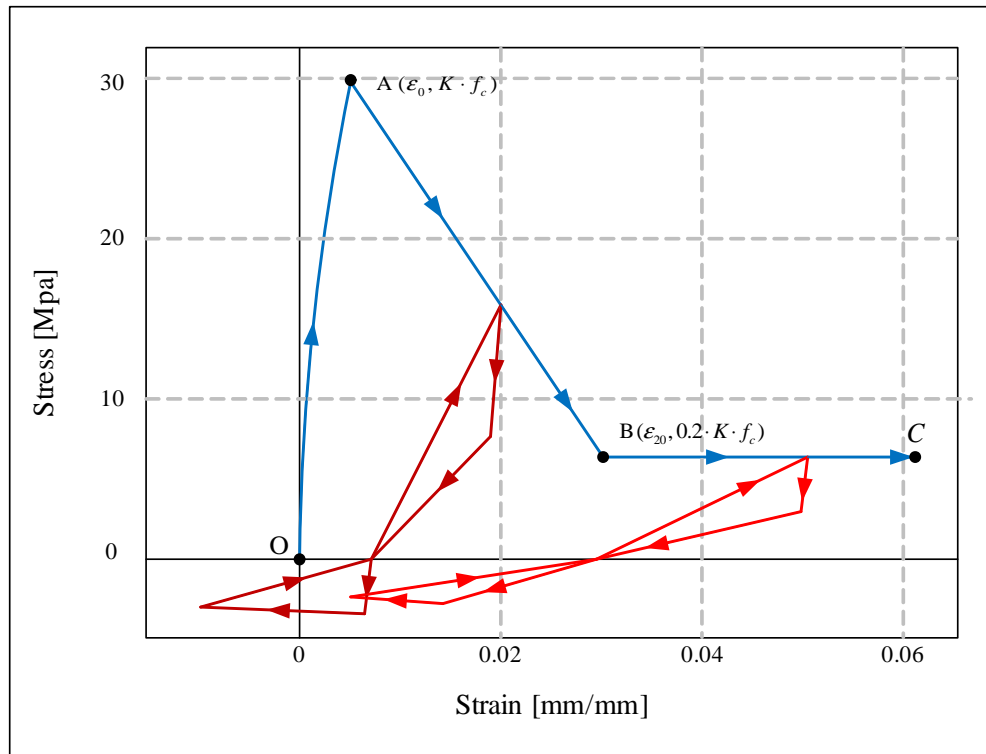


Figure 2.16: Concrete material model in compression
(?)

In the modified Kent and Park model (?) shown in Fig. 2.16, the monotonic concrete stress-strain relation in compression is empirically defined by three regions. Adopting the convention that compression is positive, the three regions are,

- Segment OA : $\varepsilon_c \leq \varepsilon_0$

$$\sigma_c = K \cdot f_c \cdot \left[2 \cdot \frac{\varepsilon_c}{\varepsilon_0} - \left(\frac{\varepsilon_c}{\varepsilon_0} \right)^2 \right]$$

From this equation, we can determine the maximum compressive strength of confined concrete:

$$f'_{c,confined} = K f'_c \quad (2.32)$$

- Segment AB: $\varepsilon_0 < \varepsilon_c \leq \varepsilon_{20}$

$$\sigma_c = K \cdot f_c \cdot [1 - Z(\varepsilon_c - \varepsilon_0)]$$

- Segment BC: $\varepsilon_c > \varepsilon_{20}$

$$\sigma_c = 0.2 \cdot K \cdot f_c$$

The corresponding tangent moduli are given by the following equations

- Segment OA: $\varepsilon_c \leq \varepsilon_0$

$$E^{tan} = \frac{2 \cdot K \cdot f_c}{\varepsilon_0} \cdot \left(1 - \frac{\varepsilon_c}{\varepsilon_0}\right)$$

- Segment AB: $\varepsilon_0 < \varepsilon_c \leq \varepsilon_{20}$

$$E^{tan} = -Z \cdot K \cdot f_c$$

- Segment BC: $\varepsilon_c > \varepsilon_{20}$

$$E^{tan} = 0$$

where,

$$\varepsilon_0 = 0.002 \cdot K$$

$$K = 1 + \frac{\rho_s \cdot f_{ys}}{f_c}$$

$$Z = \frac{0.5}{\frac{3+0.29 \cdot f_c}{145 \cdot f_c - 1000} + 0.75 \cdot \rho_s \cdot \sqrt{\frac{h}{s_h}} - 0.002 \cdot K}$$

ε_0 is the concrete strain corresponding maximum stress, ε_{20} the concrete strain at 20 percent of maximum stress, K a factor which accounts for the strength increase due to confinement, Z the strain softening slope, f_c the concrete compressive cylinder strength in MPa (1 MPa = 145 psi), f_{ys} the yield strength of stirrups in MPa, ρ_s the ratio of the volume of hoop reinforcement to the volume of concrete core measured to outside of stirrups, h the width of concrete core measured to outside of stirrups, and s_h the center to center spacing of stirrups or hoop sets.

From the previous equations, ε_{20} can be determined from

$$\varepsilon_{20} = \varepsilon_0 + \frac{K f'_c - 0.2 f'_{c,confined}}{Z K f'_c} \quad (2.33)$$

The cyclic unloading and reloading behavior is represented by a set of straight lines. Fig. 2.16 shows that hysteretic behavior occurs under, both, tensile and compressive stress. Although the compressive and tensile hysteresis loops are continuous, they will be discussed separately for the sake of clarity.

On the compressive side of the model, there is a successive degradation of stiffness of both the unloading and reloading lines for increasing values of maximum strain, as shown in Fig. 2.16. The degradation of stiffness is such that the projections of all reloading lines intersect at a common point R in Fig. 2.17. Point R is determined by the intersection of the tangent to the monotonic envelope curve at the origin and the projection of the unloading line from point B that corresponds to concrete strength of $0.2 \cdot f_c$ (Fig. 2.17). The strain and stress at the intersection point are given by the following expressions

$$\varepsilon_R = \frac{0.2 \cdot K \cdot f_c - E_{20} \cdot \varepsilon_{20}}{E_c - E_{20}} \quad (2.34)$$

$$\sigma_R = E_c \cdot \varepsilon_R$$

where E_c is the tangent modulus of the monotonic envelope curve at the origin, E_{20} is the unloading modulus at point B of the monotonic envelope curve with a strength of $0.2 \cdot f_c$. The magnitude of E_{20} has to be determined experimentally.

After unloading from a point on the compressive monotonic envelope (point D in Fig. 2.17), and before reaching the zero stress axis (point H in Fig. 2.17), the model response follows two smaller envelopes that are defined by the following equations,

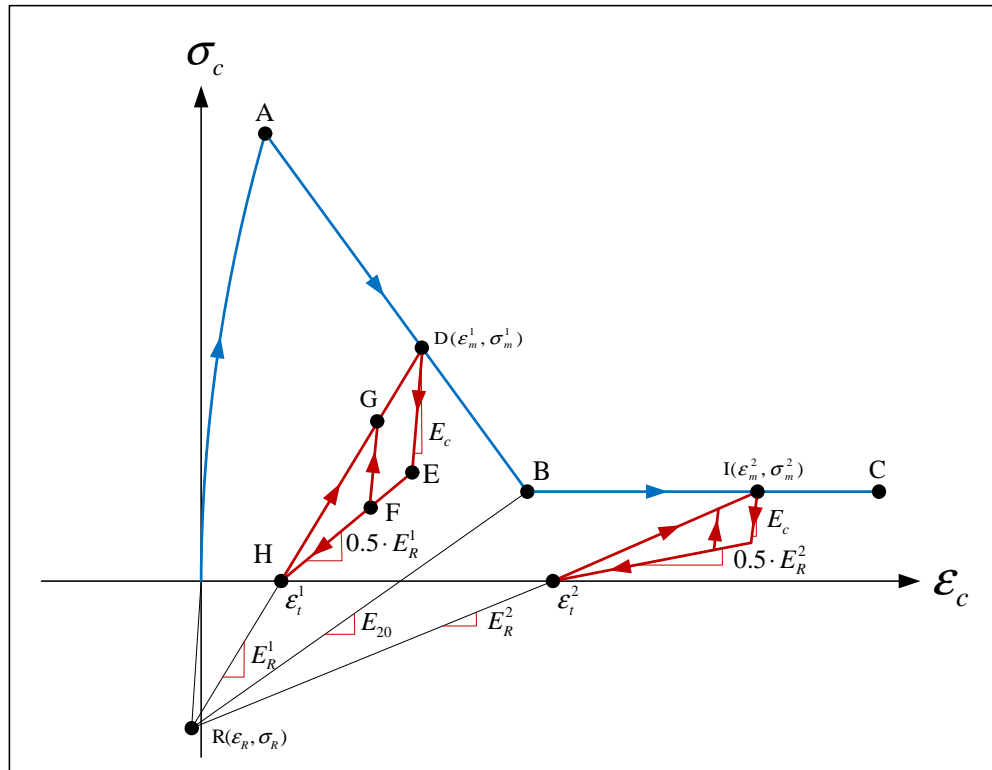


Figure 2.17: Concrete material model under cyclic loading in compression
(?)

- Maximum envelope (line HD)

$$\sigma_{max} = \sigma_m + E_R \cdot (\varepsilon_c - \varepsilon_m) \quad (2.35)$$

- Minimum envelope (line HE)

$$\sigma_{min} = 0.5 \cdot E_R \cdot (\varepsilon_c - \varepsilon_t) \quad (2.36)$$

where,

$$\begin{aligned} E_R &= \frac{\sigma_m - \sigma_R}{\varepsilon_m - \varepsilon_R} \\ \varepsilon_t &= \varepsilon_m - \frac{\sigma_m}{E_R} \end{aligned} \quad (2.37)$$

σ_m and ε_m are the stress and strain at the unloading point on the compressive monotonic envelope, respectively. Therefore, the positions of the two smaller envelopes depend on the position of the unloading point. For partial loading and unloading cycles within the smaller envelopes the model follows straight line with modulus E_c .

In the numerical implementation a trial stress and tangent modulus are assumed based on linear elastic behavior with slope E_c ,

$$\sigma_{c,n}^{tr} = \sigma_{c,n-1} + E_c \cdot \Delta\varepsilon_{c,n} \quad (2.38)$$

where $\sigma_{c,n}^{tr}$ is the updated trial stress, $\sigma_{c,n-1}$ is the previous stress state and $\Delta\varepsilon_{c,n}$ is the strain increment. The following rules are then used to determine actual stress and modulus of the model

$$\begin{aligned} \text{if } \sigma_{min} \leq \sigma_{c,n}^{tr} \leq \sigma_{max} & \text{ then } \sigma_{c,n} = \sigma_{c,n}^{tr} \text{ and } E^{tan} = E_c \\ \text{if } \sigma_{c,n}^{tr} < \sigma_{min} & \text{ then } \sigma_{c,n} = \sigma_{min} \text{ and } E^{tan} = 0.5 \cdot E_r \\ \text{if } \sigma_{c,n}^{tr} > \sigma_{max} & \text{ then } \sigma_{c,n} = \sigma_{max} \text{ and } E^{tan} = E_r \end{aligned} \quad (2.39)$$

The rules governing the hysteretic behavior of the model in compression according to Eq. 2.34 to 2.39 are illustrated by a sample history in Fig. 2.17. If unloading occurs from point D to point E, reloading will be on the same path back to D. If unloading reaches point F, the hysteresis loop DEFGD will result upon reloading. If complete unloading to point H occurs, reloading will result in the hysteresis loop DEHD. It is important to note that the reloading line will always rejoin the compression monotonic envelope at the point of initial unloading. For the case when unloading continues past point H and the model starts reloading in tension, a different set of rules govern the hysteretic behavior. However upon reloading in compression, the model will reenter the compression region at the point on the zero stress axis at the completion of unloading (point H) and whatever happens in the tension region will not affect the behavior of the model, once it returns to the compression region.

The tensile behavior of the model, as shown in Fig. 2.18, takes into account tension stiffening and the degradation of the unloading and reloading stiffness for increasing values of maximum tensile strain after initial cracking. The maximum tensile strength of the concrete (modulus of rupture) is assumed equal to,

$$f_t = 0.6228\sqrt{f_c} \quad (2.40)$$

where f_t and f_c are expressed in MPa.

Fig. 2.18 shows two consecutive tensile hysteresis loops which are part of a sample cyclic history that also include compressive stresses. The model assumes that tensile stress can occur anywhere along the strain axis, either as a result of initial tensile loading or as a result of unloading from a compressive state. In the latter case a tensile stress occurs under a compressive strain. The tensile stress-strain relation is defined by three points with coordinates $(\varepsilon_t, 0)$, $(\varepsilon_{tp}, \sigma_{tp})$ and $(\varepsilon_u, 0)$, as represented by points J, K and M in Fig. 2.18, respectively. ε_t is the strain at the point where the unloading line from the compressive stress region crosses the strain axis. ε_t is given by Eq. 2.37 and changes with maximum compressive strain. ε_{tp} and σ_{tp} are the strain and stress at the peak of the tensile stress-strain relation and are given by the following expressions,

$$\begin{aligned} \varepsilon_{tp} &= \varepsilon_t + \Delta\varepsilon_t \\ \sigma_{tp} &= f_t \cdot \left(1 + \frac{E_t}{E_c}\right) - E_t \cdot \Delta\varepsilon_t \end{aligned} \quad (2.41)$$

where $\Delta\varepsilon_t$ is the previous maximum differential between tensile strain and ε_t as shown in Fig. 2.18. Before initial cracking, $\Delta\varepsilon_t$ is equal to f_t/E_c . E_t is the slope of the softening branch of concrete in tension (typically -1/10 to -1/5

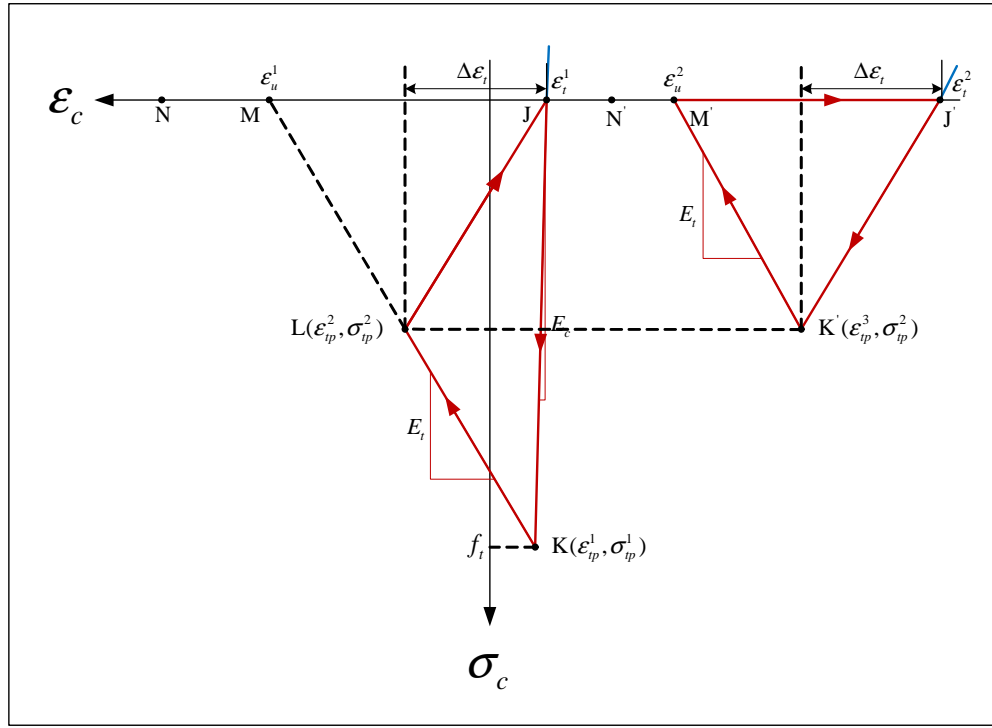


Figure 2.18: Concrete material model under cyclic loading in tension
(?)

of E_c), ε_u is the strain at the point where the tensile stress is reduced to zero and is given by the expression,

$$\varepsilon_u = \varepsilon_t + f_t \cdot \left(\frac{1}{E_t} + \frac{1}{E_c} \right) \quad (2.42)$$

Given these three control points, the tensile stress-strain relation and tangent moduli are defined by the following equations (tension is positive),

- Segment JK : $\varepsilon_t < \varepsilon_c \leq \varepsilon_{tp}$

$$\sigma_c = E^{tan} \cdot (\varepsilon_c - \varepsilon_t), \quad E^{tan} = \frac{\sigma_{tp}}{\varepsilon_{tp} - \varepsilon_t} \quad (2.43)$$

- Segment KM : $\varepsilon_{tp} < \varepsilon_c \leq \varepsilon_u$

$$\sigma_c = \sigma_{tp} + E^{tan} \cdot (\varepsilon_c - \varepsilon_{tp}), \quad E^{tan} = -E_t \quad (2.44)$$

- Segment MN : $\varepsilon_c > \varepsilon_u$

$$\sigma_c = 0, \quad E^{tan} = 0 \quad (2.45)$$

If $\varepsilon_{tp} \geq \varepsilon_u$ then σ_{tp} , σ_c and E^{tan} are all assumed to be zero. The modulus E_t controls the degree of tension stiffening by controlling the slope of Eq. 2.44. The steeper the slope, the smaller will be the effect of tension stiffening. Tensile unloading and reloading are governed by Eq. 2.43 which also includes stiffness degradation for increasing values of strain differential $\Delta\varepsilon_t$. The value of $\Delta\varepsilon_t$ changes whenever $\varepsilon_c - \varepsilon_{tp}$.

The tensile behavior of the model, as characterized by Eq. 2.40 to 2.45, can be better understood by following the example load paths in Fig. 2.18. As the model unloads from compression, it crosses the strain axis at the point J. It then loads in tension until initial cracking occurs at point K. Beyond point K softening commences until the strain reversal point L. The unloading path follows a straight line from point L to point J where the model reloads in compression. The second time the model goes into tension is at point J'. The reloading path J'K' is exactly the duplication of the previous unloading path LJ that has been shifted a distance JJ' along the strain axis. At point K' the model rejoins the softening branch which continues until the tensile stress is reduced to zero at point M'. The stress remains zero through the strain reversal point N' until the model reloads in compression at point J'. Henceforth,

the tensile stress capacity of the model is reduced to zero.

The present concrete model is relatively economical in terms of the amount of memory required of the past stress-strain history. The parameters that are used as memory can be listed as follows:

1. the stress and strain at the point corresponding to the last model state
2. the strain at the last unloading point on the compressive monotonic envelope, ε_m
3. The differential $\Delta\varepsilon_t$ between maximum previous tensile strain and ε_t

The concrete damage considered in the present model is in the form of unloading and reloading stiffness degradation for increasing values of maximum strain. But actual concrete damage also includes the reduction of the monotonic envelopes under cyclic loading.

2.2.1.1.2 Data preparation for modified Kent and Park model

Listing 2.1: Data Preparation for Modified Kent and Park Model

```

1
2 %% Modified Kent and Park Model for CORE CONCRETE
3 clear
4 clc
5
6 %% User input data
7 % Units 1: MPa; 2 ksi
8 units=2;
9 % concrete compressive strength
10 f_c_unconfined = 5;
11 % steel
12 f_ys = 60;           %Yield strength of stirrups
13 f_ys = 1.25*f_ys;   %To account for rate effect
14 % geometry
15 b = 14;             %Member width (in)
16 d = 20;             %Member height (in)
17 L = 12*13.5;        %Member length
18 cover = 2;         %Cover to stirrup
19 sh = 12;           %Center to center spacing of stirrups
20 stirrup_dia = 3/8; %in
21 n_stirrup = 15;    %Number shear reinforcing bars in member
22
23 %% Compute intermediary values
24 A_core = (b-2*cover)*(d-2*cover); %in^2
25 P_core = 2*((b-2*cover) + 2*(d-2*cover)); %Perimeter of core approx. equal to length of single stirrup
26 V_stirrup = n_stirrup*(pi/4*stirrup_dia^2*P_core); %Total volume of shear reinforcement
27 rho_s = V_stirrup/(A_core*L); %Ratio of volume of shear reinforcement to volume of
28 %concrete core measured to outside of stirrups
29 h = d-2*cover; %width of concrete core measured to outside of stirrups
30
31 %% Compute parameters
32 lambda = 0.3; %Ratio of unloading slope at epsilon_c to slope E0, Default value 0.3
33 % Factor to account for strength increase due to confinement
34 K = 1 + (rho_s*f_ys)/f_c_unconfined;
35 % Controls the descending branch of the compressive stress strain curve
36 switch units
37 case 1 % MPa
38     Z = 0.5/(((3+0.29*f_c_unconfined)/(0.145*f_c_unconfined - 1))+(0.75*rho_s*sqrt(h/sh))-0.002*K); %Strain softening slope
39 case 2 % psi
40     Z = 0.5/(((3+0.29*f_c_unconfined/0.145)/(f_c_unconfined - 1))+(0.75*rho_s*sqrt(h/sh))-0.002*K); %Strain softening slope
41 end
42 % Compressive strain corresponding to peak stress in compression CONFINED
43
44 epsilon_0_unconfined=0.003;
45 epsilon_0_confined = 0.002*K;
46 % Peak stress in CONFINED concrete
47 f_c_confined = K*f_c_unconfined;
48 % Ultimate strain in compression
49 f_c_20_unconfined=f_c_unconfined/3;
50 f_c_20_confined=0.2*f_c_confined;
51 epsilon_20_confined = epsilon_0_confined+(K*f_c_unconfined-f_c_20_confined)/(Z*K*f_c_unconfined);
52 epsilon_20_unconfined = 3*epsilon_0_unconfined;
53 % Initial E in compression (same for confined and unconfined concrete)
54 E_0_unconfined=2*f_c_unconfined/epsilon_0_unconfined;
55 E_0_confined=2*f_c_confined/epsilon_0_confined;
56 % Initial E in tension
57 E_t_unconfined=E_0_unconfined/5;
58 E_t_confined=E_0_confined/5;
59 % Tensile strength
60 switch units
61 case 1 % MPa
62     sigma_t_unconfined=0.6228*sqrt(f_c_unconfined);
63     sigma_t_confined=0.6228*sqrt(f_c_confined);
64 case 2 % psi
65     sigma_t_unconfined = 7.5*sqrt(f_c_unconfined*1000);
66     sigma_t_confined = 7.5*sqrt(f_c_confined*1000);
67 end
68
69 fid=fopen('Concrete-Properties.out','w+');
70
71 fprintf(fid,' Preparation of Concrete input data in Mercury\n');
72 fprintf(fid,' for the Modified Kent and Park Model\n');
73 switch units
74 case 1
75     fprintf(fid,' Units MPa\n\n');
76 case 2
77     fprintf(fid,' Units Ksi\n\n');

```

```

78 end
79 fprintf(fid, '          UNCONFINED CONCRETE\n\n');
80 fprintf(fid, 'Initial tangent modulus in tension.....%8.2f\n', E_t_unconfined);
81 fprintf(fid, 'Confined compressive strength.....%8.2f\n', -f_c_unconfined);
82 fprintf(fid, 'Peak strain in compression.....%8.2f\n', -epsilon_0_unconfined);
83 fprintf(fid, 'Strain at 20 percent of peak stress.....%8.2f\n', -epsilon_20_unconfined);
84 fprintf(fid, 'Stress at 20 percent of peak stress.....%8.2f\n', -f_c_20_unconfined);
85 fprintf(fid, 'Tensile strength.....%8.2f\n', sigma_t_unconfined);
86 fprintf(fid, 'Coefficient Lambda.....%8.2f\n', lambda);
87
88 fprintf(fid, '          CONFINED CONCRETE\n\n');
89 fprintf(fid, 'Initial tangent modulus in tension.....%8.2f\n', E_t_confined);
90 fprintf(fid, 'Confined compressive strength.....%8.2f\n', -f_c_confined);
91 fprintf(fid, 'Peak strain in compression.....%8.2f\n', -epsilon_0_confined);
92 fprintf(fid, 'Strain at 20 percent of peak stress.....%8.2f\n', -epsilon_20_confined);
93 fprintf(fid, 'Stress at 20 percent of peak stress.....%8.2f\n', -f_c_20_confined);
94 fprintf(fid, 'Tensile strength.....%8.2f\n', sigma_t_confined);
95 fprintf(fid, 'Coefficient Lambda.....%8.2f\n', lambda);
96 fclose(fid)

```

2.2.1.1.3 Determination for modified Kent and Park model Fig. 2.19 and 2.20 show the procedure to obtain uniaxial stress σ and tangent modulus E^{tan} from uniaxial strain ε .

2.2.2 “Exact” Models

2.2.2.1 Anisotropic damage model with effective damage and stiffness recovery

Whereas the modified Kent-Park model has been successfully used in many applications, it remains a heuristic empirical model whose accurate performance may not be assured within the context of real-time hybrid simulation with limited number of iterations. To address this possible handicap, a more rigorous and thermodynamically correct constitutive model for concrete is implemented². Such a model was developed by ?.

2.2.2.1.1 Constitutive model

Thermodynamic potential The Gibbs potential $\rho \cdot \phi^*$ is based on the Ladeveze’s framework (?) for anisotropic damage model. The splitting of the stress tensor into deviatoric and hydrostatic components allows us to consider separately the shear and the hydrostatic parts. Furthermore, each component is again split into two additional ones: the positive and the negative in order to model the unilateral effect, Fig. 2.21. The damage influences only the positive part through the damage tensor \mathbf{D} .

$$\rho \cdot \phi^* = \frac{1 + \nu}{2 \cdot E} \left[\text{Tr} \left(\mathbf{H} \cdot \boldsymbol{\sigma}_+^D \cdot \mathbf{H} \cdot \boldsymbol{\sigma}_+^D \right) + \text{Tr} \left(\langle \boldsymbol{\sigma}^D \rangle_+ \cdot \langle \boldsymbol{\sigma}^D \rangle_- \right) \right] + \frac{1 - 2\nu}{6 \cdot E} \left[\frac{\langle \text{Tr} \boldsymbol{\sigma} \rangle_+^2}{1 - D_H} + \langle \text{Tr} \boldsymbol{\sigma} \rangle_-^2 \right] \quad (2.46)$$

where ρ , ν and E are respectively the density, the Poisson’s ratio and Young modulus. The tensor \mathbf{H} is defined by

$$\mathbf{H} = (\mathbf{1} - \mathbf{D})^{\frac{1}{2}}$$

where $(\cdot)^D$ is the deviatoric of (\cdot) ,

$$(\cdot)^D = (\cdot) - \frac{1}{3} \text{Tr}(\cdot) \cdot \mathbf{1}$$

$\langle \cdot \rangle_{+,-}$ is the positive or negative part of (\cdot) , and D_H is the hydrostatic part of \mathbf{D} ,

$$D_H = \frac{1}{3} \text{Tr} \mathbf{D}$$

In Ladeveze’s framework (?), $\boldsymbol{\sigma}_+^D$ is an unusual positive part of $\boldsymbol{\sigma}^D$: if λ_I are the eigenvalues of $\mathbf{H} \boldsymbol{\sigma}^D$ and \mathbf{T}^I the corresponding eigenvectors, $\boldsymbol{\sigma}_+^D$ is

$$\boldsymbol{\sigma}_+^D = \sum_I \langle \lambda_I \rangle_+ \left(\mathbf{H}^{-1} \cdot \mathbf{T}^I \right) \left(\mathbf{H}^{-1} \cdot \mathbf{T}^I \right)^T$$

²Implementation of this model was made possible through the collaboration of Prof. Ragueneau and Mr. Lebon from ENS/Cachen

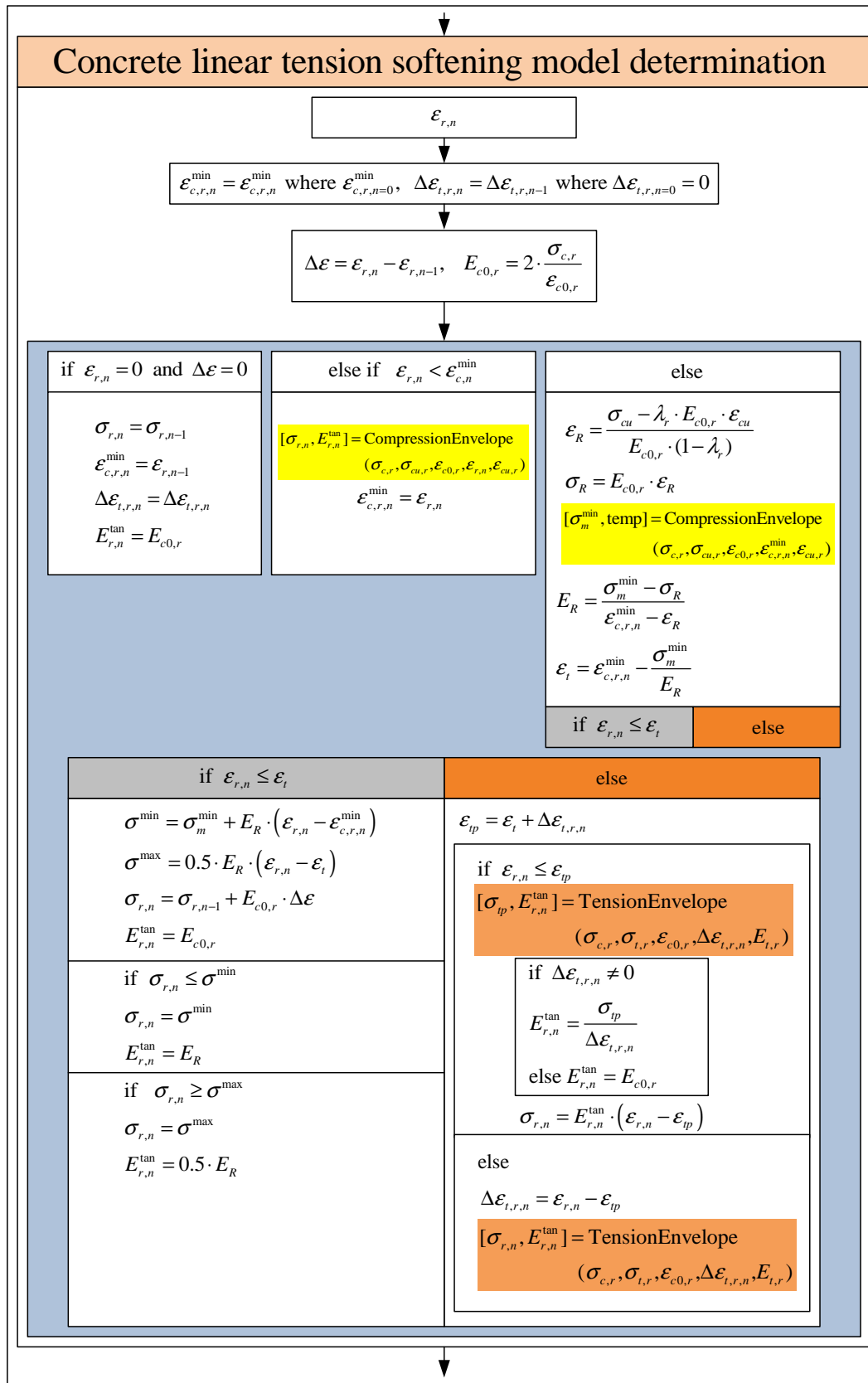


Figure 2.19: Determination (1) for modified Kent and Park model

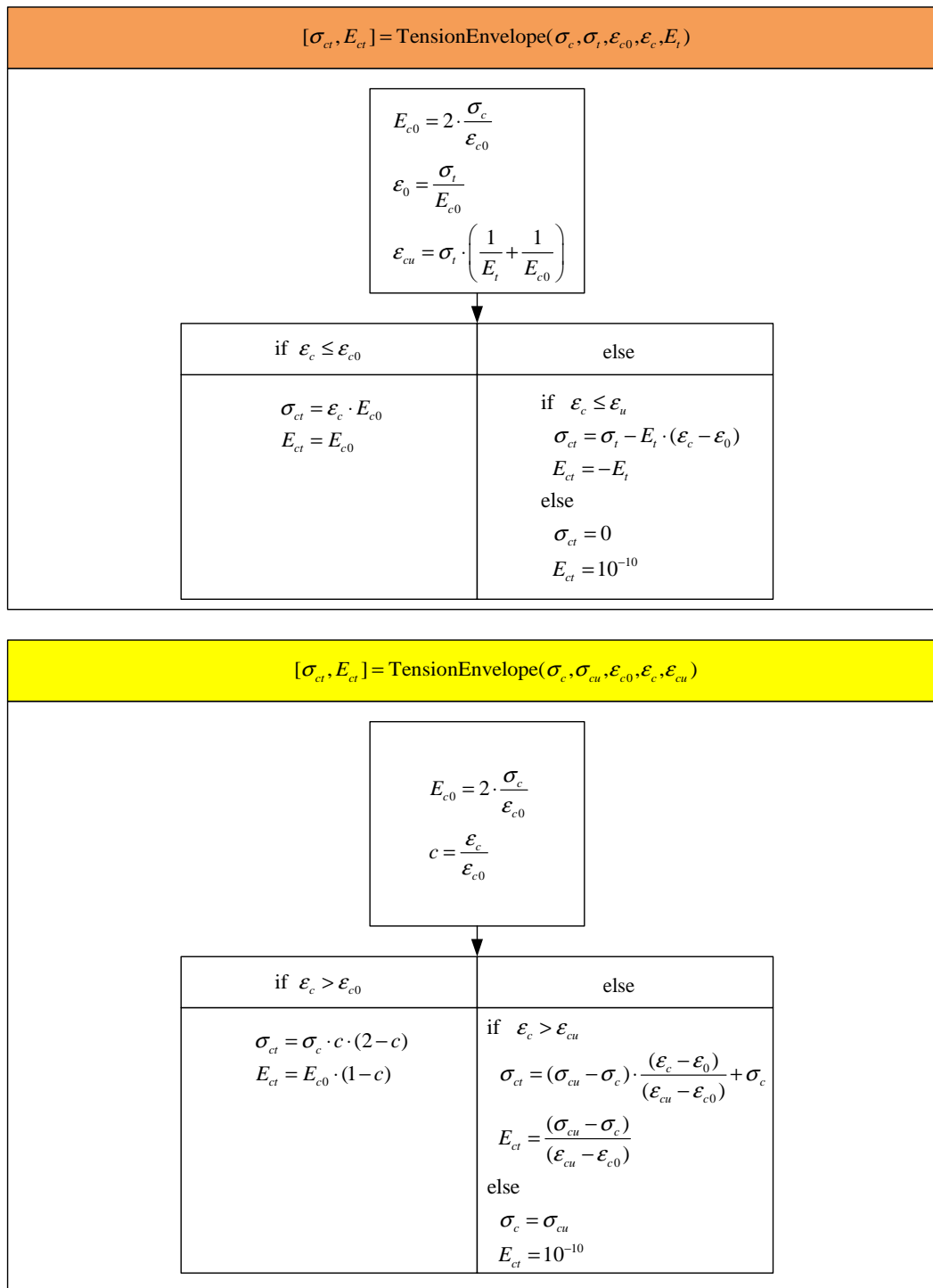


Figure 2.20: Determination (2) for modified Kent and Park model

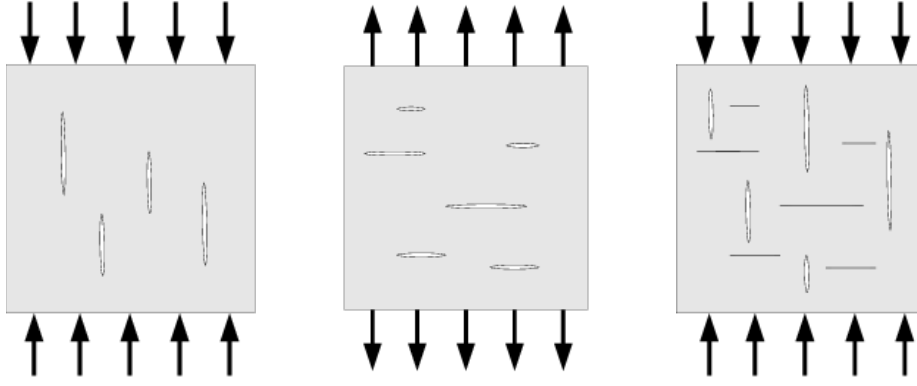


Figure 2.21: Cracks induced by compression, tension, tension and compression

Constitutive law The constitutive law is derived from Gibbs potential using Eq. 2.46:

$$\begin{aligned}
 \boldsymbol{\varepsilon} &= \rho \cdot \frac{\partial \phi^*}{\partial \boldsymbol{\sigma}} \\
 &= \frac{1+\nu}{E} \left[\left(\mathbf{H} \cdot \boldsymbol{\sigma}_+^D \mathbf{H} \right)^D + \langle \boldsymbol{\sigma}^D \rangle_-^D \right] + \frac{1-2\nu}{3 \cdot E} \left[\frac{\langle \text{Tr}(\boldsymbol{\sigma}) \rangle_+}{1-D_H} + \langle \text{Tr}(\boldsymbol{\sigma}) \rangle_- \right] \cdot \mathbf{I} \\
 &= \frac{1+\nu}{E} \cdot \tilde{\boldsymbol{\sigma}} - \frac{\nu}{E} \cdot \text{Tr}(\tilde{\boldsymbol{\sigma}}) \cdot \mathbf{I}
 \end{aligned}$$

where $\tilde{\boldsymbol{\sigma}}$ is the effective stress obtained by rewriting the stress strain law in terms of the effective stress:

$$\tilde{\boldsymbol{\sigma}} = \left[\left(\mathbf{H} \cdot \boldsymbol{\sigma}_+^D \mathbf{H} \right)^D + \langle \boldsymbol{\sigma}^D \rangle_-^D \right] + \left[\frac{\langle \text{Tr}(\boldsymbol{\sigma}) \rangle_+}{1-D_H} + \langle \text{Tr}(\boldsymbol{\sigma}) \rangle_- \right] \cdot \mathbf{I}$$

Damage evolution law The damage evolution law should account for cyclic loading. Hence, the damage criterion f should depend on an effective damage d_ε (?) such that

$$\begin{aligned}
 f &= \hat{\boldsymbol{\varepsilon}} - \kappa(d_\varepsilon) \\
 d_\varepsilon &= \frac{\mathbf{D} : \langle \boldsymbol{\varepsilon} \rangle_+}{\max(\boldsymbol{\varepsilon}_I)}
 \end{aligned}$$

where $\hat{\boldsymbol{\varepsilon}}$ is Mazars strain defined by

$$\hat{\boldsymbol{\varepsilon}} = \sqrt{\langle \boldsymbol{\varepsilon} \rangle_+ : \langle \boldsymbol{\varepsilon} \rangle_+}$$

and κ the consolidation function,

$$\kappa(d_\varepsilon) = a \cdot \tan \left[\frac{d_\varepsilon}{aA} + \arctan \left(\frac{\kappa_0}{a} \right) \right]$$

where A and a are two damage coefficients, and κ_0 is initial elasticity threshold.

Thus, if the damage criterion is negative, the material is in the linear elastic range, and if it is positive, the damage increases along the positive strain:

$$\dot{\mathbf{D}} = \dot{\lambda} \langle \boldsymbol{\varepsilon} \rangle_+$$

where the damage Lagrange multiplier $\dot{\lambda}$ is determined from the Kuhn-Tucker condition ($f = 0$ and $f' = 0$).

2.2.2.1.2 Uniaxial multi-fiber formulation Multi-fiber formulation based on Euler-Bernoulli beam theory is equivalent to a uniaxial loading. Hence, the stress tensor and the strain tensor are reduced to

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and,

$$\boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{11} & 0 & 0 \\ 0 & \varepsilon_{22} & 0 \\ 0 & 0 & \varepsilon_{33} \end{pmatrix}$$

The damage tensor becomes diagonal and the constitutive law can be rewritten as (?),

$$\boldsymbol{\varepsilon} = \mathbf{B}(D_1, D_2) \cdot \frac{\boldsymbol{\sigma}}{E} \quad (2.47)$$

where \mathbf{B} is a diagonal tensor which depends on the damage variable and loading, and D_1 and D_2 are associated with tension and compression damage respectively.

Rewriting Eq. 2.47 with matrix form,

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_2 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_2 \end{pmatrix}$$

so

$$\boldsymbol{\sigma}^D = \begin{pmatrix} \frac{2\sigma}{3} & 0 & 0 \\ 0 & \frac{-1\sigma}{3} & 0 \\ 0 & 0 & \frac{-1\sigma}{3} \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} \frac{1}{\sqrt{1-D_1}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{1-D_2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{1-D_2}} \end{pmatrix}$$

$$\text{Tr}(\boldsymbol{\sigma}) = \sigma \quad D_H = D_1 + 2D_2$$

$$\boldsymbol{\varepsilon}_e = \frac{1+\nu}{E} \left(\left(\mathbf{H}\boldsymbol{\sigma}_+^D \mathbf{H} \right)^D + \boldsymbol{\sigma}_-^D \right) + \frac{1-2\nu}{3E} \left[\frac{\langle \text{Tr}(\boldsymbol{\sigma}) \rangle_+}{1-D_H} + \langle \text{Tr}(\boldsymbol{\sigma}) \rangle_- \right] \mathbf{Id}$$

For tension ($\sigma > 0$), we assume that only D_1 increases.

$$\boldsymbol{\varepsilon}_e = \begin{pmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{pmatrix} = \frac{\sigma}{E} \mathbf{B}^t = \frac{\sigma}{E} \begin{pmatrix} B_1^t & 0 \\ 0 & B_2^t \end{pmatrix}$$

$$B_1^t = \frac{1+\nu}{9} \left(\frac{4}{1-D_1} + \frac{2}{1-D_2} \right) + \frac{1-2\nu}{3} \frac{1}{1-(D_1+2D_2)}$$

$$B_2^t = \frac{1+\nu}{9} \left(\frac{-2}{1-D_1} - \frac{1}{1-D_2} \right) + \frac{1-2\nu}{3} \frac{1}{1-(D_1+2D_2)}$$

For compression ($\sigma < 0$), we assume that only D_2 increases.

$$\boldsymbol{\varepsilon}_e = \begin{pmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{pmatrix} = \frac{\sigma}{E} \mathbf{B}^c = \frac{\sigma}{E} \begin{pmatrix} B_1^c & 0 \\ 0 & B_2^c \end{pmatrix}$$

$$B_1^c = \frac{1+\nu}{9} \left(\frac{4}{1-D_1} + \frac{2}{1-D_2} \right) + \frac{1-2\nu}{3}$$

$$B_2^c = \frac{1+\nu}{9} \left(\frac{-2}{1-D_1} - \frac{1}{1-D_2} \right) + \frac{1-2\nu}{3}$$

2.2.2.1.3 Determination for anisotropic damage model

2.2.2.1.3.1 Tension loading: $\varepsilon_{11} > 0$

- Constitutive equations

If $f = \hat{\varepsilon} - \kappa_0 \geq 0$ and $\eta_t = 1$ then:

$$\sigma = \frac{E}{B_{11}^t} \varepsilon$$

$$\sigma = \frac{E}{\frac{1+\nu}{9} \cdot \left(\frac{4}{1-D_1} + 2 \right) + \frac{1-2\nu}{3 \left(\frac{1-D_1+2D_2}{3} \right)}} \cdot \varepsilon \quad (2.48)$$

$$D_1 = a \cdot A \left(\text{atan} \left(\frac{\varepsilon_{11}}{a} \right) - \text{atan} \left(\frac{\kappa_0}{a} \right) \right)$$

where, $\hat{\varepsilon} = \sqrt{\langle \boldsymbol{\varepsilon} \rangle_+ : \langle \boldsymbol{\varepsilon} \rangle_+} = \varepsilon_{11} \geq 0$,

$$\langle \boldsymbol{\varepsilon} \rangle_+ = \begin{pmatrix} \varepsilon_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

D_2 is constant and tangent modulus is determined with Eq. 2.48:

$$\dot{\sigma} = E^{tan} \cdot \dot{\varepsilon} \quad (2.49)$$

However, if $f = \hat{\varepsilon} - \kappa_0 < 0$, the tangent modulus is equal to the secant one.

$$E^{tan} = \frac{E}{B_{11}^t}$$

- Coherent tangent stiffness modulus in numerical implementation

If $f = \hat{\varepsilon} - \kappa_0 \geq 0$ and $\eta_t = 1$, tangent modulus is determined with Eq. 2.48 and 2.49:

$$E_n^{tan} = \frac{\partial \sigma_n}{\partial \varepsilon_n}$$

$$B_{11,n}^t = \frac{1+\nu}{9} \cdot \left(\frac{4}{1-D_{1,n}} + 2 \right) + \frac{1-2\nu}{3 \cdot \left(1 - \frac{(D_{1,n}+2 \cdot D_{2,n-1})}{3} \right)}$$

$$D_{1,n} = a \cdot A \left(\text{atan} \left(\frac{\varepsilon_{11,n}}{a} \right) - \text{atan} \left(\frac{\kappa_0}{a} \right) \right)$$

where n is current excitation step.

$$\frac{\partial \sigma_n}{\partial \varepsilon_n} = \frac{\partial}{\partial \varepsilon_n} \left(\frac{E}{B_{11,n}^t} \cdot \varepsilon_n \right) = \frac{E}{B_{11,n}^t} - E \cdot \varepsilon_n \frac{1}{(B_{11,n}^t)^2} \cdot \frac{\partial B_{11,n}^t}{\partial \varepsilon_n}$$

where,

$$\frac{\partial B_{11,n}^t}{\partial \varepsilon_n} = \frac{\partial B_{11,n}^t}{\partial D_{1,n}} \cdot \frac{\partial D_{1,n}}{\partial \varepsilon_n}$$

$$C_1 = \frac{\partial B_{11,n}^t}{\partial D_{1,n}} = \frac{1+\nu}{9} \cdot \frac{4}{(1-D_{1,n})^2} + \frac{1-2\nu}{9 \left(1 - \frac{D_{1,n}+2D_{2,n-1}}{3} \right)}$$

$$C_2 = \frac{\partial D_{1,n}}{\partial \varepsilon_n} = \frac{\partial}{\partial \varepsilon_n} \left(a \cdot A \left(\text{atan} \left(\frac{\varepsilon_n}{a} \right) - \text{atan} \left(\frac{\kappa_0}{a} \right) \right) \right) = \frac{A}{1 + \left(\frac{\varepsilon_n}{a} \right)^2}$$

Finally, we obtain tangent modulus:

$$E_n^{tan} = \frac{E}{B_{11,n}^t} - E \cdot \varepsilon_n \cdot \frac{1}{(B_{11,n}^t)^2} \cdot C_1 \cdot C_2$$

2.2.2.1.3.2 Compressive loading: $\varepsilon_{11} < 0$

- Constitutive equations

If $f = \hat{\varepsilon} - \kappa_0 \geq 0$ and $\eta_t = 1$ then:

$$\sigma = \frac{E}{\frac{1+\nu}{9} \cdot \left(\frac{2}{1-D_2} + 4 \right) + \frac{1-2\nu}{3}} \cdot \varepsilon$$

$$D_2 = \frac{a \cdot A}{2} \left(\text{atan} \left(\frac{\sqrt{2} \cdot \varepsilon_{22}}{a} \right) - \text{atan} \left(\frac{\kappa_0}{a} \right) \right) \quad (2.50)$$

$$\varepsilon_{22} = \frac{B_{22}^c}{B_{11}^c} \cdot \varepsilon_{11}$$

where, $\hat{\varepsilon} = \sqrt{\langle \boldsymbol{\varepsilon} \rangle_+ : \langle \boldsymbol{\varepsilon} \rangle_+} = \sqrt{2 \cdot \varepsilon_{22}^2} \geq 0$,

$$\langle \boldsymbol{\varepsilon} \rangle_+ = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \varepsilon_{22} & 0 \\ 0 & 0 & \varepsilon_{33} = \varepsilon_{22} \end{pmatrix}$$

D_1 is constant and tangent modulus is determined with Eq. 2.50:

$$\dot{\sigma} = E^{tan} \cdot \dot{\varepsilon} \quad (2.51)$$

However, if $f = \hat{\varepsilon} - \kappa_0 < 0$, the tangent modulus is equal to the secant one.

$$E^{tan} = \frac{E}{B_{11}^c}$$

- Coherent tangent stiffness modulus in numerical implementation

If $f = \hat{\varepsilon} - \kappa_0 \geq 0$ and $\eta_t = 1$, tangent modulus is determined with Eq. 2.50 and 2.51:

$$\begin{aligned} E_n^{tan} &= \frac{\partial \sigma_n}{\partial \varepsilon_n} \\ B_{11,n-1}^c &= \frac{1+\nu}{9} \cdot \left(4 + \frac{2}{1-D_{2,n-1}} \right) + \frac{1-2\nu}{3} \\ B_{22,n-1}^c &= -\frac{1+\nu}{9} \cdot \left(2 + \frac{1}{1-D_{2,n-1}} \right) + \frac{1-2\nu}{3} \\ B_{11,n}^c &= \frac{1+\nu}{9} \cdot \left(4 + \frac{2}{1-D_{2,n}} \right) + \frac{1-2\nu}{3} \\ B_{22,n}^c &= -\frac{1+\nu}{9} \cdot \left(2 + \frac{1}{1-D_{2,n}} \right) + \frac{1-2\nu}{3} \end{aligned}$$

where n is current excitation step.

$$\begin{aligned} \tilde{\nu}_{n-1} &= -\frac{B_{22,n-1}^c}{B_{11,n-1}^c}, \quad \varepsilon_{22,n}^{tr} = -\tilde{\nu}_{n-1} \cdot \varepsilon_{11,n} \\ \tilde{\nu}_n &= -\frac{B_{22,n}^c}{B_{11,n}^c}, \quad \varepsilon_{22,n} = -\tilde{\nu}_n \cdot \varepsilon_{11,n} \end{aligned}$$

where, $\varepsilon_{22,n}^{tr}$ is trial strain on ε_{22} .

$$\frac{\partial \sigma_n}{\partial \varepsilon_n} = \frac{\partial}{\partial \varepsilon_n} \left(\frac{E}{B_{11,n}^c} \cdot \varepsilon_n \right) = \frac{E}{B_{11,n}^c} - E \cdot \varepsilon_n \frac{1}{(B_{11,n}^c)^2} \cdot \frac{\partial B_{11,n}^c}{\partial \varepsilon_n}$$

where,

$$\begin{aligned} \frac{\partial B_{11,n}^c}{\partial \varepsilon_n} &= \frac{\partial B_{11,n}^c}{\partial D_{2,n}} \cdot \frac{\partial D_{2,n}}{\partial \varepsilon_n} \\ C_1 &= \frac{\partial B_{11,n}^c}{\partial D_{2,n}} = \frac{2(1+\nu)}{9(1-D_{2,n})^2} \\ C_2 &= \frac{\partial D_{2,n}}{\partial \varepsilon_n} = \frac{\partial}{\partial \varepsilon_n} \left(a \cdot A \left(\text{atan} \left(\frac{-\tilde{\nu}_{n-1} \cdot \sqrt{2} \cdot \varepsilon_{11,n}}{a} \right) - \text{atan} \left(\frac{\kappa_0}{a} \right) \right) \right) \\ &= -\tilde{\nu}_{n-1} \frac{A}{\sqrt{2}} \cdot \frac{1}{1 + \left(\frac{-\tilde{\nu}_{n-1} \cdot \sqrt{2} \cdot \varepsilon_{11,n}}{a} \right)^2} \end{aligned}$$

Finally, we obtain tangent modulus:

$$E_n^{tan} = \frac{E}{B_{11,n}^c} - E \cdot \varepsilon_n \cdot \frac{1}{(B_{11,n}^c)^2} \cdot C_1 \cdot C_2$$

Fig. 2.22 describes the relationship between strain and stress in anisotropic damage model without permanent strain.

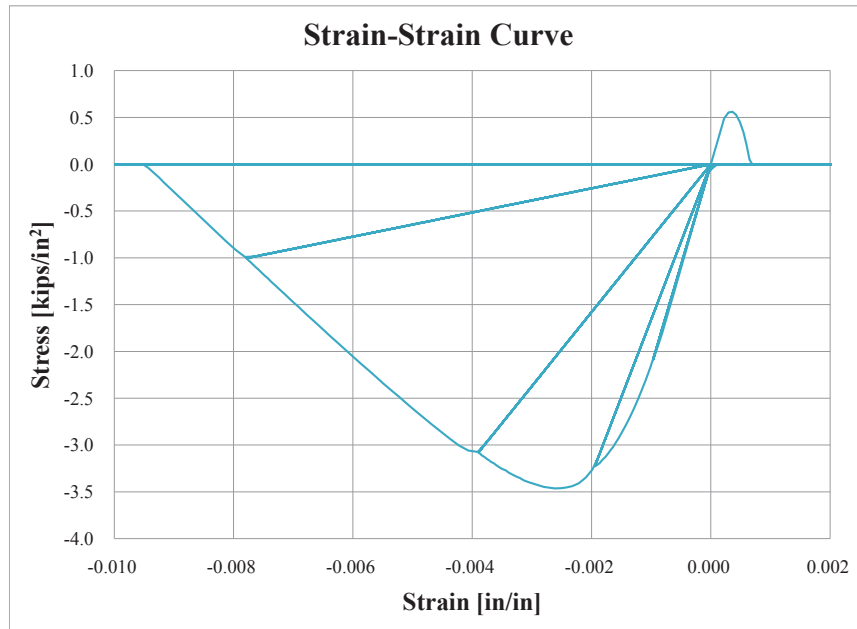


Figure 2.22: Strain-stress curve for anisotropic damage model without permanent strain

Fig 2.23 to 2.25 show the procedure to obtain uniaxial stress σ and tangent modulus E^{tan} from uniaxial strain ε .

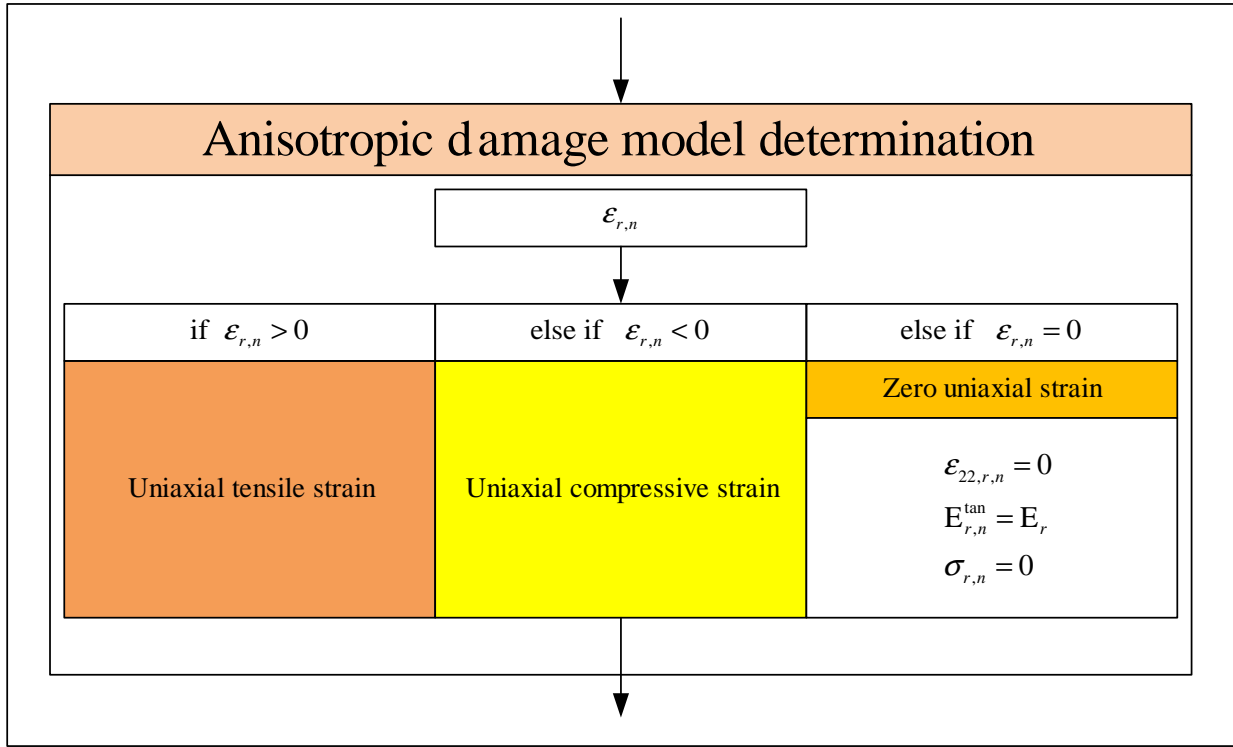


Figure 2.23: Determination (1) for the anisotropic damage model

2.2.2.2 Anisotropic damage model with effective damage, stiffness recovery and permanent strains ϵ_{an}

The preceding anisotropic damage model did not account for permanent deformations. The following extension of the model shall address this limitation.

This model is based on the anisotropic damage model ψ_e^* previously identified as ψ^* .

$$\rho\psi^* = \rho\psi_e^* + \rho\psi_{an}^* \quad (2.52)$$

For the damaged part of the model, the consolidation function is altered to take into consideration the elastic feedback in compression:

$$\kappa(d_\epsilon) = \frac{d_\epsilon}{A} + \kappa_0 \quad (2.53)$$

where A and κ_0 are two parameters which depend on the material.

The damage criterion will depend on the elastic strains and not on total ones:

$$f = \hat{\epsilon}_{el} - \kappa(d_\epsilon) \quad (2.54)$$

The anelastic part of the model is:

$$\rho\psi_{an}^* = \boldsymbol{\sigma} : \boldsymbol{\epsilon}_{an} \quad (2.55)$$

This corresponds to the partition of the total strain into an elastic and permanent component.

$$\boldsymbol{\epsilon} = \frac{\partial \rho\psi^*}{\partial \boldsymbol{\sigma}} = \frac{\partial \rho\psi_e^*}{\partial \boldsymbol{\sigma}} + \frac{\partial \rho\psi_{an}^*}{\partial \boldsymbol{\sigma}} = \boldsymbol{\epsilon}_e + \boldsymbol{\epsilon}_{an} \quad (2.56)$$

The evolution of the permanent strains are derived by the following equations:

$$\dot{\boldsymbol{\epsilon}}_{an} = g_D(d_\epsilon) \dot{\mathbf{D}}^D + g_H(d_\epsilon) \dot{d}_\epsilon \mathbf{Id} \quad (2.57)$$

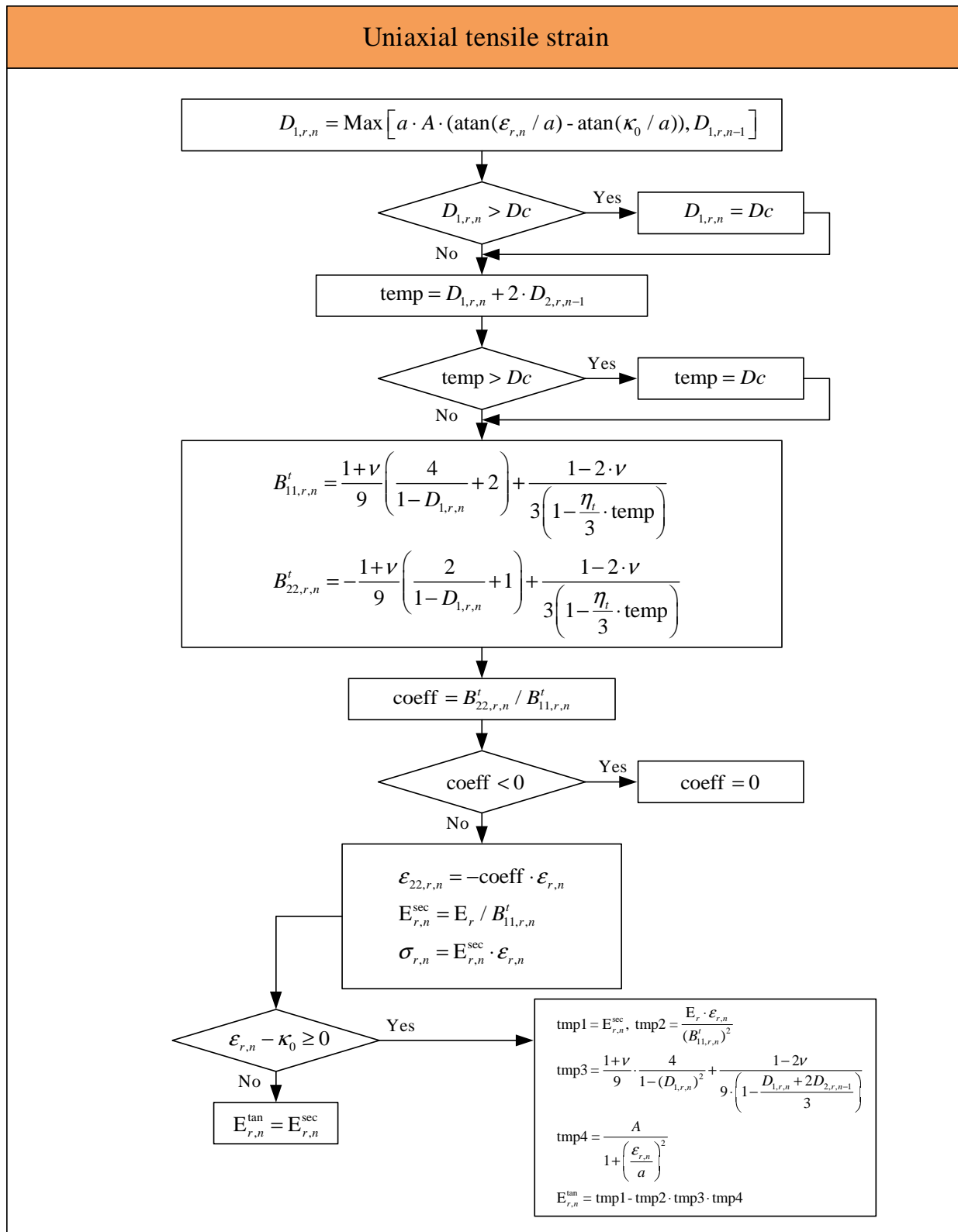


Figure 2.24: Determination (2) for the anisotropic damage model

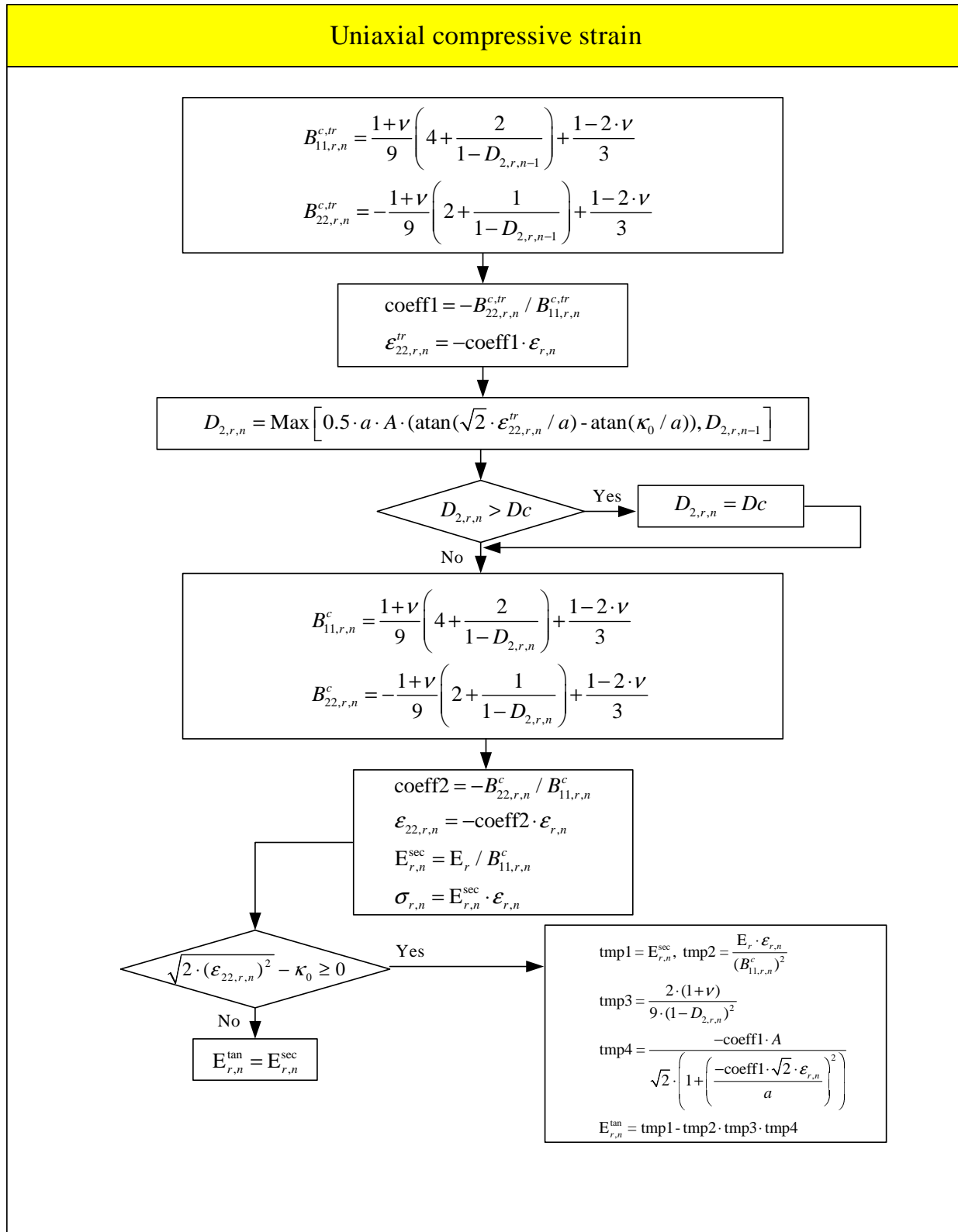


Figure 2.25: Determination (3) for the anisotropic damage model

with $g_D(d_\epsilon)$ $g_H(d_\epsilon)$ two functions depending on the activate damage to respect the dissymmetry tension/compression. Two continuum bilinear functions are proposed here:

$$\begin{aligned} \text{if } d_\epsilon < 1 \quad g_D(d_\epsilon) &= \frac{k_D}{K_D} d_\epsilon & \text{else } g_D(d_\epsilon) &= \frac{k_D}{K_D} + k_D (d_\epsilon - 1) \\ g_H(d_\epsilon) &= \frac{k_H}{K_H} d_\epsilon & g_H(d_\epsilon) &= \frac{k_H}{K_H} + k_H (d_\epsilon - 1) \end{aligned} \quad (2.58)$$

2.2.2.2.1 Uniaxial numerical scheme The loading state is defined by the sign of the elastic strain. Thus to determine the loading state at the timestep n , the increment of strain $\Delta\epsilon$ is compared with the elastic strain at timestep $n-1$: If $\epsilon_{n-1}^e < 0$ and $\Delta\epsilon < -\epsilon_{n-1}^e$, the loading state remains in compression, else it switches to tension. If $\epsilon_{n-1}^e > 0$ and $\Delta\epsilon < -\epsilon_{n-1}^e$, the loading state remains in tension, else it switches to compression.

In tension:

- Elastic forecast : $\epsilon_{1,n}^e = \epsilon_{1,n-1}^e + \Delta\epsilon_1$
- Damage criterion test: f
 - if $f < 0$, elastic behavior:

$$\begin{aligned} D_{1,n} &= D_{1,n-1} \\ D_{2,n} &= D_{2,n-1} \\ \epsilon_{1,n}^{an} &= \epsilon_{1,n-1}^{an} \\ \epsilon_{2,n}^{an} &= \epsilon_{2,n-1}^{an} \end{aligned} \quad (2.59)$$

– else, damaging behavior:

$$\begin{aligned} D_{1,n} &= A (\epsilon_{1,n}^e - \kappa_0) \\ D_{2,n} &= D_{2,n-1} \\ \epsilon_{1,n}^{an} &= \epsilon_{1,n-1}^{an} + \left(\frac{2k_D}{3K_D} + \frac{k_H}{K_H} \right) D_{1,n} \\ \epsilon_{2,n}^{an} &= \epsilon_{2,n-1}^{an} + \left(\frac{-k_D}{3K_D} + \frac{k_H}{K_H} \right) D_{1,n} \end{aligned} \quad (2.60)$$

- Computation of σ and $\epsilon_{2,n}^e$:

$$\begin{aligned} B_{1,n} &= \frac{1+\nu}{9} \cdot \left(\frac{4}{1-D_{1,n}} + 2 \right) + \frac{1-2\nu}{3 \cdot \left(1 - \frac{(D_{1,n}+2 \cdot D_{2,n-1})}{3} \right)} \\ B_{2,n} &= \frac{1+\nu}{9} \cdot \left(\frac{2}{1-D_{1,n}} + 1 \right) + \frac{1-2\nu}{3 \cdot \left[1 - \frac{\eta}{3} \cdot (D_{1,n}+2 \cdot D_{2,n-1}) \right]} \\ \sigma_n &= \frac{E}{B_{1,n}} \epsilon_n^e \\ \epsilon_{2,n}^e &= \frac{B_{2,n}}{B_{1,n}} \epsilon_{1,n}^e \end{aligned} \quad (2.61)$$

In compression:

- Elastic forecast : $\epsilon_{1,n}^e = \epsilon_{1,n-1}^e + \Delta\epsilon_1$
- Damage criterion test: f
 - if $f < 0$, elastic behavior:

$$\begin{aligned} D_{1,n} &= D_{1,n-1} \\ D_{2,n} &= D_{2,n-1} \\ \epsilon_{1,n}^{an} &= \epsilon_{1,n-1}^{an} \\ \epsilon_{2,n}^{an} &= \epsilon_{2,n-1}^{an} \end{aligned} \quad (2.62)$$

– else, damaging behavior:

The evolution of the permanent strain must be estimate:

$$\begin{aligned} \dot{\epsilon}_{1,n}^{an} &= \gamma (d_{\epsilon_{n-1}}) * d_{\epsilon_{2,n}} \\ \dot{\epsilon}_{1,n}^{an} &= 2\gamma (d_{\epsilon_{n-1}}) * (D_{2,n} - D_{2,n-1}) \end{aligned} \quad (2.63)$$

The damage evolution becomes:

$$D_{2,n} = \frac{A (-\sqrt{2}\tilde{m}u_{n-1} (\epsilon - \epsilon_{1,n-1}^{an} - \dot{\epsilon}_{1,n}^{an}) - \kappa_0)}{2} \quad (2.64)$$

$$D_{2,n} = \frac{A(-\sqrt{2}\tilde{n}u_{n-1}(\epsilon - \epsilon_{1,n-1}^{an} + 2\gamma(d_{\epsilon_{n-1}})D_{2,n-1} - \kappa_0))}{2(\sqrt{2}A\tilde{n}u_{n-1}\gamma(d_{\epsilon_{n-1}}))} \quad (2.65)$$

$$\begin{aligned} \epsilon_{2,n}^{e \text{ trial}} &= \frac{B_{2,n-1}}{B_{1,n-1}} \epsilon_{1,n}^e \\ D_{1,n} &= D_{1,n-1} \\ \epsilon_{1,n}^{an} &= \epsilon_{1,n-1}^{an} + \left(\frac{-2k_D}{3K_D} + \frac{k_H}{K_H} \right) D_{1,n} \\ \epsilon_{2,n}^{an} &= \epsilon_{2,n-1}^{an} + \left(\frac{k_D}{3K_D} + \frac{k_H}{K_H} \right) D_{1,n} \end{aligned} \quad (2.66)$$

- Computation of σ and $\epsilon_{2,n}^e$:

$$\begin{aligned} B_{1,n} &= \frac{1+\nu}{9} \cdot \left(4 + \frac{2}{1-D_{2,n}} \right) + \frac{1-2\nu}{3} \\ B_{2,n} &= -\frac{1+\nu}{9} \cdot \left(2 + \frac{1}{1-D_{2,n}} \right) + \frac{1-2\nu}{3} \\ \sigma_n &= \frac{E}{B_{1,n}} \epsilon_n^e \\ \epsilon_{2,n}^e &= \frac{B_{2,n}}{B_{1,n}} \epsilon_{1,n}^e \end{aligned} \quad (2.67)$$

Fig. 2.26 describes the relationship between strain and stress in anisotropic damage model with permanent strain.

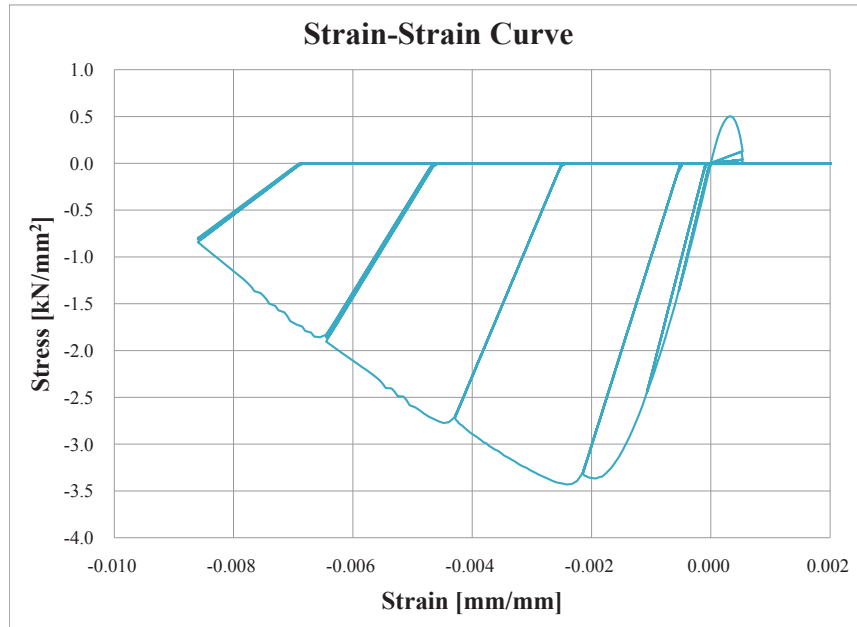


Figure 2.26: Strain-stress curve for anisotropic damage model with permanent strain

2.3 Bond-Slip

Slip due to longitudinal reinforcing bar near the column and from the anchorage can be easily determined if we assume a uniform bond stress u_b along the bars within the development length inside the footing or the beam-column

joint, (?). From equilibrium

$$u_b(\pi d_b)l_d = \frac{\pi d_b^2}{4} f_s \quad (2.68)$$

where d_b is the bar diameter, l_d is the development length over which the slip occurs, solving for l_d

$$l_d = \frac{d_b f_s}{4u_b} \quad (2.69)$$

Assuming that the maximum strain occurs at the end of the column, and a linear variation of strain along the development length, the integral of the strain curve will give the total bar slip at the footing-column interface or beam-column interface

$$S = \frac{\varepsilon_s l_d}{2} = \frac{f_s l_d}{2E_s} \quad (2.70)$$

Substituting Eq. 2.69 for l_d in the preceding equation

$$S = \frac{\varepsilon_s d_b f_s}{8u_b} \quad (2.71)$$

Finally, assuming that the cross section rotates about its neutral axis when slip occurs ($\phi_y = \varepsilon_y/(d - c)$), the displacement related to the bar slip at a point at a distance L from the column base will be

$$\Delta_{slip} = \frac{\phi_y d_b f_y L}{8u_b} \quad (2.72)$$

? presented a simplified bond model for bond stress in terms of the actual steel stress. It assumes a constant bond stress of $u_e = 12\sqrt{f'_c}$ prior to steel yielding, and another constant bond stress of $u_p = 6\sqrt{f'_c}$ past steel yielding, Fig. 2.27. Based on this assumption, the total bar slip at the edge of the anchorage is obtained by integrating the steel

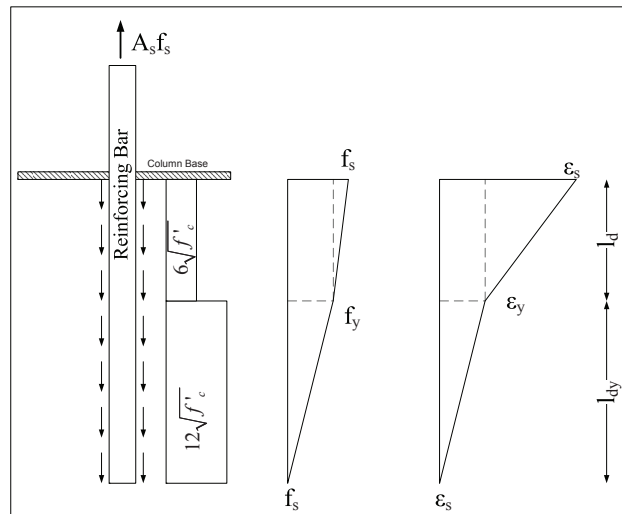


Figure 2.27: Longitudinal bar bond stress, steel stress, and strain profiles, (?)

strains over the embedded length.

? used this model to obtain a monotonic relation for bar slip versus bar stress at the column base. Assuming sufficient anchorage:

$$S_1 = \frac{\varepsilon_s f_s}{8u_e} d_b; \quad \varepsilon_s \leq \varepsilon_y \quad (2.73)$$

$$S_2 = \frac{\varepsilon_y f_y}{8u_e} d_b + \frac{(\varepsilon_s + \varepsilon_y)(f_s - f_y)}{8u_p} d_b; \quad \varepsilon_s > \varepsilon_y \quad (2.74)$$

where d_b is the bar diameter, u_e is the elastic bond stress = $12\sqrt{f'_c}$ (psi), and u_p is the plastic bond stress = $6\sqrt{f'_c}$. The following Matlab code generates a plot of normalized stress versus bond slip.

Listing 2.2: Bond-slip

```

1 clear
2 '+++++'
3 %% Input parameters
4 d_b=3/8; % bar diameter in inches
5 f_c=4000; % compressive strength (psi)
6 E_s=27300000; % psi
7 b=0.01; %
8 f_y=64000;% original yield stress in psi
9 f_y=1.25*f_y; % yield stress increased by 25% for rate effect
10 %f_u=100000; % ksi
11 %
12 epsilon_y=f_y/E_s;
13 %% Bond
14 u_e=12*sqrt(f_c);
15 u_p=6*sqrt(f_c);
16 %
17 k=0;
18 epsilon_final=30*epsilon_y;
19 delta_epsilon=epsilon_final/100;
20 for epsilon_s=0:delta_epsilon:epsilon_final
21     k=k+1;
22     if epsilon_s <= epsilon_y
23         f_s=epsilon_s*E_s;
24         slip_y=epsilon_s*f_s*d_b/(8*u_e);
25         slip(k)=slip_y;
26     else
27         f_s=epsilon_y*E_s+b*(epsilon_s-epsilon_y)*E_s;
28         slip_u=epsilon_y*f_y*d_b/(8*u_e)+(epsilon_s+epsilon_y)*(f_s-f_y)*d_b/(8*u_p);
29         slip(k)=slip_u;
30     end
31     normalized_stress(k)=f_s/f_y;
32 end
33 plot(slip, normalized_stress, 'linewidth', 2); grid; xlim([0, max(slip)]);
34 xlabel('Slip [in]', 'fontsize', 14); ylabel('f_s/f_y', 'fontsize', 14);
35 print -deps 'bond-slip-curve.eps'

```

This code generates the stress slip curve shown in Fig. 2.27.

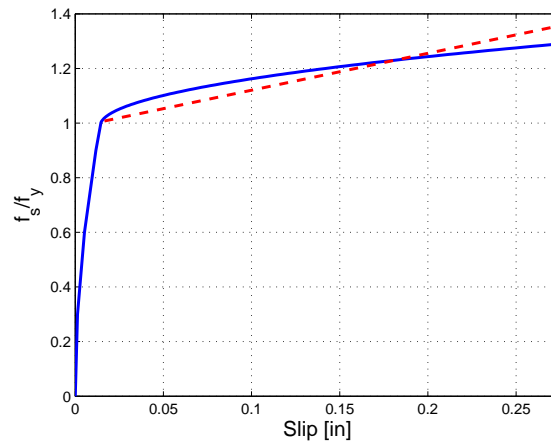


Figure 2.28: Stress slip curve

2.4 Lumped Plasticity Model

To be completed later

2.5 Constitutive models for zero length and zero length section elements

2.5.1 Zero Length Element

Zero length elements are used primarily for lumped plasticity models. In fiber models, they are used to provide a finite shear stiffness, whereas the axial and flexural ones can be considered as rigid (no corresponding entries in Mercury implies rigid connection).

$$K_S = \frac{\alpha G A_g}{L} \quad (2.75)$$

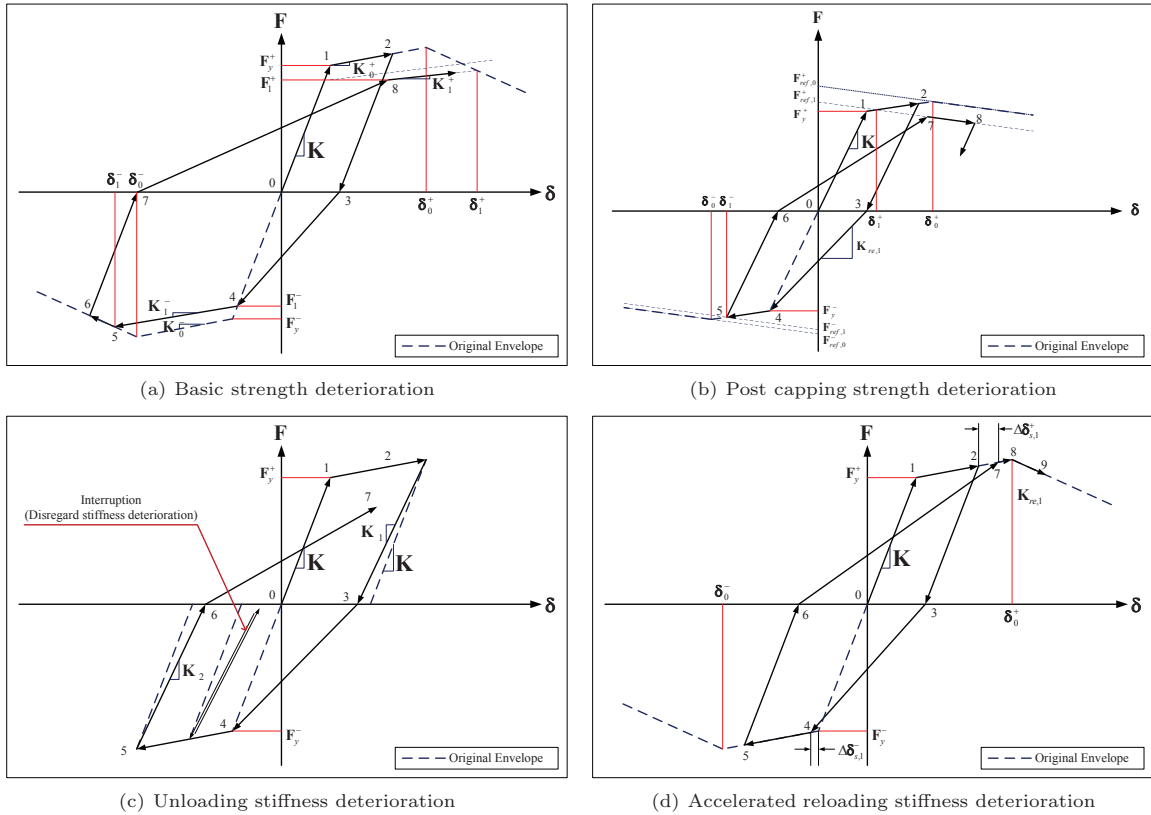


Figure 2.29: Hysteretic model with strength and stiffness deterioration, (?)

where α is the shape section coefficient (5/6 for rectangular section), G is the shear modulus ($G = E_c/2(1 + \nu)$); Concrete Poisson ratio is about 0.25.

Future release of Mercury, should have zero length elements enriched by limit state material properties to act as a fuse to possible failure.

2.5.2 Zero length section element

This section is an adaptation of ?

Zero length section element should be used only when fiber elements are used if we want to capture the bond slip between concrete and rebar. Though many models have been proposed for bond-slip, we will adopt the model of ? as it is the most rational one in handling the shifting position of the neutral axis (essential to determine the bond slip displacement).

From Fig. 2.28 we have a nonlinear post-peak response, and we need to linearize it, and then solve for u_p (which will be different than the previously given value of $6\sqrt{f'_c}$ suitable for the nonlinear hardening segment).

We seek to have the linearized segment intersect the nonlinear one at $1.25f_y$, hence

$$\varepsilon_u = \varepsilon_y + 0.25 \frac{f_y}{E_s/h} = \frac{f_y}{E_s} + 0.25h \frac{f_y}{E_s} = 0.26h\varepsilon_y \quad (2.76)$$

where h is the hardening parameter set to 100. Substituting in Eq. 2.74

$$S_u = \frac{\varepsilon_y f_y}{8u_e} d_b + \frac{(26\varepsilon_y + \varepsilon_y)(1.25f_y - f_y)}{8u_p} d_b \quad (2.77)$$

$$= S_y + \frac{27\varepsilon_y \times 0.25f_y}{8u_p} d_b \quad (2.78)$$

$$= S_y + \frac{6.75\varepsilon_y f_y}{8u_p} d_b \quad (2.79)$$

We can reasonably assume that

$$S_u = \varepsilon_s \frac{S_y}{\varepsilon_y} = 26S_y \quad (2.80)$$

Substituting in Eq. 2.79

$$26S_y = S_y + \frac{6.75\varepsilon_y f_y}{8u_p} d_b \quad (2.81)$$

or

$$25 \frac{\varepsilon_y f_y}{8u_e} d_b = \frac{6.75\varepsilon_y f_y}{8u_p} d_b \quad (2.82)$$

$$u_p = \frac{6.75}{25} u_e \quad (2.83)$$

$$= \frac{6.75}{25} 12\sqrt{f'_c} \quad (2.84)$$

$$= 3.24\sqrt{f'_c} \quad (2.85)$$

u_p may be used in so-called limit state elements to assess bond slip induced failure.

Irrespective of which steel model is used in the beam-column, it is recommended to use the bilinear one for this element. Using a bilinear model, with $h = 0.01$ will be equivalent to having a bar slip curve such that the second segment intersect the exact one at $f_s = 1.25f_y$ with $u_p = 3.24\sqrt{f'_c}$.

In the steel bilinear model, the Young's modulus should be adjusted to reflect bond slip, by replacing E_s by E_{bs} which is equal to

$$E_{bs} = \frac{f_y}{S_y} \quad (2.86)$$

It should be noted that inherent in this assumption is a *unit length* of the zero length element.

Finally, to maintain the same material stiffness ratio between bar-slip steel in the zero length section element, and the one in the frame element (longitudinal steel), we multiply the bar slip concrete material strains by E_s/E_{bs} .

The concrete properties for the zero length section element are such that the location of the neutral axis in the beam-column element and the zero length fiber section is the same. Thus

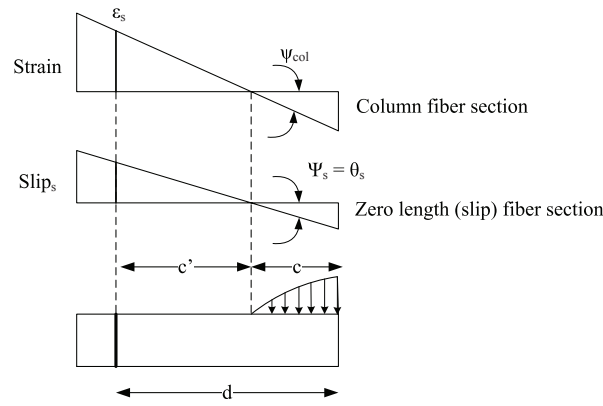


Figure 2.30: Compatibility of NA between beam-column and zero length fiber section

$$\left. \begin{aligned} \theta_s &= \frac{S_s}{c'} \\ \Psi_{col} &= \frac{\epsilon_s}{c'} \end{aligned} \right\} \theta_s = \Psi_{col} \frac{S_s}{\epsilon_s} \tag{2.87}$$

Hence all fiber strains (corresponding to steel and concrete) in the zero length section must be scaled by $\frac{S_s}{\epsilon_s}$. This can be easily achieved in altering the material input data such that

1. All stress values remain unchanged
2. Strains are scaled by $\frac{S_y}{\epsilon_y}$

2.5.3 Summary

Fig. 2.31 highlight the modeling of the zero length element and section.

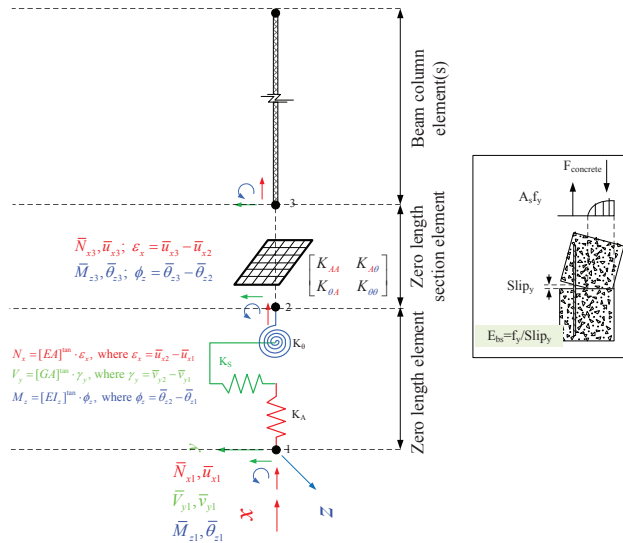


Figure 2.31: Zero length element and section

2.6 Summary

This chapter has presented the details of the constitutive models coded in Mercury. The model selected are well known and tested and would enable the analysis of complex steel or reinforced concrete frame.

Chapter 3

Structural Modeling

This chapter addresses the major issues in modeling structure for a nonlinear static or transient analysis.

Table. ?? provides a concise summary of the example validation problems. In it, we adopt the following shortcuts:

- SBC: Stiffness-based 2D beam-column
- FBC1: Flexibility-based 2D beam-column with element iteration
- FBC2: Flexibility-based 2D beam-column without element iteration
- Damage1: Anisotropic damage 1D material model without permanent strain
- Damage2: Anisotropic damage 1D material model with permanent strain
- ModKP: Modified Kent-Park material model
- ModGMP: Modified Giuffre-Monegotto-Pinto material model
- Disp(disp): Displacement.

3.1 Truss, Material Nonlinearity, Static and Dynamic

This is a transient nonlinear analysis of a truss based on the HHT algorithm (α method), and the elements are modeled by the modified Giuffre-Monegotto-Pinto materials. Truss, and results are shown in Fig. 3.1.

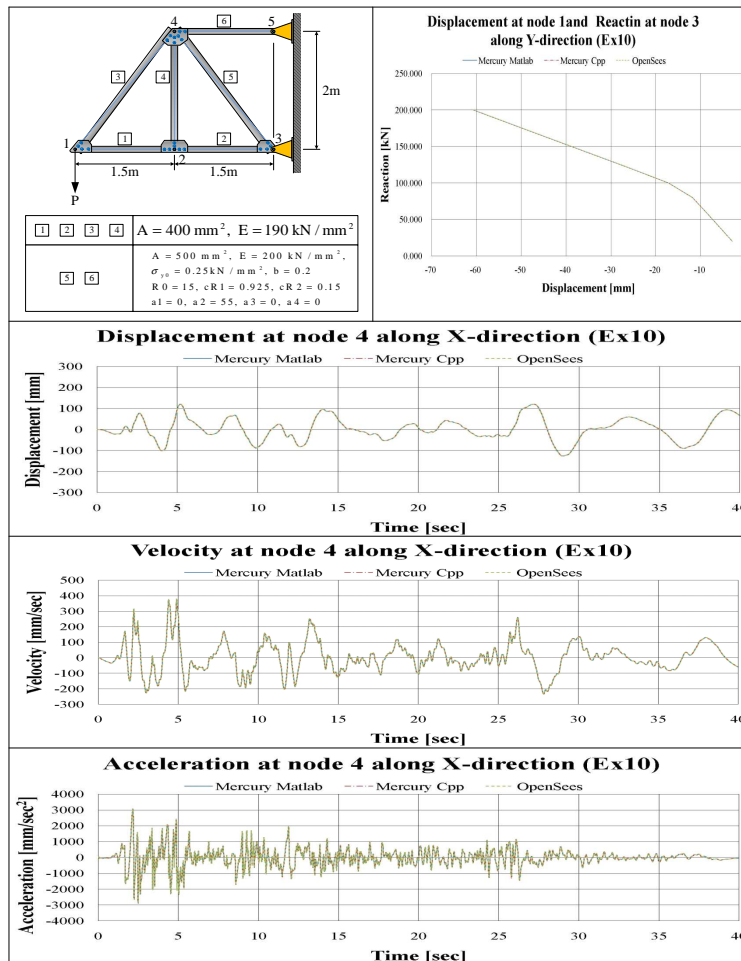


Figure 3.1: Ex10: Truss, ModGMP material, load control, HHT

In static analysis, incremental forces that increase by -20kN from -20kN to -200kN at node 1 along the Y direction are applied.

Document Section	ID	Element	Zero Length	Formulation	Section	
3.1	Ex10	Truss		Stiffness	General	
3.2	Ex11					
	Ex12					
3.3	Ex16					
	Ex17					
	Ex18					
3.4	Ex19	Beam Column		Flexibility, With Iteration	Fiber	
	Ex20					
	Ex21					
3.5	Ex22			Element	Stiffness Based	General
	Ex23					
3.6	Ex24			Section	Flexibility, iteration	Fiber
3.7	Ex25					
	Ex26					
3.8	Ex27			Element &	Flexibility (itera- tion)	Fiber
3.9	Ex29	Section				
Document Section	ID	Constitutive Model	Hardening	Static Load control	Transient	
3.1	Ex10	Elastic and ModGMP	Kinematic	Stiffness	HHT	
3.2	Ex11	Damage		Cyclic disp control		
	Ex12	ModKP				
3.3	Ex16	Simplified Bilinear	No			
	Ex17	Simplified Bilinear	Isotropic			
	Ex18	ModGMP	Kinematic			
	Ex19	ModGMP	Kinematic & Isotropic			
3.4	Ex20	Classical Hardening	Isotropic	Load control	HHT	
	Ex21		Kinematic		Shing	
	Ex22		Kinematic & Isotropic			
3.5	Ex23	Elastic and Simplified Bilinear	No	Pushover		
3.6	Ex24	Elastic and Bilinear	Isotropic			
3.7	Ex25	Classical Hardening			Cyclic disp control	HHT/Shing
	Ex26					
3.8	Ex27	ModGMP and ModKP	Kinematic &			
3.9	Ex29	Multiple				

- SBC: Stiffness-based 2D beam-column Isotropic
 FBC1: Flexibility-based 2D beam-column with element iteration
 FBC2: Flexibility-based 2D beam-column without element iteration
 Damage1: Anisotropic damage 1D material model without permanent strain
 Damage2: Anisotropic damage 1D material model with permanent strain
 ModKP: Modified Kent-Park material model
 ModGMP: Modified Giuffre-Monegotto-Pinto material model
 Disp(displacement): Displacement.

Table 3.1: Features of Example Problems

In the second part, the transient analysis without self-weight and nodal forces is performed with HHT integration scheme.

3.1.1 MATLAB

Listing 3.1: MATLAB; Truss

```

1 %=====
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : September 2009.
6 % File name: Ex10.m
7 %=====
8 %=====
9 % Description
10 % 1. Static and transient analysis
11 % 2. Load control
12 % 3. Iterative method
13 % 4. Simple 2D truss element
14 % 5. General section
15 % 6. Elastic and modified Giuffre-Monegotto-Pinto material
16 %=====
17 % Select type of analysis.
18 % AnalysisType = 1: Static analysis
19 % AnalysisType = 2: Transient analysis
20 AnalysisType = 2;
21 %=====
22 % Preface
23 Unit = {'kN', 'mm'};
24 % ndim, ndofpn
25 StrMode = {2, 2};
26 %=====
27 % Control block
28 Iteration = {'static', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
29                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
30                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
31                        }
32             'transient', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
33                          {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
34                          {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
35                          }
36             };
37 if (AnalysisType == 2)
38     Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
39     eigens = {0.02, 0.02};
40 end
41 %=====
42 % Geometry block
43 % nodtag, x, y
44 nodcoord = {1, 0, 0;
45             2, 1500, 0;
46             3, 3000, 0;
47             4, 1500, 2000;
48             5, 3000, 2000};
49 % nodtag, x, y
50 constraint = {3, 1, 1;
51              5, 1, 1};
52 %=====
53 % Element block
54 % {eletag, 'Simple2DTruss', in, jn, sectag}
55 elements = { {1, 'Simple2DTruss', 1, 2, 1};
56             {2, 'Simple2DTruss', 2, 3, 1};
57             {3, 'Simple2DTruss', 1, 4, 1};
58             {4, 'Simple2DTruss', 2, 4, 1};
59             {5, 'Simple2DTruss', 3, 4, 2};
60             {6, 'Simple2DTruss', 4, 5, 2} };
61 %=====
62 % Section block
63 % sectag, 'General', {mattag, A, Ix, Iy, Iz}
64 sections = { 1, 'General', {1, 400, 0, 0, 0};
65             2, 'General', {2, 500, 0, 0, 0}};
66 %=====
67 % Material block
68 % mattag, 'Elastic', E, G, density
69 materials = { {1, 'Elastic', 190, 0, 7850*10^-9}
70             {2, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
71              {2, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 7850*10^-9, 0, 55, 0, 55} };
72 %=====
73 % Force block
74 if (AnalysisType == 1)
75     forces = { 1, 'Static', {'NodalForces', {1, 2, -20}};
76             2, 'LoadCtrl', {1, 2, {-40, -60, -80, -100, -120, -140, -160, -180, -200}} };
77 elseif (AnalysisType == 2)
78     ga = load('ElCentro_g_0.01_Matlab.txt');
79     nga = size(ga, 1);
80     for i = 1:nga
81         groundacceleration{i,1} = ga(i,1);
82         groundacceleration{i,2} = ga(i,2);
83         groundacceleration{i,3} = ga(i,3);
84     end
85     forces = { 1, 'Static', {'NodalForces', {1, 2, 0}};
86             2, 'Acceleration', {9810, groundacceleration} };
87 end
88 %=====

```

3.1.2 C++

Listing 3.2: C++; Truss

```

1  --- *****
2  --- AnalysisType = 1: Static analysis
3  --- AnalysisType = 2: Transient analysis
4  AnalysisType = 2
5  --- *****
6  %nodes = { {1, 0, 0, 'mass', 6.28, 6.28},
7             {2, 1500, 0, 'mass', 7.85, 7.85},
8             {3, 3000, 0},
9             {4, 1500, 2000, 'mass', 14.915, 14.915},
10            {5, 3000, 2000} }
11 ---
12 Simple2DTruss = 'truss2d'
13 elements = { {1, Simple2DTruss, 1, 2, 400, 1},
14             {2, Simple2DTruss, 2, 3, 400, 1},
15             {3, Simple2DTruss, 1, 4, 400, 1},
16             {4, Simple2DTruss, 2, 4, 400, 1},
17             {5, Simple2DTruss, 3, 4, 500, 2},
18             {6, Simple2DTruss, 4, 5, 500, 2} }
19 ---[tag, 'modifiedGMPsteel', E, rho, fy, b_ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma-init]
20 materials = { {1, 'elastic', 190, 0, 0};
21             {2, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0, 55, 0, 55, 0} }
22 --- *****
23 model = StructureModel(2,2)
24 model:addNodes(nodes)
25 model:addMaterials(materials)
26 model:addElements(elements)
27 model:constrainNode(3,1,1)
28 model:constrainNode(5,1,1)
29 --- *****
30 --- Static analysis
31 if (AnalysisType == 1) then
32   print("Static analysis started\n")
33   staticloading = LoadDescription()
34   staticloading:addLoad({'incrementalnodalload', 1, 2, -20, -40, -60, -80, -100, -120, -140, -160, -180, -200})
35   --- *****
36   displ = {}
37   function displperstep(increment)
38     dx1, dy1 = model:nodeDisplacements(1)
39     dx2, dy2 = model:nodeDisplacements(2)
40     dx4, dy4 = model:nodeDisplacements(4)
41     table.insert(displ, dx1)
42     table.insert(displ, dy1)
43     table.insert(displ, dx2)
44     table.insert(displ, dy2)
45     table.insert(displ, dx4)
46     table.insert(displ, dy4)
47   end
48   ---
49   react = {}
50   function reactperstep(increment)
51     fx3, fy3 = model:nodeRestoringForces(3)
52     fx5, fy5 = model:nodeRestoringForces(5)
53     table.insert(react, fx3)
54     table.insert(react, fy3)
55     table.insert(react, fx5)
56     table.insert(react, fy5)
57   end
58   ---
59   solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
60   analysis = StaticAnalysis(solver)
61   analysis:setStructureModel(model)
62   analysis:addcallback(displperstep, "increment")
63   analysis:addcallback(reactperstep, "increment")
64   analysis:solve(staticloading)
65   --- *****
66   --- Set output file
67   function writedata6(x, fname)
68     local f = assert(io.open(fname, 'w'))
69     local writenl = 0
70     for i, v in ipairs(x) do
71       f:write(v, " ")
72       writenl = writenl + 1
73       --- length of row size: writenl
74       if (writenl > 5) then
75         writenl = 0
76         f:write("\n")
77       end
78     end
79     f:close()
80   end
81   ---
82   function writedata4(x, fname)
83     local f = assert(io.open(fname, 'w'))
84     local writenl = 0
85     for i, v in ipairs(x) do
86       f:write(v, " ")
87       writenl = writenl + 1
88       --- length of row size: writenl
89       if (writenl > 3) then
90         writenl = 0
91         f:write("\n")
92       end
93     end
94     f:close()
95   end
96   ---
97   writedata6(displ, 'Ex10StaticNodalDisp_1_2_4.dat')
98   writedata4(react, 'Ex10StaticReact_3_5.dat')
99   print("Static analysis ended\n")
100 end
101 --- *****
102 if (AnalysisType == 2) then
103   print("Transient analysis started\n")
104   earthquakeloading = LoadDescription()
105   accelamp = 9810
106   earthquakeloading:addLoad({'groundmotion', 'ElCentro_g_0_01_OpenSees.txt', dt=0.01', 1, accelamp})
107   --- *****

```



```

108 displ = {}
109 function displertime(time)
110     dx1,dy1 = model.nodeDisplacements(1)
111     dx2,dy2 = model.nodeDisplacements(2)
112     dx4,dy4 = model.nodeDisplacements(4)
113     table.insert(displ, dx1)
114     table.insert(displ, dy1)
115     table.insert(displ, dx2)
116     table.insert(displ, dy2)
117     table.insert(displ, dx4)
118     table.insert(displ, dy4)
119 end
120
121 react = {}
122 function reactvertime(time)
123     fx3,fy3 = model.nodeRestoringForces(3)
124     fx5,fy5 = model.nodeRestoringForces(5)
125     table.insert(react, fx3)
126     table.insert(react, fy3)
127     table.insert(react, fx5)
128     table.insert(react, fy5)
129 end
130
131 veloc = {}
132 function velocvertime(time)
133     vx1,vy1 = model.nodeVelocities(1)
134     vx2,vy2 = model.nodeVelocities(2)
135     vx4,vy4 = model.nodeVelocities(4)
136     table.insert(veloc, vx1)
137     table.insert(veloc, vy1)
138     table.insert(veloc, vx2)
139     table.insert(veloc, vy2)
140     table.insert(veloc, vx4)
141     table.insert(veloc, vy4)
142 end
143
144 accel = {}
145 function accelvertime(time)
146     ax1,ay1 = model.nodeAccelerations(1)
147     ax2,ay2 = model.nodeAccelerations(2)
148     ax4,ay4 = model.nodeAccelerations(4)
149     table.insert(accel, ax1)
150     table.insert(accel, ay1)
151     table.insert(accel, ax2)
152     table.insert(accel, ay2)
153     table.insert(accel, ax4)
154     table.insert(accel, ay4)
155 end
156
157 solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-6, iterations=10})
158 transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
159 transientanalysis:addcallback(displertime, "timestep")
160 transientanalysis:addcallback(reactvertime, "timestep")
161 transientanalysis:addcallback(velocvertime, "timestep")
162 transientanalysis:addcallback(accelvertime, "timestep")
163 model:setRayleighCoefficients(0.025533,0.007826)
164 transientanalysis:solve(4000)
165
166 function writedata6(x, fname)
167     local f = assert(io.open(fname,'w'))
168     local writenl = 0
169     for i,v in ipairs(x) do
170         f:write(v, " ")
171         writenl = writenl + 1
172         -- length of row size: writenl
173         if (writenl > 5) then
174             writenl = 0
175             f:write("\n")
176         end
177     end
178     f:close()
179 end
180
181 function writedata4(x, fname)
182     local f = assert(io.open(fname,'w'))
183     local writenl = 0
184     for i,v in ipairs(x) do
185         f:write(v, " ")
186         writenl = writenl + 1
187         -- length of row size: writenl
188         if (writenl > 3) then
189             writenl = 0
190             f:write("\n")
191         end
192     end
193     f:close()
194 end
195
196 writedata6(displ, 'Ex10TransientNodalDisp_1_2_4.dat')
197 writedata4(react, 'Ex10TransientReact_3_5.dat')
198 writedata6(veloc, 'Ex10TransientNodalVelo_1_2_4.dat')
199 writedata6(accel, 'Ex10TransientNodalAcce_1_2_4.dat')
200 print("Transient analysis ended\n")
201 end
202

```

3.2 Uniaxial Concrete Nonlinear Element, Cyclic Displacement

These examples assume concrete members and seek to compare the modified Kent-Park model with the anisotropic damage model with permanent strain when subjected to cyclic displacement statically, Fig. 3.2. Fig. 3.3 describes the static analysis with cyclic displacements applied at node 2 in the X direction.

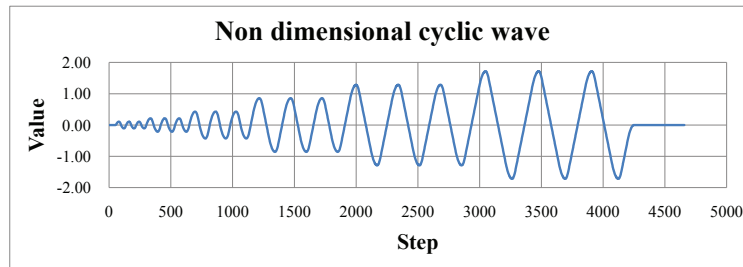


Figure 3.2: Cyclic wave without dimension

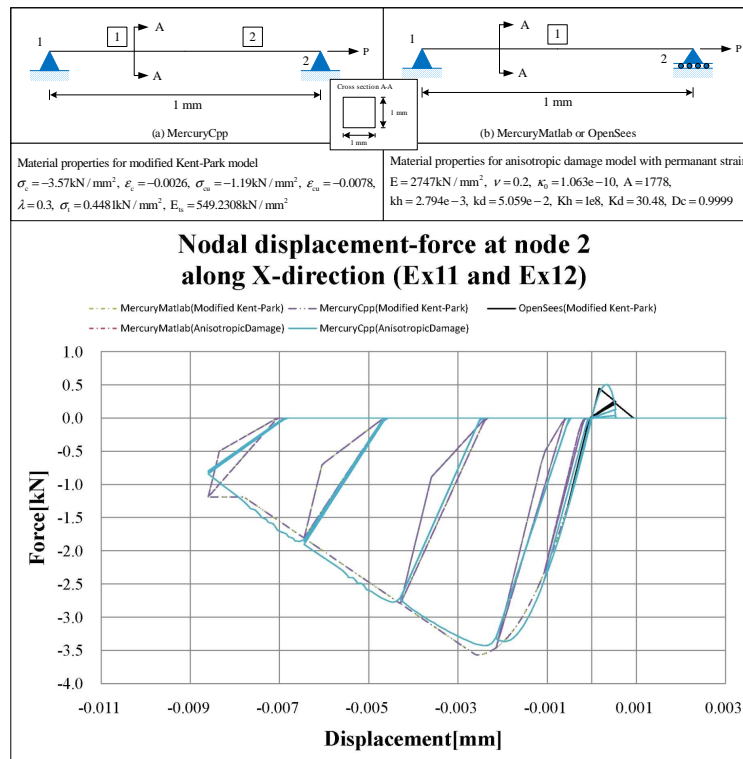


Figure 3.3: Examples 11- 12

3.2.1 MATLAB

```

1 %=====
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : November 2009.
6 % File name: Ex11to12.m
7 %=====
8 %
9 % Description
10 % 1. Static analysis
11 % 2. Displacement control
12 % 3. Iterative method
13 % 4. Simple 2D truss element
14 % 5. General section
15 % 6. Anisotropic damage material(Ex11)
16 %    and modified Kent-Park material(Ex12)
17 %=====
18 % Select material type
19 % MatType = 1: anisotropic damage material with permanent strain (Ex11)
20 % MatType = 2: modified Kent-Park material (Ex12)
21 MatType = 1;
22 %=====
23 % Preface
24 Unit = {'kN', 'mm'};
25 StrMode = {2, 2};
26 %=====
27 % Control block
28 Iteration = {'static', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
29                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
30                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
31                        }
32                };
33 %=====
34 % Geometry block
35 nodcoord = {1, 0, 0;
36            2, 1, 0};
37 constraint = {1, 1, 1;
38             2, 0, 1};
39 %=====
40 % Element block
41 elements = { {1, 'Simple2DTruss', 1, 2, 1} };
42 %=====
43 % Section block
44 sections = { 1, 'General', {1, 1, 0, 0, 0} };
45 %=====
46 % Material block
47 if (MatType == 1)
48     % {tag, 'AnisotropicDamage', E, nu, kappa0, A, kh, kd, Kh, Kd, Dc, density};
49     materials = { {1, 'AnisotropicDamage', 2.747e+003, 0.2, 1.063e-010, 1.778e+003, 2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0};
50 end
51 if (MatType == 2)
52     % {tag, 'ModKP', sc, ec, scu, ecu, lambda, st, Ets, density};
53     materials = { {1, 'ModKP', -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077, 549.2307692, 0} };
54 end
55 %=====
56 % Force block
57 DispInput = load('cyclicwave.txt');
58 row = size(DispInput, 1);
59 for i = 1:row
60     DispCell[i] = 0.005*DispInput(i);
61 end
62 % forsetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
63 forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
64          2, 'DispCtrl', {2, 1, DispCell} };
65 %=====

```

3.2.2 C++

```

1 --- *****
2 --- MatType = 1: Anisotropic damage model with permanent strain
3 --- MatType = 2: Modified Kent-Park model
4 MatType = 2
5 --- *****
6 nodes = { {1, 0, 0},
7          {2, 1, 0} }
8 --- *****
9 Simple2DTruss = 'truss2d'
10 elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = { {1, 'anisotropicdamage2', 2.747e+003, 0, 0.2, 1.063e-010, 1.778e+003, 2.794e-003, 5.059e-002, 1.000e+008, 3.048e+001, 0};
14 end
15 if (MatType == 2) then
16     concretemat = 'ConcreteLinearTensionSoftening'
17     materials = { {1, concretemat, 549.2307692, 0, -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077} };
18 end
19 --- *****
20 model = StructureModel(2,2)
21 model:addNodes(nodes)
22 model:addMaterials(materials)
23 model:addElements(elements)
24 model:constrainNode(1,1,1)
25 model:constrainNode(2,1,1)
26 --- *****
27 function generateincrementalload()
28     format tag node dof
29     local loadform = {'incrementalnodaldisplacement', 2, 1}
30     local f = assert(io.open('cyclicwave.txt', 'r'))
31     local n = f:read('*number')
32     while (n ~= nil) do

```

```

33 | table.insert(loadform, 0.005*n)
34 | n = f:read("*number")
35 | end
36 | f:close()
37 | return loadform
38 | end
39 | --- *****
40 | staticloading = LoadDescription()
41 | --- format: 'staticnodalload' <node> <dof> <amplitude>
42 | l = generateincrementalload()
43 | staticloading:addLoad(l)
44 | --- *****
45 | --- Static analysis
46 | print(" Static analysis started\n")
47 | displ = {}
48 | function displperstep(increment)
49 |     dx2,dy2 = model.nodeDisplacements(2)
50 |     table.insert(displ, dx2)
51 | end
52 | ---
53 | react = {}
54 | function reactperstep(increment)
55 |     fx1,fy1 = model.nodeRestoringForces(1)
56 |     table.insert(react, fx1)
57 | end
58 | ---
59 | solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
60 | analysis = StaticAnalysis(solver)
61 | analysis:setStructureModel(model)
62 | --- analysis:addcallback(displperstep,"increment")
63 | --- analysis:addcallback(reactperstep,"increment")
64 | analysis:solve(staticloading)
65 | --- *****
66 | --- Set output file
67 | function writedata1(x, fname)
68 |     local f = assert(io.open(fname, 'w'))
69 |     local writenl = 0
70 |     for i,v in ipairs(x) do
71 |         f:write(v, " ")
72 |         writenl = writenl + 1
73 |         --- length of row size: writenl
74 |         if (writenl > 0) then
75 |             writenl = 0
76 |             f:write("\n")
77 |         end
78 |     end
79 |     f:close()
80 | end
81 | ---
82 | if (MatType == 1) then
83 |     writedata1(displ, 'Ex11StaticDamageNodalDisp_2.dat')
84 |     writedata1(react, 'Ex11StaticDamageReact_1.dat')
85 | end
86 | if (MatType == 2) then
87 |     writedata1(displ, 'Ex12StaticModKPNodalDisp_2.dat')
88 |     writedata1(react, 'Ex12StaticModKPSReact_1.dat')
89 | end
90 | print(" Static analysis ended\n")

```

3.3 Uniaxial Steel Nonlinear Element, Cyclic Displacement

Bilinear and modified Giuffre-Monegotto-Pinto constitutive models with cyclic displacement are examined next, Fig. 3.2. Fig. ?? shows the results of the static analysis with cyclic displacements applied at node 2 in the X direction. Ex16 has bilinear material without isotropic hardening, Ex17 has bilinear material with isotropic hardening, Ex18 has modified Giuffre-Monegotto-Pinto material without isotropic hardening, and Ex19 has modified Giuffre-Monegotto-Pinto material with isotropic hardening material.

3.3.1 MATLAB

```

1 | %-----
2 | % Mercury Matlab Version 1.0.1
3 | % Written by Dae-Hung Kang, CU-NEES
4 | % Copyright 2009, CU-NEES
5 | % Written : October 2009.
6 | % File name: Ex16to17.m
7 | %-----
8 | % Description
9 | % 1. Static analysis
10 | % 2. Displacement control
11 | % 3. Iterative method
12 | % 4. Simple 2D truss element
13 | % 5. General sections
14 | % 6. Bilinear material
15 | %-----
16 | % A36 Steel properties
17 | % Density of 7.8 g/cm
18 | % Minimum yield strength of 250 MPa(0.25 GPa)
19 | % Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
20 | %-----
21 | % Select material type
22 | % MatType = 1: Bilinear matierial without isotropic hardening
23 | % MatType = 2: Bilinear matierial with isotropic hardening
24 | MatType = 2;
25 | %-----
26 | % Preface
27 | Unit = {'kN', 'mm'};
28 | StrMode = {2, 2};
29 | %-----
30 | % Control block
31 | Iteration = {'static', { 'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
32 |               {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};

```

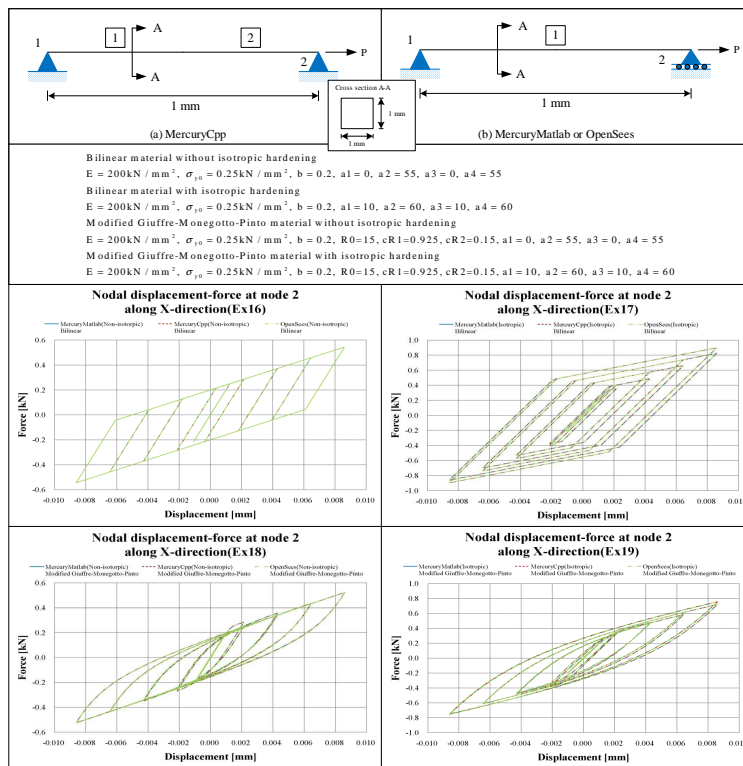


Figure 3.4: Examples 16- 19

```

33                                     {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
34                                     });
35 -----
36 % Geometry block
37 nodcoord = {1, 0, 0};
38           {2, 1, 0};
39 constraint = {1, 1, 1};
40             {2, 0, 1};
41 -----
42 % Element block
43 elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
44 -----
45 % Section block
46 sections = { {1, 'General', {1, 1, 0, 0, 0}}; };
47 -----
48 % Material block
49 if MatType == 1
50     materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 0, 55, 0, 55} };
51 elseif MatType == 2
52     materials = { {1, 'Bilinear', 200, 0.25, 0.2, 0, 10, 60, 10, 60} };
53 end
54 -----
55 % Force block
56 DispInput = load('cyclicwave.txt');
57 row = size(DispInput,1);
58 for i = 1:row
59     DispCell{i} = 0.005*DispInput(i);
60 end
61 % forcelag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
62 forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
63           2, 'DispCtrl', {2, 1, DispCell} };
64 % -----

```

3.3.2 C++

```

1  --- *****
2  --- MatType = 1: Bilinear material without isotropic hardening
3  --- MatType = 2: Bilinear material with isotropic hardening
4  MatType = 2
5  --- *****
6  nodes = { {1, 0, 0},
7            {2, 1, 0} }
8  --- *****
9  Simple2DTruss = 'truss2d'
10 elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = {{tag, 'bilinear', E, rho, fy, b_ratio, a1, a2, a3, a4}}
14     materials = { {1, 'bilinear', 200, 0, 0.25, 0.2, 0, 55, 0, 55} }

```

```

15 end
16 if (MatType == 2) then
17     materials = { {1,'bilinear', 200, 0, 0.25, 0.2, 10, 60, 10, 60} }
18 end
19 -----
20 model = StructureModel(2,2)
21 model:addNodes(nodes)
22 model:addMaterials(materials)
23 model:addElements(elements)
24 model:constrainNode(1,1,1)
25 model:constrainNode(2,1,1)
26 -----
27 function generateincrementalload()
28     -- format:          tag          node dof
29     local loadform = {'incrementalnodaldisplacement', 2, 1}
30     local f = assert(io.open('cyclicwave.txt','r'))
31     local n = f:read("*number")
32     while (n ~= nil) do
33         table.insert(loadform, 0.005*n)
34         n = f:read("*number")
35     end
36     f:close()
37     return loadform
38 end
39 -----
40 staticloading = LoadDescription()
41 -- format: 'staticnodalload' <node> <dof> <amplitude>
42 l = generateincrementalload()
43 staticloading:addLoad(l)
44 -----
45 -- Static analysis
46 print("Static analysis started\n")
47 displ = {}
48 function displperstep(increment)
49     dx2,dy2 = model:nodeDisplacements(2)
50     table.insert(displ, dx2)
51 end
52 --
53 react = {}
54 function reactperstep(increment)
55     fx1,fy1 = model:nodeRestoringForces(1)
56     table.insert(react, fx1)
57 end
58 --
59 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
60 analysis = StaticAnalysis(solver)
61 analysis:setStructureModel(model)
62 analysis:addcallback(displperstep,"increment")
63 analysis:addcallback(reactperstep,"increment")
64 analysis:solve(staticloading)
65 -----
66 -- Set output file
67 function writedata1(x, fname)
68     local f = assert(io.open(fname,'w'))
69     local writenl = 0
70     for i,v in ipairs(x) do
71         f:write(v, " ")
72         writenl = writenl + 1
73         -- length of row size: writenl
74         if (writenl > 0) then
75             writenl = 0
76             f:write("\n")
77         end
78     end
79     f:close()
80 end
81 --
82 if (MatType == 1) then
83     writedata1(displ,'Ex16StaticNonIsotropicNodalDisp_2.dat')
84     writedata1(react,'Ex16StaticNonIsotropicReact_1.dat')
85 end
86 if (MatType == 2) then
87     writedata1(displ,'Ex17StaticIsotropicNodalDisp_2.dat')
88     writedata1(react,'Ex17StaticIsotropicReact_1.dat')
89 end
90 print("Static analysis ended\n")

```

```

1 -----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex18to19.m
7 -----
8 % Description
9 % 1. Static analysis
10 % 2. Displacement control
11 % 3. Iterative method
12 % 4. Simple 2D truss element
13 % 5. General sections
14 % 6. Modified Giuffre-Monegotto-Pinto material
15 -----
16 % A36 Steel properties
17 % Density of 7.8 g/cm
18 % Minimum yield strength of 250 MPa(0.25 GPa)
19 % Ultimate tensile strength of 400-550 MPa(0.4-0.55 GPa)
20 -----
21 % Select material type
22 % MatType = 1: ModGMP material without isotropic hardening
23 % MatType = 2: ModGMP material with isotropic hardening
24 MatType = 2;
25 -----
26 % Preface
27 Unit = {'kN', 'mm'};
28 StrMode = {2, 2};

```

```

29 %-----
30 % Control block
31 Iteration = {'static', { {'NewtonRaphson', 10, 1.0e-8, 'DisplNorm'};
32                        {'ModifiedNewtonRaphson', 20, 1.0e-8, 'EnrgyNorm'};
33                        {'InitialStiffness', 30, 1.0e-8, 'ForceNorm'};
34                        }};
35 %-----
36 % Geometry block
37 nodcoord = {1, 0, 0;
38            2, 1, 0};
39 constraint = {1, 1, 1;
40             2, 0, 1};
41 %-----
42 % Element block
43 elements = { {1, 'Simple2DTruss', 1, 2, 1}; };
44 %-----
45 % Section block
46 sections = { 1, 'General', {1, 1, 0, 0, 0}};
47 %-----
48 % Material block
49 if MatType == 1
50     mattag, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4
51     materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 0, 55, 0, 55} };
52 elseif MatType == 2
53     materials = { {1, 'ModGMP', 200, 0.25, 0.2, 15, 0.925, 0.15, 0, 10, 60, 10, 60} };
54 end
55 %-----
56 % Force block
57 DispInput = load('cyclicwave.txt');
58 row = size(DispInput,1);
59 for i = 1:row
60     DispCell{i} = 0.005*DispInput(i);
61 end
62 %         forcetag, 'Static', {'NodalForces', {nodnum, globalaxis, m}}
63 forces = { 1, 'Static', {'NodalForces', {2, 1, 0}};
64           2, 'DispCtrl', {2, 1, DispCell} };
65 %-----

```

```

1  --- *****
2  --- MatType = 1: ModGMP material without isotropic hardening
3  --- MatType = 2: ModGMP material with isotropic hardening
4  MatType = 2
5  --- *****
6  nodes = { {1, 0, 0},
7           {2, 1, 0} }
8  --- *****
9  Simple2DTruss = 'truss2d'
10 elements = { {1, Simple2DTruss, 1, 2, 1, 1} }
11 --- *****
12 if (MatType == 1) then
13     materials = { {tag, 'modifiedGMPsteel', E, rho, fy, b-ratio, R0, cR1, cR2, a1, a2, a3, a4, sigma-init} }
14     materials = { {1, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 0.0, 55, 0, 55, 0} }
15 end
16 if (MatType == 2) then
17     materials = { {1, 'modifiedGMPsteel', 200, 0, 0.25, 0.2, 15, 0.925, 0.15, 10, 60, 10, 60, 0} }
18 end
19 --- *****
20 model = StructureModel(2,2)
21 model.addNodes(nodes)
22 model.addMaterials(materials)
23 model.addElements(elements)
24 model.constrainNode(1,1,1)
25 model.constrainNode(2,1,1)
26 --- *****
27 function generateincrementalload()
28     format: tag node dof
29     local loadform = {'incrementalnodaldisplacement', 2, 1}
30     local f = assert(io.open('cyclicwave.txt','r'))
31     local n = f:read("number")
32     while (n ~= nil) do
33         table.insert(loadform, 0.005*n)
34         n = f:read("number")
35     end
36     f:close()
37     return loadform
38 end
39 --- *****
40 staticloading = LoadDescription()
41 --- format: 'staticnodalload' <node> <dof> <amplitude>
42 l = generateincrementalload()
43 staticloading:addLoad(l)
44 --- *****
45 --- Static analysis
46 print("Static analysis started\n")
47 displ = {}
48 function displperstep(increment)
49     dx2,dy2 = model.nodeDisplacements(2)
50     table.insert(displ, dx2)
51 end
52 ---
53 react = {}
54 function reactperstep(increment)
55     fx1,fy1 = model.nodeRestoringForces(1)
56     table.insert(react, fx1)
57 end
58 ---
59 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=10000})
60 analysis = StaticAnalysis(solver)
61 analysis:setStructureModel(model)
62 analysis:addcallback(displperstep,"increment")
63 analysis:addcallback(reactperstep,"increment")
64 analysis:solve(staticloading)
65 --- *****
66 --- Set output file
67 function writedata1(x, fname)

```

```

68 local f = assert(io.open(fname, 'w'))
69 local writenl = 0
70 for i,v in ipairs(x) do
71   f:write(v, " ")
72   writenl = writenl + 1
73   -- length of row size: writenl
74   if (writenl > 0) then
75     writenl = 0
76     f:write("\n")
77   end
78 end
79 f:close()
80 end
81 --
82 if (MatType == 1) then
83   writedata1(displ, 'Ex18StaticNonIsotropicNodalDisp_2.dat')
84   writedata1(react, 'Ex18StaticNonIsotropicReact_1.dat')
85 end
86 if (MatType == 2) then
87   writedata1(displ, 'Ex19StaticIsotropicNodalDisp_2.dat')
88   writedata1(react, 'Ex19StaticIsotropicReact_1.dat')
89 end
90 print("Static analysis ended\n")

```

3.4 Steel Beam-columns, Fiber Section, Static, Transient (HHT and Shing)

Beam column elements with layered section, and hardening material are analyzed next, Fig. ???. The Cyclic displacement shown in Fig. 3.2, magnified by a factor of 50 are applied on node 3 in the X direction. Ex20 has stiffness-based beam-column, Ex21 has flexibility-based beam-column with element iteration, and Ex22 has flexibility-based beam-column without element iteration. For transient analysis, HHT integration scheme in Ex20 and Shing method in Ex21 and 22 are used with $\alpha = -0.1$, $\beta = 0.3025$ and $\gamma = 0.6$.

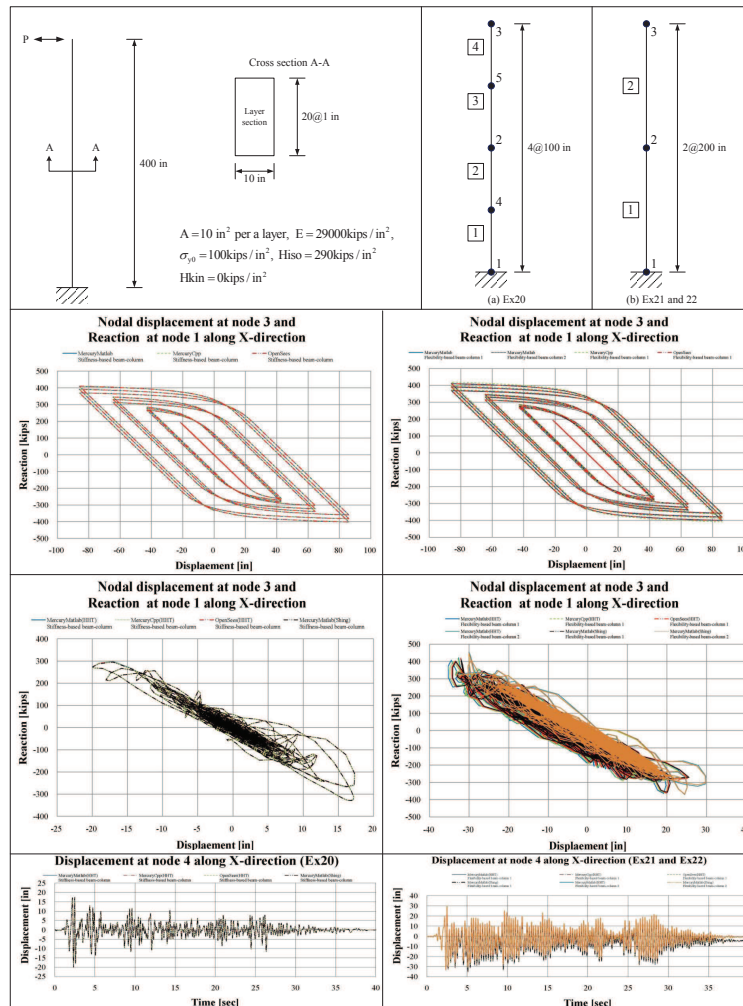


Figure 3.5: Examples 20- 22


```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex20to22.m
7 %-----
8 % Description
9 % 1. Static and transient analysis
10 % 2. Load control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam-column,
13 % flexibility-based 2D beam-column 1 and 2
14 % 5. Layer sections
15 % 6. Hardening material
16 %-----
17 % Section AnalysisType
18 % AnalysisType = 1: Displacement Control
19 % AnalysisType = 2: Transient analysis
20 AnalysisType = 2;
21 %-----
22 % Section DynType
23 % DynType = 1: Newmark beta
24 % DynType = 2: HHT
25 % DynType = 3: Shing
26 % DynType = 4: None
27 DynType = 3;
28 %-----
29 % Section EleType
30 % EleType = 1: Stiffness-based 2D beam-column
31 % EleType = 2: Flexibility-based 2D beam-column by Spacone
32 % EleType = 3: Flexibility-based 2D beam-column by Carol
33 EleType = 2;
34 %-----
35 % Preface
36 Unit = {'kip', 'in'};
37 StrMode = {2, 3};
38 %-----
39 % Control block
40 if ( (EleType == 1 || EleType == 2) && (DynType == 1 || DynType == 2) )
41     Iteration = {'static',{ 'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
42                 {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
43                 {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
44                 };
45     'transient',{ {'NewtonRaphson', 10, 1.0e-3, 'DisplNorm'};
46                 {'ModifiedNewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
47                 {'InitialStiffness', 1000, 1.0e-3, 'DisplNorm'};
48                 };
49     'element',{ {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
50                {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
51                {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'};
52                };
53 else
54     'static',{ {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};};
55     'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'};};
56 end
57 %-----
58 % Control block
59 if (DynType == 3)
60     Iteration = {'static',{ 'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
61                 {'ModifiedNewtonRaphson', 1000, 1.0e-6, 'ForceNorm'};
62                 {'InitialStiffness', 10000, 1.0e-6, 'ForceNorm'};
63                 };
64     'transient',{ {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'}; % for shing method
65                 };
66     'element',{ {'NewtonRaphson', 100, 1.0e-3, 'DisplNorm'};
67                 {'ModifiedNewtonRaphson', 1000, 1.0e-3, 'DisplNorm'};
68                 {'InitialStiffness', 100000, 1.0e-3, 'DisplNorm'};
69                 };
70 end
71 %-----
72 % Control block
73 %-----
74 if (DynType == 1)
75     Integration = {'Newmark', 0, 1/4, 1/2, 0, 0};
76     eigens = {0.02, 0.02};
77 end
78 if (DynType == 2)
79     Integration = {'HHT', 0, -0.1, 0.3025, 0.6, 0, 0};
80     eigens = {0.02, 0.02};
81 end
82 if (DynType == 3)
83     Integration = {'Shing', 0, -0.1, 0.3025, 0.6, 0, 0, 10};
84     eigens = {0.02, 0.02};
85 end
86 %-----
87 % Geometry block
88 if (EleType == 1)
89     nodcoord = {1, 0, 0;
90                4, 0, 100;
91                2, 0, 200;
92                5, 0, 300;
93                3, 0, 400};
94 else
95     nodcoord = {1, 0, 0;
96                2, 0, 200;
97                3, 0, 400};
98 end
99 %-----
100 % Element block
101 % elements = { { eletag, 'eletype', in, jn, nlp, sectag } };
102 if (EleType == 1)
103     elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
104                 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
105                 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
106                 };
107 end

```

```

108         {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1}};
109 elseif (EleType == 2)
110     elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
111                 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1}};
112 elseif (EleType == 3)
113     elements = { {1, 'FlexibilityBased2DBeamColumn2', 1, 2, 5, 1};
114                 {2, 'FlexibilityBased2DBeamColumn2', 2, 3, 5, 1}};
115 end
116 %-----
117 % Section block
118 % b = 10, h = 20, number of layer = 10, hlayer = 2
119 area = 10; mtag = 1; count = 0;
120 for ydis = -9.5:1.0:9.5
121     count = count + 1; lay(count,1:3) = [mtag, area, ydis];
122 end
123 nlay = size(lay,1);
124 for i = 1:nlay
125     laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);
126 end
127 % sections = { sectag, 'Layer', {mattag, A, y} }
128 sections = {1, 'Layer', laycell};
129 clear area; clear mtag; clear count; clear nlay; clear lay;
130 clear laycell; clear i; clear ydis;
131 %-----
132 % Material block
133 % mass density = 15.2 (slug/ft^3)
134 %               = 15.2 (lb*s^2/ft/ft^3)
135 %               = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
136 materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007}};
137 %-----
138 % Force block
139 if (AnalysisType == 1)
140     DispInput = load('cyclicwave.txt');
141     row = size(DispInput,1);
142     for i = 1:row
143         DispCell{i} = 50*DispInput(i);
144     end
145     forces = { 1, 'Static', {'NodalForces', {3, 1, 0}};
146              2, 'DispCtrl', {3, 1, DispCell} };
147     clear DispInput; clear row; clear i; clear DispCell;
148 elseif (AnalysisType == 2)
149     ga = load('ElCentro_g_0_01_Matlab.txt');
150     nga = size(ga, 1);
151     for i = 1:nga
152         groundacceleration{i,1} = ga(i,1);
153         groundacceleration{i,2} = ga(i,2);
154         groundacceleration{i,3} = ga(i,3);
155     end
156     forces = { 1, 'Static', {'NodalForces', {3, 1, 0}};
157              2, 'Acceleration', {50*386.4, groundacceleration} };
158     clear ga; clear nga; clear i; clear groundacceleration;
159 end
160 %-----

```

```

1  --- *****
2  --- AnalysisType = 1: Static analysis
3  --- AnalysisType= 2: Trnsient analysis
4  AnalysisType = 2
5  --- EleType = 1: Stiffness-based beam-column
6  --- EleType = 2: Flexibility-based beam-column
7  EleType = 1
8  --- *****
9  if (EleType == 1) then
10     nodes = {{1, 0, 0}};
11             {4, 0, 100, 'mass', 0.0146604, 0.0146604, 0};
12             {2, 0, 200, 'mass', 0.0146604, 0.0146604, 0};
13             {5, 0, 300, 'mass', 0.0146604, 0.0146604, 0};
14             {3, 0, 400, 'mass', 0.0073302, 0.0073302, 0}};
15     elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5}};
16                 { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5}};
17                 { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5}};
18                 { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5}} };
19 end
20 if (EleType == 2) then
21     nodes = {{1, 0, 0}};
22             {2, 0, 200, 'mass', 0.0293208, 0.0293208, 0};
23             {3, 0, 400, 'mass', 0.0146604, 0.0146604, 0}}
24     flexparams = {100000, 1e-3}
25     elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1, 5}, flexparams };
26                 { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1, 5}, flexparams } }
27 end
28 ---
29
30 sections = {
31 1, 'Fiber',
32 --- MatTag, Area, y-loc, z-loc
33 { 1, 10, -9.5, 0;
34   1, 10, -8.5, 0;
35   1, 10, -7.5, 0;
36   1, 10, -6.5, 0;
37   1, 10, -5.5, 0;
38   1, 10, -4.5, 0;
39   1, 10, -3.5, 0;
40   1, 10, -2.5, 0;
41   1, 10, -1.5, 0;
42   1, 10, -0.5, 0;
43   1, 10, 0.5, 0;
44   1, 10, 1.5, 0;
45   1, 10, 2.5, 0;
46   1, 10, 3.5, 0;
47   1, 10, 4.5, 0;
48   1, 10, 5.5, 0;
49   1, 10, 6.5, 0;
50   1, 10, 7.5, 0;
51   1, 10, 8.5, 0;

```

```

52 | 1, 10, 9.5, 0; } };
53 |
54 | materials = { {1, 'hardening', 29000, 0, 100, 290, 0} }
55 | -----
56 | model = StructureModel(2,3)
57 | model:addNodes(nodes)
58 | model:addMaterials(materials)
59 | model:addSections(sections)
60 | model:addElements(elements)
61 | model:constrainNode(1,1,1,1)
62 | if (AnalysisType == 1) then
63 |     model:constrainNode(3,1,0,0)
64 | end
65 | -----
66 | Static analysis
67 | if (AnalysisType == 1) then
68 |     print("Static analysis started\n")
69 |     -----
70 |     function generateincrementalload()
71 |         -- format:      tag          node dof
72 |         local loadform = {'incrementalnodaldisplacement', 3, 1}
73 |         local f = assert(io.open('cyclicwave.txt', 'r'))
74 |         local n = f:read("*number")
75 |         while (n ~= nil) do
76 |             table.insert(loadform, 50*n)
77 |             n = f:read("*number")
78 |         end
79 |         f:close()
80 |         return loadform
81 |     end
82 |     -----
83 |     staticloading = LoadDescription()
84 |     -- format: 'staticnodalload' <node> <dof> <amplitude>
85 |     l = generateincrementalload()
86 |     staticloading:addLoad(l)
87 |     -----
88 |     displ = {}
89 |     function displperstep(increment)
90 |         dx3, dy3, dz3 = model:nodeDisplacements(3)
91 |         table.insert(displ, dx3)
92 |     end
93 |     -----
94 |     react = {}
95 |     function reactperstep(increment)
96 |         fx1, fy1, fz1 = model:nodeRestoringForces(1)
97 |         table.insert(react, fx1)
98 |     end
99 |     -----
100 |     -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
101 |     solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=100000})
102 |     analysis = StaticAnalysis(solver)
103 |     analysis:setStructureModel(model)
104 |     analysis:addcallback(displperstep, "increment")
105 |     analysis:addcallback(reactperstep, "increment")
106 |     analysis:solve(staticloading)
107 |     -----
108 |     -- Set output file
109 |     function writedata1(x, fname)
110 |         local f = assert(io.open(fname, 'w'))
111 |         local writenl = 0
112 |         for i, v in ipairs(x) do
113 |             f:write(v, " ")
114 |             writenl = writenl + 1
115 |             -- length of row size: writenl
116 |             if (writenl > 0) then
117 |                 writenl = 0
118 |                 f:write("\n")
119 |             end
120 |         end
121 |         f:close()
122 |     end
123 |     -----
124 |     if (EleType == 1) then
125 |         writedata1(displ, 'Ex20StaticNodalDisp_3.dat')
126 |         writedata1(react, 'Ex20StaticReact_1.dat')
127 |     end
128 |     if (EleType == 2) then
129 |         writedata1(displ, 'Ex21StaticNodalDisp_3.dat')
130 |         writedata1(react, 'Ex21StaticReact_1.dat')
131 |     end
132 |     print("Static analysis ended\n")
133 | end
134 | -----
135 | if (AnalysisType == 2) then
136 |     print("Transient analysis started\n")
137 |     earthquakeloading = LoadDescription()
138 |     accelamp = 50*386.4
139 |     earthquakeloading:addLoad({'groundmotion', 'ElCentro_g_0_01_OpenSees.txt', dt=0.01, 1, accelamp})
140 |     -----
141 |     displ = {}
142 |     function displvertime(time)
143 |         dx3, dy3, dz3 = model:nodeDisplacements(3)
144 |         table.insert(displ, dx3)
145 |     end
146 |     -----
147 |     react = {}
148 |     function reactvertime(time)
149 |         fx1, fy1, fz1 = model:nodeRestoringForces(1)
150 |         table.insert(react, fx1)
151 |     end
152 |     -----
153 |     -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-3, iterations=100})
154 |     solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=1000})
155 |     transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.1, 0.3025, 0.6)
156 |     transientanalysis:addcallback(displvertime, "timestep")
157 |     transientanalysis:addcallback(reactvertime, "timestep")
158 |     if (EleType == 1) then

```

```

159 model:setRayleighCoefficients(9.693343,0.000038)
160 end
161 if (EleType == 2) then
162 model:setRayleighCoefficients(0.757867,0.000287)
163 end
164 transientanalysis :solve(4000)
165 -----
166 function writedata1(x, fname)
167 local f = assert(io.open(fname,'w'))
168 local writenl = 0
169 for i,v in ipairs(x) do
170 f:write(v, " ")
171 writenl = writenl + 1
172 -- length of row size: writenl
173 if (writenl > 0) then
174 writenl = 0
175 f:write("\n")
176 end
177 end
178 f:close()
179 end
180 -----
181 if (EleType == 1) then
182 writedata1(displ,'Ex20HHTNodalDisp_3.dat')
183 writedata1(react,'Ex20HHTReact_1.dat')
184 end
185 if (EleType == 2) then
186 writedata1(displ,'Ex21HHTNodalDisp_3.dat')
187 writedata1(react,'Ex21HHTReact_1.dat')
188 end
189 print("Transient analysis ended\n")
190 end
191 -----

```

3.5 Zero-length and Beam Column, Nonlinear Steel Element, load control

The implementation of the zero-length element is examined next in combination with stiffness-based beam-column and zero-length element, elastic and bilinear materials, Fig. 3.6. The incremental forces are increase by -10kN up to -50kN at node 2 in the X direction.

```

1 -----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex23.m
7 -----
8 % Description
9 % 1. Static analysis
10 % 2. Static force and load control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam column and Zero-Length 2D element
13 % 5. General section
14 % 6. Bilinear material
15 -----
16 % Preface
17 Unit = {'kip', 'in'};
18 StrMode = {2, 3};
19 -----
20 % Control block
21 Iteration = {'static',{ 'InitialStiffness', 100, 1.0e-8, 'ForceNorm'};
22 }
23 };
24 -----
25 % Geometry block
26 nodcoord = {1, 0, 0;
27 2, 0, 0;
28 3, 0, 120};
29 constraint = {1, 1, 1, 1;
30 3, 1, 1, 1};
31 -----
32 % Element block
33 elements = { {1, 'ZeroLength2D', 1, 2, 0, 1, 0, deg2rad(90)}
34 {2, 'StiffnessBased2DBeamColumn', 2, 3, 1, 1}};
35 -----
36 % Section block
37 sections = { 1, 'General', {2, 20, 0, 0, 1400} };
38 -----
39 % Material block
40 materials = { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
41 {2, 'Elastic', 30000, 0, 0} };
42 -----
43 % Force block
44 forces = { 1, 'Static', {'NodalForces', {2, 1, -10}};
45 2, 'LoadCtrl', {2, 1, {-20,-30,-40,-50} } };
46 -----

```

```

1 -----
2 nodes = { {1, 0, 0};
3 {2, 0, 0};
4 {3, 0, 120} };
5 -----
6 { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}}, {0,1,0},{-1,0,0} }
7 elements = { {1, 'InterfaceElement2D', 1, 2, { {1, {1,0,0} } }, {} };
8 {2, 'StiffnessBased2DBeamColumn', 2, 3, {1, 1} } };
9 -----
10 sections = { 1, 'general', {2, 20, 1400} };
11 -----
12 materials = { {1,'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
13 {2,'elastic',30000, 0,0} };
14 -----

```

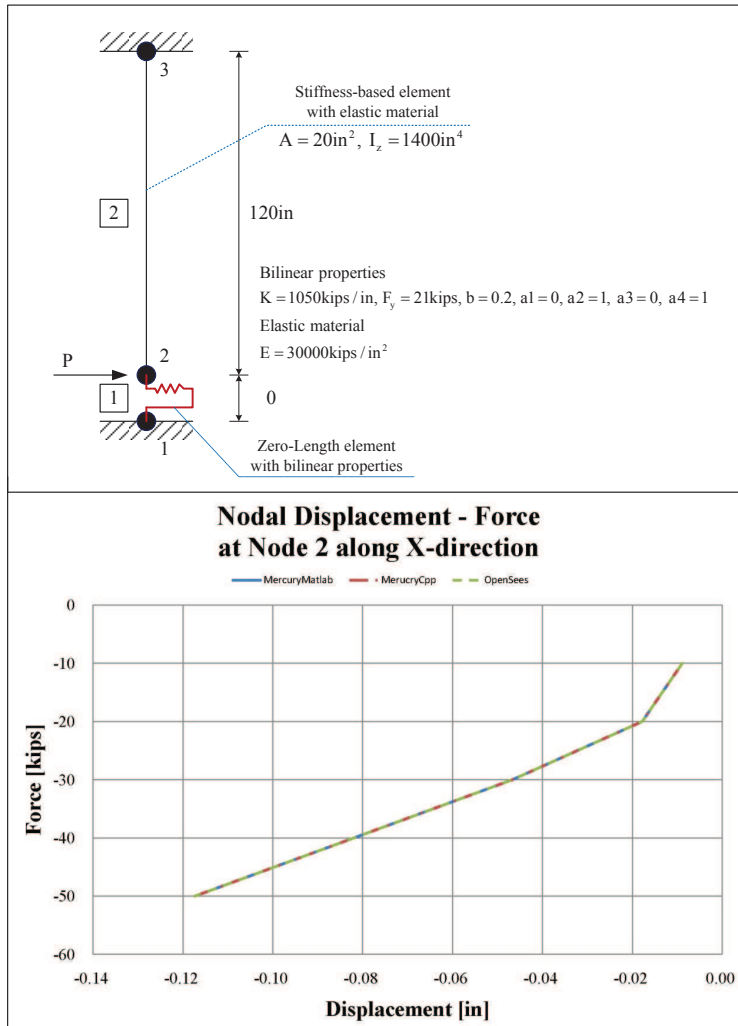


Figure 3.6: Examples 23

```

15 model = StructureModel(2,3)
16 model:addNodes(nodes)
17 model:addMaterials(materials)
18 model:addSections(sections)
19 model:addElements(elements)
20 --- *****
21 model:constrainNode(1,1,1,1)
22 model:constrainNode(3,1,1,1)
23 --- *****
24 staticloading = LoadDescription()
25 staticloading:addLoad({'incrementalnodalload', 2, 1, -10,-20,-30,-40,-50})
26 --- *****
27 displ = {}
28 function displperstep(increment)
29     dx2,dy2,dz2 = model:nodeDisplacements(2)
30     table.insert(displ, dx2)
31 end
32 solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
33 analysis = StaticAnalysis(solver)
34 analysis:setStructureModel(model)
35 analysis:addcallback(displperstep,"increment")
36 analysis:solve(staticloading)
37 --- *****
38 --- Set output file
39 function writedata1(x, fname)
40     local f = assert(io.open(fname,'w'))
41     local writenl = 0
42     for i,v in ipairs(x) do
43         f:write(v, " ")
44         writenl = writenl + 1
45         --- length of row size: writenl
46         if (writenl > 0) then
47             writenl = 0
48             f:write("\n")
49         end
50     end
51     f:close()
52 end
53 writedata1(displ,'Ex23NodalDisp-2.dat')

```

3.6 Zero-length Section, and Beam Column, Fiber, Nonlinear Steel Element, load control

The zero length section element is validated next in a similar way as in the preceding example, Fig. 3.7. Incremental forces of -20kN up to -400kN are applied on node 2 in the X direction.

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex24.m
7 %-----
8 % Description
9 % 1. Static analysis
10 % 2. Static force and load control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam column and Zero-Length 2D section element
13 % 5. General section
14 % 6. Bilinear material
15 %-----
16 % Preface
17 Unit = {'kip', 'in'};
18 StrMode = {2, 3};
19 %-----
20 % Control block
21 Iteration = {'static', {'InitialStiffness', 100, 1.0e-8, 'ForceNorm'}};
22     }
23 };
24 %-----
25 % Geometry block
26 nodcoord = {1, 0, 0;
27             2, 0, 0;
28             3, 0, 120};
29 constraint = {1, 1, 1, 1;
30              3, 1, 1, 1};
31 %-----
32 % Element block
33 elements = { {1, 'ZeroLength2DSection', 1, 2, deg2rad(90), 1}
34             {2, 'StiffnessBased2DBeamColumn', 2, 3, 3, 2}};
35 %-----
36 % Section block
37 sections = { 2, 'Layer', {2, 12, 5.5;
38                          2, 12, 4.5;
39                          2, 12, 3.5;
40                          2, 12, 2.5;
41                          2, 12, 1.5;
42                          2, 12, 0.5;
43                          2, 12, -0.5;
44                          2, 12, -1.5;
45                          2, 12, -2.5;
46                          2, 12, -3.5;
47                          2, 12, -4.5;
48                          2, 12, -5.5};
49             1, 'Layer', {1, 12, 5.5;
50                          1, 12, 4.5;
51                          1, 12, 3.5;
52                          1, 12, 2.5;
53                          1, 12, 1.5;
54                          1, 12, 0.5;
55                          1, 12, -0.5;
56                          1, 12, -1.5;
57                          1, 12, -2.5;
58                          1, 12, -3.5;

```

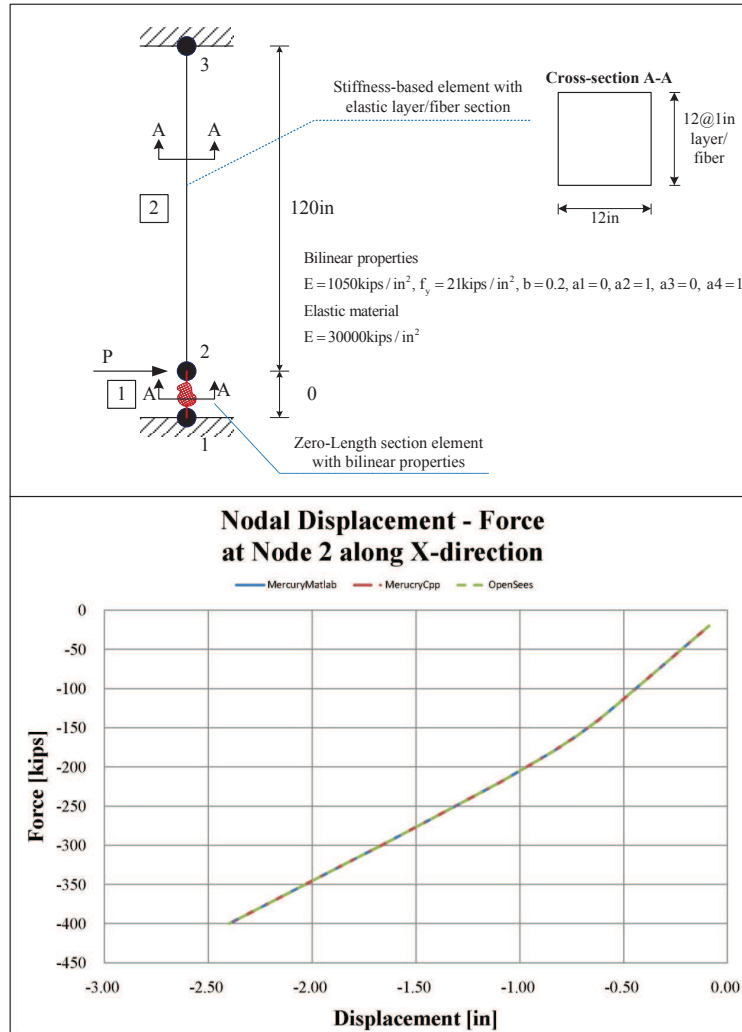


Figure 3.7: Examples 24

```

59         1, 12, -4.5;
60         1, 12, -5.5} };
61 -----
62 % Material block
63 materials = { {1, 'Bilinear', 1050, 21, 0.2, 0, 0, 1, 0, 1};
64              {2, 'Elastic', 30000, 0, 0} };
65 -----
66 % Force block
67 forces = { 1, 'Static', {'NodalForces', {2, 1, -20}};
68           2, 'LoadCtrl', {2, 1, {-40, -60, -80, -100, -120, ...,
69                               -140, -160, -180, -200, -220, ...,
70                               -240, -260, -280, -300, -320, ...,
71                               -340, -360, -380, -400} } };
72 -----

1  -----
2  nodes = { {1, 0, 0};
3            {2, 0, 0};
4            {3, 0, 120} };
5  -----
6  { eleTag, 'InterfaceElement2D', inode, jnode, { {matTag, {1,0,0} } }, { {secTag}}, {0,1,0},{-1,0,0} }
7  elements = { {1, 'InterfaceElement2D', 1, 2, {}, {{1}}, {0,1,0},{-1,0,0} };
8              {2, 'StiffnessBased2DBeamColumn', 2, 3, {2, 3} } };
9  -----
10 sections = {1, 'Fiber', {1, 12, 5.5, 0;
11                1, 12, 4.5, 0;
12                1, 12, 3.5, 0;
13                1, 12, 2.5, 0;
14                1, 12, 1.5, 0;
15                1, 12, 0.5, 0;
16                1, 12, -0.5, 0;
17                1, 12, -1.5, 0;
18                1, 12, -2.5, 0;
19                1, 12, -3.5, 0;
20                1, 12, -4.5, 0;
21                1, 12, -5.5, 0};
22          2, 'Fiber', {2, 12, 5.5, 0;
23                    2, 12, 4.5, 0;
24                    2, 12, 3.5, 0;
25                    2, 12, 2.5, 0;
26                    2, 12, 1.5, 0;
27                    2, 12, 0.5, 0;
28                    2, 12, -0.5, 0;
29                    2, 12, -1.5, 0;
30                    2, 12, -2.5, 0;
31                    2, 12, -3.5, 0;
32                    2, 12, -4.5, 0;
33                    2, 12, -5.5, 0};
34          };
35 -----
36 materials = { {1,'bilinear', 1050, 0, 21, 0.2, 0, 1, 0, 1};
37              {2,'elastic',30000, 0.0} };
38 -----
39 model = StructureModel(2,3)
40 model.addNodes(nodes)
41 model.addMaterials(materials)
42 model.addSections(sections)
43 model.addElements(elements)
44 -----
45 model.constrainNode(1,1,1)
46 model.constrainNode(3,1,1)
47 -----
48 staticloading = LoadDescription()
49 staticloading.addLoad({'incrementalnodalload', 2, 1, -20,-40,-60,-80,-100,-120,-140,-160,-180,-200,-220,-240,-260,-280,-300,-320,-340})
50 -----
51 displ = {}
52 function displperstep(increment)
53     dx2,dy2,dz2 = model.nodeDisplacements(2)
54     table.insert(displ, dx2)
55 end
56 solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
57 analysis = StaticAnalysis(solver)
58 analysis.setStructureModel(model)
59 analysis.addcallback(displperstep,"increment")
60 analysis.solve(staticloading)
61 -----
62 % Set output file
63 function writedata1(x, fname)
64     local f = assert(io.open(fname, 'w'))
65     local writenl = 0
66     for i,v in ipairs(x) do
67         f:write(v, " ")
68         writenl = writenl + 1
69         -- length of row size: writenl
70         if (writenl > 0) then
71             writenl = 0
72             f:write("\n")
73         end
74     end
75     f:close()
76 end
77 writedata1(displ, 'Ex24NodalDisp_2.dat')

```

3.7 Beam-column, Fiber Section, Nonlinear Material, multi-d.o.f.s displacement control, Pushover Analysis

Validation of displacement control at multiple free degrees of freedom, as described in 4.5 is performed next. Layered sections beam-column elements with hardening material are used, Fig. ???. Cyclic displacements shown in Fig. 3.2 are applied at node 2 and 3 magnified by a factor of -30 and 50 respectively. Ex25 has stiffness-based beam-column and Ex26 has flexibility-based beam-column.

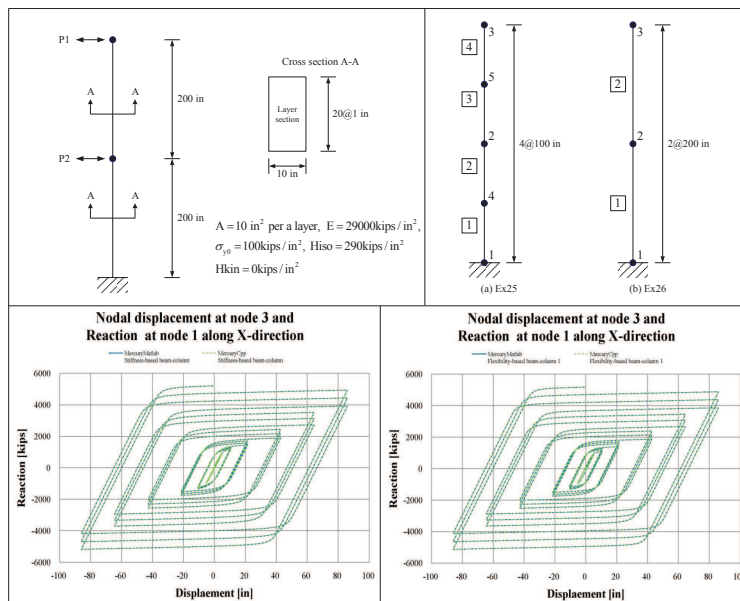


Figure 3.8: Examples 25- 26

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name : Ex25to26.m
7 %-----
8 % Description
9 % 1. Static analysis
10 % 2. Multiple displacement control
11 % 3. Iterative method
12 % 4. Stiffness-based 2D beam-column,
13 % flexibility-based 2D beam-column 1 and 2
14 % 5. Layer sections
15 % 6. Hardening material
16 %-----
17 % Section EleType
18 % EleType = 1: Stiffness-based 2D beam-column
19 % EleType = 2: Flexibility-based 2D beam-column by Spacone
20 EleType = 2;
21 %-----
22 % Preface
23 Unit = {'kip', 'in'};
24 StrMode = {2, 3};
25 %-----
26 % Control block
27 Iteration = {'static', { {'NewtonRaphson', 100, 1.0e-8, 'ForceNorm'};
28 { 'ModifiedNewtonRaphson', 1000, 1.0e-8, 'ForceNorm'};
29 { 'InitialStiffness', 10000, 1.0e-8, 'ForceNorm'};
30 };
31 'element', { {'NewtonRaphson', 100, 1.0e-5, 'DisplNorm'};
32 { 'ModifiedNewtonRaphson', 1000, 1.0e-5, 'DisplNorm'};
33 { 'InitialStiffness', 100000, 1.0e-5, 'DisplNorm'};
34 };
35 };
36 %-----
37 % Geometry block
38 if (EleType == 1)
39 nodcoord = {1, 0, 0;
40 4, 0, 100;
41 2, 0, 200;
42 5, 0, 300;
43 3, 0, 400};
44 elseif (EleType == 2)
45 nodcoord = {1, 0, 0;
46 2, 0, 200;
47 3, 0, 400};
48 end
49 % nodtag, x, y, z
50 constraint = {1, 1, 1, 1};
51 %-----
52 % Element block
53 % elements = { {eletag, 'eletype', in, jn, nlp, sectag } };
54 if (EleType == 1)
55 elements = { {1, 'StiffnessBased2DBeamColumn', 1, 4, 5, 1};
56 {2, 'StiffnessBased2DBeamColumn', 4, 2, 5, 1};
57 {3, 'StiffnessBased2DBeamColumn', 2, 5, 5, 1};
58 {4, 'StiffnessBased2DBeamColumn', 5, 3, 5, 1}};
59 elseif (EleType == 2)
60 elements = { {1, 'FlexibilityBased2DBeamColumn1', 1, 2, 5, 1};
61 {2, 'FlexibilityBased2DBeamColumn1', 2, 3, 5, 1}};
62 end

```

```

63 %-----
64 % Section block
65 % b = 10, h = 20, number of layer = 10, hlayer = 2
66 area = 10; mtag = 1; count = 0;
67 for ydis = -9.5:1.0:9.5
68     count = count + 1; lay(count,1:3) = [mtag, area, ydis];
69 end
70 nlay = size(lay,1);
71 for i = 1:nlay
72     laycell{i,1}=lay(i,1); laycell{i,2}=lay(i,2); laycell{i,3}=lay(i,3);
73 end
74 % sections = { sectag, 'Layer', 'Layer', {mattag, A, y} }
75 sections = {1, 'Layer', laycell};
76 clear area; clear mtag; clear count; clear nlay; clear lay;
77 clear laycell; clear i; clear ydis;
78 %-----
79 % Material block
80 % mass density = 15.2 (slug/ft^3)
81 %               = 15.2 (lb*s^2/ft^3)
82 %               = 15.2*(10^-3)/(12^4) (kips*s^2/in^4)
83 materials = { {1, 'Hardening', 29*10^3, 100, 290, 0, 7.3302e-007}};
84 %-----
85 % Force block
86 DispInput = load('cyclicwave.txt');
87 row = size(DispInput,1);
88 for i = 1:row
89     DispCell1{i} = -30*DispInput(i);
90     DispCell2{i} = 50*DispInput(i);
91 end
92 forces = { 1, 'Static', {'NodalForces', {3, 1, 0} };
93           2, 'DispCtrl', {2, 1, DispCell1;
94                       3, 1, DispCell2} };
95 clear DispInput; clear row; clear i; clear DispCell1; clear DispCell2;
96 %-----

```

```

1  --- *****
2  --- EleType = 1: Stiffness-based beam-column
3  --- EleType = 2: Flexibility-based beam-column
4  EleType = 2
5  --- *****
6  if (EleType == 1) then
7      nodes = {{1, 0, 0},
8              {4, 0, 100},
9              {2, 0, 200},
10             {5, 0, 300},
11             {3, 0, 400}};
12     elements = { { 1, 'StiffnessBased2DBeamColumn', 1, 4, {1, 5} };
13                 { 2, 'StiffnessBased2DBeamColumn', 4, 2, {1, 5} };
14                 { 3, 'StiffnessBased2DBeamColumn', 2, 5, {1, 5} };
15                 { 4, 'StiffnessBased2DBeamColumn', 5, 3, {1, 5} } };
16 end
17
18 if (EleType == 2) then
19     nodes = {{1, 0, 0},
20             {2, 0, 200},
21             {3, 0, 400}};
22     flexparams = {1000, 1e-5};
23     elements = { { 1, 'FlexibilityBased2DBeamColumn', 1, 2, {1, 5}, flexparams };
24                 { 2, 'FlexibilityBased2DBeamColumn', 2, 3, {1, 5}, flexparams } };
25 end
26
27 sections = {
28 1, 'Fiber',
29 --- MatTag, Area, y-loc, z-loc
30 { 1, 10, -9.5, 0;
31   1, 10, -8.5, 0;
32   1, 10, -7.5, 0;
33   1, 10, -6.5, 0;
34   1, 10, -5.5, 0;
35   1, 10, -4.5, 0;
36   1, 10, -3.5, 0;
37   1, 10, -2.5, 0;
38   1, 10, -1.5, 0;
39   1, 10, -0.5, 0;
40   1, 10, 0.5, 0;
41   1, 10, 1.5, 0;
42   1, 10, 2.5, 0;
43   1, 10, 3.5, 0;
44   1, 10, 4.5, 0;
45   1, 10, 5.5, 0;
46   1, 10, 6.5, 0;
47   1, 10, 7.5, 0;
48   1, 10, 8.5, 0;
49   1, 10, 9.5, 0; } };
50
51 materials = { {1, 'hardening', 29000, 0, 100, 290, 0} }
52 --- *****
53 model = StructureModel(2,3)
54 model.addNodes(nodes)
55 model.addMaterials(materials)
56 model.addSections(sections)
57 model.addElements(elements)
58 model.constrainNode(1,1,1,1)
59 model.constrainNode(2,1,0,0)
60 model.constrainNode(3,1,0,0)
61 --- *****
62 --- Static analysis
63 function generateincrementalload2()
64 --- format: tag node dof
65 local loadform = {'incrementalnodaldisplacement', 2, 1}
66 local f = assert(io.open('cyclicwave.txt','r'))
67 local n = f:read('*number')
68 while (n ~= nil) do
69     table.insert(loadform, -30*n)
70     n = f:read('*number')

```

```

71 end
72 f:close()
73 return loadform
74 end
75 function generateincrementalload3()
76 -- format: tag node dof
77 local loadform = {'incrementalnodaldisplacement', 3, 1}
78 local f = assert(io.open('cyclicwave.txt','r'))
79 local n = f:read("*number")
80 while (n ~= nil) do
81 table.insert(loadform, 50*n)
82 n = f:read("*number")
83 end
84 f:close()
85 return loadform
86 end
87 -- *****
88 staticloading = LoadDescription()
89 -- format: 'staticnodalload' <node> <dof> <amplitude>
90 l2 = generateincrementalload2()
91 l3 = generateincrementalload3()
92 staticloading:addLoad(l2)
93 staticloading:addLoad(l3)
94 -- *****
95 displ = {}
96 function displperstep(increment)
97 dx2,dy2,dz2 = model:nodeDisplacements(2)
98 dx3,dy3,dz3 = model:nodeDisplacements(3)
99 table.insert(displ, dx2)
100 table.insert(displ, dx3)
101 end
102 --
103 react = {}
104 function reactperstep(increment)
105 fx1,fy1,fz1 = model:nodeRestoringForces(1)
106 print("Work\n");
107 table.insert(react, fx1)
108 end
109 -- solver = NonlinearSolver("newtonraphson", { displacementdeltatolerance=1e-5, iterations=100})
110 -- solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-5, iterations=10000})
111 -- multisolver <type> <disp delta> <residual> <max iterations>
112 solver = NonlinearSolver("multisolver",
113 "newtonraphson", 1e-5, 1e-5, 100,
114 "initialstiffness", 1e-5, 1e-5, 1000)
115
116
117
118 analysis = StaticAnalysis(solver)
119 analysis:setStructureModel(model)
120 analysis:addcallback(displperstep,"increment")
121 analysis:addcallback(reactperstep,"increment")
122 analysis:solve(staticloading)
123 -- *****
124 -- Set output file
125 function writedata1(x, fname)
126 local f = assert(io.open(fname,'w'))
127 local writenl = 0
128 for i,v in ipairs(x) do
129 f:write(v, " ")
130 writenl = writenl + 1
131 -- length of row size: writenl
132 if (writenl > 0) then
133 writenl = 0
134 f:write("\n")
135 end
136 end
137 f:close()
138 end
139 function writedata2(x, fname)
140 local f = assert(io.open(fname,'w'))
141 local writenl = 0
142 for i,v in ipairs(x) do
143 f:write(v, " ")
144 writenl = writenl + 1
145 -- length of row size: writenl
146 if (writenl > 1) then
147 writenl = 0
148 f:write("\n")
149 end
150 end
151 f:close()
152 end
153 --
154 if (EleType == 1) then
155 writedata2(displ, 'Ex25NodalDisp_2_3.dat')
156 writedata1(react, 'Ex25React_1.dat')
157 end
158 if (EleType == 2) then
159 writedata2(displ, 'Ex26NodalDisp_2_3.dat')
160 writedata1(react, 'Ex26React_1.dat')
161 end

```

3.8 Reinforced Concrete Beam-Column, Fiber Section, Transient Analysis

This example consists of four types of beam-column element with layer section and material constitutive models for reinforced concrete, Fig. 3.9. In Ex27, material constitutive model of confined (core) or unconfined (cover) concrete using the modified Kent-Park model, whereas in Ex28, material constitutive model of confined (core) or unconfined (cover) concrete based on the anisotropic damage model with permanent strain are used. Steel is modeled with the modified Giuffre-Monegotto-Pinto model, and the shear spring in the zero-length element has a bilinear model.

At the base, a zero-length section and zero-length element are used to capture bond-slip. Fig. 3.10 succinctly describes the bar-slip zero-length fiber-section element (?) used. ? uses an empirically derived stress-deformation relation for the bar-slip fiber material between concrete and reinforcement. The material properties of the concrete are the same as those in the adjacent column element

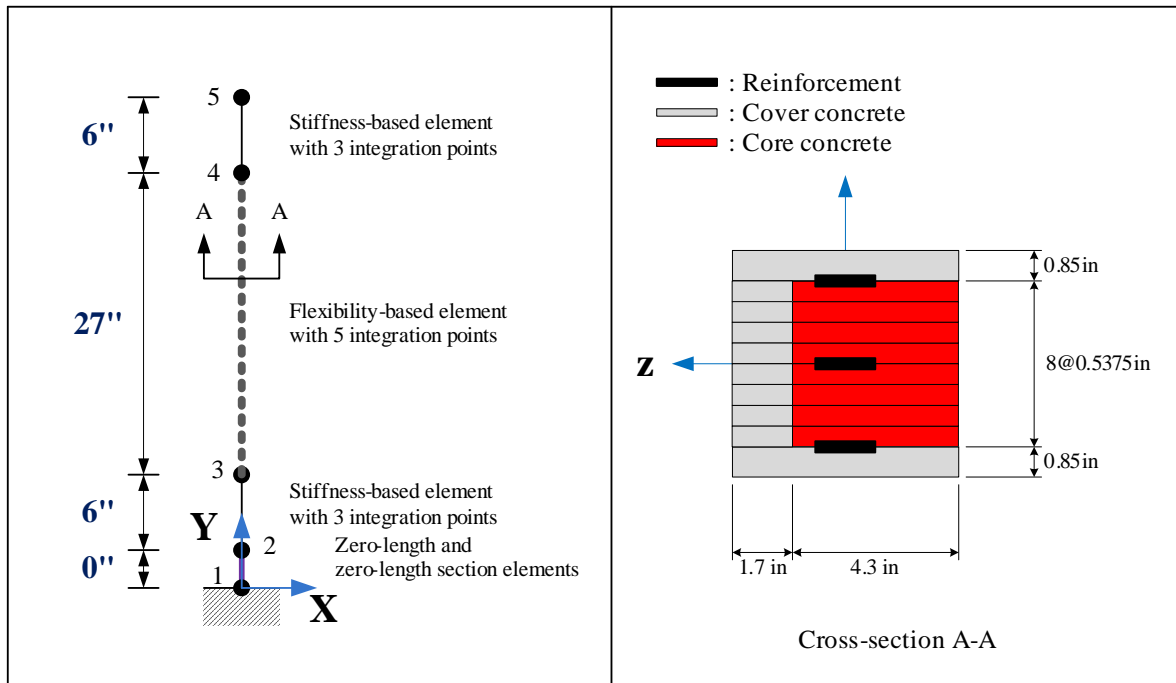


Figure 3.9: Beam-column elements for Ex27 and Ex28

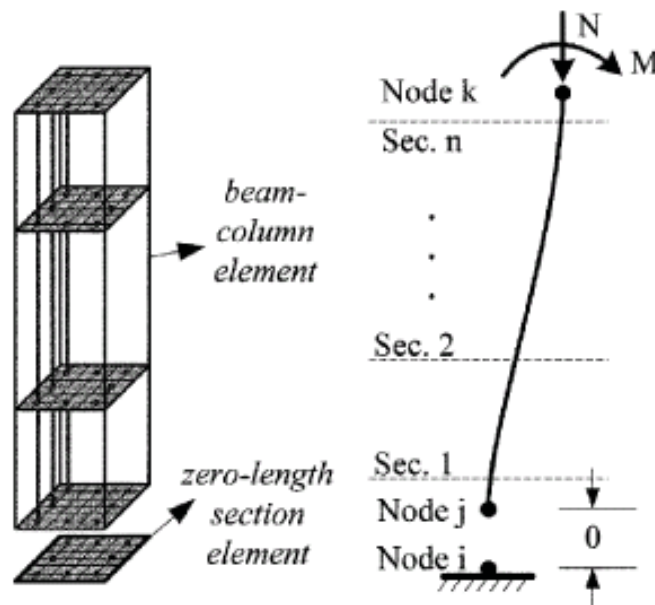


Figure 3.10: Bar slip zero-length fiber section element (?)

except the residual stress at large strains is taken as $0.8 \cdot \sigma'_c$ (?). Zero-length elements are used to account for shear-deformations using elastic spring elements at both ends of beams and columns as well. The joints are assumed to be rigid otherwise.

Table 3.2 to 3.5 describe material properties of Ex27 and Ex28. Applied masses from node 2 to node 5 are $2.50712E-05$, 0.000137891 , 0.000137891 , $(2.50712E-05 + 9.0/386.4)kips/in^3$ respectively. There are no rotation masses.

Table 3.2: Material properties of concrete for Ex27 (unit: kips, in)

Element	Concrete	E_{ts}	σ_c	ϵ_c	σ_{cu}	ϵ_{cu}	λ	σ_t
Beam-column	Cover	549.231	-3.57	-0.0026	-1.19	-0.0078	0.3	0.448
	Core	549.451	-7.5	-0.00546	-7.35	-0.01638	0.3	0.650
Zero-length section	Cover	105.000	-3.57	-0.0136	-1.19	-0.0408	0.3	0.448
	Core	104.167	-7.5	-0.0288	-6.75	-0.0864	0.3	0.650

Table 3.3: Material properties of concrete for Ex28 (unit: kips, in)

Element	Concrete	E	ν	κ_0	A	k_b	k_d	K_b	K_d	D_c
Beam-column	Cover	2829	0.2	5.855E-08	1870	0.003248	0.05168	3.889E+10	22.27	1.00
	Core	1804	0.2	6.483E-06	503.6	3.125E-07	0.4054	3.159E+09	102	1.00
Zero-length section	Cover	569.3	0.2	1.689E-19	371.8	0.832	1.461	9.529E+13	100.4	1.00
	Core	396.7	0.2	0.0008326	124.8	2.084E-07	1.61	8.856E+11	62.48	1.00

Table 3.4: Material properties of reinforcement for Ex27 and Ex28 (unit: kips, in)

Element	E	σ_y	b	$R0$	$cR1$	$cR2$	$a1$	$a2$	$a3$	$a4$
Beam-column element	26500	87.5	0.01	15	0.925	0.15	0	55	0	55
Zero-length section element	6949	87.5	0.01	15	0.925	0.15	0	55	0	55

Fig. ?? describes results of Ex27 and Ex28.

```

1 %
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex27.m (1 column)
7 %
8 % Preface
9 Unit = {'kip', 'in'};
10 % ndim, ndofpn
11 StrMode = {2, 3};
12 %
13 % Control block
14 Iteration = {'static', {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
15 % {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};
16 % }
17 % 'element', {'NewtonRaphson', 1000, 1.0e-6, 'DisplNorm'};
18 % {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
19 % }
20 % 'transient', {'NewtonRaphson', 100, 1.0e-6, 'DisplNorm'};
21 % {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
22 % }
23 % };
24 % Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};
25 Integration = {'Shing', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252, 10};
26 addMass = {1, 0, 0, 0;
27 % 2, 2.50712E-05, 2.50712E-05, 0;
28 % 3, 0.000137891, 0.000137891, 0;
29 % 4, 0.000137891, 0.000137891, 0;
30 % 5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
31 %
32 nodcoord = {1, 0, 0;
33 % 2, 0, 0;
34 % 3, 0, 6;
35 % 4, 0, 33;
36 % 5, 0, 39};
37 constraint = {1, 1, 1, 1};
38 %
39 %
40 elements = {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
41 % {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
42 % {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
43 % {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
44 % {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1};
45 %
46 sections = {1, 'Layer', {1, 5.1, 2.575;
47 % 1, 5.1, -2.575;
48 % 1, 0.91375, 1.88125;
49 % 1, 0.91375, 1.34375;
50 % 1, 0.91375, 0.80625;
51 % 1, 0.91375, 0.26875;
52 % 1, 0.91375, -0.26875;
53 % 1, 0.91375, -0.80625;
54 % 1, 0.91375, -1.34375;
55 % 1, 0.91375, -1.88125;
56 % 2, 2.31125, 1.88125;
57 % 2, 2.31125, 1.34375;
58 % 2, 2.31125, 0.80625;
59 % 2, 2.31125, 0.26875;
60 % 2, 2.31125, -0.26875;

```

Table 3.5: Property of shear spring in zero-length element for Ex27 and Ex28 (unit: kips, in)

Element	E	σ_y	b	a_1	a_2	a_3	a_4
Shear spring in zero-length	1690	78.2	0.173	0	55	0	55

```

61      2, 2.31125, -0.80625;
62      2, 2.31125, -1.34375;
63      2, 2.31125, -1.88125;
64      17, 0.147, 2.15;
65      17, 0.098, 0;
66      17, 0.147, -2.15;};
67      3, 'Layer', {5, 5.1, 2.575;
68                5, 5.1, -2.575;
69                5, 0.91375, 1.88125;
70                5, 0.91375, 1.34375;
71                5, 0.91375, 0.80625;
72                5, 0.91375, 0.26875;
73                5, 0.91375, -0.26875;
74                5, 0.91375, -0.80625;
75                5, 0.91375, -1.34375;
76                5, 0.91375, -1.88125;
77                6, 2.31125, 1.88125;
78                6, 2.31125, 1.34375;
79                6, 2.31125, 0.80625;
80                6, 2.31125, 0.26875;
81                6, 2.31125, -0.26875;
82                6, 2.31125, -0.80625;
83                6, 2.31125, -1.34375;
84                6, 2.31125, -1.88125;
85      19, 0.147, 2.15;
86      19, 0.098, 0;
87      19, 0.147, -2.15;};};
88
89 %
90 materials = { { 1, 'ModKP', -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077, 549.2307692, 0
91              };
92              { 2, 'ModKP', -7.5, -0.00546, -7.35, -0.01638, 0.3, 0.649519053, 549.4505495
93              };
94              { 5, 'ModKP', -3.57, -0.0136, -1.19, -0.0408, 0.3, 0.448121077, 105, 0 };
95              { 6, 'ModKP', -7.5, -0.0288, -6.75, -0.0864, 0.3, 0.649519053, 104.1666667, 0
96              };
97              { 17, 'ModGMP', 26500, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55,
98              0, 55 };
99              { 19, 'ModGMP', 6949, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55,
100             0, 55 };
101             { 26, 'Bilinear', 1690, 78.2, 0.173, 0, 0, 55, 0, 55 };};};
102
103 % Force block
104 ga = load('NR_g-dt_0-01-Matlab.txt');
105 nga = size(ga, 1);
106 for i = 1:nga
107     groundacceleration{i,1} = ga(i,1);
108     groundacceleration{i,2} = ga(i,2);
109     groundacceleration{i,3} = ga(i,3);
110 end
111 forces = { 1, 'Static', {'NodalForces', {5, 1, 0}};
112           2, 'Acceleration', {386.4, groundacceleration}};

```

```

1  --- *****
2  --- 1 column
3  --- *****
4  --- o (39)
5  --- | Stiffness-based beam-column with 2 integration points
6  --- o (33)
7  --- |
8  --- | Flexibility-based beam-column with 5 integration points
9  --- |
10 --- o (6)
11 --- | Stiffness-based beam-column with 2 integration points
12 --- o (0)
13 --- | Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
14 --- -o- (0) (Fixed support)
15
16 --- *****
17 elements = {}
18
19 --- create ductile column node coordinates
20 --- create nodes
21 nodes = { {1, 0, 0, 'mass', 0, 0,0};
22           {2, 0, 0, 'mass', 2.50712E-05, 2.50712E-05,0};
23           {3, 0, 6, 'mass', 0.000137891, 0.000137891, 0};
24           {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
25           {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0} }
26
27 --- figure out section names
28 barslipsectionf = 'BSColDFSection'
29 barslipspringf = 'BSColDFSS'
30 columnsection = 'ColDSection'
31 columnrigidsection = 'ColRigidSection'
32 nIp_stif = 3;
33 nIp_flex = 5;
34 flexparams = {10000.1e-6}
35
36 --- Define elements
37 barslipbottom = { 1, 'InterfaceElement2D', 1, 2, { {barslipspringf, {1,0,0}} }, { {barslipsectionf}, {0,1,0},{-1,0,0} }
38 plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nIp_stif} }
39 flexcolumn = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection, nIp_flex}, flexparams }
40 plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn', 4, 5, {columnsection, nIp_stif} }
41 table.insert(elements, barslipbottom)
42 table.insert(elements, plasticcolumn1)
43 table.insert(elements, flexcolumn)
44 table.insert(elements, plasticcolumn2)
45
46 --- *****
47 --- Set section properties.

```

```

45 --
46 allsections = {
47 --
48 'ColDSection' , 'Fiber' ,
49 -- Tag, Area, y-loc, z-loc
50 {
51   'ColDCover' , 5.1 , 2.575 , 0 ,
52   'ColDCover' , 5.1 , -2.575 , 0 ,
53   'ColDCover' , 0.91375 , 1.88125 , 0 ,
54   'ColDCover' , 0.91375 , 1.34375 , 0 ,
55   'ColDCover' , 0.91375 , 0.80625 , 0 ,
56   'ColDCover' , 0.91375 , 0.26875 , 0 ,
57   'ColDCover' , 0.91375 , -0.26875 , 0 ,
58   'ColDCover' , 0.91375 , -0.80625 , 0 ,
59   'ColDCover' , 0.91375 , -1.34375 , 0 ,
60   'ColDCover' , 0.91375 , -1.88125 , 0 ,
61   'ColDCore' , 2.31125 , 1.88125 , 0 ,
62   'ColDCore' , 2.31125 , 1.34375 , 0 ,
63   'ColDCore' , 2.31125 , 0.80625 , 0 ,
64   'ColDCore' , 2.31125 , 0.26875 , 0 ,
65   'ColDCore' , 2.31125 , -0.26875 , 0 ,
66   'ColDCore' , 2.31125 , -0.80625 , 0 ,
67   'ColDCore' , 2.31125 , -1.34375 , 0 ,
68   'ColDCore' , 2.31125 , -1.88125 , 0 ,
69   'ColDSteel' , 0.147 , 2.15 , 0 ,
70   'ColDSteel' , 0.098 , 0 , 0 ,
71   'ColDSteel' , 0.147 , -2.15 , 0 , };
72 --
73 'BSColDFSection' , 'Fiber' ,
74 {
75   'BSColDFCover' , 5.1 , 2.575 , 0 ,
76   'BSColDFCover' , 5.1 , -2.575 , 0 ,
77   'BSColDFCover' , 0.91375 , 1.88125 , 0 ,
78   'BSColDFCover' , 0.91375 , 1.34375 , 0 ,
79   'BSColDFCover' , 0.91375 , 0.80625 , 0 ,
80   'BSColDFCover' , 0.91375 , 0.26875 , 0 ,
81   'BSColDFCover' , 0.91375 , -0.26875 , 0 ,
82   'BSColDFCover' , 0.91375 , -0.80625 , 0 ,
83   'BSColDFCover' , 0.91375 , -1.34375 , 0 ,
84   'BSColDFCover' , 0.91375 , -1.88125 , 0 ,
85   'BSColDFCore' , 2.31125 , 1.88125 , 0 ,
86   'BSColDFCore' , 2.31125 , 1.34375 , 0 ,
87   'BSColDFCore' , 2.31125 , 0.80625 , 0 ,
88   'BSColDFCore' , 2.31125 , 0.26875 , 0 ,
89   'BSColDFCore' , 2.31125 , -0.26875 , 0 ,
90   'BSColDFCore' , 2.31125 , -0.80625 , 0 ,
91   'BSColDFCore' , 2.31125 , -1.34375 , 0 ,
92   'BSColDFCore' , 2.31125 , -1.88125 , 0 ,
93   'BSColDFSteel' , 0.147 , 2.15 , 0 ,
94   'BSColDFSteel' , 0.098 , 0 , 0 ,
95   'BSColDFSteel' , 0.147 , -2.15 , 0 , }; }
96 -- *****
97 -- Set material properties
98 concretemat = 'ConcreteLinearTensionSoftening'
99 steelmat = 'GiuffreMenegottoPinto'
100 sspringmat = 'Bilinear'
101 materials = {
102   {'ColDCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077} ,
103   {'ColDCore' , concretemat , 549.4505495 , 0 , -7.5 , -0.00546 , -7.35 , -0.01638 , 0.3 , 0.649519053} ,
104   {'BSColDFCover' , concretemat , 105 , 0 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077} ,
105   {'BSColDFCore' , concretemat , 104.1666667 , 0 , -7.5 , -0.0288 , -6.75 , -0.0864 , 0.3 , 0.649519053} ,
106   {'ColDSteel' , steelmat , 26500 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0} ,
107   {'BSColDFSteel' , steelmat , 6949 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0} ,
108   {'BSColDFSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55} ,
109 }
110 -- *****
111 function dumptable(x)
112   local result = '{'
113   for i,v in ipairs(x) do
114     if (type(v) == 'table') then
115       result = result .. dumptable(v)
116     else
117       result = result .. v .. ','
118     end
119   end
120   return result .. '}\n'
121 end
122 --
123 -- print(dumptable(nodes))
124 -- print(dumptable(elements))
125 -- print(dumptable(materials))
126 -- print(dumptable(allsections))
127 -- *****
128 -- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
129 model = StructureModel(2,3)
130 -- Assign all input data to Mercury
131 model:addNodes(nodes)
132 model:addMaterials(materials)
133 model:addSections(allsections)
134 model:addElements(elements)
135 -- *****
136 -- constrain bottom nodes
137 model:constrainNode(1,1,1,1)
138 -- *****
139 print("Transient analysis started\n")
140 earthquakeloading = LoadDescription()
141 earthquakeloading:addLoad({'groundmotion' , 'NR_g-dt_0-01_OpneSees.txt' , dt=0.01' , 1 , 386.4})
142 -- *****
143 displ = {}
144 function displpertime(time)
145   dx5,dy5,dz5 = model:nodeDisplacements(5)
146   table.insert(displ , dx5)
147   print("Work\n");
148 end
149 --
150 react = {}
151 function reactpertime(time)
152   fx1,fy1 = model:nodeRestoringForces(1)
153   table.insert(react , fx1)

```

```

152 end
153 --- *****
154 solver = NonlinearSolver(" initialstiffness", { displacementdeltatolerance=1e-6, iterations=10000})
155 transientanalysis = DynamicAnalysis("HHT", model, solver, earthquakeloading, 0.01, -0.2, 0.36, 0.7)
156 transientanalysis: addcallback( displpertime, "timestep")
157 transientanalysis: addcallback( reactpertime, "timestep")
158 model: setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
159 transientanalysis: solve(7641)
160 --- *****
161 function writedata1(x, fname)
162     local f = assert(io.open(fname, 'w'))
163     local writenl = 0
164     for i,v in ipairs(x) do
165         f:write(v, " ")
166         writenl = writenl + 1
167         --- length of row size: writenl
168         if (writenl > 0) then
169             writenl = 0
170             f:write("\n")
171         end
172     end
173     f:close()
174 end
175 writedata1(displ, 'Ex27HHTNodalDisp_5.dat')
176 writedata1(react, 'Ex27HHTReact_1.dat')
177 print("Transient analysis ended\n")

```

```

1 %-----
2 % Mercury Matlab Version 1.0.1
3 % Written by Dae-Hung Kang, CU-NEES
4 % Copyright 2009, CU-NEES
5 % Written : October 2009.
6 % File name: Ex28.m (1 column)
7 %-----
8 % Preface
9 Unit = {'kip', 'in'};
10 % ndim, ndofpn
11 StrMode = {2, 3};
12 %-----
13 % Control block
14 Iteration = {'static', { {'NewtonRaphson', 100, 1.0e-6, 'ForceNorm'};
15                        {'InitialStiffness', 10000, 1.0e-6, 'DisplNorm'};
16                    },
17             'element', { {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};
18                        {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
19                    },
20             'transient', { {'NewtonRaphson', 10, 1.0e-6, 'DisplNorm'};
21                          {'InitialStiffness', 100000, 1.0e-6, 'DisplNorm'};
22                        }
23         };
24 Integration = {'HHT', 0, -0.2, 0.36, 0.7, 0.6318799279194399, 0.00015503501814608252};
25 addMass = {1, 0, 0, 0;
26            2, 2.50712E-05, 2.50712E-05, 0;
27            3, 0.000137891, 0.000137891, 0;
28            4, 0.000137891, 0.000137891, 0;
29            5, 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0};
30 %-----
31 nodcoord = {1, 0, 0;
32             2, 0, 0;
33             3, 0, 6;
34             4, 0, 33;
35             5, 0, 39};
36 constraint = {1, 1, 1, 1};
37 %-----
38 elements = { {1, 'ZeroLength2D', 1, 2, 0, 26, 0, pi()/2};
39             {2, 'ZeroLength2DSection', 1, 2, pi()/2, 3};
40             {3, 'StiffnessBased2DBeamColumn', 2, 3, 3, 1};
41             {4, 'FlexibilityBased2DBeamColumn1', 3, 4, 5, 1};
42             {5, 'StiffnessBased2DBeamColumn', 4, 5, 3, 1}};
43 %-----
44 sections = {1, 'Layer', {1, 5.1, 2.575;
45                        1, 5.1, -2.575;
46                        1, 0.91375, 1.88125;
47                        1, 0.91375, 1.34375;
48                        1, 0.91375, 0.80625;
49                        1, 0.91375, 0.26875;
50                        1, 0.91375, -0.26875;
51                        1, 0.91375, -0.80625;
52                        1, 0.91375, -1.34375;
53                        1, 0.91375, -1.88125;
54                        2, 2.31125, 1.88125;
55                        2, 2.31125, 1.34375;
56                        2, 2.31125, 0.80625;
57                        2, 2.31125, 0.26875;
58                        2, 2.31125, -0.26875;
59                        2, 2.31125, -0.80625;
60                        2, 2.31125, -1.34375;
61                        2, 2.31125, -1.88125;
62                        17, 0.147, 2.15;
63                        17, 0.098, 0;
64                        17, 0.147, -2.15};
65             3, 'Layer', {5, 5.1, 2.575;
66                        5, 5.1, -2.575;
67                        5, 0.91375, 1.88125;
68                        5, 0.91375, 1.34375;
69                        5, 0.91375, 0.80625;
70                        5, 0.91375, 0.26875;
71                        5, 0.91375, -0.26875;
72                        5, 0.91375, -0.80625;
73                        5, 0.91375, -1.34375;
74                        5, 0.91375, -1.88125;
75                        6, 2.31125, 1.88125;
76                        6, 2.31125, 1.34375;
77                        6, 2.31125, 0.80625;
78                        6, 2.31125, 0.26875;

```



```

79      6, 2.31125, 0.26875;
80      6, 2.31125, -0.26875;
81      6, 2.31125, -0.80625;
82      6, 2.31125, -1.34375;
83      6, 2.31125, -1.88125;
84      19, 0.147, 2.15;
85      19, 0.098, 0;
86      19, 0.147, -2.15;};};
87
88 %-----
89 materials = { { 1, 'AnisotropicDamage', 2829, 0.2, 5.855E-08, 1870, 0.003248, 0.05168,
38890000000, { 22.27, 0.9999999, 0 };},
89 { 2, 'AnisotropicDamage', 1804, 0.2, 0.000006483, 503.6, 3.125E-07, 0.4054,
3159000000, { 102, 0.9999999, 0 };},
90 { 5, 'AnisotropicDamage', 569.3, 0.2, 1.689E-19, 371.8, 0.832, 1.461,
9.529E+13, { 100.4, 0.9999999, 0 };},
91 { 6, 'AnisotropicDamage', 396.7, 0.2, 0.0008326, 124.8, 2.084E-07, 1.61,
8.856E+11, { 62.48, 0.9999999, 0 };},
92 { 17, 'ModGMP', 26500, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55,
0, 55 };},
93 { 19, 'ModGMP', 6949, 87.5, 0.01, 15, 0.925, 0.15, 0, 0, 55,
0, 55 };},
94 { 26, 'Bilinear', 1690, 78.2, 0.173, 0, 0, 55, 0, 55 };};};
95
96 % Force block
97 ga = load('NR_g-dt-0.01-Matlab.txt');
98 nga = size(ga, 1);
99 for i = 1:nga
100     groundacceleration{i,1} = ga(i,1);
101     groundacceleration{i,2} = ga(i,2);
102     groundacceleration{i,3} = ga(i,3);
103 end
104 forces = { 1, 'Static', {'NodalForces', {5, 1, 0}};
105           2, 'Acceleration', {386.4, groundacceleration}};

```

```

1  --- *****
2  --- 1 column
3  --- *****
4  --- o (39)
5  --- | Stiffness-based beam-column with 2 integration points
6  --- o (33)
7  --- |
8  --- | Flexibility-based beam-column with 5 integration points
9  --- |
10 --- o (6)
11 --- | Stiffness-based beam-column with 2 integration points
12 --- o (0)
13 --- | Zero-Length and zero-Length section elements(bottom bar slip and shear deformation)
14 --- .o. (0) (Fixed support)
15
16 --- *****
17 elements = {}
18
19 --- create ductile column node coordinates
20 --- create nodes
21 nodes = {{1, 0, 0, 'mass', 0, 0,0};
22          {2, 0, 0, 'mass', 2.50712E-05, 2.50712E-05,0};
23          {3, 0, 6, 'mass', 0.000137891, 0.000137891, 0};
24          {4, 0, 33, 'mass', 0.000137891, 0.000137891, 0};
25          {5, 0, 39, 'mass', 2.50712E-05+9.0/386.4, 2.50712E-05+9.0/386.4, 0}}
26
27 --- figure out section names
28 barslipsectionf = 'BSColDFSection'
29 barslipspringf = 'BSColDFSS'
30 columnsection = 'ColDSection'
31 columnrigidsection = 'ColRigidSection'
32 nlp_stif = 3;
33 nlp_flex = 5;
34 flexparams = {10000,1e-3}
35
36 --- Define elements
37 barslipbottom = { 1, 'InterfaceElement2D', 1, 2, {{barslipspringf, {1,0,0}}}, {{barslipsectionf}}, {0,1,0},{-1,0,0}}
38 plasticcolumn1 = { 2, 'StiffnessBased2DBeamColumn', 2, 3, {columnsection, nlp_stif}}
39 flexcolumn = { 3, 'FlexibilityBased2DBeamColumn', 3, 4, {columnsection, nlp_flex}, flexparams }
40 plasticcolumn2 = { 4, 'StiffnessBased2DBeamColumn', 4, 5, {columnsection, nlp_stif}}
41
42 table.insert(elements, barslipbottom)
43 table.insert(elements, plasticcolumn1)
44 table.insert(elements, flexcolumn)
45 table.insert(elements, plasticcolumn2)
46
47 --- *****
48 --- Set section properties.
49 ---
50 allsections = {
51 'ColDSection', 'Fiber',
52 --- Tag, Area, y-loc, z-loc
53 { 'ColDCover', 5.1, 2.575, 0,
54   'ColDCover', 5.1, -2.575, 0,
55   'ColDCover', 0.91375, 1.88125, 0,
56   'ColDCover', 0.91375, 1.34375, 0,
57   'ColDCover', 0.91375, 0.80625, 0,
58   'ColDCover', 0.91375, 0.26875, 0,
59   'ColDCover', 0.91375, -0.26875, 0,
60   'ColDCover', 0.91375, -0.80625, 0,
61   'ColDCover', 0.91375, -1.34375, 0,
62   'ColDCover', 0.91375, -1.88125, 0,
63   'ColDCore', 2.31125, 1.88125, 0,
64   'ColDCore', 2.31125, 1.34375, 0,
65   'ColDCore', 2.31125, 0.80625, 0,
66   'ColDCore', 2.31125, 0.26875, 0,
67   'ColDCore', 2.31125, -0.26875, 0,
68   'ColDCore', 2.31125, -0.80625, 0,
69   'ColDCore', 2.31125, -1.34375, 0,
70   'ColDCore', 2.31125, -1.88125, 0,
71   'ColDSteel', 0.147, 2.15, 0,
72   'ColDSteel', 0.098, 0, 0,
73   'ColDSteel', 0.147, -2.15, 0 };};

```

```

72 'BSColDFSection' , 'Fiber' ,
73 { 'BSColDFCover' , 5.1 , 2.575 , 0 ,
74   'BSColDFCover' , 5.1 , -2.575 , 0 ,
75   'BSColDFCover' , 0.91375 , 1.88125 , 0 ,
76   'BSColDFCover' , 0.91375 , 1.34375 , 0 ,
77   'BSColDFCover' , 0.91375 , 0.80625 , 0 ,
78   'BSColDFCover' , 0.91375 , 0.26875 , 0 ,
79   'BSColDFCover' , 0.91375 , -0.26875 , 0 ,
80   'BSColDFCover' , 0.91375 , -0.80625 , 0 ,
81   'BSColDFCover' , 0.91375 , -1.34375 , 0 ,
82   'BSColDFCover' , 0.91375 , -1.88125 , 0 ,
83   'BSColDFCore' , 2.31125 , 1.88125 , 0 ,
84   'BSColDFCore' , 2.31125 , 1.34375 , 0 ,
85   'BSColDFCore' , 2.31125 , 0.80625 , 0 ,
86   'BSColDFCore' , 2.31125 , 0.26875 , 0 ,
87   'BSColDFCore' , 2.31125 , -0.26875 , 0 ,
88   'BSColDFCore' , 2.31125 , -0.80625 , 0 ,
89   'BSColDFCore' , 2.31125 , -1.34375 , 0 ,
90   'BSColDFCore' , 2.31125 , -1.88125 , 0 ,
91   'BSColDFSteel' , 0.147 , 2.15 , 0 ,
92   'BSColDFSteel' , 0.098 , 0 , 0 ,
93   'BSColDFSteel' , 0.147 , -2.15 , 0 }; };
94 --- *****
95 --- Set material properties
96 concretemat = 'ConcreteLinearTensionSoftening'
97 steelmat = 'GiuffreMenegottoPinto'
98 sspringmat = 'Bilinear'
99 materials = {
100 {'ColDCover' , 'anisotropicdamage2' , 2829 , 0 , 0.2 , 5.855E-08 , 1870 , 0.003248 , 0.05168 ,
101   38890000000 , 22.27 , 0.9999999 };
102 {'ColDCore' , 'anisotropicdamage2' , 1804 , 0 , 0.2 , 0.000006483 , 503.6 , 3.125E-07 , 0.4054 , 3159000000
103   , 102 , 0.9999999 };
104 {'BSColDFCover' , 'anisotropicdamage2' , 569.3 , 0 , 0.2 , 1.689E-19 , 371.8 , 0.832 , 1.461 , 9.529E+13
105   , 100.4 , 0.9999999 };
106 {'BSColDFCore' , 'anisotropicdamage2' , 396.7 , 0 , 0.2 , 0.0008326 , 124.8 , 2.084E-07 , 1.61 ,
107   8.856E+11 , 62.48 , 0.9999999 };
108 {'ColDSteel' , steelmat , 26500 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
109 {'BSColDFSteel' , steelmat , 6949 , 0 , 87.5 , 0.01 , 15 , 0.925 , 0.15 , 0 , 55 , 0 , 55 , 0 } ,
110 {'BSColDFSS' , sspringmat , 1690 , 0 , 78.2 , 0.173 , 0 , 55 , 0 , 55 } ,
111 }
112 --- *****
113 function dumptable(x)
114   local result = '{'
115   for i,v in ipairs(x) do
116     if (type(v) == 'table') then
117       result = result .. dumptable(v)
118     else
119       result = result .. v .. ','
120     end
121   end
122   return result .. '}'\n'
123 end
124 ---
125 ---print(dumptable(nodes))
126 ---print(dumptable(elements))
127 ---print(dumptable(materials))
128 ---print(dumptable(allsections))
129 --- *****
130 --- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
131 model = StructureModel(2,3)
132 --- Assign all input data to Mercury
133 model:addNodes(nodes)
134 model:addMaterials(materials)
135 model:addSections(allsections)
136 model:addElements(elements)
137 --- *****
138 --- constrain bottom nodes
139 model:constrainNode(1,1,1,1)
140 --- *****
141 print("Transient analysis started\n")
142 earth quake loading = LoadDescription()
143 earth quake loading:addLoad({'groundmotion' , 'NR-g-dt-0-01-OpneSees.txt', dt=0.01' , 1 , 386.4})
144 --- *****
145 displ = {}
146 function displpertime(time)
147   dx5,dy5,dz5 = model:nodeDisplacements(5)
148   table.insert(displ, dx5)
149   print("Work\n");
150 end
151 ---
152 react = {}
153 function reactpertime(time)
154   fx1,fy1 = model:nodeRestoringForces(1)
155   table.insert(react, fx1)
156 end
157 --- *****
158 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-3, iterations=100000})
159 transientanalysis = DynamicAnalysis("HHT", model, solver, earth quake loading , 0.01, -0.2, 0.36, 0.7)
160 transientanalysis:addcallback(displpertime, "timestep")
161 transientanalysis:addcallback(reactpertime, "timestep")
162 model:setRayleighCoefficients(0.6318799279194399, 0.00015503501814608252)
163 transientanalysis:solve(7641)
164 --- *****
165 function writedata1(x, fname)
166   local f = assert(io.open(fname, 'w'))
167   local writenl = 0
168   for i,v in ipairs(x) do
169     f:write(v, " ")
170     writenl = writenl + 1
171     --- length of row size: writenl
172     if (writenl > 0) then
173       writenl = 0
174       f:write("\n")
175     end
176   end
177   f:close()
178 end
179 end

```

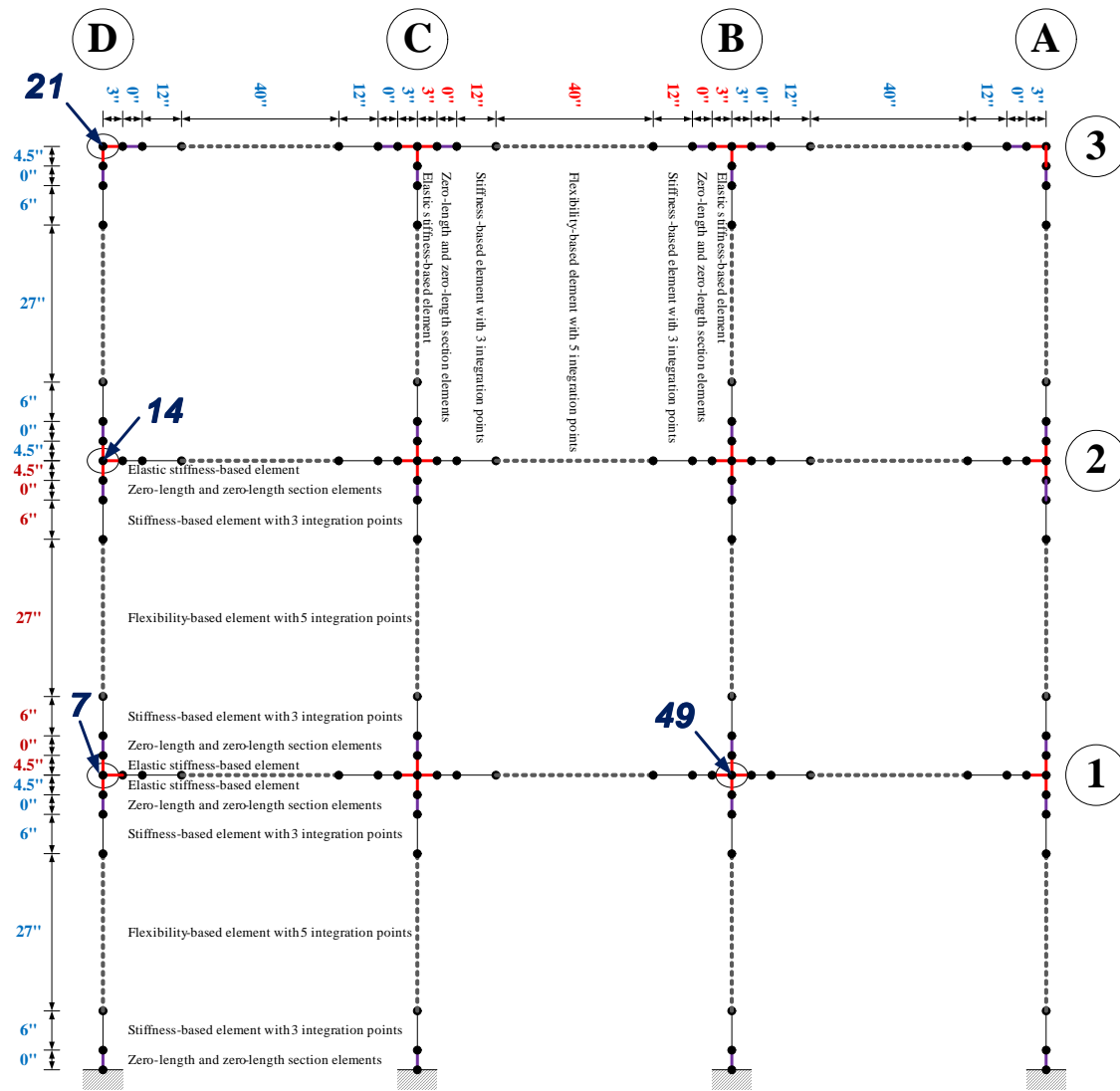


Figure 3.11: Numerical model for real-time hybrid simulation

```

175 writedata1 ( displ , 'Ex28HHTNodalDisp_5.dat ' )
176 writedata1 ( react , 'Ex28HHTReact_1.dat ' )
177 print ( " Transient analysis ended\n" )
    
```

3.9 Ex29: Full Reinforced Concrete Frame, HHT, Shing

This most complex validation example, Fig. 3.11, analyses the reinforced concrete frame tested on a shake table by ?, Fig. 3.9, and which will be later used to assess the real time hybrid simulation of Mercury, Fig. 3.9.

This example is sufficiently complex to warrant detailed description.

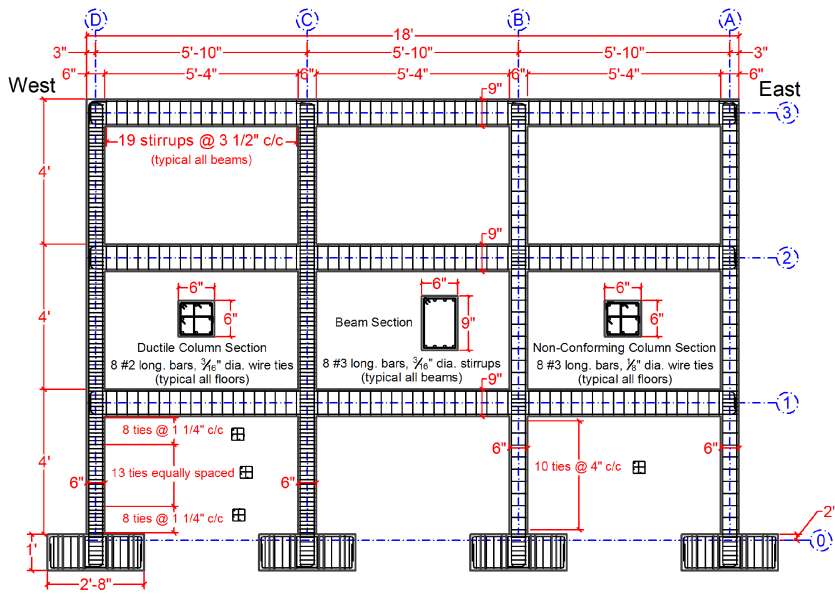
3.9.0.1 Frame discretization and properties

Beams and columns are modeled with one stiffness-based beam-columns (3 integration points) at each end, and one flexibility-based beam-column with 5 integrations.

Nodes 7, 14, 21 and 49 are monitored, in Fig. 3.11 to assess the seismic while ignoring self-weight. Columns A and B are shear critical (i.e under-reinforced), while C and D are ductile. Sectional characteristics are shown in Fig. 3.13 while Table 3.6 to 3.9 summarize those properties and constitutive model parameters used¹.

Nodal masses are used in the dynamic analysis, Table 3.10 to 3.12, where each beam has an added lead bundle masses at 4th and 5th nodes. For comparison HHT integration scheme is used in OpenSees, and HHT and Shing integration scheme are used in Mercury

¹ Col: Column, Beam: Beam, F: Footing section, D: Ductile section, ND: Shear critical section, BS: Bar-Slip section, Rigid: Rigid section, Cover: Concrete cover fiber, Core: Concrete core fiber, SS: Shear spring of zero-length element, steel: Reinforcement fiber, ModKP: modified Kent-Park model, ModGMP: modified Giuffre-Monegotto-Pinto



Reinforced concrete details, ?

Figure 3.12: Shake table test of reinforce concrete frame, ?

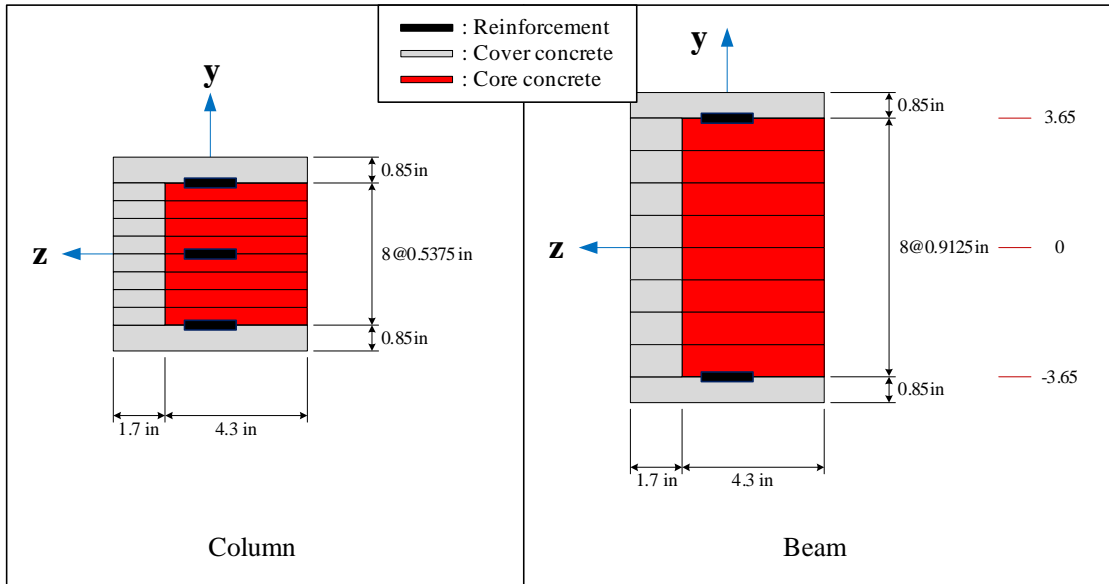


Figure 3.13: Section properties of Ex29

Table 3.6: Concrete material properties of concrete for Ex29 (unit:kips, in)

Fiber name	Material	$E_t s$	σ_c	ϵ_c	σ_{cu}	ϵ_{cu}	λ	σ_t
Col-D-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Col-D-Core	ModKP	549.45	-7.5	-0.00546	-7.35	-0.01638	0.3	0.6495
BS-Col-D-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-D-Core	ModKP	104.17	-7.5	-0.0288	-7.35	-0.0864	0.3	0.6495
BS-Col-D-F-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-D-F-Core	ModKP	104.17	-7.5	-0.0288	-6.75	-0.0864	0.3	0.6495
Col-ND-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Col-ND-Core	ModKP	549.30	-3.9	-0.00284	-3.51	-0.00852	0.3	0.4684
BS-Col-ND-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Col-ND-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684
BS-Col-ND-F-Cover	ModKP	144.24	-3.57	-0.0099	-1.19	-0.0297	0.3	0.4481
BS-Col-ND-F-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684
Beam-Cover	ModKP	549.23	-3.57	-0.0026	-1.19	-0.0078	0.3	0.4481
Beam-Core	ModKP	549.30	-3.9	-0.00284	-3.51	-0.00852	0.3	0.4684
BS-Beam-Cover	ModKP	105.00	-3.57	-0.0136	-1.19	-0.0408	0.3	0.4481
BS-Beam-Core	ModKP	106.85	-3.9	-0.0146	-3.51	-0.0438	0.3	0.4684

Table 3.7: Material properties of reinforcement for Ex29 (unit: kips, in)

Fiber name	Material	E	σ_y	b	$R0$	$cR1$	$cR2$	$a1$	$a2$	$a3$	$a4$
Col-D-Steel	ModGMP	26500	87.5	0.01	15	0.925	0.15	0	55	0	55
BS-Col-D-Steel	ModGMP	5067	87.5	0.01	15	0.925	0.15	0	55	0	55
BS-Col-D-F-Steel	ModGMP	6949	87.5	0.01	15	0.925	0.15	0	55	0	55
Col-ND-Steel	ModGMP	27300	80	0.01	15	0.925	0.15	0	55	0	55
BS-Col-ND-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55
BS-Col-ND-F-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55
Beam-Steel	ModGMP	27300	80	0.01	15	0.925	0.15	0	55	0	55
BS-Beam-Steel	ModGMP	5220	80	0.01	15	0.925	0.15	0	55	0	55

Table 3.8: Property of shear spring for Ex29 (unit: kips, in)

Fiber name	Material	E	σ_y	b	$a1$	$a2$	$a3$	$a4$
BS-Col-D-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-D-F-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-ND-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Col-ND-F-SS	Bilinear	1690	78.2	0.173	0	55	0	55
BS-Beam-SS	Bilinear	1545	75.8	0.396	0	55	0	55

Table 3.9: Properties of rigid elements (unit: kips, in)

Element	Model	E
Col-Rigid	Elastic	1E+10
Beam-Rigid	Elastic	1E+10

c++ version with $\alpha = -0.2$. This last model is particularly relevant for the real-time hybrid simulation which will be conducted later. Rayleigh damping coefficients were determined to be 1.177 for mass and 0.001599 for stiffness. The reinforced concrete frame is subjected to the seismic excitation shown in Fig. 3.14 (which was measured at the base of the shake table in Ghannoum's tests).

Figure 3.14: Seismic excitation for Ex29

Table 3.10: Mass properties in first floor column nodes (unit: $kips \cdot sec^2/in$)

First floor node	m_x	m_y	m_z
7th	0.0000752	0.0000752	0.00000752
6th	0.0000188	0.0000188	0.00000188
5th	0.0000251	0.0000251	0.00000251
4th	0.0001379	0.0001379	0.00001379
3rd	0.0001379	0.0001379	0.00001379
2nd	0.0000251	0.0000251	0.00000251
1st	0.0000000	0.0000000	0.00000000

Table 3.11: Mass properties in column nodes except first floor columns (unit: $kips \cdot sec^2/in$)

Other floor node	m_x	m_y	m_z
8th	0.0000627	0.0000627	0.00000627
7th	0.0000188	0.0000188	0.00000188
6th	0.0000251	0.0000251	0.00000251
5th	0.0001379	0.0001379	0.00001379
4th	0.0001379	0.0001379	0.00001379
3rd	0.0000251	0.0000251	0.00000251
2nd	0.0000188	0.0000188	0.00000188

```

1 *****
2 earthquakefiles = {
3   'Test16AccelHHHalfYield.txt',
4   'Test19AccelH1stCollapse.txt',
5   'Test30AccelH2ndCollapse.txt',
6   'Test32AccelH3rdCollapse.txt'
7 }
8 earthquakefile = earthquakefiles[2]
9 print('Excitation:', earthquakefile)
10 -----
11 alpha = -0.2
12 beta = (1-alpha)*(1-alpha)/4
13 gamma = (1-2*alpha)/2
14 -- alpha : alpha coefficient in HHT integration
15 -- beta : beta coefficient in HHT integration
16 -- gamma : gamma coefficient in HHT integration
17 *****
18 -- Multiple bay/floor building model
19 *****
20 -- Column : first floor
21 -- o (43.5)
22 -- | Elastic stiffness-based beam-column
23 -- o (39)
24 -- | Zero-Length and zero-Length section elements (top bar slip and shear deformation)
25 -- o (39)
26 -- | Stiffness-based beam-column with 2 integration points
27 -- o (33)
28 -- |
29 -- | Flexibility-based beam-column with 5 integration points
30 -- |
31 -- o (6)
32 -- | Stiffness-based beam-column with 2 integration points
33 -- o (0)
34 -- | Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
35 -- -o- (0) (Fixed support)
36
37 -----
38 -- Column : other floor
39 -- o (48)
40 -- | Elastic stiffness-based beam-column
41 -- o (43.5)
42 -- | Zero-Length and zero-Length section elements (top bar slip and shear deformation)
43 -- o (43.5)
44 -- | Stiffness-based beam-column with 2 integration points
45 -- o (37.5)
46 -- |
47 -- | Flexibility-based beam-column with 5 integration points
48 -- |
49 -- o (10.5)
50 -- | Stiffness-based beam-column with 2 integration points
51 -- o (4.5)
52 -- | Zero-Length and zero-Length section elements (bottom bar slip and shear deformation)
53 -- o (4.5)

```

Table 3.12: Mass properties in beam nodes (unit: $kips \cdot sec^2/in$)

Beam node	m_x	m_y	m_z
2nd	0.0000188	0.0000188	0.00000188
3rd	0.0000752	0.0000752	0.00000752
4th	0.0080899	0.0080899	0.00080899
5th	0.0080899	0.0080899	0.00080899
6th	0.0000752	0.0000752	0.00000752
7th	0.0000188	0.0000188	0.00000188

```

54 |-----| Elastic stiffness-based beam-column
55 |-----o (0)
56 |-----*****
57 |-----Beam
58 |-----o-----o-----o-----o-----o-----o
59 |----- (0) a (3) b (3) c (15) d (55)e(67) f (67) g (70)
60 |-----a: Elastic stiffness-based beam-column
61 |-----b: Zero-Length and zero-Length section elements(left bar slip and shear deformation)
62 |-----c: Stiffness-based beam-column with 2 integration points
63 |-----d: Flexibility-based beam-column with 5 integration points
64 |-----e: Stiffness-based beam-column with 2 integration points
65 |-----f: Zero-Length and zero-Length section elements(right bar slip and shear deformation)
66 |-----g: Elastic stiffness-based beam-column
67 |-----*****
68 |-----Basic variables for node information
69 |-----g : gravity acceleration
70 |-----nbays : Number of bays (nbays must be odd number)
71 |-----nflrs : Number of floors
72 |-----nlcnod: Number of nodes of the first column
73 |-----nocnod: Number of nodes of other columns
74 |-----nbmnod: Number of nodes of beams
75 |-----cilen : Total length of the first column
76 |-----colen : Total length of other columns
77 |-----bmilen : Total length of beams
78 |-----clnod : Coordinate information on the first column
79 |-----conod : Coordinate information on other columns
80 |-----bmnod : Coordinate information on beams
81 |-----cdlnodmass : Nodal mass of the first ductile column
82 |-----cdonodmass : Nodal mass of other ductile columns
83 |-----cndlnodmass: Nodal mass of the first shear critical column
84 |-----cndonodmass: Nodal mass of other shear critical column
85 |-----nlp_elas : Number of integration points of elastic stiffness-based beam-column
86 |-----nlp_stif : Number of integration points of nonlinear stiffness-based beam-column
87 |-----nlp_flex : Number of integration points of nonlinear flexibility-based beam-column
88 |-----flexparams : Iteration config for flexibility elements
89 |-----flexparams = {number of iterations, tolerance}
90 |-----$$$ start of node and element information $$$
91 |-----g = 386.4
92 |-----nbays = 3; --- nbays must be odd number.
93 |-----nflrs = 3;
94 |-----nlcnod= 7;
95 |-----nocnod= 8;
96 |-----nbmnod= 8;
97 |-----cilen = 43.5;
98 |-----colen = 48;
99 |-----bmilen = 70;
100 |-----clnod = {0,0,6,33,39,39,43.5};
101 |-----conod = {0,4.5,4.5,10.5,37.5,43.5,43.5,48};
102 |-----bmnod = {0,3,3,15,55,67,67,70};
103 |-----bmaddWeight = 3
104 |-----bmaddMass = bmaddWeight/g
105 |-----
106 |-----clnodmass = {0.0, 2.50712E-05, 0.000137891, 0.000137891, 2.50712E-05, 1.88034E-05, 7.52135E-05}
107 |-----conodmass = {0.0, 1.88034E-05, 2.50712E-05, 0.000137891, 0.000137891, 2.50712E-05, 1.88034E-05, 6.26779E-05}
108 |-----bmnodmass = {0.0, 1.88034E-05, 7.52135E-05, 0.000325925+bmaddMass, 0.000325925+bmaddMass, 7.52135E-05, 1.88034E-05, 0.0}
109 |-----
110 |-----nlp_elas = 1;
111 |-----nlp_stif = 3;
112 |-----nlp_flex = 5;
113 |-----
114 |-----flexparams = {1000,1e-6}
115 |-----$$$ END of node and element information $$$
116 |-----*****
117 |-----Set nodes and elements.
118 |-----
119 |-----
120 |-----nodes = {}
121 |-----elements = {}
122 |-----
123 |-----function nodename(bay, floor)
124 |-----return string.format('node_%d_%d', bay, floor)
125 |-----end
126 |-----
127 |-----function columnnodename(bay, floor, subsection)
128 |-----if (subsection == 1 and floor > 1) then
129 |-----return columnnodename(bay, floor-1,nocnod)
130 |-----if (floor == 2) then
131 |-----return columnnodename(bay, floor-1,nlcnod)
132 |-----else
133 |-----return columnnodename(bay, floor-1,nocnod)
134 |-----end
135 |-----else
136 |-----return string.format('colnode_%d_%d_%d', bay, floor, subsection)
137 |-----end
138 |-----end
139 |-----
140 |-----function beamnodename(bay, floor, subsection)
141 |-------- section 1 nodes re-use the column nodes
142 |-----if (subsection == 1) then
143 |-----return columnnodename(bay, floor, 1)
144 |-----else

```

```

145     return string.format('beamnode.%d.%d.%d', bay, floor, subsection)
146   end
147 end
148 -----
149
150 -- create ductile column node coordinates
151 ncold = (nbays+1)/2;
152 for colbay = 1,ncold do
153   for colfloor = 1,nflrs do
154     x = (colbay-1)*bmlen
155     nodeyfirst = clnod
156     nodeyhigher = conod
157     -- create nodes
158     if (colfloor==1) then
159       bottomnode = columnnodename(colbay, colfloor, 1)
160       for k=1,ncnod do
161         y = nodeyfirst[k]
162         masscd1 = clnodmass[k]
163         curnode = { columnnodename(colbay, colfloor, k), x, y, 'mass', masscd1, masscd1, 0.1*masscd1 }
164         table.insert(nodes, curnode)
165       end
166     else
167       bottomnode = columnnodename(colbay, colfloor-1, nocnod)
168       for k=2,nocnod do
169         y = cllen + colen * (colfloor-2) + nodeyhigher[k]
170         masscdo = conodmass[k]
171         curnode = { columnnodename(colbay, colfloor, k), x, y, 'mass', masscdo, masscdo, 0.1*masscdo }
172         table.insert(nodes, curnode)
173       end
174     end
175     -- figure out section names
176     barslipsectionf = 'BSColDFSection'
177     barslipspringf = 'BSColDFSS'
178     columnsection = 'ColDSection'
179     barslipspring = 'BSColDSS'
180     barslipsection = 'BSColDSection'
181     columnrigidsection = 'ColRigidSection'
182     if (colfloor==1) then
183       node1 = columnnodename(colbay, colfloor, 1)
184       node2 = columnnodename(colbay, colfloor, 2)
185       node3 = columnnodename(colbay, colfloor, 3)
186       node4 = columnnodename(colbay, colfloor, 4)
187       node5 = columnnodename(colbay, colfloor, 5)
188       node6 = columnnodename(colbay, colfloor, 6)
189       node7 = columnnodename(colbay, colfloor, 7)
190       -- Define elements
191       barslipbottom = { string.format('columnbsb.%d.%d', colbay, colfloor), 'InterfaceElement2D', node1, node2, { barslipspring,
192       plasticcolumn1 = { string.format('columnpl1.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node2, node3, { columnssec
193       flexcolumn = { string.format('columnflx.%d.%d', colbay, colfloor), 'FlexibilityBased2DBeamColumn', node3, node4, { columns
194       plasticcolumn2 = { string.format('columnpl2.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node4, node5, { columnsect
195       barsliptop = { string.format('columnbst.%d.%d', colbay, colfloor), 'InterfaceElement2D', node5, node6, { barslipspring,
196       rigidcolumn2top = { string.format('columnrdt.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node6, node7, { column
197       table.insert(elements, barslipbottom)
198       table.insert(elements, plasticcolumn1)
199       table.insert(elements, flexcolumn)
200       table.insert(elements, plasticcolumn2)
201       table.insert(elements, barsliptop)
202       table.insert(elements, rigidcolumn2top)
203     else
204       if (colfloor == 2) then
205         node1 = columnnodename(colbay, colfloor-1, ncnod)
206         node2 = columnnodename(colbay, colfloor-1, nocnod)
207       else
208         node1 = columnnodename(colbay, colfloor-1, nocnod)
209         node2 = columnnodename(colbay, colfloor, 2)
210         node3 = columnnodename(colbay, colfloor, 3)
211         node4 = columnnodename(colbay, colfloor, 4)
212         node5 = columnnodename(colbay, colfloor, 5)
213         node6 = columnnodename(colbay, colfloor, 6)
214         node7 = columnnodename(colbay, colfloor, 7)
215         node8 = columnnodename(colbay, colfloor, 8)
216         -- Define elements
217         rigidcolumnbottom = { string.format('columnrdb.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node1, node2, { colu
218         barslipbottom = { string.format('columnbsb.%d.%d', colbay, colfloor), 'InterfaceElement2D', node2, node3, { barslipspr
219         plasticcolumn1 = { string.format('columnpl1.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node3, node4, { column
220         flexcolumn = { string.format('columnflx.%d.%d', colbay, colfloor), 'FlexibilityBased2DBeamColumn', node4, node5, { colu
221         plasticcolumn2 = { string.format('columnpl2.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node5, node6, { column
222         barsliptop = { string.format('columnbst.%d.%d', colbay, colfloor), 'InterfaceElement2D', node6, node7, { barslipsprin
223         rigidcolumn2top = { string.format('columnrdt.%d.%d', colbay, colfloor), 'StiffnessBased2DBeamColumn', node7, node8, { colu
224         table.insert(elements, rigidcolumnbottom)
225         table.insert(elements, barslipbottom)
226         table.insert(elements, plasticcolumn1)
227         table.insert(elements, flexcolumn)
228         table.insert(elements, plasticcolumn2)
229         table.insert(elements, barsliptop)
230         table.insert(elements, rigidcolumn2top)
231       end
232     end
233 end
234
235 -- create shear critical column node coordinates
236 for colbay = ncold+1,nbays+1 do
237   for colfloor = 1,nflrs do
238     x = (colbay-1)*bmlen
239     nodeyfirst = clnod
240     nodeyhigher = conod
241     -- create nodes
242     if (colfloor==1) then
243       bottomnode = columnnodename(colbay, colfloor, 1)
244       for k=1,ncnod do
245         y = nodeyfirst[k]
246         masscnd1 = clnodmass[k]
247         curnode = { columnnodename(colbay, colfloor, k), x, y, 'mass', masscnd1, masscnd1, 0.1*masscnd1 }
248         table.insert(nodes, curnode)
249       end
250     else
251       bottomnode = columnnodename(colbay, colfloor-1, nocnod)

```



```

252     for k=2,nocnod do
253         y = cilen + colen * (colfloor-2) + nodeyhigher[k]
254         masscno = conodmass[k]
255         curnode = { columnnodename(colbay,colfloor,k), x,y, 'mass', masscno, masscno, 0.1*masscno }
256         table.insert(nodes,curnode)
257     end
258 end
259
260 --- figure out section names
261 barslipsectionf = 'BSCoINDSection'
262 barslipsection = 'BSCoINDSS'
263 columnsection = 'CoINDSection'
264 barslipsection = 'BSCoINDSection'
265 columnrigidsection = 'ColRigidSection'
266
267 if (colfloor==1) then
268     node1 = columnnodename(colbay,colfloor,1)
269     node2 = columnnodename(colbay,colfloor,2)
270     node3 = columnnodename(colbay,colfloor,3)
271     node4 = columnnodename(colbay,colfloor,4)
272     node5 = columnnodename(colbay,colfloor,5)
273     node6 = columnnodename(colbay,colfloor,6)
274     node7 = columnnodename(colbay,colfloor,7)
275
276     Define elements
277     barslipbottom = { string.format('columnbsb-%d-%d',colbay,colfloor), 'InterfaceElement2D', node1, node2, {barslipsection, {barslipsectionf}, {barslipsectionf}}
278     plasticcolumn1 = { string.format('columnpl1-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node2, node3, {columnsection, {columnsectionf}, {columnsectionf}}
279     flexcolumn = { string.format('columnflx-%d-%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn', node3, node4, {columnsection, {columnsectionf}, {columnsectionf}}
280     plasticcolumn2 = { string.format('columnpl2-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node4, node5, {columnsection, {columnsectionf}, {columnsectionf}}
281     barsliptop = { string.format('columnbst-%d-%d',colbay,colfloor), 'InterfaceElement2D', node5, node6, {barslipsection, {barslipsectionf}, {barslipsectionf}}
282     rigidcolumn = { string.format('columnrdt-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node6, node7, {columnsection, {columnsectionf}, {columnsectionf}}
283     table.insert(elements, barslipbottom)
284     table.insert(elements, plasticcolumn1)
285     table.insert(elements, flexcolumn)
286     table.insert(elements, plasticcolumn2)
287     table.insert(elements, barsliptop)
288     table.insert(elements, rigidcolumn)
289
290 else
291     if (colfloor == 2) then
292         node1 = columnnodename(colbay,colfloor-1,nlcnod)
293     else
294         node1 = columnnodename(colbay,colfloor-1,nocnod)
295     end
296     node2 = columnnodename(colbay,colfloor,2)
297     node3 = columnnodename(colbay,colfloor,3)
298     node4 = columnnodename(colbay,colfloor,4)
299     node5 = columnnodename(colbay,colfloor,5)
300     node6 = columnnodename(colbay,colfloor,6)
301     node7 = columnnodename(colbay,colfloor,7)
302     node8 = columnnodename(colbay,colfloor,8)
303
304     --- Define elements
305     rigidcolumnbottom = { string.format('columnrdb-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node1, node2, {columnsection, {columnsectionf}, {columnsectionf}}
306     barslipbottom = { string.format('columnbsb-%d-%d',colbay,colfloor), 'InterfaceElement2D', node2, node3, {barslipsection, {barslipsectionf}, {barslipsectionf}}
307     plasticcolumn1 = { string.format('columnpl1-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node3, node4, {columnsection, {columnsectionf}, {columnsectionf}}
308     flexcolumn = { string.format('columnflx-%d-%d',colbay,colfloor), 'FlexibilityBased2DBeamColumn', node4, node5, {columnsection, {columnsectionf}, {columnsectionf}}
309     plasticcolumn2 = { string.format('columnpl2-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node5, node6, {columnsection, {columnsectionf}, {columnsectionf}}
310     barsliptop = { string.format('columnbst-%d-%d',colbay,colfloor), 'InterfaceElement2D', node6, node7, {barslipsection, {barslipsectionf}, {barslipsectionf}}
311     rigidcolumn = { string.format('columnrdt-%d-%d',colbay,colfloor), 'StiffnessBased2DBeamColumn', node7, node8, {columnsection, {columnsectionf}, {columnsectionf}}
312     table.insert(elements, rigidcolumnbottom)
313     table.insert(elements, barslipbottom)
314     table.insert(elements, plasticcolumn1)
315     table.insert(elements, flexcolumn)
316     table.insert(elements, plasticcolumn2)
317     table.insert(elements, barsliptop)
318     table.insert(elements, rigidcolumn)
319
320 end
321 end
322
323 --- create beam node coordinates
324 for beambay = 1,nbays do
325     for beamfloor = 2,nflrs+1 do
326         y=cilen+colen*(beamfloor-2)
327         --- create nodes
328         for k=2,nbmnod-1 do
329             x = (beambay-1)*bmilen + bmnod[k]
330             massbm= bmnodmass[k]
331             curnode = { beamnodename(beambay,beamfloor,k), x ,y, 'mass', massbm, massbm, 0.1*massbm }
332             table.insert(nodes,curnode)
333         end
334
335         --- figure out section names
336         beamsection = 'BeamSection'
337         barslipsection = 'BSBeamSection'
338         beamrigidsection = 'BeamRigidSection'
339         barslipsection = 'BSBeamSS'
340
341         --- figure out node tag for node connectivity
342         node1 = beamnodename(beambay,beamfloor,1)
343         node2 = beamnodename(beambay,beamfloor,2)
344         node3 = beamnodename(beambay,beamfloor,3)
345         node4 = beamnodename(beambay,beamfloor,4)
346         node5 = beamnodename(beambay,beamfloor,5)
347         node6 = beamnodename(beambay,beamfloor,6)
348         node7 = beamnodename(beambay,beamfloor,7)
349         node8 = beamnodename(beambay+1, beamfloor,1)
350
351         --- Define elements
352         rigidbeamleft = { string.format('beamrdl-%d-%d',beambay,beamfloor), 'StiffnessBased2DBeamColumn', node1, node2, {beamrigidsection, {beamrigidsectionf}, {beamrigidsectionf}}
353         barslipleft = { string.format('beambsl-%d-%d',beambay,beamfloor), 'InterfaceElement2D', node2, node3, {barslipsection, {barslipsectionf}, {barslipsectionf}}
354         plasticbeam1 = { string.format('beamp1-%d-%d',beambay,beamfloor), 'StiffnessBased2DBeamColumn', node3, node4, {beamsection, {beamsectionf}, {beamsectionf}}
355         flexbeam = { string.format('beamflx-%d-%d',beambay,beamfloor), 'FlexibilityBased2DBeamColumn', node4, node5, {beamsection, {beamsectionf}, {beamsectionf}}
356         plasticbeam2 = { string.format('beamp2-%d-%d',beambay,beamfloor), 'StiffnessBased2DBeamColumn', node5, node6, {beamsection, {beamsectionf}, {beamsectionf}}
357         barslipright = { string.format('beambsr-%d-%d',beambay,beamfloor), 'InterfaceElement2D', node6, node7, {barslipsection, {barslipsectionf}, {barslipsectionf}}
358         rigidbeamright = { string.format('beamrdl-%d-%d',beambay,beamfloor), 'StiffnessBased2DBeamColumn', node7, node8, {beamrigidsection, {beamrigidsectionf}, {beamrigidsectionf}}
359         table.insert(elements, rigidbeamleft)
360         table.insert(elements, barslipleft)
361         table.insert(elements, plasticbeam1)
362         table.insert(elements, flexbeam)
363         table.insert(elements, plasticbeam2)
364         table.insert(elements, barslipright)
365         table.insert(elements, rigidbeamright)

```

```

359 end
360 end
361 --- *****
362 --- Set section properties.
363 ---
364 ---
365 allsections = {
366 ---
367 'ColDSection' , 'Fiber' ,
368 --- Tag, Area, y-loc, z-loc
369 {
370 'ColDCover' , 5.1 , 2.575 , 0 ,
371 'ColDCover' , 5.1 , -2.575 , 0 ,
372 'ColDCover' , 0.91375 , 1.88125 , 0 ,
373 'ColDCover' , 0.91375 , 1.34375 , 0 ,
374 'ColDCover' , 0.91375 , 0.80625 , 0 ,
375 'ColDCover' , 0.91375 , 0.26875 , 0 ,
376 'ColDCover' , 0.91375 , -0.26875 , 0 ,
377 'ColDCover' , 0.91375 , -0.80625 , 0 ,
378 'ColDCover' , 0.91375 , -1.34375 , 0 ,
379 'ColDCover' , 0.91375 , -1.88125 , 0 ,
380 'ColDCore' , 2.31125 , 1.88125 , 0 ,
381 'ColDCore' , 2.31125 , 1.34375 , 0 ,
382 'ColDCore' , 2.31125 , 0.80625 , 0 ,
383 'ColDCore' , 2.31125 , 0.26875 , 0 ,
384 'ColDCore' , 2.31125 , -0.26875 , 0 ,
385 'ColDCore' , 2.31125 , -0.80625 , 0 ,
386 'ColDCore' , 2.31125 , -1.34375 , 0 ,
387 'ColDCore' , 2.31125 , -1.88125 , 0 ,
388 'ColDSteel' , 0.147 , 2.15 , 0 ,
389 'ColDSteel' , 0.098 , 0 , 0 ,
390 'ColDSteel' , 0.147 , -2.15 , 0 } ;
391 ---
392 'BSColDSection' , 'Fiber' ,
393 {
394 'BSColDCover' , 5.1 , 2.575 , 0 ,
395 'BSColDCover' , 5.1 , -2.575 , 0 ,
396 'BSColDCover' , 0.91375 , 1.88125 , 0 ,
397 'BSColDCover' , 0.91375 , 1.34375 , 0 ,
398 'BSColDCover' , 0.91375 , 0.80625 , 0 ,
399 'BSColDCover' , 0.91375 , 0.26875 , 0 ,
400 'BSColDCover' , 0.91375 , -0.26875 , 0 ,
401 'BSColDCover' , 0.91375 , -0.80625 , 0 ,
402 'BSColDCover' , 0.91375 , -1.34375 , 0 ,
403 'BSColDCover' , 0.91375 , -1.88125 , 0 ,
404 'BSColDCore' , 2.31125 , 1.88125 , 0 ,
405 'BSColDCore' , 2.31125 , 1.34375 , 0 ,
406 'BSColDCore' , 2.31125 , 0.80625 , 0 ,
407 'BSColDCore' , 2.31125 , 0.26875 , 0 ,
408 'BSColDCore' , 2.31125 , -0.26875 , 0 ,
409 'BSColDCore' , 2.31125 , -0.80625 , 0 ,
410 'BSColDCore' , 2.31125 , -1.34375 , 0 ,
411 'BSColDCore' , 2.31125 , -1.88125 , 0 ,
412 'BSColDSteel' , 0.147 , 2.15 , 0 ,
413 'BSColDSteel' , 0.098 , 0 , 0 ,
414 'BSColDSteel' , 0.147 , -2.15 , 0 } ;
415 ---
416 'BSColDFSection' , 'Fiber' ,
417 {
418 'BSColDFCover' , 5.1 , 2.575 , 0 ,
419 'BSColDFCover' , 5.1 , -2.575 , 0 ,
420 'BSColDFCover' , 0.91375 , 1.88125 , 0 ,
421 'BSColDFCover' , 0.91375 , 1.34375 , 0 ,
422 'BSColDFCover' , 0.91375 , 0.80625 , 0 ,
423 'BSColDFCover' , 0.91375 , 0.26875 , 0 ,
424 'BSColDFCover' , 0.91375 , -0.26875 , 0 ,
425 'BSColDFCover' , 0.91375 , -0.80625 , 0 ,
426 'BSColDFCover' , 0.91375 , -1.34375 , 0 ,
427 'BSColDFCover' , 0.91375 , -1.88125 , 0 ,
428 'BSColDFCore' , 2.31125 , 1.88125 , 0 ,
429 'BSColDFCore' , 2.31125 , 1.34375 , 0 ,
430 'BSColDFCore' , 2.31125 , 0.80625 , 0 ,
431 'BSColDFCore' , 2.31125 , 0.26875 , 0 ,
432 'BSColDFCore' , 2.31125 , -0.26875 , 0 ,
433 'BSColDFCore' , 2.31125 , -0.80625 , 0 ,
434 'BSColDFCore' , 2.31125 , -1.34375 , 0 ,
435 'BSColDFCore' , 2.31125 , -1.88125 , 0 ,
436 'BSColDFSteel' , 0.147 , 2.15 , 0 ,
437 'BSColDFSteel' , 0.098 , 0 , 0 ,
438 'BSColDFSteel' , 0.147 , -2.15 , 0 } ;
439 ---
440 'ColINDSection' , 'Fiber' ,
441 {
442 'ColINDCover' , 5.1 , 2.575 , 0 ,
443 'ColINDCover' , 5.1 , -2.575 , 0 ,
444 'ColINDCover' , 0.91375 , 1.88125 , 0 ,
445 'ColINDCover' , 0.91375 , 1.34375 , 0 ,
446 'ColINDCover' , 0.91375 , 0.80625 , 0 ,
447 'ColINDCover' , 0.91375 , 0.26875 , 0 ,
448 'ColINDCover' , 0.91375 , -0.26875 , 0 ,
449 'ColINDCover' , 0.91375 , -0.80625 , 0 ,
450 'ColINDCover' , 0.91375 , -1.34375 , 0 ,
451 'ColINDCover' , 0.91375 , -1.88125 , 0 ,
452 'ColINDCore' , 2.31125 , 1.88125 , 0 ,
453 'ColINDCore' , 2.31125 , 1.34375 , 0 ,
454 'ColINDCore' , 2.31125 , 0.80625 , 0 ,
455 'ColINDCore' , 2.31125 , 0.26875 , 0 ,
456 'ColINDCore' , 2.31125 , -0.26875 , 0 ,
457 'ColINDCore' , 2.31125 , -0.80625 , 0 ,
458 'ColINDCore' , 2.31125 , -1.34375 , 0 ,
459 'ColINDCore' , 2.31125 , -1.88125 , 0 ,
460 'ColINDSteel' , 0.33 , 2.15 , 0 ,
461 'ColINDSteel' , 0.22 , 0 , 0 ,
462 'ColINDSteel' , 0.33 , -2.15 , 0 } ;
463 ---
464 'BSColINDSection' , 'Fiber' ,
465 {
466 'BSColINDCover' , 5.1 , 2.575 , 0 ,
467 'BSColINDCover' , 5.1 , -2.575 , 0 ,
468 'BSColINDCover' , 0.91375 , 1.88125 , 0 ,
469 'BSColINDCover' , 0.91375 , 1.34375 , 0 ,
470 'BSColINDCover' , 0.91375 , 0.80625 , 0 ,

```

```

466 'BSColNDCover' , 0.91375 , 0.26875 , 0 ,
467 'BSColNDCover' , 0.91375 , -0.26875 , 0 ,
468 'BSColNDCover' , 0.91375 , -0.80625 , 0 ,
469 'BSColNDCover' , 0.91375 , -1.34375 , 0 ,
470 'BSColNDCover' , 0.91375 , -1.88125 , 0 ,
471 'BSColNDCore' , 2.31125 , 1.88125 , 0 ,
472 'BSColNDCore' , 2.31125 , 1.34375 , 0 ,
473 'BSColNDCore' , 2.31125 , 0.80625 , 0 ,
474 'BSColNDCore' , 2.31125 , 0.26875 , 0 ,
475 'BSColNDCore' , 2.31125 , -0.26875 , 0 ,
476 'BSColNDCore' , 2.31125 , -0.80625 , 0 ,
477 'BSColNDCore' , 2.31125 , -1.34375 , 0 ,
478 'BSColNDCore' , 2.31125 , -1.88125 , 0 ,
479 'BSColNDSteel' , 0.33 , 2.15 , 0 ,
480 'BSColNDSteel' , 0.22 , 0 , 0 ,
481 'BSColNDSteel' , 0.33 , -2.15 , 0 };
482
483 'BSColNDFSection' , 'Fiber' ,
484 { 'BSColNDFCover' , 5.1 , 2.575 , 0 ,
485 'BSColNDFCover' , 5.1 , -2.575 , 0 ,
486 'BSColNDFCover' , 0.91375 , 1.88125 , 0 ,
487 'BSColNDFCover' , 0.91375 , 1.34375 , 0 ,
488 'BSColNDFCover' , 0.91375 , 0.80625 , 0 ,
489 'BSColNDFCover' , 0.91375 , 0.26875 , 0 ,
490 'BSColNDFCover' , 0.91375 , -0.26875 , 0 ,
491 'BSColNDFCover' , 0.91375 , -0.80625 , 0 ,
492 'BSColNDFCover' , 0.91375 , -1.34375 , 0 ,
493 'BSColNDFCover' , 0.91375 , -1.88125 , 0 ,
494 'BSColNDFCore' , 2.31125 , 1.88125 , 0 ,
495 'BSColNDFCore' , 2.31125 , 1.34375 , 0 ,
496 'BSColNDFCore' , 2.31125 , 0.80625 , 0 ,
497 'BSColNDFCore' , 2.31125 , 0.26875 , 0 ,
498 'BSColNDFCore' , 2.31125 , -0.26875 , 0 ,
499 'BSColNDFCore' , 2.31125 , -0.80625 , 0 ,
500 'BSColNDFCore' , 2.31125 , -1.34375 , 0 ,
501 'BSColNDFCore' , 2.31125 , -1.88125 , 0 ,
502 'BSColNDFSteel' , 0.33 , 2.15 , 0 ,
503 'BSColNDFSteel' , 0.22 , 0 , 0 ,
504 'BSColNDFSteel' , 0.33 , -2.15 , 0 };
505
506 'BeamSection' , 'Fiber' ,
507 { 'BeamCover' , 5.1 , 4.075 , 0 ,
508 'BeamCover' , 5.1 , -4.075 , 0 ,
509 'BeamCover' , 1.55125 , 3.19375 , 0 ,
510 'BeamCover' , 1.55125 , 2.28125 , 0 ,
511 'BeamCover' , 1.55125 , 1.36875 , 0 ,
512 'BeamCover' , 1.55125 , 0.45625 , 0 ,
513 'BeamCover' , 1.55125 , -0.45625 , 0 ,
514 'BeamCover' , 1.55125 , -1.36875 , 0 ,
515 'BeamCover' , 1.55125 , -2.28125 , 0 ,
516 'BeamCover' , 1.55125 , -3.19375 , 0 ,
517 'BeamCore' , 3.92375 , 3.19375 , 0 ,
518 'BeamCore' , 3.92375 , 2.28125 , 0 ,
519 'BeamCore' , 3.92375 , 1.36875 , 0 ,
520 'BeamCore' , 3.92375 , 0.45625 , 0 ,
521 'BeamCore' , 3.92375 , -0.45625 , 0 ,
522 'BeamCore' , 3.92375 , -1.36875 , 0 ,
523 'BeamCore' , 3.92375 , -2.28125 , 0 ,
524 'BeamCore' , 3.92375 , -3.19375 , 0 ,
525 'BeamSteel' , 0.44 , 3.65 , 0 ,
526 'BeamSteel' , 0.44 , -3.65 , 0 };
527
528 'BSBeamSection' , 'Fiber' ,
529 { 'BSBeamCover' , 5.1 , 4.075 , 0 ,
530 'BSBeamCover' , 5.1 , -4.075 , 0 ,
531 'BSBeamCover' , 1.55125 , 3.19375 , 0 ,
532 'BSBeamCover' , 1.55125 , 2.28125 , 0 ,
533 'BSBeamCover' , 1.55125 , 1.36875 , 0 ,
534 'BSBeamCover' , 1.55125 , 0.45625 , 0 ,
535 'BSBeamCover' , 1.55125 , -0.45625 , 0 ,
536 'BSBeamCover' , 1.55125 , -1.36875 , 0 ,
537 'BSBeamCover' , 1.55125 , -2.28125 , 0 ,
538 'BSBeamCover' , 1.55125 , -3.19375 , 0 ,
539 'BSBeamCore' , 3.92375 , 3.19375 , 0 ,
540 'BSBeamCore' , 3.92375 , 2.28125 , 0 ,
541 'BSBeamCore' , 3.92375 , 1.36875 , 0 ,
542 'BSBeamCore' , 3.92375 , 0.45625 , 0 ,
543 'BSBeamCore' , 3.92375 , -0.45625 , 0 ,
544 'BSBeamCore' , 3.92375 , -1.36875 , 0 ,
545 'BSBeamCore' , 3.92375 , -2.28125 , 0 ,
546 'BSBeamCore' , 3.92375 , -3.19375 , 0 ,
547 'BSBeamSteel' , 0.44 , 3.65 , 0 ,
548 'BSBeamSteel' , 0.44 , -3.65 , 0 };
549
550 'ColRigidSection' , 'general' ,
551 { 'ColRigid' , 36 , 108 };
552
553 'BeamRigidSection' , 'general' ,
554 { 'BeamRigid' , 54 , 364.5 };
555
556 --- *****
557 --- Set material properties .
558 ---
559 ---
560 concretemat = 'ConcreteLinearTensionSoftening'
561 steelmat = 'GiuffreMenegottoPinto'
562 sspringmat = 'Bilinear'
563 materials = {
564 --- Concrete (ConcreteLinearTensionSoftening)
565 Tag , mat_type , Ets , density , sc , ec , scu , ecu , lam , st
566 { 'ColDCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077 } ,
567 { 'ColDCore' , concretemat , 549.4505495 , 0 , -7.5 , -0.00546 , -7.35 , -0.01638 , 0.3 , 0.649519053 } ,
568 { 'BSColDCover' , concretemat , 105 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
569 { 'BSColDCore' , concretemat , 104.1666667 , 0 , -7.5 , -0.0288 , -7.35 , -0.0864 , 0.3 , 0.649519053 } ,
570 { 'BSColDFCover' , concretemat , 105 , -3.57 , -0.0136 , -1.19 , -0.0408 , 0.3 , 0.448121077 } ,
571 { 'BSColDFCore' , concretemat , 104.1666667 , 0 , -7.5 , -0.0288 , -6.75 , -0.0864 , 0.3 , 0.649519053 } ,
572 { 'ColNDCover' , concretemat , 549.2307692 , 0 , -3.57 , -0.0026 , -1.19 , -0.0078 , 0.3 , 0.448121077 } ,

```

```

573 {'ColNDCore', concretemat, 549.2957746, 0, -3.9, -0.00284, -3.51, -0.00852, 0.3, 0.46837485 },
574 {'BSColNDCover', concretemat, 105, 0, -3.57, -0.0136, -1.19, -0.0408, 0.3, 0.448121077},
575 {'BSColNDCore', concretemat, 106.8493151, 0, -3.9, -0.0146, -3.51, -0.0438, 0.3, 0.46837485 },
576 {'BSColNDFCover', concretemat, 144.2424242, 0, -3.57, -0.0099, -1.19, -0.0297, 0.3, 0.448121077},
577 {'BSColNDFCore', concretemat, 106.8493151, 0, -3.9, -0.0146, -3.51, -0.0438, 0.3, 0.46837485 },
578 {'BeamCover', concretemat, 549.2307692, 0, -3.57, -0.0026, -1.19, -0.0078, 0.3, 0.448121077},
579 {'BeamCore', concretemat, 549.2957746, 0, -3.9, -0.00284, -3.51, -0.00852, 0.3, 0.46837485 },
580 {'BSBeamCover', concretemat, 105, 0, -3.57, -0.0136, -1.19, -0.0408, 0.3, 0.448121077},
581 {'BSBeamCore', concretemat, 106.8493151, 0, -3.9, -0.0146, -3.51, -0.0438, 0.3, 0.46837485 },
582
583 --- Steel (GiuffreMenegottoPinto)
584 --- Tag, mat-type, E, density, sy, b, R0, cR1, cR2, a1, a2, a3, a4, s_init
585 {'ColDSteel', steelmat, 26500, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
586 {'BSColDSteel', steelmat, 5067, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
587 {'BSColDFSteel', steelmat, 6949, 0, 87.5, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
588 {'ColNDSteel', steelmat, 27300, 0, 80, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
589 {'BSColNDSteel', steelmat, 5220, 0, 80, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
590 {'BSColNDFSteel', steelmat, 5220, 0, 80, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
591 {'BeamSteel', steelmat, 27300, 0, 80, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
592 {'BSBeamSteel', steelmat, 5220, 0, 80, 0.01, 15, 0.925, 0.15, 0, 55, 0, 55, 0},
593
594 --- Shear spring (Bilinear)
595 --- Tag, mat-type, E, density, sy, b, a1, a2, a3, a4
596 {'BSColDSS', springmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
597 {'BSColDFSS', springmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
598 {'BSColNDSS', springmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
599 {'BSColNDFSS', springmat, 1690, 0, 78.2, 0.173, 0, 55, 0, 55},
600 {'BSBeamSS', springmat, 1545, 0, 75.8, 0.396, 0, 55, 0, 55},
601
602 --- Rigid element (Elastic)
603 --- Tag, mat-type, E, density
604 {'ColRigid', 'Elastic', 10000000000, 0},
605 {'BeamRigid', 'Elastic', 10000000000, 0}
606 }
607
608 -----
609 function dumptable(x)
610 local result = '{'
611 for i,v in ipairs(x) do
612   if (type(v) == 'table') then
613     result = result .. dumptable(v)
614   else
615     result = result .. v .. ', '
616   end
617 end
618 return result .. '}'\n'
619 end
620
621 --- print(dumptable(nodes))
622 --- print(dumptable(elements))
623 --- print(dumptable(materials))
624 --- print(dumptable(allsections))
625
626 -----
627 --- Preface ndim and ndofpn (ndim: dimension, ndofpn: number of degrees of freedom per node)
628 model = StructureModel(2,3)
629 --- Assign all input data to Mercury
630 model:addNodes(nodes)
631 model:addMaterials(materials)
632 model:addSections(allsections)
633 model:addElements(elements)
634
635 -----
636 --- constrain bottom nodes
637 for bay=1, nbays+1 do
638   model:constrainNode(columnnodename(bay,1,1), 1,1,1)
639 end
640
641 -----
642 solver = NonlinearSolver("initialstiffness", { displacementdeltatolerance=1e-5, iterations=100000})
643
644 -----
645 --- Monitoring node number
646 node06 = columnnodename(1,1,6);
647 node13 = columnnodename(1,2,7);
648 node20 = columnnodename(1,3,7);
649 node07 = columnnodename(1,1,7);
650 node14 = columnnodename(1,2,8);
651 node21 = columnnodename(1,3,8);
652 node48 = columnnodename(3,1,6);
653 node49 = columnnodename(3,1,7);
654 node01 = columnnodename(1,1,1);
655 node22 = columnnodename(2,1,1);
656 node43 = columnnodename(3,1,1);
657 node64 = columnnodename(4,1,1);
658
659 -----
660 disp_06_13_20 = {}
661 function disp1pertime(timestep)
662   dx06, dy06, dz06 = model:nodeDisplacements(node06)
663   dx13, dy13, dz13 = model:nodeDisplacements(node13)
664   dx20, dy20, dz20 = model:nodeDisplacements(node20)
665   table.insert(disp_06_13_20, dx06)
666   table.insert(disp_06_13_20, dy06)
667   table.insert(disp_06_13_20, dz06)
668   table.insert(disp_06_13_20, dx13)
669   table.insert(disp_06_13_20, dy13)
670   table.insert(disp_06_13_20, dz13)
671   table.insert(disp_06_13_20, dx20)
672   table.insert(disp_06_13_20, dy20)
673   table.insert(disp_06_13_20, dz20)
674 end
675
676 -----
677 disp_07_14_21 = {}
678 function disp2pertime(timestep)
679   dx07, dy07, dz07 = model:nodeDisplacements(node07)
680   dx14, dy14, dz14 = model:nodeDisplacements(node14)
681   dx21, dy21, dz21 = model:nodeDisplacements(node21)
682   table.insert(disp_07_14_21, dx07)
683   table.insert(disp_07_14_21, dy07)
684   table.insert(disp_07_14_21, dz07)
685   table.insert(disp_07_14_21, dx14)
686   table.insert(disp_07_14_21, dy14)
687   table.insert(disp_07_14_21, dz14)
688   table.insert(disp_07_14_21, dx21)
689   table.insert(disp_07_14_21, dy21)
690   table.insert(disp_07_14_21, dz21)

```

```

680 end
681 disp_48_49 = {}
682 function disp3pertime(timestep)
683     dx48,dy48,dz48 = model:nodeDisplacements(node48)
684     dx49,dy49,dz49 = model:nodeDisplacements(node49)
685     table.insert(disp_48_49,dx48)
686     table.insert(disp_48_49,dy48)
687     table.insert(disp_48_49,dz48)
688     table.insert(disp_48_49,dx49)
689     table.insert(disp_48_49,dy49)
690     table.insert(disp_48_49,dz49)
691 end
692
693 react = {}
694 function reactpertime(timestep)
695     fx01,fy01,fz01 = model:nodeRestoringForces(node01)
696     fx22,fy22,fz22 = model:nodeRestoringForces(node22)
697     fx43,fy43,fz43 = model:nodeRestoringForces(node43)
698     fx64,fy64,fz64 = model:nodeRestoringForces(node64)
699     table.insert(react,fx01)
700     table.insert(react,fy01)
701     table.insert(react,fz01)
702     table.insert(react,fx22)
703     table.insert(react,fy22)
704     table.insert(react,fz22)
705     table.insert(react,fx43)
706     table.insert(react,fy43)
707     table.insert(react,fz43)
708     table.insert(react,fx64)
709     table.insert(react,fy64)
710     table.insert(react,fz64)
711     print(timestep);
712 end
713
714 --- *****
715 earthquakeloading = LoadDescription()
716 accelamp = 1.0*g
717 earthquakeloading:addLoad({'groundmotion',earthquakefile .. ',dt=0.005',1,accelamp})
718 --- *****
719 ---transientanalysis = DynamicAnalysis("hht",model,solver,earthquakeloading,--[dt,alpha,beta,gamma]] 0.005,alpha,beta,gamma)
720 transientanalysis = DynamicAnalysis("hybridhht",model,solver,earthquakeloading,--[dt,alpha,beta,gamma,iteration]] 0.005,alpha,beta,gamma,iteration)
721 model:setRayleighCoefficients(1.1770500887303603,0.0015992014979696392)
722 --- *****
723 transientanalysis:addcallback(disp1pertime,"timestep")
724 transientanalysis:addcallback(disp2pertime,"timestep")
725 transientanalysis:addcallback(disp3pertime,"timestep")
726 transientanalysis:addcallback(reactpertime,"timestep")
727 --- *****
728 ---transientanalysis:solve(15281)
729 transientanalysis:solve(152810)
730 --- *****
731 --- Set output file
732 function writedata6(x,fname)
733     local f = assert(io.open(fname,'w'))
734     local writenl = 0
735     for i,v in ipairs(x) do
736         f:write(v," ")
737         writenl = writenl + 1
738         --- length of row size: writenl
739         if (writenl > 5) then
740             writenl = 0
741             f:write("\n")
742         end
743     end
744     f:close()
745 end
746
747 function writedata9(x,fname)
748     local f = assert(io.open(fname,'w'))
749     local writenl = 0
750     for i,v in ipairs(x) do
751         f:write(v," ")
752         writenl = writenl + 1
753         --- length of row size: writenl
754         if (writenl > 8) then
755             writenl = 0
756             f:write("\n")
757         end
758     end
759     f:close()
760 end
761
762 function writedata12(x,fname)
763     local f = assert(io.open(fname,'w'))
764     local writenl = 0
765     for i,v in ipairs(x) do
766         f:write(v," ")
767         writenl = writenl + 1
768         --- length of row size: writenl
769         if (writenl > 11) then
770             writenl = 0
771             f:write("\n")
772         end
773     end
774     f:close()
775 end
776 --- *****
777 ---writedata9(disp_06_13_20,'Ex29HHT_06_13_20_displ.dat')
778 ---writedata9(disp_07_14_21,'Ex29HHT_07_14_21_displ.dat')
779 ---writedata6(disp_48_49,'Ex29HHT_48_49_displ.dat')
780 ---writedata12(react,'Ex29HHT_01_22_43_64.react.dat')
781
782 writedata9(disp_06_13_20,'Ex29Shing_06_13_20_displ.dat')
783 writedata9(disp_07_14_21,'Ex29Shing_07_14_21_displ.dat')
784 writedata6(disp_48_49,'Ex29Shing_48_49_displ.dat')
785 writedata12(react,'Ex29Shing_01_22_43_64.react.dat')
786 --- *****

```


Chapter 4

NONLINEAR PUSHOVER ANALYSIS

first part requires some major editing

4.1 pushover

A pushover analysis is a nonlinear static procedure wherein monotonically increasing lateral loads are applied to the structure till a target displacement is achieved or the structure is unable to resist further loads.

A pushover analysis is associated with a load pattern. In this case, use the same later load pattern that was used for the seismic design. It is necessary to apply dead and live loads before doing a pushover analysis. Since the pushover procedure is nonlinear, it is necessary to setup a new load case for dead and live which is also nonlinear. The recent advent of performance based design has brought the static nonlinear analysis, also known as sequential yield analysis, or simply “push-over” analysis has gained significant popularity during the past few years. It is a static, nonlinear procedure in which the magnitude of the structural loading is incrementally increased in accordance with a certain predefined pattern. Typically, a predefined lateral load pattern is distributed along the building height. The lateral forces are then monotonically increased in constant proportion with a displacement control at the top of the building until a certain level of deformation is reached. The target top displacement may be the deformation expected in the design earthquake in case of designing a new structure, or the drift corresponding to structural collapse for assessment purposes. The method allows tracing the sequence of yielding and failure on the member and the structure levels as well as the progress of the overall capacity curve of the structure, (?).

With the increase in the magnitude of the loading, weak links and failure modes of the structure are found. The loading is monotonic with the effects of the cyclic behavior and load reversals being estimated by using a modified monotonic force-deformation criteria and with damping approximations. Static pushover analysis is an attempt by the structural engineering profession to evaluate the real strength of the structure and it promises to be a useful and effective tool for performance based design. The ATC-40 and FEMA-273 documents have developed modeling procedures, acceptance criteria and analysis procedures for pushover analysis. These documents define force-deformation criteria for hinges used in pushover analysis.

The design of structures based on plastic approach is called **limit design** and is at the core of most modern design codes (ACI, AISC).

4.2 Shakedown

A structure subjected to a general variable load can collapse even if the loads remain inside the elastic plastic domain of the load space. Thus the elasto plastic domain represents a safe domain only for monotonic loads. Under general, non-monotonic loading, a structure can nevertheless fail by incremental collapse or plastic fatigue. The behavior of a structure is termed **shakedown**

4.3 Control Method

In pushover analysis it is highly desirable to first subject the structure to an initial force without constrained degrees of freedom for imposed pushover displacements. This would allow the structure to properly deform, be subjected to the correct state of initial stress prior to the application of the actual imposed pushover displacements. This section presents a simple method to perform such an analysis in one step. Traditional method in Sec. 4.4 would require two separate analysis shown in Fig. ?? . Sec. 4.5 describes pushover analysis with imposed displacement on unconstrained degrees of freedom.

4.4 Classical Pushover

A nonlinear pushover analysis entails two parts, Fig. 4.3:

Vertical Load is the first step where the structure is subjected to the vertical load (such as gravity and possibly a fraction of the live load).

Lateral Loas is the application of increasing lateral force or displacement up to failure. Imposed displacement is preferred as it enables us to capture the post-peak response.

A nonlinear pushover analysis can not separate the vertical loads from the lateral loads. Application of the vertical loads will results in an initial state of stress to which those induced by the lateral loads must be added.

Traditional methods entail performing the first analysis under load control, and then results are either used as part of the restart (correct), or they are added to the results of the displacement control when this one is complete (erroneous).

Since only imposed displacements can enable us to capture the post-peak response, the question is how should those displacements be applied.

With reference to Fig. 4.2, one could apply a single imposed displacement at the top of the discretized building. However, the resulting displacements may not be compatible with the dynamic response of the structure.

A better approach consists in first evaluating the first mode shape, determine the normalized displacements (with respect to the one of the top floor) of each floor α_i , and then use this α_i distribution for the imposed lateral displacements in the subsequent nonlinear pushover analysis.

4.5 Improved Pushover Analysis Procedure

With reference to Fig. 4.3, we shall examine the two steps of the pushover analysis.

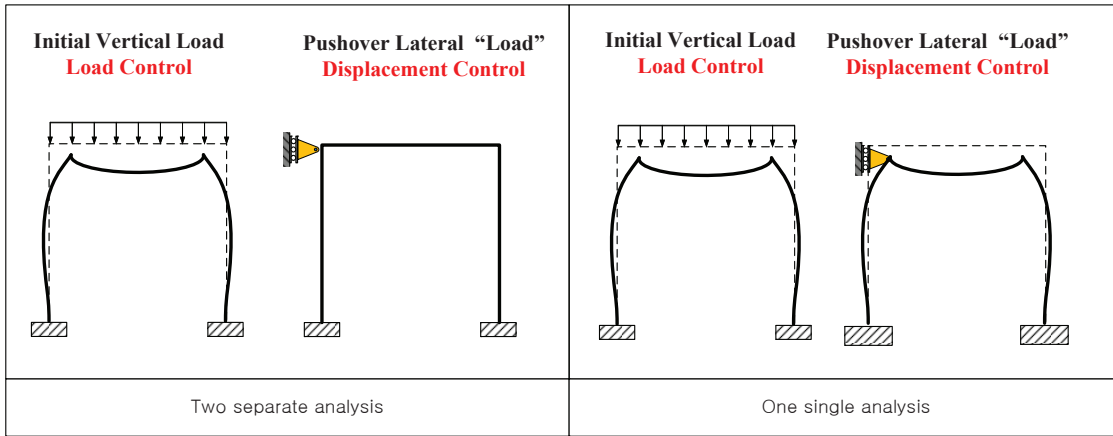


Figure 4.1: Pushover analysis

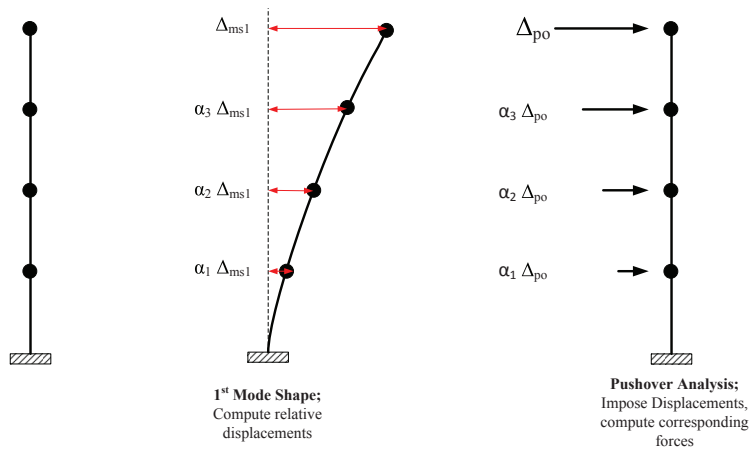


Figure 4.2: Determination of pushover displacements magnitudes

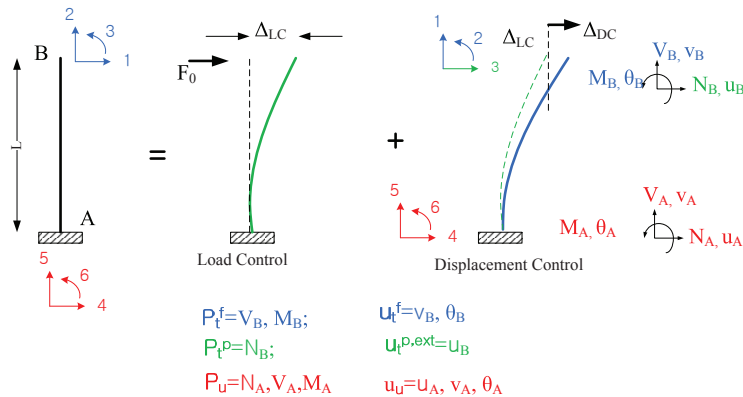


Figure 4.3: Pushover Analysis of a Frame

4.5.1 Load Control

This first part is consistent with classical methods where the governing equilibrium equation of a nonlinear system can be expressed as

$$\underbrace{\begin{Bmatrix} \mathbf{P}_t^{ext} \\ \mathbf{P}_u \end{Bmatrix}}_{\mathbf{P}} = \underbrace{\begin{bmatrix} \mathbf{K}_{tt} & \mathbf{K}_{tu} \\ \mathbf{K}_{ut} & \mathbf{K}_{uu} \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{Bmatrix} \mathbf{u}_t \\ \mathbf{u}_u \end{Bmatrix}}_{\mathbf{u}} \quad (4.1)$$

where \mathbf{P} , \mathbf{u} and \mathbf{K} are the nodal force, nodal displacement and stiffness terms expressed in the structural level, and the subscript u and t are associated with Γ_u and Γ_t where the essential and natural boundary conditions are defined.

Equilibrium is satisfied when external nodal forces are equal to the internal ones,

$$\mathbf{P}^{ext} - \mathbf{P}^{int} = 0 \quad (4.2)$$

and \mathbf{P}^{int} is determined from state determination in Ch. 1.

In nonlinear system, the incremental displacement vector is defined by

$$\Delta \mathbf{u}_n = \mathbf{u}_n - \mathbf{u}_{n-1}$$

and the corresponding nodal displacement vector at the k^{th} iteration is given by:

$$\mathbf{u}_n^k = \mathbf{u}_n^{k-1} + \delta \mathbf{u}_n^k$$

where $\delta \mathbf{u}_n^k = \mathbf{u}_n^k - \mathbf{u}_n^{k-1}$.

We can rewrite Eq. 4.1 in incremental form at the k^{th} iteration of the n^{th} increment as:

$$\underbrace{\begin{Bmatrix} \mathbf{P}_{t,n}^{R,k} = \mathbf{P}_{t,n}^{ext} - \mathbf{P}_{t,n}^{int,k\checkmark} \\ \mathbf{P}_{u,n}^{R,k} = \mathbf{P}_{u,n}^{k\checkmark} - \mathbf{P}_{u,n}^{int,k\checkmark} \end{Bmatrix}}_{\mathbf{P}_n^{R,k} = \delta \mathbf{P}_n^k} = \underbrace{\begin{bmatrix} \mathbf{K}_{tt,n}^{k-1} & \mathbf{K}_{tu,n}^{k-1} \\ \mathbf{K}_{ut,n}^{k-1} & \mathbf{K}_{uu,n}^{k-1} \end{bmatrix}}_{\mathbf{K}_n^{k-1}} \underbrace{\begin{Bmatrix} \delta \mathbf{u}_{t,n}^{k\checkmark} \\ \Delta \mathbf{u}_{u,n}^{ext\checkmark} \end{Bmatrix}}_{\delta \mathbf{u}_n^k} \quad (4.3)$$

where, superscript ' \checkmark ' and '?' refer to known and unknown quantities. It should be noted that the known imposed displacement $\Delta \mathbf{u}_{u,n}^{ext\checkmark}$ is applied in the first iteration only, and then it is set to $\mathbf{0}$.

The incremental displacements can now be expressed as:

$$\begin{aligned} \delta \mathbf{u}_{t,n}^{k\checkmark} &= \mathbf{u}_{t,n}^{k\checkmark} - \mathbf{u}_{t,n}^{k-1\checkmark} \\ \Delta \mathbf{u}_{u,n}^{ext\checkmark} &= \mathbf{u}_{u,n}^{ext\checkmark} - \mathbf{u}_{u,n-1}^{ext\checkmark} \end{aligned}$$

and the essence of the finite element analysis is to first determine the displacement $\delta \mathbf{u}_{t,n}^{k\checkmark}$, and then $\mathbf{P}_{u,n}^{k\checkmark}$ from Eq. 4.3.

In this preliminary analysis, the structure is subjected to its original load, the (yet to be constrained dofs in the pushover analysis) are left as free. At the end of this analysis, we keep track of the nodal displacement

4.5.2 Displacement Control

Following the load control part, we now move to displacement control for the pushover analysis.

Rewriting Eq. 4.1 with prescribed external nodal displacements

$$\underbrace{\begin{Bmatrix} \mathbf{P}_t^{f\checkmark} \\ \mathbf{P}_t^{p\checkmark} \\ \mathbf{P}_u^{p\checkmark} \end{Bmatrix}} = \underbrace{\begin{bmatrix} \mathbf{K}_{tt}^{ff} & \mathbf{K}_{tt}^{fp} & \mathbf{K}_{tu} \\ \mathbf{K}_{tt}^{pf} & \mathbf{K}_{tt}^{pp} & \mathbf{K}_{tu} \\ \mathbf{K}_{ut} & \mathbf{K}_{uu} & \mathbf{K}_{uu} \end{bmatrix}} \underbrace{\begin{Bmatrix} \mathbf{u}_t^{f\checkmark} \\ \mathbf{u}_t^{p,ext\checkmark} \\ \mathbf{u}_u^{p\checkmark} \end{Bmatrix}} \quad (4.4)$$

where, superscript ' f ' and ' p ' refer to unknown and prescribed displacement. With reference to Fig. ??, we would have

$$\begin{aligned} \mathbf{P}_t^{f\checkmark} &= [V_B, M_B]^T \\ \mathbf{P}_t^{p\checkmark} &= [N_B]^T \\ \mathbf{P}_u^{p\checkmark} &= [N_A, V_A, M_A]^T \\ \mathbf{u}_t^{f\checkmark} &= [v_B, \theta_B]^T \\ \mathbf{u}_t^{p,ext\checkmark} &= [u_B]^T \\ \mathbf{u}_u^{p\checkmark} &= [u_A, v_A, \theta_A]^T \end{aligned}$$

Again, in the context of a nonlinear pushover analysis, we rewrite the \mathbf{K}_{tt} submatrix in incremental form for the k^{th} iteration of the n^{th} increment, and since we are now in displacement control, there are no more applied forces. Thus:

$$\begin{Bmatrix} \mathbf{0} \\ \delta \mathbf{P}_{t,n}^{p,k-1\checkmark} \end{Bmatrix} = \begin{bmatrix} \mathbf{K}_{tt,n}^{ff,k-1} & \mathbf{K}_{tt,n}^{fp,k-1} \\ \mathbf{K}_{tt,n}^{pf,k-1} & \mathbf{K}_{tt,n}^{pp,k-1} \end{bmatrix} \begin{Bmatrix} \delta \mathbf{u}_t^{f,k-1\checkmark} \\ \Delta \mathbf{u}_t^{p,ext\checkmark} \end{Bmatrix} \quad (4.5)$$

We should keep in mind that we are primarily interested in $\delta \mathbf{P}_{t,n}^{p,k-1\checkmark}$ which is the vector of force(s) corresponding to the imposed displacement(s) in the pushover analysis. Also, it should be noted that $\Delta \mathbf{u}_t^{p,ext\checkmark}$ is only applied in the first iteration, and in subsequent ones, it is $\mathbf{0}$.

We first solve for the unknown displacements corresponding to the free degrees of freedom $\delta \mathbf{u}_t^{f,k-1\checkmark}$.

$$\begin{aligned} \mathbf{0} &= \mathbf{K}_{tt,n}^{ff,k-1} \cdot \delta \mathbf{u}_t^{f,k-1\checkmark} + \mathbf{K}_{tt,n}^{fp,k-1} \cdot \delta \mathbf{u}_t^{p,ext\checkmark} \\ \delta \mathbf{u}_t^{f,k-1\checkmark} &= -[\mathbf{K}_{tt,n}^{ff,k-1}]^{-1} \cdot \mathbf{K}_{tt,n}^{fp,k-1} \cdot \Delta \mathbf{u}_t^{p,ext\checkmark} \end{aligned} \quad (4.6)$$

Then we address the second row of Eq. to solve for the forces corresponding to the imposed displacements in the pushover analysis, $\delta \mathbf{P}_{t,n}^{p,k-1\checkmark}$.

It should be noted that the original ID matrix which refers to the global degrees of freedom has to be readjusted into a new one dispID.

Also, the internal forces computed from the load control part are transferred to the displacement control part. Thus the entire nonlinear pushover analysis can be performed in a single analysis in Mercury without the need for a restart. The algorithm for this procedure is shown in Fig. 4.4.

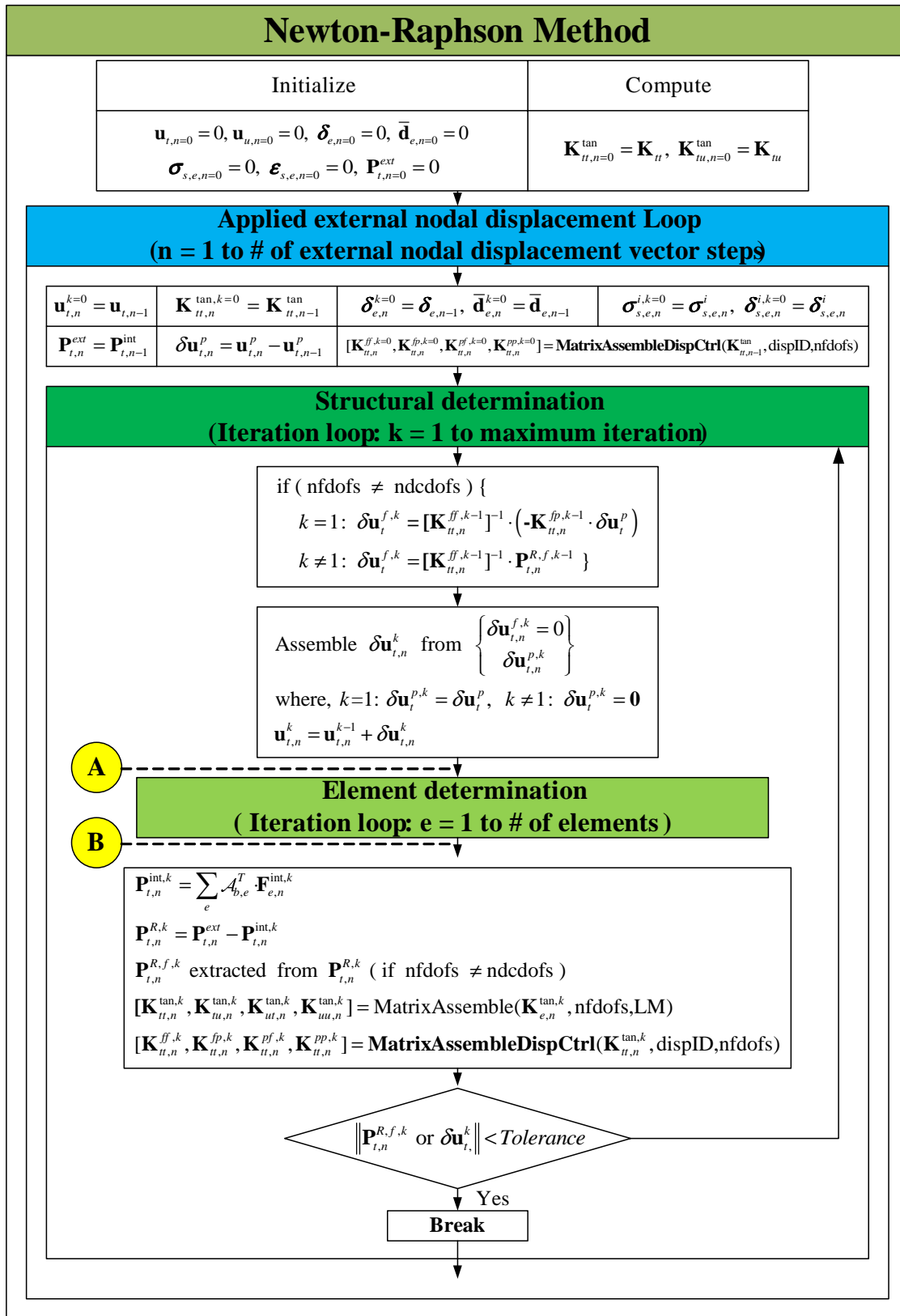


Figure 4.4: Implementation for displacement control

Chapter 5

DYNAMIC ANALYSIS

5.1 Preliminaries

1 When the frequency of excitation of a structure is less than about a third of its lowest natural frequency of vibration, then we can neglect inertia effects and treat the problem as a quasi-static one.

2 If the structure is subjected to an impact load, than one must be primarily concerned with (stress) wave propagation. In such a problem, we often have high frequencies and the duration of the dynamic analysis is about the time it takes for the wave to travel across the structure.

3 If inertia forces are present, then we are confronted with a dynamic problem and can analyse it through any one of the following solution procedures:

1. Natural frequencies and mode shapes (only linear elastic systems)
2. Time history analysis through modal analysis (again linear elastic), or direct integration.

4 Methods of structural dynamics are essentially independent of finite element analysis as these methods presume that we already have the stiffness, mass, and damping matrices. Those matrices may be obtained from a single degree of freedom system, from an idealization/simplification of a frame structure, or from a very complex finite element mesh. The time history analysis procedure remains the same.

5.1.1 Variational Formulation

5 In a general three-dimensional continuum, the equations of motion of an elementary volume Ω without damping is

$$\mathbf{L}^T \boldsymbol{\sigma} + \mathbf{b} = \mathbf{m} \ddot{\mathbf{u}} \quad (5.1)$$

where \mathbf{m} is the mass density matrix equal to $\rho \mathbf{I}$, and \mathbf{b} is the vector of body forces. The Differential operator \mathbf{L} is

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \end{bmatrix} \quad (5.2)$$

6 For linear elastic material

$$\boldsymbol{\sigma} = \mathbf{D}_e \boldsymbol{\varepsilon} \quad (5.3)$$

for nonlinear material, the constitutive equations can be written as

$$\dot{\boldsymbol{\sigma}} = \mathbf{D}_i \dot{\boldsymbol{\varepsilon}} \quad (5.4)$$

where \mathbf{D}_i is the tangent stiffness matrix.

7 Whereas Eq. 5.1 describes the body motion in a strong sense, a weak formulation is obtained by the principle of minimum complementary virtual work (or Weighted Residual/Galerkin)

$$\int_{\Omega} \delta \mathbf{u}^T [\mathbf{L}^T \boldsymbol{\sigma} + \mathbf{b} - \mathbf{m} \ddot{\mathbf{u}}] d\Omega = 0 \quad (5.5)$$

Applying Gauss divergence theorem, Eq. ??:

$$\int_{\Omega} \int_A \phi \operatorname{div} \mathbf{q} dA = \oint_s \phi \mathbf{q}^T \mathbf{n} ds - \int_A (\nabla \phi)^T \mathbf{q} dA \quad (5.6)$$

and recalling that $\mathbf{L} \mathbf{u} = \boldsymbol{\varepsilon}$,

$$\int_{\Omega} \delta \mathbf{u}^T \mathbf{L}^T \boldsymbol{\sigma} d\Omega = - \int_{\Omega} \delta \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} d\Omega + \int_{\Gamma} \delta \mathbf{u}^T \boldsymbol{\sigma} \mathbf{n} d\Gamma \quad (5.7)$$

thus, Eq. 5.5 transforms into

$$\int_{\Omega} [\delta \mathbf{u}^T (\mathbf{m} \ddot{\mathbf{u}} - \mathbf{b}) + \delta \boldsymbol{\varepsilon}^T \boldsymbol{\sigma}] d\Omega - \int_{\Gamma} \delta \mathbf{u}^T \mathbf{t} d\Gamma = 0 \quad (5.8)$$

so far no assumption has been made with regard to material behavior.

8 Next we will seek the spatial discretization of the virtual work equation (Eq. 5.8). From the previous chapters, we have the following relations

$$\begin{aligned} \mathbf{u} &= \mathbf{N} \bar{\mathbf{u}}; & \delta \mathbf{u}^T &= \delta \bar{\mathbf{u}}^T \mathbf{N}^T; & \ddot{\mathbf{u}} &= \mathbf{N} \ddot{\bar{\mathbf{u}}} \\ \mathbf{B} &= \mathbf{L} \mathbf{N}; & \boldsymbol{\varepsilon} &= \mathbf{B} \bar{\mathbf{u}}; & \delta \boldsymbol{\varepsilon}^T &= \delta \bar{\mathbf{u}}^T \mathbf{B}^T \\ \dot{\boldsymbol{\varepsilon}} &= \mathbf{B} \dot{\bar{\mathbf{u}}} \end{aligned} \quad (5.9)$$

9 For implicit time integration, equilibrium was reached at time step $n - 1$, and we write the equation of equilibrium for time step n . thus Eq. 5.8 transforms into

$$\int_{\Omega} \delta \mathbf{u}^T \mathbf{m} d\Omega \ddot{\mathbf{u}}^{t+\Delta t} + \int_{\Omega} \delta \boldsymbol{\varepsilon}^T \boldsymbol{\sigma}_{t,n} d\Omega = \int_{\Gamma} \delta \mathbf{u}^T \mathbf{t}_{t,n} d\Gamma + \int_{\Omega} \delta \mathbf{u}^T \mathbf{b}_{t,n} d\Omega \quad (5.10)$$

10 For linear problems

$$\boldsymbol{\sigma}_{t,n} = \mathbf{D}_e \mathbf{B} \bar{\mathbf{u}}_{t,n} \quad (5.11)$$

11 Again, upon proper substitution, this would yield

$$\delta \bar{\mathbf{u}}^T \left[\underbrace{\int_{\Omega} \mathbf{N}^T \mathbf{m} N d\Omega}_{\mathbf{M}_{tt}} \ddot{\bar{\mathbf{u}}}_{t,n} + \underbrace{\int_{\Omega} \mathbf{B}^T \mathbf{D}_e \mathbf{B} d\Omega}_{\mathbf{K}} \bar{\mathbf{u}}_{t,n} - \underbrace{\left(\int_{\Omega} \mathbf{N}^T \mathbf{P}_{t,n}^{ext} d\Omega + \int_{\Gamma} \mathbf{N}^T \mathbf{t}_{t,n} d\Gamma \right)}_{\mathbf{P}_{t,n}^{ext}} \right] = 0 \quad (5.12)$$

or

$$\boxed{\mathbf{M} \ddot{\bar{\mathbf{u}}}_{t,n} + \mathbf{K} \bar{\mathbf{u}}_{t,n} = \mathbf{P}_{t,n}^{ext}} \quad (5.13)$$

Which represents the semi-discrete linear equation of motion in the implicit time integration.

Generalizing the previous equation to include the effect of damping, and replacing $\bar{\mathbf{u}}$ by \mathbf{u} we have

$$\mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_t + \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_t + \mathbf{P}_t^{int} = \mathbf{P}_t^{ext} \quad (5.14)$$

where \mathbf{M}_{tt} and \mathbf{C}_{tt} are the mass and viscous damping matrices for the idealization of the structure; $\ddot{\mathbf{u}}_t$ is the nodal acceleration vector, $\dot{\mathbf{u}}_t$ is the nodal velocity vector, \mathbf{P}_t^{int} is the static restoring or internal nodal force vector resulting from the nodal displacement vector \mathbf{u}_t , and \mathbf{P}_t^{ext} is the vector of applied nodal forces due to a seismic loading. Numerical methods for solving Eq. 5.14 are divided into two major categories; explicit and implicit methods. This chapter will limit its coverage to implicit schemes and in particular: 1) Newmark β method, 2) the Hilber-Hughes-Taylor (HHT) method.

5.1.2 Mass Representation

There are two possible representation of the mass matrix: lumped and consisten.

Nodal lumped masses contribute only to the diagonal terms of the mass matrix and the terms associated with the rotational degrees of freedom are often taken as zero. However, in some cases the rotational inertias can be accounted for. The x , y and z inertia quantities may be different in a structure particularly in a two-dimensional analysis where the frame being analyzed is flanked by adjoining frames which carry vertical loads but have relatively insignificant lateral stiffness. In this case the vertical inertia is associated only with that of the frame being analyzed while the horizontal inertia has to represent the sum of the inertia contributions of all frames being supported, in the lateral direction, by the frame being analyzed.

The mass matrix may take one of two forms.

1. In this formulation it is assumed that all the masses are concentrated at the end nodes. Though not exactly correct, the advantage of this model is that we will have a diagonal matrix which can be easily inverted. The term associated with rotation is often neglected.
2. The consistent mass model uses a kinematically equivalent mass matrix where inertia forces are associated with all degrees of freedom. This will result in a mass matrix for the structures with the same skyline form as that of the stiffness matrix. The consistent mass model requires a greater computational cost in the multiplication by the nodal accelerations to get the inertia forces at each time-step in the analysis. It also includes all natural frequencies and consequently gives a slight bias to the frequency content of the structure.

5.1.2.1 Lumped mass

In prismatic 2D framed structure, the lumped mass matrix in the global reference is simply given by

$$\mathbf{M}_e = \rho \cdot A \cdot L_e \begin{bmatrix} 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_r \cdot L_e^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha_r \cdot L_e^2 \end{bmatrix} \quad (5.15)$$

Here α_r is a nonnegative coefficient for rotation. α_r zero will result in a singular mass matrix which is undesirable if we have to invert the mass matrix. An *ad hoc* solution to define α_r is to imagine that a uniform slender bar of length $L_e/2$ and mass $m/2$ is attached to each node and rotates with it. The associated mass moment of inertia would $I_z = (m/2)(L_e/2)^2/3$, and consequently $\alpha_r = 1/24$, (?). It should be noted that models based on lumped mass can run substantially faster than those based on consistent mass.

5.1.2.2 Consistent mass

The consistent mass matrix is defined in the local reference frame as

$$\mathbf{m}_e = \int_0^{L_e} \rho \cdot A(x) \cdot \mathbf{N}_d(x)^T \cdot \mathbf{N}_d(x) dx$$

where,

$$\mathbf{N}_d(x) = \begin{bmatrix} 1 - \frac{x}{L_e} & 0 & 0 & \frac{x}{L_e} & 0 & 0 \\ 0 & 1 + 2\left(\frac{x}{L_e}\right)^3 - 3\left(\frac{x}{L_e}\right)^2 & x\left(1 - \frac{x}{L_e}\right)^2 & 0 & -2\left(\frac{x}{L_e}\right)^3 + 3\left(\frac{x}{L_e}\right)^2 & x\left(\left(\frac{x}{L_e}\right)^2 - \frac{x}{L_e}\right) \end{bmatrix}$$

Hence,

$$\mathbf{m}_e = \frac{\rho \cdot A \cdot L_e}{420} \begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22 \cdot L_e & 0 & 54 & -13 \cdot L_e \\ 0 & 22 \cdot L_e & 4 \cdot L_e^2 & 0 & 13 \cdot L_e & -3 \cdot L_e^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13 \cdot L_e & 0 & 156 & -22 \cdot L_e \\ 0 & -13 \cdot L_e & -3 \cdot L_e^2 & 0 & -22 \cdot L_e & 4 \cdot L_e^2 \end{bmatrix} \quad (5.16)$$

The mass matrix is then transformed into the global reference by rewriting Eq. 5.16 in terms of the rotation matrix, $\mathbf{\Gamma}_e$.

$$\mathbf{M}_e = \mathbf{\Gamma}_e^T \cdot \mathbf{m}_e \cdot \mathbf{\Gamma}_e$$

5.1.3 Damping Representation

All structures are damped, otherwise their oscillations will never stop. Damping can be viewed as a frictional force which dissipates energy, and can take different form.

The most commonly used form of damping is the so-called viscous or Rayleigh damping which, when inserted in the equation of motion, has the following form

$$\mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n} + \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n} + \mathbf{P}_{t,n}^{int} = \mathbf{P}_{t,n}^{ext}$$

where, $\ddot{\mathbf{u}}_{t,n}$, $\dot{\mathbf{u}}_{t,n}$ and $\mathbf{u}_{t,n}$ are the nodal acceleration, velocity, and displacement vectors at the current time step, respectively; $\mathbf{P}_{t,n}^{int}$ is the static restoring or internal nodal force vector at the current time step.

Damping is supposed to model the dissipation of energy. In a nonlinear analysis, this is accounted for by some constitutive models which include hysteresis damping, such as the Modified Kent and Park model for concrete (Sect. 2.2.1.1). In linear elastic analysis, the most common form of damping is the so-called viscous damping (better known as Rayleigh damping). In this simplification, we assume the presence of a viscous damper (which by definition is sensitive to velocity) between the structure and an external fixed point (mass proportional), and another set of dampers inside the structure connecting all the degrees of freedom (stiffness proportional damper). Hence, we assume that

$$\mathbf{C}_{tt} = a_m \cdot \mathbf{M}_{tt} + b_k \cdot \mathbf{K}_{tt} \quad (5.17)$$

where, a_m and b_k are coefficients which pre-multiply the mass and stiffness terms respectively. Eq. 5.17 is known as proportional Rayleigh damping, Fig. 5.1.

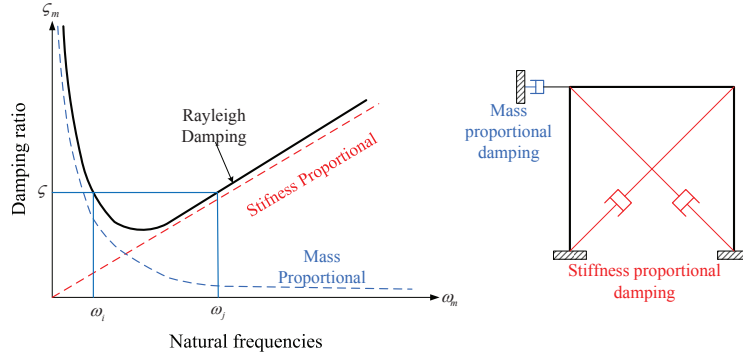


Figure 5.1: Rayleigh damping

Rayleigh damping is the most widely used (but not only) model for damping. The coefficients a_m and b_k in Eq. 5.17 are calculated based upon two circular frequencies (ω_1 and ω_2 , radians/sec.) to be damped at ζ_1 and ζ_2 respectively. Where ω_m and ζ_m are the circular frequency and the damping ratio of the m^{th} mode.

We recall that the damping ratio for a single degree of freedom (SDF) for mode m is given by

$$\zeta_m = \frac{C_{tt,m}}{2 \cdot M_{tt,m} \cdot \omega_m} \quad (5.18)$$

Thus for mass proportional damping of multi degree of freedom (MDF) system, with $\mathbf{C}_{tt,m} = a_m \cdot \mathbf{M}_{tt,m}$, this would lead to

$$\zeta_m = \frac{a_m}{2} \cdot \frac{1}{\omega_m}$$

The damping ratio is thus inversely proportional to the natural frequency and a_m can be selected to obtain a specified damping ratio in any one mode i or

$$a_m = 2 \cdot \zeta_i \cdot \omega_i \quad (5.19)$$

Similarly, and recalling that $\mathbf{K}_{tt} \cdot \phi_m = \omega_m^2 \cdot \mathbf{M}_{tt} \cdot \phi_m$, a stiffness proportional damping $\mathbf{C}_{tt,m} = b_k \cdot \mathbf{K}_{tt,m}$ combined with Eq. 5.18 will lead to

$$\zeta_m = \frac{b_k}{2} \cdot \omega_m \quad (5.20)$$

In this case the damping ratio is proportional to the natural frequency and b_k can be selected to obtain a specified damping ratio in any one mode j or

$$b_k = \frac{2 \cdot \zeta_j}{\omega_j} \quad (5.21)$$

Combining Eq. 5.19 and 5.21 leads to the following linear equations

$$\frac{1}{2} \begin{bmatrix} \frac{1}{\omega_i} & \omega_i \\ \frac{1}{\omega_j} & \omega_j \end{bmatrix} \begin{Bmatrix} a_m \\ b_k \end{Bmatrix} = \begin{Bmatrix} \zeta_i \\ \zeta_j \end{Bmatrix} \quad (5.22)$$

If one assumes the same damping ratio ζ for both modes (reasonable practical assumption), then

$$a_m = \zeta \frac{2\omega_i \cdot \omega_j}{\omega_i + \omega_j}$$

$$b_k = \zeta \frac{2}{\omega_i + \omega_j}$$

Again, it should be emphasized that different damping coefficients should be used in linear and in nonlinear analysis (specially if the nonlinear constitutive model accounts for hysteresis damping). Furthermore, if Rayleigh damping is used in a nonlinear analysis, then coefficients a_m and b_k may have to be updated at each time increment to reflect the change in the tangential stiffness matrix \mathbf{K}_t .

5.1.4 Euler Methods

Euler method is a numerical procedure to solve initial value ordinary differential equations (as in structural dynamics). In other words, given a solution at time t_n , how do we get the solution at time t_{n+1} .

In our case, we discretize space by the finite element, and discretize time by the finite difference. Spatial discretization is by now well understood, the question is how do we discretize time, or in its most elementary form a derivative.

As with Newton's method, it all starts with the Taylor's series. We begin with the forward version

$$y(t_n + h) \equiv y_{n+1} = y(t_n) + h \left. \frac{dy}{dt} \right|_{t_n} + O(h^2) \quad (5.23-a)$$

$$\Rightarrow y_{n+1} \simeq y_n + hf(y_n, t_n) \quad (5.23-b)$$

This *forward Euler* method is also referred to as *explicit* since y_{n+1} is given explicitly in terms of known quantities such as y_n and $f(y_n, t_n)$ and there is no equation to solve. Explicit methods are easy to implement but are conditionally stable (i.e. h should be smaller than a critical value).

An alternative formulation is one based on the *backward Euler* method which starts with the following Taylor series expansion

$$y(t_n) \equiv y_n = y(t_{n+1} - h) = y(t_{n+1}) - h \left. \frac{dy}{dt} \right|_{t_{n+1}} + O(h^2) \quad (5.24-a)$$

$$\Rightarrow y_{n+1} \simeq y_n + hf(y_{n+1}, t_{n+1}) \quad (5.24-b)$$

This is an *implicit* method since $f(y_{n+1}, t_{n+1})$ is not known and a (usually) nonlinear equation must be solved at every time step (possibly by the Newton-Raphson method). Evidently, this is more computationally expensive than the explicit method, however the method is *unconditionally stable*.

The geometric interpretation of Euler's method is shown in Fig. 5.2. We note that the implicit method (at the cost of a Newton-Raphson solution) always provides an "exact" solution. In the context of structural dynamics, we can say that equilibrium is satisfied. This is not the case in the explicit method.

■ Example 5.1. Euler Method

Solve the following ordinary linear first order differential equation:

$$\frac{dy}{dt} = 1 + (t - y)^2; \quad 2 \leq t \leq 3; \quad y(2) = 1 \quad (5.25)$$

Solution:

Forward Euler with $h=0.1$

$$y_{n+1} = y_n + hf(y_n, t_n) \quad (5.26-a)$$

$$y_1 = 1 + 0.1 [1 + (2 - 1)^2] = 1.2 \quad (5.26-b)$$

The Backward Euler will give

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \quad (5.27-a)$$

$$y_1 = 1 + 0.1 [1 + (2.1 - y_1)^2] \quad (5.27-b)$$

$$0 = 0.1y_1^2 - 1.42y_1 + 1.541 \quad (5.27-c)$$

$$y_1 = 1.1839 \quad (5.27-d)$$

In this case we had a quadratic equation to solve, however in general we may have to use Newton's method to solve for y_n . ■

5.2 Time Integration Methods

Time integration is performed through a finite difference solution of the time discretization. We will (have to) assume that equilibrium was reached in the previous time step $n - 1$

$$\mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n-1} + \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n-1} + \mathbf{P}_{t,n-1}^{int} = \mathbf{P}_{t,n-1}^{ext}$$

At the current time step n , the equation of motion will be given by

$$\mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n} + \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n} + \mathbf{P}_{t,n}^{int} = \mathbf{P}_{t,n}^{ext} \quad (5.28)$$

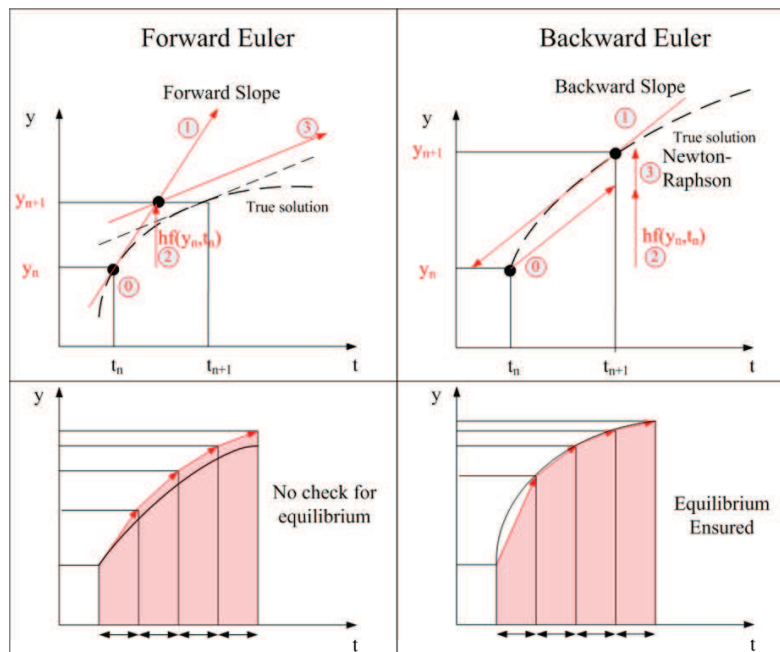


Figure 5.2: Geometric interpretation of Euler's method

There are two broad classes of solution to solve this equation

1. Explicit methods

The nodal displacement vector $\mathbf{u}_{t,n}$ at the current time step can be calculated from a function of structural response solutions at the previous time step $n - 1$. The effective stiffness matrix for solving the equation of motion consists of mass matrix and/or damping matrix. This indicates that it is not necessary to invert the structural stiffness matrix.

2. Implicit methods

The nodal displacement vector $\mathbf{u}_{t,n}$ at the current time step is often represented as a function of structural response quantities with both the previous time step $n - 1$ and the current time step n . For this reason, iteration procedure such as Newton-Raphson iterative method has to be applied so as to solve equations and to achieve convergence if the response is nonlinear.

Advantages of using explicit methods for transient analysis include the simplicity of the algorithm, easy implementation and their fast and efficient computation without the demand for iterations. In addition, there are no need for the knowledge of the tangent stiffness during the test. However, explicit methods are only conditionally stable. This means that the computed response may grow without bound when the product of the time step and the highest natural frequency of the structure $\Delta t \cdot \omega$ exceeds a limitation. In analyzing a structure with a very high natural frequency ω , a small time interval needs to be used and it can be too small to be practical. On the other hand, most implicit methods are unconditionally stable. This means that it is possible to analyze a multi-degree-of-freedom system with high frequency modes using a reasonably large time step without stability problems. Furthermore, implicit methods can be customized to provide favorable numerical energy dissipation properties. In addition to the prescribed viscous damping matrix \mathbf{C}_{vt} , implicit methods provide numerical damping which can be controlled by certain parameters. The value of these parameters can be selected in order to suppress the spurious higher-mode responses, excited by experimental errors during the test. However, an iterative solution procedure for a nonlinear system is computationally more demanding. It introduces the possibility of inducing undesirable loading and unloading hysteresis in a structure whose behavior can be highly sensitive to prior inelastic deformation history. Last but not least implicit methods are ideal candidates for real time hybrid simulation.

5.2.1 Newmark β method

Euler's method is now generalized for second order differential equations (equation of motion)

$$\begin{aligned} \begin{pmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{pmatrix}^n &= \begin{pmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{pmatrix}^{n-1} + \Delta t \begin{pmatrix} \dot{\mathbf{u}} \\ \ddot{\mathbf{u}} \end{pmatrix}^{n-1} && \text{Forward Euler} \\ \begin{pmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{pmatrix}^n &= \begin{pmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{pmatrix}^{n-1} + \Delta t \begin{pmatrix} \dot{\mathbf{u}} \\ \ddot{\mathbf{u}} \end{pmatrix}^n && \text{Backward Euler} \end{aligned} \quad (5.29)$$

Newmark's method is a variation of Euler's method, where higher order derivatives (Eq. 5.28) are computed with lower accuracy for the sake of efficiency.

Again, we first consider the Taylor series expansions of the nodal displacement and velocity vector terms about the values at the previous time $n - 1$.

$$\begin{aligned} \mathbf{u}_{t,n} &\approx \mathbf{u}_{t,n-1} + \frac{\partial \mathbf{u}_{t,n-1}}{\partial t} \Delta t + \frac{\partial^2 \mathbf{u}_{t,n-1}}{\partial t^2} \frac{\Delta t^2}{2!} + \frac{\partial^3 \mathbf{u}_{t,n-1}}{\partial t^3} \frac{\Delta t^3}{3!} + \dots \\ \dot{\mathbf{u}}_{t,n} &\approx \dot{\mathbf{u}}_{t,n-1} + \frac{\partial^2 \mathbf{u}_{t,n-1}}{\partial t^2} \Delta t + \frac{\partial^3 \mathbf{u}_{t,n-1}}{\partial t^3} \frac{\Delta t^2}{2!} + \dots \end{aligned} \quad (5.30)$$

The above two equations represent the approximate displacement and velocity vectors ($\mathbf{u}_{t,n}$ and $\dot{\mathbf{u}}_{t,n}$) except for high order terms

Table 5.1: Properties of the Newmark β method

Method	Type	β	γ	Stability condition	Order of accuracy
Constant acceleration	Implicit	1/4	1/2	Unconditional	2
Linear acceleration	implicit	1/6	1/2	$\Delta t \leq 2\sqrt{3}/\omega$	2
Central difference	Explicit	0	1/2	$\Delta t \leq 2/\omega$	2

of Taylor series. We represent the last terms of the above two equations as follow:

$$\begin{aligned} \frac{\partial^3 \mathbf{u}_{t,n}}{\partial^3 t} \frac{\Delta t^3}{3!} &\approx \frac{\frac{\partial^2 \mathbf{u}_{t,n}}{\partial^2 t} - \frac{\partial^2 \mathbf{u}_{t,n-1}}{\partial^2 t}}{\Delta t} \frac{\Delta t^3}{3!} \approx (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1}) \frac{\Delta t^2}{3!} \\ &\approx \beta (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1}) \Delta t^2 \end{aligned} \quad (5.31)$$

$$\begin{aligned} \frac{\partial^3 \mathbf{u}_{t,n}}{\partial^3 t} \frac{\Delta t^2}{2!} &\approx \frac{\frac{\partial^2 \mathbf{u}_{t,n}}{\partial^2 t} - \frac{\partial^2 \mathbf{u}_{t,n-1}}{\partial^2 t}}{\Delta t} \frac{\Delta t^2}{2!} \approx (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1}) \frac{\Delta t}{2!} \\ &\approx \gamma (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1}) \Delta t \end{aligned} \quad (5.32)$$

where β and γ are parameters which depict numerical approximations. These parameters will account for Eq. 5.31 and Eq. 5.32 including additional terms which were dropped from Taylor series approximation. Substituting Eq. 5.31 and Eq. 5.32 into Eq. ?? and Eq. 5.30, respectively, we obtain the two equations as follow:

$$\mathbf{u}_{t,n} = \mathbf{u}_{t,n-1} + \Delta t \cdot \dot{\mathbf{u}}_{t,n-1} + \frac{\Delta t^2}{2} \cdot \ddot{\mathbf{u}}_{t,n-1} + \Delta t^2 \cdot \beta \cdot (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1}) \quad (5.33)$$

$$\dot{\mathbf{u}}_{t,n} = \dot{\mathbf{u}}_{t,n-1} + \Delta t \cdot \ddot{\mathbf{u}}_{t,n-1} + \Delta t \cdot \gamma \cdot (\ddot{\mathbf{u}}_{t,n} - \ddot{\mathbf{u}}_{t,n-1})$$

Hence, we obtain the Newmark β method, which consists of the following equations:

$$\mathbf{P}_{t,n}^{ext} = \mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n} + \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n} + \mathbf{P}_{t,n}^{int} \quad (5.34)$$

$$\mathbf{u}_{t,n} = \mathbf{u}_{t,n-1} + \Delta t \cdot \dot{\mathbf{u}}_{t,n-1} + \frac{\Delta t^2}{2} [(1-2\beta)\ddot{\mathbf{u}}_{t,n-1} + 2\beta \cdot \ddot{\mathbf{u}}_{t,n}] \quad (5.35)$$

$$\dot{\mathbf{u}}_{t,n} = \dot{\mathbf{u}}_{t,n-1} + \Delta t [(1-\gamma)\ddot{\mathbf{u}}_{t,n-1} + \gamma \cdot \ddot{\mathbf{u}}_{t,n}] \quad (5.36)$$

where, Eq. 5.34 is the equation of equilibrium expressed at time $n+1$, and Eq. 5.35 and Eq.5.36 are finite difference formulas describing the evolution of the approximation solution. β and γ are parameters that determine the stability and accuracy characteristics. Stability conditions for the Newmark β method follows:

1. unconditionally stable if

$$\begin{aligned} \gamma &\geq \frac{1}{2} \\ \beta &\geq \frac{\gamma}{2} \end{aligned}$$

2. conditionally stable if

$$\begin{aligned} \gamma &\geq \frac{1}{2} \\ \beta &< \frac{\gamma}{2} \end{aligned}$$

with the following stability limit:

$$\omega \cdot \Delta t \leq \Omega_{crit} = \frac{\xi(\gamma - 1/2) + [\gamma/2 - \beta + \xi^2(\gamma - 1/2)^2]^{1/2}}{\gamma/2 - \beta} \quad (5.37)$$

where ω is maximum natural frequency, Ω_{crit} is critical sampling frequency, and ξ is the damping ratio.

The constant acceleration (trapezoidal rule) method is implicit and unconditionally stable. The linear acceleration method is implicit and conditionally stable. The central difference method is conditionally stable. Note that if $\gamma = 1/2$ has no effect on stability. In practice it is more convenient to express Eq. 5.37 in terms of the period of vibration, $T = 2\pi/\omega$, in which case Eq. 5.37 becomes $\Delta t/T \leq \Omega_{crit}/(2\pi)$. In the case of the linear acceleration, $\Delta t/T$ is calculated as follows:

$$\frac{\Delta t}{T} \leq \frac{1}{2\pi} \frac{1}{\sqrt{\gamma - 2\beta}} = 0.551$$

A summary for the Newmark β method is shown in Table. 5.1.

5.2.1.1 Newmark β implicit method

The Newmark β implicit method can be expressed with Eq. 5.34, 5.35, and 5.36. Rewriting Eq. 5.35 and 5.36:

$$\mathbf{u}_{t,n} = \tilde{\mathbf{u}}_{t,n} + \Delta t^2 \cdot \beta \cdot \ddot{\mathbf{u}}_{t,n} \quad (5.38)$$

$$\dot{\mathbf{u}}_{t,n} = \tilde{\dot{\mathbf{u}}}_{t,n} + \Delta t \cdot \gamma \cdot \ddot{\mathbf{u}}_{t,n} \quad (5.39)$$

where we have labeled the known quantities at time step $n-1$ as $(\tilde{\cdot})$

$$\tilde{\mathbf{u}}_{t,n} = \mathbf{u}_{t,n-1} + \Delta t \cdot \dot{\mathbf{u}}_{t,n-1} + \frac{\Delta t^2}{2} (1-2\beta)\ddot{\mathbf{u}}_{t,n-1}$$

$$\tilde{\dot{\mathbf{u}}}_{t,n} = \dot{\mathbf{u}}_{t,n-1} + \Delta t(1-\gamma)\ddot{\mathbf{u}}_{t,n-1}$$

Rewriting Eq. 5.38, we can solve for $\ddot{\mathbf{u}}_{t,n}$:

$$\ddot{\mathbf{u}}_{t,n} = \frac{\mathbf{u}_{t,n} - \tilde{\mathbf{u}}_{t,n}}{\Delta t^2 \cdot \beta} \quad (5.40)$$

Substituting Eq. 5.40 into Eq. 5.39, we can rewrite $\dot{\mathbf{u}}_{t,n}$:

$$\dot{\mathbf{u}}_{t,n} = \tilde{\dot{\mathbf{u}}}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} (\mathbf{u}_{t,n} - \tilde{\mathbf{u}}_{t,n}) \quad (5.41)$$

Substituting Eq. 5.40 and Eq. 5.41 into Eq. 5.34, we have:

$$\mathbf{M}_{tt} \left[\frac{\mathbf{u}_{t,n} - \tilde{\mathbf{u}}_{t,n}}{\Delta t^2 \cdot \beta} \right] + \mathbf{C}_{tt} \left[\tilde{\dot{\mathbf{u}}}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} (\mathbf{u}_{t,n} - \tilde{\mathbf{u}}_{t,n}) \right] + \mathbf{P}_{t,n}^{int} = \mathbf{P}_{t,n}^{ext} \quad (5.42)$$

Rewriting Eq. 5.42,

$$\begin{aligned} & \frac{1}{\Delta t^2 \cdot \beta} \mathbf{M}_{tt} \cdot \mathbf{u}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} \mathbf{C}_{tt} \cdot \mathbf{u}_{t,n} + \mathbf{P}_{t,n}^{int} \\ &= \mathbf{P}_{t,n}^{ext} + \frac{1}{\Delta t^2 \cdot \beta} \mathbf{M}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} \mathbf{C}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} - \mathbf{C}_{tt} \cdot \tilde{\dot{\mathbf{u}}}_{t,n} \end{aligned} \quad (5.43)$$

If the trial solutions in given iteration step k are $\mathbf{u}_{t,n}^k$, and $\mathbf{P}_{t,n}^{int,k}$, then it does not satisfy the equations of motion. Hence, we can write for this particular step with residual force vector $\mathbf{P}_{t,n}^{R,k}$:

$$\mathbf{P}_{t,n}^{R,k} = \mathbf{P}_{t,n}^{ext} + \overline{\mathbf{M}}_{tt} (\tilde{\mathbf{u}}_{t,n} - \mathbf{u}_{t,n}^k) - \mathbf{C}_{tt} \cdot \tilde{\dot{\mathbf{u}}}_{t,n} - \mathbf{P}_{t,n}^{int,k}$$

where,

$$\overline{\mathbf{M}}_{tt} = \frac{\mathbf{M}_{tt} + \Delta t \cdot \gamma \cdot \mathbf{C}}{\Delta t^2 \cdot \beta}$$

Using initial stiffness iterative method, we can solve for $\Delta \mathbf{u}_{t,n}^k$:

$$\mathbf{P}_{t,n}^{R,k} = \mathbf{K}_{eff} \cdot \Delta \mathbf{u}_{t,n}^k \quad (5.44)$$

where, \mathbf{K}_{eff} is the effective stiffness matrix, and $\Delta \mathbf{u}_{t,n}^k$ is:

$$\Delta \mathbf{u}_{t,n}^k = \mathbf{u}_{t,n} - \mathbf{u}_{t,n}^k$$

In elastic section, we can rewrite $\mathbf{P}_{t,n}^{int}$ to compute the effective stiffness matrix with initial stiffness matrix \mathbf{K}_{tt} :

$$\mathbf{P}_{t,n}^{int} = \mathbf{K}_{tt} \cdot \mathbf{u}_{t,n} \quad (5.45)$$

Substituting Eq. 5.45 into Eq. 5.43, we can solve for $\mathbf{u}_{t,n}$:

$$\mathbf{K}_{eff} \cdot \mathbf{u}_{t,n} = \mathbf{P}_{t,n}^{ext} + \overline{\mathbf{M}}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} - \mathbf{C}_{tt} \cdot \tilde{\dot{\mathbf{u}}}_{t,n}$$

where,

$$\mathbf{K}_{eff} = \overline{\mathbf{M}}_{tt} + \mathbf{K}_{tt}$$

From Eq. 5.44, we can solve for $\delta \mathbf{u}_{t,n}^k$ and the updated displacement vector $\mathbf{u}_{t,n}^{k+1}$ at the next iteration step $k+1$:

$$\begin{aligned} \delta \mathbf{u}_{t,n}^k &= [\mathbf{K}_{eff}]^{-1} \cdot \mathbf{P}_{t,n}^{R,k} \\ \mathbf{u}_{t,n}^{k+1} &= \mathbf{u}_{t,n}^k + \delta \mathbf{u}_{t,n}^k \end{aligned}$$

Fig. 5.3 and 5.4 explain the implementation of transient analysis using the Newmark β implicit method with flexibility-based 2D beam-column elements.

5.2.2 Hilber-Hughes-Taylor Method

5.2.2.1 General Formulation

A major drawback of Newmark β method is the tendency for high frequency noise to persist in the solution. On the other hand, when linear damping or artificial viscosity is added via the parameter γ , the accuracy is markedly degraded. The α method, (?) improves numerical dissipation for high frequency without degrading the accuracy as much.

Equation of motion in HHT method is written at current time step n (forward difference) as:

$$\mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n} + \mathbf{P}_{t,n}^{int} = \mathbf{P}_{t,n}^{ext}$$

Seeking an approximate solution of this equation by one-step difference, we write,

$$\mathbf{M}_{tt} \cdot \dot{\mathbf{u}}_{t,n} + (1 + \alpha) \mathbf{P}_{t,n}^{int} - \alpha \cdot \mathbf{P}_{t,n-1}^{int} = \mathbf{P}_{t,n}^{ext}$$

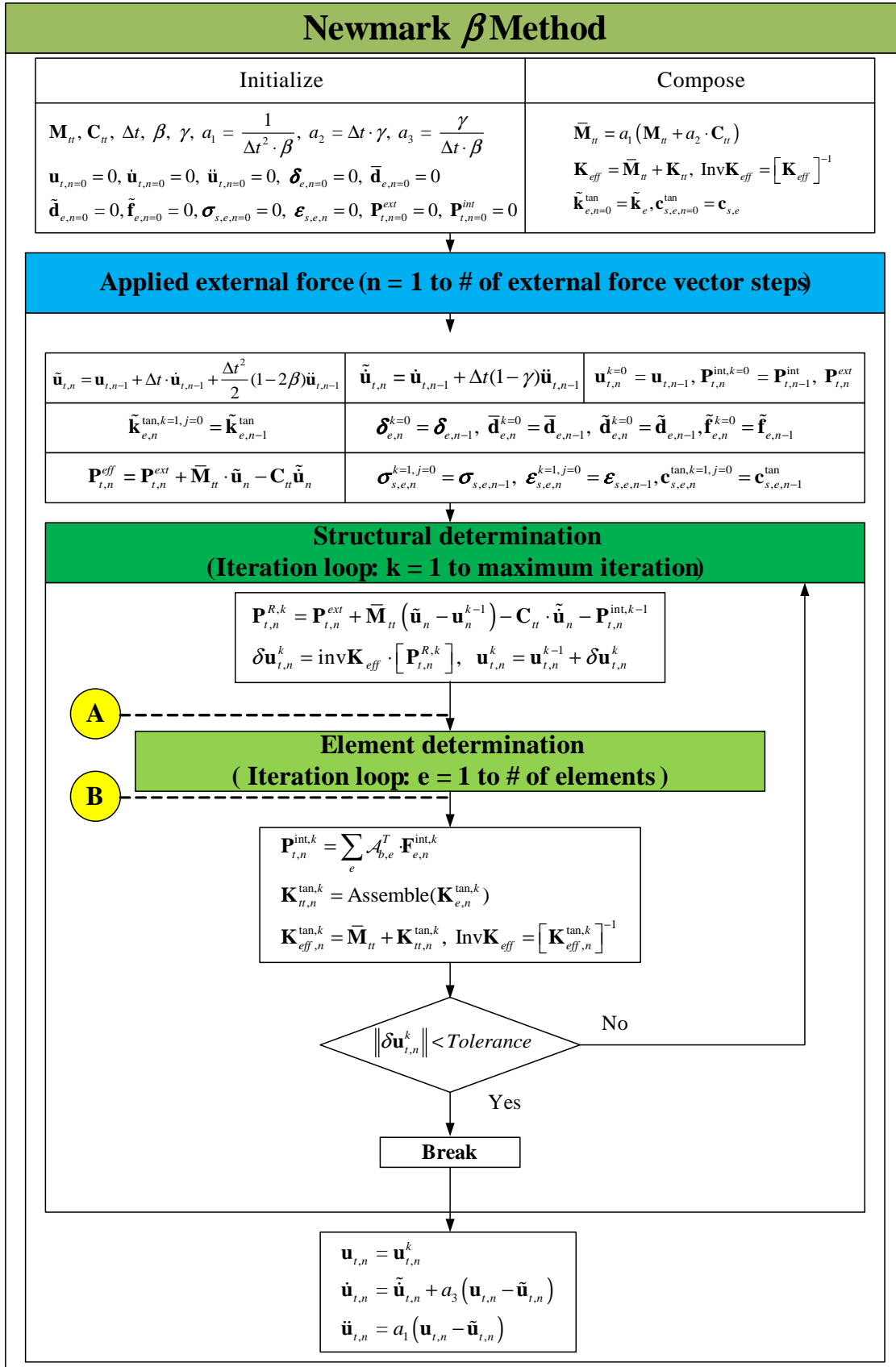
with Eq. 5.33.

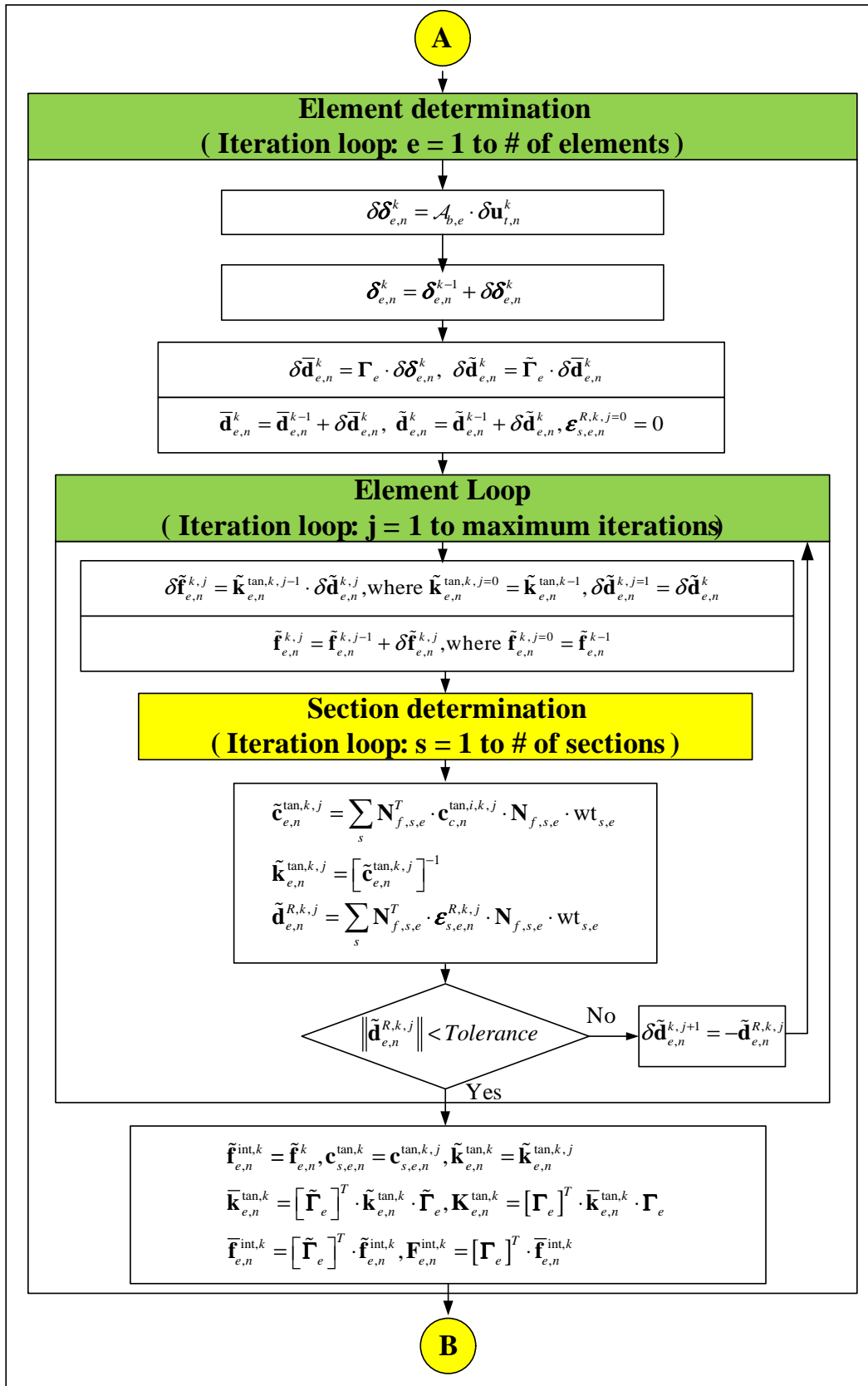
We note that the HHT method introduces $\alpha(\mathbf{P}_{t,n}^{int} - \mathbf{P}_{t,n-1}^{int})$ which is akin of stiffness proportional damping (indeed it is commonly said that the α method provides numerical damping). If the above equation is expanded, effect of damping introduced, and possible material nonlinearity introduced, we obtain:

$$(1 + \alpha) \mathbf{P}_{t,n}^{ext} - \alpha \mathbf{P}_{t,n-1}^{ext} = \mathbf{M}_{tt} \cdot \ddot{\mathbf{u}}_{t,n} + (1 + \alpha) \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n} - \alpha \cdot \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n-1} + (1 + \alpha) \mathbf{P}_{t,n}^{int} - \alpha \cdot \mathbf{P}_{t,n-1}^{int} \quad (5.46)$$

If $-1/3 \leq \alpha \leq 0$, $\beta = (1 - \alpha)^2/4$, and $\gamma = (1 - 2\alpha)/2$, then the α method is unconditionally stable and has a second-order accuracy. Hence, Eq. 5.33 and 5.46 must all be simultaneously satisfied through an iterative method.

Assuming that we have obtained the response at the previous time step $n-1$, i.e. $\mathbf{u}_{t,n-1}$, $\dot{\mathbf{u}}_{t,n-1}$ and $\ddot{\mathbf{u}}_{t,n-1}$ which satisfy the equation of motion, we now seek to determine the solution at the current time step n by iteration. First of all, we need to determine

Figure 5.3: Flow chart (1) of transient analysis using the Newmark β implicit method

Figure 5.4: Flow chart (2) of transient analysis using the Newmark β implicit method

effective external force and effective stiffness. These are calculated from Eq. 5.40, 5.41 and 5.46.

$$\begin{aligned} & \frac{1}{\Delta t^2 \cdot \beta} \mathbf{M}_{tt} \cdot \mathbf{u}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} (1 + \alpha) \mathbf{C}_{tt} \cdot \mathbf{u}_{t,n} + (1 + \alpha) \mathbf{P}_{t,n}^{int} \\ &= (1 + \alpha) \mathbf{P}_{t,n}^{ext} - \alpha \cdot \mathbf{P}_{t,n-1}^{ext} + \frac{1}{\Delta t^2 \cdot \beta} \mathbf{M}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} + \frac{\gamma}{\Delta t \cdot \beta} (1 + \alpha) \mathbf{C}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} \\ & - (1 + \alpha) \mathbf{C}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} + \alpha \cdot \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{n-1} + \alpha \cdot \mathbf{P}_{t,n}^{int} \end{aligned} \quad (5.47)$$

The trial solutions in iteration step k are $\mathbf{u}_{t,n}^k$, and $\mathbf{P}_{t,n}^{int,k}$, does not necessarily satisfy the equations of motion. Hence, we can write for this particular step:

$$\begin{aligned} \mathbf{P}_{t,n}^{R,k} &= (1 + \alpha) \mathbf{P}_{t,n}^{ext} - \alpha \cdot \mathbf{P}_{t,n-1}^{ext} + \overline{\mathbf{M}}_{tt} \left(\tilde{\mathbf{u}}_{t,n} - \mathbf{u}_{t,n}^k \right) - (1 + \alpha) \mathbf{C}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} + \alpha \cdot \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n-1} \\ & - (1 + \alpha) \mathbf{P}_{t,n}^{int,k} + \alpha \cdot \mathbf{P}_{t,n-1}^{int} \end{aligned}$$

where,

$$\overline{\mathbf{M}}_{tt} = \frac{\mathbf{M}_{tt} + \Delta t \cdot \gamma (1 + \alpha) \mathbf{C}_{tt}}{\Delta t^2 \cdot \beta}$$

and $\mathbf{P}_{t,n}^{R,k}$ is the residual force vector.

Using the initial stiffness iterative method, we can solve for $\Delta \mathbf{u}_{t,n}^k$:

$$\mathbf{P}_{t,n}^{R,k} = \mathbf{K}_{eff} \cdot \Delta \mathbf{u}_{t,n}^k \quad (5.48)$$

where, \mathbf{K}_{eff} is the effective stiffness matrix, and $\Delta \mathbf{u}_{t,n}^k$ is:

$$\Delta \mathbf{u}_{t,n}^k = \mathbf{u}_{t,n} - \mathbf{u}_{t,n}^k$$

In elastic section, we can express $\mathbf{P}_{t,n}^{int}$ to compute the effective stiffness matrix with initial stiffness matrix \mathbf{K}_{tt} as:

$$\mathbf{P}_{t,n}^{int} = \mathbf{K}_{tt} \cdot \mathbf{u}_{t,n} \quad (5.49)$$

Substituting Eq. 5.49 into Eq. 5.47, we solve for $\mathbf{u}_{t,n}$:

$$\begin{aligned} \mathbf{K}_{eff} \cdot \mathbf{u}_{t,n} &= (1 + \alpha) \mathbf{P}_{t,n}^{ext} - \alpha \cdot \mathbf{P}_{t,n-1}^{ext} + \overline{\mathbf{M}}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} \\ & - (1 + \alpha) \cdot \mathbf{C}_{tt} \cdot \tilde{\mathbf{u}}_{t,n} + \alpha \cdot \mathbf{C}_{tt} \cdot \dot{\mathbf{u}}_{t,n-1} + \alpha \cdot \mathbf{P}_{t,n-1}^{int} \end{aligned}$$

where,

$$\mathbf{K}_{eff} = \overline{\mathbf{M}}_{tt} + (1 + \alpha) \mathbf{K}_{tt}$$

From Eq. 5.48, we solve for $\delta \mathbf{u}_{t,n}^k$ and the updated displacement vector $\mathbf{u}_{t,n}^{k+1}$ at the next iteration step $k + 1$:

$$\begin{aligned} \delta \mathbf{u}_{t,n}^k &= [\mathbf{K}_{eff}]^{-1} \cdot \mathbf{P}_{t,n}^{R,k} \\ \mathbf{u}_{t,n}^{k+1} &= \mathbf{u}_{t,n}^k + \delta \mathbf{u}_{t,n}^k \end{aligned}$$

Finally, we note that:

1. α introduces a damping that grows with the ratio of time increment to the period of vibration of a node.
2. Negative values of α provide damping
3. If $\alpha = 0$, we have no artificial damping (energy preseving) and is exactly the constant acceleration (trapezoidal rule) - Newmark's β method if $\beta = 1/4$ and $\gamma = 1/2$.
4. Maximum value is $\alpha = -1/3$ which provides the maximum artificial damping. This results in a damping ratio of about 6% when the time increment is 40% of the period of oscillation of the mode being studied and smaller if the oscillation period increases.
5. This artificial damping is not very substantial for realistic time increment and low frequencies, but is non-negligible for high frequencies.
6. A default value of -0.05 is recommended.

Fig. 5.5 explains the implementation of transient analysis using the HHT implicit method in structural level. Element state determination is identical to Fig. 5.4.

5.3 Free Vibration

12 The governing equation for the free (undamped) vibration of a structure is

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = 0 \quad (5.50)$$

where the motion is referred to being free, since there are no applied loads.

13 By assuming a *harmonic motion*

$$\mathbf{u} = \phi \sin \omega t \quad (5.51)$$

the *natural frequencies* ω and the corresponding *mode shapes* ϕ can be determined from the *generalized* eigenvalue problem

$$\omega^2 \mathbf{M}\phi = \mathbf{K}\phi \quad (5.52)$$

or

$$(\mathbf{K} - \omega^2 \mathbf{M})\phi = 0 \quad (5.53)$$

Since ϕ is nontrivial

$$|\mathbf{K} - \omega^2 \mathbf{M}| = 0 \quad (5.54)$$

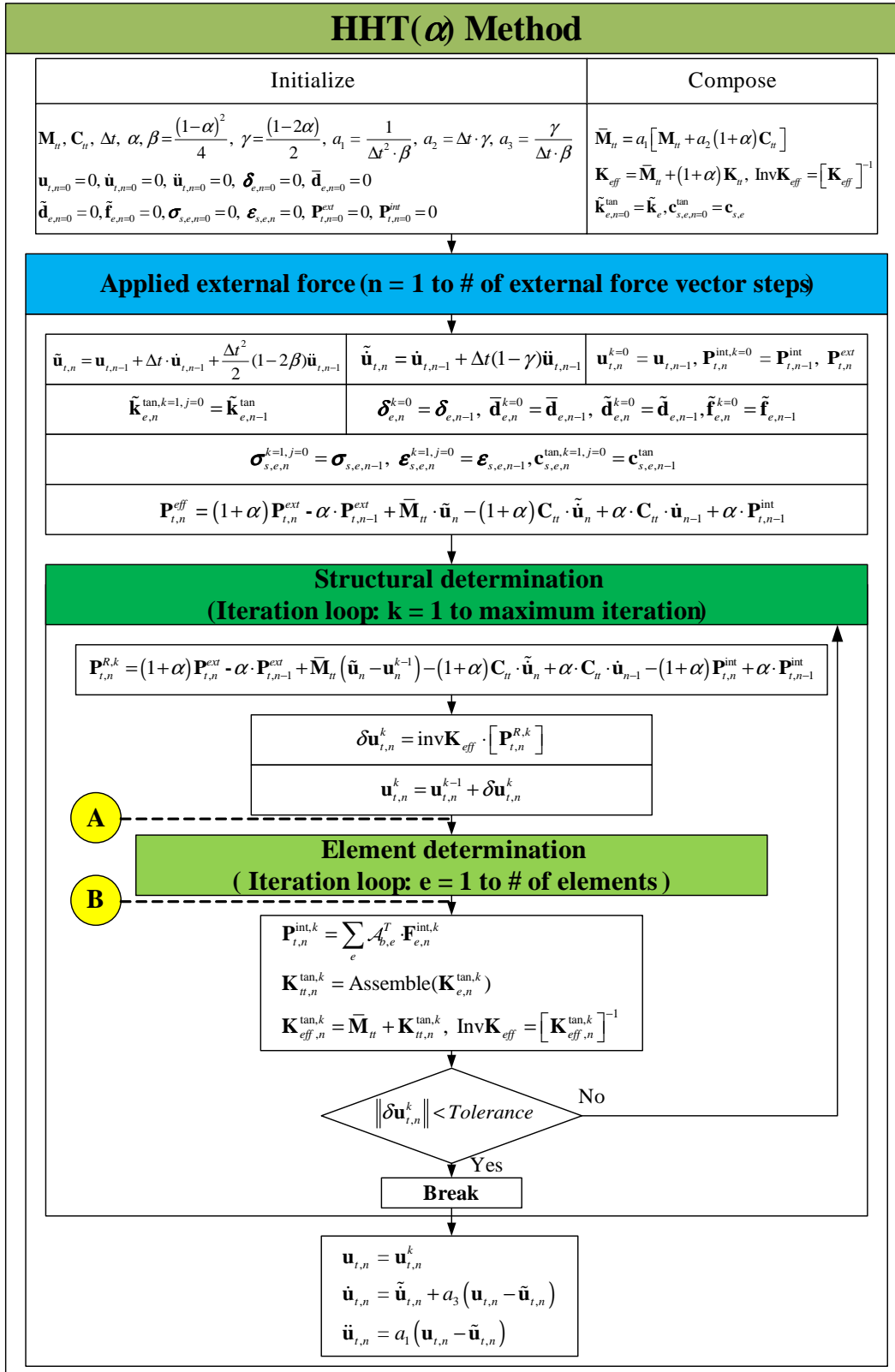


Figure 5.5: Flow chart of transient analysis using the HHT implicit method

or with $\omega^2 = \lambda$

$$|\mathbf{K} - \lambda\mathbf{M}| = 0$$

(5.55)

which is the *characteristic equation*, and λ is called the *eigenvalue* of the equation, and the structure is said to respond in the *mode* corresponding to a particular frequency.

14 For computational purposes, if we premultiply each side of the preceding equation by \mathbf{M}^{-1} , then

$$(\mathbf{M}^{-1}\mathbf{K} - \lambda\mathbf{I})\phi = 0$$

(5.56)

It should be noted that a zero eigenvalue is obtained for each possible rigid body motion of a structure that is not completely supported.

15 Depending on which mass matrix is adopted, slightly different results are obtained. In general, lumped mass matrices approach the exact value (consistent mass matrix) from below.

16 The mode shapes ϕ are “shapes”, and give a relative magnitude of the DOF, not the absolute values (since they are the solution to a set of homogeneous equations).

17 The natural frequencies and mode shapes provide a fundamental description of the vibrating structure.

Chapter 6

Notation

\mathbf{P}_S^{int}	:	Internal nodal force vector
\mathbf{P}_t^{ext}	:	External nodal force vector at free degrees of freedom at structural level
\mathbf{P}_t^{int}	:	Internal nodal force vector at free degrees of freedom at structural level
\mathbf{P}_t^{int}	:	Internal nodal force vector at constraint degrees of freedom at structural level
\mathbf{P}_t^R	:	Residual nodal force vector at free degrees of freedom at structural level
\mathbf{u}_t	:	Nodal displacement vector at free degrees of freedom at structural level
\mathbf{F}_e	:	Element nodal force vector in global reference
\mathbf{F}_e	=	$[\overline{N}_{X1}, \overline{V}_{Y1}, \overline{M}_{Z1}, \overline{N}_{X2}, \overline{V}_{Y2}, \overline{M}_{Z2}]^T$
\mathbf{F}_e^{int}	:	Internal element nodal force vector in global reference
$\tilde{\mathbf{d}}_e$:	Element nodal displacement vector in global reference
$\tilde{\mathbf{d}}_e$	=	$[\overline{u}_{X1}, \overline{v}_{Y1}, \overline{\theta}_{Z1}, \overline{u}_{X2}, \overline{v}_{Y2}, \overline{\theta}_{Z2}]^T$
$\tilde{\mathbf{f}}_e$:	Element nodal force vector in local reference with rigid body modes
$\tilde{\mathbf{f}}_e$	=	$[\overline{N}_{x1}, \overline{V}_{y1}, \overline{M}_{z1}, \overline{N}_{x2}, \overline{V}_{y2}, \overline{M}_{z2}]^T$
$\tilde{\mathbf{f}}_e^{int}$:	Internal element nodal force vector in local reference with rigid body modes
$\tilde{\mathbf{d}}_e$:	Element nodal displacement vector in local reference with rigid body modes
$\tilde{\mathbf{d}}_e$	=	$[\overline{u}_{x1}, \overline{v}_{y1}, \overline{\theta}_{z1}, \overline{u}_{x2}, \overline{v}_{y2}, \overline{\theta}_{z2}]^T$
$\tilde{\mathbf{f}}_e$:	Element nodal force vector in local reference without rigid body modes
$\tilde{\mathbf{f}}_e$	=	$[\tilde{M}_{z1}, \tilde{M}_{z2}, \tilde{N}_{x2}]^T$
$\tilde{\mathbf{f}}_e^{int}$:	Internal element nodal force vector in local reference without rigid body modes
$\tilde{\mathbf{f}}_e^R$:	Residual element nodal force vector in local reference without rigid body modes
$\tilde{\mathbf{d}}_e$:	Element nodal displacement vector in local reference without rigid body modes
$\tilde{\mathbf{d}}_e$	=	$[\tilde{\theta}_{z1}, \tilde{\theta}_{z2}, \tilde{u}_{x2}]^T$
$\tilde{\mathbf{d}}_e^R$:	Residual element nodal displacement vector in local reference without rigid body modes
$\mathbf{d}_s(x)$:	Section displacement vector
$\mathbf{d}_s(x)$	=	$[u(x), v(x)]^T$
$\boldsymbol{\sigma}_s(x)$:	Section force vector
κ	:	Plastic stress

$\boldsymbol{\sigma}_s(x)$	=	$[N(x), M(x)]^T$
$\boldsymbol{\sigma}_s^{int}(x)$:	Internal section force vector
$\boldsymbol{\sigma}_s^R(x)$:	Residual section force vector
$\boldsymbol{\varepsilon}_s(x)$:	Section deformation vector
$\boldsymbol{\varepsilon}_s(x)$	=	$[\varepsilon_x(x), \phi_z(x)]^T$
$\boldsymbol{\varepsilon}_s^{int}(x)$:	Residual section deformation vector
σ	:	Uniaxial stress
ε	:	Uniaxial strain
σ_r	:	Uniaxial stress of layer/fiber
ε_r	:	Uniaxial strain of layer/fiber
$\mathbf{N}_d(x)$:	Shape function on displacement field
$\mathbf{B}_d(x)$:	The matrix derived from the derivatives of $\mathbf{N}_d(x)$
$\mathbf{N}_f(x)$:	Shape function on force field
\mathbf{K}_S	:	Augmented stiffness matrix at structural level
\mathbf{K}_{nt}	:	Stiffness matrix associated natural boundary conditions
\mathbf{K}_{tu}	:	Stiffness matrix associated natural and essential boundary conditions
\mathbf{K}_{ut}	:	Stiffness matrix associated essential and natural boundary conditions
\mathbf{K}_{uu}	:	Stiffness matrix associated essential boundary conditions
\mathbf{K}_e	:	Element stiffness matrix in global reference
$\bar{\mathbf{k}}_e$:	Element stiffness matrix in local reference with rigid body modes
$\bar{\mathbf{k}}_e^{tan}$:	Element tangent stiffness matrix in local reference with rigid body modes
$\tilde{\mathbf{k}}_e$:	Element stiffness matrix in local reference without rigid body modes
$\tilde{\mathbf{c}}_e$:	Element flexibility matrix in local reference without rigid body modes
$\mathbf{k}_s(x)$:	Section stiffness matrix
$\mathbf{k}_s^{tan}(x)$:	Section tangent stiffness matrix
$\mathbf{c}_s(x)$:	Section flexibility matrix
$E(x)$:	Elastic modulus
$A(x)$:	Section area
$I_z(x)$:	Moment of inertia on section area
$\delta\bar{\mathbf{d}}_e$:	Virtual element nodal displacement vector in local reference
$\delta\boldsymbol{\varepsilon}_s(x)$:	Virtual section deformation vector
L_e	:	Element length
$\boldsymbol{\Gamma}_e$:	Transformation matrix between local and global coordinate system
$\bar{\boldsymbol{\Gamma}}_e$:	Transformation matrix between rigid body modes and no rigid body modes

Subscript

t : Known traction
u : Known displacement
S : Structural level
e : Element level or e^{th} element at element state determination
r : Layer/fiber level or r^{th} layer/fiber at layer/fiber state determination
s : Section level or s^{th} section at section state determination
d : Displacement field
f : Force field
n : Current step of External force/displacement vector

Superscript

int : Internal
ext : External
R : Residual
k : k^{th} iteration at structural level
j : j^{th} iteration at element level

NOTATION; Old Matrix Lecture Notes

a	Vector of coefficients in assumed displacement field
<i>A</i>	Area
A	Kinematics Matrix
b	Body force vector
B	Statics Matrix, relating external nodal forces to internal forces
[B']	Statics Matrix relating nodal load to internal forces $\mathbf{p} = [\mathbf{B}']\mathbf{P}$
[B]	Matrix relating assumed displacement fields parameters to joint displacements
<i>C</i>	Cosine
[C1 C2]	Matrices derived from the statics matrix
{d}	Element flexibility matrix (lc)
{d_c}	
[D]	Structure flexibility matrix (GC)
<i>E</i>	Elastic Modulus
[E]	Matrix of elastic constants (Constitutive Matrix)
{F}	Unknown element forces and unknown support reactions
{F₀}	Nonredundant element forces (lc)
{F_x}	Redundant element forces (lc)
{F_e}	Element forces (lc)
{F⁰}	Nodal initial forces
{F^e}	Nodal energy equivalent forces
{F}	Externally applied nodal forces
FEA	Fixed end actions of a restrained member
<i>G</i>	Shear modulus
<i>I</i>	Moment of inertia
[L]	Matrix relating the assumed displacement field parameters to joint displacements
[I]	Identity matrix
[ID]	Matrix relating nodal dof to structure dof
<i>J</i>	St Venant's torsional constant
[k]	Element stiffness matrix (lc)
[p]	Matrix of coefficients of a polynomial series
[k_g]	Geometric element stiffness matrix (lc)
[k_r]	Rotational stiffness matrix ([d] inverse)
[K]	Structure stiffness matrix (GC)
[K_g]	Structure's geometric stiffness matrix (GC)
<i>L</i>	Length
L	Linear differential operator relating displacement to strains
<i>l_{ij}</i>	Direction cosine of rotated axis i with respect to original axis j
{LM}	structure dof of nodes connected to a given element
{N}	Shape functions
{p}	Element nodal forces = F (lc)
{P}	Structure nodal forces (GC)
<i>P, V, M, T</i>	Internal forces acting on a beam column (axial, shear, moment, torsion)
R	Structure reactions (GC)
<i>S</i>	Sine
t	Traction vector
t̂	Specified tractions along Γ_t
u	Displacement vector
ũ	Neighbour function to $u(x)$
ũ(x)	Specified displacements along Γ_u
<i>u, v, w</i>	Translational displacements along the x, y, and z directions
<i>U</i>	Strain energy
<i>U*</i>	Complementary strain energy
<i>x, y</i>	local coordinate system (lc)
<i>X, Y</i>	Global coordinate system (GC)
<i>W</i>	Work
α	Coefficient of thermal expansion
[Γ]	Transformation matrix
{δ}	Element nodal displacements (lc)
{Δ}	Nodal displacements in a continuous system
{Δ}	Structure nodal displacements (GC)
ϵ	Strain vector
ϵ_0	Initial strain vector
{Υ}	Element relative displacement (lc)
{Υ₀}	Nonredundant element relative displacement (lc)
{Υ_x}	Redundant element relative displacement (lc)
θ	rotational displacement with respect to z direction (for 2D structures)
δ	Variational operator
δM	Virtual moment
δP	Virtual force
$\delta\theta$	Virtual rotation
δu	Virtual displacement
$\delta\phi$	Virtual curvature
δU	Virtual internal strain energy
δW	Virtual external work
$\delta\epsilon$	Virtual strain vector
$\delta\sigma$	Virtual stress vector
Γ	Surface

Γ_t	Surface subjected to surface tractions
Γ_u	Surface associated with known displacements
$\boldsymbol{\sigma}$	Stress vector
$\boldsymbol{\sigma}_0$	Initial stress vector
Ω	Volume of body

lc: Local Coordinate system
GC: Global Coordinate System

