Software Patterns for Nonlinear Beam-Column Models

Michael H. Scott[1]; Gregory L. Fenves[2]; Frank McKenna[3]; and Filip C. Filippou[4]

Abstract: A framework for simulating the material and geometric nonlinear response of frame members is developed from the equations of beam mechanics. The implementation of a beam-column finite element is reduced to the state determination procedure for a basic system that displaces and rotates with the element. An abstraction for geometric nonlinearity represents the kinematic and equilibrium transformations between the basic and global reference systems, while an abstraction for force-deformation response represents material nonlinearity for the basic system. Separate objects encapsulate material stress-strain behavior and cross-sectional integration in order to increase the modeling flexibility for computing the response of fiber-discretized cross sections. Multiple forms of distributed plasticity in beam-column elements are incorporated in the framework through objects that encapsulate one-dimensional quadrature rules. Software design patterns are utilized to create complex beam-column simulation models by composition of basic building blocks. The modeling flexibility of the software design is demonstrated through the simulation of a reinforced concrete column.

DOI: 10.1061/(ASCE)0733-9445(2008)134:4(562)

CE Database subject headings: Beam columns; Displacement; Nonlinear analysis; Simulation models; Computer software.

Introduction

A vast number of simulation models for the material nonlinear behavior of beam-column members have been developed, ranging from concentrated plasticity formulations that confine nonlinear deformations to the ends of an elastic element (Clough et al. 1965; Giberson 1967; Hilmy and Abel 1985; Powell and Chen 1986; Ziemian and McGuire 2002), to distributed plasticity formulations based on finite-element methods (Ciampi and Carlesimo 1986; Spacone et al. 1996; Hjelmstad and Taciroglu 2005; Alemdar and White 2005). Separate from the developments for material nonlinearity, simulation models for the geometric nonlinear response of beam-column members have been developed (Crisfield 1991; Neuenhofer and Filippou 1998; De Souza 2000; Sivaselvan and Reinhorn 2002; Zhou and Chan 2004). Flexible and reusable software designs are required to implement these and other simulation models under a single framework.

Typically, software implementations of beam-column simulation models implement the data and algorithms for representing all sources of nonlinear behavior in a single subroutine or procedure. Although this approach may provide computational efficiency, it can lead to unnecessary code duplication when new simulation models are developed, making software maintenance difficult. It is also difficult to extend and reuse a code if an element procedure operates on global data of a fixed size because this places limits on the amount of element data and history variables. The object-oriented concepts of data encapsulation and polymorphism, however, provide for a more modular and reliable software design, where an element implementation controls its data and can reuse an existing code in order to acquire new functionality with a minor computational penalty (Rucki and Miller 1996).

An object encapsulates data with algorithms that operate on the data when the object receives a request to do so. An object's type is defined by its interface [sometimes referred to as an application programming interface (API)], which is the collection of all operations the object can perform. Separate objects can be of the same type, sharing a common interface while providing distinct definitions for the methods in the interface. The specification of an object's internal data and methods is defined by its class. An abstract class, or abstraction, defines a common interface for its subclasses and defers the implementation of its abstract methods to these subclasses. The benefits of engineering software development from an object-oriented approach were outlined by Fenves (1990). Since that time, several object-oriented designs for structural analysis have been proposed (Forde et al. 1990; Mackie 1992; Zimmermann et al. 1992; Rucki and Miller 1996; Archer et al. 1999).

Recent advances in object-oriented structural analysis have focused on software design patterns (Gamma et al. 1995), which are generic descriptions of communicating classes that make a system more flexible and reusable by emphasizing object composition for software extensibility. The object-oriented software framework OpenSees makes extensive use of design patterns to represent the fundamental relationships of finite-element solution algorithms and constitutive models (McKenna 1997; Scott 2004). Software patterns also appear in the development of advanced computational applications in OpenSees, such as parallel computing

[1]Assistant Professor, School of Civil and Construction Engineering, Oregon State Univ., Corvallis, OR 97331 (corresponding author). E-mail: michael.scott@oregonstate.edu

[2]Professor, Dept. of Civil and Environmental Engineering, Univ. of California, Berkeley, CA 94720. E-mail: fenves@berkeley.edu

[3]Research Engineer, Dept. of Civil and Environmental Engineering, Univ. of California, Berkeley, CA 94720. E-mail: fmckenna@ce.berkeley.edu

[4]Professor, Dept. of Civil and Environmental Engineering, Univ. of California, Berkeley, CA 94720. E-mail: filippou@ce.berkeley.edu

Note. Associate Editor: Finley A. Charney. Discussion open until September 1, 2008. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on July 5, 2006; approved on April 30, 2007. This paper is part of the ***Journal of Structural Engineering***, Vol. 134, No. 4, April 1, 2008. ©ASCE, ISSN 0733-9445/2008/4-562–571/$25.00.

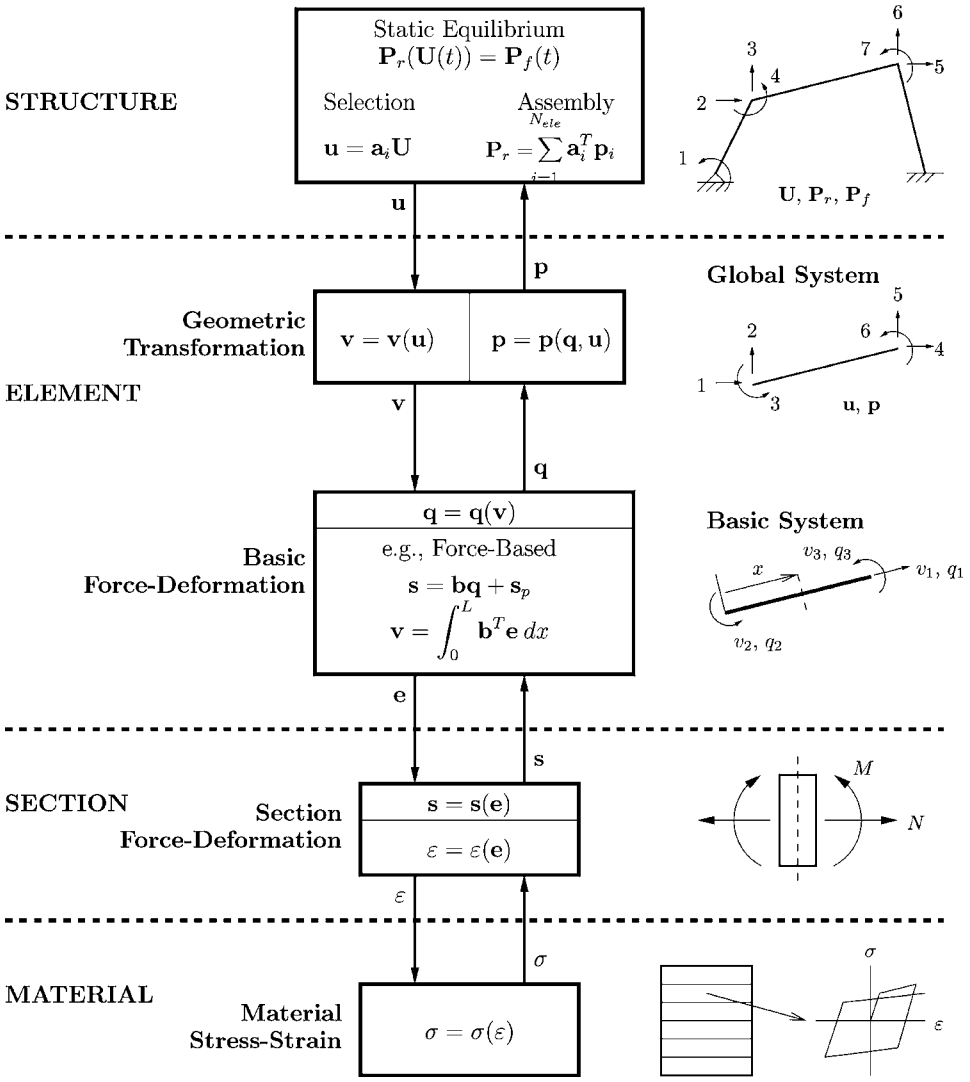

Fig. 1. Modeling hierarchy for nonlinear structural analysis

(McKenna and Fenves 2000) and distributed hybrid simulation (Takahashi and Fenves 2005).

The objective of this paper is to identify, from the governing equations of beam mechanics, fundamental software abstractions that represent the nonlinear response of frame members. Abstract classes for global and basic reference systems, section constitutive models, and numerical integration are developed. With this approach, the majority of beam-column simulation models can be implemented by software design patterns in a framework that is flexible and extensible. The implementation of a force-based beam-column model in OpenSees and its application to simulating the cyclic response of a reinforced concrete column demonstrate the modeling flexibility afforded by the software design.

Abstractions of Nonlinear Structural Analysis

A common abstraction in an object-oriented structural analysis is a Domain, which is an aggregation of node, boundary condition, load, and element objects (Forde et al. 1990; McKenna 1997). Analysis objects encapsulate algorithms that direct the Domain components to form and solve the governing equations of structural equilibrium for the nodal displacement vector, $\mathbf{U}(t)$, at each simulation time step. The case of structural equilibrium is shown at the top level of the modeling hierarchy in Fig. 1, where the time-varying external load vector, $\mathbf{P}_f(t)$, must be in equilibrium with the resisting force vector, $\mathbf{P}_r$. The focus of this paper is the software design of a single beam-column element in the structural system, for which the selection of element displacements, **u**, and assembly of resisting forces, **p**, are accomplished by standard finite-element procedures, which are represented symbolically in Fig. 1 by a boolean selection matrix, $\mathbf{a}_i$.

In the following presentation, it is assumed that the beam-column formulations take place in a basic system or corotating frame of reference that is free of rigid body displacement modes. This approach to the element formulation separates material nonlinearity inside the basic system from geometric nonlinearity of the corotating frame, and is crucial in developing a flexible software design for beam-column simulation models (Filippou and Fenves 2004). Furthermore, without loss of generality in the software design, two-dimensional elements are considered and small strain beam mechanics are assumed. Extensions of the design to three-dimensional elements and finite strain are straightforward.

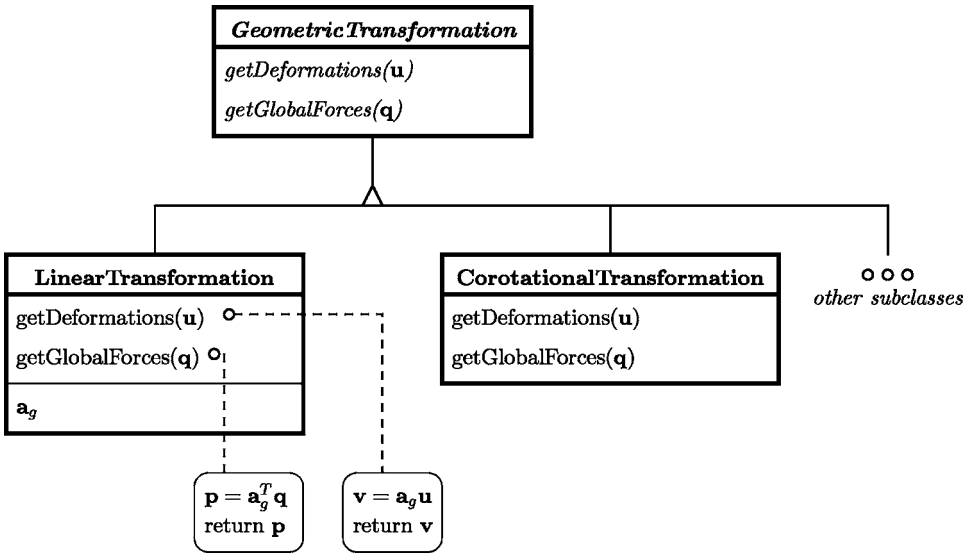

Fig. 2. *GeometricTransformation* class diagram, where subclasses implement transformation between global and basic systems of beam-column element

Transformation between Global Coordinates and Basic System

The extraction of the rigid body modes from the element displacement vector to give the element deformations, **v**, in the basic system is a nonlinear function, $\mathbf{v}=\mathbf{v}(\mathbf{u})$. For the simply supported basic system shown in Fig. 1, there are three element deformations: the axial extension, v_1, and the end rotations, v_2 and v_3, relative to the element chord. The axial force, q_1, and the end moments, q_2 and q_3, are the corresponding basic forces, which are computed from the element deformations in accordance with the governing equations of equilibrium, compatibility, and constitution in the basic system. Rigid body equilibrium of the basic forces in the displaced configuration gives the element forces in the global system.

An abstract class, *GeometricTransformation*, is defined to provide multiple definitions of the displacement and force transformations between global and basic systems. With this approach, an element can compute basic forces without regard for the kinematic and equilibrium transformations outside the basic system. The class diagram for geometric transformations is shown in Fig. 2 where subclasses of *GeometricTransformation* implement specific kinematic and equilibrium assumptions in the methods *getDeformations*() and *getGlobalForces*(). For the case where the corotating frame remains coincident with the original element configuration, the LinearTransformation subclass encapsulates the element length and direction cosines in order to populate the matrix $\mathbf{a}_g$ and enforce a linear relationship between global and basic vectors. Other subclasses encapsulate nonlinear kinematic and equilibrium relationships that account for large displacement of the element, such as the corotational theory (Crisfeld 1991; De Souza 2000).

Section Force-Deformation Relationship

For the distributed plasticity elements considered in this paper, the material nonlinear response is obtained by integration of the force-deformation response in the corotating frame of reference. As shown in Fig. 1, at any location x along the element, there are internal section forces, or stress resultants, **s**, and their corresponding section deformations, **e**. The constitutive relationship between the forces and deformations at each section takes the form $\mathbf{s}=\mathbf{s}(\mathbf{e})$.

To provide an interface for the multitude of approaches to computing section stress resultants, the abstract *ForceDeformation* class is defined. The important operations for subclasses of *ForceDeformation* are the computation of section stress resultants and the commit of section history variables upon convergence of the global solution for equilibrium. With this abstraction, an element can evaluate its equations of equilibrium and compatibility inside the basic system without consideration of how the constitutive response is computed at each section. In addition to closed-form stress resultant plasticity (El-Tawil and Deierlein 1998), subclasses of *ForceDeformation* can implement a fiber section model where stress resultant plasticity is computed by numerical integration of material plasticity over the section area

$$\mathbf{s} = \sum_{i=1}^{N_f} \mathbf{a}_{si}^T \sigma_i A_i \tag{1}$$

where σ_i, A_i, and $\mathbf{a}_{si}$=stress, area, and compatibility matrix, respectively, for the ith fiber in the section discretization. Without loss of generality in the software design, the following presentation assumes Euler–Bernoulli beam kinematics where the compatibility matrix is a function of the fiber location $\mathbf{a}_{si}=[1-y_i]$ and there is a uniaxial state of fiber stress. It is straightforward to include shear stress-strain response in Eq. (1) for a Timoshenko beam (Park and Lee 1996; Saritas 2006).

Two abstract classes are defined for the software representation of Eq. (1). The first class, *SectionIntegration*, encapsulates section dimensions, reinforcing details, and numbers of fibers from which the section is discretized to give the size, A, and location, (y, z), of each fiber, as shown in Fig. 3. The second class, *UniaxialMaterial*, is responsible for the path-dependent stress-strain response of each fiber in the cross section and has operations to return the material stress and to commit the material state upon global convergence. The FiberSection subclass of *ForceDeformation* is composed of one *SectionIntegration* and many *UniaxialMaterials*, as shown in Fig. 4 using UML class

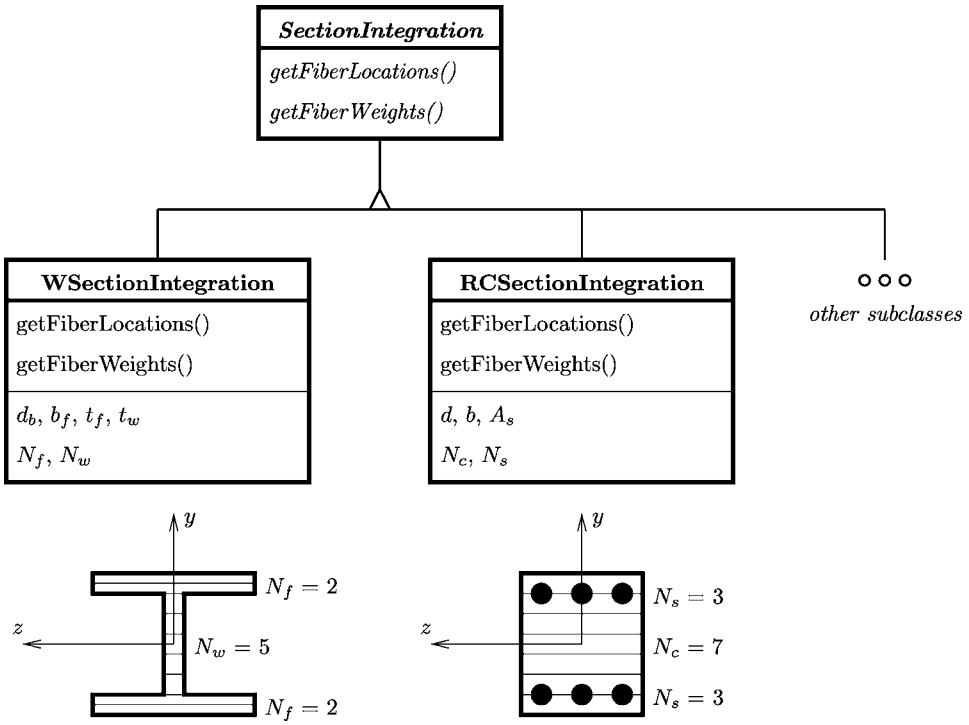

Fig. 3. *SectionIntegration* class, which provides fiber locations and weights to calling object

diagram notation (Booch et al. 1998), where the material stress and section discretization are interchangeable components of the FiberSection computation. In a similar fashion, the software design can incorporate biaxial and triaxial material models to account for the interaction of axial and shear stresses in a Timoshenko beam implementation.

Software Design Patterns in Section Constitutive Models

The interchangeable components of the FiberSection class described in the previous section reflect the modeling flexibility afforded by the Strategy software design pattern (Gamma et al. 1995). The Strategy pattern appears repeatedly in the equations of structural mechanics and in finite-element solution procedures, e.g., interchangeable root-finding algorithms to solve the equations of equilibrium (McKenna 1997). Other patterns identified by Gamma (1995) can increase the flexibility of section constitutive models, examples of which follow:

1. The Composite pattern groups primitive objects into a composite object that calling objects treat uniformly, such as the parallel and series configurations of material stress-strain relationships shown in Fig. 5. This pattern allows arbitrarily complex uniaxial models to be created, e.g., for multilinear

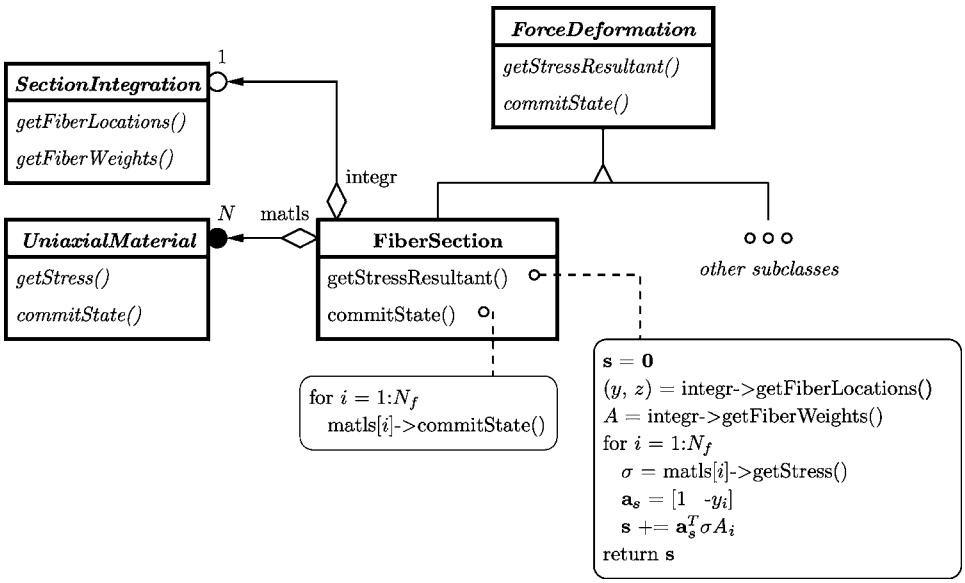

Fig. 4. FiberSection class diagram, where section stress resultants are obtained from *UniaxialMaterial* response at locations dictated by instance of *SectionIntegration*

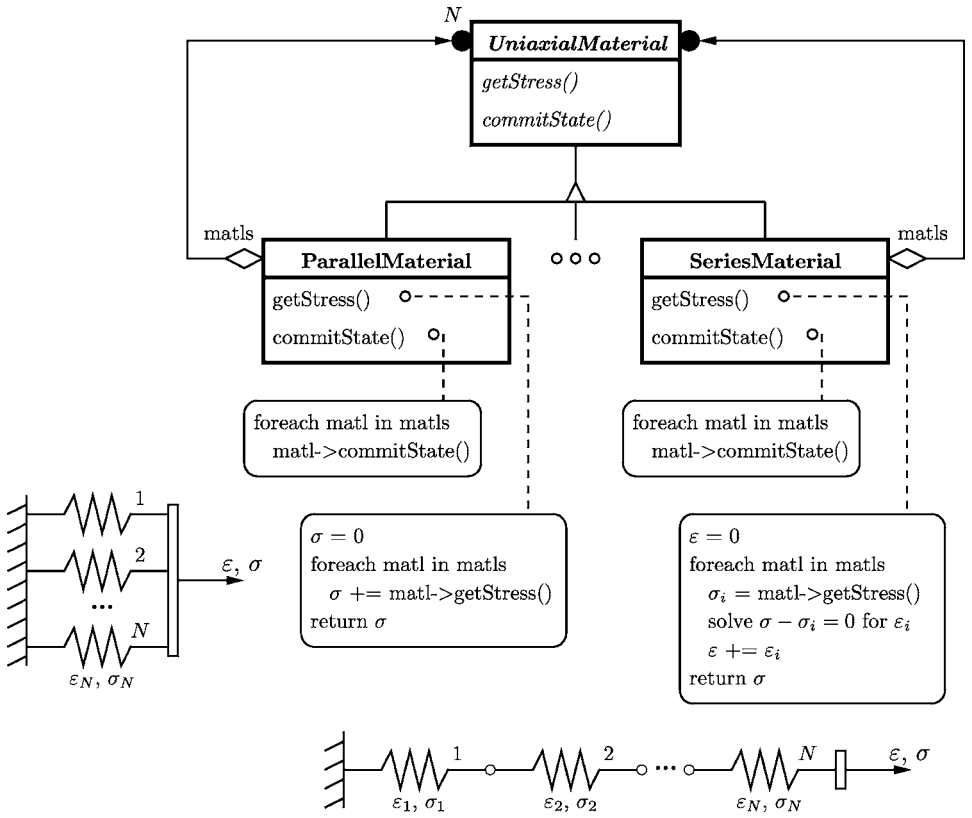

Fig. 5. ParallelMaterial and SeriesMaterial class diagram, where the response is obtained by recursive composition of base *UniaxialMaterial* class

and gapping constitutive laws. The Composite concept is readily extended to multiaxial constitutive models of stress-strain and force-deformation behavior;

2. To incorporate simulation models written in a different programming language or software framework, the Adapter pattern encapsulates the necessary operations and data structures to convert the model to the interface that calling objects expect. The conversion of a *ForceDeformation* model written in a procedural language such as FORTRAN is shown in Fig. 6, where the wrapper class fills the necessary common blocks prior to invoking the state determination subroutine and updates history variables in the common block upon a commit; and
3. The Decorator pattern attaches additional attributes to an object to extend functionality without reimplementation. An example of the Decorator pattern is the SectionAggregator class, which combines several *UniaxialMaterials* with an existing subclass of *ForceDeformation* to increase the number of stress resultants in the section force-deformation response. The shear force aggregation shown in Fig. 7 promotes reuse of flexural section models in a Timoshenko beam implementation (Taylor et al. 2003) and in force-based beam implementations that account for shear in the section response (Ranzo and Petrangeli 1998).

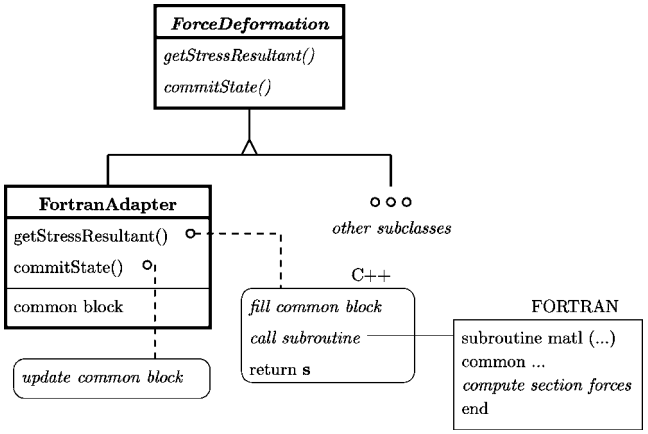

Fig. 6. Example of Adapter software pattern to implement *ForceDeformation* interface by calling FORTRAN subroutine to determine constitutive response

Additional software design patterns for nonlinear constitutive models are described by Scott (2004), including the implementation of uniaxial hysteretic models by interchangeable backbone and strength and stiffness degradation objects.

Software Implementation of Force-Based Beam-Column Element

The software building blocks presented in this paper for geometric and material nonlinear response are brought together in this section to implement a force-based beam-column element with small deformations in the corotating system (De Souza 2000). After the element formulation is summarized, its implementation in the OpenSees software framework (McKenna et al. 2000) is presented.

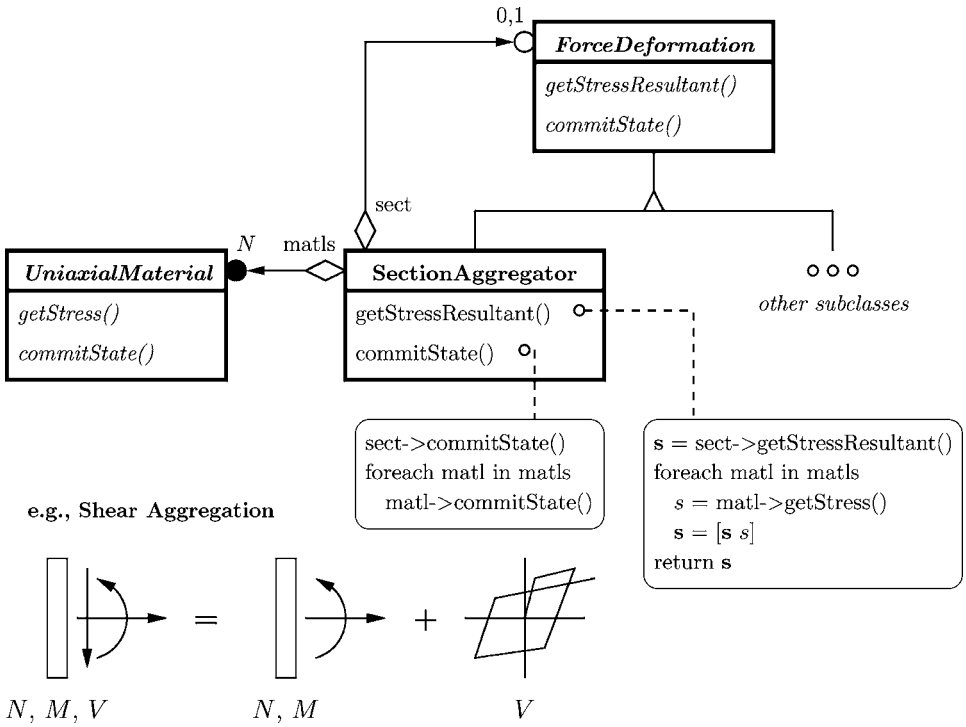

Fig. 7. Class diagram for SectionAggregator, which combines uniaxial force-deformation relationships with existing *ForceDeformation* object

Element Formulation

In the force-based formulation (Ciampi and Carlesimo 1986; Spacone et al. 1996) there is a strong form of equilibrium between basic forces and section forces at any location of the element

$$\mathbf{s}(x) = \mathbf{b}(x)\mathbf{q} + \mathbf{s}_p(x) \tag{2}$$

where the interpolation matrix $\mathbf{b}$ represents the homogeneous solution to static equilibrium of basic forces; and $\mathbf{s}_p$=particular solution for applied member loads. Compatibility of section and element deformations is expressed in integral form according to the principle of virtual forces

$$\mathbf{v} = \int_0^L \mathbf{b}^T(x)\mathbf{e}(x)\,dx \tag{3}$$

The advantage of force-based elements over displacement-based approaches is that they satisfy equilibrium in the nonlinear range of material response, alleviating the need for mesh refinement, keeping the number of degrees of freedom in a structural model to a minimum (Ciampi and Carlesimo 1986; Neuenhofer and Filippou 1997).

Element Implementation

The force-based element in OpenSees is implemented in the ForceBeamColumn class, which is shown in Fig. 9 as a subclass of the abstract *Element* class. The essential operations of an *Element* are to compute resisting forces and to commit the element state. Additional *Element* functionality is required in a finite-element analysis, e.g., to compute a tangent stiffness matrix; however, these operations are omitted for brevity.

The implementation of the force-based beam-column element encapsulates the non-iterative state determination algorithm described by Neuenhofer and Filippou (1997) where residual errors in the element compatibility relationship are propagated to the global system of equations. In accordance with this state determination algorithm, the ForceBeamColumn class uses numerical integration to evaluate the element compatibility relationship of Eq. (3)

$$\mathbf{v} = \sum_{i=1}^{N_p} \mathbf{b}^T(x_i)\mathbf{e}(x_i)w_i \tag{4}$$

where N_p=number sections, each with location x_i and corresponding weight, w_i.

The standard approach to evaluate Eq. (4) is Gauss–Lobatto quadrature, which allows plastic hinges to form at any section along the element but leads to a loss of objectivity for strain-softening hinges (Coleman and Spacone 2001). Element integration methods that maintain objectivity under softening response have been proposed by Scott and Fenves (2006) and Addessi and Ciambi (2007). Recognizing there are several strategies to numerical integration in force-based elements, the abstract class *BeamIntegration* is defined. As shown in Fig. 8, subclasses encapsulate the locations and weights of the sections where the equilibrium and compatibility relationships are evaluated. *BeamIntegration* mitigates code duplication since one implementation of the force-based state determination procedure can use many integration approaches to describe the spread of plasticity.

The ForceBeamColumn class diagram in Fig. 9 shows there are three strategies defined for the element. *GeometricTransformation* defines the transformation of element vectors between global and basic systems while *BeamIntegration* enforces a strategy for the spread of plasticity along the element, which is described by an instance of *ForceDeformation* at each integration point. This approach to the software design is not limited to the force-based formulation, since it is possible to use these abstractions for the implementation of displacement-based and mixed formulations inside the basic system.

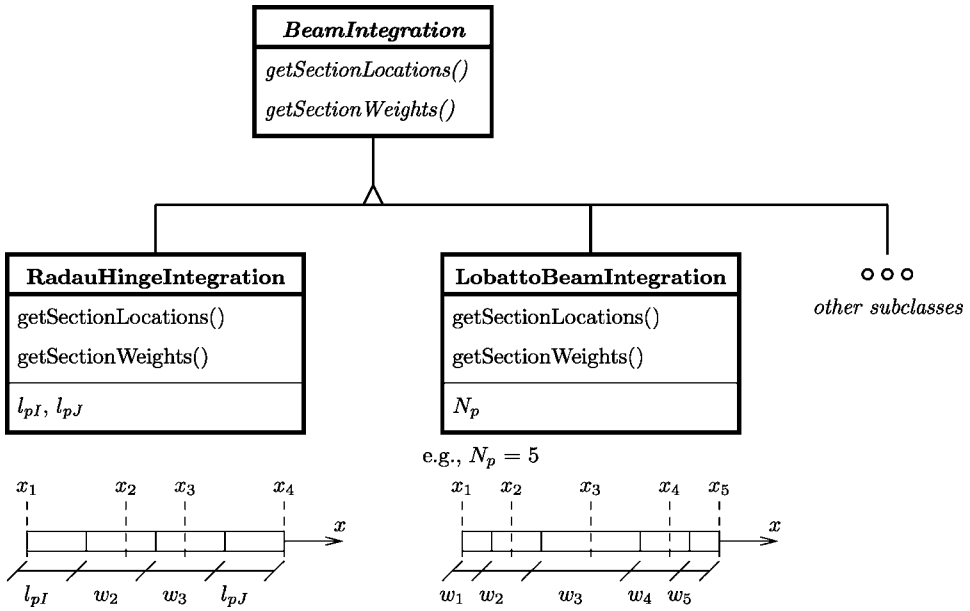

Fig. 8. *BeamIntegration* class that encapsulates locations and weights of integration points along beam-column element

Application to Reinforced Concrete Column Modeling

To demonstrate the modeling flexibility enabled by the software design, two approaches to simulating the cyclic load-displacement response of a reinforced concrete bridge pier, Specimen 7 from the tests of Tanaka and Park (1990), are examined. The specimen geometry, reinforcing details, material properties, gravity load, and lateral load-displacement history are available for download from the PEER Structural Performance Database (Berry and Eberhard 2003).

The first approach to simulating the bridge pier response is to use a single force-based element with four Gauss–Lobatto points, small displacements, and a fiber discretization of the reinforced concrete section. An object diagram showing the classes instantiated for this simulation is shown in Fig. 10, where a Menegotto–Pinto steel model (Menegotto and Pinto 1973) and Kent–Park concrete model (Kent and Park 1971) represent the uniaxial stress-strain relationships of the fibers. The lateral displacement history of the experiment drives the simulation and the computed lateral load-displacement response is compared to the experiment in Fig. 11(a). After forming a plastic hinge, deformations localize at the base of the bridge pier over a length corresponding to the Gauss–Lobatto weight at the base ($L/12=138$ mm). This localization causes the concrete to crush and the load to be resisted by only the reinforcing steel, resulting in a significant difference between the computed results and experimental data for increasing cycles of displacement.

To remedy the poor representation of the softening hinge as a result of Gauss–Lobatto integration, the second approach replaces

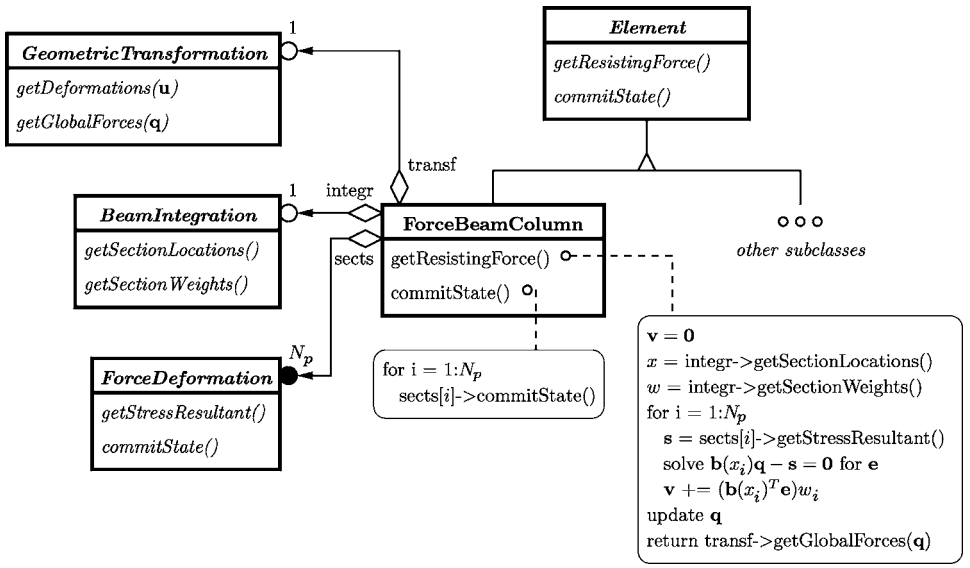

Fig. 9. Class diagram for ForceBeamColumn implementation, where element behavior is composition of *GeometricTransformation*, *BeamIntegration*, and *ForceDeformation* classes

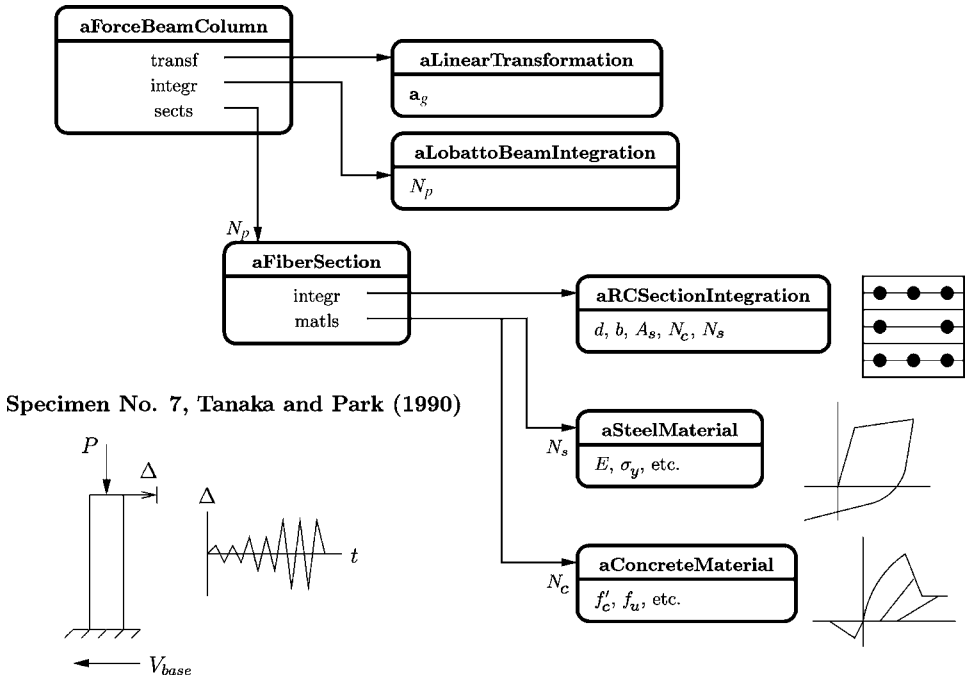

Fig. 10. Diagram showing objects instantiated in order to simulate response of reinforced concrete bridge pier

the LobattoBeamIntegration object in Fig. 10 with an instance of the RadauHingeIntegration class, which encapsulates the Gauss–Radau plastic hinge integration method of Scott and Fenves (2006). The plastic hinge length at the base of the pier is set to 356 mm according to the empirical equation, $l_p=0.08L+0.022f_y d_b$, derived by Paulay and Priestley (1992). A significant improvement in the computed response is shown in Fig. 11(b) since the localized flexural deformations at the base of the bridge pier are integrated over a physically significant length.

The change in numerical integration approach to represent more accurately the formation of a plastic hinge reflects the modularity of the software design. Although the effects of large displacements and shear were not significant for this simulation, it is straightforward to include in the analysis an object that encapsulates the corotational transformation as well as an object that aggregates shear force-deformation response with the fiber section. This example indicates the software design can successfully incorporate many beam-column simulation models as part of a common framework. Furthermore, the framework provides an ideal environment to develop, verify, and validate new simulation models prior to optimizing their implementation for production.

Conclusions

A modular, flexible, and reusable software framework for simulating the response of frame members has been developed from abstractions that encapsulate the material and geometric nonlinear behavior of beam-column elements. *GeometricTransformation* represents the transformation of vectors between the global system and corotating reference frame of the element; *ForceDeformation* encapsulates the computation of section stress resultants for distributed plasticity formulations; and *BeamIntegration* encapsulates numerical integration to represent the spread of plasticity in beam-column elements. Each of these abstractions represents a Strategy pattern in simulating the response of a beam-column member, as demonstrated in the software imple-

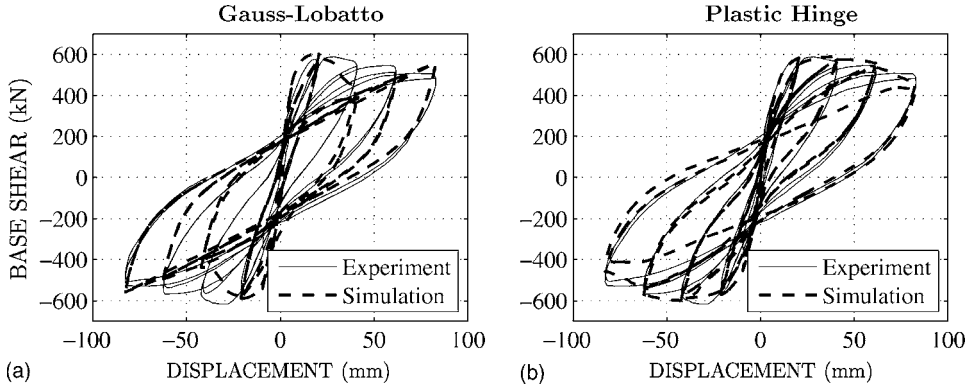

Fig. 11. Computed load-displacement history for reinforced concrete bridge pier using: (a) Gauss–Lobatto quadrature with $N_p=4$ integration points; (b) modified Gauss–Radau plastic hinge integration with $l_p=356$ mm

mentation of the force-based element formulation in the OpenSees framework. Additional design patterns (Composite, Adapter, and Decorator) were identified to build complex section constitutive models without code duplication. A future extension of the software design in OpenSees is parameterized finite-element models that support gradient-based applications with uncertain material and geometric parameters of nonlinear beam-column elements. These applications require a flexible and extensible software design for identifying and updating the parameters encapsulated by objects in the framework.

Acknowledgments

This work and the development of OpenSees have been supported by the Pacific Earthquake Engineering Research Center under Grant No. EEC-9701568 from the National Science Foundation to the University of California, Berkeley. The implementation of the geometric transformation classes in OpenSees by Dr. Remo Magalhaes de Souza is gratefully acknowledged.

References

Addessi, D., and Ciampi, V. (2007). "A regularized force-based beam element with a damage-plastic section constitutive law." *Int. J. Numer. Methods Eng.*, 70, 610–629.

Alemdar, B. N., and White, D. W. (2005). "Displacement, flexibility, and mixed beam-column finite element formulations for distributed plasticity analysis." *J. Struct. Eng.*, 131(12), 1811–1819.

Archer, G. C., Fenves, G., and Thewalt, C. (1999). "A new object-oriented finite element analysis architecture." *Comput. Struct.*, 70, 63–75.

Berry, M., and Eberhard, M. (2003). "PEER structural performance database." ⟨http://nisee.berkeley.edu/spd/⟩.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998). *The unified modeling language user guide*, Addison-Wesley, Reading, Mass.

Ciampi, V., and Carlesimo, L. (1986). "A nonlinear beam element for seismic analysis of structures." *Proc., 8th European Conf., on Earthquake Engineering*, Lisbon, Portugal.

Clough, R. W., Benuska, K. L., and Wilson, E. L. (1965). "Inelastic earthquake response of tall buildings." *Proc., 3rd World Conf. on Earthquake Engineering*, Wellington, New Zealand.

Coleman, J., and Spacone, E. (2001). "Localization issues in force-based frame elements." *J. Struct. Eng.*, 127(11), 1257–1265.

Crisfield, M. A. (1991). *Nonlinear finite element analysis of solids and structures*, Vol. 1, Wiley, New York.

De Souza, R. M. (2000). "Force-based finite element for large displacement inelastic analysis of frames." Ph.D. thesis, Univ. of California, Berkeley, Calif.

El-Tawil, S., and Deierlein, G. G. (1998). "Stress-resultant plasticity for frame structures." *J. Eng. Mech.*, 124(12), 1360–1370.

Fenves, G. L. (1990). "Object-oriented programming for engineering software development." *Eng. Comput.*, 6(1), 1–15.

Filippou, F. C., and Fenves, G. L. (2004). "Methods of analysis for earthquake-resistant structures." *Earthquake engineering: From engineering seismology to performance-based engineering*, Y. Bozorgnia and V. V. Bertero, eds., Chap. 6, CRC, Boca Raton, Fla.

Forde, B. W. R., Foschi, R. O., and Stiemer, S. F. (1990). "Object-oriented finite element analysis." *Comput. Struct.*, 34(3), 355–374.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley, Reading, Mass.

Giberson, M. F. (1967). "The response of nonlinear multistory structures subjected to earthquake excitation." Ph.D. thesis, California Institute of Technology, Pasadena, Calif.

Hilmy, S. I., and Abel, J. F. (1985). "A strain-hardening concentrated plasticity model for nonlinear dynamic analysis of steel buildings." *Proc., NUMETA85, Numerical Methods in Engineering, Theory and Applications*, Boston, 305–314.

Hjelmstad, K. D., and Taciroglu, E. (2005). "Variational basis of nonlinear flexibility methods for structural analysis of frames." *J. Eng. Mech.*, 131(11), 1157–1169.

Kent, D. C., and Park, R. (1971). "Flexural members with confined concrete." *J. Struct. Div.*, 97(7), 1969–1990.

Mackie, R. I. (1992). "Object-oriented programming of the finite element method." *Int. J. Numer. Methods Eng.*, 35(2), 425–436.

McKenna, F. (1997). "Object-oriented finite element programming: Frameworks for analysis, algorithms, and parallel computing," Ph.D. thesis, Univ. of California, Berkeley, Calif.

McKenna, F., and Fenves, G. L. (2000). "An object-oriented software design for parallel structural analysis." *Proc., ASCE Structures Congress*, ASCE, Reston, Va.

McKenna, F., Fenves, G. L., and Scott, M. H. (2000). "Open system for earthquake engineering simulation." Univ. of California, Berkeley, Calif., ⟨http://opensees.berkeley.edu⟩.

Menegotto, M., and Pinto, P. E. (1973). "Method of analysis for cyclically loaded R.C. plane frames including changes in geometry and nonelastic behaviour of elements under combined normal force and bending." *Proc., Symp. on the Resistance and Ultimate Deformability of Structures Acted on by Well Defined Repeated Loads*, International Association for Bridge and Structural Engineering, Zurich, Switzerland, 15–22.

Neuenhofer, A., and Filippou, F. C. (1997). "Evaluation of nonlinear frame finite-element models." *J. Struct. Eng.*, 123(7), 958–966.

Neuenhofer, A., and Filippou, F. C. (1998). "Geometrically nonlinear flexibility-based frame finite element." *J. Struct. Eng.*, 124(6), 704–711.

Park, M. S., and Lee, B. C. (1996). "Geometrically nonlinear and elastoplastic three-dimensional shear flexible beam element of Von-Mises-type hardening material." *Int. J. Numer. Methods Eng.*, 39, 383–408.

Paulay, T., and Priestley, M. J. N. (1992). *Seismic design of reinforced concrete and masonry buildings*, Wiley, New York.

Powell, G. H., and Chen, P. F. (1986). "3D beam-column element with generalized plastic hinges." *J. Eng. Mech.*, 112(7), 627–641.

Ranzo, G., and Petrangeli, M. (1998). "A fibre finite beam element with section shear modelling for seismic analysis of RC structures." *J. Earthquake Eng.*, 2(3), 443–473.

Rucki, M. D., and Miller, G. R. (1996). "An algorithmic framework for flexible finite element-based structural modeling." *Comput. Methods Appl. Mech. Eng.*, 136, 363–384.

Saritas, A. (2006). "Frame element with mixed formulation for shear critical steel and reinforced concrete members." Ph.D. thesis, Univ. of California, Berkeley, Calif.

Scott, M. H. (2004). "Software frameworks for the computational simulation of structural systems." Ph.D. thesis, Univ. of California, Berkeley, Calif.

Scott, M. H., and Fenves, G. L. (2006). "Plastic hinge integration methods for force-based beam-column elements." *J. Struct. Eng.*, 132(2), 244–252.

Sivaselvan, M. V., and Reinhorn, A. M. (2002). "Collapse analysis: Large inelastic deformations analysis of planar frames." *J. Struct. Eng.*, 128(12), 1575–1583.

Spacone, E., Ciampi, V., and Filippou, F. C. (1996). "Mixed formulation of nonlinear beam finite element." *Comput. Struct.*, 58(1), 71–83.

Takahashi, Y., and Fenves, G. L. (2005). "Software framework for distributed experimental-computational simulation of structural systems." *Earthquake Eng. Struct. Dyn.*, 35, 267–291.

Tanaka, H., and Park, R. (1990). "Effect of lateral confining reinforcement on the ductile behaviour of reinforced concrete columns." *Rep. No. 90-2*, Dept. of Civil Engineering, Univ. of Canterbury, Canterbury, U.K.

Taylor, R. L., Filippou, F. C., Saritas, A., and Auricchio, F. (2003). "A mixed finite element method for beam and frame problems." *Comput. Mech.*, 31, 192–203.

Zhou, Z., and Chan, S. (2004). "Elastoplastic and large deflection analysis of steel frame by one element per member. I: One hinge along member." *J. Struct. Eng.*, 130(4), 538–544.

Ziemian, R. D., and McGuire, W. (2002). "Modified tangent modulus approach: A contribution to plastic hinge analysis." *J. Struct. Eng.*, 128(10), 1301–1307.

Zimmermann, T., Dubois-Pelerin, Y., and Bomme, P. (1992). "Object-oriented finite element programming: I. governing principles." *Comput. Methods Appl. Mech. Eng.*, 98(2), 291–303.