Engineering with Computers (1999) 15: 73–89

Engineering with Computers

Nonlinear Finite Element Analysis using an Object-Oriented Philosophy – Application to Beam Elements and to the Cosserat Continuum

E.N. Lages[1], G.H. Paulino[2], I.F.M. Menezes[3] and R.R. Silva[4]

[1]Department of Structural Engineering, Universidade Federal de Alagoas (UFAL), Maceió, AL, Brazil; [2]Department of Civil & Environmental Engineering, and Graduate Group in Applied Mathematics (GGAM), University of California, Davis, CA, USA; [3]TeCGraf (Computer Graphics Technology Group), PUC-Rio, Rio de Janeiro, RJ, Brazil; [4]Department of Civil Engineering, PUC-Rio, Rio de Janeiro, RJ, Brazil

Abstract. *An Object-Oriented Programming (OOP) framework is presented for solving nonlinear structural mechanics problems by means of the Finite Element Method (FEM). Emphasis is placed on engineering applications (geometrically nonlinear beam model, and elastoplastic Cosserat continuum), and OOP is employed as an effective* ***tool****, which plays an important role in the FEM treatment of such applications. The implementation is based on computational abstractions of both mathematical and physical concepts associated to structural mechanics problems involving geometrical and material nonlinearities. The overall class organization for nonlinear mechanics modeling is discussed in detail. All the analyses rely on a generic control class where several classical and modern nonlinear solution schemes are available. Examples which explore, demonstrate and validate the main features of the overall computational system are presented and discussed.*

Keywords. Beam elements; Cosserat continuum; Finite Element Analysis (FEA); Geometric nonlinearity; Material nonlinearity; Nonlinear solution schemes; Object-Oriented Programming (OOP); Structural mechanics

1. Introduction

Object-Oriented Programming (OOP) is a relatively new philosophy of programming which aims at increasing the overall quality of computational systems. Computer codes that are written based on such a philosophy are easier to maintain and to expand, reducing probability of errors due to programming. Moreover, by means of OOP concepts, new implementations can take place with no need to restructure the pre-existing code, and thus code reuse is maximized to a large extent. To achieve this level of development, however, a thorough understanding of the whole methodology employed and an extensive effort in program organization are required. Such efforts are demonstrated in the development of this work.

Compared to classical programming, OOP requires a more intrinsic integration between theory and numerical implementation. This integration is explored here in engineering applications involving geometrically nonlinear analysis of frames discretized with beam elements, and materially nonlinear problems considering the elastoplastic Cosserat continuum. The main reason for choosing these two applications is because both models present the same Degrees Of Freedom (DOF), although in different settings (discrete versus continuum, respectively). Moreover, according to Roux [1], the equations of two-dimensional (2D) network of beam elements are a straightforward discretization of the equations of a 2D Cosserat elastic medium.

The goal for the remainder of this paper consists of developing a comprehensive presentation, and the next sections are organized as follows. First, a brief literature review is provided, and some background on OOP is given where the basic terminology is established. Next, a 'unified approach' is discussed for solution of nonlinear systems which arise in the FEM — the nonlinear solution process is accomplished by means of a control class where several solution schemes are available. Afterwards, geometrically (beam elements considering shear

Correspondence and offprint requests to: Professor G. H. Paulino, Department of Civil and Environmental Engineering, University of Illinois, 2209 Newmark Laboratory, 205 N. Mathews Ave., Urbana IL 61801-2352, U.S.A. Email: paulino@uiuc.edu

effects) and materially (elastoplastic Cosserat continuum) nonlinear problems are discussed within an OOP framework. Subsequently, conclusions are inferred and directions for future research are discussed.

2. Brief Literature Review

Preliminary ideas about OOP date back to the 1960s, with the development of the *Simula* language [2]. As pointed out by Fenves [3], this language introduced the idea of a class which can create instances that respond to procedures in a similar manner. Based upon the ideas presented in the *Simula* language, research was conducted at the Xerox Palo Alto Research Center resulting in the first substantial interactive, display-based implementation, called the *Smalltalk-80* language [4], which provides a very uniform application of the OOP paradigm [3,5]. After the establishment of the *Smalltalk* language, several other languages became available, for example, *Flavors* [6] (with the idea of multiple inheritance), *Objective C* [7] and *C++* [8] (which are extensions of the C language for objects).

The OOP technique has been widely used in various applications, such as simulation programs, graphical user interfaces, and artificial intelligence (AI), just to mention a few of them. However, according to Zimmermann et al. [9], the application of OOP to the Finite Element Method (FEM) only appeared at the end of the last decade, with the work by Rehak and Baugh [10] and Forde et al. [11]. Zimmermann and co-authors [9,12–16] have published several interesting articles describing in detail the fundamental aspects relating application of OOP techniques to implementation of the FEM. For instance, Menétrey and Zimmermann [17] have applied the concepts of OOP to the FEM for nonlinear static analysis, specifically to J_2 plasticity problems. A detailed object-oriented implementation has been presented, which includes the description of the main classes and methods for solving the nonlinear problem. Mackie [18] has described the benefits that can be attained by applying OOP to Finite Element Analysis (FEA). The concepts of OOP are presented directly in terms of the FEM. Alves Filho and Devloo [19] have discussed the basic aspects of the OOP philosophy, and its implementation in scientific computations, using the FEM as an example of such implementation. Another example of an object-oriented finite element model has been presented by Raphael and Krishnamoorthy [20]. They have also discussed the general concepts of OOP, together with a detailed description of the main classes in a finite element model. Bettig and Han [21] have presented an object-oriented framework for interactive numerical analysis in a graphical user interface environment. Besson and Foerch [22] have discussed aspects of object-oriented finite element design that become relevant as the project size increases. They have presented a detailed description of the computational implementation issues, and also a very interesting comparison between performance of the OOP implementation using C++, and an existing FORTRAN implementation, with respect to the Central Processing Unit (CPU) time, for obtaining the elastic and viscoplastic solutions of a plate problem. Recently, Jeremić and Sture [23] presented a programming tool which facilitates implementation of tensorial formulae associated with the numerical solution of nonlinear problems (e.g. elastoplastic) by means of the FEM. This brief literature review gives some idea of past work involving both OOP and FEM. However, the concepts above are not restricted to FEM, and extend to other numerical methods such as the boundary element method (BEM) [24,25].

Most of the above cited papers focus on computational aspects associated with OOP, rather than on actual engineering applications. The approach adopted in the present work consists of employing OOP as an effective **tool**, which plays an essential role in the FEM solution of structural mechanics problems.

3. OOP Terminology

The basic OOP terminology, which is adopted here, is provided below. The following terms are presented and briefly discussed: *object, class, method, inheritance, polymorphism* and *reusability*. Another interesting explanation of these concepts, including application examples, can be found in the recent paper by Olsson [26].

- **object** – is the primitive element of OOP. It consists of a self-contained entity composed of *data* and *procedures*. Their data items are referred to as *instance variables*. Data and procedures (functions) are said to be *encapsulated* into an object, as shown in Fig. 1.
- **class** – is a concept analogous to *struct* in the conventional C language. Nevertheless, unlike the notion of *struct*, classes contain functions as their members [19]. Objects are *instances* of a class. However, it is important to note that simply

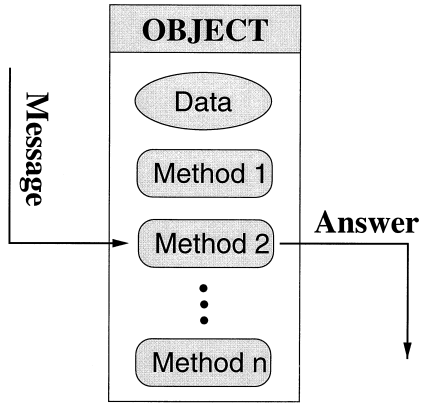

Fig. 1. Message, methods and object.

defining a class does not necessarily create any object [27]. According to Forde et al. [11], classes may also be viewed as templates which describe the organization of a given object type.

- **method** – is a procedure attached to an object. This procedure is activated when the object receives a *message*. Therefore, the objects communicate by sending messages to each other. Figure 1 also illustrates the idea of sending messages.
- **inheritance** – is a mechanism that allows a new class to be derived from an existing one. This new class, called *derived class*, inherits both data and procedures from the existing or *base class*. In addition, new data and methods may be defined in the derived class. This idea is illustrated in Fig. 2. The concept of *inheritance* allows new objects to be customized according to a given application [11].

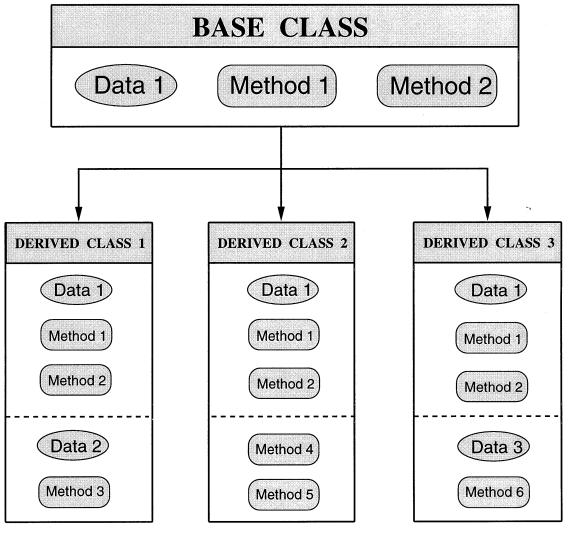

Fig. 2. Example of class hierarchy.

- **polymorphism** – is one of the most important concepts in OOP. It allows different methods, from objects of different derived classes, to be activated by the same *message*. As a consequence, new objects can be created and added to the program without having to change the existing code.
- **reusability** – is a concept analogous to the role played by the library of functions in procedural languages. In OOP, once a class is created, it can be distributed among other OOP programs. Moreover, the shared classes can be further extended by the concept of inheritance.

4. NonLinear Solver: NLS++

In this section, a supporting module for solving nonlinear finite element systems of equations is introduced [28]. It is called NLS++ (NonLinear Solver) and consists of a simple, robust, and unified object-oriented (C++) implementation of several solution algorithms, such as load control, displacement control and arc-length control. By means of a 'unified approach' for solving nonlinear finite element equations, the various solution algorithms share a common interface, just differing on their constraint equation, which is typical of each particular algorithm. Moreover, this module can be used as a computational laboratory where new solution schemes can be implemented and tested [28].

The nonlinear system of equations can be solved by means of an incremental-iterative procedure, based on the following general equation:

$$\boldsymbol{K}_j^i \, \delta \boldsymbol{u}_{j+1}^i = \delta\lambda_{j+1}^i \, \bar{\boldsymbol{p}} + \boldsymbol{r}_j^i \tag{1}$$

where superscripts and subscripts refer to step and iteration numbers, respectively, $\boldsymbol{K}_j^i$ is the tangent stiffness matrix, $\bar{\boldsymbol{p}}$ is the reference load vector, $\boldsymbol{r}_j^i$ is the unbalanced load vector, $\delta \boldsymbol{u}_{j+1}^i$ is the unknown incremental displacement vector, and $\delta\lambda_{j+1}^i$ is the unknown incremental load factor. Together with Eq. (1), each scheme possesses its own constraint equation of the form

$$\Phi(\delta \boldsymbol{u}, \delta\lambda) = 0 \tag{2}$$

resulting in a total of $n + 1$ equations for $n + 1$ unknowns (n components of the vector $\delta \boldsymbol{u}_{j+1}^i$ plus the incremental load factor $\delta\lambda_{j+1}^i$), where n is the total number of degrees of freedom.

A technique which preserves the overall efficiency of the solution (bandness and symmetry of the system matrix) consists of decomposing the displacement vector in two components such that [29]

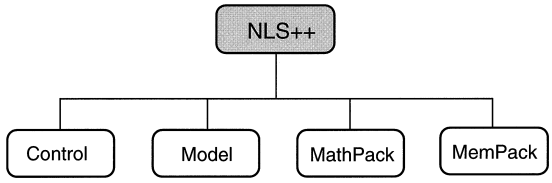

Fig. 3. NLS++ class organization.

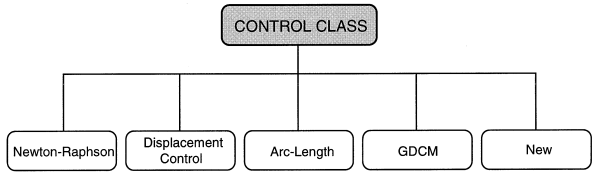

Fig. 4. *Control* Class Organization. GDCM stands for 'generalized displacement control method'.

$$\delta \boldsymbol{u}^i_{j+1} = \delta\lambda^i_{j+1}\, \delta \boldsymbol{u}_{\mathbf{I}^i_{j+1}} + \delta \boldsymbol{u}_{\mathbf{II}^i_{j+1}} \tag{3}$$

and

$$\boldsymbol{K}^i_j\, \delta \boldsymbol{u}_{\mathbf{I}^i_{j+1}} = \bar{\boldsymbol{p}}; \quad \boldsymbol{K}^i_j\, \delta \boldsymbol{u}_{\mathbf{II}^i_{j+1}} = \boldsymbol{r}^i_j \tag{4}$$

By means of Eq. (3), Eqs (1) and (4) can be shown to be equivalent [28].

Figure 3 shows the current class organization adopted by the NLS++ program. As illustrated in Fig. 4, the *Control* class is responsible for the implementation of several different solution schemes (e.g. Newton–Raphson, displacement control, arc-length, Generalized Displacement Control Method (GDCM) [28,30]) for solving nonlinear systems of the type described by Eq. (1). A comprehensive exposition of all these schemes can be found in the book by Yang and Kuo [31]. The main steps of the unified approach adopted here are shown in Pseudo-code 1. For the sake of completeness, and also to illustrate how the incremental load factor ($\delta\lambda^i_{j+1}$) is obtained (see Step 1.4 of Pseudo-code 1), the main steps of the Newton–Raphson scheme are shown in the Pseudo-code 2.

```
begin
  < 1 > foreach Iteration j of Step i:
        begin
          < 1.1 > Get Global Tangent Stiffness Matrix (K^i_j) from Model Class
          < 1.2 > Compute δu_I^i_{j+1} by means of:
                  δu_I^i_{j+1} ⟵ (K^i_j)^{-1} p̄
          < 1.3 > Compute δu_II^i_{j+1} by means of:
                  δu_II^i_{j+1} ⟵ (K^i_j)^{-1} r^i_j
          < 1.4 > Compute δλ^i_{j+1} According to the Selected Solution Scheme
          < 1.5 > Update Total Displacement Vector:
                  u^i_{j+1} ⟵ u^i_j + δλ^i_{j+1} δu_I^i_{j+1} + δu_II^i_{j+1}
          < 1.6 > Update Total Load Factor:
                  λ^i_{j+1} ⟵ λ^i_j + δλ^i_{j+1}
          < 1.7 > Get Global Internal Force Vector (f̂^i_{j+1}) from Model Class
          < 1.8 > Compute New Unbalanced Load Vector (r^i_{j+1}):
                  r^i_{j+1} ⟵ λ^i_{j+1} p̄ − f̂^i_{j+1}
          < 1.9 > Check Convergence
        end
end
```

Pseudo-code 1. Control Class – the unified approach.

```
begin
  if (j == First Iteration)
    δλ'_{j+1} ⟵ f̄ // Prescribed Load Increment for Current Step (i)
  else
    δλ'_{j+1} ⟵ 0
  endif
end
```

Pseudo-code 2. Control Class – the Newton–Raphson scheme.

The *Model* class is responsible for providing the physical meaning of vectors and matrices of Eq. (1). Its main methods are:

- computation of the reference load vector ($\bar{\boldsymbol{p}}$);
- computation of the global internal force vector ($\hat{\boldsymbol{f}}^g$), used for computing the unbalanced load vector; and
- computation of the global tangent stiffness matrix ($\boldsymbol{K}^g$).

In order to illustrate the assembly of the global tangent stiffness matrix, Pseudo-code 3 shows the main steps performed by a general method of the *Model* class.

The *Model* class of Fig. 3 is linked to engineering applications and modeling through the FEM. Figure 5 shows the organization adopted for this class, where the derived classes and their meanings are listed below:

- *ModBeam2D* – Two-dimensional Geometric Nonlinear Beam Model;
- *Constitutive Model* – Generic Model for Material Nonlinear Problems.

Note that the derived *ModBeam2D* class contains general methods for performing geometrical nonlinear analyses of discrete beam structures, while the methods of the derived *Constitutive Model* class are associated to the material nonlinear behavior of continuum structures. The shaded boxes in Fig. 5 indicate the classes which are discussed in detail in this work (see Sections 5 and 6). It is important to mention that new physical models can also be nat-

```
begin
  < 1 > Initialize Global Stiffness Matrix (K^g)
  < 2 > foreach element of Current Model:
        begin
          < 2.1 > Get Element Stiffness Matrix (K^g_e)
          < 2.2 > Assemble Global Stiffness Matrix, i.e.
                  K^g ⟵ K^g + K^g_e
        end
end
```

Pseudo-code 3. Model Class – Global tangent stiffness matrix.

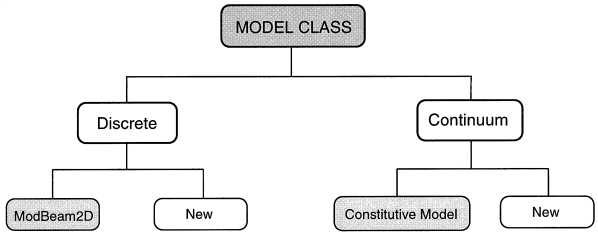

Fig. 5. *Model* class organization.

urally incorporated in the *Model* class (see Reference 28). Easy incorporation of new models is one of the main features of the present FEM OOP system. Finally, *MathPack* and *MemPack* (see Fig. 3) are auxiliary classes responsible for general mathematical computations (e.g. norm of a vector, solution of linear systems of equations) and dynamic memory allocation, respectively.

5. Geometrically Nonlinear Problems

A brief description of a geometrically nonlinear two-dimensional beam model is given here. It follows the so-called 'total formulation', and is based on the work by Pacoste and Eriksson [32,33], which accounts for shear effects. This model is incorporated in the present FEM OOP system in the particular class *ModBeam2D* (see Fig. 5). The following derivation gives an idea of the procedures required to add a new model in the system.

Figure 6 shows the deformed configuration of a plane beam subjected to large displacements and rotations, but small strains. The reference configuration corresponds to a straight line element of length L, initially on the local x axis.

Each point on the beam axis is subjected to axial ($u(x)$) and transversal ($\nu(x)$) displacements, and is associated to a generic cross section S, which in turn may undergo finite rotations ($\theta(x)$), where the abscissa $x \in [0,L]$ is measured on the reference configuration. The deformed configuration is described by means of the position vector $\boldsymbol{r}(x)$, defined as

$$\boldsymbol{r}(x) = [x + u(x)]\,\boldsymbol{i} + v(x)\,\boldsymbol{j} \tag{5}$$

where $\boldsymbol{i}^T = [1,0]$, $\boldsymbol{j}^T = [0,1]$, and the superscript T means transpose. Deformation measures can be obtained by writing the tangent vector $(\partial \boldsymbol{r}(x)/\partial x)$ with respect to the orthogonal basis $(\boldsymbol{a},\boldsymbol{b})$, i.e.

$$\frac{\partial \boldsymbol{r}(x)}{\partial x} = [1 + \epsilon(x)]\,\boldsymbol{a} + \gamma(x)\,\boldsymbol{b} \tag{6}$$

$$\kappa(x) = \frac{\partial \theta(x)}{\partial x}$$

where $\epsilon(x)$, $\gamma(x)$ and $\kappa(x)$ are linear strain, angular strain, and curvature, respectively, and

$$\boldsymbol{a}(x) = \cos\theta(x)\,\boldsymbol{i} + \sin\theta(x)\,\boldsymbol{j} \tag{7}$$

$$\boldsymbol{b}(x) = -\sin\theta(x)\,\boldsymbol{i} + \cos\theta(x)\,\boldsymbol{j}$$

are unit vectors orthogonal and parallel to the deformed cross section. In general, the vectors $\partial \boldsymbol{r}(x)/\partial x$ and $\boldsymbol{a}$ are not aligned (see Fig. 7). However, alignment is preserved if the angular strain ($\gamma(x)$) is neglected, which occurs, for instance, in beams described by Bernoulli hypothesis.

Combining Eqs (5), (6) and (7), one obtains

$$\epsilon(x) = \left[1 + \frac{\partial u(x)}{\partial x}\right]\cos\theta(x) + \frac{\partial v(x)}{\partial x}\sin\theta(x) - 1$$

$$\gamma(x) = -\left[1 + \frac{\partial u(x)}{\partial x}\right]\sin\theta(x) + \frac{\partial v(x)}{\partial x}\cos\theta(x) \tag{8}$$

$$\kappa(x) = \frac{\partial \theta(x)}{\partial x}$$

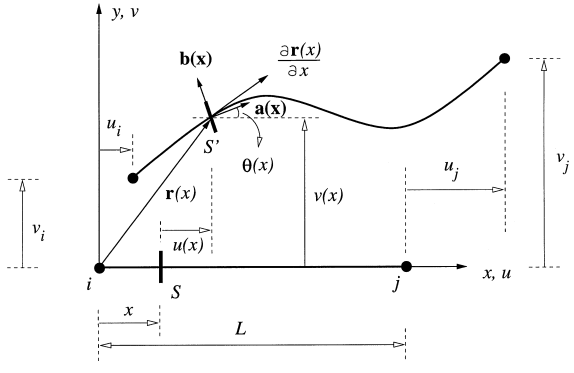

Fig. 6. Plane beam: Initial and deformed configurations.

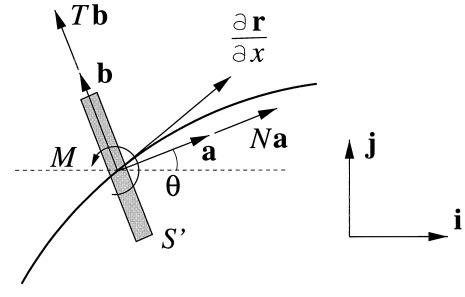

Fig. 7. Detail of a generic plane beam cross section.

The internal force ($\boldsymbol{F}$) and moment ($\boldsymbol{M}$) acting on a generic cross section can be written as

$$\boldsymbol{F} = N\,\boldsymbol{a} + T\,\boldsymbol{b}$$

$$\boldsymbol{M} = M\,\boldsymbol{a} \times \boldsymbol{b} \tag{9}$$

where the symbol $\times$ denotes the cross-product and N, T and M are conjugate scalar variables with respect to the deformation measures (in the energy sense). Note that, according to Fig. 7, the axial force (N) and the shear force (T) are normal and parallel to the deformed cross section, respectively. Assuming linear constitutive relation, one may write

$$N = EA\epsilon$$

$$T = GA\gamma$$

$$M = EI\kappa \tag{10}$$

where EA, GA and EI are the axial, shear and flexural rigidities, respectively. Accordingly, the strain energy U is given by

$$U = \frac{1}{2}\int_0^L (EA\epsilon^2 + GA\gamma^2 + EI\kappa^2)\,dx \tag{11}$$

5.1. Finite Element Aspects

The element for performing geometrically nonlinear analysis of plane frames, taking shear deformation into account, is described here. It is based on a linear interpolation of the displacement field

$$\boldsymbol{u}^T = [u(x),\, v(x),\, \theta(x)]$$

with respect to the nodal degrees of freedom

$$(\hat{\boldsymbol{u}}_e^l)^T = [u_i,\, v_i,\theta_i;\, u_j,v_j,\theta_j]$$

in a local coordinate system, as illustrated by Fig. 8.

Equation (11) can be evaluated by one-point Gaussian quadrature. As pointed out by Pacoste and Eriksson [33], the procedure of using reduced integration avoids locking problems. Once the interpolation function and the integration strategy are defined, all terms necessary to evaluate Eq. (11) can be computed, e.g.

$$\frac{\partial u(x)}{\partial x} = \frac{u_j - u_i}{L}$$

$$\frac{\partial v(x)}{\partial x} = \frac{v_j - v_i}{L}$$

$$\frac{\partial \theta(x)}{\partial x} = \frac{\theta_j - \theta_i}{L}$$

$$\theta(x) = \frac{\theta_i + \theta_j}{2} \tag{12}$$

The local internal force vector[1]

$$\left(\hat{\boldsymbol{f}}_e^{\,l}\right)^T = [N_i,T_i,M_i;\, N_j,T_j,M_j]$$

and the local tangent stiffness matrix $\boldsymbol{K}_e^l$ are obtained by means of the following expressions:

$$\left(\hat{\boldsymbol{f}}_e^{\,l}\right)^T = \left[\frac{\partial U}{\partial u_i}, \frac{\partial U}{\partial v_i}, \ldots, \frac{\partial U}{\partial \theta_j}\right] \tag{13}$$

and

$$\boldsymbol{K}_e^l = \begin{bmatrix} \dfrac{\partial^2 U}{\partial u_i \partial u_i} & \dfrac{\partial^2 U}{\partial u_i \partial v_i} & \cdots & \dfrac{\partial^2 U}{\partial u_i \partial \theta_j} \\ \vdots & & & \\ \dfrac{\partial^2 U}{\partial \theta_j \partial u_i} & \dfrac{\partial^2 U}{\partial \theta_j \partial v_i} & \cdots & \dfrac{\partial^2 U}{\partial \theta_j \partial \theta_j} \end{bmatrix} \tag{14}$$

respectively. Finally, to obtain the internal force vector and the tangent stiffness matrix with respect to the global coordinate system ($\hat{\boldsymbol{f}}_e^{\,g}$ and $\boldsymbol{K}_e^g$, respectively), a transformation matrix $\boldsymbol{T} \equiv \boldsymbol{T}(\varphi)$ of order 6 is defined, where φ is illustrated in Fig. 8.

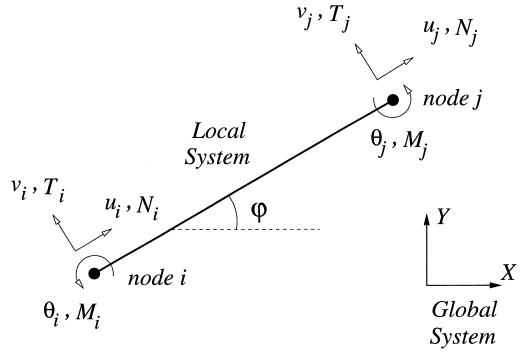

Fig. 8. Finite element for plane beam problems.

5.2. Pseudo-codes of ModBeam2D

Pseudo-codes 4–6 illustrate the addition of a nonlinear beam model into the computational system.

[1] Note that (N_i,T_i,M_i; N_j,T_j,M_j) are the nodal values of the internal forces with respect to the local coordinate system, while (N,T,M) are the internal forces acting on a generic cross-section, according to a reference coordinate system as illustrated in Fig. 7.

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
//     ModBeam2D – Generic Bidimensional Beam Model
//
//   Based on Reference [33].
//
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
cModBeam2D :: cModBeam2D ( void ) : cDiscrete ( ) : cModel ( )
begin
  < 1 >  Read Input Data
  < 2 >  Generate d.o.f.'s for Current Model
  < 3 >  Generate Reference Load Vector
end

cModBeam2D :: ~cModBeam2D ( void )
begin
  < 1 >  De-allocate Arrays Dynamically Allocated
  < 2 >  Close All Files
end

cModBeam2D :: Init ( void )
begin
  < 1 >  Open Output Files
end

cModBeam2D :: StepConvergence ( double dλ,  double *solution )
begin
  < 1 >  Print Partial Solution:
         (e.g. current load level, current equilibrium configuration)
end
```

Pseudo-code 4. ModBeam2D Class – eneral methods.

```
begin
  < 1 >  Compute Initial Element Length and Orientation
  < 2 >  Compute Transformation Matrix (T)
         between Local and Global Coordinate Systems
  < 3 >  Generate Element Displacement Vector (û_e^g)
         from the given Global Displacement Vector (û^g)
  < 4 >  Rotate Element Displacements (û_e^g) from Global
         to Local Coordinate Systems, i.e. :
         û_e^l ⟵ T û_e^g
  < 5 >  Compute Element Local Internal Force Vector (f̂_e^l)
         from the Element Local Displacement Vector (û_e^l)
         (See Equation (13))
  < 6 >  Rotate Element Internal Forces (f̂_e^l) from Local
         to Global Coordinate System, i.e. :
         f̂_e^g ⟵ T^T f̂_e^l
end
```

Pseudo-code 5. *ModBeam2D* Class – Element internal force vector.

Pseudo-codes 5 and 6 show how the element internal force vector ($\hat{\boldsymbol{f}}_e^g$) and element tangent stiffness matrix ($\boldsymbol{K}_e^g$), respectively, are obtained with respect to the global coordinate system, within the context of the *ModBeam2D* class. Other models (for performing geometrically and/or materially nonlinear analyses of trusses, for example) can also be easily added to the system. This feature (i.e. 'ease of use') has been one of the main guiding principles in the design of the present FEM OOP environment.

```
begin
  < 1 >  Compute Initial Element Length and Orientation
  < 2 >  Compute Transformation Matrix (T)
         between Local and Global Coordinate Systems
  < 3 >  Generate Element Displacement Vector (û_e^g)
         from the given Global Displacement Vector (û^g)
  < 4 >  Rotate Element Displacements (û_e^g) from Global
         to Local Coordinate Systems, i.e. :
         û_e^l ⟵ T û_e^g
  < 5 >  Compute Element Local Stiffness Matrix (K_e^l)
         from the Element Local Displacement Vector (û_e^l)
         (See Equation (14))
  < 6 >  Rotate Element Local Stiffness Matrix (K_e^l) from Local
         to Global Coordinate Systems, i.e. :
         K_e^g ⟵ T^T K_e^l T
end
```

Pseudo-code 6. *ModBeam2D* Class – Element tangent stiffness matrix.

5.3. Example using the ModBeam2D Class

The goal of this section is to illustrate the practical use of the *ModBeam2D* class by means of a nontrivial example of a frame-type structure, which exhibits a complicated equilibrium path with snap back behavior. Figure 9 shows a pin-supported plane frame subjected to a single vertical concentrated load. This problem was originally presented by Lee et al. [34], and was also later studied by several researchers such as Frey and Cescotto [35], Schweizerhof and Wriggers [36], Simo and Vu-Quoc [37], Chen and Blandford [38] and Pacoste and Eriksson [33]. Lee et al. [34] have presented an analytical solution for the structural behavior of their frame problem, which is characterized by large displacements and rotations. However, they did not consider shear effects in their derivations. These effects are considered in the formulation implemented in the *ModBeam2D* class.

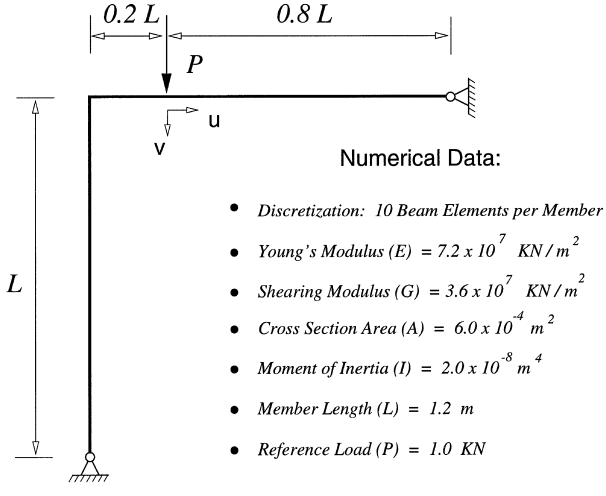

Fig. 9. Lee frame example description.

The numerical data adopted for the Lee frame example is also presented in Fig. 9. Because a load control algorithm (e.g. Newton–Raphson) cannot capture snap back behavior, this problem was solved by means of the GDCM [30,39] (see Fig. 4). The incremental-iterative solution procedure was carried out with 200 steps, and the maximum number of iterations that occurred in a step was 3. The convergence criterion was established in terms of the ratio of the Euclidean norms of the unbalanced forces (residual) and the reference load vector. Since the latter is the unit for this example, the tolerance (TOL) is the norm of the unbalanced forces which was set as TOL = 0.001. The CPU time in a SUN SunSPARCstation 20 (96 Mbytes of memory) was 46 seconds.

The load-displacement curves corresponding to the displacements u and v, at the load application point (fixed-point load), are given in Fig. 10, where each dot on the curves corresponds to one step. These results clearly show snap back behavior. The capital letters in Fig. 10 denote a point in the solution path corresponding to the deformed shapes (for various load levels) shown in Fig. 11. The results obtained here are in agreement with those obtained by the program MASTRAN2 (Matrix Structural Analysis 2) [40]. For instance, the limit load obtained in the present study is 18.792KN (see Fig. 10), and the MASTRAN2 result considering classical beam theory (i.e. no transverse shear effect[2]) is 18.454KN.

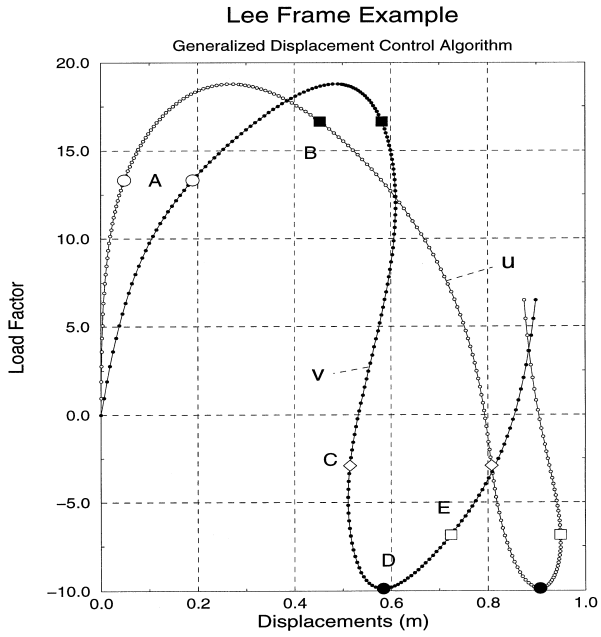

Fig. 10. Load × displacement curves.

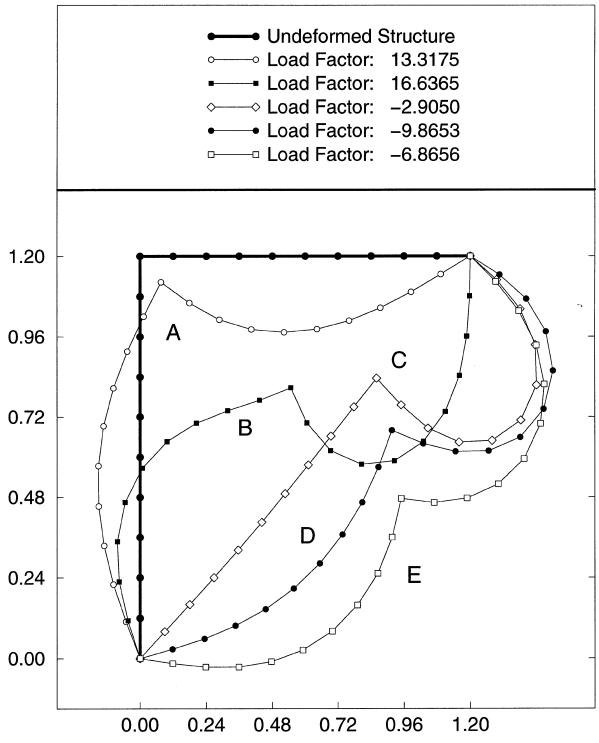

Fig. 11. Deformed shapes.

6. Materially Nonlinear Problems

Many materials (e.g. metals, polymers, ceramics, soil, concrete and rock) may fail by some form of localization of deformation. Consider, for example, the compression test illustrated in Fig. 12. Localization is often followed by a decrease of the load bearing capacity after reaching the peak load. It often leads to fracture (either ductile or brittle) because the deformations accumulated in the small localization band facilitate rupture. As illustrated by Fig. 12, an initially homogeneous state of deformation is differentiated from the one at the localization band, where the deformation increments localize in a small region of the material. Experimental findings indicate that there is a relationship between the size of the localization band and the material microstructure. Thus, the important notion of a 'characteristic length', which sets the size of the band, is introduced.

With respect to the behavior described above, consider the constitutive (stress/strain) relation

[2] At the time of this writing, MASTRAN2 uses the simplified notion of shear area (A_s) to deal with transverse shear deformation, however, this consideration was not employed here.

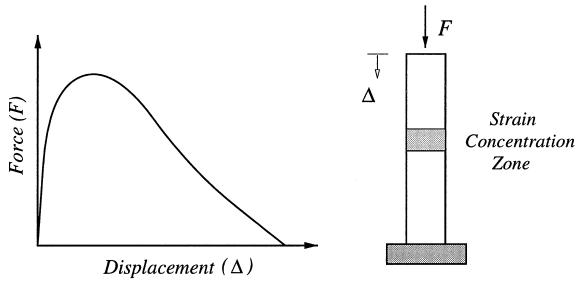

Fig. 12. Macroscopic behavior in a compression test.

through an homogenization procedure with respect to the results of the test. The applied force is divided by the cross-sectional area and the displacement at the tip of the bar is divided by a certain gage length (e.g. the initial length of the bar). Therefore, the observed macroscopic (stress/strain) behavior is described in a pointwise fashion.

After discretization of the continuum by the FEM, the numerical solutions show pronounced dependence upon the discretization level, and are therefore physically inconsistent [41]. The reason for this is because classical constitutive models are expressed in terms of (averaged) stresses and strains defined locally. When a localized deformation mode gets activated by a local defect, the characteristic deformation scale and the microstructure size become comparable. The state of the material at a point depends on the deformation history of a certain neighborhood of this point (i.e. the material behavior is *nonlocal*). Therefore, to remedy the situation, an internal length scale (characteristic length) is used in the continuum description.

Enhanced continuum models provide a consistent treatment of localization problems, e.g. Cosserat (micropolar) continuum, nonlocal (integral) model, and higher-order gradient continuum. Here, an elastoplastic model within the framework of Cosserat continuum, and its corresponding implementation using object-oriented concepts, are presented [42]. Additional strain (curvatures) and stress (couple-stress) measures enter the kinematic and static description of the continuum, which require the introduction of additional material parameters (internal length scale). In this way, the boundary value problem remains well-posed after strain localization occurs [43].

6.1. Cosserat (Micropolar) Continuum

In the Cosserat continuum (also denoted micropolar continuum), the microstructure of the material is explicitly taken into account, which permits to differentiate the rotation of the continuum (macrorotation) from the rotation of the microstructure (microrotation) [44,45]. This leads to the following (linear) strain tensors

$$\gamma_{ij} = u_{j,i} - \epsilon_{ijk}\phi_k \tag{15}$$

$$k_{ij} = \phi_{j,i} \tag{16}$$

where u_i and ϕ_i are the displacement and microrotation vectors, respectively, and ϵ_{ijk} is the alternator tensor. The tensor γ_{ij} is associated to change of dimensions and distortion, while the tensor k_{ij} is associated to curvatures and twist of microstructure. Figure 13 shows the generalized stresses associated with the strain measures. The stress tensor is now composed of stresses σ_{ij} and couple-stresses μ_{ij}, and it is not necessarily symmetric anymore.

For plane strain and no initial stresses or deformations, the constitutive relationship for the linear elastic and isotropic material is given by Eq. (17), where G and ν are classical parameters, denoted as shear modulus and Poisson ratio, respectively. The parameter α relates the stiffness between the macrorotation and the microrotation, and the parameter ℓ

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{21} \\ \hline \mu_{13} \\ \mu_{23} \end{Bmatrix} = \left[\begin{array}{ccccc|cc} \frac{2G(1-\nu)}{1-2\nu} & \frac{2G\nu}{1-2\nu} & \frac{2G\nu}{1-2\nu} & 0 & 0 & 0 & 0 \\ \frac{2G\nu}{1-2\nu} & \frac{2G(1-\nu)}{1-2\nu} & \frac{2G\nu}{1-2\nu} & 0 & 0 & 0 & 0 \\ \frac{2G\nu}{1-2\nu} & \frac{2G\nu}{1-2\nu} & \frac{2G(1-\nu)}{1-2\nu} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & G+\alpha & G-\alpha & 0 & 0 \\ 0 & 0 & 0 & G-\alpha & G+\alpha & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 2G\ell^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2G\ell^2 \end{array}\right] \begin{Bmatrix} \gamma_{11} \\ \gamma_{22} \\ \gamma_{33} \\ \gamma_{12} \\ \gamma_{21} \\ \hline \kappa_{13} \\ \kappa_{23} \end{Bmatrix} \tag{17}$$

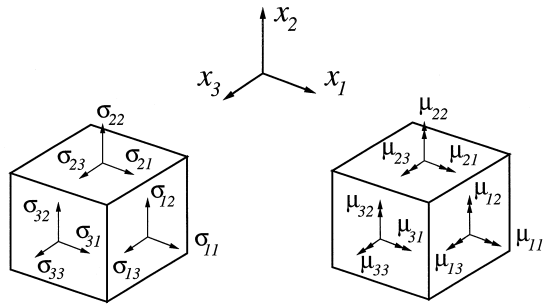

Fig. 13. Stress components for micropolar continuum.

has the dimension of length, and therefore introduces a length scale in the continuum description.

The differential equations of equilibrium are expressed by

$$\sigma_{ji,j} + \bar{f}_i = 0 \tag{18}$$

$$\mu_{ji,j} + \epsilon_{ijk}\,\sigma_{jk} + \bar{I}_i = 0 \tag{19}$$

where $\bar{f}_i$ and $\bar{I}_i$ are body forces and body couples per unit volume, respectively.

In this enriched continuum, the boundary conditions to be specified are

$$\sigma_{ji}\,n_j = \bar{t}_i \text{ in } S_t \quad \text{or} \quad u_i = \bar{u}_i \text{ in } S_u \tag{20}$$

$$\mu_{ji}\,n_j = \overline{m}_i \text{ in } S_m \quad \text{or} \quad \phi_i = \overline{\phi}_i \text{ in } S_\phi \tag{21}$$

where S_t, S_m, S_u and S_ϕ denote regions of the boundary with prescribed stress, couple-stress, displacements and microrotations, respectively.

6.2. Cosserat Elastoplastic Model

The main feature of Cosserat elastoplasticity is the use of the classical plasticity framework [46] by augmenting the vectors and matrices involved. This leads to an elegant formulation, which is also natural to the OOP philosophy.

The yield function adopted here is a generalization of the classical von Mises criterion, and it is given by

$$f = \sqrt{3J_2} - \overline{\sigma}(\beta) \tag{22}$$

where J_2 is the second invariant of the deviatoric stresses, and $\overline{\sigma}(\beta)$ is the yield stress which depends upon the hardening (or softening) parameter β. The specific function adopted in this work is

$$\overline{\sigma}(\beta) = \sigma_Y + h\beta \tag{23}$$

where σ_Y is the yield stress, and h is the plastic modulus. Other hardening functions can also be employed within the present framework of analysis.

As usually done in computational solid mechanics, the strain and stress tensors are recast in vector form. All the strain components are assembled in the strain vector

$$\gamma^T = [\gamma_{11}\ \gamma_{22}\ \gamma_{33}\ \gamma_{12}\ \gamma_{21} \mid \kappa_{13}\ell\ \kappa_{23}\ell\ ] \tag{24}$$

where the length parameter ℓ has been introduced so that all entries in γ are dimensionless. Accordingly, the stress components are assembled in the stress vector

$$\boldsymbol{\sigma}^T = [\sigma_{11}\ \sigma_{22}\ \sigma_{33}\ \sigma_{12}\ \sigma_{21} \mid \mu_{13}/\ell\ \mu_{23}/\ell] \tag{25}$$

In compact notation, J_2 (see Eq. (22)) can be rewritten as [47,48]

$$J_2 = \frac{1}{2}\,\sigma_i\,P_{ij}\,\sigma_j \tag{26}$$

where

$$\mathbf{P} = \left[\begin{array}{ccccc|cc} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{27}$$

According to the plasticity criterion adopted here, the material is not sensitive to the hydrostatic and anti-symmetric states of the stress tensor. For a point $(\sigma_{ij},\ \beta)$ on the yield surface, the flow rule (associated) is given by

$$\dot{\gamma}_i^p = \frac{3\dot{\lambda}}{2\overline{\sigma}(\beta)} P_{ij}\,\sigma_j \tag{28}$$

where the superscript p stands for plastic, and the superimposed dot refers to the rate variation, which is a standard terminology in theoretical and computational inelasticity. The variation of the parameter β, which is conjugate to the definition of J_2 [46], is

$$\dot{\beta} = \sqrt{\frac{2}{3}\,\dot{\gamma}_i^p\,P_{ij}\,\dot{\gamma}_j^p} \tag{29}$$

leading to

$$\dot{\beta} = \dot{\lambda} \tag{30}$$

6.3. Class Organization

In general, the establishment of equilibrium in a system exhibiting material nonlinear behavior can be understood in two levels: first, the *global equilibrium*, which is considered here through the environment NLS++ (see Section 4) and, secondly, the *local equilibrium*, at the level of quadrature points with specific treatment for each constitutive model. The discretization procedure used in the *structural model class*, which is part of the NLS++ environment, is responsible for defining the internal force vector of the structure, as well as its tangent stiffness matrix. In the FEM context, these quantities are generated from contributions of the elements in the mesh, which are evaluated by means of pieces of information obtained in a finite number of points (i.e. quadrature points). To build the internal force vector, the basic problem consists of determining the increment of stress at the integration point, from the increment of deformation. The tangent stiffness matrix is defined such that it is consistent with the current deformation state [46].

On the basis of how constitutive behavior affects the overall nonlinear process, an organization for the class *Constitutive Model* is suggested in Fig. 14. The objective of this class is to effect the integration of constitutive equations for each mathematical model used to describe the nonlinear behavior. Initially, two major groups generating subclasses of the *Constitutive Model* class are identified, each of which executes specific tasks to integrate the constitutive equations. The sub-classes are: *Linear Elastic* and *Elastoplastic*. An instance of this class is referenced at each quadrature point of the element. Although the *von Mises* model is a particular case of the *Micropolar von Mises* model, they are at the same level in the diagram of Fig. 14 because inheritance of methods has not been employed in the present computational setting.

According to the scope of the present work, Table 1 presents the main equations involved in the sub-class *Elastoplastic* [46]. In elastoplastic constitutive models, stresses can be determined from the strain state (*strain driven problem*) through return-type algorithms (e.g. *Cutting-plane algorithm*). In this work, emphasis is given to the *Closest Point Projection* [46]. Since this algorithm is a common task to all sub-classes derived from the *Elastoplastic* class, it is one of the main methods of this class. The *Closest Point Projection* algorithm is based on linearization and simultaneous satisfaction of the *yield function*, *flow rule* and *hardening law*; and can be thought as an application of the Newton–Raphson method to this set of equations. Table 2 presents the main steps involved in this algorithm.

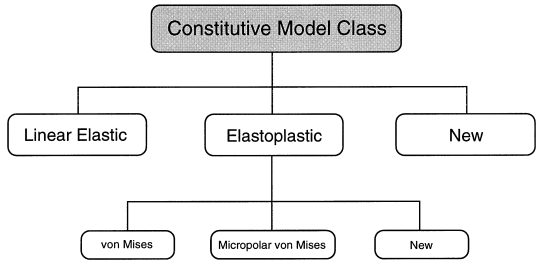

Fig. 14. Hierarchy of the *Constitutive Model Class*.

The sub-classes derived from the *Elastoplastic* model should provide the dimensions of vectors and matrices involved, as well as specific pieces of information associated with each model (e.g. f,$\partial f/\partial \boldsymbol{\sigma}$,$\partial f/\partial \boldsymbol{q}$,$\partial^2 f/\partial \boldsymbol{\sigma}^2$,$\partial^2 f/\partial \boldsymbol{q}^2$,$\partial^2 f/\partial \boldsymbol{\sigma}\partial \boldsymbol{q}$,$\partial^2 f/\partial \boldsymbol{q}\partial \boldsymbol{\sigma}$ and $\boldsymbol{D}$). The elastic constitutive matrix $\boldsymbol{C}$ is provided by an auxiliary class responsible for the analysis type (e.g. plane stress, plane strain, axisymmetric). This class, called *Analysis Model* class, is also responsible for providing the strain-displacement matrix, which

Table 1. Classical rate independent plasticity

(1)	Additive decomposition of strain tensor: $\gamma = \gamma^e + \gamma^p$, where γ^e and γ^p represent elastic and plastic deformations, respectively.
(2)	Elastic constitutive equation: $\boldsymbol{\sigma} = \boldsymbol{C}\gamma^e = \boldsymbol{C}(\boldsymbol{\gamma} - \boldsymbol{\gamma}^p)$, where $\boldsymbol{C}$ is the linear elastic constitutive matrix.
(3)	Yield function: $f(\boldsymbol{\sigma},\boldsymbol{q}) \leq 0$, where $\boldsymbol{\sigma}$ and $\boldsymbol{q}$ represent the stress state and the internal variables of the constitutive model, respectively. For the specific micropolar elastoplastic constitutive model presented, $\boldsymbol{q} = -\{h\beta\}$.
(4)	Associative flow rule: $\dot{\boldsymbol{\gamma}}^p = \dot{\lambda}\frac{\partial f}{\partial \boldsymbol{\sigma}}$, where $\dot{\lambda}$ is the plastic multiplier (or consistency parameter)
(5)	Hardening law: $\dot{\boldsymbol{q}} = -\dot{\lambda}\boldsymbol{D}\frac{\partial f}{\partial \boldsymbol{q}}$ or $\dot{\alpha} = \dot{\lambda}\frac{\partial f}{\partial \boldsymbol{q}}$, where: $\boldsymbol{\alpha} = -\boldsymbol{D}^{-1}\boldsymbol{q}$. For the micropolar elastoplastic constitutive model, $\boldsymbol{D} = [h]$ and $\boldsymbol{\alpha} = \{\beta\}$.
(6)	Kuhn–Tucker loading/unloading conditions: $\dot{\lambda} \geq 0,\ f \leq 0,\ \dot{\lambda}f = 0$
(7)	Consistency condition: $\dot{\lambda}\dot{f} = 0$

Table 2. Closest point projection algorithm

(1) Initialize:
$k = 0, \quad \boldsymbol{\gamma}^{p(k)}_{n+1} = \boldsymbol{\gamma}^{p}_{n}, \quad \boldsymbol{\alpha}^{(k)}_{n+1} = \boldsymbol{\alpha}_{n}, \quad \Delta\lambda^{(k)}_{n+1} = 0$

(2) Compute stresses, internal variables, yield function and flow rule/hardening law residuals:

$$\boldsymbol{\sigma}^{(k)}_{n+1} = \boldsymbol{C}\left(\boldsymbol{\gamma}_{n+1} - \boldsymbol{\gamma}^{p(k)}_{n+1}\right)$$

$$\boldsymbol{q}^{(k)}_{n+1} = -\boldsymbol{D}\,\boldsymbol{\alpha}^{(k)}_{n+1}$$

$$f^{(k)}_{n+1} = f\left(\boldsymbol{\sigma}^{(k)}_{n+1}, \boldsymbol{q}^{(k)}_{n+1}\right)$$

$$\boldsymbol{R}^{(k)}_{n+1} = \begin{Bmatrix} -\boldsymbol{\gamma}^{p(k)}_{n+1} + \boldsymbol{\gamma}^{p}_{n+1} \\ -\boldsymbol{\alpha}^{(k)}_{n+1} + \boldsymbol{\alpha}_{n} \end{Bmatrix} + \Delta\lambda^{(k)}_{n+1} \begin{Bmatrix} \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}} \\ \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{q}} \end{Bmatrix}$$

(3) If $f^{(k)}_{n+1} \le TOL_1$ and $\|\boldsymbol{R}^{(k)}_{n+1}\| \le TOL_2$ then *STOP*, else:

(4) Compute Hessian matrix of the Newton's problem:

$$\left(\boldsymbol{A}^{(k)}_{n+1}\right)^{-1} = \begin{bmatrix} \boldsymbol{C}^{-1} + \Delta\lambda^{(k)}_{n+1}\dfrac{\partial^2 f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}^2} & \Delta\lambda^{(k)}_{n+1}\dfrac{\partial^2 f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}\partial \boldsymbol{q}} \\ \Delta\lambda^{(k)}_{n+1}\dfrac{\partial^2 f^{(k)}_{n+1}}{\partial \boldsymbol{q}\partial \boldsymbol{\sigma}} & \boldsymbol{D}^{-1} + \Delta\lambda^{(k)}_{n+1}\dfrac{\partial^2 f^{(k)}_{n+1}}{\partial \boldsymbol{q}^2} \end{bmatrix}$$

(5) Compute increment of increment of plastic consistency parameter:

$$\Delta\Delta\lambda^{(k)}_{n+1} = \frac{f^{(k)}_{n+1} - \left[\left(\dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}}\right)^T \left(\dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{q}}\right)^T\right] \boldsymbol{A}^{(k)}_{n+1}\boldsymbol{R}^{(k)}_{n+1}}{\left[\left(\dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}}\right)^T \left(\dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{q}}\right)^T\right] \boldsymbol{A}^{(k)}_{n+1} \begin{Bmatrix} \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}} \\ \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{q}} \end{Bmatrix}}$$

(6) Obtain incremental plastic strains and internal variables:

$$\begin{Bmatrix} \Delta\boldsymbol{\gamma}^{p(k)}_{n+1} \\ \Delta\boldsymbol{\alpha}^{(k)}_{n+1} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{C}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{-1} \end{bmatrix} \boldsymbol{A}^{(k)}_{n+1} \left\{ \boldsymbol{R}^{(k)}_{n+1} + \Delta\Delta\lambda^{(k)}_{n+1} \begin{Bmatrix} \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{\sigma}} \\ \dfrac{\partial f^{(k)}_{n+1}}{\partial \boldsymbol{q}} \end{Bmatrix} \right\}$$

(7) Update state variables and increment of consistency parameter:

$$\boldsymbol{\gamma}^{p(k+1)}_{n+1} = \boldsymbol{\gamma}^{p(k)}_{n+1} + \Delta\boldsymbol{\gamma}^{p(k)}_{n+1}$$
$$\boldsymbol{\alpha}^{(k+1)}_{n+1} = \boldsymbol{\alpha}^{(k)}_{n+1} + \Delta\boldsymbol{\alpha}^{(k)}_{n+1}$$
$$\Delta\lambda^{(k+1)}_{n+1} = \Delta\lambda^{(k)}_{n+1} + \Delta\Delta\lambda^{(k)}_{n+1}$$

(8) $k = k + 1$ and *GOTO* Step (2).

is used for computing the element internal force vector and the element tangent stiffness matrix. This is an auxiliary class for the *Constitutive Model* class, which plays a role somehow analogous to the auxiliary classes *MathPack* and *MemPack* of the NLS++ system, illustrated in Fig. 3. Improvement in computational performance can be achieved if the inverses of $\boldsymbol{C}$ and $\boldsymbol{D}$ are also provided by the corresponding classes.

When the structure reaches *global equilibrium*, the *Elastoplastic* class requires a method for updating the 'state variables' at the equilibrium point (e.g. $\boldsymbol{\gamma}^{p}_{n+1}$ and $\boldsymbol{\alpha}_{n+1}$). Finally, the tangent constitutive matrix must be defined in the *Elastoplastic* class. Here the so-called *Algorithmic Tangent Elastoplastic Moduli* is used, which is consistent with the algorithm employed to update the stresses [46], and it leads to

$$C^{ep}_{alg} = \boldsymbol{A}_{n+1} - \begin{cases} \boldsymbol{C} & \text{if } \dot{\lambda} = 0 \\ \dfrac{\boldsymbol{A}_{n+1}\dfrac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}\left(\dfrac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}\right)^T \boldsymbol{A}_{n+1}}{\left[\left(\dfrac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}}\right)^T \left(\dfrac{\partial f_{n+1}}{\partial \boldsymbol{q}}\right)^T\right] \boldsymbol{A}_{n+1} \begin{Bmatrix} \dfrac{\partial f_{n+1}}{\partial \boldsymbol{\sigma}} \\ \dfrac{\partial f_{n+1}}{\partial \boldsymbol{q}} \end{Bmatrix}} & \text{if } \dot{\lambda} > 0 \end{cases} \tag{31}$$

where $\boldsymbol{A}_{n+1}$ is given in Step 4 of Table 2. It is worth noting that for a linear elastic analysis, the matrix $\boldsymbol{C}$ is not modified by the *Linear Elastic* class, and for an elastoplastic analysis, it is modified according to Eq. (31). Thus, whenever necessary, the derived classes of the *Constitutive Model* class can modify the matrix $\boldsymbol{C}$.

6.4. Example Involving Localization of Deformation

To demonstrate that the present implementation works and to illustrate the use of the OOP environment, consider the compression test illustrated in Fig. 15, in which strain localization into a shear band takes place at the onset of softening. This figure illustrates the geometry and boundary conditions of the test specimen, which is modeled as a plane strain problem. The bottom edge does not move and the upper edge is constrained to move as a linear segment (without rotation). For comparison, this problem is analyzed using both classical and Cosserat elastoplastic models.

The problem domain is discretized with 9-node

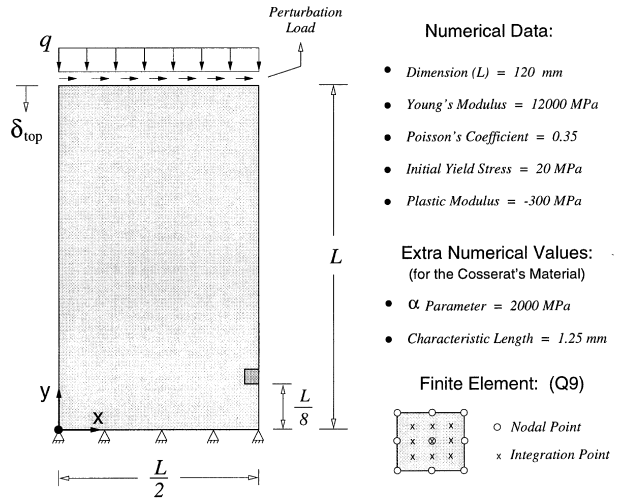

Fig. 15. Compression test layout.

Table 3. Numerical results for the classical continuum[a]

Discretization	#DOF	#Steps	#Iterations(max)
08×16	1056	240	11
16×32	4160	136	14
28×56	12656	92	76

[a]SUN SunSPARCstation 20 with 96 Mbytes of memory running *SunOS 4.1.4*

quadratic Lagrangian finite elements using Gaussian integration with nine quadrature points per element. The nonlinear system of equations is solved by means of the GDCM (see Fig. 4), which is the same method employed to solve the Lee frame in Section 5.3. The tolerance criterion for convergence of the iterations is defined as the ratio between the norm of the unbalanced force vector and the norm of the reference force vector, and it is set as TOL = 0.0001. To follow an equilibrium path associated with a localized deformation mode, a relatively small (1% of the vertical loading) uniformly distributed horizontal loading at the upper face, and a 5% reduction of σ_Y for the element with bottom right-hand side node located at L/8 from the bottom edge of the specimen (see Fig. 15) are used.

6.4.1. FEA Using Classical Elastoplastic Model
The values of the parameters necessary to perform this analysis are given in Fig. 15. Three levels of mesh discretization are employed, and the deformed configurations for the final step of the analyses are shown in Figs 16, 17 and 18. For each discretization, Table 3 shows the number of degrees of freedom (#DOF) in the finite element model, the total number of steps (#Steps), and the maximum number of iterations in a step (#Iterations (max)).

Figure 19 shows the relation between the applied load level and the vertical displacement at the upper face of the test specimen for different discretization levels. This graph shows that the descending branches of the curve tend to get closer to the ascending branch as the mesh is refined. The results also depend on the arrangement of the elements, i.e. exhibit directional bias. Moreover, the deformation tend to localize in a region of zero thickness (see Figs 16, 17 and 18) with vanishing dissipative energy. This pathological behavior is remedied by means of the micromorphic continuum model, which is discussed next.

6.4.2. FEA Using Cosserat Elastoplastic Model
In this analysis, the degrees of freedom corresponding to microrotation are set free for all the nodes of the finite element meshes. The additional parameters required for the Cosserat material are also given in Fig. 15. The same initial meshes adopted in the classical analysis are used here, and the deformed configurations for the final step of the analyses are shown in Figs 20, 21 and 22. Selected representative results are provided in Table 4, which uses the same nomenclature as Table 3.

The equilibrium trajectories for the various levels of discretization are given in Fig. 23, which shows reduced sensitivity to mesh refinement, leading to a more stable solution. The agreement on the post-peak trajectory, especially for the meshes 16 × 32 and 28 × 56, guarantees a finite dissipative energy and localization to a band of finite thickness (see Figs 20, 21 and 22).

6.4.3. Comparison
It is worth comparing Table 3, and Figs 16, 17, 18 and 19 for the classical elastoplastic model with Table 4, and Figs 20, 21, 22 and 23 for the Cosserat elastoplastic model. In Figs 16 to 18, and 20 to 22, the deformations have been amplified by a factor of 15. Moreover, the CPU time required by the Cosserat continuum is approximately three times the one required by the classical continuum.

7. Conclusions and Extensions

An OOP framework for solving physical problems in solid and structural mechanics, by means of the FEM, has been presented. The approach adopted herein consists of using OOP as a **tool**, which plays a crucial role in the application of the FEM to the

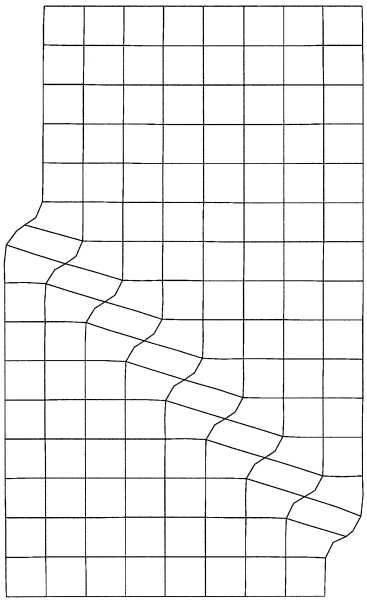

Fig. 16. Classical: 8 × 16.

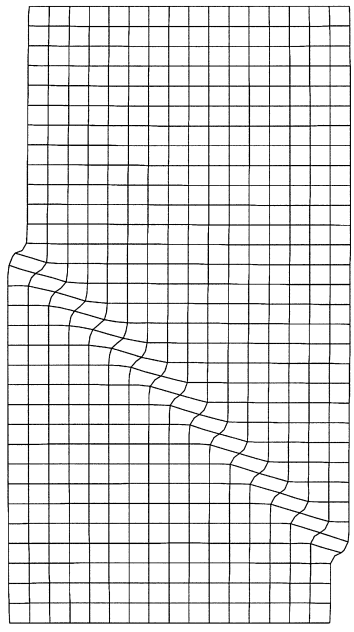

Fig. 17. Classical: 16 × 32.

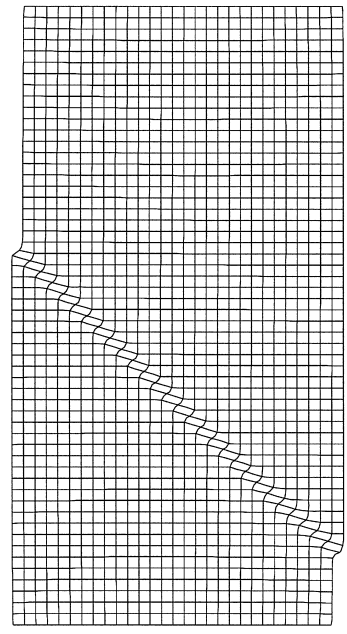

Fig. 18. Classical: 28 × 56.

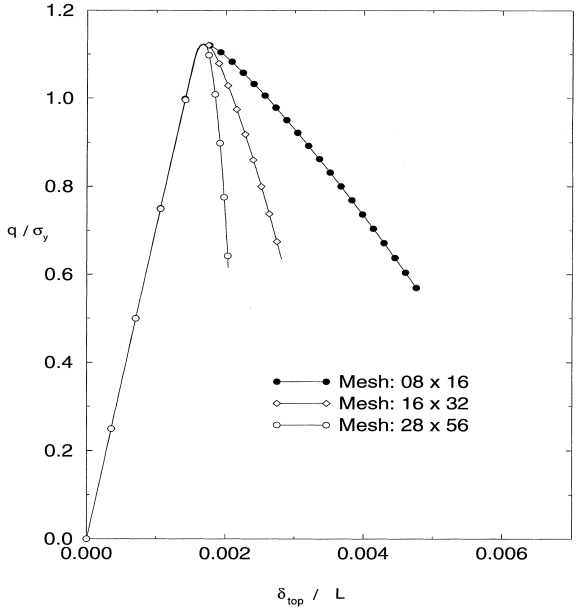

Fig. 19. Equilibrium trajectories for the classical continuum. The inclination of the post-peak branch is different for each mesh.

Table 4. Numerical results for the Cosserat continuum[a]

Discretization	#DOF	#Steps	#Iterations(max)
08×16	1617	320	26
16×32	6305	224	22
28×56	19097	184	20

[a]SUN SunSPARCstation 20 with 96 Mbytes of memory running *SunOS 4.1.4*

solution of engineering mechanics problems. Specific applications involve a geometrically nonlinear beam model and the elastoplastic Cosserat continuum. Examples have been provided, which illustrate the main features of the computational system.

The overall class organization for nonlinear mechanics modeling has been presented. All analyses rely on a general control class where several classical (e.g. Newton–Raphson) and modern (e.g. arc-length) nonlinear solution schemes are available.

Potential extensions of this work include development of a parallel computing object oriented environment, and extension of the present ideas to other numerical methods, such as the Boundary Element Method (BEM). It would be interesting to integrate the parallel computing techniques presented by Hsieh et al. [49,50] with the present ideas on finite element analysis using OOP, and additional concepts on distributed data objects and tasks. With respect to extension of the OOP framework to other numerical methods, it is worth investigating applications of OOP to novel numerical techniques such as the symmetric-Galerkin BEM [51, 52]. These topics are currently under investigation by the authors.

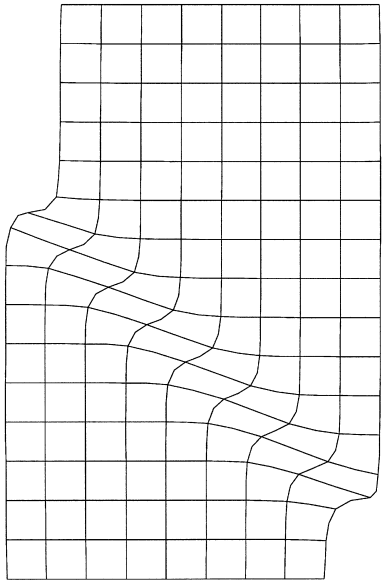

Fig. 20. Micropolar: 8 × 16.

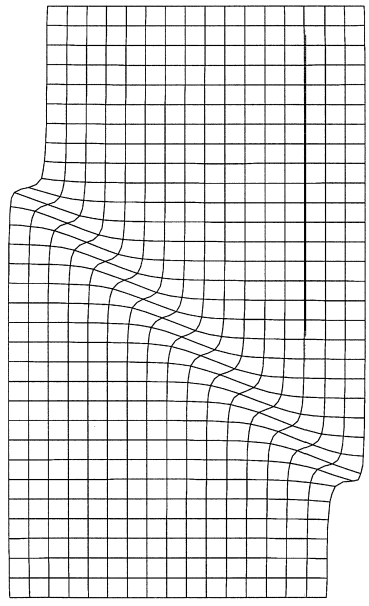

Fig. 21. Micropolar: 16 × 32.

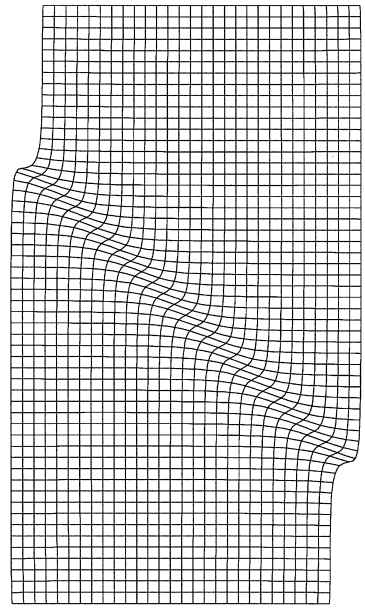

Fig. 22. Micropolar: 28 × 56.

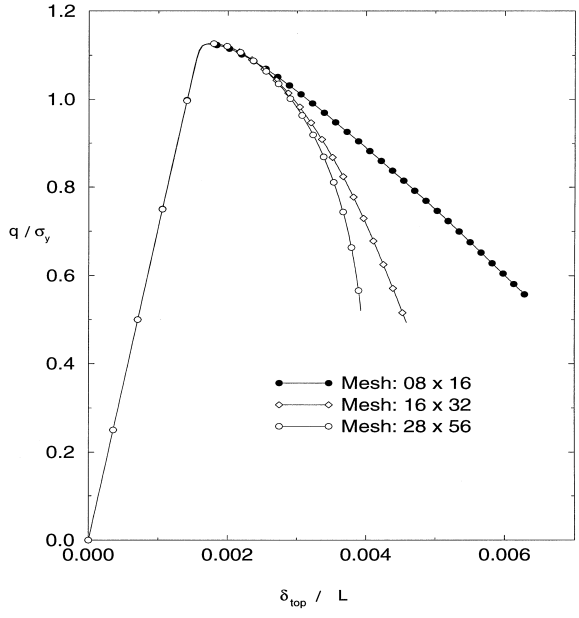

Fig. 23. Equilibrium trajectories for the Cosserat continuum. The inclination of the post-peak branch shows reduced sensitivity to mesh refinement, especially for the finer meshes.

Acknowledgements

GHP thanks the United States National Science Foundation (NSF) through grant No. CMS-9713798 (Mechanics and Materials Program). IFMM acknowledges the financial support provided by FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro), and RRS acknowledges CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), which are Brazilian agencies for research and development. Finally, all the authors would like to thank two anonymous reviewers for valuable suggestions which contributed to substantial improvements to this paper.

References

1. Roux, S. (1966) Continuum and discrete description of elasticity and other rheological behavior. In: Hermann, H.J.; Roux, S. (eds), Statistical Models for the Fracture of Disordered Media, North-Holland, pp 87–114
2. Dahl, O.J.; Nygaard, K. (1966) SIMULA – An Algol-based simulation language. Communications of the ACM 9, 671–678
3. Fenves, G.L. (1990) Object-oriented programming for engineering software development. Engineering with Computers 6, 1–15
4. Goldberg, A.; Robson, D. (1983) Smalltalk-80: The Language and its Implementation. Addison-Wesley, Reading, MA
5. Stefik, M.; Bobrow, D.G. (1985) Object oriented programming: themes and variations. AI Magazine 40–62
6. Weinreb, D.; Moon, D. (1981) Lisp machine manual. Symbolics Inc.
7. Cox, B.J. (1986) Object Oriented Programming. Addison-Wesley, Reading, MA
8. Stroustrup, B. (1997) The C++ Programming Language, third edition. Addison-Wesley, Reading, MA
9. Zimmermman, Th.; Dubois-Pèlerin, Y.; Bomme, P. (1992) Object-oriented finite element programming: I. Governing principles. Computer Methods in Applied Mechanics and Engineering 98, 291–303
10. Rehak, D.R.; Baugh Jr., J.W. (1989) Alternative pro-

gramming techniques for finite element program development. IABSE Colloquium on Expert Systems in Civil Engineering, Bergamo, Italy
11. Forde, B.W.R; Foschi, R.O.; Stiemer, S.F. (1990) Object-oriented finite element analysis: Computers and Structures 34(3), 355–374
12. Dubois-Pèlerin, Y.; Zimmermman, Th.; Bomme, P. (1992) Object-oriented finite element programming: II. A prototype program in smalltalk. Computer Methods in Applied Mechanics and Engineering 98, 361–397
13. Dubois-Pèlerin, Y.; Zimmermman, Th. (1993) Object-oriented finite element programming: III. An efficient implementation in C++. Computer Methods in Applied Mechanics and Engineering 108, 165–183
14. Zimmermman, Th.; Eyheramendy, D. (1996) Object-oriented finite elements I. Principles of symbolic derivations and automatic programming. Computer Methods in Applied Mechanics and Engineering 132(3–4), 259–276
15. Eyheramendy, D.; Zimmermman, Th. (1996) Object-oriented finite elements II. A symbolic environment for automatic programming. Computer Methods in Applied Mechanics and Engineering 132(3–4), 277–304
16. Eyheramendy, D.; Zimmermman, Th. (1998) Object-oriented finite elements III. Theory and application of automatic programming. Computer Methods in Applied Mechanics and Engineering 154 (1–2), 41–68
17. Menétrey, P.; Zimmermann, Th. (1993) Object-oriented nonlinear finite element analysis: application to J2 plasticity. Computers and Structures 49(5), 767–777
18. Mackie, R.I. (1992) Object oriented programming of the finite element method. International Journal for Numerical Methods in Engineering 35, 425–436
19. Alves Filho, J.S.R; Devloo, P.R.B. (1991) Object oriented programming in scientific computations: The beginning of a new era. Engineering Computations 8, 81–87
20. Raphael, B.; Krishnamoorthy, S. (1993) Automatic finite element development using object oriented techniques. Engineering Computations 10(3), 267–278
21. Bettig, B.P.; Han, R.P.S. (1996) An object-oriented framework for interactive numerical analysis in a graphical user interface environment. International Journal for Numerical Methods in Engineering 39(17), 2945–2971
22. Besson, J.; Foerch, R. (1997) Large scale object-oriented finite element code design. Computer Methods in Applied Mechanics and Engineering 142 (1–2), 165–187
23. Jeremić, B.; Sture, S. (1998) Tensor objects in finite element programming. International Journal for Numerical Methods in Engineering 41(1), 113–126
24. Noronha, M.; Wagner, M.; Wirnitzer, J.; Dumont, N.A.; Gaul, L. (1996) On a robust object-oriented code for the implementation of conventional and hybrid boundary element methods. In SIBRAT (Simpósio Brasileiro sobre Tubulações e Vasos de Pressão), Rio de Janeiro, Brazil, pp 239–241
25. Lage, C. (1998) The application of object-oriented methods to boundary elements. Computer Methods in Applied Mechanics and Engineering 157 (3–4), 205–213
26. Olsson, A. (1998) An object-oriented implementation of structural path following. Computer Methods in Applied Mechanics and Engineering 161 (1–2), 19–47
27. Lafore, R. (1991) Objected-Oriented Programming in TURBO C++. Waite Group Press, A Division of the Waite Group, Inc, first edition
28. Paulino, G.H.; Menezes, I.F.M; Lages, E.N. A unified approach for solving nonlinear finite element systems—Implementation and Applications (to be submitted)
29. Batoz, J.L.; Dhatt, G. (1979) Incremental displacement algorithms for nonlinear problems. International Journal for Numerical Methods in Engineering 14, 1262–1267
30. Yang, Y.B.; Shieh, M.S. (1990) Solution method for nonlinear problems with multiple critical points. AIAA Journal 28 (12), 2110–2116
31. Yang, Y.B.; Kuo, S.R. (1994) Theory & Analysis of Nonlinear Framed Structures. Prentice-Hall, New York
32. Pacoste, C.; Eriksson, A. (1995) Element behaviour in post-critical plane frame analysis. Computer Methods in Applied Mechanics and Engineering 125 (1–4), 319–343
33. Pacoste, C.; Eriksson, A. (1997) Beam elements in instability problems. Computer Methods in Applied Mechanics and Engineering 144 (1–2), 163–197
34. Lee, S-L.; Manuel, F.S.; Rossow, E.C. (1968) Large deflections and stability of elastic frames. Journal of Engineering Mechanics (ASCE) 94(EM2), 521–547
35. Frey, F.; Cescotto, S. (1977) Some new aspects of the incremental total Lagrangian description in nonlinear analysis. International Conference on Finite Elements in Nonlinear Solid and Structural Mechanics, vol 1, Geilo, Norway, pp 323–343
36. Schweizerhof, K.H.; Wriggers, P. (1986) Consistent linearization for path following methods in nonlinear finite element analysis. Computer Methods in Applied Mechanics and Engineering 59, 261–279
37. Simo, J.C.; Vu-Quoc, L. (1986) A three-dimensional finite-strain rod model. Part II: computational aspects. Computer Methods in Applied Mechanics and Engineering 58, 79–116
38. Chen, H.; Blandford, G.E. (1993) Work-increment-control method for nonlinear analysis. International Journal for Numerical Methods in Engineering 36, 909–930
39. Menezes, I.F.M.; Lages, E.N.; Paulino, G.H. (1997) A unified approach for solving nonlinear finite element systems of equations. In M.S. Shephard (ed) Fourth U.S. National Congress on Computational Mechanics, San Francisco, CA, p 138
40. McGuire, W.; Gallagher, R.H.; Ziemian, R.D. (1999) Matrix Structural Analysis. John Wiley, New York, second edition
41. Bazant, Z.P. (1986) Mechanics of distributed cracking. Applied Mechanics Reviews 39(5), 675–705
42. Lages, E.N. (1997) Modeling of strain localization with generalized continuum theories. PhD thesis, Department of Civil Engineering, PUC-Rio, Rio de Janeiro, Brazil, (in Portuguese)
43. de Borst, R.; Sluys, L.J.; Mühlhaus, H-B.; Pamin, J. (1993) Fundamental issues in finite element analyses of localization of deformation. Engineering Computations 10(2), 99–121
44. Iordache, M-M.; Willam, K. (1998) Localized failure

analysis in elastoplastic Cosserat continua. Computer Methods in Applied Mechanics and Engineering 151, 559–586
45. Mori, K.; Shiomi, M.; Osakada, K. (1998) Inclusion of microscopic rotation of powder particles during compaction in finite element method using Cosserat continuum theory. International Journal for Numerical Methods in Engineering 42(5), 847–856
46. Simo, J.C.; Hughes, T.J.R. (1988) Elastoplasticity and Viscoplasticity – Computational Aspects. Stanford University, Department of Applied Mechanics
47. de Borst, R. (1991) Simulation of strain localization: A reappraisal of the Cosserat continuum. Engineering Computations 8, 317–332
48. de Borst, R. (1993) A generalisation of J_2-flow theory for polar continua. International Journal for Numerical Methods in Engineering 103(3), 347–362
49. Hsieh, S-H.; Paulino, G.H.; Abel, J.F. (1995) Recursive spectral algorithms for automatic domain partitioning in parallel finite element analysis. Computer Methods in Applied Mechanics and Engineering 121 (1–4), 137–162
50. Hsieh, S.-H.; Paulino, G.H.; Abel, J.F. (1997) Evaluation of automatic domain partitioning algorithms for parallel finite element analysis. International Journal for Numerical Methods in Engineering 40(6), 1025–1051
51. Gray, L.J.; Paulino, G.H. (1997) Symmetric Galerkin boundary integral formulation for interface and multizone problems. International Journal for Numerical Methods in Engineering 40 (16), 3085–3101
52. Paulino, G.H.; Gray, L.J. Error estimation and adaptivity for the symmetric Galerkin boundary element method. Journal of Engineering Mechanics (ASCE) (in press)