UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERS
DE CAMINS, CANALS I PORTS

DEPARTAMENT DE MATEMÀTICA APLICADA III

ARBITRARY LAGRANGIAN-EULERIAN FORMULATION OF QUASISTATIC NONLINEAR PROBLEMS

by

ANTONIO RODRÍGUEZ FERRAN

ADVISOR: ANTONIO HUERTA CEREZUELA

BARCELONA, JULY 1996

T ROD

UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERS
DE CAMINS, CANALS I PORTS

DEPARTAMENT DE MATEMÀTICA APLICADA III

ARBITRARY LAGRANGIAN-EULERIAN FORMULATION OF QUASISTATIC NONLINEAR PROBLEMS

by

ANTONIO RODRÍGUEZ FERRAN

ADVISOR: ANTONIO HUERTA CEREZUELA

BARCELONA, JULY 1996

A l'Agnès

ACKNOWLEDGEMENTS

I want to express my gratitude and acknowledgement to all the people who, in one way or another, have helped me during this work:

Antonio Huerta, my thesis advisor, for his guidance, for the many hours he dedicated to me and for his permanent encouragement; Manuel Casteleiro, who offered me the chance to join in teaching and research tasks; Pedro Díez, for all the time we spent discussing about our theses; Pep Sarrate and Nuria Montolío, for their continuous help, especially in my relation with computers; Miguel Ángel Bretones, who is always ready to help –no matter how busy he is with his own things–, for his enthusiastic cooperation; Juanjo Egozcue and the people of the Departament de Matemàtica Aplicada III (DMA3) and the Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports (ETSECCPB), for the good athmosphere; Ramon Codina and Carlos García Garino, for fruitful discussions; Jean Donéa, Folco Casadei and Pierre Pegon, from the Joint Research Centre of the European Commission (Ispra), for their most interesting comments and remarks and their accessibility; Alain Millard and Lionel Bohar, from the Commissariat à l'Énergie Atomique (France), for all their help with CASTEM 2000; Raúl Giménez, for an excellent work with the drawings; the ETSECCPB, the DMA3, the Universitat Politècnica de Catalunya, the Human Capital and Mobility ERB-4050-PL-94-1116 and DGICYT PB94-1200, for their financial support.

All my family, for all the support and patience over these years; my parents, for their permanent contribution to my education and well-being; my wife, Agnès, for her cooperation, encouragement, patience and love.

CONTENTS

Chapter 1

INTRODUCTION

1.1 Scope and objectives

Many processes of practical interest in engineering and industry involve large deformations, and a combination of sophisticated nonlinear theories, advanced numerical methods and computer resources are needed for their adequate numerical modelling.

Large deformations can occur during the manufacturing stage. This is the case, for instance, with various techniques for the forming of metals, polymers and other materials (forging, rolling, extrusion, drawing, sheet-metal forming, injection molding, among others), see NUMIFORM (1995). It is also possible that a structure or component undergoes large strains under normal working conditions. Some examples are the structural behaviour of box girder bridges and other relatively slender structures, or the large elastic deformations of rubber or rubber-like materials used in shock-absorbers, seals or vibration-isolation bearings. Finally, large deformations can also be associated to exceptional/accidental loads: impact problems, brittle and ductile fracture, buckling and post-buckling response of flexible structures (Crisfield, 1991).

A good understanding of these large deformations processes is essential in engineering practice. It allows, for instance, to improve the design of metal forming processes, resulting in a higher quality of manufactured parts and a longer working life for the tools;

to assess the functionality of a component or structure; to predict the ultimate load carrying capacity and the post-peak behaviour of structures, thus evaluating structural safety,...

This required knowledge was traditionally acquired through experience, empirical rules and experimental testing. A standard approach for studies of deformation in metal forming processes, for instance, was the use of grid networks affixed to the piece (either on the surface or in the interior, a difficult task that involves cutting the billet in half, affixing the grid network and putting the two parts back together), see Dieter (1976). During the last decades, however, numerical simulation has received increased attention and is nowadays an essential tool for the modelling of complex phenomena in many engineering disciplines (civil, mechanical, aeronautical, chemical,...).

Large deformation processes are geometrically nonlinear by nature, because the snape of the deformed body is unknown a priori. Moreover, many engineering materials (steel, aluminium and other metals, concrete and other geomaterials, ...) show a nonlinear material behaviour under large strains. As a consequence, a geometrically and materially nonlinear problems results. This means that classical linear theories —such as infinitesimal elasticity— are not valid, and that more advanced, **nonlinear theories** are needed. The development of nonlinear theories that account for large strains and increasingly complex nonlinear material responses has been one of the main concerns of **continuum mechanics** in the second half of this century. Two basic issues of nonlinear continuum mechanics are large strain kinematics (Truesdell & Toupin, 1960) and large-strain constitutive theories, which originated as extensions of the small-strain counterparts, see for instance Khan & Huang (1995).

If a nonlinear theory is used, the governing equations of continuum mechanics are nonlinear partial differential equations (PDEs). An analytical solution to these PDEs is typically not available, and an approximate solution must be obtained by means of **numerical methods**. First, a discretization technique, such as finite differences, boundary elements or, most notably, the finite element method (Hughes, 1987; Bathe, 1982; Zienkiewicz & Taylor, 1991) is employed to transform the nonlinear PDEs into nonlinear systems of algebraic equations. An analytical solution to these nonlinear systems is also commonly unavailable, and it must be approximated by means of iterative nonlinear solvers, (Orthega & Rheinboldt, 1970; Dennis & Schnabel, 1983).

The effective implementation, testing and application of these numerical algorithms relies heavily in the use of appropriate **computer resources**. The growing complexity of the numerical simulations performed (from 2-D to 3-D; from tens or hundreds of degrees of freedom to tens of thousands, or even several millions;...) has been parallel to the rapid development in computing power, both in terms of hardware (faster, cheaper computers) and software (better, more robust codes).

In fact, the succesful numerical simulation of a large deformation process is only possible with the adequate combination of **nonlinear continuum mechanics, numerical methods** and **computer science**, which can be regarded as the three basic pillars of **nonlinear computational mechanics.** As remarked by Oden (1970), *"In the middle ground* [between computational sciences and computational mechanics] *lies a fertile and potentially important field: numerical analysis of nonlinear continua. It represents a marriage of modern theories of continuous media and modern methods of numerical analysis, so that, with the aid of electronic computation, quantitative information on the nonlinear behavior of solid and structures can be obtained."* (Nonlinear) computational mechanics is clearly an interdisciplinary field which involves engineers, mathematicians and physicists, among others.

In many large deformation processes in solid mechanics, inertia forces are negligible when compared to the other forces acting on the body, and the process is called **quasistatic.** This is the situation, for example, with some forming processes (NUMIFORM, 1995). Although accelerations might not be small in absolute terms, internal forces associated to deformation are much larger then inertia forces, and the latter can be disregarded.

These quasistatic processes involve large strains and large boundary motion, so an adaptive technique is needed. In this work, the arbitrary Lagrangian-Eulerian (ALE) formulation is employed. The ALE formulation originated in fluid dynamics for transient problems, see Donéa (1983). The basic objective of this work is to offer a **clear adaptation of the ALE formulation to quasistatic problems**, and to present a

methodology for ALE quasistatic analysis.

Many approaches can be found in the literature for the statement and numerical time-integration of large strain constitutive equations. There is little information, however, on the **accuracy** of the numerical algorithms. Another goal of this work is to present **a general strategy for the a-priori analysis of the accuracy of stress update algorithms.**

Two issues which are basic in nonlinear computational mechanics in general, and in ALE quasistatic analysis in particular, are the choice of a programming environment and the solution of nonlinear systems of algebraic equations.

An object-oriented code (CASTEM 2000) has been employed as the basic research tool in this work, (Verpeaux & Millard, 1988). In these codes, information is stored and manipulated as objects which directly represent the entities required in a finite element computation. Throughout this work, it will be attempted **to illustrate the suitability of object-oriented codes for research tasks.**

CASTEM 2000 uses the Lagrange-multiplier technique to impose linear constraints (associated to boundary conditions) to the nonlinear system of equilibrium equations. This technique, which is not standard in nonlinear computational mechanics, see Bathe (1982) and Crisfield (1991), is well adapted to the object-oriented approach. In such a situation, another goal of this work is **to adapt nonlinear equation solvers to the object-oriented, Lagrange-multiplier context.**

1.2 Contents

Some basic notions about object-oriented codes in general, and CASTEM 2000 in particular, can be found in Chapter 2. The applicability of object-oriented codes for both real engineering computations and research tasks is illustrated with various examples.

The adaptation of the most widely used nonlinear solvers (Newton-Raphson methods, Quasi-Newton methods, Secant-Newton methods) to CASTEM 2000 (that is, to an object-oriented code and Lagrange multipliers) is presented in Chapter 3. Three topics are covered: algorithmic development, convergence analysis and numerical implementation. Some tests are employed to assess the effectiveness and relative performance of the different algorithms.

Two simple stress update algorithms for large strains are discussed in Chapter 4. The two algorithms, originally developed in a fixed Cartesian frame (Bathe *et al.*, 1975; Pinsky, *et al.*, 1983), are presented here with a convected frame formalism. The use of convected frames enables a unified presentation of both algorithms and, more importantly, the development of a general strategy for the a-priori accuracy analysis of algorithms for the numerical time-integration of nonlinear constitutive equations. Some implementation aspects are also discussed.

Chapter 5 is dedicated to the ALE analysis of quasistatic processes. It has been attempted to offer a unified presentation of the ALE formulation of quasistatic and transient nonlinear solid mechanics. The nonlinear systems of equations arising from the finite element discretization of the momentum balance are solved with the methods of Chapter 3. The material terms in the ALE constitutive equation are handled with the stress update algorithms of Chapter 4. Various explicit algorithms are employed for the treatment of the convective terms in the ALE constitutive equation.

It can be seen from this outline that the ALE analysis of quasistatic problems has served as motivation for research in more fundamental areas of numerical analysis and computational mechanics, such as the solution of nonlinear systems of equations and the time-integration of nonlinear constitutive equations. In fact, some of the results of this work apply to other problems (apart from ALE quasistatic analysis), as illustrated with the numerical examples at the end of each Chapter. In exchange for this global approach to the problem, some relevant issues for more realistic simulations (a general ALE remeshing algorithm, friction and lubrification in the piece-tool interface,...) are not addressed here.

Due to the various topics covered, there is a state-of-the-arf review within each Chapter, instead of a general overview in a separate Chapter. For the same reason, some concluding remarks are made at the end of each Chapter. These conclusions are summarized in Chapter 6, which also outlines the future research.

Chapter 2

AN INTRODUCTION TO A USEFUL TOOL FOR RESEARCH: OBJECT-ORIENTED CODES

2.1 Preliminaries

The Finite Element Method (FEM) is nowadays regarded as one of the most powerful and versatile techniques for the numerical solution of partial differential equations, (Bathe, 1982; Hughes, 1987; Zienkiewicz & Taylor, 1991; Zienkiewicz & Morgan, 1983; Oñate, 1991). It is a basic tool for the numerical simulation of a vast range of phenomena in the fields of science and engineering.

A global finite element computation can be regarded as a sequence of elementary processes –which are relatively simple but, at the same time, general enough so as to be useful in different situations–. The various stages in a computation (for a standard elastic analysis: input, mesh generation, evaluation of elemental stiffness matrices and assembly of the global stiffness matrix, evaluation of the load vector, solution of the linear system of equations, postprocess) correspond to one or several of these elementary processes.

Conventional codes are designed to solve a certain range of problems (2-D elastic analysis, for instance) and it is sometimes cumbersome to modify them to extend their applicability. The main cause, (Dubois-Pèlerin *et al.*, 1992; Miller 1991) is that they are **function-oriented** (or procedure-oriented). In conventional codes, the elementary processes are carried out by subroutines written in a "classical" programming language: C, Pascal or, in most cases, FORTRAN. Each of these subroutines performs a given function. Emphasis is put on functions, and the variables (i.e. the data) play a secondary role. In fact, a conventional code is basically a sequence of calls to functions that perform operations on the variables.

The use of subroutines allows for a certain modularity of the code, in exchange for a careful definition of information transfer between the different modules. Since the code is organized as a sequence of functions, it is typically not easy in large codes to keep track of a variable.

As remarked by Dubois-Pèlerin *et al.* (1992), maintenance and extension of large function-oriented codes is a very difficult task, because of two major drawbacks: sequenciality of actions and weak data types (or data structures).

Sequenciality of actions. A conventional code has a predefined information flow pattern –represented by the "main" module–, which is not easy to change. Each function (i.e. each subroutine) is designed to operate at a given point in the flowchart. Typically it assumes a certain current state of the problem variables, so it cannot be moved to another point in the flowchart.

Weak data types. A conventional code uses very rudimentary variables: scalars, arrays,... As a consequence, all the required quantities in a finite element analysis must be translated into such variables. The finite element mesh, for instance, is represented by a matrix of nodal coordinates and a connectivity matrix (which, in fact, are handled as arrays of numbers, not as true matrices). This translation process is often a tedious and error-prone task. It forces the programmer to concentrate on the technical details, rather than in the basic steps required to perform the analysis. Moreover, these rudimentary variables have very limited information and capabilities.

As a result, according to Dubois-Pèlerin *et al.* (1992), *"a procedure handles data without really controlling them. The control is external"*, thus limiting procedure-oriented codes.

In an attempt to overcome the limitations of conventional codes, the first finite element object-oriented codes (OOC) have appeared recently, (Miller, 1991; Dubois-Pèlerin *et al.*, 1992; Mackie, 1992; Galindo, 1994). The aim is a greater flexibility of the code, both from the user's and the programmer's viewpoint: the range of applicability should be as large as possible, and further modification and improvement of the code should be a relatively simple task.

The essential feature of the object-oriented codes is the storage and manipulation of information as **objects** which are of a more complex nature than variables in function-oriented codes. The basic idea is that higher complexity means more information, so the manipulation of these objects is an easier task than the manipulation of variables in a conventional code.

In a finite element object-oriented code, these objects directly represent the entities needed to solve the problem. It is no longer necessary for the user to translate them into more rudimentary variables. In computational mechanics, the required objects are nodes, meshes, nodal fields (displacements or loads), fields-by-element (strains or stresses), stiffness and mass matrices,...In an OOC, for instance, the mesh is an object in itself, which can be manipulated as an entity: it may be plotted, deformed by a displacement field, support a certain finite element formulation (plane stress, plane strain, plate, shell,...).

2.2 Basic features of object-oriented codes

In this Section, some basic concepts about object-oriented codes are briefly reviewed. More information can be found in Miller (1991), Dubois-Pèlerin *et al.* (1992), Mackie (1992), Galindo (1994) and Bretones *et al.* (1995).

An **object** is characterized by a set of data (the object's attributes) and a set of associated operations (the object's methods), see Dubois-Pèlerin *et al.* (1992). For instance, a `Rectangle` object can be defined with the attributes `WIDTH`, `HEIGHT`, `X-COORDINATE OF CENTRE`, `Y-COORDINATE OF CENTRE` and with methods for initializing it, computing its area or translating it (Galindo, 1994).

In an OOC, an object communicates with other objects by sending messages to them. When an object receives a message, it performs the requested operation (which must be one of the object's methods). This operation is carried out by accessing the object's attributes. In this manner, all the information about an object is enclosed within the object itself, and can only be reached by sending a message to the object. This feature is called **data encapsulation** (Dubois-Pèlerin *et al.*, 1992; Miller, 1991; Galindo, 1994), and ensures that information is not scattered all over the code in various arrays.

Objects of the same kind are grouped into **classes**. Each object is then an instance (i.e. a "materialization") of its class. The classes are organized in a class hierarchy, which provides an inheritance mechanism. **Inheritance** allows the sharing of attributes and methods that are common to more than one class, thus precluding unnecessary code duplications. For example, the classes of symmetric, skew-symmetric, banded and diagonal matrices can all be defined as subclasses of the superclass of matrices. The attributes and methods that are general to all matrices are directly inherited by the subclasses, and only the new attributes and methods specific to each subclass must be defined.

Another important feature of OOC is **polymorphism**, which allows objects of different classes to react in a different, appropriate manner to the same message. For instance, the message "compute your own area" can be sent to objects of class `Rectangle`, `Circle` or `Triangle`, and is adequately handled by each of these classes.

The basic features of an object-oriented code are summarized in Box 2.1.

- **Object:** a set of **attributes** and **methods.**
- **Class:** group of objects of the same kind.
- Objects intercommunicate by sending **messages** to each other.
- **Data encapsulation:** all infomation is enclosed within the object. Prevents scattering of information.
- **Inheritance:** common attributes and methods can be shared. Avoids code duplications.
- **Polymorphism:** the same message is correctly understood by objects of different classes. Provides flexibility.

Box 2.1 Basic features of object-oriented codes

2.3 An object-oriented code: CASTEM 2000

An object-oriented code, CASTEM 2000, is used as the basic research tool in this work. This code has been developed by the Laboratoire d'Analyse Mécanique des Structures (L.A.M.S.) of the Commissariat à l'Énergie Atomique (France) and is basically dedicated to the thermo-mechanical analysis of solids and structures, see Verpeaux & Millard (1988) or Bretones *et al.* (1995).

In CASTEM 2000, elementary processes of a finite element computation are represented by means of **operators**, which manipulate one or more existing objects to create a new object, see L.A.M.S. (1988). The task of sending messages to the objects is carried out by these operators, not by the objects themselves. The code may then be regarded as a programming environment, where sentences of GIBIANE, an internal interactive object-oriented language, are employed to carry out the analysis. These GIBIANE sentences have the form

$$\texttt{new\_object} = \texttt{OPERATOR}\quad \texttt{old\_object\_1}\quad \texttt{old\_object\_2...;} \tag{2.1}$$

For instance, in the GIBIANE sentence

$$\texttt{line} = \texttt{DROITE} \quad \texttt{10} \quad \texttt{point1} \quad \texttt{point2;} \tag{2.2}$$

two existing objects of class `Point` are manipulated with operator `DROITE` to create a new object of class `Mesh`, made up of 10 elements.

More than 400 operators are currently available. Some fundamental operators are `SURFACE` (to generate 2-D meshes), `RIGIDITE` (to build stiffness matrices) and `RESOUDRE` (to solve linear systems of equations). As an example of how a problem is solved with CASTEM 2000, the GIBIANE file for a simple 2-D elastic analysis is shown in Box 2.2.

An object is defined in CASTEM 2000 by its name, its type (or class) and its value. The name of the object is chosen by the user and allows its manipulation with the GIBIANE language. The type or class of an object refers to a certain data structure. Some of the available classes of objects are: `Integer`, `Real`, `Point`, `Mesh`, `Model`, `Material`, `Stiffness matrix`, `Nodal field`, `Field-by-element`, ... The value of an object enables the access to the information it contains.

Thus, elementary processes in a FEM analysis are represented by operators, and the required quantities are represented by objects. In such an environment, the user/developer can concentrate on **what** to do (i.e. which steps and quantities are needed for the analysis), rather than worry about **how** to do it.

A very useful facility provided by CASTEM 2000 is that a set of sentences of the type (2.1) can be defined as a metaoperator (Soria & Pegon, 1992; Bretones *et al.*, 1995), which is subsequently treated as a new operator.

2.4 CASTEM 2000, a useful tool for research

All these features make CASTEM 2000 a valid tool for research, teaching and engineering applications. It can be employed for standard finite element analysis, by using the existing objects and operators, as done in the example in Box 2.2. Such simple examples can also be used to teach the FEM. The object-oriented environment allows the student to focus on the basic concepts.

The code can also be employed for real engineering analysis. To illustrate this point, the structural analysis of a bridge is shown in Subsection 3.5.5. Various firms currently use the commercial version of the code.

More relevant to this work is the fact that the code is also a valid tool to implement and test new developments, by defining new metaoperators. In this work, new metaoperators have been written for the solution of nonlinear systems of equations (Chapter 3), for the time-integration of the constitutive equation in large strain solid mechanics (Chapter 4) and for ALE quasistatic analysis –remeshing and stress transport– (Chapter 5).

Various workteams currently cooperate in the extension of CASTEM 2000 to solve new problems (L.A.M.S. ; Laboratoire de Mécanique et Technologie, Cachan (France); Institute for Systems and Safety of the Joint Research Centre of the European Commission, Ispra (Italy); Departament de Matemàtica Aplicada III, Universitat Politècnica de Catalunya (Spain)), see for instance Pegon (1993) and Bretones *et al.* (1995). The independency between operators and the sharing of new metaoperators by the various groups greatly facilitates this common task. Extendibility is enabled by the object-oriented nature of the code. For instance, it has been recently used to simulate sound propagation by solving the Helmholtz equation, (Ortiz, 1993), to test artificial boundary conditions for semiinfinite domains (Ausaverri, 1993), and to study the formation of plastic hinges in frames (Bretones, 1993).

```
*****************************************************************
* PLANE STRESS ANALYSIS OF A STEEL SHEET WITH
* A CLAMPED SIDE AND SUBJECTED TO A CONCENTRATED LOAD
*****************************************************************
*
* Setting of general options: two-dimensional problem; plane stress
* analysis; bilinear quadrilateral elements; graphics in X-Windows
*
OPTION DIMENSION 2 MODE PLAN CONTRAINTE ELEMENT QUA4 TRACE X ;
*
* Mesh generation
* 1) Definition of the vertexes p1, p2, p3 and p4 of the sheet
* 2) Definition, via operator DROITE, of lines l1, l2, l3 and l4
* 3) Mesh generation with operator DALLER
* 4) Mesh display with operator TRACE
*
p1 = 0.  0.  ; p2 = 1.  0.  ; p3 = 1.  0.5 ; p4 = 0.  0.5 ;
l1 = p1 DROITE 5 p2 ; l2 = p2 DROITE 5 p3 ;
l3 = p3 DROITE 5 p4 ; l4 = p4 DROITE 5 p1 ;
mesh1 = DALLER l1 l2 l3 l4 ;
TITRE 'Mesh' ; TRACER mesh1 ;
*
* Model and material
* 1) Definition of mechanical model with operator MODE
* 2) Definition of material parameters with operator MATE
*
mod1 = MODE mesh1 MECANIQUE ELASTIQUE ISOTROPE ;
mat1 = MATE mod1 YOUNG 2.1e7 NU 0.3 RHO 7.85 ;
```

Notes:

- Lines beginning with * are comments
- For code lines, uppercase letters have been employed for operators and lowercase letters for objects

Box 2.2 GIBIANE file for a 2-D elastic analysis

```
* Boundary conditions
* 1) Concentrated load in point p3 via operator FORCE
* 2) Restrained displacements in side l1 via operator BLOQUER[1]
*
load3 = FORCE FX 5000.  FY 3000.  p3 ;
bc1 = BLOQUER DEPLACEMENT l1 ;
*
* Stiffness matrix
* 1) Internal stiffness matrix with operator RIGIDITE
* 2) Assembly of internal and boundary stiffnesses with operator ET[1]
*
stiff1 = RIGIDITE mod1 mat1 ;
stiff2 = stiff1 ET bc1 ;
*
* Solver
* 1) Solution of linear system with operator RESOUDRE
*
displ = RESOUDRE stiff2 load3 ;
*
* Postprocess and results
* 1) Stress with operator SIGMA
* 2) Von Mises stress with operator VMISES
* 3) Display with operator TRACE
*
tens1 = SIGMA mod1 mat1 despl ;
vmis1 = VMISES mod1 tens1 ;
TITRE 'Von Mises stress ' ; TRACE sup1 mod1 vmis1 ;
```

Box 2.2 GIBIANE file for a 2-D elastic analysis (continued)

[1] CASTEM 2000 employs the Lagrange-multiplier technique to impose the boundary conditions, see Chapter 3.

Chapter 3

NONLINEAR EQUATION SOLVERS AND LAGRANGE MULTIPLIERS

3.1 Introduction

If the Finite Element Method (or any other discretization technique) is employed, a quasistatic problem in nonlinear mechanics is transformed into a nonlinear system of equations, complemented with some constraints that represent the boundary conditions. Many useful boundary conditions have the form of linear constraints on the system's unknowns: restrained or prescribed displacements, skewed supports, symmetric or cyclic boundary conditions, rigid inclusions,...

The nonlinear system of equations, which expresses static equilibrium, is typically solved with an incremental-iterative strategy (Bathe, 1982; Crisfield, 1991), based on the linearization of the nonlinear system. Various algorithms can be found in the literature to impose the linear constraints into the linear system to be solved at each iteration. For one-point constraints, involving only one degree of freedom (d.o.f.), the implementation is often performed directly by adjusting rows and columns, see for instance Crisfield (1991). This procedure, however, is not readily extended to the general case of multiple multipoint constraints, with several d.o.f. involved in each constraint (and, eventually, some d.o.f. shared by more than one constraint). On the other hand, the Lagrange-multiplier technique (Zienkiewicz & Taylor, 1991; Barlow, 1982) is general, but has the drawback that the dimension of the system of equations is increased.

To remedy this situation, more advanced algorithms have appeared recently (Barlow, 1982; Shephard, 1984; Webb, 1990; Schreyer & Parsons, 1995). They are based on transforming the original system by imposing the constraints. The common goal is to obtain symmetric and positive definite (SPD) stiffness matrices (when the nature of the problem allows, of course) and to maintain or reduce by the number of constraints the dimension of the problem.

As commented in Chapter 2, an object-oriented code is employed as the programming environment in this work. One of the requirements of such codes is that the various objects (which directly represent the quantities needed for the analysis) must be independent from one another. In such a context, the Lagrange-multiplier technique is a natural way to impose linear constraints. Indeed, as shown in Section 3.3, the stiffness and the linear constraints are then represented by two distinct objects: the stiffness matrix $\boldsymbol{K}$ and the constraint matrix $\boldsymbol{A}$. These two objects are combined to form an enlarged matrix, but the original stiffness matrix $\boldsymbol{K}$ is not transformed in any way, contrary to the previously cited methods. If different sets of constraints must be implemented for the same problem, it suffices to build the corresponding constraint matrices $\boldsymbol{A}$, $\boldsymbol{A}'$, $\boldsymbol{A}''$,... In view of this situation, and in spite of the increase in the dimension of the problem, the Lagrange-multiplier technique is chosen here to impose linear constraints.

The goal of this Chapter is to show how various nonlinear equation solvers can be adapted to handle linear constraints imposed via Lagrange multipliers (Vila *et al.*, 1995; Rodríguez-Ferran *et al.*, 1996). It is shown that the adaptation is rather straightforward for Newton-Raphson methods (Section 3.3), but more involved for Quasi-Newton and Secant-Newton methods (Section 3.4). Emphasis is put on some issues which are specific to the Lagrange-multiplier context. A classical presentation of the different solvers can be found in Appendix A.

3.2 Preliminaries

The notation of nonlinear mechanics will be employed here to state the problem. The basic ideas and the algorithms, however, can be extended to other nonlinear problems.

After finite element discretization, (Zienkiewicz & Taylor, 1991), static equilibrium in nonlinear mechanics is typically expressed as a nonlinear system of n equations complemented with m linear constraints

$$\boldsymbol{r}(\boldsymbol{u}) = \boldsymbol{f}_{\text{int}}(\boldsymbol{u}) - \boldsymbol{f}_{\text{ext}} + \boldsymbol{f}_{\text{reac}} = \boldsymbol{0} \tag{3.1a}$$

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} \tag{3.1b}$$

In the equilibrium equation (3.1a), $\boldsymbol{u}$ is the vector of nodal displacements, $\boldsymbol{f}_{\text{int}}(\boldsymbol{u})$ is the vector of internal forces, which depends nonlinearly on displacements, $\boldsymbol{f}_{\text{ext}}$ is the vector of applied external forces, $\boldsymbol{f}_{\text{reac}}$ is the vector of reaction forces and $\boldsymbol{r}(\boldsymbol{u})$ is the vector of residual (off-balance) forces, which are null if equilibrium is verified. For simplicity, it is assumed that the external loads $\boldsymbol{f}_{\text{ext}}$ are independent from displacements. Equation (3.1b) represents the linear constraints, with $\boldsymbol{A}$ an $m \times n$ rectangular matrix and $\boldsymbol{b}$ a vector with the prescribed values of the constraints. It allows to represent the most frequent boundary conditions: prescribed displacements (that is, $u_i = b_i$ for $i = 1 \dots m$), skewed supports, rigid inclusions, periodic or symmetric boundary conditions (see Section 3.5) or, more generally, any equality constraint between the degrees of freedom of the problem. The reaction forces $\boldsymbol{f}_{\text{reac}}$ in Eq. (3.1a) are necessary so that the constraints (3.1b) are verified.

Remark 3.1: In Eq. (3.1a), $\boldsymbol{f}_{\text{ext}}$ represents only the **known** applied loads, so the reaction forces $\boldsymbol{f}_{\text{reac}}$ must be written explicitly in the equilibrium equation. A different and common choice, (Bathe, 1982; Crisfield, 1991), however, is to write simply

$$\boldsymbol{r}(\boldsymbol{u}) = \boldsymbol{f}_{\text{int}}(\boldsymbol{u}) - \boldsymbol{f}_{\text{ext}} = \boldsymbol{0}, \tag{3.2}$$

and assume that some components of $\boldsymbol{f}_{\text{ext}}$ are unknown a priori, because reactions are included in $\boldsymbol{f}_{\text{ext}}$.

A standard approach to solve Eqs. (3.1) is to employ the constraints (3.1b) to reduce the order of the problem, from n to $n-m$. This reduction is typically performed by transforming the tangent stiffness matrix and residual vector that arise from the linearization of Eq. (3.1a), (Bathe, 1982; Shephard, 1984; Crisfield, 1991).

Another possible approach is the Lagrange-multiplier technique, (Barlow, 1982; Zienkiewicz & Taylor, 1991). In this case, the reaction forces are written as

$$f_{\text{reac}} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}, \tag{3.3}$$

where T means transpose, $\boldsymbol{\lambda}$ is a vector of m Lagrange multipliers (one per constraint), and Eqs. (3.1) are rewritten as

$$\left.\begin{array}{rcl} \boldsymbol{r}_1(\boldsymbol{u},\boldsymbol{\lambda}) & = & f_{\text{int}}(\boldsymbol{u}) - f_{\text{ext}} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{0} \\ \boldsymbol{r}_2(\boldsymbol{u}) & = & \boldsymbol{A}\boldsymbol{u} - \boldsymbol{b} = \boldsymbol{0} \end{array}\right\}. \tag{3.4}$$

In Eq. (3.4), $\boldsymbol{r}_1$ measures equilibrium, and $\boldsymbol{r}_2$ measures compliance of the constraints. These constraints are added to the nonlinear equations, and the Lagrange multipliers $\boldsymbol{\lambda}$ are added to the displacements $\boldsymbol{u}$ as unknowns. Equation (3.4) is then an enlarged system of order $n+m$,

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}, \tag{3.5}$$

where the notation $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{\lambda})$ and $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2)$ has been used. Thus, both displacements and Lagrange multipliers are regarded as problem unknowns.

Remark 3.2: When the Lagrange-multiplier technique is employed, the dimension of the problem is increased (from n to $n+m$), Eq. (3.5). In exchange for this disadvantage, Lagrange multipliers allow to implement general linear constraints, Eq. (3.1b), in a direct way. Note that handling linear constraints is very simple for the simple case of m prescribed d.o.f. but becomes rather more involved for multipoint constraints, see Bathe (1982). In fact, the imposition of general constraint relationships in finite element analyses is a topic of current research, see Barlow (1982), Shephard (1984) Webb (1990), Schreyer & Parsons (1995).

The basic goal of this Chapter is to show how various nonlinear solvers can be adapted to solve Eq. (3.5) in an efficient way. The key idea is exploiting the partial linearity of Eq. (3.4) ($\boldsymbol{r}_1$ is linear with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{r}_2$ is linear with respect to $\boldsymbol{u}$). Thus, whenever possible, the algorithms of the next Sections seek to work only with the displacements $\boldsymbol{u}$, which are the fundamental unknowns, and not with the enlarged vector $\boldsymbol{x}$.

3.3 Newton-Raphson methods

The nonlinear system (3.1a) is typically solved with an incremental-iterative strategy, (Bathe, 1982; Crisfield, 1991; Zienkiewicz & Taylor, 1991). The displacements $\boldsymbol{u}$ are updated according to

$$ {}^{n}\boldsymbol{u}^{k+1} = {}^{n}\boldsymbol{u}^{k} + \delta\boldsymbol{u}^{k+1}, \tag{3.6} $$

where n denotes instant t_n and k is the iteration counter. If the full Newton-Raphson method is employed, the iterative correction $\delta\boldsymbol{u}^{k+1}$ is computed as

$$ \boldsymbol{K}({}^{n}\boldsymbol{u}^{k})\delta\boldsymbol{u}^{k+1} = -\boldsymbol{r}({}^{n}\boldsymbol{u}^{k}), \tag{3.7} $$

where $\boldsymbol{K}(\boldsymbol{u}) = \partial\boldsymbol{r}/\partial\boldsymbol{u} = \partial\boldsymbol{f}_{\text{int}}/\partial\boldsymbol{u}$ is the tangent stiffness matrix. This stiffness matrix is singular, because constraints are not accounted for and rigid motions are not precluded. Various methods have been devised to impose the constraints (3.1b) into the linear system (3.7), see Barlow (1982), Shephard (1984), Schreyer & Parsons (1995). One common approach is to employ the constraints to write m dependent (or slave) d.o.f. in terms of various independent (or master) d.o.f. The vector of displacements is then split into three subvectors: slave d.o.f., master d.o.f. and unconstrained d.o.f.

(i.e. unaffected by the constraints). By eliminating the slave d.o.f., the linear system (3.7) is transformed into a system of order $n - m$

$$ {}^{n}\boldsymbol{K}_{\mathrm{F}}^{k} \delta \boldsymbol{u}_{\mathrm{F}}^{k+1} = - {}^{n}\boldsymbol{r}_{\mathrm{F}}^{k}, \tag{3.8} $$

where the subscript F denotes free d.o.f. (i.e. master and unconstrained d.o.f.) and ${}^{n}\boldsymbol{K}_{\mathrm{F}}^{k}$ is a regular matrix (except at limit points). Various strategies can be employed to produce ${}^{n}\boldsymbol{K}_{\mathrm{F}}^{k}$ and ${}^{n}\boldsymbol{r}_{\mathrm{F}}^{k}$, see for instance Crisfield (1991), Shephard (1984), and will be denoted here with the generic name of **Transformation Method**, to emphasize that the singular linear system (3.7) is transformed into the regular system (3.8).

A very similar approach can be used if the Lagrange-multiplier technique is chosen to impose boundary conditions. In this case, the iterative scheme is

$$ {}^{n}\boldsymbol{x}^{k+1} = {}^{n}\boldsymbol{x}^{k} + \delta \boldsymbol{x}^{k+1}, \tag{3.9} $$

that is, both displacements $\boldsymbol{u}$ (free and dependent d.o.f.) and Lagrange multipliers $\boldsymbol{\lambda}$ are updated. In analogy with Eq. (3.7), the iterative correction $\delta \boldsymbol{x}^{k+1}$ is obtained from

$$ \boldsymbol{J}({}^{n}\boldsymbol{x}^{k}) \delta \boldsymbol{x}^{k+1} = -\boldsymbol{r}({}^{n}\boldsymbol{x}^{k}), \tag{3.10} $$

where $\boldsymbol{J}(\boldsymbol{x}) = \partial \boldsymbol{r} / \partial \boldsymbol{x}$ is the Jacobian matrix. Recalling Eq. (3.4), the Jacobian matrix can be expressed as

$$ \boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \partial \boldsymbol{r}_1 / \partial \boldsymbol{u} & \partial \boldsymbol{r}_1 / \partial \boldsymbol{\lambda} \\ \partial \boldsymbol{r}_2 / \partial \boldsymbol{u} & \partial \boldsymbol{r}_2 / \partial \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}(\boldsymbol{u}) & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}. \tag{3.11} $$

The key point of Eq. (3.11) is that the tangent stiffness matrix $\boldsymbol{K}(\boldsymbol{u})$ is the only variable block-matrix in $\boldsymbol{J}(\boldsymbol{x})$. The rest of the block-matrices are constant, thus reflecting that $\boldsymbol{r}(\boldsymbol{x})$ is partially linear. By employing Eq. (3.11), the linear system (3.10) can be written explicitly as

$$ \begin{aligned} \begin{bmatrix} \boldsymbol{K}({}^{n}\boldsymbol{u}^{k}) & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{u}^{k+1} \\ \delta \boldsymbol{\lambda}^{k+1} \end{Bmatrix} &= \begin{Bmatrix} -\boldsymbol{r}_1({}^{n}\boldsymbol{u}^{k},\ {}^{n}\boldsymbol{\lambda}^{k}) \\ \boldsymbol{0} \end{Bmatrix} \\ &= \begin{Bmatrix} {}^{n}\boldsymbol{f}_{\mathrm{ext}} - {}^{n}\boldsymbol{f}_{\mathrm{int}}^{k} - \boldsymbol{A}^{\mathrm{T}}\, {}^{n}\boldsymbol{\lambda}^{k} \\ \boldsymbol{0} \end{Bmatrix} \end{aligned} \tag{3.12} $$

Equation (3.12) has various interesting features. As commented previously, the tangent stiffness matrix $\boldsymbol{K}$ is singular, because it does not include the constraints. The Jacobian matrix $\boldsymbol{J}$, however, is regular, because the block-matrices $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}$ account for both the linear constraints and the variation in reaction forces respectively. It can be easily checked that, since constraints are linear, the residual $\boldsymbol{r}_2$ is null after the first iteration, so the correction in displacements must satisfy null boundary conditions, $\boldsymbol{A}\ \delta\boldsymbol{u}^{k+1} = \boldsymbol{0}$. As for reaction forces, both the current value at iteration k and the iterative correction appear explicitly in Eq. (3.12). So, contrary to what happened in Eq. (3.8), displacements and reactions are computed simultaneously.

Since the reaction forces are linear on the Lagrange multipliers, Eq. (3.3), the system (3.12) can be rearranged into the more compact form, (Pegon & Anthoine, 1994),

$$\begin{bmatrix} \boldsymbol{K}({}^{n}\boldsymbol{u}^{k}) & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta\boldsymbol{u}^{k+1} \\ \boldsymbol{\lambda}^{k+1} \end{Bmatrix} = \begin{Bmatrix} {}^{n}\boldsymbol{f}_{\mathrm{ext}} - {}^{n}\boldsymbol{f}^{k}_{\mathrm{int}} \\ \boldsymbol{0} \end{Bmatrix}, \qquad (3.13)$$

where total Lagrange multipliers ${}^{n}\boldsymbol{\lambda}^{k+1}$, and not the iterative correction ${}^{n}\delta\boldsymbol{\lambda}^{k+1}$ appear in the vector of unknowns, so there is no need to update them. Note, however, that equilibrium cannot be checked with the RHS in Eq. (3.13), because it does not include reaction forces, so the full residual vector $\boldsymbol{r}_1$ is still needed for convergence control. Moreover, the original formulation (3.12) allows for a more natural implementation of Quasi-Newton and Secant-Newton methods, as shown in the next Section. For these two reasons, Eq. (3.12) is preferred over Eq. (3.13).

To avoid the recomputation of the tangent stiffness matrix required by the full Newton-Raphson method, various modifications have been classically employed, see Crisfield (1991). For instance, in the modified Newton-Raphson method, the stiffness matrix is computed just once per increment; in the initial stress method, the initial elastic matrix is used throughout the analysis. With these modifications, the quadratic rate of convergence of the full Newton-Raphson method is lost and only linear convergence is achieved. These two methods can be adapted to a Lagrange-multiplier context in a straightforward manner, simply by replacing ${}^{n}\boldsymbol{K}^{k}$ in the Jacobian matrix, Eq. (3.12), with ${}^{n}\boldsymbol{K}^{0}$ or ${}^{0}\boldsymbol{K}$ respectively.

In conclusion, if a Newton-Raphson is employed in combination with the Lagrange-multiplier technique, the corresponding stiffness matrix $\boldsymbol{K}$ is enlarged into a Jacobian matrix $\boldsymbol{J}$ which accounts for both the linear constraints and the reaction forces, Eq. (3.12), instead of transformed into the stiffness matrix $\boldsymbol{K}_{\mathrm{F}}$, Eq. (3.8). It must be remarked, however, that these two approaches are **equivalent** and lead to the same solutions. Indeed, manipulating matrix $\boldsymbol{K}$ to get matrix $\boldsymbol{K}_{\mathrm{F}}$ or enlarging it into matrix $\boldsymbol{J}$ are two algebraically equivalent ways of imposing the linear constraints into the nonlinear system.

3.4 The Quasi-Newton family

3.4.1 Introduction

The basic idea of (direct) Quasi-Newton methods is to employ secant approximations to the tangent stiffness matrix $\boldsymbol{K}_{\mathrm{F}}$ which are easy to compute, (Dennis & Moré, 1977). By doing so, the stiffness matrix is neither recomputed from scratch at every iteration (full Newton-Raphson method) nor kept constant for several iterations (modified Newton-Raphson methods), but updated in a simple manner. This results in a good balance between computational cost per iteration and convergence properties: a superlinear rate of convergence is obtained, see Dennis & Moré (1977).

The condition of "secant approximation" to the tangent stifnness matrix is expressed by the so-called Quasi-Newton equation, (Dennis & Schnabel, 1983). If a Transformation Method is employed to impose the boundary conditions, this equation reads

$$\boldsymbol{B}_{\mathrm{F}}^{k}\delta\boldsymbol{u}_{\mathrm{F}}^{k} = \boldsymbol{r}_{\mathrm{F}}^{k} - \boldsymbol{r}_{\mathrm{F}}^{k-1}, \tag{3.14}$$

where $\boldsymbol{B}_{\mathrm{F}}^{k}$ is a secant approximation to the tangent stiffness matrix $\boldsymbol{K}_{\mathrm{F}}^{k}$. Equation (3.14) states that $\boldsymbol{B}_{\mathrm{F}}^{k}$ must "pass through" the last two iterations $(\boldsymbol{u}_{\mathrm{F}}^{k-1}, \boldsymbol{r}_{\mathrm{F}}^{k-1})$ and $(\boldsymbol{u}_{\mathrm{F}}^{k}, \boldsymbol{r}_{\mathrm{F}}^{k})$. From here on, the left superscripts denoting time are dropped to ease the notation.

For implementation purposes, **inverse** Quasi-Newton methods are often preferred, (Matthies & Strang, 1979; Soria & Pegon, 1990). In this case, secant approximations

H^k_{F} to the inverse of the tangent stiffness matrix are employed. An inverse version of the Quasi-Newton equation, Eq. (3.14), is then employed.

Either for direct or inverse Quasi-Newton methods, the Quasi-Newton equation must be complemented with some additional conditions to completely define B^k_{F} (direct QN) or H^k_{F} (inverse QN). These additional requirements are characteristic of every particular Quasi-Newton method. The final output is a simple update scheme of the form

$$B^k_{\mathrm{F}} = B^{k-1}_{\mathrm{F}} + M^k_{\mathrm{F}} \tag{3.15}$$

for direct Quasi-Newton methods or

$$H^k_{\mathrm{F}} = H^{k-1}_{\mathrm{F}} + N^k_{\mathrm{F}} \tag{3.16}$$

for inverse Quasi-Newton methods, where M^k_{F} and N^k_{F} are modification matrices of low rank (typically, one or two). Two common choices for B^0_{F} (or H^0_{F}) are the tangent stiffness matrix at the beginning of the increment or the initial elastic stiffness matrix (or its inverses).

If the Quasi-Newton methods are used in combination with Lagrange multipliers, then the natural approach is to approximate the Jacobian matrix J. The (direct) Quasi-Newton equation is now

$$B^k \delta x^k = r^k - r^{k-1}, \tag{3.17}$$

and states that a secant approximation B^k to J^k must "pass through" the points (u^{k-1}, r^{k-1}) and (u^k, r^k). Again, additional conditions are necessary to define matrix B^k in a unique way.

It must be remarked that, with this approach, matrix B^k is not required a priori to maintain the structure of the Jacobian matrix (i.e. the three constant block-matrices A, A^{T} and $\mathbf{0}$). Two questions then arise: do the resulting matrices B^k automatically have the structure of the Jacobian matrix, although it is not explicitly required? And, more importantly: are the obtained Quasi-Newton methods the same as if a Transformation Method is employed? This point has been addressed for the most usual Quasi-Newton methods and, as shown next, the answer depends on the particular method.

3.4.2 Rank-two Quasi-Newton methods: DFP and BFGS

In a wide range of problems, tangent stiffness matrices $\boldsymbol{K}_{\mathrm{F}}$ are symmetric and positive definite. An appropriate condition additional to the Quasi-Newton equation (3.14) is then **hereditary symmetry and positive-definiteness**, (i.e. $\boldsymbol{B}^k$ (or $\boldsymbol{H}^k$) is SPD if $\boldsymbol{B}^{k-1}$ (or $\boldsymbol{H}^{k-1}$) is SPD). This requirement leads to the DFP method (direct QN) and the BFGS method (inverse QN), see Dennis & Moré (1977). The BFGS method is often preferred to the DFP method (Matthies & Strang, 1979; Soria & Pegon, 1990; Crisfield, 1991), and is also chosen in this work. The DFP method will be employed here to discuss the implementation of rank-two methods.

DFP method

Combining the Quasi-Newton equation (3.14) with the requirement of hereditary SDP results in, (Dennis & Moré, 1977),

$$\boldsymbol{B}^k_{\mathrm{DFP}} = \boldsymbol{B}^{k-1}_{\mathrm{DFP}} + \boldsymbol{M}^k_{\mathrm{F}} = \boldsymbol{B}^{k-1}_{\mathrm{DFP}} + \frac{\boldsymbol{r}^k_{\mathrm{F}}(\boldsymbol{y}^k_{\mathrm{F}})^{\mathrm{T}} + \boldsymbol{y}^k_{\mathrm{F}}(\boldsymbol{r}^k_{\mathrm{F}})^{\mathrm{T}}}{(\delta\boldsymbol{u}^k_{\mathrm{F}})^{\mathrm{T}}\boldsymbol{y}^k_{\mathrm{F}}} - \frac{(\delta\boldsymbol{u}^k_{\mathrm{F}})^{\mathrm{T}}\boldsymbol{r}^k_{\mathrm{F}}}{\left[(\delta\boldsymbol{u}^k_{\mathrm{F}})^{\mathrm{T}}\boldsymbol{y}^k_{\mathrm{F}}\right]^2}\boldsymbol{y}^k_{\mathrm{F}}(\boldsymbol{y}^k_{\mathrm{F}})^{\mathrm{T}}, \quad (3.18)$$

where $\boldsymbol{r}_{\mathrm{F}}$ contains the residual forces in the free d.o.f., $\boldsymbol{y}^k_{\mathrm{F}}$ is an auxiliary vector defined as $\boldsymbol{y}^k_{\mathrm{F}} := \boldsymbol{r}^k_{\mathrm{F}} - \boldsymbol{r}^{k-1}_{\mathrm{F}}$, $\boldsymbol{M}^k_{\mathrm{F}}$ is a rank-two modification matrix and $\boldsymbol{B}^k_{\mathrm{DFP}}$ is a secant approximation to $\boldsymbol{K}^k_{\mathrm{F}}$.

In a Lagrange-multiplier context, the requirement of hereditary SDP must be added to the adapted Quasi-Newton equation (3.17). This renders

$$\boldsymbol{B}^k_{\mathrm{DFP}} = \boldsymbol{B}^{k-1}_{\mathrm{DFP}} + \boldsymbol{M}^k = \boldsymbol{B}^{k-1}_{\mathrm{DFP}} + \frac{\boldsymbol{r}^k(\boldsymbol{y}^k)^{\mathrm{T}} + \boldsymbol{y}^k(\boldsymbol{r}^k)^{\mathrm{T}}}{(\delta\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{y}^k} - \frac{(\delta\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{r}^k}{[(\delta\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{y}^k]^2}\boldsymbol{y}^k(\boldsymbol{y}^k)^{\mathrm{T}}. \quad (3.19)$$

Equation (3.19) is almost identical formally to Eq. (3.18). However, it is very important to keep in mind that $\boldsymbol{x}$ contains all displacement d.o.f. and Lagrange multipliers, $\boldsymbol{r}$ represents the enlarged residual vector defined in Eq. (3.5), and $\boldsymbol{B}^k_{\mathrm{DFP}}$ is a secant approximation to the Jacobian matrix $\boldsymbol{J}$, not to the stiffness matrix.

To answer the questions at the end of last Subsection, the structure of the modification matrix M^k must be carefully investigated. Is it a full matrix, thus affecting all the block-matrices in B^k_{DFP}? Or does it have some null block-matrices?

By expressing r^k as $(r_1{}^k, r_2{}^k)$ and y^k as $(y_1{}^k, y_2{}^k)$, the matrix M^k in Eq. (3.19) can be written as

$$M^k = \begin{bmatrix} M_{11}{}^k & M_{12}{}^k \\ M_{21}{}^k & M_{22}{}^k \end{bmatrix}, \tag{3.20}$$

where the block-matrices $M_{ij}{}^k$ are

$$M_{ij}{}^k = \frac{r_i{}^k(y_j{}^k)^{\mathrm{T}} + y_i{}^k(r_j{}^k)^{\mathrm{T}}}{(\delta x^k)^{\mathrm{T}} y^k} - \frac{(\delta x^k)^{\mathrm{T}} r^k}{[(\delta x^k)^{\mathrm{T}} y^k]^2} y_i{}^k (y_j{}^k)^{\mathrm{T}}. \tag{3.21}$$

It will be assumed that the common choice ${}^n x^0 = {}^{n-1}x$ is made (i.e., the converged solution in a time-step is employed as initial approximation for the next time-step). The first iteration (prediction) is performed with matrix B^0_{DFP}. After that, the first auxiliary vector y^1 is then a full vector

$$y^1 = r^1 - r^0 = \begin{Bmatrix} r_1{}^1 \\ 0 \end{Bmatrix} - \begin{Bmatrix} -{}^n\Delta f_{\mathrm{ext}} \\ -{}^n\Delta b \end{Bmatrix} = \begin{Bmatrix} r_1{}^1 + {}^n\Delta f_{\mathrm{ext}} \\ {}^n\Delta b \end{Bmatrix} = \begin{Bmatrix} y_1{}^1 \\ y_2{}^1 \end{Bmatrix}, \tag{3.22}$$

because r^0 includes both the increment in the external loads and in the prescribed values of the linear constraints. Note that, on the other hand, only the first subvector of r^1 is non-zero, because the linear constraints are exactly satisfied after the prediction. Equation (3.22) states that there is a jump in the value of the residual vector r for the nonlinear equilibrium equations and, more importantly, for the linear constraint equations, and this has a crucial influence in the structure of the modification matrix of the DFP update.

Indeed, by combining Eqs. (3.19)-(3.22), it can be checked that the secant approximation to the Jacobian matrix after the prediction is

$$B^1_{\mathrm{DFP}} = B^0_{\mathrm{DFP}} + M^1 = \begin{bmatrix} \{K + M_{11}{}^1\} & \{A^{\mathrm{T}} + M_{12}{}^1\} \\ \{A + M_{21}{}^1\} & M_{22}{}^1 \end{bmatrix}, \tag{3.23}$$

where $\boldsymbol{K}$ is the stiffness matrix of $\boldsymbol{B}^0_{\mathrm{DFP}}$. Since $\boldsymbol{y}^1$ is a full vector, $\boldsymbol{M}^1$ is a full matrix and the four block-matrices are affected by the update. As a result, the advantages associated to the linearity of the constraints are lost. If the matrix $\boldsymbol{B}^1_{\mathrm{DFP}}$, Eq. (3.23), is employed to compute the next correction $\boldsymbol{\delta x}^2$, the second block-equation reads

$$\{\boldsymbol{A} + \boldsymbol{M_{21}}^1\}\boldsymbol{\delta u}^2 + \boldsymbol{M_{22}}^1\boldsymbol{\delta\lambda}^2 = \boldsymbol{0}, \tag{3.24}$$

instead of $\boldsymbol{A}\ \boldsymbol{\delta u}^2 = \boldsymbol{0}$, as for the Newton-Raphson methods. This means that $\boldsymbol{u}^2$ does not verify the linear constraints, the associated residual $\boldsymbol{r_2}^2$ is non-zero and the iterative process must cancel both $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$, not only $\boldsymbol{r_1}$ (in spite of this drawback, the method is valid, because it is the standard DFP method applied to the nonlinear system of equations (3.5)).

This unwanted behaviour can be suppressed in a very simple way. Let the first **two** iterations be performed with the initialization matrix $\boldsymbol{B}^0_{\mathrm{DFP}}$ (i.e. the DFP update is not activated after the prediction; first, a modified Newton-Raphson iteration is performed). After that, the auxiliary vector $\boldsymbol{y}^2$ has the form $\boldsymbol{y}^2 = (\boldsymbol{y_1}^2, \boldsymbol{0})$, because $\boldsymbol{r_2}^2 = \boldsymbol{r_2}^1 = \boldsymbol{0}$. If the DFP update is now plugged in, the secant matrix $\boldsymbol{B}^2_{\mathrm{DFP}}$ is

$$\boldsymbol{B}^2_{\mathrm{DFP}} = \boldsymbol{B}^0_{\mathrm{DFP}} + \boldsymbol{M}^2 = \begin{bmatrix} \{\boldsymbol{K} + \boldsymbol{M_{11}}^2\} & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}, \tag{3.25}$$

where only the stiffness matrix is modified, because $\boldsymbol{M_{11}}^2$ is the only non-zero block-matrix in $\boldsymbol{M}^2$, see Eq. (3.21). In a generic iteration k, the secant approximation to the Jacobian matrix is

$$\boldsymbol{B}^k_{\mathrm{DFP}} = \begin{bmatrix} \{\boldsymbol{K} + \sum_{i=2}^{k} \boldsymbol{M_{11}}^i\} & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}, \tag{3.26}$$

where the summation ranges from 2 to k, reflecting that the update is not performed after the prediction. With this minor change, a matrix $\boldsymbol{B}^k_{\mathrm{DFP}}$ with the structure of the Jacobian matrix, Eq. (3.11), is obtained.

Remark 3.3: The need for this change in the generic DFP update scheme can be circumvented if the initial approximation ${}^n\boldsymbol{x}^0$ verifies the linear constraints, (i.e. $\boldsymbol{A}\ {}^n\boldsymbol{u}^0 = {}^n\boldsymbol{b}$). Then $\boldsymbol{r_2}^0 = \boldsymbol{0}$, the auxiliary vector $\boldsymbol{y}^1$ has the form $(\boldsymbol{y_1}^1, \boldsymbol{0})$, $\boldsymbol{M_{11}}^1$ is the only non-null block-matrix in $\boldsymbol{M}^1$ and $\boldsymbol{B}^1_{\mathrm{DFP}}$ has the structure of a Jacobian matrix. Note, however, that choosing the right initial approximation ${}^n\boldsymbol{x}^0$ may not be a straightforward task. For the simple case of prescribed displacements ($u_i = b_i$ for $i = 1 \ldots m$), ${}^n\boldsymbol{x}^0$ is easily taken as $({}^{n-1}\boldsymbol{u} + {}^n\Delta\boldsymbol{b},\ {}^{n-1}\boldsymbol{\lambda})$, but in the general case where each constraint affects more than one d.o.f., the problem becomes more involved.

Remark 3.4: In fact, using the same matrix for the first two iterations, (that is, using the modified Newton-Raphson method for the first two iterations) as done in Eq. (3.26), can be interpreted precisely as a way to choose a "good" initial approximation. Indeed, ${}^n\boldsymbol{x}^0 = {}^{n-1}\boldsymbol{x}$ does not comply with the constraints, but after the prediction, ${}^n\boldsymbol{x}^1$ does. If ${}^n\boldsymbol{x}^1$ is regarded as an "improved initial approximation", then Eq. (3.26) represents the generic DFP update scheme.

To end this Subsection, one last point needs to be addressed: are the DFP methods represented by Eq. (3.18) (Transformation Method) and Eq. (3.26) (Lagrange-multiplier technique) equivalent?

To answer this question, the two modification matrices $\boldsymbol{M}^k_{\mathrm{F}}$ and $\boldsymbol{M_{11}}^k$ must be compared. Recalling that $\boldsymbol{y}^k = (\boldsymbol{y_1}^k, \boldsymbol{0})$ for $k \geq 2$, the block-matrix $\boldsymbol{M_{11}}^k$ can be put as

$$\boldsymbol{M_{11}}^k = \frac{\boldsymbol{r_1}^k(\boldsymbol{y_1}^k)^{\mathrm{T}} + \boldsymbol{y_1}^k(\boldsymbol{r_1}^k)^{\mathrm{T}}}{(\delta\boldsymbol{u}^k)^{\mathrm{T}}\boldsymbol{y_1}^k} - \frac{(\delta\boldsymbol{u}^k)^{\mathrm{T}}\boldsymbol{r_1}^k}{[(\delta\boldsymbol{u}^k)^{\mathrm{T}}\boldsymbol{y_1}^k]^2}\boldsymbol{y_1}^k(\boldsymbol{y_1}^k)^{\mathrm{T}}. \qquad (3.27)$$

Equations (3.18) and (3.27) show that, as expected, $\boldsymbol{M}^k_{\mathrm{F}}$ and $\boldsymbol{M_{11}}^k$ have the same structure, but there are some subtle differences: the scalar products in $\boldsymbol{M}^k_{\mathrm{F}}$ only involve free d.o.f., while the scalar products in $\boldsymbol{M_{11}}^k$ involve full vectors. In a general case, where both free and slave d.o.f. change during the iteration process, $(\delta\boldsymbol{u}^k_{\mathrm{F}})^{\mathrm{T}}\boldsymbol{y}^k_{\mathrm{F}} \neq (\delta\boldsymbol{u}^k)^{\mathrm{T}}\boldsymbol{y_1}^k$

and $(\delta u_{\mathrm{F}}^{k})^{\mathrm{T}} r_{\mathrm{F}}^{k} \neq (\delta u^{k})^{\mathrm{T}} r_{1}{}^{k}$, so the two DFP methods are not algebraically equivalent, although they are entirely analogous.

The two DFP methods do coincide in the particular (but very frequent) case where the linear constraints (3.1b) are simply m prescribed displacements, $u_i = b_i$. The iterative correction in displacements can be put as $\delta u^k = (\delta u_{\mathrm{F}}^{k}, 0)$. Only the free d.o.f. must be iteratively corrected, because the prescribed d.o.f. are set to their known values b_i during the prediction and not further modified. In such a situation, the inequalities of the previous paragraph turn into equalities, $(\delta u_{\mathrm{F}}^{k})^{\mathrm{T}} y_{\mathrm{F}}^{k} = (\delta u^{k})^{\mathrm{T}} y_{1}{}^{k}$ and $(\delta u_{\mathrm{F}}^{k})^{\mathrm{T}} r_{\mathrm{F}}^{k} = (\delta u^{k})^{\mathrm{T}} r_{1}{}^{k}$ and the two DFP methods (Transformation Method and Lagrange-multiplier technique) are algebraically equivalent.

BFGS method

The inverse counterpart of the DFP method is the BFGS method, which is often regarded as the most popular Quasi-Newton method, (Dennis & Moré, 1977). In a Lagrange-multiplier context, the BFGS update is

$$H_{\mathrm{BFGS}}^{k} = H_{\mathrm{BFGS}}^{k-1} + N^{k} = H_{\mathrm{BFGS}}^{k-1} +$$

$$\frac{(-H_{\mathrm{BFGS}}^{k-1} r^{k})(\delta x^{k})^{\mathrm{T}} + \delta x^{k}(-H_{\mathrm{BFGS}}^{k-1} r^{k})^{\mathrm{T}}}{(\delta x^{k})^{\mathrm{T}} y^{k}} - \frac{(-H_{\mathrm{BFGS}}^{k-1} r^{k})^{\mathrm{T}} y^{k}}{[(\delta x^{k})^{\mathrm{T}} y^{k}]^{2}} \delta x^{k} (\delta x^{k})^{\mathrm{T}}, \tag{3.28}$$

where H_{BFGS}^{k} is a secant approximation to $\left[J^{k}\right]^{-1}$, the inverse of the Jacobian matrix. Since the BFGS is an inverse Quasi-Newton method, it is difficult to check if the pattern of the Jacobian matrix is respected directly from Eq. (3.28). To do that, Eq. (3.28) must be inverted to produced the so-called direct version of the BFGS method, (Dennis & Moré, 1977),

$$B_{\mathrm{BFGS}}^{k} = B_{\mathrm{BFGS}}^{k-1} + M^{k} = B_{\mathrm{BFGS}}^{k-1} + \frac{y^{k}(y^{k})^{\mathrm{T}}}{(\delta x^{k})^{\mathrm{T}} y^{k}} + \frac{r^{k}(r^{k})^{\mathrm{T}}}{(\delta x^{k})^{\mathrm{T}} r^{k}}. \tag{3.29}$$

The same arguments previously used for the DFP method apply here. For $k \geq 2$, the only non-zero block-matrix of M^k is $M_{11}{}^{k}$, which can be expressed as

$$M_{11}{}^{k} = \frac{y_{1}{}^{k}(y_{1}{}^{k})^{\mathrm{T}}}{(\delta u^{k})^{\mathrm{T}}(y_{1}{}^{k})} + \frac{r_{1}{}^{k}(r_{1}{}^{k})^{\mathrm{T}}}{(\delta u^{k})^{\mathrm{T}} r_{1}{}^{k}}. \tag{3.30}$$

and the matrix B^k_{BFGS} can be written as B^k_{DFP}, Eq. (3.26).

Remark 3.5: For both the DFP and BFGS methods, there are compact product expressions of the update schemes, Eqs. (3.19) and (3.28), which are amenable to efficient algorithmic implementations, see (Matthies & Strang, 1979; Soria & Pegon, 1990, Brodlie *et al.*, 1973).

In conclusion, for the rank-two Quasi-Newton methods, the natural approach represented by the Quasi-Newton equation (3.17) yields matrices B_{DFP} and B_{BFGS} which automatically have the three constant block-matrices of the Jacobian matrix, although this is not required a priori. To achieve this result, it is necessary to perform the first two iterations with the modified Newton-Raphson method, see Remarks 3.3 and 3.4. Moreover, the resulting Quasi-Newton methods are completely analogous (or even identical, for single-d.o.f. constraints) to the methods obtained if linear constraints are imposed with a Transformation Method.

Convergence properties

In the Lagrange-multiplier context of this work, the DFP and BFGS methods represented by Eqs. (3.19) and (3.28) respectively are the standard methods applied to the nonlinear system $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}$, Eq. (3.5). As a consequence, the usual convergence theorems for rank-two Quasi-Newton methods directly apply: under certain regularity conditions, the two methods converge locally and superlinearly to a solution $\boldsymbol{x}^*$ of the nonlinear system (that is, $\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\| \leq c^k \|\boldsymbol{x}^k - \boldsymbol{x}^*\|$ for some sequence $\{c^k\}$ that converges to 0, provided that $\|\boldsymbol{x}^0 - \boldsymbol{x}^*\| \leq \varepsilon$ and $\|B^0 - \boldsymbol{J}^*\| \leq \delta$), see Broyden *et al.* (1973) and Dennis & Schnabel (1983).

3.4.3 Rank-one Quasi-Newton methods: Broyden

In some nonlinear problems, tangent stiffness matrices are not SDP. This is the situation, for instance, in non-associated plasticity, (Crisfield, 1991), or in materially-nonlinear thermal problems, see Soria & Pegon (1990) and Section 3.5. In such cases, hereditary SDP is no longer an appealing feature of the Quasi-Newton update scheme, and other requirements can be more interesting.

For the Broyden method, (Dennis & Schnabel, 1983), this requirement is that two consecutive matrices $\boldsymbol{B}^{k-1}$ and $\boldsymbol{B}^{k}$ have the same behaviour in all directions except $\boldsymbol{\delta u}_{\mathrm{F}}^{k}$ (note that the Quasi-Newton equation (3.14) already determines how $\boldsymbol{B}^{k}$ transforms $\boldsymbol{\delta u}_{\mathrm{F}}^{k}$):

$$\boldsymbol{B}^{k}\boldsymbol{z}_{\mathrm{F}} = \boldsymbol{B}^{k-1}\boldsymbol{z}_{\mathrm{F}} \qquad \text{for} \qquad \forall \boldsymbol{z}_{\mathrm{F}} \perp \boldsymbol{\delta u}_{\mathrm{F}}^{k}. \tag{3.31}$$

Combining Eqs. (3.14) and (3.31) leads to the update scheme

$$\boldsymbol{B}_{\text{Broyden}}^{k} = \boldsymbol{B}_{\text{Broyden}}^{k-1} + \boldsymbol{M}^{k} = \boldsymbol{B}_{\text{Broyden}}^{k-1} + \frac{\boldsymbol{r}_{\mathrm{F}}^{k}(\boldsymbol{\delta u}_{\mathrm{F}}^{k})^{\mathrm{T}}}{(\boldsymbol{\delta u}_{\mathrm{F}}^{k})^{\mathrm{T}}\boldsymbol{\delta u}_{\mathrm{F}}^{k}}. \tag{3.32}$$

where $\boldsymbol{M}^{k}$ is a rank-one matrix.

In the context of Lagrange multipliers, Eq. (3.31) is transformed into

$$\boldsymbol{B}^{k}\boldsymbol{z} = \boldsymbol{B}^{k-1}\boldsymbol{z} \qquad \text{for} \qquad \forall \boldsymbol{z} \perp \boldsymbol{\delta x}^{k}, \tag{3.33}$$

which renders, together with the (adapted) Quasi-Newton equation (3.17), the update scheme

$$\boldsymbol{B}_{\text{Broyden}}^{k} = \boldsymbol{B}_{\text{Broyden}}^{k-1} + \boldsymbol{M}^{k} = \boldsymbol{B}_{\text{Broyden}}^{k-1} + \frac{\boldsymbol{r}^{k}(\boldsymbol{\delta x}^{k})^{\mathrm{T}}}{(\boldsymbol{\delta x}^{k})^{\mathrm{T}}\boldsymbol{\delta x}^{k}}. \tag{3.34}$$

As commented for the rank-two methods, Eqs. (3.32) and (3.34) are formally very similar but have important differences: the former provides secant approximations to the stiffness matrix $\boldsymbol{K}_{\mathrm{F}}$, while the latter approximates the Jacobian matrix $\boldsymbol{J}$.

Recalling that $\boldsymbol{r}^{k} = (\boldsymbol{r_1}^{k}, \boldsymbol{0})$ for $k \geq 1$ and $\boldsymbol{\delta x}^{k} = (\boldsymbol{\delta u}^{k}, \boldsymbol{\delta\lambda}^{k})$, the rank-one modification matrix $\boldsymbol{M}^{k}$ of Eq. (3.34) can be expressed as

$$\boldsymbol{M}^{k} = \begin{bmatrix} \dfrac{\boldsymbol{r_1}^{k}(\boldsymbol{\delta u}^{k})^{\mathrm{T}}}{(\boldsymbol{\delta x}^{k})^{\mathrm{T}}\boldsymbol{\delta x}^{k}} & \dfrac{\boldsymbol{r_1}^{k}(\boldsymbol{\delta\lambda}^{k})^{\mathrm{T}}}{(\boldsymbol{\delta x}^{k})^{\mathrm{T}}\boldsymbol{\delta x}^{k}} \\ & \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{3.35}$$

Equation (3.35) has an important difference with the rank-two methods of the previous Subsection: there are two non-zero block-matrices. This means that the secant matrix in a generic iteration k is

$$B^k_{\text{Broyden}} = \begin{bmatrix} \{K + \sum_{i=1}^{k} \frac{r_1{}^i(\delta u^i)^{\mathrm{T}}}{(\delta x^i)^{\mathrm{T}}\delta x^i}\} & \{A^{\mathrm{T}} + \sum_{i=1}^{k} \frac{r_1{}^i(\delta \lambda^i)^{\mathrm{T}}}{(\delta x^i)^{\mathrm{T}}\delta x^i}\} \\ & \\ A & \mathbf{0} \end{bmatrix}. \tag{3.36}$$

It can be seen that B^k_{Broyden} does **not** have the structure of the Jacobian matrix, because the block-matrix A^{T} is modified. Moreover, the Lagrange multipliers appear explicitly in the modification matrix. As a consequence, the Broyden method represented by Eq. (3.36) (Lagrange multipliers) is **not** the same that results from Eq. (3.32) (Transformation Method).

To recover the Broyden method associated to a Transformation Method in a Lagrange-multiplier context, the natural approach represented by the Quasi-Newton equation (3.17) must be changed. The basic idea is to impose a priori that the resulting B^k has the three constant block-matrices of the Jacobian matrix. It consists of two steps:

1.- Obtain a secant approximation ${B_{11}}^k$ to the tangent stiffness matrix K^k:

$${B_{11}}^k \approx K^k. \tag{3.37}$$

2.- Enlarge ${B_{11}}^k$ with the **constant** block-matrices A^{T}, A and $\mathbf{0}$ to get a secant approximation to the Jacobian matrix:

$$B^k_{\text{Partial}} = \begin{bmatrix} {B_{11}}^k & A^{\mathrm{T}} \\ & \\ A & \mathbf{0} \end{bmatrix} \approx J^k. \tag{3.38}$$

The subscript Partial denotes that only the tangent stiffness matrix is secantly approximated, thus profiting from the partial linearity of the problem. On the other hand, the matrix B^k_{Broyden} of Eq. (3.36) will be denoted as B^k_{Total} from here on, to emphasize that the whole Jacobian matrix is approximated.

To perform step **1.-**, both the Quasi-Newton equation (3.17) and the additional condition (3.33) must be adapted to this alternative approach. Imposing that the full B^k_{Partial} verifies the original Quasi-Newton equation (3.17) renders

$$\begin{bmatrix} \boldsymbol{B_{11}}^k & \boldsymbol{A}^{\text{T}} \\ \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\delta u}^k \\ \\ \boldsymbol{\delta\lambda}^k \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{r_1}^k - \boldsymbol{r_1}^{k-1} \\ \\ \boldsymbol{0} \end{Bmatrix}. \tag{3.39}$$

The first-block equation in Eq. (3.39) reads

$$\boldsymbol{B_{11}}^k \boldsymbol{\delta u}^k = \boldsymbol{r_1}^k - \boldsymbol{r_1}^{k-1} - \boldsymbol{A}^{\text{T}} \boldsymbol{\delta\lambda}^k, \tag{3.40}$$

which is the Quasi-Newton equation for matrix $\boldsymbol{B_{11}}^k$.

Regarding the additional condition (3.33), it can be recast as

$$\boldsymbol{B_{11}}^k \boldsymbol{z_1} = \boldsymbol{B_{11}}^{k-1} \boldsymbol{z_1} \qquad \text{for} \qquad \forall \boldsymbol{z_1} \perp \boldsymbol{\delta u}^k. \tag{3.41}$$

Combining Eqs. (3.40) and (3.41) yields the update scheme

$$\boldsymbol{B_{11}}^k = \boldsymbol{B_{11}}^{k-1} + \frac{\boldsymbol{r_1}^k (\boldsymbol{\delta u}^k)^{\text{T}}}{(\boldsymbol{\delta u}^k)^{\text{T}} \boldsymbol{\delta u}^k}, \tag{3.42}$$

so the secant approximation to the Jacobian matrix in a generic iteration k is

$$\boldsymbol{B}^k_{\text{Partial}} = \begin{bmatrix} \{\boldsymbol{K} + \sum_{i=1}^k \frac{\boldsymbol{r_1}^i (\boldsymbol{\delta u}^i)^{\text{T}}}{(\boldsymbol{\delta u}^i)^{\text{T}} \boldsymbol{\delta u}^i}\} & \boldsymbol{A}^{\text{T}} \\ \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}, \tag{3.43}$$

where $\boldsymbol{B}^k_{\text{Partial}}$ has, by construction, the structure of a Jacobian matrix. When comparing the Broyden methods represented by Eq. (3.32) (Transformation Method) and Eq. (3.43) (Lagrange multipliers, partial version), the same remarks made for the rank-two methods apply. In a general case, the slave d.o.f. change during the iterations, so $(\boldsymbol{\delta u}^k_{\text{F}})^{\text{T}} \boldsymbol{\delta u}^k_{\text{F}} \neq (\boldsymbol{\delta u}^k)^{\text{T}} \boldsymbol{\delta u}^k$. This means that the two Broyden methods are very similar but not identical. For the particular case of single-d.o.f. constraints, however, $(\boldsymbol{\delta u}^k_{\text{F}})^{\text{T}} \boldsymbol{\delta u}^k_{\text{F}} = (\boldsymbol{\delta u}^k)^{\text{T}} \boldsymbol{\delta u}^k$ and the two Broyden methods are identical.

Convergence properties

If the standard Broyden update is applied to the nonlinear system $\boldsymbol{r}(\boldsymbol{x}) = \mathbf{0}$, Eq. (3.5), the total Broyden method represented by Eq. (3.34) is obtained. As a consequence, in a Lagrange-multiplier context, the standard convergence theorem for the Broyden method applies to the total Broyden method. This theorem (Broyden *et al.*, 1973; Dennis & Schnabel, 1983), is similar to the one for rank-two methods: under some regularity conditions, the Broyden method converges superlinearly to a solution $\boldsymbol{x}^*$ of $\boldsymbol{r}(\boldsymbol{x}) = \mathbf{0}$, provided that the initial approximations $\boldsymbol{x}^0$ and $\boldsymbol{B}^0$ are sufficiently close to $\boldsymbol{x}^*$ and $\boldsymbol{J}^*$ respectively.

The partial Broyden method, on the other hand, is obtained by the modified approach represented by Eqs. (3.37) and (3.38). This method can be regarded as a "least-change secant method", as defined in Dennis & Schnabel (1983).

Least-change secant methods are designed for nonlinear systems with Jacobian matrices of the form

$$\boldsymbol{J}(\boldsymbol{x}) = \boldsymbol{J}_1(\boldsymbol{x}) + \boldsymbol{J}_2(\boldsymbol{x}), \tag{3.44}$$

where $\boldsymbol{J}_1(\boldsymbol{x})$ must be secantly approximated but $\boldsymbol{J}_2(\boldsymbol{x})$ can be easily computed. To build secant approximations $\boldsymbol{B}_1(\boldsymbol{x})$ to $\boldsymbol{J}_1(\boldsymbol{x})$, least-change secant methods proceed in three steps, see Dennis & Schnabel (1983):

1.- Decide the secant condition (i.e. the modified Quasi-Newton equation) that matrices $\boldsymbol{B}_1(\boldsymbol{x})$ must verify.

2.- Choose an affine subspace $\mathcal{A}$ defined by properties such as symmetry or sparsity that all the approximants $\boldsymbol{B}_1(\boldsymbol{x})$ should have.

3.- Select ${\boldsymbol{B}_1}^k$ to be the matrix closest to ${\boldsymbol{B}_1}^{k-1}$ that simultaneously *i)* verifies the secant condition of step **1.-** and *ii)* belongs to the affine subspace $\mathcal{A}$ selected in step **2.-**. In this manner, ${\boldsymbol{B}_1}^k$ is the least-change update to ${\boldsymbol{B}_1}^{k-1}$ with the desired properties.

It can be checked that the partial Broyden method fits into this framework. Recalling the expression of the Jacobian matrix, Eq. (3.11), it can be seen that the matrices $\boldsymbol{J}_1(\boldsymbol{x})$

and $J_2(x)$ for the nonlinear system under consideration, Eq. (3.5), are

$$J_1(x) = \begin{bmatrix} K(u) & 0 \\ 0 & 0 \end{bmatrix} \quad ; \quad J_2(x) = J_2 = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}, \tag{3.45}$$

where $J_2(x)$ is, in fact, a constant matrix J_2.

By combining Eqs. (3.44) and (3.45), it can be verified that the modified Quasi-Newton equation for the partial Broyden method, Eq. (3.40), can be put as

$$B_1{}^k \delta x^k = r^k - r^{k-1} - J_2 \delta x^k, \tag{3.46}$$

which is the secant condition for matrix $B_1{}^k$.

Regarding the choice of the a affine subspace $\mathcal{A}$, all the approximations $B_1{}^k$ are required to have the sparsity pattern of $J_1(x)$ (i.e. the three null block-matrices, see Eq. (3.45)). This requirement defines $\mathcal{A}$.

Finally, the additional condition (3.33) can be interpreted as requiring $B_1{}^k$ to be the matrix closest to $B_1{}^{k-1}$ (measured on the Frobenius norm) which verifies Eq. (3.46) and has the sparsity pattern of $J_1(x)$.

Since the partial Broyden method is a least-change secant method, the general convergence theorem of Dennis & Walker (1981) applies. The theorem states the **superlinear** convergence of these type of methods under certain conditions. In particular, if some regularity assumptions are made, a **sufficient** condition for local superlinear convergence to a solution x^* of the nonlinear system is that $J_1(x^*)$ belongs to the affine subspace $\mathcal{A}$.

For the partial Broyden method, this sufficient condition is trivially verified, because $\mathcal{A}$ is selected precisely by imposing the sparsity pattern of $J_1(x)$. In conclusion, the partial Broyden method is locally and superlinearly convergent.

The inverse Broyden method

From an algorithmic viewpoint, the inverse Broyden method is preferred to the direct Broyden method just discussed, see Dennis & Moré (1977) and Soria & Pegon (1990). The basic idea is to employ the Sherman & Morrison lemma, see Sherman & Morrison (1949) and Box 3.1, to invert the direct update formula. This lemma gives an explicit expression for the inverse of $\boldsymbol{B} + \boldsymbol{\alpha}\boldsymbol{\beta}^{\mathrm{T}}$, that is, a rank-one modification of matrix $\boldsymbol{B}$. The total Broyden method, Eq. (3.34), corresponds to $\boldsymbol{\alpha}^k = \boldsymbol{r}^k$ and $\boldsymbol{\beta}^k = \frac{\delta\boldsymbol{x}^k}{(\delta\boldsymbol{x}^k)^{\mathrm{T}}\delta\boldsymbol{x}^k}$, and can be inverted into

$$\boldsymbol{H}^k_{\text{Total}} = \boldsymbol{H}^{k-1}_{\text{Total}} - \frac{\boldsymbol{H}^{k-1}_{\text{Total}}\boldsymbol{r}^k(\delta\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{H}^{k-1}_{\text{Total}}}{(\delta\boldsymbol{x}^k)^{\mathrm{T}}\boldsymbol{H}^{k-1}_{\text{Total}}\boldsymbol{y}^k}, \tag{3.47}$$

where $\boldsymbol{H}^k_{\text{Total}}$ approximates $\left[\boldsymbol{J}^k\right]^{-1}$, the inverse of the Jacobian matrix.

Let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta} \in \mathbb{R}^n$ and assume that $\boldsymbol{B}$ is a real n-dimensional regular matrix. Then:

1. The matrix $\boldsymbol{B} + \boldsymbol{\alpha}\boldsymbol{\beta}^{\mathrm{T}}$ is regular if and only if

$$\sigma := 1 + \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{B}^{-1}\boldsymbol{\alpha} \neq 0.$$

2. If $\sigma \neq 0$, then the inverse of $\boldsymbol{B} + \boldsymbol{\alpha}\boldsymbol{\beta}^{\mathrm{T}}$ is

$$\left(\boldsymbol{B} + \boldsymbol{\alpha}\boldsymbol{\beta}^{\mathrm{T}}\right)^{-1} = \boldsymbol{B}^{-1} - \frac{1}{\sigma}\boldsymbol{B}^{-1}\boldsymbol{\alpha}\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{B}^{-1}.$$

Box 3.1 The Sherman & Morrison lemma

For the partial Broyden method, this inversion procedure is not so straightforward. The update scheme for matrix $\boldsymbol{B}_{11}$, Eq. (3.42), **cannot** be inverted via the Sherman & Morrison lemma because matrix $\boldsymbol{B}_{11}$, which is an approximation of the full stiffness matrix $\boldsymbol{K}$, is singular. As commented previously, the transformed stiffness matrix

$\boldsymbol{K}_{\mathrm{F}}$ and the Jacobian matrix $\boldsymbol{J}$ are typically regular, but the full stiffness matrix $\boldsymbol{K}$ is singular, because the former include the boundary conditions while the latter does not.

For this reason, the Sherman & Morrison lemma must be applied to the full matrix $\boldsymbol{B}^k_{\mathrm{Partial}}$, which is regular because $\boldsymbol{A}$ accounts for boundary conditions. To that aim, it is necessary that the update formula for the block-matrix $\boldsymbol{B}_{11}$ can be transformed into an update formula for $\boldsymbol{B}_{\mathrm{Partial}}$ of the form

$$\boldsymbol{B}^k_{\mathrm{Partial}} = \boldsymbol{B}^{k-1}_{\mathrm{Partial}} + \boldsymbol{\alpha}^k(\boldsymbol{\beta}^k)^{\mathrm{T}}. \tag{3.48}$$

Recalling that $\boldsymbol{r}^k = (\boldsymbol{r_1}^k, \boldsymbol{0})$ for $k \geq 1$, it can be checked that Eq. (3.48) holds for $\boldsymbol{\alpha}^k = \boldsymbol{r}^k$ and $\boldsymbol{\beta}^k = (\frac{\delta\boldsymbol{u}^k}{(\delta\boldsymbol{u}^k)^{\mathrm{T}}\delta\boldsymbol{u}^k}, \boldsymbol{0})$, and then it can be inverted with the Sherman & Morrison lemma into

$$\boldsymbol{H}^k_{\mathrm{Partial}} = \boldsymbol{H}^{k-1}_{\mathrm{Partial}} - \frac{\boldsymbol{H}^{k-1}_{\mathrm{Partial}}\boldsymbol{r}^k(\delta\boldsymbol{u}^k, \boldsymbol{0})^{\mathrm{T}}\boldsymbol{H}^{k-1}_{\mathrm{Partial}}}{(\delta\boldsymbol{u}^k, \boldsymbol{0})^{\mathrm{T}}\boldsymbol{H}^{k-1}_{\mathrm{Partial}}\boldsymbol{y}^k}. \tag{3.49}$$

Comparing Eqs. (3.47) and (3.49) clearly shows that the only difference between the total and partial Broyden methods lies in the definition of vector $\boldsymbol{\beta}^k$: the whole vector of unknowns $\delta\boldsymbol{x}^k$ is employed for the total Broyden method, while only displacements $\delta\boldsymbol{u}^k$ (that is, the fundamental unknowns) are used in the partial Broyden method.

The inverse Broyden method is typically implemented by means of a recurrent product formula which relates $\boldsymbol{H}^k$ to the initialization matrix $\boldsymbol{H}^0$, (Engelman *et al.*, 1981; Soria & Pegon, 1990). This procedure has been adapted to both the partial and total Broyden methods that arise in a Lagrange-multiplier context, see Box 3.2.

It can be seen in Box 3.2 that the two algorithms are identical, except for the definition of vector $\boldsymbol{\beta}^k$ in step **8.** It is also worth mentioning that the scalar products in steps **9.** and **10.** involve only the displacement d.o.f. for the partial Broyden method, but both displacements and Lagrange multipliers for the total Broyden method.

In conclusion, for the rank-one Broyden method, the natural approach associated to the Quasi-Newton equation (3.17) leads to the total Broyden method, with matrices $\boldsymbol{B}^k_{\mathrm{Total}}$ that do **not** maintain the structure of the Jacobian matrix. Contrary to what happens for rank-two methods, it is necessary to require a priori that constant block-matrices $\boldsymbol{A}$, $\boldsymbol{A}^{\mathrm{T}}$ and $\mathbf{0}$ remain unchanged. This is achieved by means of an alternative approach, represented by Eqs. (3.37) and (3.38), where only the tangent stiffness matrix is secantly approximated. The resulting partial Broyden method is the analogue of the Broyden method with linear constraints imposed via a Transformation Method. The relative performance of the total and partial Broyden methods will be assessed in Section 3.5 with some numerical examples.

3.4.4 Secant-Newton methods

The major drawback of the inverse Quasi-Newton methods is that the computational cost **per iteration** increases at each iteration. For the inverse Broyden method, this can be checked from Box 3.2, where the number of stored vectors (steps **7.** and **8.**) and of scalar products (steps **7.**, **9.** and **10.**) increase with the iteration counter k. A similar situation is found with the BFGS method, (Crisfield, 1991; Soria & Pegon, 1990).

The Secant-Newton methods, (Crisfield, 1991), were developed to remedy this problem. The basic idea is to replace matrix $\boldsymbol{H}^{k-1}$ with the initialization matrix $\boldsymbol{H}^{0}$ in the update schemes. In this manner, $\boldsymbol{H}^{k}$ is computed at each iteration from the **same** matrix $\boldsymbol{H}^{0}$, not from the matrix $\boldsymbol{H}^{k-1}$ of the previous iteration. From an algorithmic stand, this results in a constant number of scalar products and stored vectors, that is, a constant cost per iteration, (Crisfield, 1991). The simplifying strategy is general, so a Secant-Newton method can be obtained from every inverse Quasi-Newton method. Thus, in a Lagrange-multiplier setting, it is possible to devise two Secant Broyden methods (total and partial) and one Secant BFGS method.

```
FOR EVERY LOAD-STEP:
1.– Choose an initial approximation $\boldsymbol{x}^0$ and compute the residual vector $\boldsymbol{r}^0$.
2.– Compute the initialization matrix $\boldsymbol{B}^0$.
3.– Solve the linear system $\boldsymbol{B}^0\ \delta\boldsymbol{x}^1 = -\boldsymbol{r}^0$ and store $\delta\boldsymbol{x}^1$.
4.– Update $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \delta\boldsymbol{x}^1$ and compute $\boldsymbol{r}^1$.
5.– Convergence control: if $\boldsymbol{x}^1$ is good enough, exit.
    $k \leftarrow 1$
6.– Solve the linear system $\boldsymbol{B}^0\ \boldsymbol{v} = -\boldsymbol{r}^k$.
7.– For $k = 1$ set $\boldsymbol{t} = \boldsymbol{v}$.
    For $k > 1$ recover $\boldsymbol{\omega}^i$, $\delta\boldsymbol{x}^i$ for $i = 1, \dots k-1$ from storage and compute
        $\boldsymbol{t} = \prod_{i=k-1}^{1} \left[ \boldsymbol{I} + \boldsymbol{\omega}^i (\delta\boldsymbol{x}^i)^{\mathrm{T}} \right] \boldsymbol{v}$
8.– Recover $\delta\boldsymbol{x}^k$ from storage and define $\boldsymbol{\beta}^k$:
        TOTAL :    $\boldsymbol{\beta}^k = (\delta\boldsymbol{u}^k, \delta\lambda^k) = \delta\boldsymbol{x}^k$
        PARTIAL :  $\boldsymbol{\beta}^k = (\delta\boldsymbol{u}^k, 0)$
9.– Compute and store $\boldsymbol{\omega}^k = \frac{\boldsymbol{t}}{(\boldsymbol{\beta}^k)^{\mathrm{T}}(\delta\boldsymbol{x}^k - \boldsymbol{t})}$.
10.– Compute and store $\delta\boldsymbol{x}^{k+1} = \boldsymbol{t} + \left[ (\boldsymbol{\beta}^k)^{\mathrm{T}} \boldsymbol{t} \right] \boldsymbol{\omega}^k$.
11.– Update $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \delta\boldsymbol{x}^{k+1}$ and compute $\boldsymbol{r}^{k+1}$.
12.– Convergence control: if $\boldsymbol{x}^{k+1}$ is good enough, exit.
13.– Set $k \leftarrow k + 1$ and go back to 6.
```

Box 3.2 Total and partial Broyden methods

The Secant BFGS method

If matrix $\boldsymbol{H}^{k-1}_{\mathrm{BFGS}}$ is replaced with matrix $\boldsymbol{H}^{0}_{\mathrm{BFGS}}$ in the BFGS method, Eq. (3.28), it renders, after accounting for the relation $\boldsymbol{\delta x}^{k+1} = \boldsymbol{H}^{k}_{\mathrm{BFGS}}(-\boldsymbol{r}^k)$ and some manipulation, (Crisfield, 1991),

$$\boldsymbol{\delta x}^{k+1} = (1+C)\boldsymbol{\delta\tilde{x}}^{k+1} - C\boldsymbol{\delta\tilde{x}}^{k} + [C-(1+C)B+CA]\,\boldsymbol{\delta x}^{k}, \tag{3.50}$$

where $\boldsymbol{\delta\tilde{x}}^{k}$ is the solution to the linear system $\boldsymbol{B}^{0}_{\mathrm{BFGS}}\boldsymbol{\delta\tilde{x}}^{k} = -\boldsymbol{r}^k$, and the scalars A, B and C involve scalar products between vectors $\boldsymbol{\delta x}^{k}$, $\boldsymbol{\delta\tilde{x}}^{k}$, $\boldsymbol{\delta\tilde{x}}^{k+1}$, $\boldsymbol{r}^{k-1}$ and $\boldsymbol{r}^{k}$. For the same reasons pointed out for the (Quasi-Newton) BFGS method, it is necessary to perform the first two iterations with the initialization matrix, and use the Secant BFGS update, Eq. (3.50), for $k \geq 2$.

The Secant Broyden method

Two Secant Broyden methods can be obtained if linear constraints are imposed via Lagrange multipliers: a total Secant Broyden method, where $\boldsymbol{H}^{k-1}_{\mathrm{Total}}$ is replaced with $\boldsymbol{H}^{0}_{\mathrm{Total}}$ in the total Broyden method, Eq. (3.47), and a partial Secant Broyden method, with a similar transformation performed in Eq. (3.49). After some manipulation, these changes lead to the algorithms shown in Box 3.3. As remarked for the (Quasi-Newton) Broyden methods, the only difference resides in the definition of vector $\boldsymbol{\beta}^k$. The two Secant Broyden methods will be compared by means of some numerical experiments in Section 3.5.

```
FOR EVERY LOAD-STEP:
 1.– Choose an initial approximation $\boldsymbol{x}^0$ and compute the residual vector $\boldsymbol{r}^0$.
 2.– Compute the initialization matrix $B^0$.
 3.– Solve the linear system $B^0\ \delta\boldsymbol{x}^1 = -\boldsymbol{r}^0$ and set $\delta\tilde{\boldsymbol{x}}^1 = \delta\boldsymbol{x}^1$.
 4.– Update $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \delta\boldsymbol{x}^1$ and compute $\boldsymbol{r}^1$.
 5.– Convergence control: if $\boldsymbol{x}^1$ is good enough, exit.
     $k \leftarrow 1$
 6.– Solve the linear system $B^0\ \delta\tilde{\boldsymbol{x}}^k = -\boldsymbol{r}^k$.
 7.– Recover $\delta\boldsymbol{x}^k$ from storage and define $\boldsymbol{\beta}^k$:
         TOTAL :    $\boldsymbol{\beta}^k = (\delta\boldsymbol{u}^k, \delta\boldsymbol{\lambda}^k) = \delta\boldsymbol{x}^k$
         PARTIAL :  $\boldsymbol{\beta}^k = (\delta\boldsymbol{u}^k, 0)$
 8.– Compute $\rho = \frac{(\boldsymbol{\beta}^k)^{\mathrm{T}}\delta\tilde{\boldsymbol{x}}^{k+1}}{(\boldsymbol{\beta}^k)^{\mathrm{T}}(\delta\tilde{\boldsymbol{x}}^k - \delta\tilde{\boldsymbol{x}}^{k+1})}$.
 9.– Compute $\delta\boldsymbol{x}^{k+1} = (1+\rho)\delta\tilde{\boldsymbol{x}}^{k+1} + \rho\delta\boldsymbol{x}^k - \rho\delta\tilde{\boldsymbol{x}}^k$.
10.– Update $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \delta\boldsymbol{x}^{k+1}$ and compute $\boldsymbol{r}^{k+1}$.
11.– Convergence control: if $\boldsymbol{x}^{k+1}$ is good enough, exit.
12.– Set $k \leftarrow k+1$ and go back to 6.
```

Box 3.3 Total and partial Secant Broyden methods

3.5 Numerical examples

3.5.1 Introduction

The algorithms just discussed have been implemented in CASTEM 2000, an object-oriented code that employs the Lagrange-multiplier technique to handle linear constraints, see Chapter 2. For that purpose, new metaoperators have been written in the object-oriented language GIBIANE, thus extending the code's capabilities.

Several numerical tests in the fields of nonlinear mechanical and thermal analysis have been performed. Four of these numerical tests are shown in this Section, with the purpose of:

1.– Showing that the various methods can be effectively implemented in the object-oriented, Lagrange-multiplier environment of this work.

2.– Assessing the relative performance of the different nonlinear solvers.

3.– Comparing the partial and total Broyden methods, both for Quasi-Newton and Secant-Newton versions.

It has been attempted to cover the most relevant issues: SPD/non-symmetric stiffness matrices; one-point constraints/multipoint constraints; academic tests/more realistic analyses, etc

The four presented tests are:

Test: SHELL (Subsection 3.5.2)

Brief description: axisymmetrical shell with an apex load

Source of nonlinearity: large strains

Linear constraints: restrained d.o.f.

Basic feature: structural finite elements

Test: THERMAL (Subsection 3.5.3)

Brief description: nonlinear diffusion of heat in a cyclic domain

Source of nonlinearity: temperature-dependent conductivity

Linear constraints: cyclic/periodic boundary conditions

Basic feature: non-symmetric Jacobian matrices

Test: TWO HOLES (Subsection 3.5.4)

Brief description: strip under compression with two perforations

Source of nonlinearity: large strains and elastoplaticity

Linear constraints: central symmetry in boundary conditions

Basic feature: central symmetry and localization

Test: BRIDGE (Subsection 3.5.5)

Brief description: structural analysis of an existing bridge

Source of nonlinearity: nonlinear material behaviour and large strains

Linear constraints: restrained d.o.f.

Basic feature: engineering analysis

3.5.2 Test SHELL

The first example is a well-known benchmark test in nonlinear mechanics, see Zienkiewicz & Taylor (1991). An axisymmetrical shell, see Figure 3.1, is clamped at its border (i.e. displacements and rotations are restrained) and loaded with a vertical force in its apex ($r = 0$). The shell is made of an elastic material but undergoes large deformations, so a geometrically nonlinear problem results. To account for large strains, the Updated Lagrangian formulation (Bathe, 1982) is employed, in combination with the second-order stress update algorithm discussed in Chapter 4.

A load-controlled incremental-iterative analysis is performed. A total load of $P = 80$lb is applied in 16 load-steps. A relative criterion in displacements with a tolerance of $\varepsilon = 10^{-4}$ is used as the basic convergence criterion, and convergence in forces is also checked. Figure 3.2 shows the load P versus the deflection of the apex, v.

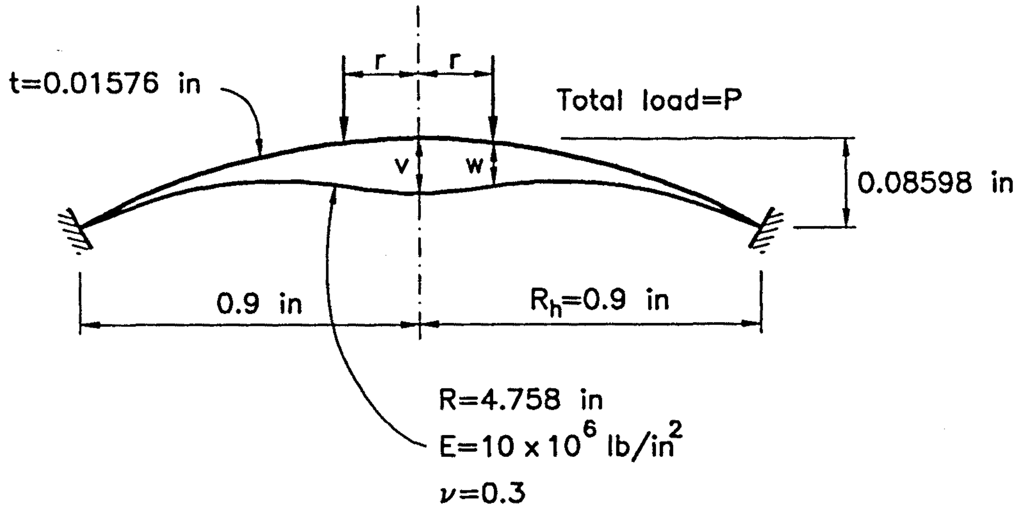

Figure 3.1 Test SHELL. Problem statement

The test has been performed with all the implemented nonlinear solvers. Table 3.1 shows their computational cost, both in terms of total number of iterations and CPU time.

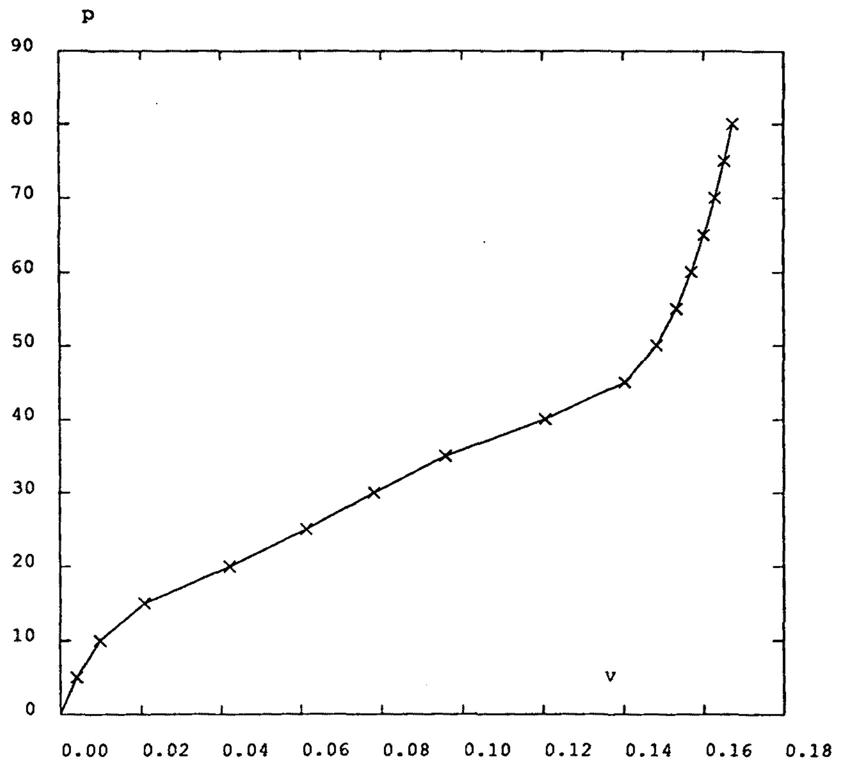

Figure 3.2 Test SHELL. Load vs. deflection curve

Method	Iterations	CPU time
Full Newton-Raphson	74	—
Modified Newton-Raphson	301	175%
Partial Broyden (QN)	189	136%
Total Broyden (QN)	—	—
BFGS (QN)	130	100%
Partial Secant-Broyden	225	137%
Total Secant-Broyden	—	—
Secant-BFGS	—	—

Table 3.1 Test SHELL. Computational cost of various methods

It can be noted from Table 3.1 that the full Newton-Raphson method is superior to the rest in this case. The CPU time is not shown for this method because it is implemented at the operator level in the object-oriented code, while all the other methods are implemented as metaoperators, which are computationally less efficient. Among the Quasi-Newton methods, the best is the BFGS. This is an expected result, because the Jacobian matrices are symmetric for this problem. Second comes the partial Broyden method. The total Broyden method does not converge (a maximun number of 50 iterations per step is set). Among the Secant-Newton methods, the only one that achieves convergence is the partial Secant-Broyden method.

3.5.3 Test THERMAL

The second test studies the nonlinear diffusion of heat, see Soria & Pegon (1990). Nonlinearity is associated to a non-constant, temperature-dependent thermal conductivity K of the form

$$K(T) = 1 + 2T^m, \tag{3.51}$$

where T is the temperature. For this problem, the conductivity matrices (and hence the Jacobian matrices) are non-symmetric, see Soria & Pegon (1990).

The problem domain is a hollow disc with six rectangular perforations, see Figure 3.3. The boundary conditions, also shown in Figure 3.3, are: prescribed temperature in the inner circle ($T = 0$) and in the rectangular inclusions ($T = 10$), and a convection condition in the outer circle.

Since the problem has a cyclic periodicity of angle $\pi/3$, the solution must have the same periodicity. This allows to solve the problem with only one-sixth of the total domain, see Figure 3.4. The temperature is set to be equal (pointwise) in the two straight lines, thus imposing the periodicity in the solution. This linear constraint can be implemented in a simple and efficient manner thanks to the combination of Lagrange multipliers and object-oriented environment: the two straight lines are two independent objects, and the constraint matrix A is produced with an appropriate operator. The resulting temperature field is shown in Figure 3.5.

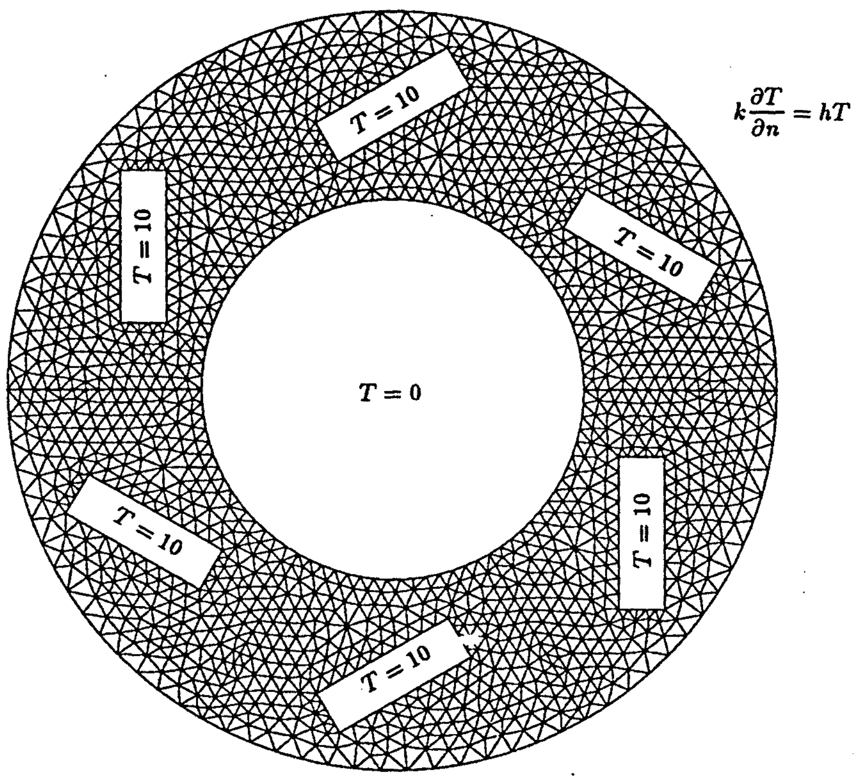

Figure 3.3 Test THERMAL. Domain, mesh and boundary conditions

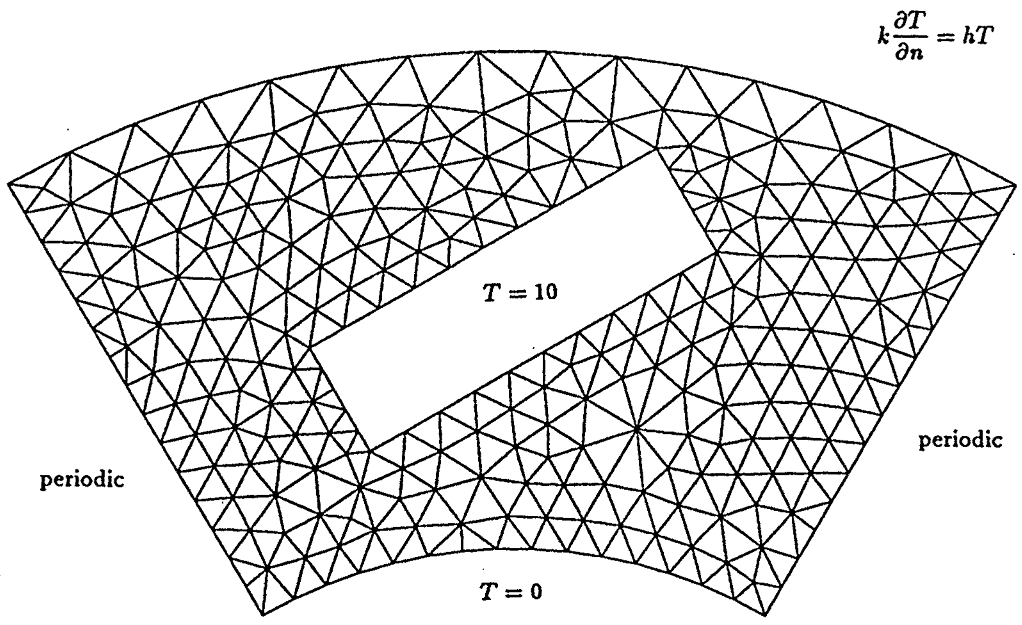

Figure 3.4 Test THERMAL. One-sixth of domain

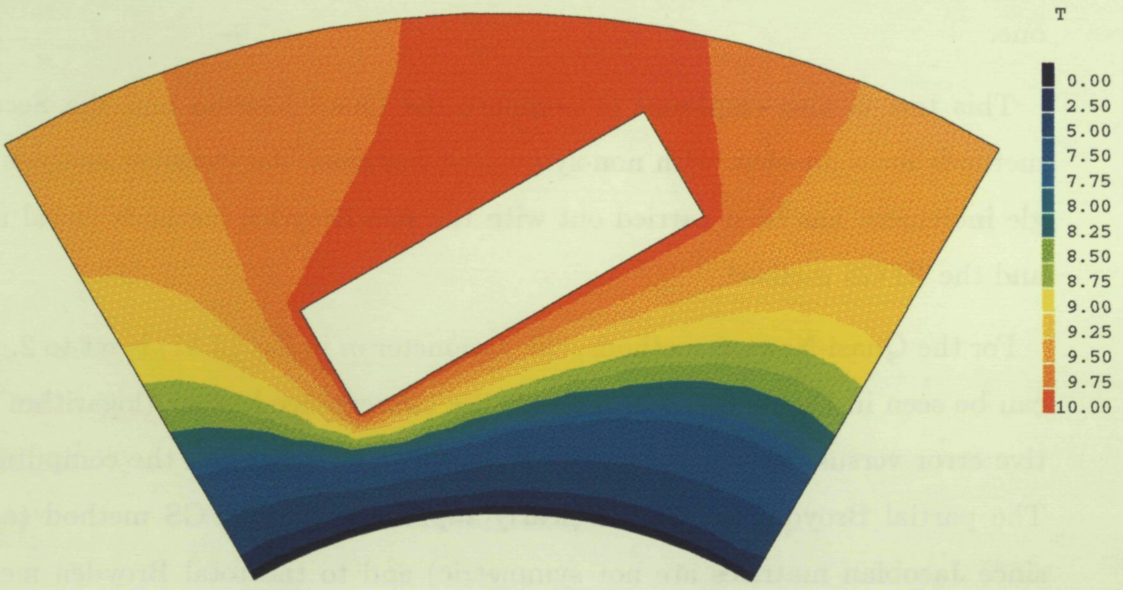

Figure 3.5 Test THERMAL. Temperature field

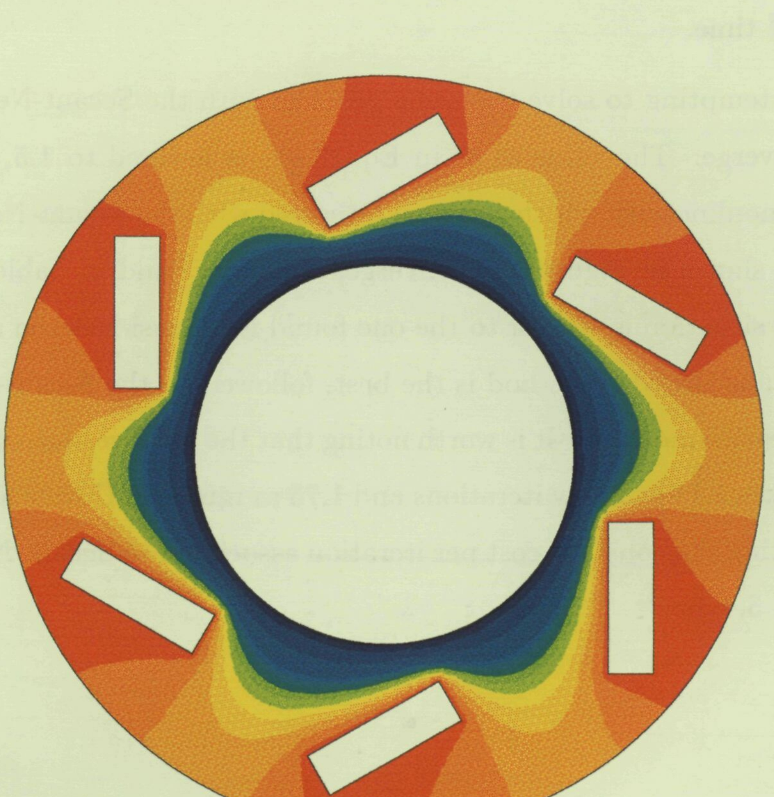

Figure 3.6 Test THERMAL. Temperature field. Problem solved over the whole domain

As a verification, the problem was also solved over the whole domain of Figure 3.3. As expected, the obtained solution, see Figure 3.6, is in total agreement with the previous one.

This test is also employed to compare the Quasi-Newton and the Secant-Newton methods for a problem with non-symmetric matrices. An iterative analysis with a single increment has been carried out with the two Broyden methods (total and partial) and the BFGS method.

For the Quasi-Newton methods, the parameter m in Eq. (3.51) is set to 2. The results can be seen in Figure 3.7, which shows the convergence history (logarithm of the relative error versus iterations), and in Table 3.2, which displays the computational cost. The partial Broyden method is clearly superior to the BFGS method (as expected, since Jacobian matrices are not symmetric) and to the total Broyden method. It is also worth mentioning that the effect of the increasing cost per iteration associated to Quasi-Newton methods (see Subsection 3.4.4) is manifest in Table 3.2: the total Broyden methods takes 1.56 as many iterations as the partial Broyden method, but twice as much CPU time.

When attempting to solve the same problem with the Secant-Newton methods, they fail to converge. The value of m in Eq. (3.51) is lowered to 1.5, thus decreasing the problem's nonlinearity, which can then be solved with the Secant-Newton methods. The results are shown in Figure 3.8 (convergence history) and in Table 3.3 (computational cost). The situation is similar to the one found for Quasi-Newton methods. Again, the partial Secant-Broyden method is the best, followed by the Secant-BFGS and the total Secant-Broyden methods. It is worth noting that the total version of the Secant-Broyden method takes 1.74 as many iterations and 1.75 as much CPU time as the partial version, thus reflecting the constant cost per iteration associated to Secant-Newton methods (see Subsection 3.4.4).

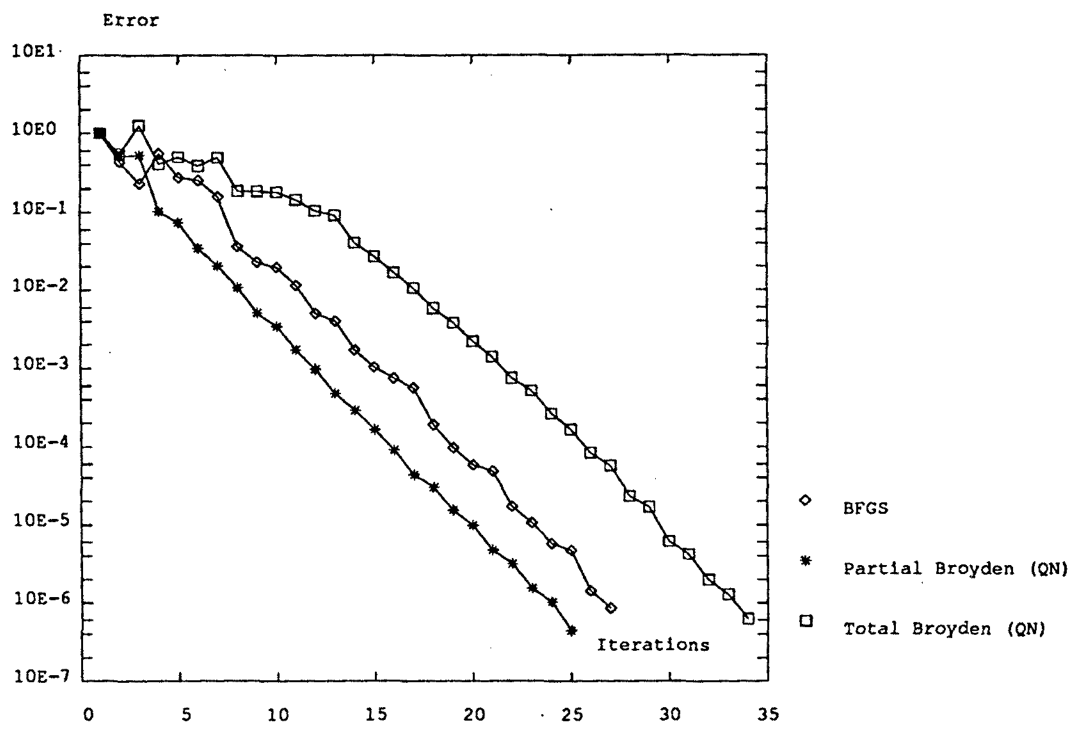

Figure 3.7 Test THERMAL. Convergence history for Quasi-Newton methods

Method	Iterations	CPU time
Partial Broyden (QN)	24	100%
Total Broyden (QN)	34	168%
BFGS (QN)	27	150%

Table 3.2 Test THERMAL. Computational cost for Quasi-Newton methods

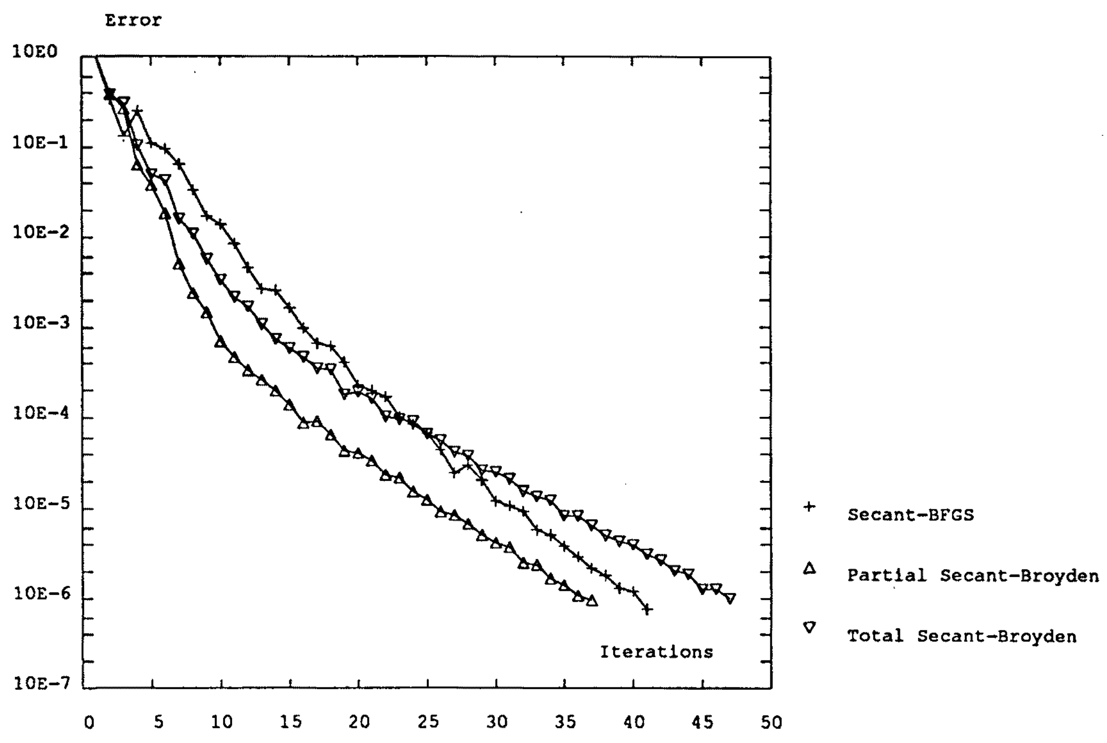

Figure 3.8 Test THERMAL. Convergence history for Secant-Newton methods

Method	Iterations	CPU time
Partial Secant-Broyden	37	100%
Total Secant-Broyden	47	125%
Secant-BFGS	41	110%

Table 3.3 Test THERMAL. Computational cost for Secant-Newton methods

3.5.4 Test TWO HOLES

A strip with two perforations, see Figure 3.9, is subjected to uniaxial compression. This test is inspired in an example found in Zienkiewicz & Huang (1990). In this reference, the two holes are along the longitudinal axis of symmetry, so only a quarter of the specimen is modelled. Here the position of the two holes is altered. The domain has central symmetry with respect to its centre, O, but there are no axes of symmetry, see Figure 3.9.

Thanks to the central symmetry, the problem can be solved by modelling only half the specimen, see Figure 3.10. The finite element mesh, also shown in Figure 3.10, has been obtained through a process of adaptive remeshing, which combines an unstructured quadrilateral mesh generator (Sarrate, 1996; Sarrate *et al.*, 1993; Montolío *et al.*, 1995), and an a-posteriori error estimator for nonlinear finite element analysis, see Díez *et al.* (1995), Huerta *et al.* (1996). The sequence of meshes obtained during the process of adaptive remeshing can be seen in Figure 3.11.

The required boundary conditions in line AB, see Figure 3.10, are 1) restrained displacements in the centre O and 2) symmetric displacements with respect to O along line AB. For any pair of opposed nodes A' and B', this means

$$\left.\begin{aligned} u_x(A') &= -u_x(B') \\ u_y(A') &= -u_y(B') \end{aligned}\right\}, \tag{3.52}$$

where u_x and u_y are the displacement components. This results in a set of linear constraints, which are handled via Lagrange multipliers (see Chapter 3). In the object-oriented environment of this work, objects OA and OB (of class `Mesh`) are created, so these linear constraints can be imposed with just one sentence of the GIBIANE language, see Section 2.3.

A displacement-controlled, plane strain analysis of the piece, with large deformations and nonlinear material behaviour, has been performed. To handle large strains, the second-order stress update algorithm discussed in Chapter 4 has been employed. Elastoplastic behaviour is modelled by a bilinear law.

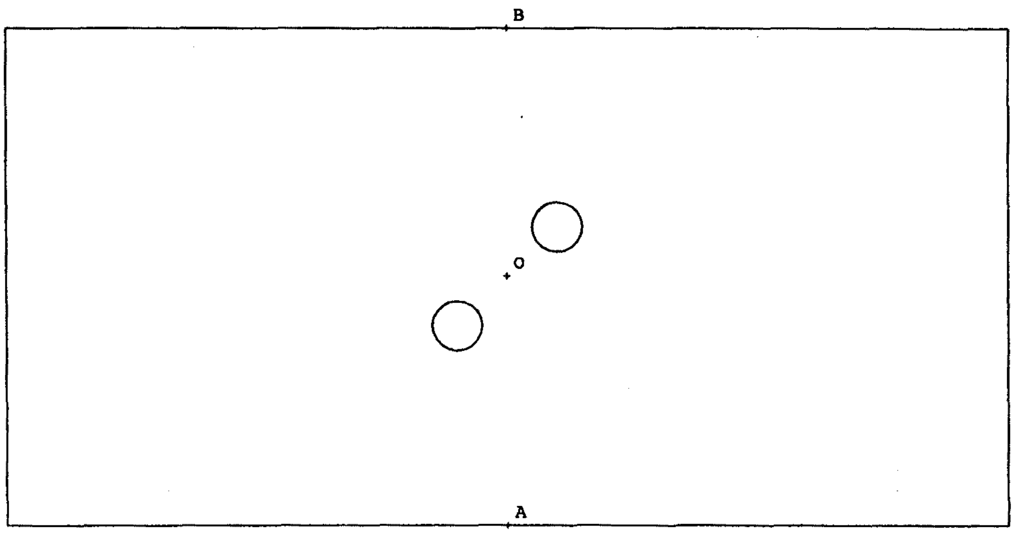

Figure 3.9 Test TWO HOLES. Problem domain

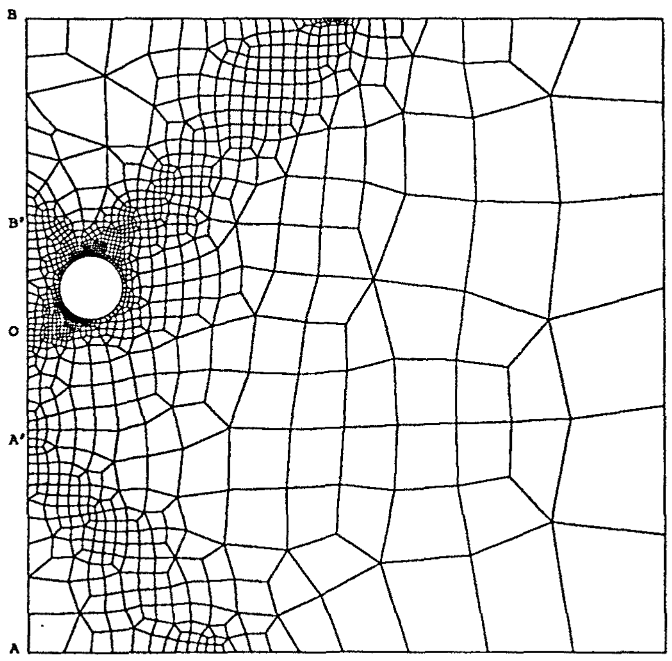

Figure 3.10 Test TWO HOLES. Computational domain and finite element mesh

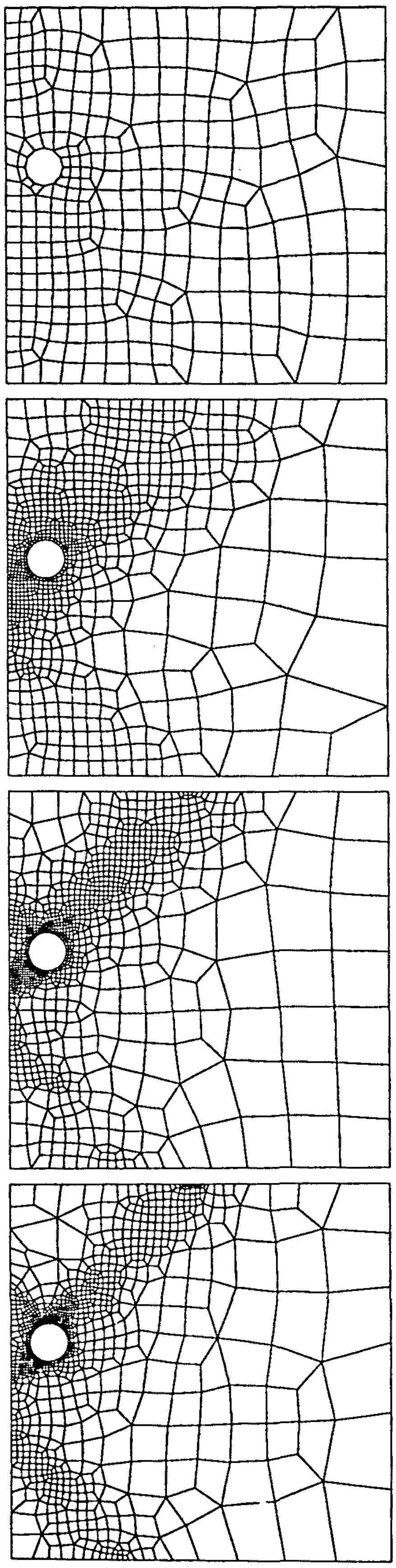

Figure 3.11 Test TWO HOLES. Sequence of meshes obtained during adaptive remeshing

Figure 3.12 shows the equivalent plastic strain over the deformed shape. The band formation due to strain concentration is clearly visible. The effect of the central-symmetry boundary conditions is also apparent, with a "mysterious" band appearing from nowhere in the bottom left corner. Of course, this band is originated by the hole in the other half of the specimen. This can be seen in Figure 3.13, where the plastic strain is depicted over the whole piece. The deformed line AB is also shown in Figure 3.13. As prescribed, this line is symmetric with respect to point O.

Various of the nonlinear solvers of Sections 3.3 and 3.4 have been employed. The time-step is automatically updated, as suggested by Crisfield (1990). The results are shown in Table 3.4. None of the Newton-Raphson methods allows to carry out the complete computation. For the full Newton-Raphson method, the iterative process diverges at about 54% of the total planned displacement. The modified Newton-Raphson and initial stress methods, on the other hand, exhibit a extremely slow convergence (no convergence in 500 iterations for a time-step at about 36% of the analysis), and are disregarded. The complete analysis can be performed, on the contrary, with two Quasi-Newton methods: the partial Broyden and the BFGS, see Table 3.4. In view of the behaviour of the full Newton-Raphson method, the elastic stiffness matrix ${}^0K$ is used in the initialization matrix B^0, see Subsection 3.4.1. Regarding the total Broyden method, it diverges at about 86% of the total analysis.

Method	Time-steps	Iterations	Fraction of analysis
Full Newton-Raphson	—	—	54%
Modified Newton-Raphson	—	—	36%
Initial stress	—	—	36%
Partial Broyden (QN)	33	492	100%
Total Broyden (QN)	—	—	86%
BFGS (QN)	35	451	100%

Table 3.4 Test TWO HOLES. Behaviour of various nonlinear solvers

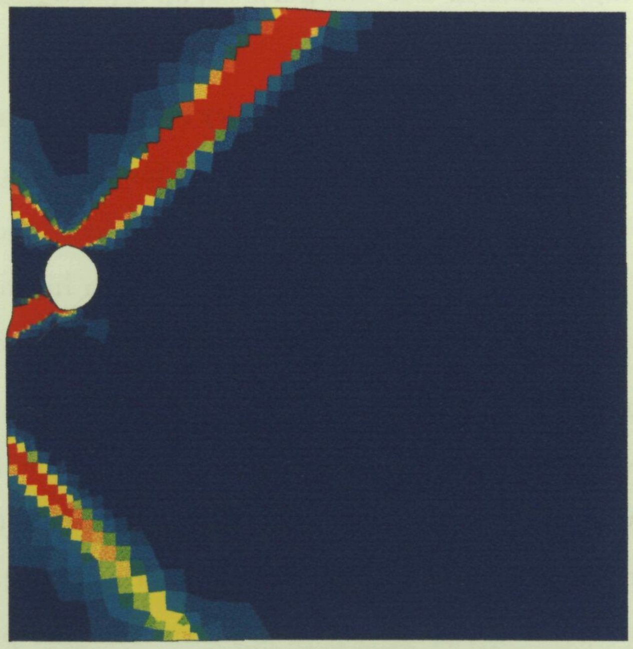
Figure 3.12 Test TWO HOLES. Deformed shape and equivalent plastic strain

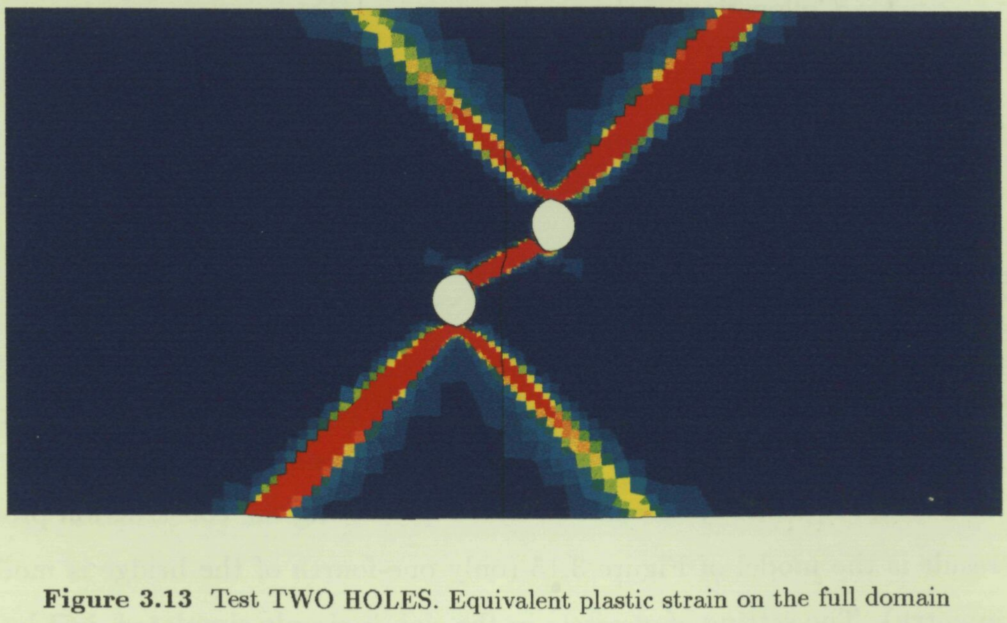
Figure 3.13 Test TWO HOLES. Equivalent plastic strain on the full domain

3.5.5 Test BRIDGE

The goal of this test is to check the object-oriented code and the nonlinear solvers in a realistic engineering situation. For this purpose, an a-posteriori structural analysis of an existing bridge is performed.

The Onze de Setembre bridge is a slab-on-girder bridge, built in Barcelona in 1995. It is a 60m, simply supported, single-span structure. The 6 girders have a variable height (minimum 1.887m – maximum 2.460m), and the thickness of the wings is also variable (minimum 15mm – maximum 30mm). The concrete slab has a constant thickness of 0.20m. Diaphragms are placed every 12m to add torsional stiffness.

Both linear and nonlinear analyses have been made, in cooperation with Miguel Ángel Bretones (Bretones, 1996). In the linear analysis, the real loading test is numerically reproduced. Afterwards, a nonlinear analysis allows to simulate the bridge failure. The fact that a real structure is being analyzed a posteriori has enabled a realistic modelling. All the geometrical definition has been taken from the project plans. Material parameters are obtained from the in-situ quality control testing during construction.

During the linear computation, the whole construction process is simulated. The basic steps are:

1.– Collocation of the steel girders and diaphragms

2.– Metallic formwork for concrete slab

3.– In-situ concreting of the slab

4.– Setting of concrete

5.– Superimposed dead load (paviment...)

6.– Loading test

The object-oriented environment considerably eases this detailed simulation. The various structural elements (girders, diaphragms, slab) are defined as objects, see Figure 3.14, and then sequentially assembled, thus reproducing the construction process. The final result is the model of Figure 3.15 (only one-fourth of the bridge is modelled due to symmetry). The setting of concrete in the slab is simply simulated, just by changing the elasticity modulus, one of the object's attributes.

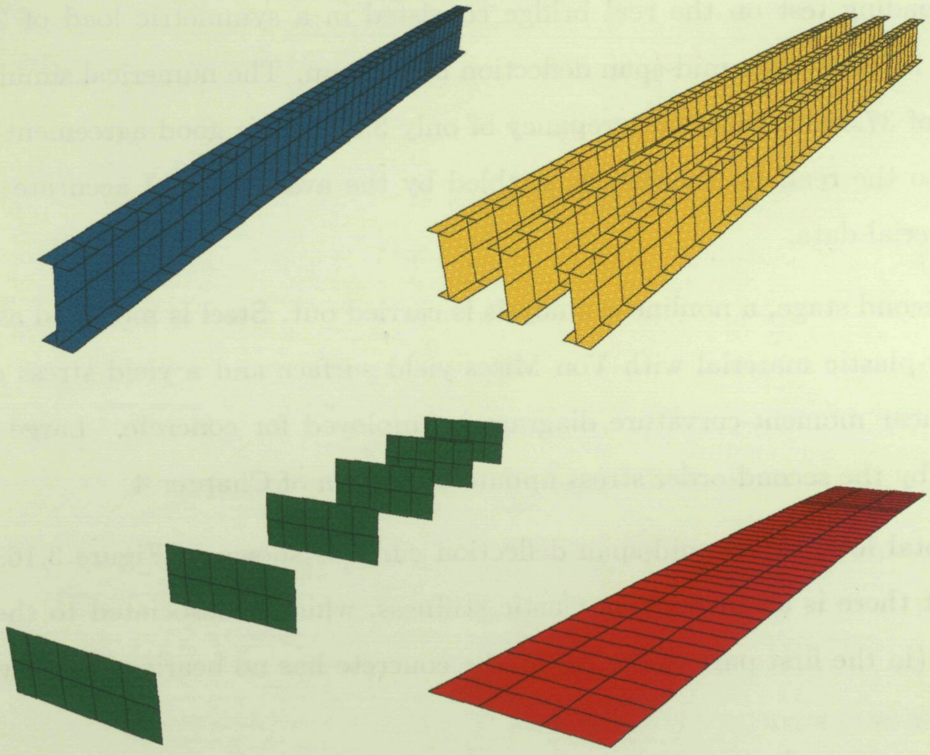

Figure 3.14 Test BRIDGE. Structural elements are defined as objects

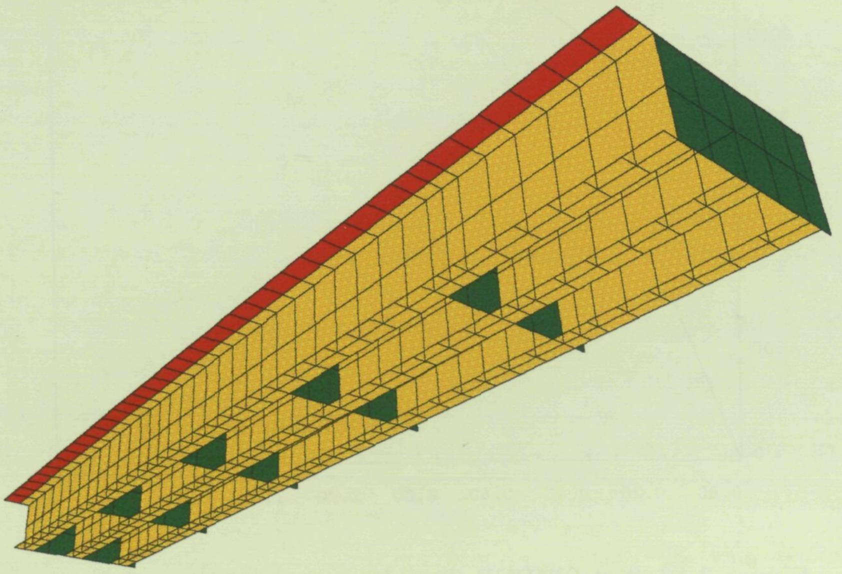

Figure 3.15 Test BRIDGE. Complete model is obtained by assembling elements

The loading test on the real bridge consisted in a symmetric load of 229.5T, and resulted in an average mid-span deflection of 36.2mm. The numerical simulation yields a value of 37.3mm, with a discrepancy of only 3.2%. This good agreement is obtained thanks to the realistic simulation, enabled by the availability of accurate geometrical and material data.

In a second stage, a nonlinear analysis is carried out. Steel is modelled as an elastic-perfectly-plastic material with Von Mises yield surface and a yield stress of 260MPa. A nonlinear moment-curvature diagram is employed for concrete. Large strains are handled by the second-order stress update algorithm of Chapter 4.

The total load versus mid-span deflection curve is shown in Figure 3.16. It can be seen that there is an increase in elastic stiffness, which is associated to the setting of concrete (in the first part of the curve, the concrete has no bearing capacity).

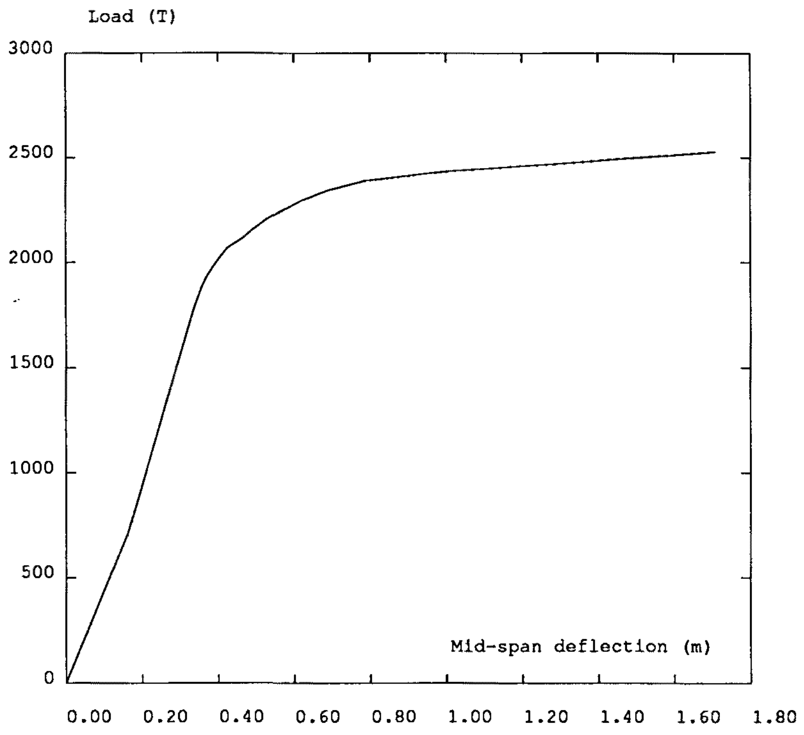

Figure 3.16 Test BRIDGE. Total load versus mid-span deflection

The plastification of the steel girders begins for a total load of 1770T, which is obtained for a live load four times that of the loading test. Figure 3.17 shows the Von Mises stress distribution corresponding to this load level. It is interesting to note that the first zone to plastify is not the mid-span section, but is located under the second diaphragm from mid-span, see Figure 3.17. This effect is caused by the variable girder height and wing thickness. If the load is further increased, the plastification zone rapidly progresses. This can be seen in Figure 3.18, which shows the Von Mises stress for a total load of 2481T.

To reduce to a minimum the computation of stiffness matrices, this problem has been solved with the initial stress method and with the Secant-Newton methods initialized with the elastic stiffness matrix. Figure 3.19 shows the accumulated iterations versus the total load. It can be seen that, up to the inception of plastification (1770T), the four methods behave very similarly. When the load increases, however, differences appear. To reach the load level of Figure 3.18, for instance, the initial stress method takes about twice the number of iterations of the partial Secant-Broyden and Secant-BFGS methods. As in the previous examples, the partial version of the Secant Broyden method outperforms the total version.

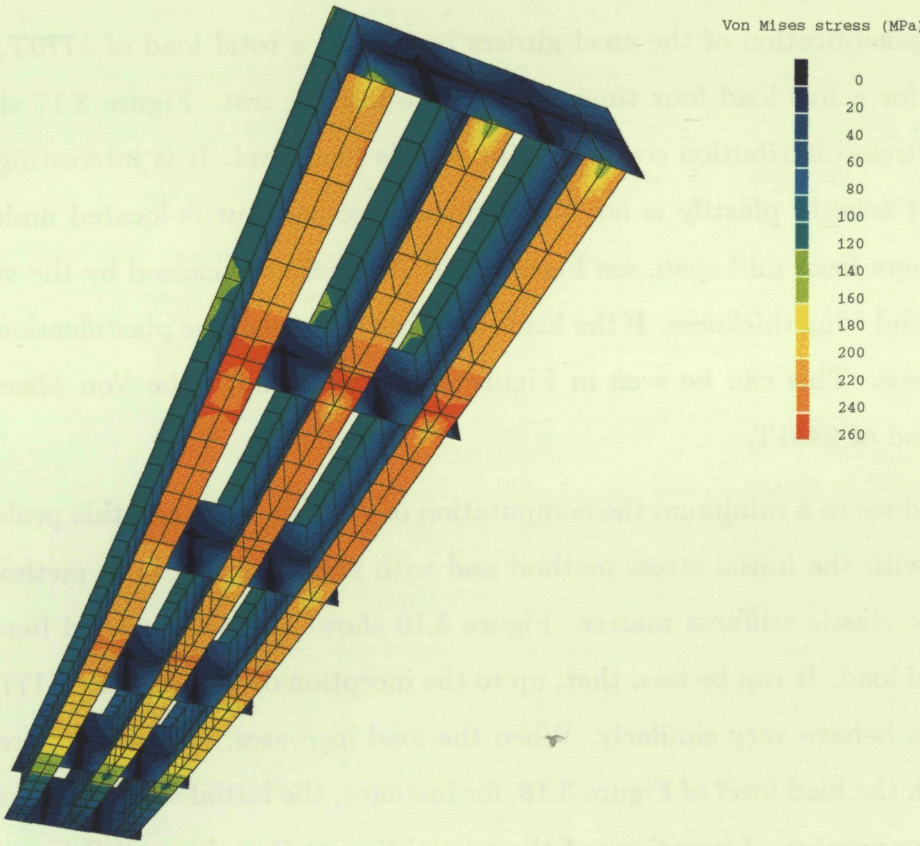

Figure 3.17 Test BRIDGE. Von Mises stress for a total load of 1770T

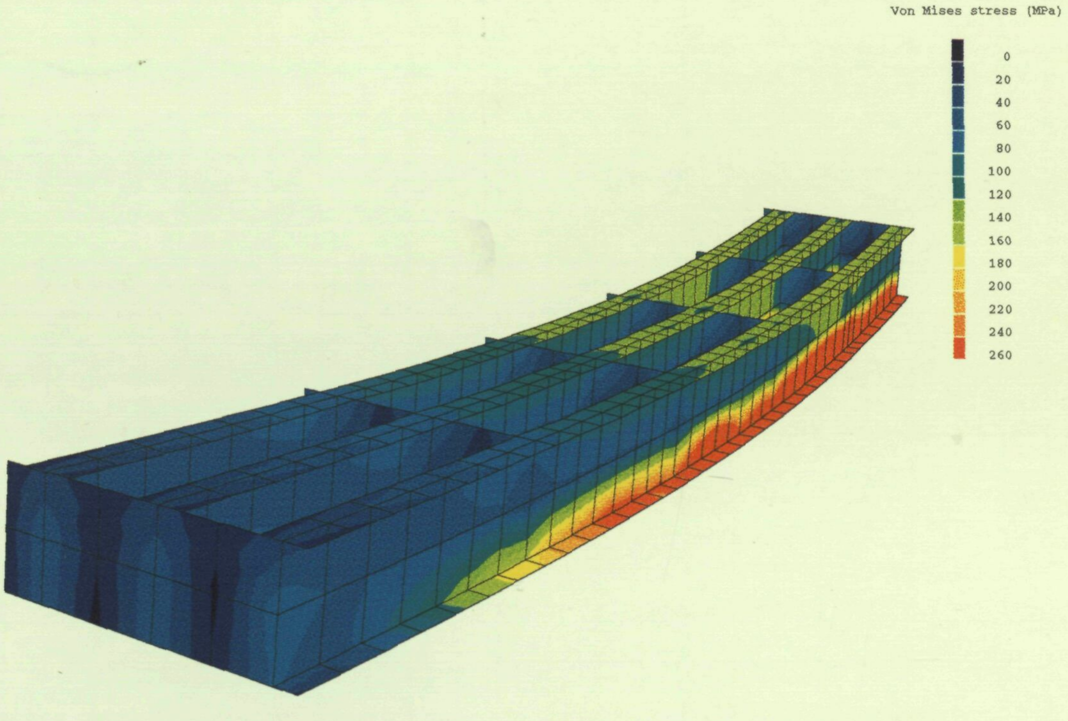

Figure 3.18 Test BRIDGE. Von Mises stress for a total load of 2481T

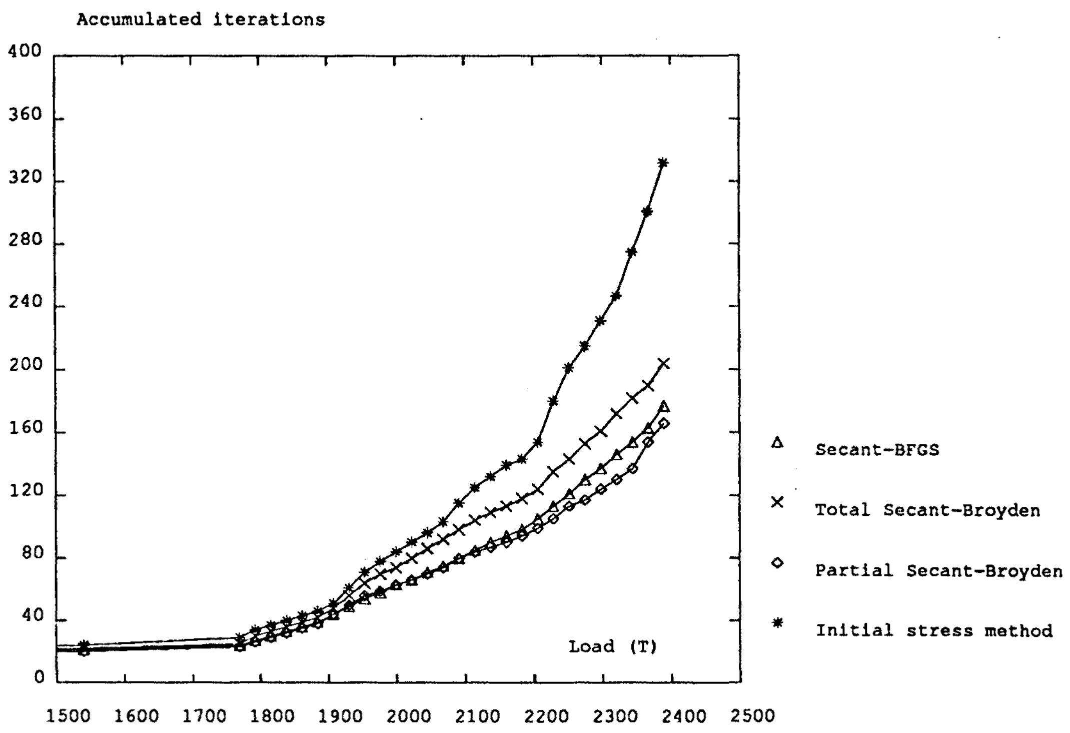

Figure 3.19 Test BRIDGE. Accumulated iterations versus total load

3.6 Concluding remarks

In this Chapter, the most common nonlinear solvers have been adapted to handle linear constraints via the Lagrange-multiplier technique. The original nonlinear system of equations is enlarged by taking the linear constraints as additional equations and the Lagrange multipliers as additional unknowns. The resulting Jacobian matrix consists of four block-matrices, of which only the tangent stiffness matrix is variable.

The adaptation of the Newton-Raphson methods to the Lagrange-multiplier framework is straightforward. Once a stiffness matrix is selected –either the current tangent stiffness matrix (full Newton-Raphson), the tangent stiffness matrix at the beginning of the increment (modified Newton-Raphson) or the elastic stiffness matrix (initial stress method)–, the Jacobian matrix is formed by enlarging it with the constraint matrix that accounts for the linear constraints and the associated reaction forces. A standard iterative process then allows to update simultaneously the displacements and the Lagrange multipliers.

The treatment of Quasi-Newton and Secant-Newton methods is rather more involved. It has been shown that there are two approaches to obtain secant matrices: the natural approach and an alternative approach which profits from the partial linearity of the enlarged system of equations.

The natural approach consists of employing the standard Quasi-Newton equation and the standard additional conditions (which are specific to each particular Quasi-Newton method) to define the updates. It is not required a priori that the resulting secant matrix has the structure of the Jacobian matrix.

With the alternative approach, the secant matrices are explicitly required to maintain the block-structure of the Jacobian matrix. This is achieved by using a modified Quasi-Newton equation and modified additional conditions to obtain secant approximations to the tangent stiffness matrix. This secant matrix is then enlarged with the three constant block-matrices of the Jacobian matrix.

For rank-two Quasi-Newton methods (DFP and BFGS), the two approaches lead to the same algorithms. In other words, if the natural approach is taken, the the secant matrices automatically have the structure of the Jacobian matrix, although it is not requested a priori. The resulting methods, moreover, are basically the same that arise if a Transformation Method is employed to impose the linear constraints.

For Broyden's method, the situation is quite different. The natural approach leads to secant matrices which do not have the structure of the Jacobian matrix. The associated method is called here total Broyden to emphasize that the whole Jacobian matrix is secantly approximated. The alternative approach leads to the partial Broyden method (i.e. only the tangent stiffness matrix is secantly approximated), in which secant matrices respect, by construction, the three constant block-matrices of the Jacobian matrices. This partial Broyden method is the analogue of the Broyden method with linear constraints imposed via a Transformation Method. It has been shown that the Sherman &

Morrison lemma can be employed to produce inverse versions of both Broyden methods.

Various numerical experiments have shown the superiority of the partial Broyden method over the total Broyden method. This is also true for the Secant-Newton simplifications of both methods.

The numerical examples also illustrate that the Lagrange-multiplier technique allows to implement multipoint linear constraints in a simple way, and that object-oriented codes are valid tools both for research and realistic applications.

Chapter 4

TWO STRESS UPDATE ALGORITHMS FOR LARGE STRAIN SOLID MECHANICS

4.1 Introduction

Two basic types of nonlinearity are encountered in nonlinear solid mechanics: material (inelastic constitutive behaviour) and geometric (large strains). Nonlinear material behaviour is typically described by a rate-form constitutive equation, relating the rate of stress to the rate of strain and some internal variables, (Malvern, 1969). If deformations are large, then the changing configuration must be accounted for when stating and handling the rate-form constitutive equation.

In this Chapter, two stress update algorithms for the numerical time-integration of hypoelastic laws in large strain analysis are discussed, and an accuracy analysis of the algorithms is presented. That is, the order of the time discretization error associated to each algorithm is evaluated by means of standard tools in numerical analysis. Therefore, an a priori knowledge of the accuracy for each algorithm is provided.

The accuracy analysis is enabled by the use of a convected frame formalism. Convected frames are attached to the body and deform with it; thus, the statement and time-integration of the constitutive equation becomes a straightforward task. In fact, if convected frames are chosen as reference, any standard finite difference scheme can be employed for the numerical time-integration of the constitutive equation, and the truncation error can be determined in the usual way.

The use of convected frames has also allowed a complete derivation of both algorithms from a general and unified perspective, in spite of the important differences inherent to both methods. The two methods have been previously presented in a fixed frame setting, see Bathe *et al.* (1975) and Pinsky *et al.* (1983). The first algorithm, (Bathe *et al.*, 1975), uses the full strain tensor including quadratic terms to account for large strains, and it is first-order accurate in time. The second algorithm, (Pinsky *et al.*, 1983), employs the same strain measure as in small strain analysis but computed in the midstep configuration. It is second-order accurate in time.

After the error analysis, the two stress update algorithms are adapted to a fixed frame setting in order to be easily implemented in an existing nonlinear finite element code. A set of numerical experiments are then performed to validate the two algorithms, both for elastic and plastic analysis. These numerical tests corroborate the accuracy analysis.

This Chapter is organized as follows. The basic notions of large strain solid mechanics in a convected frame context are reviewed in Section 4.2. Section 4.3 deals with the two stress update algorithms. After some introductory remarks in Subsection 4.3.1, the two algorithms are presented in Subsection 4.3.2. The error analysis is then developed in Subsections 4.3.3 and 4.3.4. The adaptation to Cartesian coordinates and the implementation in a small-strain code is discussed in Section 4.4. Various numerical examples are presented in Section 4.5. Finally, some concluding remarks are made in Section 4.6.

4.2 Large strain solid mechanics in convected frames

4.2.1 Introductory remarks

A common approach in continuum mechanics is to use a fixed orthonormal frame, denoting each spatial point by its Cartesian coordinates with respect to that frame. It is also possible, however, to employ convected material coordinates. By doing so, every material particle is identified, throughout the deformation process, by the same material coordinates, which are convected by the body motion. The reference is then a field of material convected frames, associated to material coordinates at every point.

This represents a fully Lagrangian approach: not only the description focuses on the motion of material particles, but also on a reference field of material convected frames which deform with the body. These convected frames may never be orthonormal. Moreover, according to Truesdell and Toupin (1960), *"the mathematical difficulties which accompany the material description have limited its use to two special ends: problems in one dimension and the proof of general theorems."* However, in exchange for these **difficulties** (which are minor, since only a few, very basic concepts in differential geometry are required), the use of convected frames considerably simplify both the statement and the numerical time-integration of constitutive equations for large strain solid mechanics. For this reason, a convected frame formalism will be employed here to present the two stress update algorithms.

4.2.2 Kinematics

Consider the motion of a deformable body Ω through the usual Euclidean space $\mathbb{R}^3$. The position of its particles is referred to a fixed orthonormal frame (O, e_α), which is the frame of the observer, see Figure 4.1. Let $\boldsymbol{Op} = z^\alpha e_\alpha$ be the current position vector at time t of the material particle initially located at $\boldsymbol{OP} = Z^\alpha e_\alpha$, where the convention of summation on repeated indices is adopted.

The body Ω can be regarded as an ordered set of material points M. In consequence, these points may be related to an embedded mapping manifold $C(\Omega)$ of curvilinear material coordinates x^i. Each material particle M is identified by three material coordinates (x^1, x^2, x^3) which are convected by the body motion, thus remaining "attached" to the particle. For instance, a possible choice of material coordinates is the initial orthonormal coordinates, $x^i = Z^i$.

The motion of the body is expressed by

$$z^\alpha = z^\alpha(x^i, t), \tag{4.1}$$

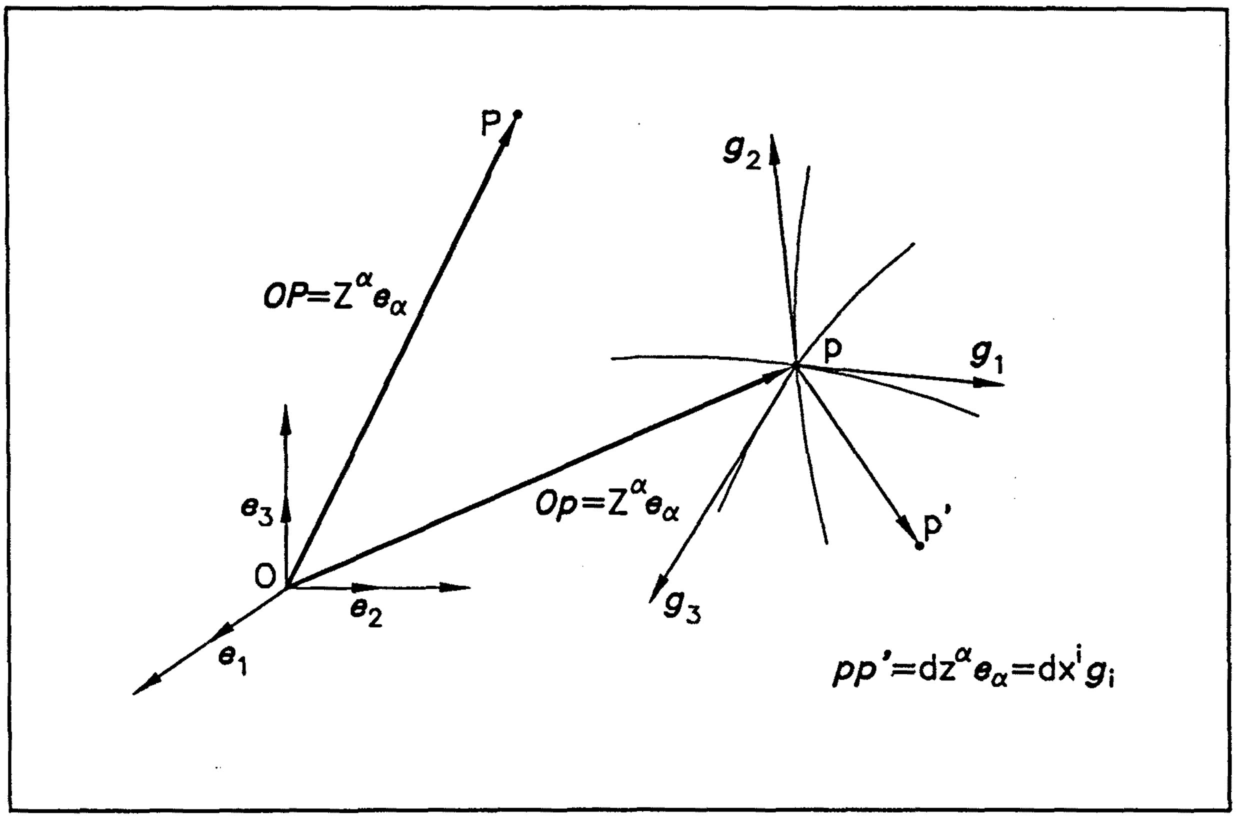
P
$OP=Z^{\alpha}e_{\alpha}$
e_3
O
e_2
e_1
g_2
p
g_1
$Op=Z^{\alpha}e_{\alpha}$
p'
g_3
$pp'=dz^{\alpha}e_{\alpha}=dx^{i}g_{i}$

Figure 4.1 Fixed and material reference frames

which gives the orthonormal coordinates z^{α} at time t of the particle with material coordinates x^{i}. Throughout the Chapter, Latin indices are employed for material components and Greek indices for orthonormal components.

To present the basic features of convected frames, let us focus on two neighbouring material points. At the current time t, consider the material point $M(x^{i})$ located at the geometrical point p. Any geometrical point p' in the vicinity of p may be referenced to the orthonormal frame by

$$pp' = \mathrm{d}z^{\alpha}e_{\alpha}. \tag{4.2}$$

On the other hand, point p' may also be considered the location of the material point $M'(x^{i} + \mathrm{d}x^{i})$, see Figure 4.1. Then, the basis vectors g_{i} of the natural reference frame at the material point $M(x^i)$ are implicitly defined by

$$\boldsymbol{pp'} = \mathrm{d}x^i \boldsymbol{g}_i. \tag{4.3}$$

Combining Eqs. (4.1)-(4.3), the relation between the vectors $\boldsymbol{e}_\alpha$ of the orthonormal frame and the vectors $\boldsymbol{g}_i$ of the natural reference frame field is easily derived as

$$\boldsymbol{g}_i(x^j, t) = \frac{\partial z^\alpha(x^j, t)}{\partial x^i} \boldsymbol{e}_\alpha. \tag{4.4}$$

Starting from the motion defined by Eq. (4.1), the current velocity of the material point $M(x^i)$ is obtained after partial derivation with respect to time as

$$\boldsymbol{v}(x^i, t) = \frac{\partial z^\alpha(x^i, t)}{\partial t} \boldsymbol{e}_\alpha. \tag{4.5}$$

4.2.3 The dragging-along process

The comparison between the definitions of $\boldsymbol{g}_i$ and $\boldsymbol{v}$ given respectively at Eqs. (4.4) and (4.5) allows to derive the fundamental relation

$$\frac{\partial \boldsymbol{g}_i}{\partial t} = \frac{\partial \boldsymbol{v}}{\partial x^i}. \tag{4.6}$$

A graphical interpretation of this relation is shown in Figure 4.2. For simplicity, all material coordinates x^j are fixed except x^i. Two neighbouring material points, $M(x^i)$ and $M'(x^i + \mathrm{d}x^i)$, are under consideration:

i) At current time t, they occupy the geometrical locations $p(x^i, t)$ and $p'(x^i + \mathrm{d}x^i, t)$, and $\|\boldsymbol{pp'}\| = \|\boldsymbol{g}_i(t)\mathrm{d}x^i\|$.

ii) At time $t + \mathrm{d}t$, they occupy the locations $p_+(x^i, t + \mathrm{d}t)$ and $p'_+(x^i + \mathrm{d}x^i, t + \mathrm{d}t)$, with $\|\boldsymbol{pp}_+\| = \|\boldsymbol{v}\mathrm{d}t\|$ and $\|\boldsymbol{p}_+\boldsymbol{p}'_+\| = \|\boldsymbol{g}_i(t + \mathrm{d}t)\mathrm{d}x^i\|$.

It can be seen in Figure 4.2 how the vector field $\boldsymbol{v}\mathrm{d}t$, which transforms p into p_+ and p' into p'_+, also transforms the vector $\boldsymbol{g}_i(x^j, t)$ into the vector $\boldsymbol{g}_i(x^j, t + \mathrm{d}t)$. In other

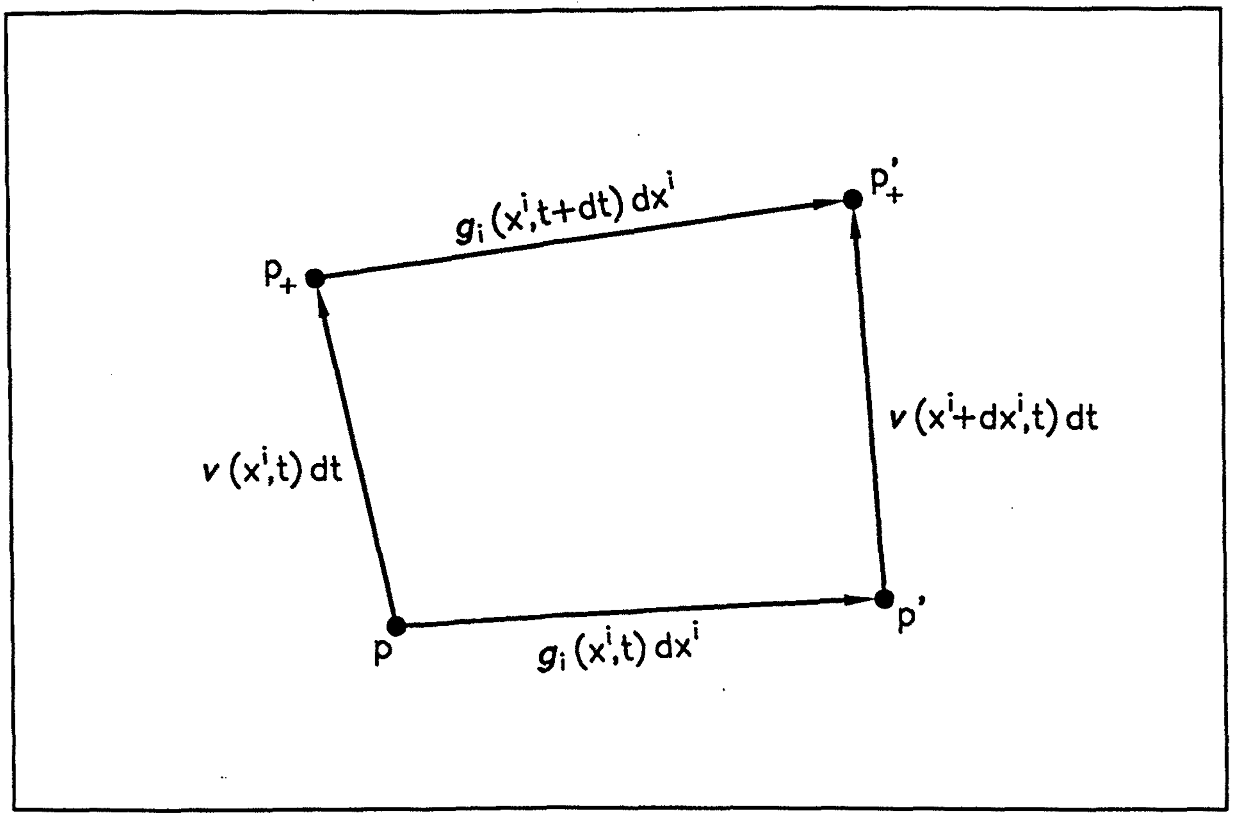

Figure 4.2 Dragging-along process of the reference frame field

words, g_i is dragged along (or convected) by the vector field $\boldsymbol{v}dt$, hence by the motion itself. Therefore, the natural reference frame is convected by the body motion.

The main consequence, underlined by Oldroyd (1950), is the following: *"the device of representing all tensor quantities (say A) associated with the material by their convected components (say A^{ij}) allows similar quantities associated with the same material point at different times to be added, component by component, to give a similar tensor quantity as the sum."*

For instance, in the current reference frame field (M, g_i), the quantities $\int_{t_1}^{t} A^{ij}(t')dt'$ and $\partial A^{ij}/\partial t$ are still components of a tensor field, although they involve adding or subtracting components A^{ij} at different instants. This situation is in sharp contrast with the one encountered with a fixed orthonormal frame. If A is represented by its orthonormal components $A^{\alpha\beta}$, then $\int_{t_1}^{t} A^{\alpha\beta}(t')\mathrm{d}t'$ and $\partial A^{\alpha\beta}/\partial t$ are **not** components of a tensor field. The basic idea is that $A^{\alpha\beta}(t_1)$ and $A^{\alpha\beta}(t_2)$ are referred to different configurations, and they must be transformed into a common configuration before they can be added or subtracted, see Pinsky *et al.* (1983).

This different behaviour makes material coordinates especially attractive for the development and numerical treatment of integro-differential constitutive relationships, which must be **objective** (that is, meaningful for the body, but independent from the observer's viewpoint). Indeed, by employing material coordinates which are convected by body motion itself, all the discussions about the verification of the **principle of material frame-indifference** (see Truesdell and Noll, 1965), also called **objectivity principle**, a priori disappear.

Regarding the numerical treatment, any rate-form constitutive equation may be integrated in time using any standard finite difference scheme, because all the quantities appearing in these schemes have a tensorial meaning related to the body and can be directly combined, as pointed out above.

4.2.4 The strain and rate-of-deformation tensors

Once kinematics and the dragging-along process have been presented, the next required ingredient is a measure of strain and strain rate. The starting point is the current separation $\mathrm{d}s$ between two neighbouring material points $M(x^i)$ and $M'(x^i + \mathrm{d}x^i)$. Combining Eqs. (4.2) and (4.3), $\mathrm{d}s$ can be represented as

$$(\mathrm{d}s)^2 = \boldsymbol{pp'} \cdot \boldsymbol{pp'} = \mathrm{d}z^\alpha \mathrm{d}z^\alpha = (\boldsymbol{g}_i \cdot \boldsymbol{g}_j)\mathrm{d}x^i \mathrm{d}x^j = g_{ij}\mathrm{d}x^i \mathrm{d}x^j, \tag{4.7}$$

where g_{ij} are the covariant components of the metric tensor field G in the convected frame. An important scalar associated to G is the ratio of spatial unit volume to material unit volume $\sqrt{g}$, (Pegon & Guélin, 1986), given by

$$\sqrt{g} = \sqrt{\det(g_{ij})}. \tag{4.8}$$

Both G and $\sqrt{g}$ contain information about the current configuration of the body. Note, for instance, that if the initial orthonormal coordinates are taken as material coordinates ($x^i = Z^i$), then in the initial configuration G is the identity tensor and $\sqrt{g}$ is one, thus reflecting an undeformed state.

The deformation between the initial time $t = 0$, which is chosen as a reference, and the current time t can be obtained by comparing the two separations $\mathrm{d}s(t)$ and $\mathrm{d}s(0)$. Recalling the definition of the metric tensor, Eq. (4.7), the difference between the two separations is expressed as

$$[\mathrm{d}s(t)]^2 - [\mathrm{d}s(0)]^2 = \left[g_{ij}(t) - g_{ij}(0)\right] \mathrm{d}x^i \mathrm{d}x^j = 2\, \varepsilon_{ij}(t) \mathrm{d}x^i \mathrm{d}x^j . \tag{4.9}$$

Thanks to the dragging-along process, see Figure 4.2, $g_{ij}(0)$ is a quantity memorized at time $t = 0$ but which still has a tensorial meaning at time t, so it can be subtracted from $g_{ij}(t)$ to define the components of a meaningful strain tensor $\boldsymbol{\varepsilon}$, which is in fact the well-known Almansi-Euler strain tensor.

The rate-of-deformation tensor $\boldsymbol{d}$ can be defined in the current reference frame field $(M, \boldsymbol{g}_i)$ simply by taking the time derivative of $\varepsilon_{ij}(t)$,

$$d_{ij}(t) = \frac{\partial \varepsilon_{ij}(t)}{\partial t} = \frac{1}{2}\frac{\partial g_{ij}(t)}{\partial t}. \tag{4.10}$$

It is possible to show that $\boldsymbol{d}$ is indeed the symmetric part of the velocity gradient in the convected frame. If such definition of $\boldsymbol{d}$ is adopted, it is possible to define the strain tensor $\boldsymbol{\varepsilon}$ a posteriori according to

$$\varepsilon_{ij}(t) = \int_0^t d_{ij}(t')\mathrm{d}t'. \tag{4.11}$$

Note that, as commented previously, the use of material components to represent tensors has enabled a straightforward definition of strain and strain rate measures, Eqs. (4.9)-(4.11).

4.2.5 The stress tensor

The next essential ingredient is a stress measure. To describe the current stress state at the material point $M(x^i)$, the Cauchy analysis in curvilinear coordinates leads to the introduction of the **relative** stress tensor $\boldsymbol{\sigma}(M,t)$, (Brillouin, 1949). In the fixed orthonormal frame, the components $\sigma^{\alpha\beta}$ of $\boldsymbol{\sigma}$ are those of the classical Cauchy stress tensor. The material components σ^{ij} are related to $\sigma^{\alpha\beta}$ by the transformation rule

$$\sigma^{ij} = \sqrt{g}\frac{\partial x^i}{\partial z^\alpha}\frac{\partial x^j}{\partial z^\beta}\sigma^{\alpha\beta}. \tag{4.12}$$

4.2.6 The governing equations

For a wide range of problems in solid mechanics, it is a common assumption that *i)* mechanical and thermal effects are uncoupled and *ii)* the density is a constant. The mechanical problem then involves the momentum balance and the constitutive equation.

The weak form of the momentum balance is the so-called balance of virtual power, which equates the virtual power of internal forces and the virtual power of external forces, (Hughes, 1987; Zienkiewicz & Taylor, 1991). These virtual powers are global quantities, obtained through integration over the whole domain. The convected material approach allows to merge two attractive features: dealing with **current** quantities associated with the **current** state of the body Ω and, at the same time, being attached to a **time invariant** mapping $C(\Omega)$ of material coordinates. The integration domain is therefore $C(\Omega)$ throughout the whole body motion.

As for the constitutive equation, it is typically written in rate form, relating a stress rate to a certain measure of strain rate, (Malvern, 1969). If material coordinates are employed, a stress rate tensor can be defined in the current reference frame field $(M, \boldsymbol{g}_i)$ simply by taking the time derivative of σ^{ij},

$$\mathcal{T}(\sigma)^{ij} = \frac{\partial \sigma^{ij}}{\partial t}. \tag{4.13}$$

This is the well-known Truesdell stress rate, see Truesdell (1953). This particular rate was introduced because, very physically, during the dragging-along process of the stress it gives priority to the resultant force **vector** rather than to the stress **tensor** itself, see also Truesdell (1955). Many other stress rates have been employed in nonlinear solid mechanics, but a review of the various choices is beyond the scope of this work.

The Truesdell rate can be employed to write a general nonlinear rate-form constitutive equation as

$$\mathcal{T}(\sigma)^{ij} = A^{ij}(\boldsymbol{\sigma}, \boldsymbol{v}), \tag{4.14}$$

where $\boldsymbol{A}$ may depend both on the strain rate (symbolically represented in Eq. (4.14) by velocity $\boldsymbol{v}$) and the state of stress (and, eventually, some internal variables). A particular case of this general expression is the hypoelastic constitutive law

$$\mathcal{T}(\sigma)^{ij} = C^{ijkl} d_{kl}, \tag{4.15}$$

which states the linear dependence of the Truesdell rate on the rate-of-deformation tensor. C^{ijkl} are the material components of the fourth-order elastic modulus tensor.

As commented previously, the use of convected frames guarantees a priori the objectivity of the constitutive equation (4.14), because it is stated in terms of material components (i.e., associated to the body and not to the observer's frame). Numerical time-integration is also straightforward. The stress rate can be discretized, by means of a simple finite difference formula, into

$$\mathcal{T}(\sigma)^{ij}(t_{n+\theta}) \approx \frac{\sigma^{ij}(t_{n+1}) - \sigma^{ij}(t_n)}{\Delta t}, \tag{4.16}$$

where $t_{n+\theta}$ is an intermediate instant between t_n and $t_{n+1} = t_n + \Delta t$. Again, the use of material components is essential to allow $\sigma^{ij}(t_{n+1})$ and $\sigma^{ij}(t_n)$ to be subtracted and yield meaningful tensorial quantities. This is not the situation if a fixed orthonormal frame is employed: as shown in Pinsky *et al.* (1983), Rodríguez-Ferran & Huerta (1996) and in Section 4.4, $\sigma^{\alpha\beta}(t_{n+1})$ and $\sigma^{\alpha\beta}(t_n)$ have to be transformed into a common configuration before they can be subtracted, thus making the time-integration of the constitutive equation more elaborate.

4.3 Two stress update algorithms for large strains

4.3.1 Introductory remarks

The FEM is commonly employed to solve the weak form of the momentum balance. If inertia terms are neglected (quasistatic process), the displacements $\boldsymbol{u}$ are employed as the fundamental unknowns, and the Lagrange-multiplier technique is chosen to impose the boundary conditions, see Chapter 3, the resulting system of nonlinear equations is, Eq. (3.5),

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}, \tag{4.17}$$

where $\boldsymbol{x}$ and $\boldsymbol{r}$ are the enlarged vectors of unknowns and residuals, respectively.

As commented in Chapter 3, Eq. (4.17) is typically solved in an incremental manner. The vector of unknowns is then $\Delta\boldsymbol{x} = (\Delta\boldsymbol{u}, \Delta\boldsymbol{\lambda})$ from one (known) equilibrium configuration of the body at time t_n to a new (unknown) equilibrium configuration at time t_{n+1}, a time-increment Δt later. The resulting nonlinear system at each increment may be solved by a number of iterative techniques, see Chapter 3 and Crisfield (1991), Dennis & Schnabel (1983), Ortega & Rheinboldt (1970). The two key ideas are that *i)* a linearized form of Eq. (4.17) is used to get an initial prediction and successive corrections to $\Delta\boldsymbol{x}$ and *ii)* the constitutive equation (4.14) must be integrated over the time-increment after each iteration to get the stresses at t_{n+1} and check the equilibrium. In fact, the time-integration of the constitutive equation is a key point in nonlinear solid mechanics and affects fundamentally the accuracy of the global solution, (Hughes, 1984).

4.3.2 Two stress update algorithms

Two algorithms for the numerical time-integration of the constitutive equation will be presented in this Subsection. An intuitive approach will be made. First, a restriction will be imposed on the problem, to get a particular version of the algorithms in a straightforward manner. Then the restriction will be removed, and the general expressions of the algorithms will be obtained. A more rigorous course will be followed in the next Subsection, where an error analysis for both algorithms is performed.

Profiting from the simple expression of the Truesdell rate in material components, Eq. (4.13), the rate-form constitutive equation (4.14) can be integrated over the time-increment. For the hypoelastic equation (4.15), for instance, it yields

$$\int_{t_n}^{t_{n+1}} \frac{\partial \sigma^{ij}(t)}{\partial t} \mathrm{d}t = {}^{n+1}\sigma^{ij} - {}^{n}\sigma^{ij} = \int_{t_n}^{t_{n+1}} C^{ijkl}(t) d_{kl}(t) \mathrm{d}t, \tag{4.18}$$

where left superscripts denote time (i.e., ${}^{n}\sigma^{ij} \equiv \sigma^{ij}(t_n)$).

Assume for the moment that the material components C^{ijkl} are constant. With this simplifying restriction, the RHS of Eq. (4.18) can be put, recalling Eq. (4.11), as

$$C^{ijkl} \int_{t_n}^{t_{n+1}} d_{kl}(t) \mathrm{d}t = C^{ijkl} \left[{}^{n+1}\varepsilon_{kl} - {}^{n}\varepsilon_{kl} \right] = C^{ijkl} \Delta\varepsilon_{kl}, \tag{4.19}$$

where $\Delta\varepsilon_{kl}$ are the material components of the increment of strain tensor. Recalling the definition of the strain tensor in Eq. (4.9), the strain increment $\Delta\varepsilon_{kl}$ can be obtained as

$$\Delta\varepsilon_{kl} = \frac{1}{2} \left[{}^{n+1}g_{kl} - {}^{n}g_{kl} \right] = \frac{1}{2} \left[\frac{\partial\, {}^{n+1}z^{\alpha}}{\partial x^k} \frac{\partial\, {}^{n+1}z^{\alpha}}{\partial x^l} - \frac{\partial\, {}^{n}z^{\alpha}}{\partial x^k} \frac{\partial\, {}^{n}z^{\alpha}}{\partial x^l} \right] \tag{4.20}$$

Remark 4.1: The increment of strain $\Delta\varepsilon_{kl}$ is a purely kinematical quantity, independent from material behaviour.

Remark 4.2: Equation (4.20) allows to compute $\Delta\varepsilon_{kl}$ from the orthonormal coordinates ${}^{n}z^{\alpha}$ and ${}^{n+1}z^{\alpha}$, which are available in the incremental analysis. No assumption about the body motion between t_n and t_{n+1} is required.

Combining Eqs. (4.18) and (4.19), a stress update algorithm can be written as

$${}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + C^{ijkl} \Delta\varepsilon_{kl}. \tag{4.21}$$

with $\Delta\varepsilon_{kl}$ computed according to Eq. (4.20). It must be remarked that, in this particular case of constant material components C^{ijkl}, the algorithm given by Eq. (4.21)

has no time-discretization error, because it has been obtained through exact time-integration of the constitutive equation, with no need of numerical time-integration.

In many cases, however, the material components of C are not constant. A common assumption in computational mechanics is that of constant **orthonormal** components $C^{\alpha\beta\gamma\delta}$, written in terms of the Lamé constants λ and μ as

$$C^{\alpha\beta\gamma\delta} = \lambda\delta_{\alpha\beta}\delta_{\gamma\delta} + \mu\left(\delta_{\alpha\gamma}\delta_{\beta\delta} + \delta_{\alpha\delta}\delta_{\beta\gamma}\right), \tag{4.22}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta. If the orthonormal components $C^{\alpha\beta\gamma\delta}$ are constant, the material components C^{ijkl} are not. That means that the stress update algorithm (4.21) must be modified by specifying when C^{ijkl} is evaluated. Two possible choices, resulting in two stress update algorithms, will be shown next.

First algorithm

The first choice is to evaluate C^{ijkl} at the beginning of the increment, t_n. The increment of strain $\Delta\varepsilon_{kl}$ presented in Eq. (4.20) is written in an **equivalent** form, using only quantities at t_n, after substitution of ${}^{n+1}z^\alpha = {}^{n}z^\alpha + \Delta z^\alpha$ in Eq. (4.20). The stress algorithm is then:

$${}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n}C^{ijkl}\Delta\varepsilon_{kl}. \tag{4.23a}$$

$$\Delta\varepsilon_{kl} = \frac{1}{2}\left[\frac{\partial^n z^\alpha}{\partial x^k}\frac{\partial(\Delta z^\alpha)}{\partial x^l} + \frac{\partial(\Delta z^\alpha)}{\partial x^k}\frac{\partial^n z^\alpha}{\partial x^l} + \frac{\partial(\Delta z^\alpha)}{\partial x^k}\frac{\partial(\Delta z^\alpha)}{\partial x^l}\right]. \tag{4.23b}$$

Remark 4.3: The expressions of $\Delta\varepsilon_{kl}$ in Eqs. (4.20) and (4.23b) are **equivalent**, so the strain increment employed in the first algorithm is still the **exact** one.

Remark 4.4: However, the stress update algorithm is no longer exact. Contrary to what happened in the particular case of constant C^{ijkl}, Eq. (4.21), the last term in Eq. (4.23a) is not the exact value of the stress increment, but an approximation affected by a time-discretization error. An error analysis will be performed in the next Subsection.

Second algorithm

An intuitively better approach is a time-centered scheme, where quantities are evaluated at the midstep instant $t_{n+\frac{1}{2}}$. However, since an incremental strategy has been adopted, the only available kinematical data are the orthonormal coordinates at the beginning and the end of the time-step. Nothing is known about the body motion between t_n and t_{n+1}. As a consequence, an hypothesis is needed to evaluate quantities at $t_{n+\frac{1}{2}}$. A common choice is to assume that the velocity is **constant** within the time-step, $\bar{v}^\alpha = \Delta z^\alpha / \Delta t$. The midstep configuration is then

$$^{n+\frac{1}{2}}\bar{z}^\alpha = {}^n z^\alpha + \frac{1}{2}\Delta z^\alpha = {}^{n+1}z^\alpha - \frac{1}{2}\Delta z^\alpha. \tag{4.24}$$

The bar is used to emphasize that the expressions of $\bar{v}^\alpha$ and $^{n+\frac{1}{2}}\bar{z}^\alpha$ are based on the constant-velocity assumption. Equation (4.24) allows to express the strain increment given by Eq. (4.20) in an **equivalent** form, employing midstep orthonormal coordinates. The stress update algorithm is

$$^{n+1}\sigma^{ij} = {}^n\sigma^{ij} + {}^{n+\frac{1}{2}}\overline{C}^{ijkl}\Delta\varepsilon_{kl}. \tag{4.25a}$$

$$\Delta\varepsilon_{kl} = \frac{1}{2}\left[\frac{\partial^{n+\frac{1}{2}}\bar{z}^\alpha}{\partial x^k}\frac{\partial(\Delta z^\alpha)}{\partial x^l} + \frac{\partial(\Delta z^\alpha)}{\partial x^k}\frac{\partial^{n+\frac{1}{2}}\bar{z}^\alpha}{\partial x^l}\right], \tag{4.25b}$$

where $^{n+\frac{1}{2}}\overline{C}^{ijkl}$ are the material components of the modulus tensor computed at the midstep under the constant-velocity assumption.

Remark 4.5: The constant-velocity assumption is just a computational convenience that allows to compute the **exact** strain increment in terms of midstep quantities, Eq. (4.25b). In fact, no restriction is placed on the **true** body motion between t_n and t_{n+1} and, similarly to *Remark 4.3*, the expressions of $\Delta\varepsilon_{kl}$ in Eqs. (4.20) and (4.25b) are **equivalent**.

Remark 4.6: The stress update algorithm given by Eqs. (4.25), however, is not exact. As remarked for the first algorithm, the last term in Eq. (4.25a) is not the exact stress increment, but only an approximation.

Remark 4.7: The two stress update algorithms, Eqs. (4.23) and (4.25), are **identical** if the material components of the modulus tensor are constant, Eq. (4.21). They differ, however, for variable C^{ijkl}. The error analysis of the next Subsection shows that the time-centered strategy of the second algorithm is indeed more accurate than the forward scheme of the first algorithm.

The two presented algorithms treat rigid rotations correctly. Indeed, if the body undergoes a rigid rotation, the separation between its particles does not change, thus resulting in a null strain increment, Eq. (4.9). Since the two algorithms employ the exact strain increment, Eqs. (4.23b) and (4.25b), they both correctly predict no variation in the material stress components, ${}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij}$. This behaviour of a stress update algorithm is trivially verified if a convected frame is employed but must be carefully checked if a fixed reference frame is used, see Pinsky *et al.* (1983) and Rodríguez-Ferran & Huerta (1994,1996). In the literature this behaviour is often referred to as **incremental objectivity**, (Hughes, 1984; Hughes & Winget, 1980). However, the strain increment $\Delta\varepsilon_{kl}$ being null does not necessarily imply the motion between t_n and t_{n+1} to be a rigid rotation. Infinitely many other paths are possible, including such simple ones as the parallel translation of all the material points, depicted in Figur^ 4.3, which may cause loading/unloading effects. Since an incremental analysis is performed, nothing is known about the body motion during the time-step. As a consequence, **assuming** a rigid rotation and demanding the numerical algorithm to perform in accordance with that choice is an observer's choice. Although this choice may be very reasonable, it has, contrary to the principle of objectivity, no physical justification. For this reason, the term **priority on rigid motion** (i.e., rigid rotations are assumed when possible) is preferred here.

Moreover, the treatment of rigid rotations is not the only concern. Although both algorithms behave identically for this particular test, the error analysis and the numerical examples show that the general behaviour is quite different. These differences are in accuracy, on the numerical implementation and, also, on the response to several

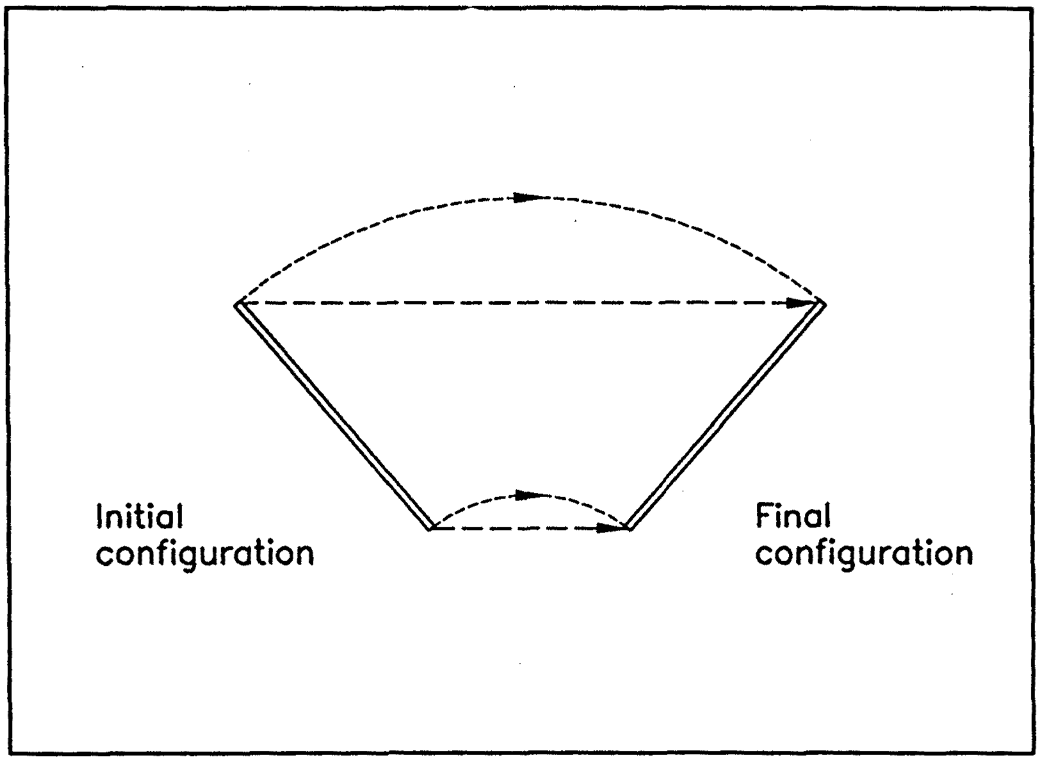

Figure 4.3 Two possible paths from an initial to a final configuration

non-rigid deformation paths different from simple rigid rotation. Thus, the performance of large strain algorithms must be evaluated beyond rigid motion into a more general error analysis.

It is important to note that both algorithms can be adapted to handle rate-form elastoplastic constitutive equations. As commented in Pinsky *et al.* (1983), the key idea is to profit the additive decomposition of the rate-of-deformation tensor $\boldsymbol{d}$ into elastic and plastic parts to perform an operator split of the constitutive equation.

4.3.3 Error analysis of the algorithms

The two algorithms are compared with the help of various numerical tests in Section 4.5. The basic goal of the comparison is to show with numerical experiments the superior performance of the second algorithm. A theoretical justification of such behaviour is presented here, where the error associated to the time discretization is analyzed. The analysis shows that the second algorithm is second-order accurate in time, while the first one is only first-order accurate.

To perform the error analysis, a more rigorous approach than in the presentation of the algorithms is followed, accounting from the start with variable material components C^{ijkl}. If finite differences are employed to discretize the hypoelastic equation (4.15), the midpoint rule renders

$$\frac{\partial \sigma^{ij}}{\partial t}(t_{n+\frac{1}{2}}) = {}^{n+\frac{1}{2}}C^{ijkl}\ {}^{n+\frac{1}{2}}d_{kl} = \frac{{}^{n+1}\sigma^{ij} - {}^{n}\sigma^{ij}}{\Delta t} + \mathcal{O}(\Delta t^2), \tag{4.26}$$

which can be rearranged into

$${}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n+\frac{1}{2}}C^{ijkl}\left(\Delta t\ {}^{n+\frac{1}{2}}d_{kl}\right) + \mathcal{O}(\Delta t^3). \tag{4.27}$$

The constant-velocity assumption has not been introduced in Eq. (4.27), so ${}^{n+\frac{1}{2}}C^{ijkl}$ and ${}^{n+\frac{1}{2}}d_{kl}$ are the true midstep values. It is shown in the next Subsection that $\Delta t\ {}^{n+\frac{1}{2}}d_{kl}$ is a third-order approximation to the strain increment $\Delta\varepsilon_{kl}$,

$$\Delta\varepsilon_{kl} = \Delta t\ {}^{n+\frac{1}{2}}d_{kl} + \mathcal{O}(\Delta t^3). \tag{4.28}$$

Note that, since ${}^{n+\frac{1}{2}}d_{kl}$ is an instantaneous quantity which does not depend on Δt, Eq. (4.28) also states that the strain increment is $\mathcal{O}(\Delta t)$.

Substituting Eq. (4.28) into Eq. (4.27) gives

$${}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n+\frac{1}{2}}C^{ijkl}\Delta\varepsilon_{kl} + \mathcal{O}(\Delta t^3). \tag{4.29}$$

Apart from the error term, which was not accounted for in the presentation of the algorithms, the only difference between Eq. (4.29) and the algorithms, Eqs. (4.23) and (4.25), is in the material components of the modulus tensor. The error analysis is completed by studying the effect of replacing ${}^{n+\frac{1}{2}}C^{ijkl}$ in Eq. (4.29) by ${}^{n}C^{ijkl}$ (first algorithm) or by ${}^{n+\frac{1}{2}}\overline{C}^{ijkl}$ (second algorithm).

First-order algorithm

A first-order Taylor's expansion shows that

$$ {}^{n+\frac{1}{2}}C^{ijkl} = {}^{n}C^{ijkl} + \mathcal{O}(\Delta t). \tag{4.30}$$

Combining Eqs. (4.28), (4.29) and (4.30) results in

$$ {}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n}C^{ijkl}\Delta\varepsilon_{kl} + \mathcal{O}(\Delta t^2). \tag{4.31}$$

Equation (4.31) only differs from the first stress update algorithm, Eqs. (4.23), on the error term. Since this error is order 2 for one time-step (from t_n to t_{n+1}), the global error is order 1. In conclusion, **the first algorithm is first-order accurate in time.**

Second-order algorithm

As shown in Appendix 4.A, the constant-velocity assumption provides second-order approximation in the midstep material components of the modulus tensor,

$$ {}^{n+\frac{1}{2}}C^{ijkl} = {}^{n+\frac{1}{2}}\overline{C}^{ijkl} + \mathcal{O}(\Delta t^2). \tag{4.32}$$

Substituting Eq. (4.32) into Eq. (4.29), and employing the expression of $\Delta\varepsilon_{kl}$ in Eq. (4.28) yields

$$ {}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n+\frac{1}{2}}\overline{C}^{ijkl}\Delta\varepsilon_{kl} + \mathcal{O}(\Delta t^3), \tag{4.33}$$

which, except for the error term, coincides with the second algorithm, Eq. (4.25). Since the error is order 3 for one time-step, the global error is order 2. In conclusion, **the second algorithm is second-order accurate in time.**

4.3.4 The strain increment and the midpoint rule

The superiority of a time-centered approach (second-order algorithm) over a forward scheme (first-order algorithm) is illustrated by the error analysis just presented. As a further justification, it is shown here that, if a generalized trapezoidal rule is employed to integrate the hypoelastic equation (4.15),

$$ {}^{n+1}\sigma^{ij} = {}^{n}\sigma^{ij} + {}^{n+\theta}C^{ijkl}\left(\Delta t \; {}^{n+\theta}d_{kl}\right); \qquad 0 \le \theta \le 1, \tag{4.34} $$

the midpoint rule ($\theta = 1/2$), in combination with the constant-velocity assumption, is the only one which works with the **exact** strain increment.

The material components of the rate-of-deformation tensor defined in Eq. (4.10) can be put more explicitly, considering Eqs. (4.4), (4.6) and (4.7), as

$$ d_{kl}(t) = \frac{1}{2}\left[\frac{\partial z^\alpha(t)}{\partial x^k}\frac{\partial v^\alpha(t)}{\partial x^l} + \frac{\partial v^\alpha(t)}{\partial x^k}\frac{\partial z^\alpha(t)}{\partial x^l}\right]. \tag{4.35} $$

If the velocity is assumed constant within the time-step, $\bar{v}^\alpha = \Delta z^\alpha/\Delta t$, the body motion from t_n to t_{n+1} is represented by

$$ {}^{n+\theta}\bar{z}^\alpha = {}^{n}z^\alpha + \theta\Delta z^\alpha, \qquad 0 \le \theta \le 1. \tag{4.36} $$

By modifying Eq. (4.35) according to Eq. (4.36), the approximation to the rate-of-deformation based on the constant-velocity assumption can be written as

$$ {}^{n+\theta}\bar{d}_{kl} = \frac{1}{2}\frac{1}{\Delta t}\left[\frac{\partial^n z^\alpha}{\partial x^k}\frac{\partial(\Delta z^\alpha)}{\partial x^l} + \frac{\partial(\Delta z^\alpha)}{\partial x^k}\frac{\partial^n z^\alpha}{\partial x^l} + 2\theta\frac{\partial(\Delta z^\alpha)}{\partial x^k}\frac{\partial(\Delta z^\alpha)}{\partial x^l}\right]. \tag{4.37} $$

Comparing Eq. (4.37) to the **exact** expression of the strain increment employed in Eq. (4.23b) gives

$$ \Delta\varepsilon_{kl} = \Delta t \; {}^{n+\theta}\bar{d}_{kl} \quad \Longleftrightarrow \quad \theta = \frac{1}{2}. \tag{4.38} $$

Since $\Delta t \; {}^{n+\theta}\bar{d}_{kl}$ is employed as an approximation to the strain increment in the trapezoidal rule, see Eq. (4.34), it can be concluded from Eq. (4.38) that the midpoint rule ($\theta = 1/2$) is the only one which allows to compute the **exact** strain increment.

Remark 4.8: Because of Eq. (4.38), the second-order algorithm, Eq. (4.25), can be interpreted as a direct application of the midpoint rule.

Remark 4.9: Since Eq. (4.38) states that the midpoint rule is the only one which allows to compute the exact strain increment, the first-order algorithm, Eqs. (4.23), is not deduced from the general trapezoidal rule particularized at $\theta = 0$. In an effort to obtain better accuracy, the first-order algorithm employs the **exact** strain increment expressed in a convenient form, Eq. (4.23b), instead of a forward-in-time **approximation** of $\Delta\varepsilon_{kl}$, such as $\Delta t \ {}^{n}d_{kl}$.

Remark 4.10: It must be observed that in Hughes (1984) the midpoint rule only allows to compute the **most accurate** value of the strain increment, instead of the **exact** value. The discrepancy is due to the different setting: a fixed frame is employed in Hughes (1984), and the strain increment is defined there as

$$\Delta\varepsilon_{\alpha\beta} = \int_{t_n}^{t_{n+1}} d_{\alpha\beta}(t)\mathrm{d}t$$

However, according to Malvern (1969) *"there is no physical significance to such an integration at one space point even in a steady flow simulation."*

The mid-point rule also ensures that a second-order approximation for the position and the velocity is achieved,

$$^{n+\frac{1}{2}}z^{\alpha} = {}^{n+\frac{1}{2}}\bar{z}^{\alpha} + \mathcal{O}(\Delta t^2) \quad ; \quad {}^{n+\frac{1}{2}}v^{\alpha} = \bar{v}^{\alpha} + \mathcal{O}(\Delta t^2). \tag{4.39}$$

By combining Eqs. (4.35), (4.37) and (4.39), it is possible to conclude that second-order approximation is also obtained for the rate-of-deformation tensor,

$$^{n+\frac{1}{2}}d_{kl} = {}^{n+\frac{1}{2}}\bar{d}_{kl} + \mathcal{O}(\Delta t^2), \tag{4.40}$$

which, after substitution in Eq. (4.38) yields the relation employed for the error analysis, Eq. (4.28):

$$\Delta\varepsilon_{kl} = \Delta t\ {}^{n+\frac{1}{2}}d_{kl} + \mathcal{O}(\Delta t^3).$$

4.4 Numerical implementation

4.4.1 From convected frames to Cartesian coordinates

The convected frame formalism may be employed to develop a large strain finite element code, thus resulting in a direct treatment of constitutive equations. Most existing large strain codes, however, use a fixed frame. Moreover, one of the goals of this Chapter is to present a simple way of enhancing a small strain code –written in a fixed frame– into a large strain code. For this reason, the adaptation of the algorithms presented above to a fixed orthonormal frame is shown next. It will be assumed that the initial orthonormal coordinates are employed as material coordinates, $x^i = Z^i$.

First-order algorithm

The transformation rule between the contravariant material and orthonormal components is shown in Eq. (4.12) for the stress tensor. A similar relation holds for the fourth-order modulus tensor, namely

$$C^{ijkl} = \sqrt{g}\frac{\partial Z^i}{\partial z^\alpha}\frac{\partial Z^j}{\partial z^\beta}\frac{\partial Z^k}{\partial z^\gamma}\frac{\partial Z^l}{\partial z^\delta}C^{\alpha\beta\gamma\delta}\,. \tag{4.41}$$

Various representations of the strain increment in a fixed frame are possible, depending on which configuration they are referred to. Three common choices, see Hughes (1984), are the Lagrangian strain increment (referred to the configuration at t_n), the midstep strain increment (at $t_{n+\frac{1}{2}}$) and the Eulerian strain increment (at t_{n+1}). The first-order algorithm uses quantities at the beginning of the step, so the Lagrangian strain increment is chosen,

$${}^{n}\Delta\varepsilon_{\mu\nu} = \frac{1}{2}\left[\frac{\partial(\Delta z^\mu)}{\partial^n z^\nu} + \frac{\partial(\Delta z^\nu)}{\partial^n z^\mu} + \frac{\partial(\Delta z^\alpha)}{\partial^n z^\nu}\frac{\partial(\Delta z^\alpha)}{\partial^n z^\mu}\right]. \tag{4.42}$$

In this equation the superscript n emphasizes that ${}^{n}\Delta\varepsilon_{\mu\nu}$ are the orthonormal components of the strain increment referred to the configuration at t_n. By using the expression of the material components of the strain increment of the first-order algorithm, Eq. (4.23b), it can be checked that the relation between $\Delta\varepsilon_{kl}$ and ${}^{n}\Delta\varepsilon_{\mu\nu}$ is

$$\Delta\varepsilon_{kl} = \frac{\partial^{n} z^{\mu}}{\partial Z^{k}} \frac{\partial^{n} z^{\nu}}{\partial Z^{l}} \, {}^{n}\Delta\varepsilon_{\mu\nu}. \tag{4.43}$$

Using the transformation rules given by Eqs. (4.12), (4.41) and (4.42) in Eq. (4.23a) yields, after some manipulation,

$${}^{n+1}\sigma^{\alpha\beta} = \frac{\sqrt{{}^{n}g}}{\sqrt{{}^{n+1}g}} \frac{\partial^{n+1} z^{\alpha}}{\partial Z^{i}} \frac{\partial^{n+1} z^{\beta}}{\partial Z^{j}} \frac{\partial Z^{i}}{\partial^{n} z^{\gamma}} \frac{\partial Z^{j}}{\partial^{n} z^{\delta}} \left({}^{n}\sigma^{\gamma\delta} + {}^{n}C^{\gamma\delta\mu\nu} \, {}^{n}\Delta\varepsilon_{\mu\nu} \right). \tag{4.44}$$

The partial derivatives that appear in Eq. (4.44) are the orthonormal components of the deformation gradient $\boldsymbol{F}$ and its inverse, see Malvern (1969). In an incremental analysis, the incremental deformation gradient ${}^{n}\boldsymbol{A}$ that relates the configuration at the beginning of the time-step to the configuration at the end of the step, see Pinsky *et al.* (1983) and Figure 4.4, is defined as

$${}^{n}\boldsymbol{A} = {}^{n+1}\boldsymbol{F} \cdot {}^{n}\boldsymbol{F}^{-1} \tag{4.45}$$

By modifying Eq. (4.44) according to Eq. (4.45) and taking into account that $\sqrt{g} = \det(\boldsymbol{F})$, the first-order algorithm is finally written in a fixed orthonormal frame, using compact notation, as

$${}^{n+1}\boldsymbol{\sigma} = \frac{1}{\det({}^{n}\boldsymbol{A})} {}^{n}\boldsymbol{A} \cdot [{}^{n}\boldsymbol{\sigma} + {}^{n}\boldsymbol{C} : {}^{n}\Delta\boldsymbol{\varepsilon}] \cdot {}^{n}\boldsymbol{A}^{\mathrm{T}} \tag{4.46a}$$

$${}^{n}\Delta\boldsymbol{\varepsilon} = \frac{1}{2}\left\{ \left[\frac{\partial(\Delta\boldsymbol{u})}{\partial^{n}\boldsymbol{z}}\right] + \left[\frac{\partial(\Delta\boldsymbol{u})}{\partial^{n}\boldsymbol{z}}\right]^{\mathrm{T}} + \left[\frac{\partial(\Delta\boldsymbol{u})}{\partial^{n}\boldsymbol{z}}\right]^{\mathrm{T}} \cdot \left[\frac{\partial(\Delta\boldsymbol{u})}{\partial^{n}\boldsymbol{z}}\right] \right\}. \tag{4.46b}$$

Remark 4.11: The incremental deformation gradient ${}^{n}\boldsymbol{A}$ is employed in Eq. (4.46a) to push forward both the stress at time t_n and the stress increment, also referred to the configuration at t_n, to the final configuration at time t_{n+1}.

Remark 4.12: The midstep configuration is not needed in this algorithm. As a counterpart, however, quadratic terms are required to compute the strain increment referred to the configuration at t_n, Eq. (4.46b).

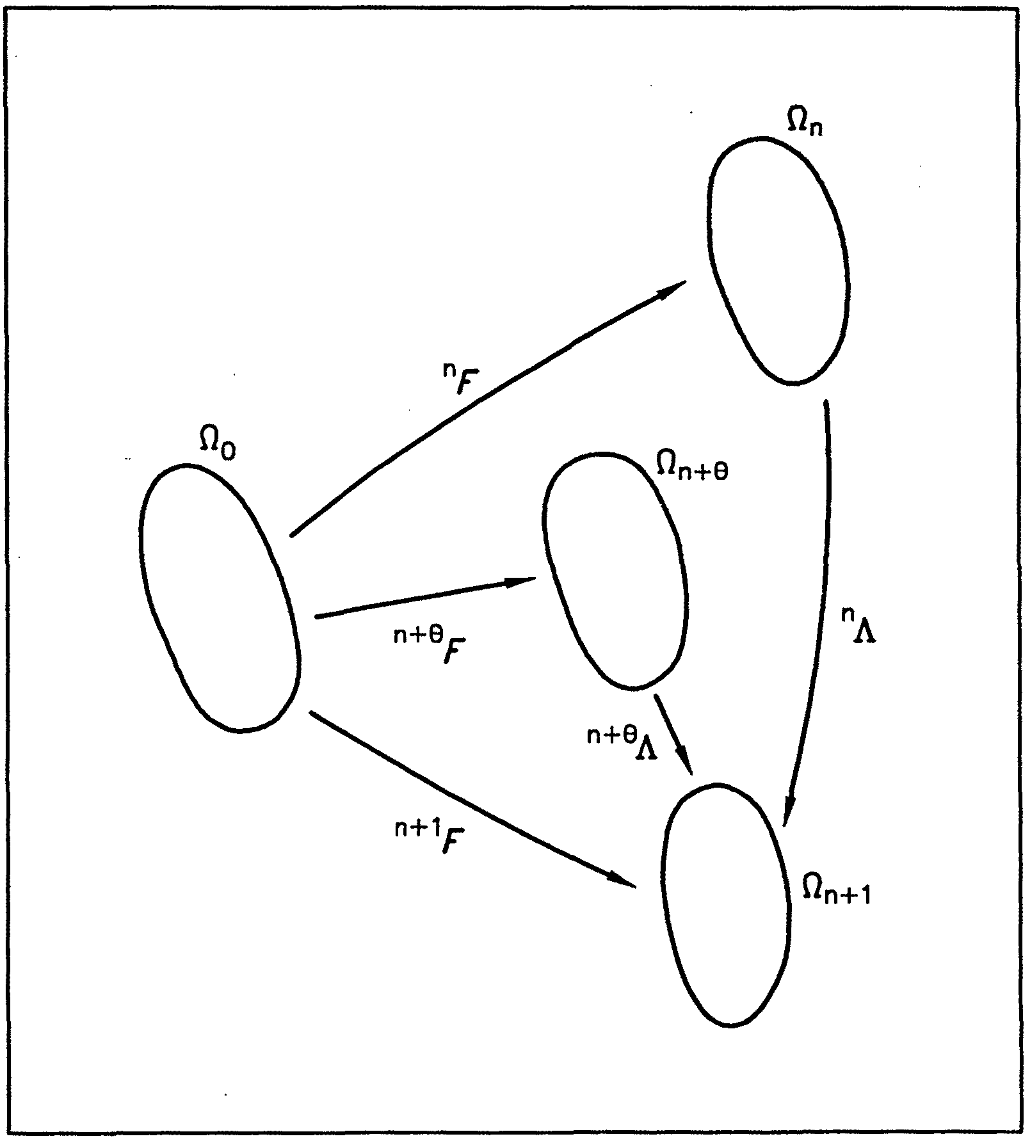

Figure 4.4 Incremental deformation gradients

Second-order algorithm

A similar process is carried out for the second-order algorithm. Since this algorithm employs midstep quantities, the midstep strain increment, (Hughes, 1984), is chosen to represent the increment of strain in the fixed frame,

$$^{n+\frac{1}{2}}\overline{\Delta\varepsilon}_{\mu\nu} = \frac{1}{2}\left[\frac{\partial(\Delta z^{\mu})}{\partial^{n+\frac{1}{2}}z^{\mu}} + \frac{\partial(\Delta z^{\nu})}{\partial^{n+\frac{1}{2}}z^{\mu}}\right]. \tag{4.47}$$

As previously commented, the bar remarks that $^{n+\frac{1}{2}}\overline{\Delta\varepsilon}_{\mu\nu}$ is referred to the midstep configuration associated to the constant-velocity assumption. To study the relation between $\Delta\varepsilon_{kl}$ and $^{n+\frac{1}{2}}\overline{\Delta\varepsilon}_{\mu\nu}$, it is convenient to use the expression of the material components of the strain increment of Eq. (4.25b). It can be checked then that

$$\Delta\varepsilon_{kl} = \frac{\partial^{n+\frac{1}{2}}z^{\mu}}{\partial Z^{k}}\frac{\partial^{n+\frac{1}{2}}z^{\nu}}{\partial Z^{l}}\, {}^{n+\frac{1}{2}}\overline{\Delta\varepsilon}_{\mu\nu}. \tag{4.48}$$

Transforming Eq. (4.25a) from material to orthonormal components with the aid of the transformation rules of Eqs. (4.12), (4.41) and (4.48) results in

$$\begin{aligned} {}^{n+1}\sigma^{\alpha\beta} = & \frac{\sqrt{^{n}g}}{\sqrt{^{n+1}g}}\frac{\partial^{n+1}z^{\alpha}}{\partial Z^{i}}\frac{\partial^{n+1}z^{\beta}}{\partial Z^{j}}\frac{\partial Z^{i}}{\partial^{n}z^{\gamma}}\frac{\partial Z^{j}}{\partial^{n}z^{\delta}}\,{}^{n}\sigma^{\gamma\delta} + \\ & \frac{\sqrt{^{n+\frac{1}{2}}g}}{\sqrt{^{n+1}g}}\frac{\partial^{n+1}z^{\alpha}}{\partial Z^{i}}\frac{\partial^{n+1}z^{\beta}}{\partial Z^{j}}\frac{\partial Z^{i}}{\partial^{n+\frac{1}{2}}z^{\gamma}}\frac{\partial Z^{j}}{\partial^{n+\frac{1}{2}}z^{\delta}}\,{}^{n+\frac{1}{2}}\overline{C}^{\gamma\delta\mu\nu}\,{}^{n+\frac{1}{2}}\overline{\Delta\varepsilon}_{\mu\nu}. \end{aligned} \tag{4.49}$$

Since the midstep configuration is involved, the incremental deformation gradient $^{n+\frac{1}{2}}\boldsymbol{\Lambda}$ is required, it is expressed as

$$^{n+\frac{1}{2}}\boldsymbol{\Lambda} = {}^{n+1}\boldsymbol{F}\,.\,{}^{n+\frac{1}{2}}\boldsymbol{F}^{-1}, \tag{4.50}$$

and relates the midstep configuration to the end-of-step configuration, see Pinsky *et al.* (1983) and Figure 4.4. With the help of $^{n+\frac{1}{2}}\boldsymbol{\Lambda}$, the second-order algorithm can be written in a fixed orthonormal frame as

$$ {}^{n+1}\sigma = \frac{1}{\det({}^{n}A)}\,{}^{n}A\cdot{}^{n}\sigma\cdot{}^{n}A^{\mathrm{T}} + \frac{1}{\det({}^{n+\frac{1}{2}}A)}\,{}^{n+\frac{1}{2}}A\cdot\left[{}^{n+\frac{1}{2}}\bar{C} : {}^{n+\frac{1}{2}}\overline{\Delta\varepsilon}\right]\cdot{}^{n+\frac{1}{2}}A^{\mathrm{T}} \qquad (4.51a) $$

$$ {}^{n+\frac{1}{2}}\overline{\Delta\varepsilon} = \frac{1}{2}\left\{\left[\frac{\partial(\Delta u)}{\partial^{n+\frac{1}{2}}\bar{z}}\right] + \left[\frac{\partial(\Delta u)}{\partial^{n+\frac{1}{2}}\bar{z}}\right]^{\mathrm{T}}\right\}. \qquad (4.51b) $$

Remark 4.13: The incremental deformation gradients ${}^{n}A$ and ${}^{n+\frac{1}{2}}A$ are employed in Eq. (4.51a) to push forward the stress at time t_n and the stress increment, referred to the midstep configuration, to the final configuration at time t_{n+1}, so they can be properly added to yield a relevant tensor in this configuration. This transformation is essential, since orthonormal components of a tensor at different instants cannot be directly added, because they are referred to different configurations (Pinsky *et al.*, 1983).

Remark 4.14: The choice of the midstep configuration allows to compute the strain increment as the symmetrized gradient of the increment of displacements, with no quadratic terms. Because of this, Eqs. (4.51) provide a straightforward generalization to large strains of the classical small strain analysis.

As commented previously the stress update algorithms present more elaborated expressions in a fixed frame, Eqs. (4.46) and (4.51), than in a convected frame, Eqs. (4.23) and (4.25). As explained in Remarks 4.11 and 4.13, quantities in a fixed frame must be transformed into a common configuration before they can be combined. If, from the start, a fixed frame context is chosen to state and time-integrate the rate-form constitutive equation, as done in Bathe *et al.* (1975) and Pinsky *et al.* (1983), these transformations are performed by means of pull-back and push-forward operators. A classical presentation of the two algorithms using an orthonormal fixed frame can be found in Appendix B and in Rodríguez-Ferran & Huerta (1996).

4.4.2 Implementation in a small-strain code

It is shown in this Subsection that both stress update algorithms can be employed to add large-strain capabilities to a small-strain FE code in a simple way.

The basic idea is that the incremental deformation gradients required in Eqs. (4.46a) and (4.51a) can be computed in a straightforward manner using quantities that are available in a small strain code. Consider, for instance, the incremental deformation gradient ${}^{n}\boldsymbol{A}$ relating Ω_n to Ω_{n+1}, Eq. (4.45). Recalling the definition of $\boldsymbol{F}$ and the expression of incremental displacements, it can be easily checked that ${}^{n}\boldsymbol{A}$ can be written as

$$ {}^{n}\boldsymbol{A} = \frac{\partial({}^{n+1}\boldsymbol{z})}{\partial({}^{n}\boldsymbol{z})} = \boldsymbol{I} + \frac{\partial(\Delta \boldsymbol{u})}{\partial({}^{n}\boldsymbol{z})}. \tag{4.52} $$

If an Updated Lagrangian formulation is used, (Bathe, 1982), the configuration Ω_n is taken as a reference to compute the incremental displacements. In such a context, ${}^{n}\boldsymbol{A}$ can be computed from Eq. (4.52) with the aid of standard nodal shape functions, by expressing $\Delta \boldsymbol{u}$ in terms of the nodal values of incremental displacements. Since the derivatives of shape functions are available in a standard FE code, (Hughes, 1987; Zienkiewicz & Taylor, 1991), no new quantities must be computed to obtain ${}^{n}\boldsymbol{A}$.

As for ${}^{n+\frac{1}{2}}\boldsymbol{A}$, it can be computed as

$$ {}^{n+\frac{1}{2}}\boldsymbol{A} = \frac{\partial({}^{n+1}\boldsymbol{z})}{\partial({}^{n+\frac{1}{2}}\boldsymbol{z})} = \boldsymbol{I} + \frac{1}{2}\frac{\partial(\Delta \boldsymbol{u})}{\partial({}^{n+\frac{1}{2}}\boldsymbol{z})}, \tag{4.53} $$

so ${}^{n+\frac{1}{2}}\boldsymbol{A}$ can also be directly computed with the aid of the shape functions, once the configuration of the mesh has been updated from Ω_n to $\Omega_{n+\frac{1}{2}}$.

As a result, the only two additional features that are required to handle large strains are (1) the updating of mesh configuration, and (2) the computation of incremental deformation gradients, Eqs. (4.52) and (4.53). This can be seen by comparing the schematic algorithm for a small-strain analysis with nonlinear material behaviour, shown in Box 4.1 with the large-strain versions, depicted in Box 4.2 (first stress update algorithm) and Box 4.3 (second stress update algorithm). In Boxes 4.2 and 4.3 (large strains), the modifications with respect to Box 4.1 (small strains) are highlighted with boldface, and a star, $\star$, is employed to designate additional steps.

Regarding the computational cost, it can be seen in Boxes 4.2 and 4.3 that the second algorithm is more expensive. Indeed, the second algorithm requires two configuration updates (from Ω_n to $\Omega_{n+\frac{1}{2}}$ and from $\Omega_{n+\frac{1}{2}}$ to Ω_{n+1}) and, more importantly, the computation of the incremental deformation gradients and Jacobians in two different configurations at each iteration. Whereas the first algorithm only requires one computation for each iteration. Thus, the computational cost per iteration of the second algorithm will be taken as approximately twice that of the first algorithm (this factor of two is an overestimation, because steps of the stress update —such as the plastic corrector— are performed once in both algorithms, but this factor is valid as a first approximation for comparison purposes).

```
FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;
    FOR EVERY ITERATION $k$ WITHIN THE TIME-INCREMENT;
        1.– Compute the incremental displacements $\Delta u^k$ by solving a linearized form of the equilibrium equation
        2.– Compute the incremental strains $\Delta\varepsilon^k$ as the symmetrized gradient of displacements:
            $\Delta\varepsilon^k = \frac{1}{2}\left\{\left[\frac{\partial(\Delta u^k)}{\partial Z}\right] + \left[\frac{\partial(\Delta u^k)}{\partial Z}\right]^{\mathrm{T}}\right\}$
        3.– Compute the elastic trial incremental stresses $\Delta\sigma^k_{\text{trial}}$ via the elastic modulus tensor:
            $\Delta\sigma^k_{\text{trial}} = C : \Delta\varepsilon^k$
        4.– Compute the elastic trial stresses at $t_{n+1}$:
            $\sigma^k_{\text{trial}} = {}^n\sigma + \Delta\sigma^k_{\text{trial}}$
        5.– Compute the final stresses $\sigma^k$ at $t_{n+1}$ by performing the plastic correction
        6.– Compute the internal forces $f^k_{\text{int}}$ by integrating the stresses $\sigma^k$
        7.– Check convergence. If it is not attained, go back to step 1.
```

Box 4.1 Small-strain analysis with nonlinear material behaviour

FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;

FOR EVERY ITERATION k WITHIN THE TIME-INCREMENT;

1.– Compute the incremental displacements $\Delta \boldsymbol{u}^k$ by solving a linearized form of the equilibrium equation

2.– Compute the incremental strains $\Delta \boldsymbol{\varepsilon}^k$ **accounting for quadratic terms, Eq. (4.46b):**

$$\Delta \boldsymbol{\varepsilon}^k = \frac{1}{2}\left\{\left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial({}^n\boldsymbol{z})}\right] + \left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial({}^n\boldsymbol{z})}\right]^{\mathrm{T}} + \left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial({}^n\boldsymbol{z})}\right]^{\mathrm{T}}\left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial({}^n\boldsymbol{z})}\right]\right\}$$

3.– Compute the elastic trial incremental stresses $\Delta \boldsymbol{\sigma}^k_{\text{trial}}$ via the elastic modulus tensor:

$$\Delta \boldsymbol{\sigma}^k_{\text{trial}} = \boldsymbol{C} : \Delta \boldsymbol{\varepsilon}^k$$

⋆ **Compute the incremental deformation gradient ${}^n\boldsymbol{\Lambda}$, Eq. (4.52), and its determinant ${}^nJ$**

4.– Compute the elastic trial stresses at t_{n+1}, **pushing forward both** ${}^n\boldsymbol{\sigma}$ **and** $\Delta \boldsymbol{\sigma}^k_{\text{trial}}$ **from configuration** Ω_n **to** Ω_{n+1}, **Eq. (4.46a):**

$$\boldsymbol{\sigma}^k_{\text{trial}} = {}^nJ^{-1}\, {}^n\boldsymbol{\Lambda}\, {}^n\boldsymbol{\sigma}\, {}^n\boldsymbol{\Lambda}^{\mathrm{T}} + {}^nJ^{-1}\, {}^n\boldsymbol{\Lambda} \left(\Delta \boldsymbol{\sigma}^k_{\text{trial}}\right) {}^n\boldsymbol{\Lambda}^{\mathrm{T}}$$

⋆ **Update the configuration from Ω_n to Ω_{n+1} by using the incremental displacements of the current step**

5.– Compute the final stresses $\boldsymbol{\sigma}^k$ at t_{n+1} by performing the plastic correction

6.– Compute the internal forces $\boldsymbol{f}^k_{\text{int}}$ by integrating the stresses $\boldsymbol{\sigma}^k$

7.– Check convergence. If it is not attained, **recover configuration** Ω_n and go back to step **1.**

Box 4.2 First stress update algorithm

FOR EVERY TIME-INCREMENT $[t_n, t_{n+1}]$;

FOR EVERY ITERATION k WITHIN THE TIME-INCREMENT;

1.– Compute the incremental displacements $\Delta \boldsymbol{u}^k$ by solving a linearized form of the equilibrium equation

★ **Update the configuration from Ω_n to $\Omega_{n+\frac{1}{2}}$**

2.– Compute the incremental strains $\Delta\varepsilon^k$ as the symmetrized gradient of displacements, Eq. (4.51b):

$$\Delta\varepsilon^k = \frac{1}{2}\left\{\left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial^{n+\frac{1}{2}}\boldsymbol{z}}\right] + \left[\frac{\partial(\Delta \boldsymbol{u}^k)}{\partial^{n+\frac{1}{2}}\boldsymbol{z}}\right]^{\mathrm{T}}\right\}$$

3.– Compute the elastic trial incremental stresses $\Delta\sigma^k_{\text{trial}}$ via the elastic modulus tensor:

$$\Delta\sigma^k_{\text{trial}} = C : \Delta\varepsilon^k$$

★ **Compute the incremental deformation gradients ${}^nA$ and ${}^{n+\frac{1}{2}}A$, Eqs. (4.52) and (4.53), and its determinants ${}^nJ$ and ${}^{n+\frac{1}{2}}J$**

4.– Compute the elastic trial stresses at t_{n+1}, **pushing forward ${}^n\sigma$ and $\Delta\sigma^k_{\text{trial}}$ to Ω_{n+1}, Eq. (4.51a):**

$$\sigma^k_{\text{trial}} = {}^nJ^{-1}\ {}^nA\ {}^n\sigma\ {}^nA^{\mathrm{T}} + {}^{n+\frac{1}{2}}J^{-1}\ {}^{n+\frac{1}{2}}A\Delta\sigma^k_{\text{trial}}\ {}^{n+\frac{1}{2}}A^{\mathrm{T}}$$

★ **Update the configuration from $\Omega_{n+\frac{1}{2}}$ to Ω_{n+1}**

5.– Compute the final stresses σ^k at t_{n+1} by performing the plastic correction

6.– Compute the internal forces f^k_{int} by integrating the stresses σ^k

7.– Check convergence. If it is not attained, **recover configuration** Ω_n and go back to step **1.**

Box 4.3 Second stress update algorithm

4.5 Numerical examples

4.5.1 Introduction

The two algorithms presented previously are compared with the help of various simple deformation paths (simple shear, uniaxial extension, extension and compression, dilatation, extension and rotation) and two benchmark tests in nonlinear mechanics: the necking of a circular bar and a shell under ring loads. Both elastic and elastoplastic cases are considered.

Elastic behaviour will be represented by a modulus tensor with constant orthonormal components which depend on the Lamé parameters λ and μ. All the simple deformation paths have been performed with a null Poisson's coefficient, resulting in $\lambda = 0$ and $\mu = E/2$, with E the Young's modulus. A bilinear elastoplastic law is employed for the plastic cases, with a plastic modulus of $E_p = E/100$ and a yield stress of $\sigma_y = E/2$.

A general result is that the values predicted with the first algorithm are more dependent on the number of time-increments than for the second one. This behaviour, which is in agreement with the error analysis, has been detected in the simple deformation paths and in both benchmark tests.

4.5.2 Simple shear

The problem statement for the simple shear test is shown in Figure 4.5, where the initial ($t = 0$) and final ($t = 1$) configurations are presented. The initial stress field is zero. The equations of motion are

$$\left.\begin{aligned} x(t) &= X + Yt \\ y(t) &= Y \end{aligned}\right\}, \tag{4.54}$$

If the rate-form hypoelastic behaviour is employed, Eq. (4.54) can be transformed

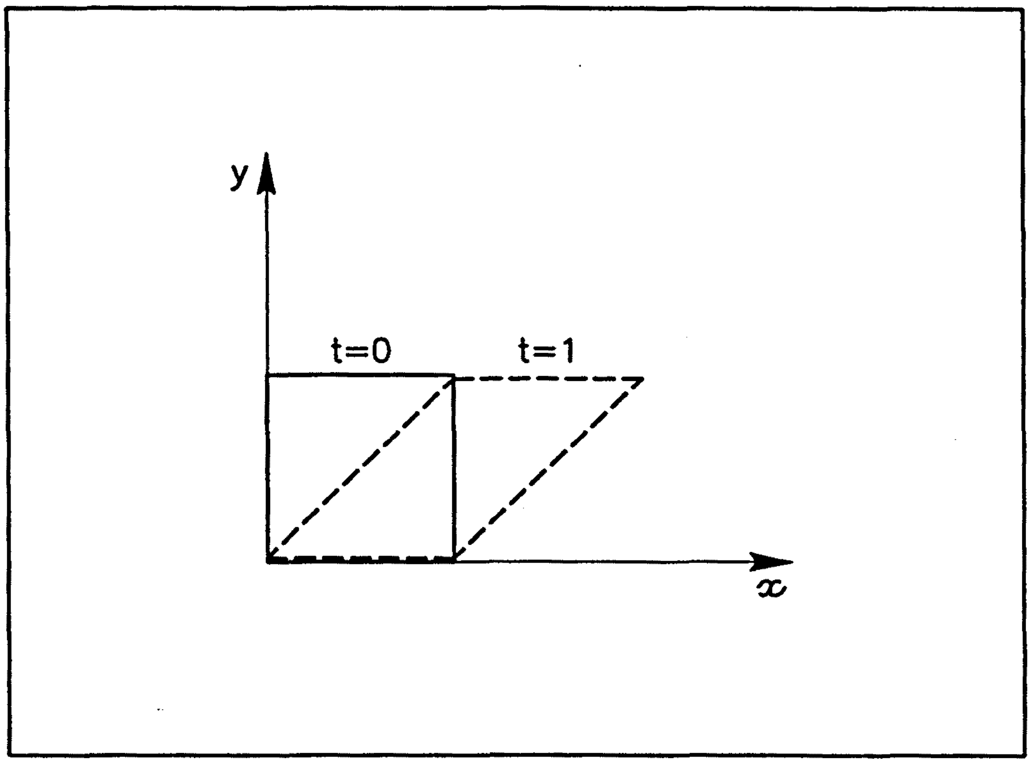

Figure 4.5 Simple shear test. Initial and final configurations

into the set of ordinary differential equations

$$\left.\begin{aligned} \dot{\sigma}_{xx} - 2\sigma_{xy} &= 0 \\ \dot{\sigma}_{xy} - \sigma_{yy} &= E/2 \\ \dot{\sigma}_{yy} &= 0 \end{aligned}\right\}, \tag{4.55}$$

which, complemented with null initial conditions, may be solved to provide the analytical solution

$$\left.\begin{aligned} \sigma_{xx}(t) &= \frac{E}{2}t^2 \\ \sigma_{xy}(t) &= \frac{E}{2}t \\ \sigma_{yy}(t) &= 0 \end{aligned}\right\}. \tag{4.56}$$

The two algorithms have been employed, with different values of the time-step (1, 2, 3, 5, 10, 20, 50 increments), to integrate the constitutive equation for the deformation path (4.54), from $t = 0$ to $t = 1$.

Figure 4.6 presents the results for the elastic case (1, 5, 50 increments). It can be seen that the second-order algorithm gets, for the three components of stress, the exact analytical values, whereas the first one grossly overestimates stresses when not enough increments are employed, and demands a small time-step to get close to the analytical solution. The output of a small strain analysis (1 increment), is also presented in Figure 4.6: the null σ_{yy} and linear σ_{xy} are correctly predicted, but this linear analysis is not able to reproduce quadratic σ_{xx}, Eq. (4.56), caused by a quadratic strain.

Taking the analytical solution (4.56), the error in the final stress ($t = 1$) can be computed and plotted versus the number of time-increments. In a log-log scale, an straight line with a slope equal to the order of the algorithm is expected (that is, 1 for the first algorithm and 2 for the second). To compute an average observed slope, a straight line is fitted via least-squares interpolation. The results are shown in Figure 4.7. Only the first algorithm is shown, because the second one provides the exact analytical solution (except for rounding errors) for any number of time-steps. It can be seen that for the three components of stress, Figures 4.7a, 4.7b and 4.7c, the observed slopes are very close to the expected value of 1 (1.09 for σ_{xx}, 1.16 for σ_{xy} and 1.00 for σ_{yy}). The first-order accuracy of the first algorithm is thus corroborated by this simple test.

The shear test has also been performed in the elastoplastic case. The results are presented in Figure 4.8 (1, 10, 50 increments). As in the elastic problem, the first-order algorithm grossly overestimates the final stress if only one increment is employed, while the second-order one provides much more accurate values. With a higher number of time-steps, both algorithms converge to the same response. It may be observed that, of course, when plastification starts (around $t = 0.6$), the curves differ from their elastic counterparts of Figure 4.6.

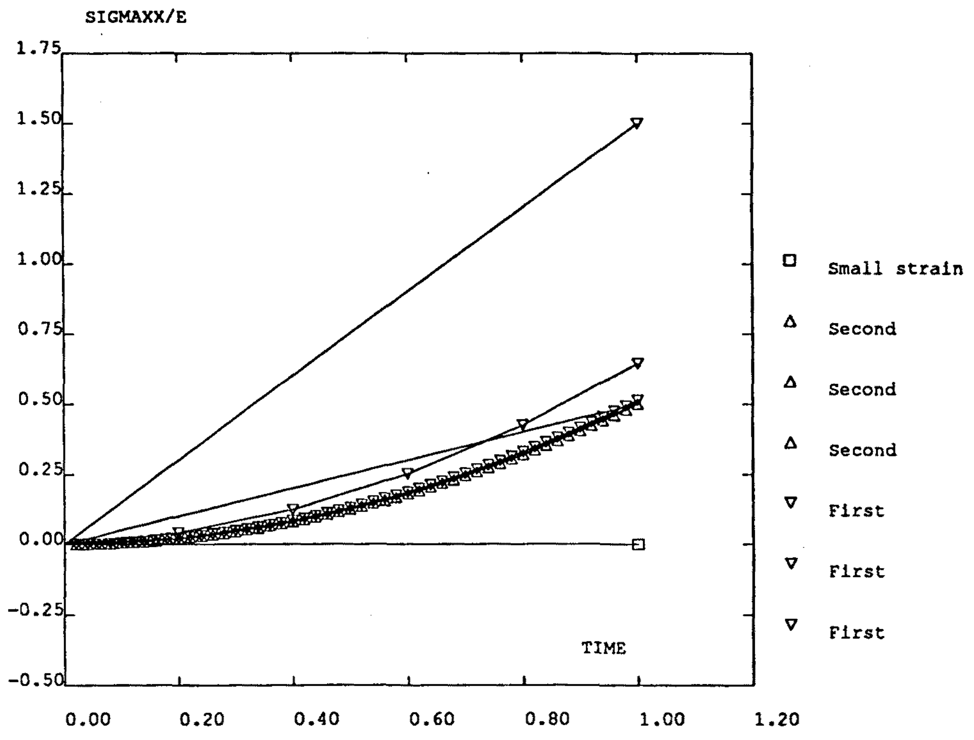

Fig. 4.6a

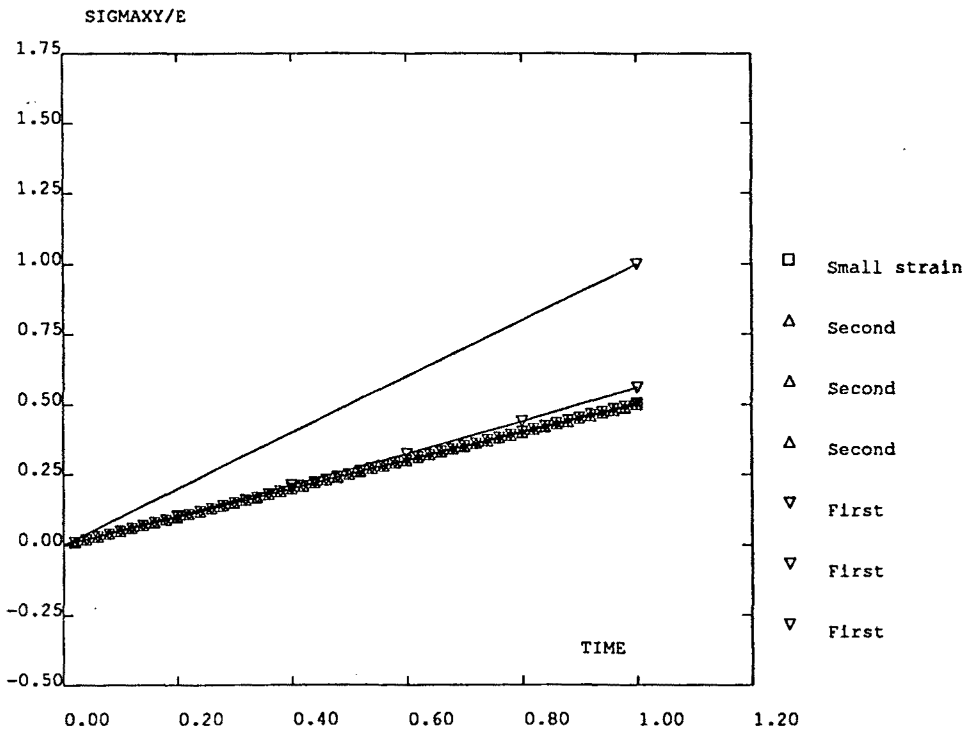

Fig. 4.6b

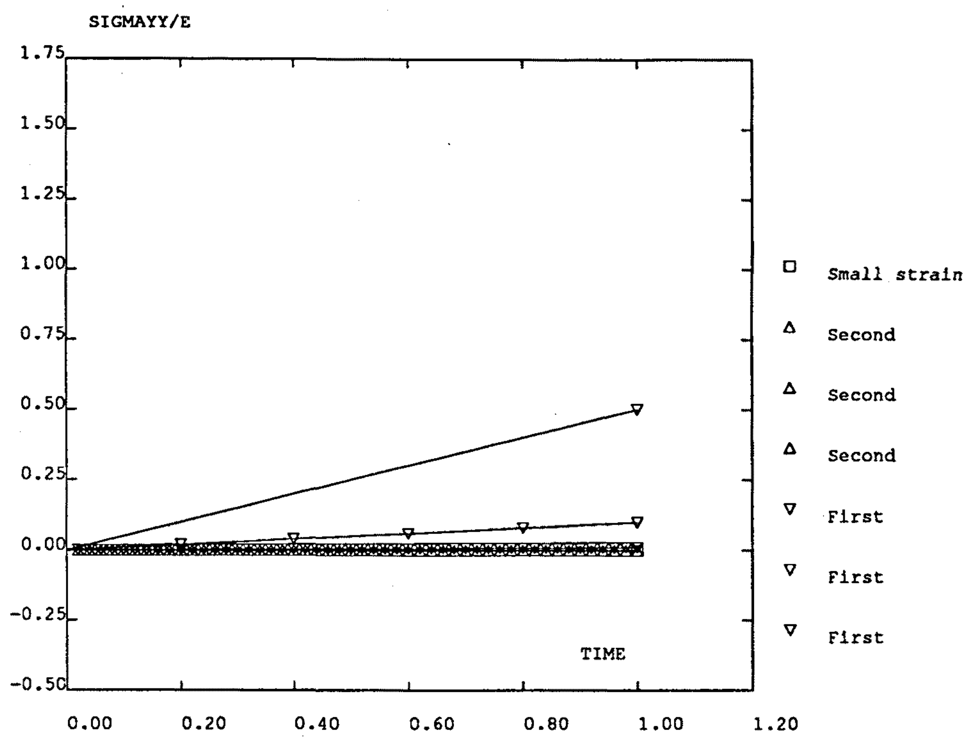

Fig. 4.6c

Figure 4.6 Simple shear test, elastic analysis. Stress vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}

A small strain analysis, this time with 50 increments to account for material nonlinearity, is again not satisfactory. As commented above, σ_{xx} is incorrectly kept at its zero initial value throughout the first steps of the computation, with elastic behaviour. As σ_{yy} is also zero, the only non-zero stress is σ_{xy}, and the elastic stress tensor is

$$\sigma = \begin{bmatrix} 0 & \sigma_{xy} \\ \sigma_{xy} & 0 \end{bmatrix} \tag{4.57}$$

that is, a fully deviatoric tensor. As the plastic correction, once plastification begins, is carried along precisely along the deviatoric part of the stress tensor, σ_{xy} is the only component of stress affected by plastification, while σ_{xx} and σ_{yy} keep their elastic behaviour. The need for a large strain analysis is again demonstrated.

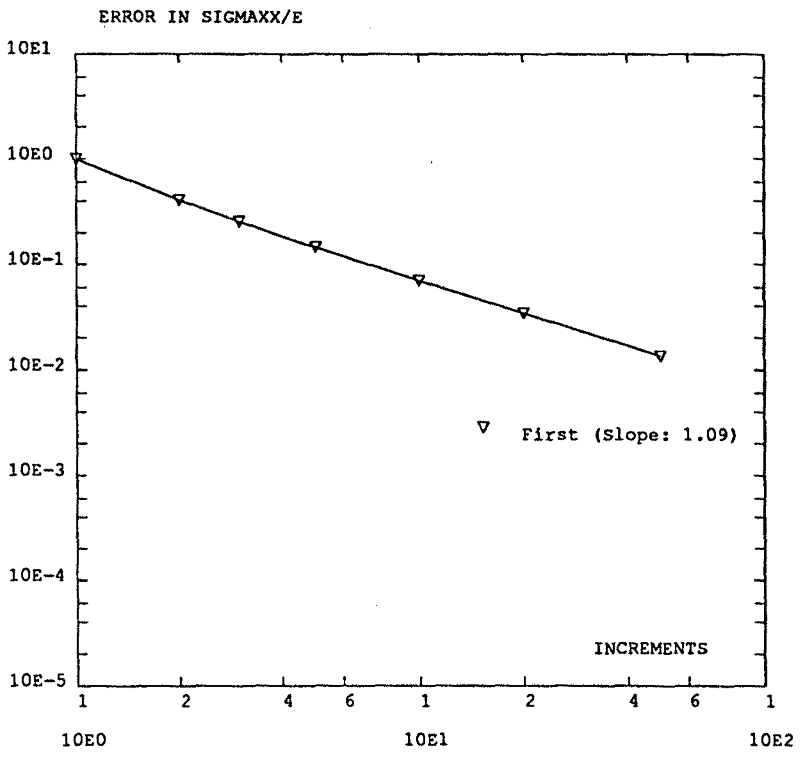

Fig. 4.7a

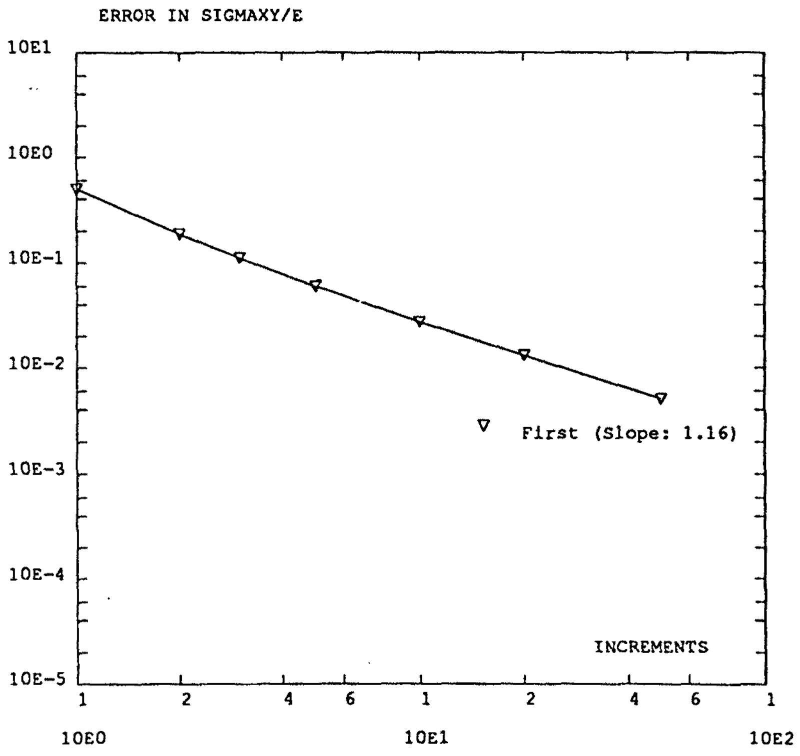

Fig. 4.7b

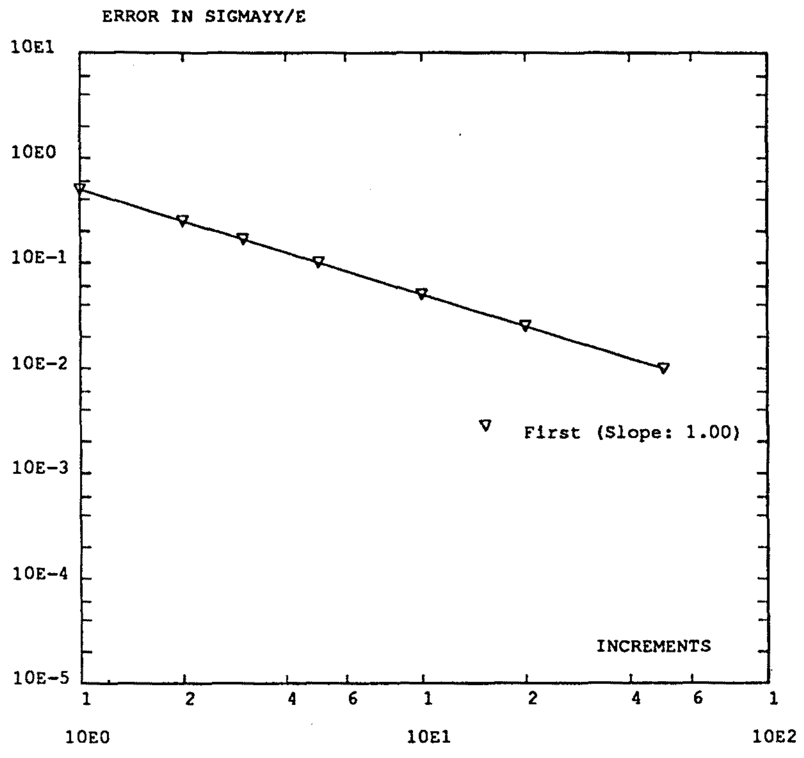

Fig. 4.7c

Figure 4.7 Simple shear test, elastic analysis. Error in the final value of stress ($t = 1$) vs. number of time-steps (1, 2, 3, 5, 10, 20, 50). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}

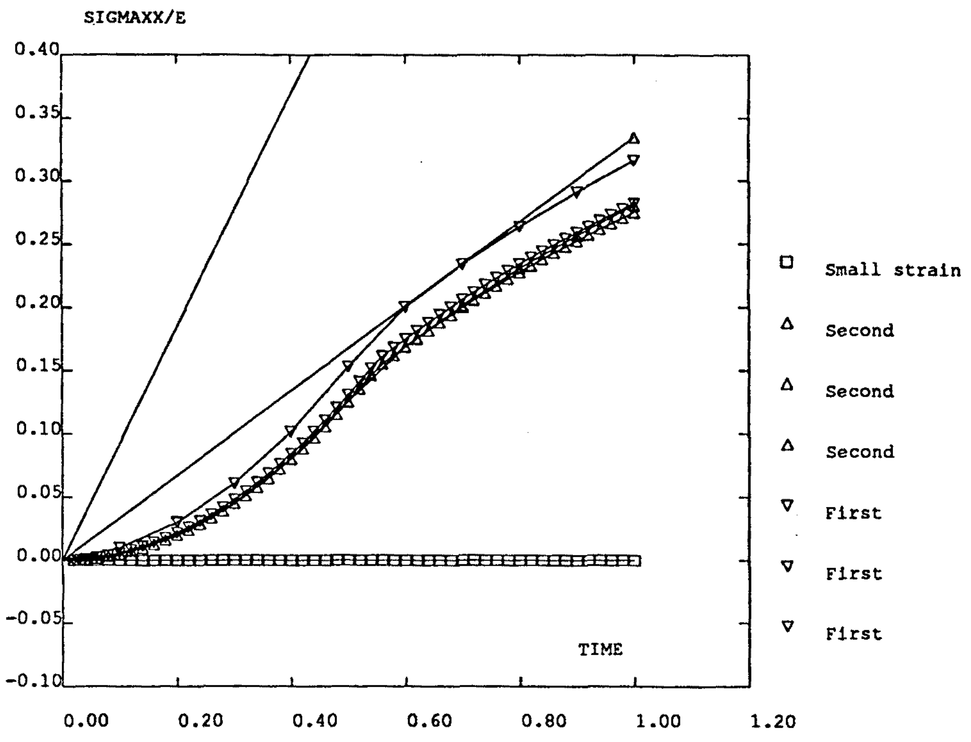

Fig. 4.8a

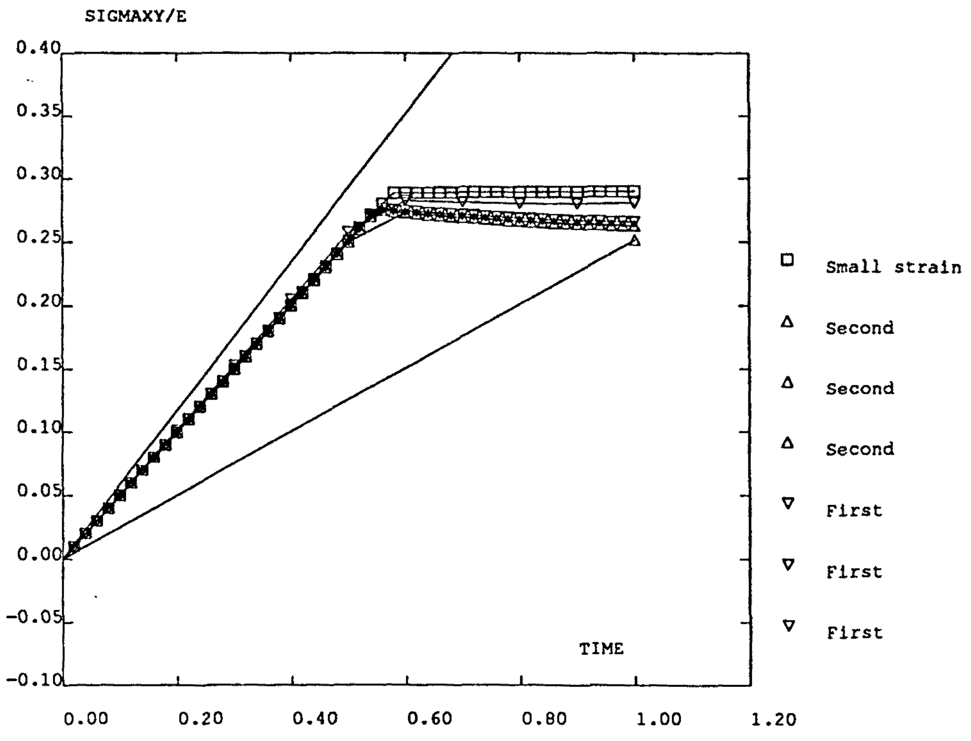

Fig. 4.8b

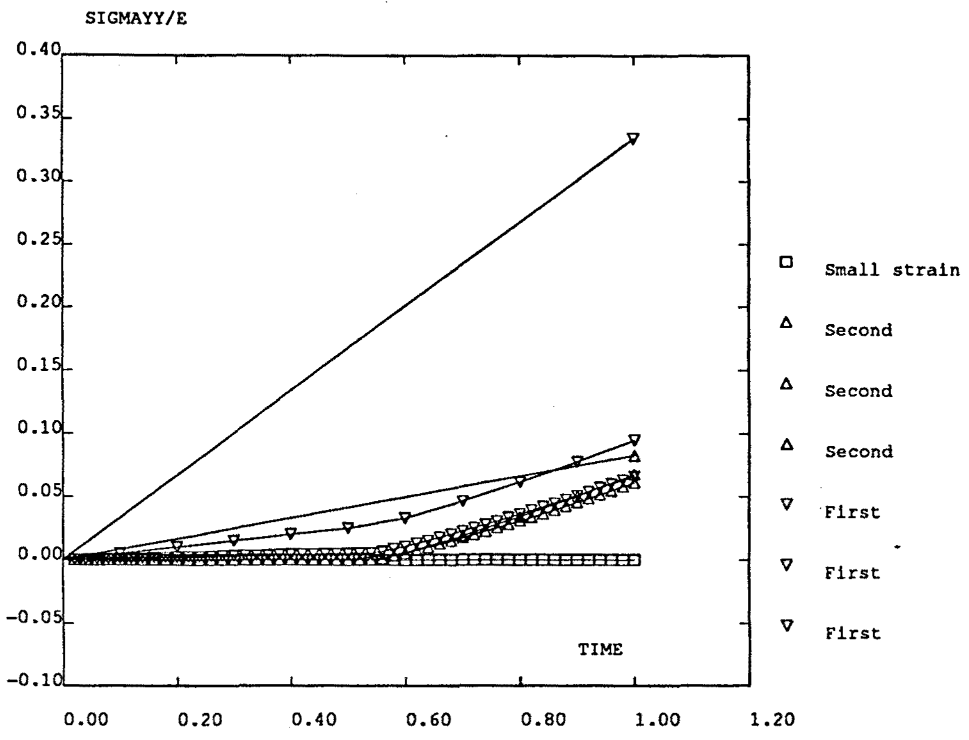

Fig. 4.8c

Figure 4.8 Simple shear test, elastoplastic analysis. Stress vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}

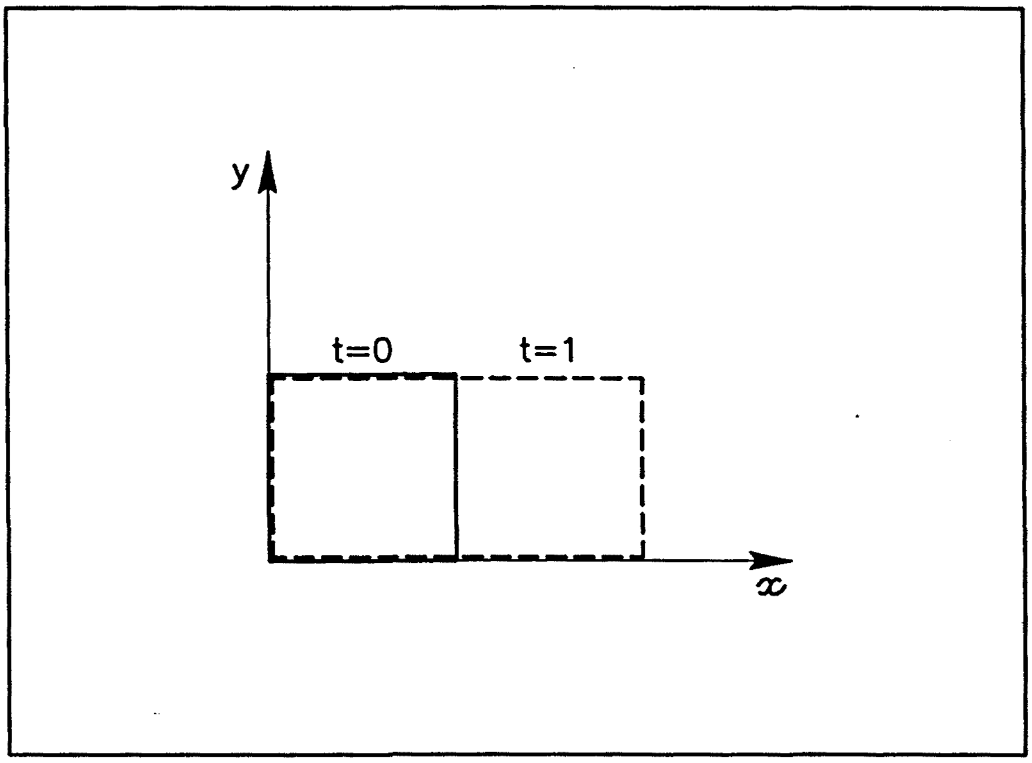

Figure 4.9 Uniaxial extension test. Initial and final configurations

4.5.3 Uniaxial extension

A unit square is subjected to uniaxial extension in the x direction, see Figure 4.9. The equations of motion are

$$\left.\begin{aligned} x(t) &= X(1+t) \\ y(t) &= Y \end{aligned}\right\}, \tag{4.58}$$

and the analytical solution of the hypoelastic constitutive equation is

$$\left.\begin{aligned} \sigma_{xx}(t) &= Et \\ \sigma_{xy}(t) &= 0 \\ \sigma_{yy}(t) &= 0 \end{aligned}\right\}. \tag{4.59}$$

The two algorithms correctly predict null values for σ_{xy} and σ_{yy}. Differences are found, on the contrary, for σ_{xx}: while the first algorithm grossly overestimates it, the second one slightly underestimates it, see Figure 4.10. It can be seen in Figure 4.10 that, for this particular test, a small strain analysis, with one time-step, produces the exact solution. This is due to the fact that the two sources of error (neglecting quadratic terms in the strain tensor and the time-discretization error) compensate each other. Of course, this is not the situation in a general case, as shown for the other tests.

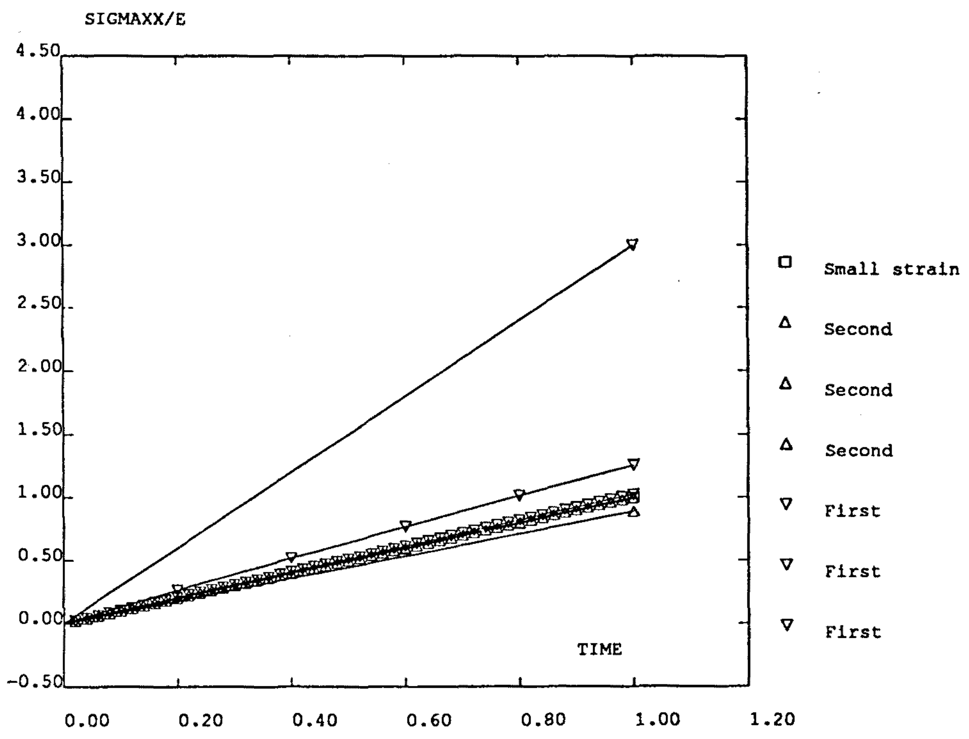

Figure 4.10 Uniaxial extension test, elastic analysis. Horizontal normal stress σ_{xx} vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step)

The error in the final value of σ_{xx} versus the number of time-steps is presented in Figure 4.11. The observed slopes for the two algorithms are in this case 1.13 for the first algorithm and 1.95 for the second one, in very good agreement with expected values of 1 and 2 respectively.

The plastic response is presented in Figure 4.12. When plastification begins at $t = 0.5$, the plastic correction is performed along the deviatoric part of the stress tensor, thus resulting in an increase of σ_{yy} at the expense of σ_{xx}. Again, the second-order algorithm behaves much better than the first one if a small number of time-increments is employed, especially for σ_{xx}, see Figure 4.12a. With a large number of time-steps (50), the two algorithms provide very similar results, different from the small strain analysis, see Figure 4.12b.

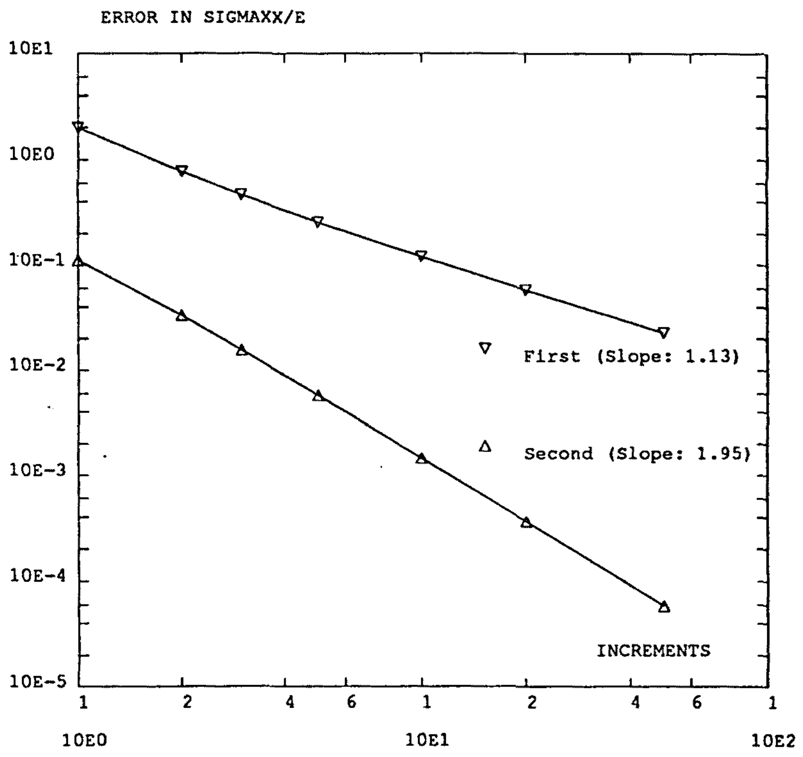

Figure 4.11 Uniaxial extension test, elastic analysis. Error in the final value of horizontal normal stress σ_{xx} ($t = 1$) vs. number of time-steps (1, 2, 3, 5, 10, 20, 50)

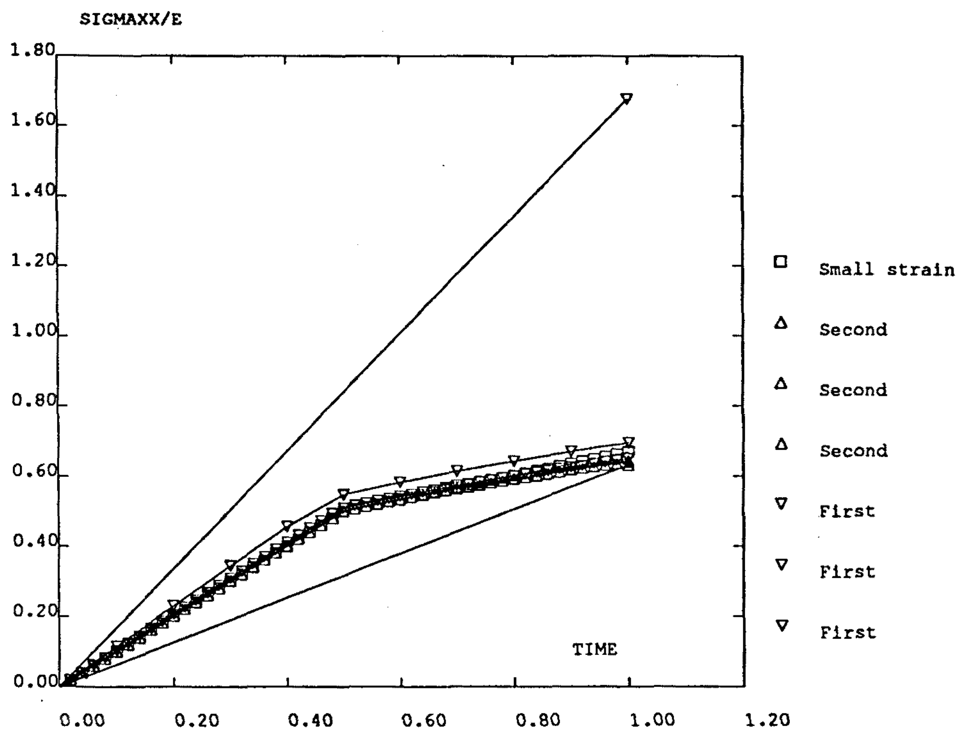

Fig. 4.12a

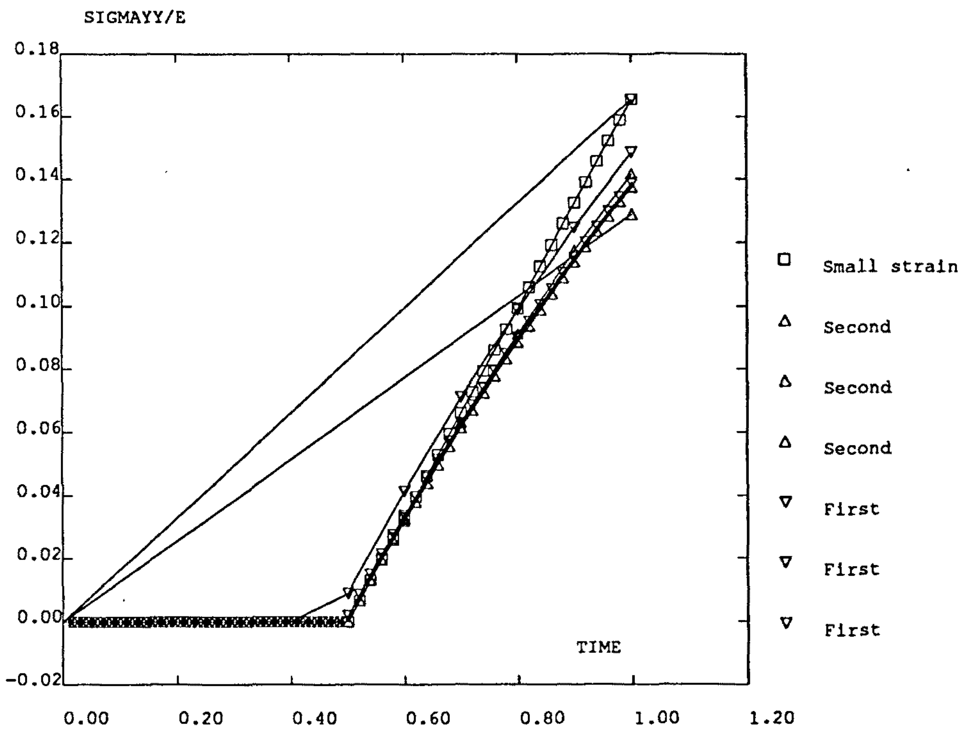

Fig. 4.12b

Figure 4.12 Uniaxial extension test, elastoplastic analysis. Stress vs. time curves computed with the two algorithms (1, 10 and 50 time-steps) and small-strain analysis (50 time-steps). a) σ_{xx}. b) σ_{yy}

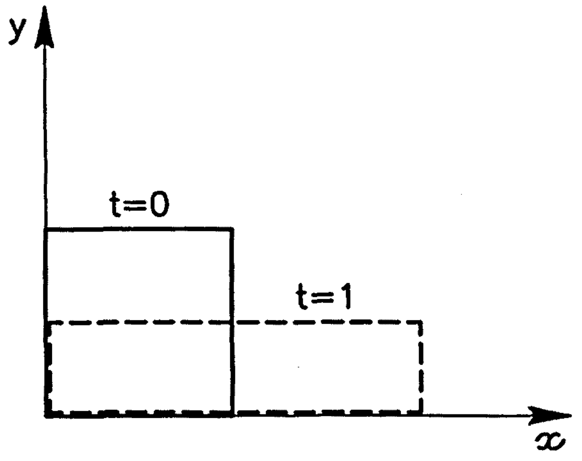

Figure 4.13 Extension and compression test. Initial and final configurations

4.5.4 Extension and compression

A unit square undergoes extension in the x direction and compression in the y direction, with no change in volume, see Figure 4.13. The equations of motion are

$$\left.\begin{aligned} x(t) &= X(1+t) \\ y(t) &= Y/(1+t) \end{aligned}\right\}, \tag{4.60}$$

and the analytical solution for the elastic case is

$$\left.\begin{aligned}\sigma_{xx}(t) &= E\left(t+\frac{t^2}{2}\right)\\ \sigma_{xy}(t) &= 0\\ \sigma_{yy}(t) &= \frac{E}{2}\left[\frac{1}{(1+t)^2}-1\right]\end{aligned}\right\}. \tag{4.61}$$

Again, both algorithms are capable of predicting null σ_{xy} for the elastic test, but differences appear for σ_{xx} and σ_{yy}, see Figure 4.14. The observed orders are in this case 1.15 (first algorithm) and 1.93 (second algorithm) for σ_{xx}, see Figure 4.15a, and 0.87 (first algorithm) and 1.99 (second algorithm) for σ_{yy}, see Figure 4.15b.

As for the plastic test, results are shown in Figure 4.16. Once again, convergence to the "reference" solution (with 50 time-increments) is faster with the second algorithm, while the small strain analysis yields a qualitatively different response.

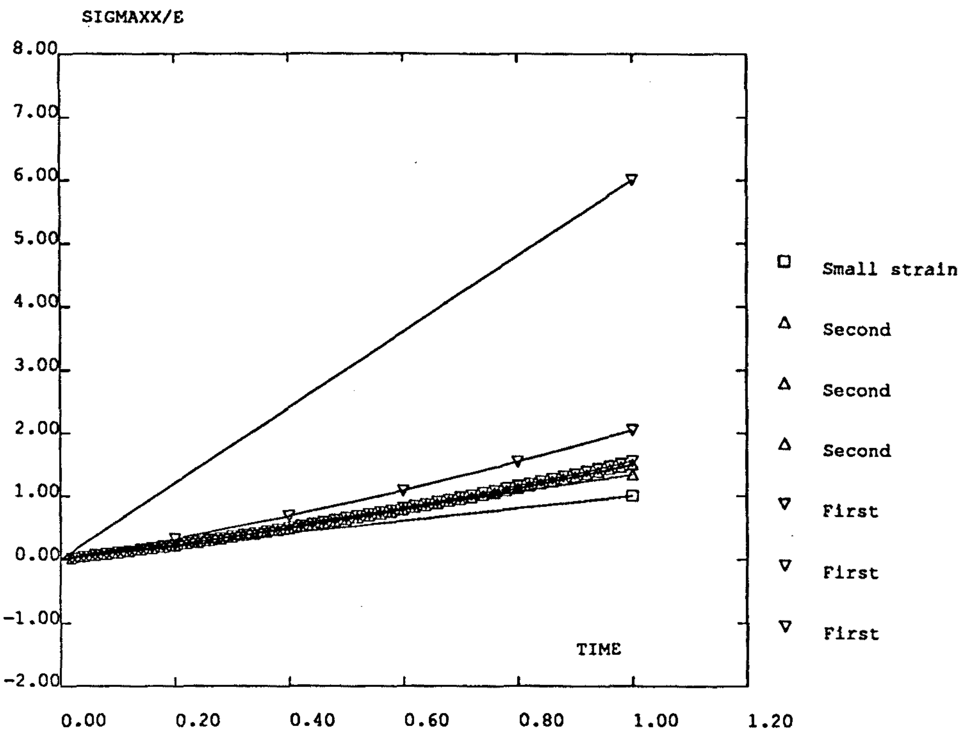

Fig. 4.14a

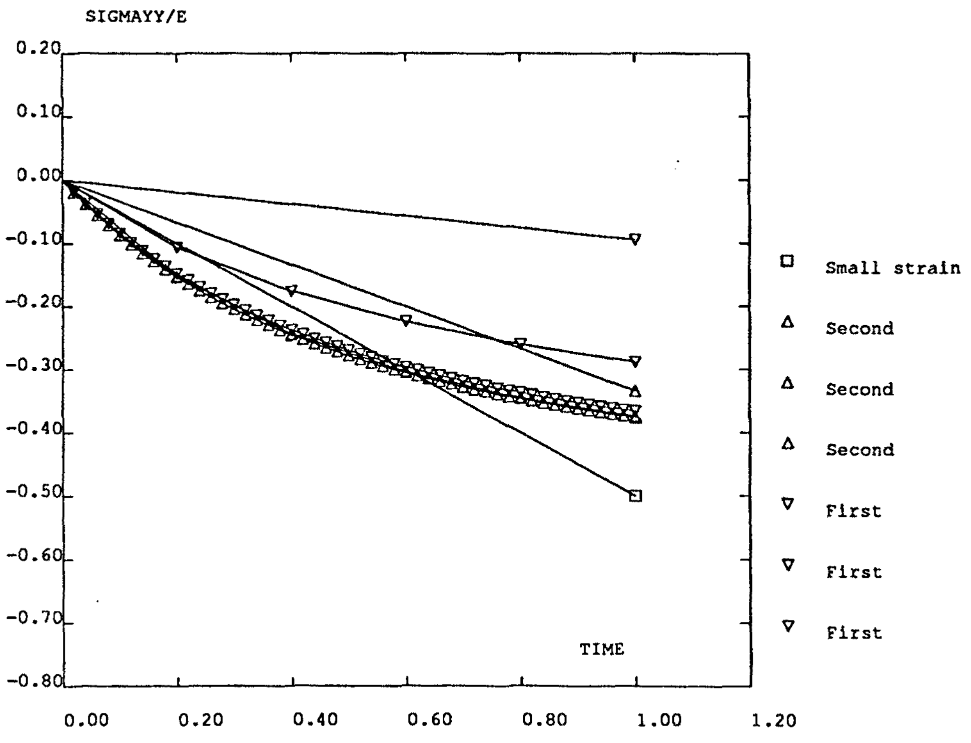

Fig. 4.14b

Figure 4.14 Extension and compression test, elastic analysis. Stress vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step). a) σ_{xx}. b) σ_{yy}

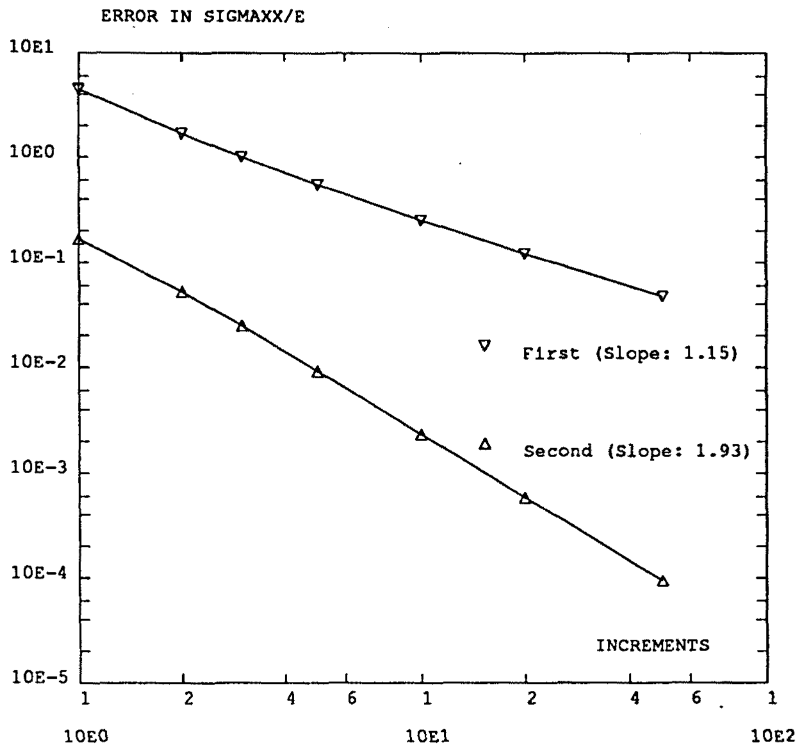

Fig. 4.15a

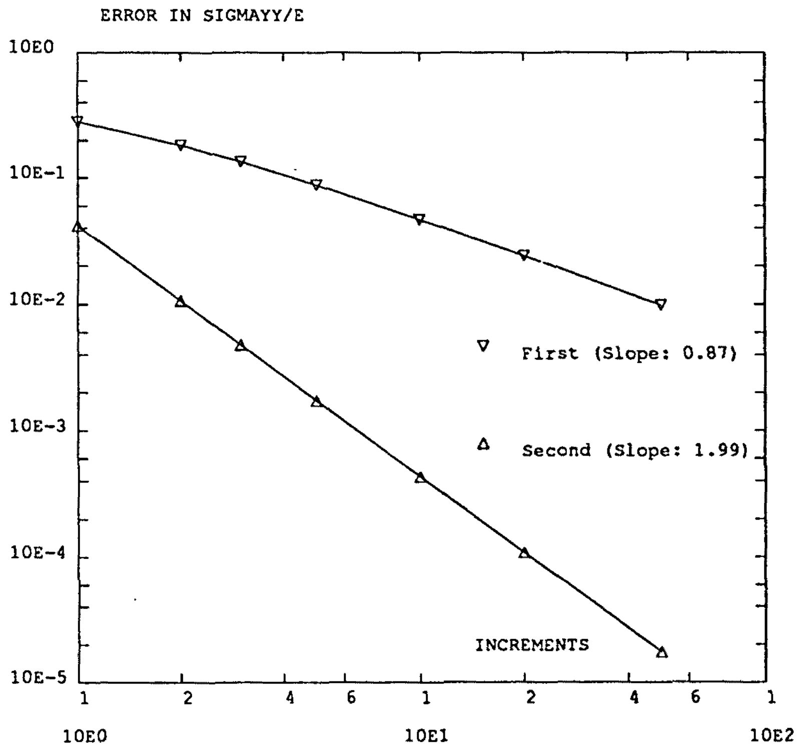

Fig. 4.15b

Figure 4.15 Extension and compression test, elastic analysis. Error in the final value of stress ($t = 1$) vs. number of time-steps (1, 2, 3, 5, 10, 20, 50). a) σ_{xx}. b) σ_{yy}

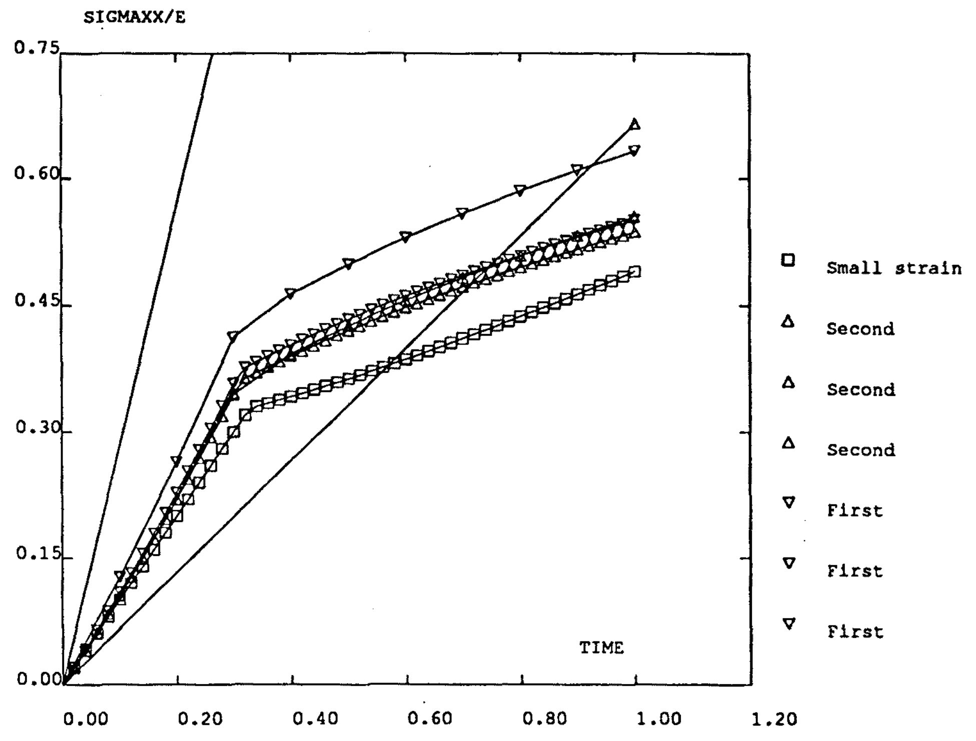

Fig. 4.16a

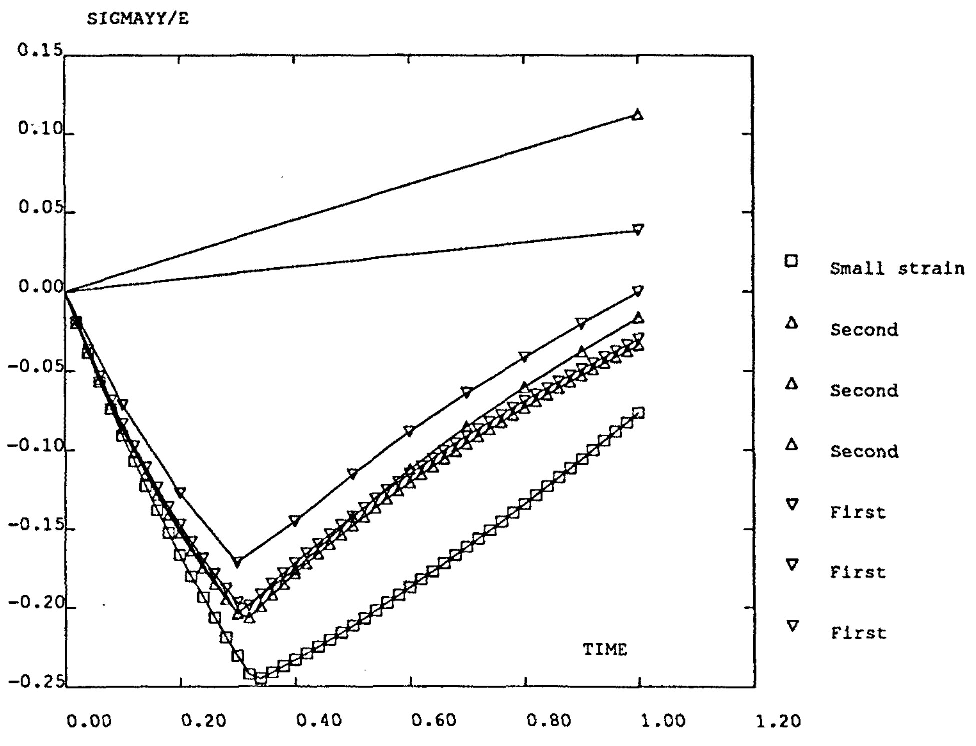

Fig. 4.16b

Figure 4.16 Extension and compression test, elastoplastic analysis. Stress vs. time curves computed with the two algorithms (1, 10 and 50 time-steps) and small-strain analysis (50 time-steps). a) σ_{xx}. b) σ_{yy}

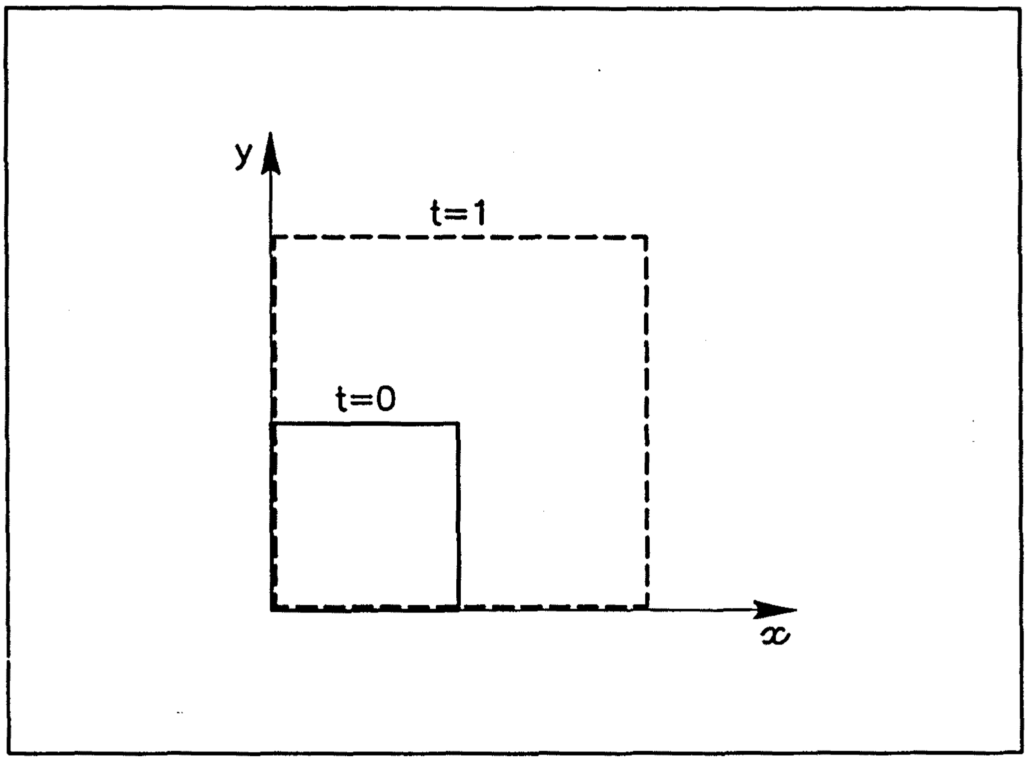

Figure 4.17 Dilatation test. Initial and final configurations

4.5.5 Dilatation

A unit square undergoes biaxial extension, see Figure 4.17. The equations of motion are

$$\left.\begin{aligned} x(t) &= X(1+t) \\ y(t) &= Y(1+t) \end{aligned}\right\}, \tag{4.62}$$

and the analytical solution is

$$\left.\begin{aligned} \sigma_{xx}(t) &= E\ \ln(1+t) \\ \sigma_{xy}(t) &= 0 \\ \sigma_{yy}(t) &= E\ \ln(1+t) \end{aligned}\right\}. \tag{4.63}$$

As in the two previous tests, both algorithms provide qualitatively correct results, in the sense that σ_{xy} is zero and σ_{xx} equals σ_{yy} for any number of time-steps and in both elastic and plastic modes. There are sharp differences, however, concerning convergence behaviour. For the elastic case, for instance, the second algorithm provides a better prediction with one time-increment than the first one with five, see Figure 4.18. It can be seen in Figure 4.19 that, once again, the algorithms behave as expected. The computed slopes are in this case 1.11 for the first algorithm and 1.97 for the second one.

A similar comparison is valid in the plastic test, where one increment with the second algorithm gets closer to the reference solution than ten steps of the first algorithm, see Figure 4.20.

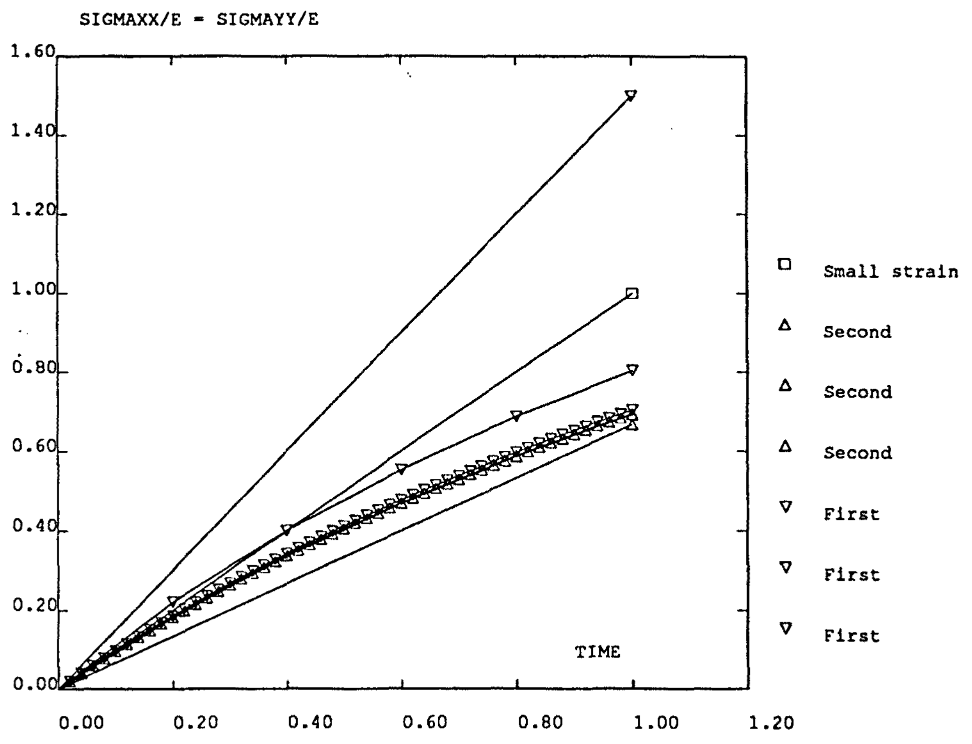

Figure 4.18 Dilatation test, elastic analysis. Normal stress vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step)

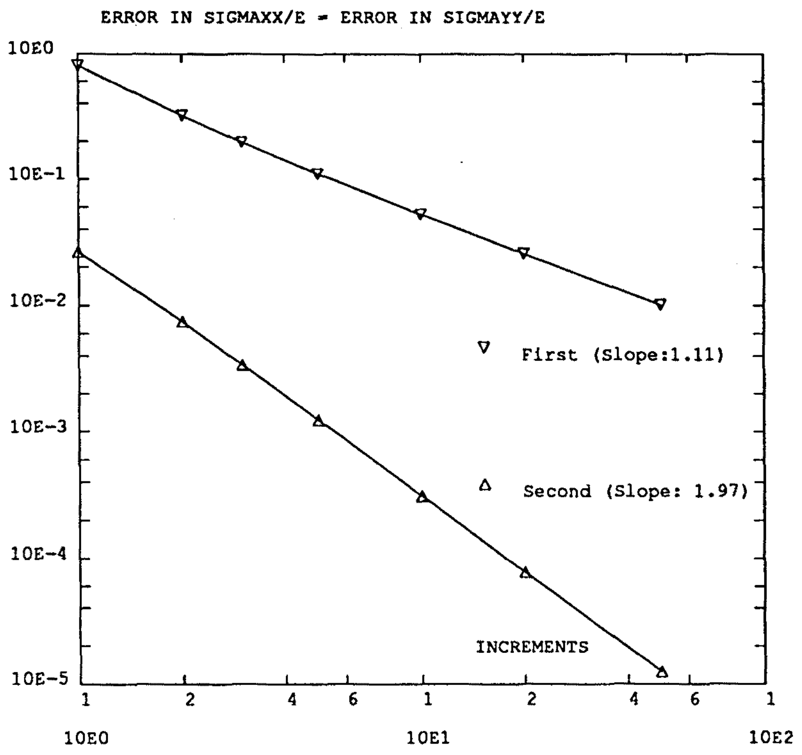

Figure 4.19 Dilatation test, elastic analysis. Error in the final value of normal stress ($t = 1$) vs. number of time-steps (1, 2, 3, 5, 10, 20, 50)

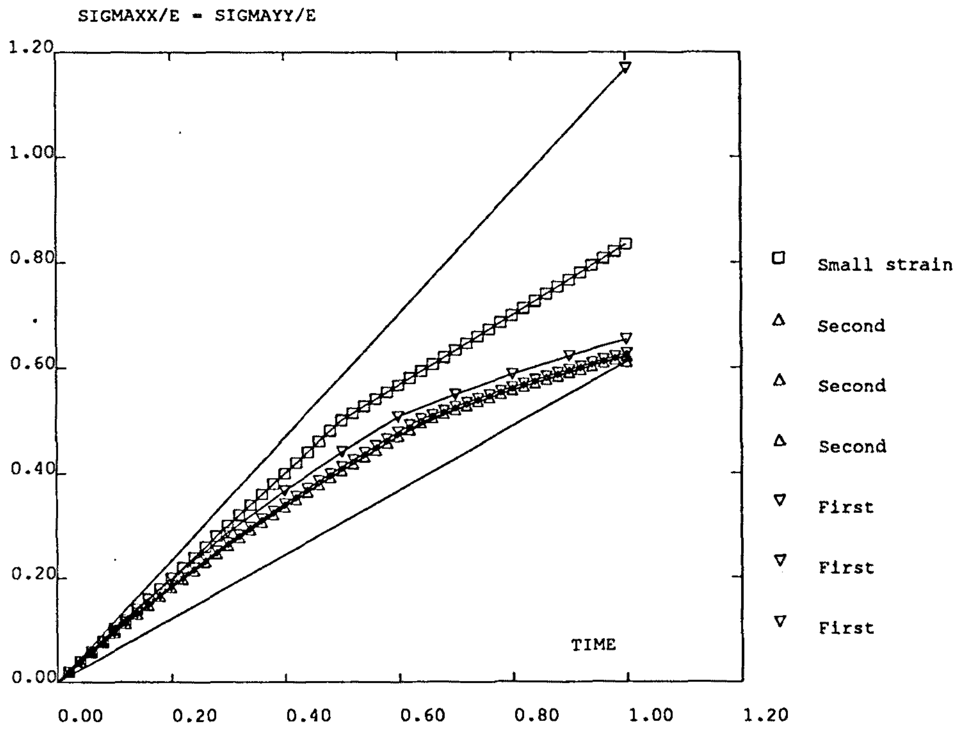

Figure 4.20 Dilatation test, elastoplastic analysis. Normal stress vs. time curves computed with the two algorithms (1, 10 and 50 time-steps) and small-strain analysis (50 time-steps)

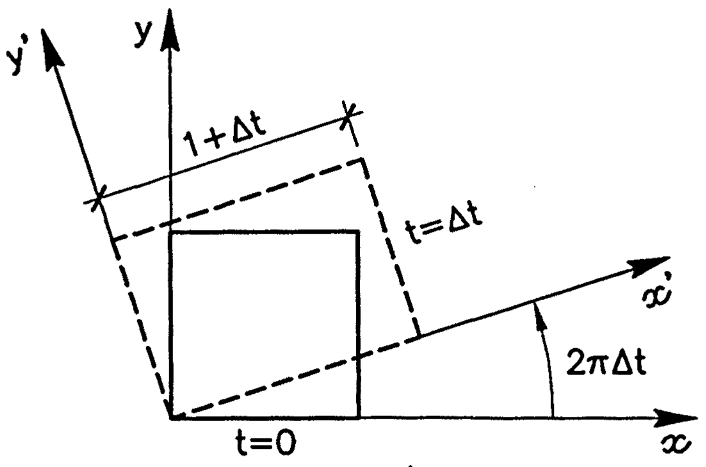

Figure 4.21 Extension and rotation test. Problem statement

4.5.6 Extension and rotation

In this last deformation path, a unit square undergoes a uniaxial extension and a superposed rigid rotation, see Figure 4.21. The equations of motion are

$$\left.\begin{aligned} x(t) &= X(1+t)\cos(2\pi t) - Y\sin(2\pi t) \\ y(t) &= X(1+t)\sin(2\pi t) + Y\cos(2\pi t) \end{aligned}\right\}, \tag{4.64}$$

and the analytical solution is

$$\left.\begin{aligned} \sigma_{xx}(t) &= Et\ \cos^2(2\pi t) \\ \sigma_{xy}(t) &= Et\ \sin^2(2\pi t) \\ \sigma_{yy}(t) &= Et\ \sin(2\pi t)\cos(2\pi t) \end{aligned}\right\}, \tag{4.65}$$

which is, as expected, a rotation of the solution to the uniaxial extension test, Eq. (4.59).

The output of the elastic analysis with 1, 5 and 50 time-steps can be seen in Figure 4.22. If only one increment is employed, the rotation part of the motion is not captured and the predicted stress is identical to that of the uniaxial extension test. With a higher number of steps, the comparative performance of the two algorithms is different from that of the previous tests. For the stress components σ_{xy} and σ_{yy}, for instance, the first algorithm is the one which predicts correct null final values for any number of time-increments. In fact, this result illustrates a more general behaviour. As shown in Appendix 4.B for a general shear-free deformation path (i.e., the original square is transformed into a rectangle), the first algorithm correctly predicts null shear stresses for any number of time-steps, while the second one does not.

As for the stress component σ_{xx}, the first algorithm performs better if a reduced number of time-steps (5) is employed, and a larger number is required for the second algorithm to produce more accurate results. This behaviour is illustrated by Figure 4.23a, where the error curves for σ_{xx} of the two algorithms intersect each other. Again, the observed order of both schemes (1.07 for the first one and 2.30 for the second one) is in accordance with the expected values. Figures 4.23b and 4.23c show the convergence behaviour of the second algorithm for the other two stress components. It can be seen that the convergence to the exact analytical value is very fast, especially for σ_{yy}, Figure 4.23c.

The outcome of the plastic analysis is depicted in Figure 4.24. Once again, a small-strain analysis with nonlinear material behaviour turns out to be completely unsatisfactory, providing a solution which is qualitatively different from that of a large-strain analysis.

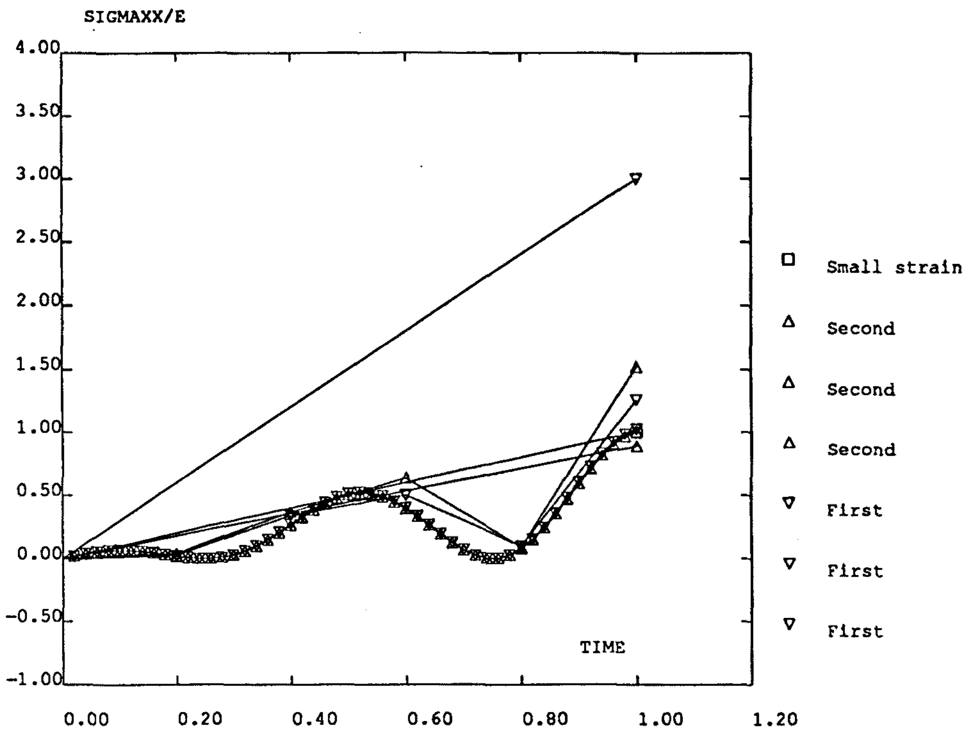

Fig. 4.22a

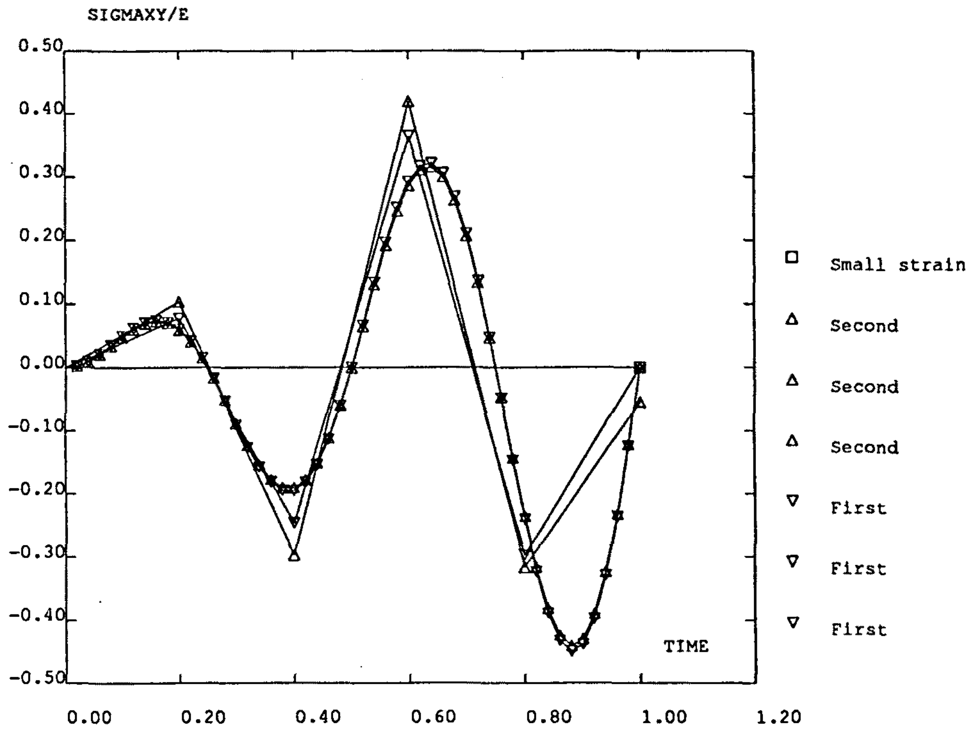

Fig. 4.22b

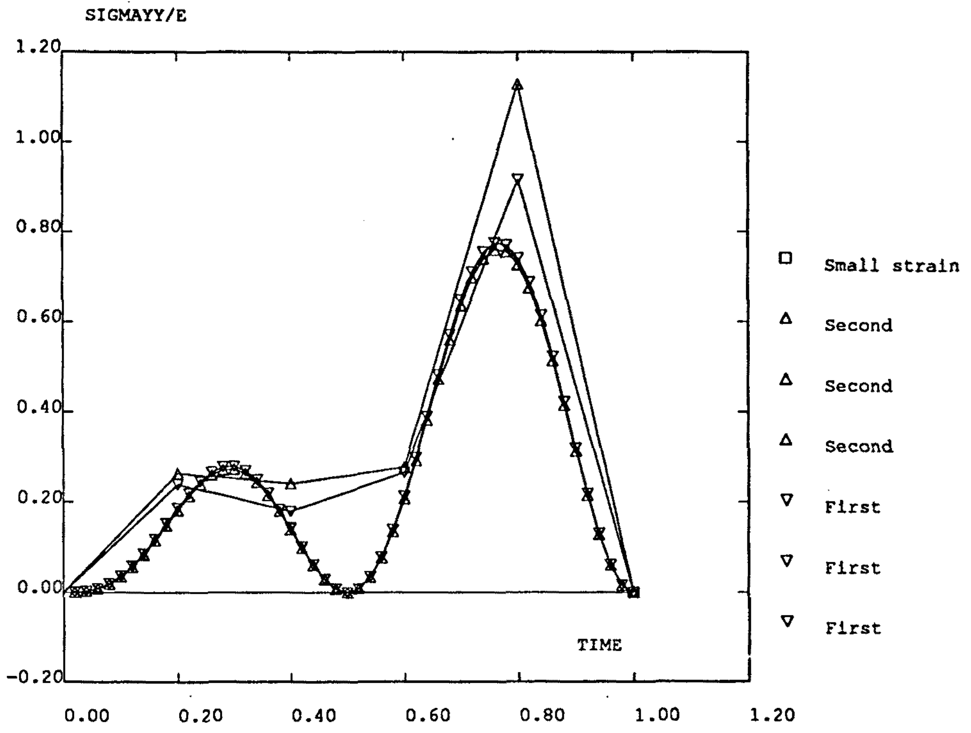

Fig. 4.22c

Figure 4.22 Extension and rotation test, elastic analysis. Stress vs. time curves computed with the two algorithms (1, 5 and 50 time-steps) and small-strain analysis (1 time-step). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}

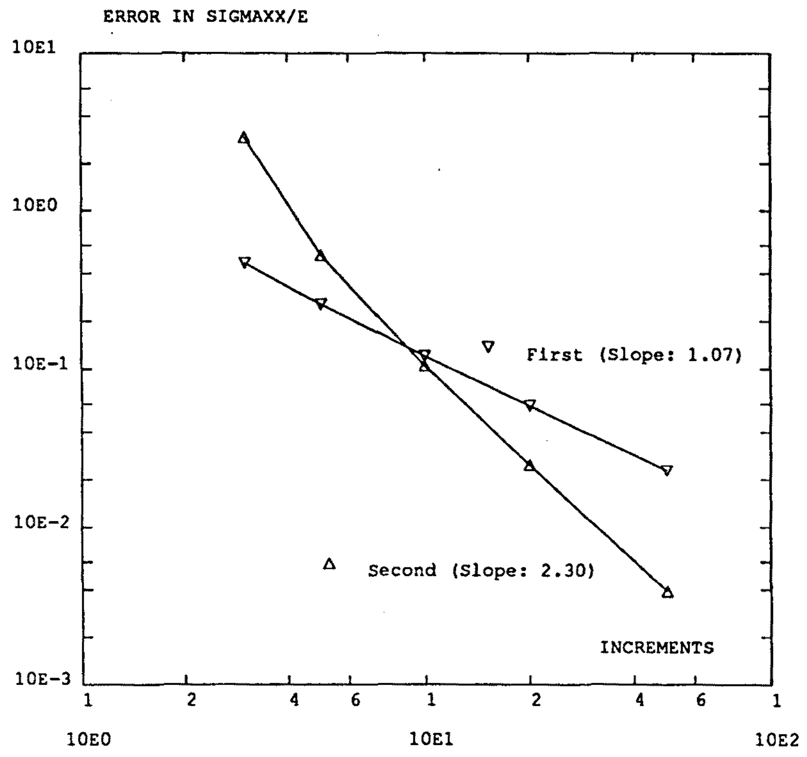

Fig. 4.23a

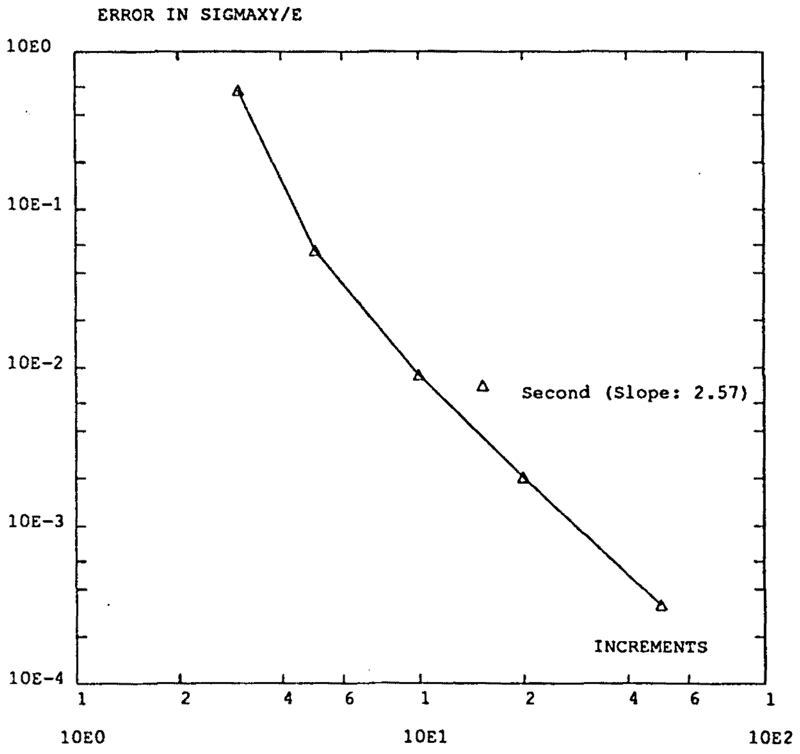

Fig. 4.23b

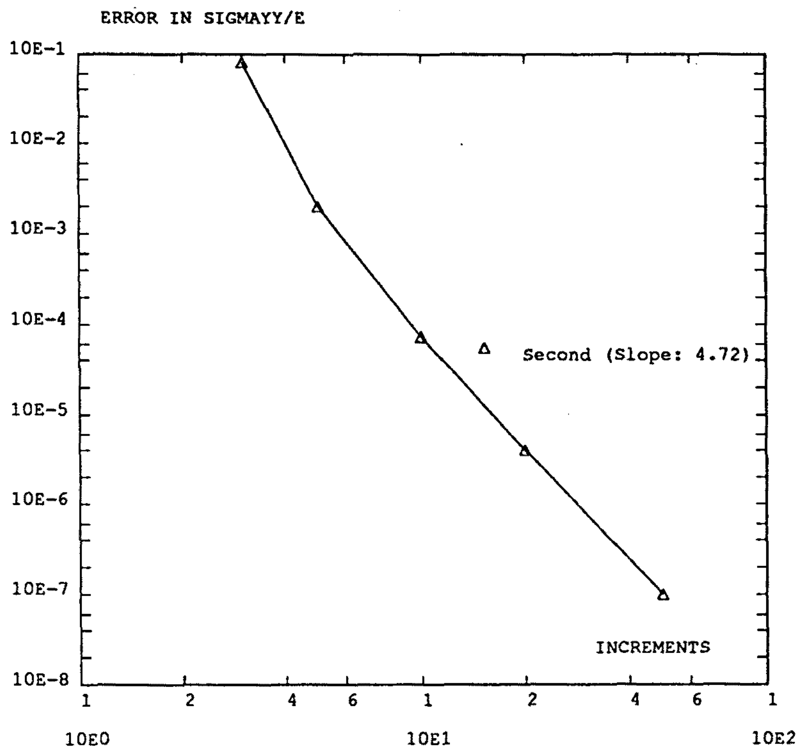

Fig. 4.23c

Figure 4.23 Extension and rotation test, elastic analysis. Error in the final value of stress ($t = 1$) vs. number of time-steps (3, 5, 10, 20, 50). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}

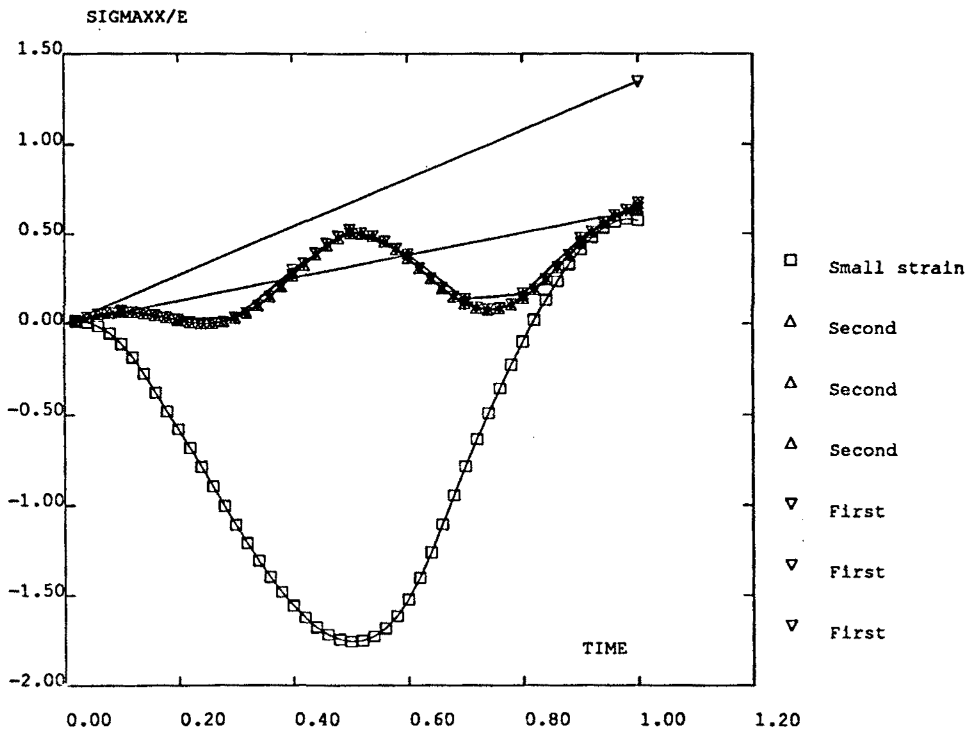

Fig. 4.24a

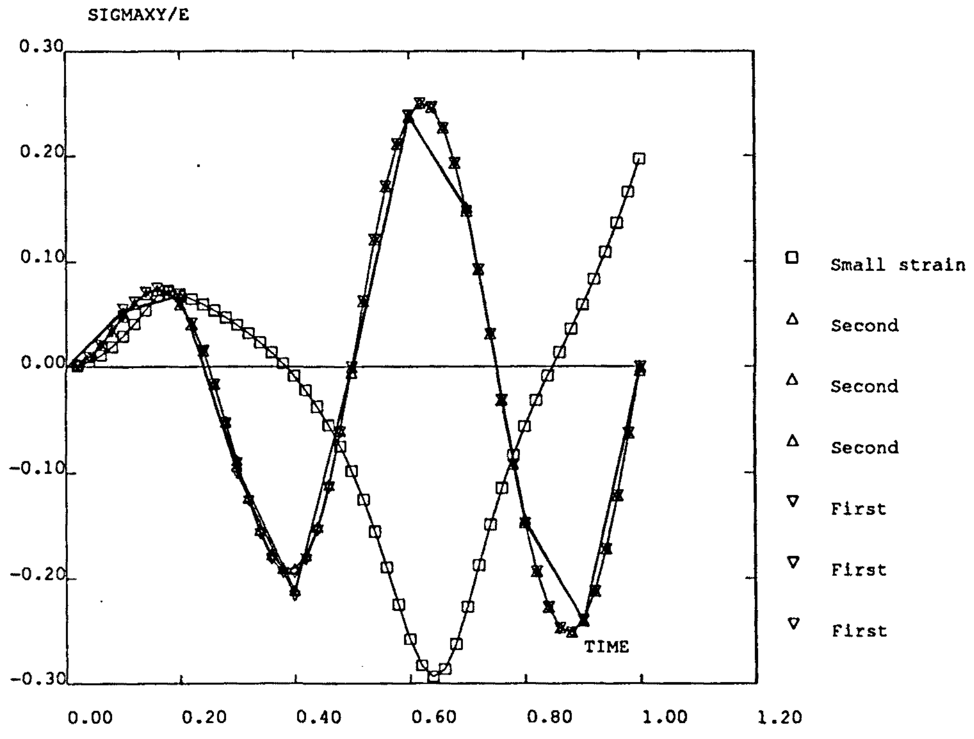

Fig. 4.24b

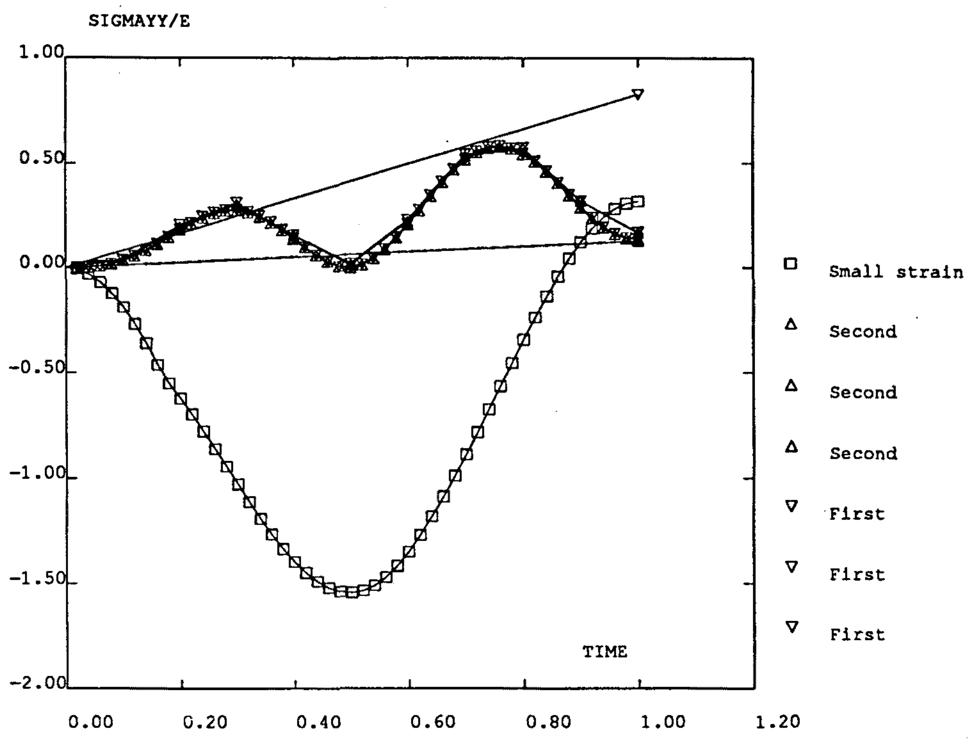

Fig. 4.24c

Figure 4.24 Extension and rotation test, elastoplastic analysis. Stress vs. time curves computed with the two algorithms (1, 10 and 50 time-steps) and small-strain analysis (50 time-steps). a) σ_{xx}. b) σ_{xy}. c) σ_{yy}