1 INTRODUCTION

O

VER the last four decades, the use of computational techniques based on the Finite Element Method has become a firmly established practice in the numerical solution of nonlinear solid mechanics problems both in academia and industry.

In their early days, these techniques were largely limited to infinitesimal deformation and strain problems with the main complexity arising from the nonlinear constitutive characterization of the underlying material by means of basic elastoplastic or elasto-viscoplastic theories. Applications were mostly confined to the modelling of the behaviour of solids in conventional areas of engineering and analyses were carried out on crude, user-unfriendly software that typically required highly specialized users. Since those days, this area of solid mechanics – generally known as computational plasticity – has experienced dramatic developments. Fuelled by the steady increase in computing power at decreasing costs together with the continuous industrial demand for accurate models of solids, the evolution of computational plasticity techniques have made possible the development of refined software packages with a considerable degree of automation that are today routinely employed by an ever-increasing number of engineers and scientists. The variety of practical problems of interest to which such techniques are currently applied with acceptable levels of predictive capability is very wide. They range from traditional engineering applications, such as stress analysis in structures, soil and rock mechanics, to the simulation of manufacturing processes such as metal forming. Also included are much less conventional applications, such as the simulation of food processing, mining operations and biological tissue behaviour. Many such problems are characterised by extremely large straining and material behaviour often described by means of rather complex constitutive equations.

1.1. Aims and scope

The main objective of this text is to describe in detail numerical techniques used in the small and large strain analysis of elastic and inelastic solids by the Finite Element Method. Particular emphasis is placed on the derivation and description of various constitutive models – based on phenomenological hyperelasticity, elastoplasticity and elasto-viscoplasticity – as well as on the relevant numerical procedures and the practical issues arising in their computer implementation. The range covered goes from basic infinitesimal isotropic to more sophisticated finite strain theories, including anisotropy. Many of the techniques discussed in the text are implemented in the FORTRAN computer program, named HYPLAS, which accompanies this book. Parts of its source code are included in the text and should help readers correlate the relevant numerical methods with their computer implementation in practice. Another important aspect to emphasise is that the performance of many of
the models/techniques described in the text is documented in numerical examples. These should be of particular relevance to those involved in software research and development in computational plasticity.

In order to make the book more self-contained, we have chosen to incorporate a considerable amount of basic theory within the text. This includes some material on elementary tensor analysis, an introduction to the nonlinear mechanics and thermodynamics of continuous media and an overview of small and large strain elastoplasticity and viscoplasticity theory, finite hyperelasticity and finite element techniques in nonlinear solid mechanics. Having a sound knowledge of such topics is essential, we believe, to the clear understanding of the very problems the numerical techniques discussed in this book are meant to simulate.

We reiterate, however, that our main focus here is computational. Thus, the volume of theory and the depth at which it is presented is kept to the minimum necessary for the above task. For example, in the presentation of tensor analysis and continuum mechanics and thermodynamics, we omit most proofs for standard relations. In plasticity, viscoplasticity and hyperelasticity, we limit ourselves mainly to presenting the constitutive models together with their most relevant properties and the essential relations needed in their formulation. Issues such as material stability and the existence and uniqueness of solutions to initial boundary value problems are generally not addressed.

1.1.1. READERSHIP

This book is intended for graduate students, research engineers and scientists working in the field of computational continuum mechanics. The text requires a basic knowledge of solid mechanics – especially the theory of linear elasticity – as well as the Finite Element Method and numerical procedures for the approximate solution of ordinary differential equations. An elementary understanding of vector and tensor calculus is also very helpful. Readers wishing to follow the computer implementation of the procedures described in the text should, in addition, be familiar at a fairly basic level with the FORTRAN computer programming language. It is worth remarking here that the choice of the FORTRAN language is motivated mainly by the following:

(a) its widespread acceptance in engineering computing in general and, in particular, within the finite element community;

(b) the suitability of procedural languages for codes with relatively low level of complexity, such as HYPLAS. In the present case, the use of more advanced programming concepts (e.g. object-oriented programming) could add a further difficulty in the learning of the essential concepts the HYPLAS code is meant to convey;

(c) its relative clarity in the coding of short algorithmic procedures such as those arising typically in the implementation of elastic and inelastic material models – the main subject of this book.

1.2. Layout

In line with the above aims, the book has been divided into three parts as follows.
• **Part One: Basic concepts.** In this part we introduce concepts of fundamental relevance to the applications presented in Parts Two and Three. The following material is covered:
  
  – elementary tensor analysis;
  – introductory continuum mechanics and thermodynamics;
  – finite elements in quasi-static nonlinear solid mechanics;
  – a concise description of the computer program HYPLAS.

• **Part Two: Small strains.** Here, the theory of infinitesimal plasticity is introduced together with the relevant numerical procedures used for its implementation into a finite element environment. A relatively wide range of models is presented, including both rate-independent (elastoplastic) and rate-dependent (elasto-viscoplastic) theories. The following main topics are considered:
  
  – the theory of infinitesimal plasticity;
  – finite elements in infinitesimal plasticity;
  – advanced plasticity models, including anisotropy;
  – viscoplasticity;
  – elastoplasticity with damage.

• **Part Three: Large strains.** This part focuses on finite strain hyperelasticity and elastoplasticity problems. The models discussed here, as well as their computational implementation, are obviously more complex than those of Part Two. Their complexity stems partly from the finite strain kinematics. Thus, to follow Part Three, a sound knowledge of the kinematics of finite deformations discussed in Chapter 3 (in Part One) is essential. The following topics are addressed:
  
  – large strain isotropic hyperelasticity;
  – large strain plasticity;
  – finite element techniques for large strain incompressibility;
  – single crystal (anisotropic) finite plasticity.

The material has been organised into sixteen chapters and four appendices. These will now be briefly described. The remainder of Chapter 1 discusses the general scheme of notation adopted in the book.

   Chapter 2 contains an introduction to elementary tensor analysis. In particular, the material is presented mainly in intrinsic (or compact) tensor notation – which is heavily relied upon throughout the book.

Chapter 3 provides an introduction to the mechanics and thermodynamics of continuous media. The material presented here covers the kinematics of deformation, balance laws and constitutive theory. These topics are essential for an in-depth understanding of the theories discussed in later chapters.

Chapter 4 shows the application of the Finite Element Method to the solution of problems in quasi-static nonlinear solid mechanics. A generic dissipative constitutive model, initially presented in Chapter 3, is used as the underlying material model.
Chapter 5 describes the general structure of the program HYPLAS, where many of the techniques discussed in the book are implemented. We remark that the program description is rather concise. Further familiarisation with the program will require the reader to follow the comments in the FORTRAN source code together with the cross-referencing of the main procedures with their description in the book. This is probably more relevant to those wishing to use the HYPLAS program for research and development purposes.

Chapter 6 is devoted to the mathematical theory of infinitesimal plasticity. The main concepts associated with phenomenological time-independent plasticity are introduced here. The basic yield criteria of Tresca, von Mises, Mohr–Coulomb and Drucker–Prager are reviewed, together with the most popular plastic flow rules and hardening laws.

In Chapter 7, we introduce the essential numerical methods required in the finite element solution of initial boundary value problems with elastoplastic underlying material models. Applications of the von Mises model with both isotropic and mixed isotropic/kinematic hardening are described in detail. The most relevant subroutines of the program HYPLAS are also listed and explained in detail.

Chapter 8 focuses on the detailed description of the implementation of the basic plasticity models based on the Tresca, Mohr–Coulomb and Drucker–Prager yield criteria. Again, the relevant subroutines of HYPLAS are listed and explained in some detail.

In Chapter 9 we describe the numerical treatment of plasticity models under plane stress conditions. Different options are considered and their relative merits and limitations are discussed. Parts of source code are also included to illustrate some of the most important programming aspects. The application of the concepts introduced here to other stress-constrained states is briefly outlined at the end of the chapter.

In Chapter 10 advanced elastoplasticity models are considered. Here we describe the computational implementation of a modified Cam-Clay model for soils, a capped Drucker–Prager model for geomaterials and the Hill, Hoffman and Barlat–Lian anisotropic models for metals. The numerical techniques required for the implementation of such models are mere specialisations of the procedures already discussed in Chapters 7 and 8. However, due to the inherent complexity of the models treated in this chapter, their actual implementation is generally more intricate than those of the basic models.

Chapter 11 begins with an introduction to elasto-viscoplasticity theory within the constitutive framework for dissipative materials described in Chapter 3. The (rate-independent) plasticity theory is then obtained as a limiting case of viscoplasticity. The numerical methods for a generic viscoplastic model are described, following closely the procedures applied earlier in elastoplasticity. Application of the methodology to von Mises criterion-based viscoplastic models is described in detail.

In Chapter 12 we discuss continuum damage mechanics – the branch of Continuum Solid Mechanics devoted to the modelling of the progressive material deterioration that precedes the onset of macroscopic fracturing. Some elastoplastic damage models are reviewed and their implementation, with the relevant computational issues, is addressed in detail.

Chapter 13 introduces finite strain hyperelasticity. The basic theory is reviewed and some of the most popular isotropic models are presented. The finite element implementation of the Odgen model is discussed in detail with relevant excerpts of HYPLAS source code included. In addition, the modelling of the so-called Mullins dissipative effect by means of a hyperelastic-damage theory is addressed at the end of the chapter. This concept is closely related to those already discussed in Chapter 12 for ductile elastoplastic damage.
In Chapter 14 we introduce finite strain elastoplasticity together with the numerical procedures relevant to the finite element implementation of finite plasticity models. The main discussion is focused on hyperelastic-based finite plasticity theories with multiplicative kinematics. The finite plasticity models actually implemented in the program HYPLAS belong to this class of theories. However, for completeness, a discussion on the so-called hypoelastic-based theories is also included. The material presented is confined mostly to isotropic elastoplasticity, with anisotropy in the form of kinematic hardening added only at the end of the chapter.

Chapter 15 is concerned with the treatment of large strain incompressibility within the Finite Element Method. This issue becomes crucial in large-scale finite strain simulations where the use of low-order elements (which, without any added specific techniques, are generally inappropriate near the incompressible limit) is highly desirable. Three different approaches to tackle the problem are considered: the so-called $F$-bar method (including its more recent $F$-bar-Patch variant for simplex elements); the Enhanced Assumed Strain (EAS) technique, and the mixed $u/p$ formulation.

Finally, in Chapter 16 we describe a general model of large-strain single-crystal plasticity together with the relevant numerical procedures for its use within a finite element environment. The implementation of a specialisation of the general model based on a planar double-slip system is described in detail. This implementation is incorporated in the program HYPLAS.

In addition to the above, four appendices are included. Appendix A is concerned with isotropic scalar- and tensor-valued functions of a symmetric tensor that are widely exploited throughout the text. It presents some important basic properties as well as formulae that can be used in practice for the computation of function values and function derivatives. Appendix B addresses the tensor exponential function. The tensor exponential is of relevance for the treatment of finite plasticity presented in Chapters 14 and 16. In Appendix C we derive the linearisation of the virtual work equation both under small and large deformations. The expressions derived here provide the basic formulae for the tangent operators required in the assembly of the tangent stiffness matrix in the finite element context. Finally, in Appendix D we describe the handling – including array storage and product operations – of second and fourth-order tensors in finite element computer programs.

1.2.1. THE USE OF BOXES

Extensive use of boxes has been made to summarise constitutive models and numerical algorithms in general (refer, for example, to pages 146 and 199). Boxes should be of particular use to readers interested mostly in computer implementation aspects and who wish to skip the details of derivation of the models and numerical procedures. Numerical algorithms listed in boxes are presented in the so-called pseudo-code format – a format that resembles the actual computer code of the procedure. For boxes describing key procedures implemented in the program HYPLAS, the name of the corresponding FORTRAN subprogram is often indicated at the top of the box (see page 221, for instance).

1.3. General scheme of notation

Throughout the text, an attempt has been made to maintain the notation as uniform as possible by assigning specific letter styles to the representation of each type of mathematical entity.
At the same time, we have tried to keep the notation in line with what is generally adopted in the present subject areas. Unfortunately, these two goals often conflict so that, in many cases, we choose to adopt the notation that is more widely accepted instead of adhering to the use of specific fonts for specific mathematical entities. Whenever such exceptions occur, their meaning should either be clear from the context or will be explicitly mentioned the first time they appear.

1.3.1. CHARACTER FONTS. GENERAL CONVENTION

The mathematical meaning associated with specific font styles is given below. Some important exceptions are also highlighted. We emphasise, however, that other exceptions, not mentioned here, may occur in the text.

- **Italic light-face letters** $A, a, \ldots$: scalars and scalar-valued functions.

- **Italic bold-face majuscules** $B, C, \ldots$: second-order tensors or tensor-valued functions. *Light-face with indices* $B_{ij}, B_{\alpha\beta}, \ldots$: components of the corresponding tensors. *Important exceptions*: $A$ (set of thermodynamical forces), $H$ (generalised elastoplastic hardening modulus), $J$ (generalised viscoplastic hardening constitutive function).

- **Italic bold-face minuscules** $p, v, \ldots$: points, vectors and vector-valued functions. *Light-face with indices* $p_i, p_{\alpha}, \ldots$: coordinates (components) of the corresponding points (vectors). *Important exception*: $s$ (stress tensor deviator).

- **Sans-serif (upright) bold-face letters** $A, a, \ldots$: fourth-order tensors. *Light-face with indices* $A_{ijkl}, A_{\alpha\beta\gamma\delta}, \ldots$: the corresponding Cartesian components.

- **Greek light-face letters** $\alpha, \beta , \ldots, \Phi, \Psi, \ldots$: scalars and scalar-valued functions. *Important exception*: $\Omega$ (region of Euclidean space occupied by a generic body).

- **Greek bold-face minuscules** $\beta, \sigma, \tau, \ldots$: second-order tensors. *Light-face with indices* $\sigma_{ij}, \sigma_{\alpha\beta}, \ldots$: the corresponding components. *Important exceptions*: $\alpha$ (generic set of internal state variables), $\varphi$ (deformation map) and $\eta$ (when meaning virtual displacement fields).

- **Upright bold-face majuscules, minuscules and greek letters** $A, a, \sigma, \ldots$: finite element arrays (vectors and matrices) representing second or fourth-order tensors and general finite element operators.

- **Script majuscules** $\mathcal{A}, \mathcal{B}, \ldots$: spaces, sets, groups, bodies.

- **German majuscules** $\mathfrak{F}, \mathfrak{G}, \ldots$: constitutive response functionals.

- **Calligraphic majuscules** $\mathcal{X}, \mathcal{Y}, \ldots$: generic mathematical entity (scalar, vector, tensor, field, etc.)

- **Typewriter style letters** HYPLAS, SUVM, \ldots: used exclusively to denote FORTRAN procedures and variable names, instructions, etc.
1.3.2. SOME IMPORTANT CHARACTERS

The specific meaning of some important characters is listed below. We remark that some of these symbols may occasionally be used with a different connotation (which should be clear from the context).

- \(A\)  
  Generic set of thermodynamical forces

- \(A\)  
  Finite element assembly operator (note the large font)

- \(A\)  
  First elasticity tensor

- \(a\)  
  Spatial elasticity tensor

- \(B\)  
  Left Cauchy–Green strain tensor

- \(B^e\)  
  Elastic left Cauchy–Green strain tensor

- \(B\)  
  Discrete (finite element) symmetric gradient operator  
  (strain-displacement matrix)

- \(\mathcal{B}\)  
  Generic body

- \(b\)  
  Body force

- \(\bar{b}\)  
  Reference body force

- \(C\)  
  Right Cauchy–Green strain tensor

- \(c\)  
  Cohesion

- \(D\)  
  Damage internal variable

- \(D\)  
  Stretching tensor

- \(D^e\)  
  Elastic stretching

- \(D^p\)  
  Plastic stretching

- \(D\)  
  Infinitesimal consistent tangent operator

- \(\mathcal{D}^e\)  
  Infinitesimal elasticity tensor

- \(D^{ep}\)  
  Infinitesimal elastoplastic consistent tangent operator

- \(D\)  
  Consistent tangent matrix (array representation of \(D\))

- \(\mathcal{D}^e\)  
  Elasticity matrix (array representation of \(\mathcal{D}^e\))

- \(D^{ep}\)  
  Elastoplastic consistent tangent matrix (array representation of \(D^{ep}\))

- \(E\)  
  Young’s modulus

- \(E_i\)  
  Eigenprojection of a symmetric tensor associated with the \(i^{th}\)  
  eigenvalue

- \(\mathcal{E}\)  
  Three-dimensional Euclidean space; elastic domain

- \(\mathcal{E}\)  
  Set of plastically admissible stresses

- \(e_i\)  
  Generic base vector; unit eigenvector of a symmetric tensor  
  associated with the \(i^{th}\)  
  eigenvalue
\( F \) Deformation gradient
\( F^e \) Elastic deformation gradient
\( F^p \) Plastic deformation gradient
\( \hat{\mathbf{f}}^{\text{ext}} \) Global (finite element) external force vector
\( \mathbf{f}^{\text{ext}}(e) \) External force vector of element \( e \)
\( \mathbf{f}^{\text{int}} \) Global (finite element) internal force vector
\( \mathbf{f}^{\text{int}}(e) \) Internal force vector of element \( e \)
\( G \) Virtual work functional; shear modulus
\( H \) Hardening modulus
\( \mathbf{H} \) Generalised hardening modulus
\( I_1, I_2, I_3 \) Principal invariants of a tensor
\( I \) Fourth-order identity tensor: \( I_{ijkl} = \delta_{ik}\delta_{jl} \)
\( I_S \) Fourth-order symmetric identity tensor: \( I_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \)
\( I_d \) Deviatoric projection tensor: \( I_d \equiv I_S - \frac{1}{3} I \otimes I \)
\( I \) Second-order identity tensor
\( I_S \) Array representation of \( I_S \)
\( i \) Array representation of \( I \)
\( J \) Jacobian of the deformation map: \( J \equiv \det F \)
\( J_2, J_3 \) Stress deviator invariants
\( \mathbf{J} \) Generalised viscoplastic hardening constitutive function
\( K \) Bulk modulus
\( K_T \) Global tangent stiffness matrix
\( K_T(e) \) Tangent stiffness matrix of element \( e \)
\( \mathcal{K} \) Set of kinematically admissible displacements
\( L \) Velocity gradient
\( L^e \) Elastic velocity gradient
\( L^p \) Plastic velocity gradient
\( m^\alpha \) Unit vector normal to the slip plane \( \alpha \) of a single crystal
\( N \) Plastic flow vector
\( \bar{N} \) Unit plastic flow vector: \( \bar{N} \equiv N/\|N\| \)
\( \theta \) The orthogonal group
\( \theta^+ \) The rotation (proper orthogonal) group
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Zero tensor; zero array; zero generic entity</td>
</tr>
<tr>
<td>α</td>
<td>Zero vector</td>
</tr>
<tr>
<td>P</td>
<td>First Piola–Kirchhoff stress tensor</td>
</tr>
<tr>
<td>p</td>
<td>Generic material point</td>
</tr>
<tr>
<td>p</td>
<td>Cauchy or Kirchhoff hydrostatic pressure</td>
</tr>
<tr>
<td>Q</td>
<td>Generic orthogonal or rotation (proper orthogonal) tensor</td>
</tr>
<tr>
<td>q</td>
<td>von Mises (Cauchy or Kirchhoff) effective stress</td>
</tr>
<tr>
<td>R</td>
<td>Rotation tensor obtained from the polar decomposition of ( F )</td>
</tr>
<tr>
<td>( R^e )</td>
<td>Elastic rotation tensor</td>
</tr>
<tr>
<td>( \mathbb{R} )</td>
<td>Real set</td>
</tr>
<tr>
<td>r</td>
<td>Global finite element residual (out-of-balance) force vector</td>
</tr>
<tr>
<td>s</td>
<td>Entropy</td>
</tr>
<tr>
<td>s</td>
<td>Cauchy or Kirchhoff stress tensor deviator</td>
</tr>
<tr>
<td>( s^\alpha )</td>
<td>Unit vector in the slip direction of slip system ( \alpha ) of a single crystal</td>
</tr>
<tr>
<td>t</td>
<td>Surface traction</td>
</tr>
<tr>
<td>( \bar{t} )</td>
<td>Reference surface traction</td>
</tr>
<tr>
<td>U</td>
<td>Right stretch tensor</td>
</tr>
<tr>
<td>( U^e )</td>
<td>Elastic right stretch tensor</td>
</tr>
<tr>
<td>( U^p )</td>
<td>Plastic right stretch tensor</td>
</tr>
<tr>
<td>( \mathcal{U} )</td>
<td>Space of vectors in ( \mathcal{E} )</td>
</tr>
<tr>
<td>u</td>
<td>Generic displacement vector field</td>
</tr>
<tr>
<td>u</td>
<td>Global finite element nodal displacement vector</td>
</tr>
<tr>
<td>V</td>
<td>Left stretch tensor</td>
</tr>
<tr>
<td>( V^e )</td>
<td>Elastic left stretch tensor</td>
</tr>
<tr>
<td>( V^p )</td>
<td>Plastic left stretch tensor</td>
</tr>
<tr>
<td>( \mathcal{V} )</td>
<td>Space of virtual displacements</td>
</tr>
<tr>
<td>v</td>
<td>Generic velocity field</td>
</tr>
<tr>
<td>W</td>
<td>Spin tensor</td>
</tr>
<tr>
<td>( W^e )</td>
<td>Elastic spin tensor</td>
</tr>
<tr>
<td>( W^p )</td>
<td>Plastic spin tensor</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>Generic point in space</td>
</tr>
<tr>
<td>( \alpha_p )</td>
<td>Ogden hyperelastic constants ((p = 1, \ldots, N)) for a model with ( N ) terms in the Ogden strain-energy function series</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>Generic set of internal state variables</td>
</tr>
</tbody>
</table>
\( \beta \) Back-stress tensor

\( \dot{\gamma} \) Plastic multiplier

\( \delta_{ij} \) Krönecker delta

\( \varepsilon \) Strain tensor; also Eulerian logarithmic strain when under large strains

\( \varepsilon^e \) Elastic strain tensor; also elastic Eulerian logarithmic strain when under large strains

\( \varepsilon^p \) Plastic strain tensor

\( \varepsilon, \varepsilon^e, \varepsilon^p \) Array representation of \( \varepsilon \), \( \varepsilon^e \) and \( \varepsilon^p \), respectively

\( \varepsilon, \varepsilon^e, \varepsilon^p \) Axial total, elastic and plastic strain in one-dimensional models

\( \bar{\varepsilon}^p \) Effective (or accumulated) plastic strain

\( \eta \) Virtual displacement field; relative stress tensor in kinematic hardening plasticity models

\( \kappa \) Isotropic hardening thermodynamical force

\( \lambda \) One of the Lamé constants of linear elasticity; axial stretch; load factor in proportional loading

\( \lambda^e, \lambda^p \) Elastic and plastic axial stretch

\( \lambda_i, \lambda_i^e, \lambda_i^p \) Total, elastic and plastic principal stretches

\( \mu \) One of the Lamé constants of linear elasticity

\( \mu_p \) Ogden hyperelastic constants \((p = 1, \ldots, N)\) for a model with \( N \) terms in the Ogden strain-energy function series

\( \nu \) Poisson ratio

\( \Xi \) Dissipation potential

\( \xi \) Isoparametric coordinates of a finite element

\( \rho \) Mass density

\( \bar{\rho} \) Reference mass density

\( \sigma \) Cauchy stress tensor

\( \sigma \) Axial stress in one-dimensional models

\( \sigma_i \) Principal Cauchy stress

\( \sigma_y \) Yield stress (uniaxial yield stress for the conventional von Mises and Tresca models)

\( \sigma_{y0} \) Initial yield stress

\( \sigma \) Array representation of \( \sigma \)

\( \tau \) Kirchhoff stress tensor

\( \tau_i \) Principal Kirchhoff stress
1.3.3. INDICIAL NOTATION, SUBSCRIPTS AND SUPERSCRIPTS

When indicial notation is used, the following convention is adopted for subscripts:

- **Italic subscripts** $i, j, k, l, \ldots$, as in the Cartesian components
  
  $$u_i, \ B_{ij}, \ a_{ijkl},$$

  or for the basis vectors
  
  $$e_i,$$

  normally range over 1, 2 and 3. In a more general context (in an $n$-dimensional space),
  their range may be 1, 2, \ldots, $n$.

- **Greek subscripts** $\alpha, \beta, \gamma, \delta, \ldots$: range over 1 and 2.

- When an index appears twice in the same product, summation over the repeated index
  (Einstein notation) is implied unless otherwise stated. For example,

  $$u_i e_i = \sum_{i=1}^{3} u_i e_i.$$  

We remark that subscripts are not employed exclusively in connection with indicial notation.
Different connotations are assigned to subscripts throughout the text and the actual meaning
of a particular subscript should be clear from the context. For example, in the context of
incremental numerical procedures, subscripts may indicate the relevant increment number. In
the expression

$$\Delta \varepsilon = \varepsilon_{n+1} - \varepsilon_n,$$

the subscripts $n$ and $n + 1$ refer to the values of $\varepsilon$, respectively, at the end of increments $n$
and $n + 1$.

**Superscripts**

Superscripts are also used extensively throughout the text. The meaning of a particular
superscript will be stated the first time it appears in the text and should be clear from
the context thereafter.
1.3.4. OTHER IMPORTANT SYMBOLS AND OPERATIONS

The meanings of other important symbols and operations are listed below.

\[ \det(\cdot) \quad \text{Determinant of } (\cdot) \]
\[ \operatorname{dev}(\cdot) \quad \text{Deviator of } (\cdot) \]
\[ \operatorname{div}_p(\cdot) \quad \text{Material divergence of } (\cdot) \]
\[ \operatorname{div}_x(\cdot) \quad \text{Spatial divergence of } (\cdot) \]
\[ \exp(\cdot) \quad \text{Exponential (including tensor exponential) of } (\cdot) \]
\[ \ln(\cdot) \quad \text{Natural logarithm (including tensor logarithm) of } (\cdot) \]
\[ o(\cdot) \quad \text{A term that vanishes faster than } (\cdot) \]
\[ \operatorname{sign}(\cdot) \quad \text{The signum function: } \operatorname{sign}(\cdot) \equiv (\cdot)/|\cdot| \]
\[ \text{skew}(\cdot) \quad \text{Skew-symmetric part of } (\cdot) \]
\[ \text{sym}(\cdot) \quad \text{Symmetric part of } (\cdot) \]
\[ \operatorname{tr}(\cdot) \quad \text{Trace of } (\cdot) \]
\[ \Delta(\cdot) \quad \text{Increment of } (\cdot). \text{ Typically, } \Delta(\cdot) = (\cdot)_{n+1} - (\cdot)_n \]
\[ \delta(\cdot) \quad \text{Iterative increment of } (\cdot) \]
\[ \nabla(\cdot) \quad \text{Gradient of } (\cdot) \]
\[ \nabla_p(\cdot) \quad \text{Material gradient of } (\cdot) \]
\[ \nabla_x(\cdot) \quad \text{Spatial gradient of } (\cdot) \]
\[ \nabla^s, \nabla^s_p, \nabla^s_x(\cdot) \quad \text{Corresponding symmetric gradients of } (\cdot) \]
\[ \partial(\cdot) \quad \text{Boundary of the domain } (\cdot) \]
\[ \partial_a(\cdot) \quad \text{Subdifferential of } (\cdot) \text{ with respect to } a \]
\[ \frac{\partial}{\partial a}(\cdot) \quad \text{Derivative of } (\cdot) \text{ with respect to } a \]

\[ \dot{\cdot} \quad \text{Material time derivative of } (\cdot) \]
\[ (\cdot)^T \quad \text{The transpose of } (\cdot) \]
\[ a \equiv b \quad \text{Means } a \text{ is defined as } b. \text{ The symbol } \equiv \text{ is often used to emphasise that} \]
\[ \text{the expression in question is a definition.} \]
\[ a := b, \quad \text{Assignment operation. The value of the right-hand side of the expression is assigned to its left-hand side. The symbol := is often used to emphasise that a given expression is an assignment operation performed by a computational algorithm.} \]
\[ a := a + b \]
\[ S : T, S : T, \quad \text{Double contraction of tensors (internal product of second-order tensors)} \]
\[ S : T \]
**INTRODUCTION**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u \cdot v, T \cdot u, S \cdot T$</td>
<td>Single contraction of vectors and tensors. The single contraction symbol (the single dot) is usually omitted in single contractions between a tensor and a vector or between tensors; that is, $T \cdot u$ and $S \cdot T$ are normally represented simply as $Tu$ and $ST$.</td>
</tr>
<tr>
<td>$u \times v$</td>
<td>Vector product</td>
</tr>
<tr>
<td>$S \otimes T, u \otimes v$</td>
<td>Tensor product of tensors or vectors</td>
</tr>
<tr>
<td>$\mathcal{X} \ast \mathcal{Y}$</td>
<td>The appropriate product between two generic entities, $\mathcal{X}$ and $\mathcal{Y}$, in a given context.</td>
</tr>
<tr>
<td>$</td>
<td>\cdot</td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td>$\rightarrow A$</td>
<td>Used in the description of arguments of FORTRAN subprograms listed in the text. An arrow pointing to the right followed by an argument name $A$ means that the value of $A$ at entry is used by the relevant subprogram and is not changed during its execution.</td>
</tr>
<tr>
<td>$\leftarrow A$</td>
<td>Analogously to $\rightarrow$ above, an arrow pointing to the left followed by an argument name $A$ means that the value of $A$ is calculated and returned by the relevant subprogram and its value at entry is ignored.</td>
</tr>
<tr>
<td>$\leftrightarrow A$</td>
<td>Analogously to $\rightarrow$ and $\leftarrow$ above, a double arrow followed by an argument name $A$ means that the value of $A$ at entry is used by the relevant subprogram and is changed during its execution.</td>
</tr>
</tbody>
</table>