1

Preliminaries

A journey of a million miles
starts with the first step

Confucius

1.1 INTRODUCTION

There are many textbooks which describe the mathematical background of boundary element methods (the interested reader may refer to an excellent compilation of literature on the BEM in the BENET homepage *http://www.ce.udel.edu/faculty/cheng/benet*). Unfortunately, most texts to date have been written by mathematicians or engineers with a strong background in mathematics and, therefore, they tend to dwell on the theoretical treatment of the method rather than concentrating on the physical meaning and implementation. Furthermore, although many books include simple programs in the appendix few have examples on how the theory explained is translated into a computer program. Since it is obvious that the methods would not be useful without computers, the lack of emphasis on computer implementation is surprising. In contrast to most mathematicians, who are happy just to prove the existence of a solution and error bounds, the engineer is interested in the application of the method in solving real problems. Invariably this means that either an existing program must be used or a program developed for solving a specific problem. Either way, a user of boundary element software must have a basic understanding about how the theory is implemented, as it is important that one be aware of the limitations of the program, possible pitfalls to look out for, etc. So in addition to understanding the theoretical assumptions made to obtain an approximate solution of the problem and the theoretical error bounds, such as they may be derived for certain examples, the user must be aware of the errors introduced by the numerical implementation, for example, the numerical integration and boundary interpolation.

Most readers of this book will be familiar with the finite element method[1]. In the most common version of this method in elasticity, we subdivide the domain into elements and approximate the displacement field with functions which are defined at element level, i.e. are piecewise continuous. The parameters of these functions, which are the displacements at the nodes where elements are connected to each other, are determined by a suitable general equilibrium condition, such as the minimum of potential energy. The functions themselves generally do not satisfy the governing differential equations. Therefore, for elasticity problems, we may have overall equilibrium satisfied, but a local violation of equilibrium exists, for example at element boundaries. It can be proven that the method converges, i.e. the exact solution is approached as the element size approaches zero.

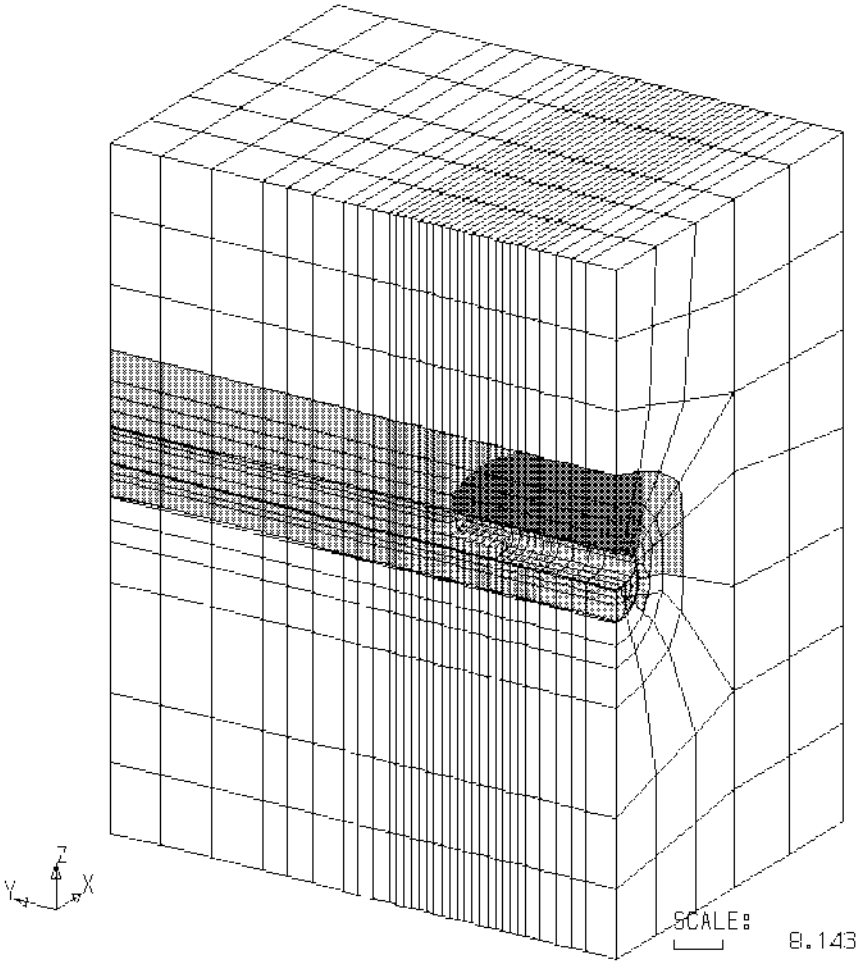

Figure 1.1 Finite element mesh for the analysis of tunnel excavation with contours of z-displacements on elements near tunnel

Figure 1.1 shows an example of a finite element mesh for the three-dimensional analysis of sequential excavation and construction of a tunnel. A plane of symmetry is applied so that only half of the tunnel is discretised. Note that to model the rock mass through which the tunnel is driven, which for all practical purposes can be assumed to be infinite, we must make a 'box' of solid elements. At the outer boundaries of this box we either set the displacements to zero or apply stress boundary conditions which represent the *in situ* stress. The mesh shown here has approximately 100 000 degrees of freedom, and a solution took several hours on a PC.

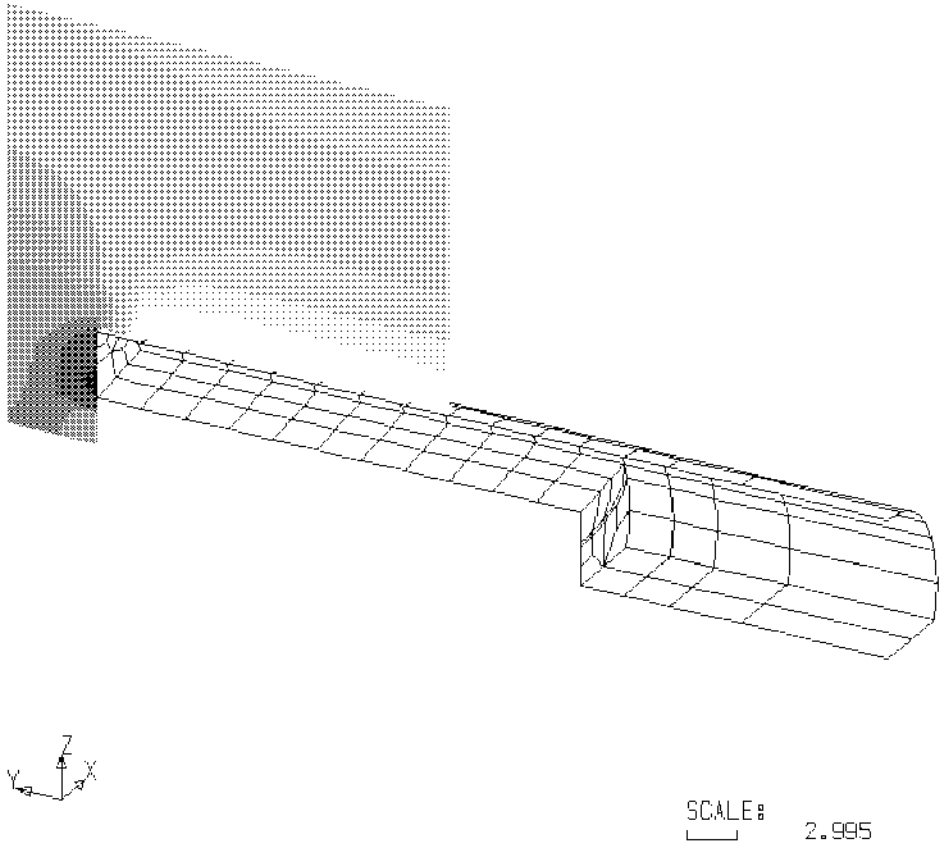

Figure 1.2 Boundary element mesh for the simulation of tunnel excavation with contours of σ_z on a result plane

In contrast, the BEM does not require the subdivision of the domain into elements because the functions used for approximating the solution inside the domain are chosen to be only those which satisfy the governing differential equations exactly. Consequently, only boundary conditions have to be satisfied. This can be achieved in an approximate way by subdividing the boundary into elements over which the values (for example, tractions or displacements in the case of elasticity) are interpolated, much in the same way

as with the FEM. The advantage of the method is obvious: the dimensionality of the problem is reduced by one order, that is, only a surface instead of a volume discretisation is required. This means that the number of unknowns is reduced dramatically, especially for three-dimensional problems, because unknowns occur only on the problem boundary. Another advantage is that equilibrium is satisfied exactly everywhere in the domain, and that it is easy to deal with an infinite continuum.

As an example, Figure 1.2 shows the boundary element mesh for the same tunnel as analysed by the FEM in Figure 1.1. This mesh has approximately 1000 degrees of freedom and takes 3 minutes to solve on a PC. The stress contours, computed and drawn on a user-defined plane inside the rock mass, show no jumps, as are sometimes seen in FEM results if no special stress recovery algorithms are used to smooth results.

Since functions must be found which satisfy the governing differential equation (DE) exactly, the BEM requires a solution of the DE. This solution must be as simple as possible because, as will be seen in the chapter on implementation, this is crucial for efficiency. Unfortunately, the simplest solutions which we can find (fundamental solutions) are due to concentrated loads or sources and are singular, i.e., have infinite values at certain points. This property has to be taken into account when integrating these functions over boundary elements. This will make the numerical integration procedure more complicated than is the case with finite elements.

There has been a general misconception that because a fundamental solution of the problem must exist for the BEM to work, the method can only be applied to linear problems with homogeneous material. As will be shown in this book, non-linear problems can be solved almost as easily as with the FEM by the repeated solution of linear problems and inhomogeneous material can be considered with the multi-region method.

1.2 OVERVIEW OF BOOK

This book is designed to be used as a basis for a first course on the BEM. It is recommended that the chapters be read consecutively, as later chapters build on material discussed in earlier chapters. Throughout the book the reader will build a suite of subprograms which perform the various tasks needed for the numerical implementation of the BEM. Various exercises are included which allow the reader to test the programs written and to experience how the method works. Answers to exercises are given in the Appendix.

We start with an introduction to the FORTRAN 95 programming language, the latest dialect of FORTRAN which was released at the time of writing the book. This is a major improvement over the old FORTRAN 77 variant in that it has extensive facilities for matrix operations which result in much shorter and more readable code. FORTRAN, which stands for FORmula TRANslation, is the most widely used language for programming engineering applications and is easier to learn than other higher level languages such as C++. However, there is no reason why the procedures outlined in some detail in this book could not be implemented in another language.

The next chapter deals with the way in which we can describe the geometrical boundary of a problem and the boundary conditions in a numerical way. This is done by subdividing the surface into small elements and by interpolating between nodal values. This is essential for the later treatment of integral equations. With the aid of the examples

we cannot only test the subroutines developed, but also get an understanding of the error introduced by the approximations used to describe boundaries.

Another fundamental building block is the description of the material response. In Chapter 4 we introduce basic concepts of elasticity and potential flow and develop fundamental solutions, that is, simple solutions which satisfy the governing differential equations. These will be central to our subsequent deliberations.

We introduce the concepts of boundary element methods next, using the method originally proposed by Trefftz. Although this very simple method cannot be used for general purpose programs, it serves very well to explain the fundamental ideas of the method. A small computer program can be developed to solve some simple problems. Again, this will serve as a tool for learning by experience.

The direct boundary element method used in the majority of BEM software is introduced next. Here we will use the reciprocal theorem by Betti, which is well known to engineers, to obtain an integral equation where the integrals contain singular functions, i.e. ones which have infinite value at certain points. The major task in the implementation is to solve the integral equations numerically.

The next chapter on numerical implementation therefore deals with the evaluation of integrals using numerical integration. Those familiar with isoparametric finite elements will recognise the Guass Quadrature method used. However, in contrast to its use in the FEM, one must be very careful to select the number of integration points, as they are dependent on how close the singularity is to the integration region. This is the most difficult and crucial part in the implementation of the BEM. The integration over the boundary surface is carried out over a boundary element and the contributions of all elements which describe a boundary are then added. We will see that this is very similar to the assembly procedure in the FEM.

After the numerical treatment of the integral equations, we end up with a system of equations. In contrast to the FEM, the coefficient matrix is fully populated and unsymmetrical. Standard Gauss elimination can be used but, for large systems, the storage requirement and the computation times may be reduced considerably by iterative solvers, such as conjugate gradient methods.

The results which we obtain after the solution are values of displacement or traction at the boundary depending on the boundary condition specified. In contrast to the FEM, values are not obtained immediately in the interior of the domain, but are computed by post-processing. In Chapter 8 it is explained how the stresses at the boundary and in the interior can be obtained from boundary displacements and tractions. This is an advantage of the method because the user has a free choice where results are obtained.

We now have all the building blocks together, and are able to compile a computer program which is able to solve two and three-dimensional problems in elasticity and potential flow, depending on which fundamental solution is used. In Chapter 9 we therefore apply the program developed to test examples to find out what level of accuracy can be obtained in comparison with the FEM.

For the solution of inhomogeneous domains, where we cannot obtain a fundamental solution, we introduce the concept of multiple regions, where the domain is subdivided into subregions, much in the same way as with the FEM. There is an additional advantage in this concept because sparseness is introduced in the system of equations. We will also find out in a later chapter that the multi-region method allows contact problems to be solved.

If we use the multi-region concept, we find that there is a possibility that the traction vector can be discontinuous if there is a corner in the boundary of a subregion. We find that this causes some difficulty, since we have more unknown than available equations to solve for them. Since in nearly all textbooks on the BEM this problem is not discussed in any detail, we devote the whole of Chapter 11 to the treatment of corners and edges.

Because there are no elements in the interior of the domain, as is the case with the FEM, the BEM has difficulty dealing with problems where 'body forces' have to be applied. In the case of gravity or centrifugal forces, which are constant, the body forces can be treated by a surface integral. If the body forces are not constant throughout the domain, a volume integral will have to be evaluated. This can be done by supplying internal cells which look like finite elements, but do not involve any additional degrees of freedom, as they are only used for integration. The implementation of this procedure, discussed in Chapter 12, however, allows the solution of problems in elastoplasticity and viscoplasticity.

In Chapter 13 we show that the solution of non-linear problems follows the same procedures as in the FEM, so the general solution algorithm is the same. Here we discuss two types of non-linear problems in more detail: plasticity and contact problems. We find that the analysis of problems in plasticity involves the consideration of residual stresses as 'body forces'. We also find that the method is well suited to model contact and crack propagation problems.

It is possible to couple the BEM with the FEM thus getting the 'best of both worlds'. In Chapter 14, methods of coupling are presented. Basically, a stiffness matrix of the BE region is obtained and assembled with the FEM stiffness matrices. Since many general purpose programs allow the input of a user defined element stiffness matrix, this may be used to extend the capabilities of a reader's FEM code.

To demonstrate that the method also works for large scale industrial problems, Chapter 15 shows some applications of the boundary and coupled methods in engineering. The purpose of this chapter is twofold: firstly, it shows how complex problems, as they invariably occur in real life, can be simplified and how a boundary element mesh can be obtained. Secondly, it shows the range of problems that can be solved.

By the end of this book, the reader should have an understanding of how the method works and how it can be implemented into a computer program.

1.3 MATHEMATICAL PRELIMINARIES

A good consistent notation is essential to any textbook. For developing and explaining numerical methods two notations are used by engineers: matrix and tensor notation. Traditionally, textbooks on the BEM have used tensors, whereas those about the FEM have used matrix notation, although many engineers working in solid mechanics are using tensors nowadays because of the ease with which non-linear problems can be treated. For this book we have chosen to use matrix/vector notation.

There are two reasons for this: firstly, the book which is probably still the most widely read on numerical modelling, *The Finite Element Method*, by O. C. Zienkiewicz and R.L. Taylor, now in its fifth edition, uses matrix notation throughout. Since we hope to attract more engineers to the BEM, this was one motivation. The other reason is that even

classical books on the BEM that use tensor notation have to revert to matrix at some stage, for example when discussing the assembly of the system of equations.

As will be shown for the (mostly) linear problems discussed here, there is not so much difference between the two notations. Indeed, the only time one notices a significant difference is when deriving fundamental solutions, which are much simpler using tensor indicial notation. The reader will see that some compromise solution has been reached in this book for this case.

In the following we discuss some basic mathematics which will be used in this book and also attempt a comparison of matrix and tensor notation. A good introduction into tensor algebra can be found in the book by Holzapfel[2].

1.3.1 Vector algebra

Vectors are used to represent a vectorial quantity such as displacement or force. They can also be used to define the position of a point relative to a set of Cartesian axes. By contrast, a column vector is simply a matrix with one column.

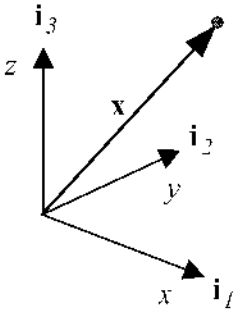

Figure 1.3 Position vector **x** defining a point in space

We define the position of a point in 3-D space with respect to Cartesian axes x, y, z (Figure 1.3) as

$$\mathbf{x} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \tag{1.1}$$

Alternatively, we may represent the point in terms of Cartesian coordinates x_i, where $i = 1,2,3$ (this is referred to as *range*).

The Cartesian coordinates are specified with respect to a set of orthogonal coordinate axes, which are defined by base vectors of unit length, $\mathbf{i}_i$ and which have the property

$$\mathbf{i}_i \bullet \mathbf{i}_j = \delta_{ij} \quad \begin{Bmatrix} 1 \text{ for } i = j \\ 0 \text{ for } i \neq j \end{Bmatrix} \tag{1.2}$$

where $(\ )\bullet(\ )$ denotes the scalar (dot) product, as explained on page 17 and δ_{ij} is known as the Kronekker delta.

Vector $\mathbf{x}$ may then be represented in indicial notation as

$$\mathbf{x} = \sum_{i=1}^{3} x_i \mathbf{i}_i = x_i \mathbf{i}_i \tag{1.3}$$

where the summation convention has been used for the last expression. This convention specifies that for any index which is repeated, a summation of all terms within the *range* is implied.

Another vector quantity is the displacement which can be written either as

$$\mathbf{u} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix} \quad \text{or} \quad \mathbf{u} = u_i \mathbf{i}_i \tag{1.4}$$

in indicial notation.

Coordinate transformation

If we want to express the location of a point in another orthogonal system with the directions $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ then we have

$$\mathbf{x}' = \mathbf{T}_g \mathbf{x} \tag{1.5}$$

where the transformation matrix is defined as

$$\mathbf{T}_g = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3] \tag{1.6}$$

Alternatively, we may write in indicial notation

$$x_i' = \Lambda_{ij} x_j \tag{1.7}$$

where

$$\Lambda_{ij} = \mathbf{v}_i \bullet \mathbf{i}_j \tag{1.8}$$

Projection of one vector onto another

If we want to compute the projection of a vector onto a direction specified by a unit vector $\mathbf{v}$, then it is very convenient to use the dot product. For example, the component, u' of the displacement $\mathbf{u}$ in the direction specified by $\mathbf{v}$ is given by (Figure 1.4)

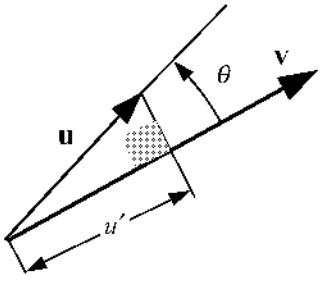

Figure 1.4 Projection of vector

$$u' = \mathbf{u} \bullet \mathbf{v} \tag{1.9}$$

The angle θ between the two vectors is computed by

$$cos\theta = \frac{1}{|\mathbf{u}|}\mathbf{u} \bullet \mathbf{v} \tag{1.10}$$

where the length of vector **u** is given by

$$|\mathbf{u}| = \sqrt{u_x^2 + u_y^2 + u_z^2} \tag{1.11}$$

Derivatives of vectors

The derivatives of the displacement vector may be written as

$$\frac{\partial \mathbf{u}}{\partial x} = \begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial x} \\ \frac{\partial u_z}{\partial x} \end{Bmatrix}; \quad \frac{\partial \mathbf{u}}{\partial y} = \begin{Bmatrix} \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_z}{\partial y} \end{Bmatrix}; \quad \frac{\partial \mathbf{u}}{\partial z} = \begin{Bmatrix} \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial z} \\ \frac{\partial u_z}{\partial z} \end{Bmatrix} \tag{1.12}$$

In indicial notation we simply write

$$\frac{\partial u_i}{\partial x_j} = u_{i,j} \tag{1.13}$$

1.3.2 Stress and strain

Stresses and strains are tensorial quantities. In the indicial notation the strain tensor is defined by

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{1.14}$$

In this book, however, we use a notation originally proposed by Timoshenko[3]. We define a *pseudo-vector* of strain, i.e. a matrix with one column:

$$\boldsymbol{\varepsilon} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial u_x}{\partial x} \\ \dfrac{\partial u_y}{\partial y} \\ \dfrac{\partial u_z}{\partial z} \\ \dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x} \\ \dfrac{\partial u_y}{\partial z} + \dfrac{\partial u_z}{\partial y} \\ \dfrac{\partial u_x}{\partial z} + \dfrac{\partial u_z}{\partial x} \end{Bmatrix} \tag{1.15}$$

Note that in the *pseudo-vector* notation we only have six strain components, whereas the symmetric strain tensor has nine. Also note that the ½ term is missing for the shear strains in order to achieve consistency between the tensor and matrix operations. The index number of the location of the strain or stress components for matrix notation and tensor notation is given in Table 1.1.

Table 1.1 Index numbering for strain and stress

Notation	Index number					
Matrix	1	2	3	4	5	6
Tensor	11 xx	22 yy	33 zz	12 and 21 xy and yx	23 and 32 yz and zy	31&13 zx and xz

Similarly, the stress tensor σ_{ij} can be written as a *pseudo-vector*

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{xz} \end{bmatrix}^T \tag{1.16}$$

1.4 CONCLUSIONS

At the beginning of this chapter we have shown on a geomechanics example that substantial gains can be made with the BEM, in terms of mesh generation and solution times. These gains are most pronounced for problems which involve infinite or semi-infinite domains. Other examples where the BEM seems to be superior to the FEM is for problems where boundary stresses are important, for example, in mechanical engineering. Examples of this will be shown later.

The purpose of this book is to encourage the use of the method. The simple computer programs included here contain all the necessary building blocks necessary for building more advanced and more specific computer programs for research or industrial applications. For example, by simply substituting different fundamental solutions one can solve problems other than elasticity and potential flow.

The reader will note that although the size of this book is similar to *Programming the Finite Element Method* [4], the range of applications that have been discussed here is less. For example, here we do not mention transient and dynamic problems. The reason for this is twofold: firstly, the subject matter is more complex and an attempt has been made to gently lead the reader into it. The integration over elements to obtain the stiffness matrix in the FEM can be explained, for example, in a few pages, whereas due to the singularity of the fundamental solutions a whole chapter has to be dedicated to integration in the BEM. Secondly, some problems simply do not arise at all in the FEM. The treatment of edges and corners causes no problems in the FEM because boundary tractions are not referred to at all.

In conclusion, the reader should see this book as an introduction to the BEM, containing some basic building blocks for computer programming. Armed with this knowledge, the readers interested in more advanced topics may find it easier to understand more advanced text books and publications on this subject.

1.5 REFERENCES

1. Zienkiewicz, O.C. and Taylor, R.L. (2000) *The Finite Element Method,* (5th ed.). Butterworth-Heinemann, Oxford.
2. Holzapfel, G.A. (2000) *Non-linear Solid Mechanics.* Wiley, Chichester.
3. Timoshenko, S.P. and Goodier, J.N. (1970) *Theory of Elasticity.* McGraw-Hill, London.
4. Smith, I.M. and Griffiths, D.V. (1998) *Programming the Finite Element Method,* (3rd ed.). Wiley, Chichester.