Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Rúben Miguel
Borges Lourenço

Desenvolvimento de *software* modular para análise pelo Método dos Elementos Finitos em incompressibilidade

Development of a modular software based on the Finite Element Method for incompressible problems

Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Rúben Miguel Borges Lourenço

Desenvolvimento de *software* modular para análise pelo Método dos Elementos Finitos em incompressibilidade

Development of a modular software based on the Finite Element Method for incompressible problems

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Robertt Angelo Fontes Valente, Professor Associado do Departamento de Engenharia Mecânica e de Joaquim Alexandre Mendes de Pinho da Cruz, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Este trabalho de investigação teve o apoio do Departamento de Engenharia Mecânica da Universidade de Aveiro e do Centro de Tecnologia Mecânica e Automação (TEMA), ao abrigo dos projectos 00481/2013-FCT e CENTRO-01-0145-FEDER-022083.

o júri / the jury

presidente / president

Prof. Doutor Ricardo José Alves de Sousa
Professor Auxiliar com Agregação, Universidade de Aveiro

Doutor Renato Manuel Natal Jorge
Professor Associado com Agregação, Faculdade de Engenharia, Universidade do Porto

Prof. Doutor Robertt Angelo Fontes Valente
Professor Associado, Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Ao Professor Doutor Robertt Valente pelo rigor e organização na orientação deste trabalho, pelo conhecimento transmitido e disponibilidade permanente. Um agradecimento especial pela amizade, colaboração e apoio demonstrados ao longo deste percurso.

Ao Professor Doutor João Oliveira pela sugestão do tema deste trabalho, pelo conhecimento transmitido e em especial pela grande amizade e estreita colaboração ao longo destes quatro anos de trabalho.

Ao Professor Doutor Joaquim Cruz, pelos preciosos ensinamentos nesta área, pelo entusiasmo contagiante e disponibilidade demonstrada.

Ao Tiago Ávila e ao Luís Ávila, pela grande amizade e apoio prestados.

Ao Eng.º José Bastos pelas grandes jantaradas e noitadas de trabalho e estudo. E em especial pela grande amizade e apoio prestado.

Aos meus pais e avós, pela preocupação e apoio incansáveis. Um eterno agradecimento pelo sacrifício demonstrado e por possibilitarem esta oportunidade.

keywords

Finite Element Method, Incompressibility, Locking, Enhanced strain, Selective integration, MATLAB, Software development, Numerical simulation

abstract

The importance of numerical simulation in the engineering field makes relevant the development of computational tools, based on the Finite Element Method, to solve structural problems. However, the application of the classical Finite Element approach to incompressible (or near-incompressible) situations has been a source of numerical problems, such as volumetric locking.

This Dissertation describes several formulations used in the analysis of incompressible problems, with focus on the four-node bi-linear quadrilateral element. Over the last decades special attention has been given to improve the performance of this element under incompressibility due to its computational effectiveness, making it attractive to be used in complex problems. The devised solutions are often targeted at treating the effects of volumetric locking, although some of them were originally targeted at the treatment of shear locking. In this context, the studied formulations were the selective integration approach, the B-bar method, the mixed (u/p) method and the enhanced strain method, which includes compatible and incompatible mode elements.

In order to computationally implement these finite element formulations, a revision of the fundamental concepts and basic formulas of the classical method, and for each alternative formulation, is carried out. In the context of incompressibility, the underlying concept is presented and the problem of locking is described.

The development of an in-house software, also in the context of this Dissertation, targeted at solving incompressible problems is discussed. Some important sub-routines and programming approaches are referred to with code examples.

Finally, the quality of implementation and efficiency of the finite element formulations is evaluated by analyzing a series of benchmark tests, using the developed in-house software and comparing the results against the ones coming from a commercial finite element software.

palavras-chave

Método dos Elementos Finitos, Incompressibilidade, Retenção, Deformações acrescentadas, Integração selectiva, MATLAB, Desenvolvimento de *software*, Simulação numérica

resumo

A importância da simulação numérica no campo da engenharia torna relevante o desenvolvimento de ferramentas computacionais, baseadas no Método dos Elementos Finitos, destinadas à resolução de problemas estruturais. Contudo, a aplicação do Método dos Elementos Finitos a problemas incompressíveis ou (*quasi*-incompressíveis) apresenta tipicamente uma série de problemas, nomeadamente de retenção numérica.

Nesta dissertação são descritas várias formulações destinadas à análise de problemas incompressívies com enfoque no elemento finito quadrilátero (bilinear) de quatro nós. Nas últimas décadas, este elemento tem vindo a receber especial atenção no sentido de melhorar a sua performance em problemas de incompressibilidade, devido à sua elevada eficiência em termos computacionais, o que o torna atractivo para ser utilizado em problemas mais complexos. Geralmente, as soluções desenvolvidas passam por estratégias de minimização dos efeitos de retenção volumétrica, embora algumas dessas formulações se destinassem orginalmente a resolver problemas de retenção associados às componentes de deformação de corte. Neste contexto, as formulações estudadas foram a integração selectiva, o método B-*bar*, o método misto deslocamento/pressão e o método das deformações acrescentadas, o qual inclui formulações de modos compatíveis e incompatíveis.

Tendo em vista a implementação computacional destas formulações de elementos finitos, efectua-se uma revisão dos conceitos básicos fundamentais do método clássico e das formulações alternativas. No contexto da incompressibilidade, apresenta-se o conceito básico e uma breve referência ao problema da retenção.

No contexto desta dissertação foi desenvolvido um software destinado a resolver problemas de incompressibilidade. Algumas sub-rotinas são apresentadas fazendo referência a algumas abordagens de programação com exemplificação de código.

Finalmente, avalia-se a qualidade da implementação e a eficiência das formulações implementadas através da análise de resultados de diversos *benchmarks*, usando o *software* desenvolvido e comparando os resultados obtidos com aqueles provenientes de um *software* comercial.

Contents

List of Tables

Intentionally blank page.

List of Figures

Chapter 1

Introduction

1.1 Overview and motivation

Engineering problems often involve complicated domains, loads and nonlinearities that hinder the use of analytical solutions. Numerical methods provide an alternative means of finding solutions in these cases. Over the last decades, the use of computers and numerical methods made it possible to solve mechanical problems in many different fields. Numerical simulation is cost-effective and saves time and resources when compared to physical experiments. Additionally, it can be used to predict defects and refine production parameters [27, 35, 36].

The Finite Element Method (FEM) is a powerful numerical method that has been playing an essential role in the fields of engineering design and manufacturing, being extensively used for the study of solids, structures, heat transfer, fluids, electricity and magnetism [3, 27, 36]. Graduate engineers will most likely encounter advanced commercial finite element analysis (FEA) software in their future offices. These programs are capable of simulating complex problems involving nonlinearities, whether material or geometrical, contact conditions, among others, besides offering advanced pre- and post-processing abilities [17]. When using these programs, it is extremely important for the user to have a deep understanding of the physical and mechanical processes involved and to be aware of the specific characteristics of FEM when applyed to a particular model. Moreover, by studying a virtual model of the real physical system approximation errors can compromise the results. Therefore, the user has to cautiously evaluate the limitations of each used model [36].

The motivation for this work came from the fact that during undergraduate studies the training given to students on FEA is often accompanied by the use of commercial software. The three most common examples used in our department are Abaqus, FEMAP and ANSYS, whose general capabilities are far superior to those seen in classes. Although the complexities and capabilities of these programs are unquestionable, the learning process is tipically focused on the pre- and post-processing parts and overlooks the core part in-between, the FEA itself. Proficiency in FEA certainly only comes after several years of training and experience. However, during undergraduate studies a better way for students to gain a deeper understanding about FEA is to study and implement the formulations by themselves, by means of developing their own finite element codes, which could be used as complementary tools for the teaching of FEA, alongside the commercial software already in use.

1.2 Objectives

The primary objective of this Dissertation is to develop a FEM program for pedagogical and research purposes that could be used as a base platform to test and implement different finite element formulations. To do so, the idea was to create a modular and open-source software that could serve as an auxiliary tool for engineering students learning FEA or for researchers in order to study, implement and validate more advanced finite element formulations.

Beyond the implementation and development of a software product, the scientific part of this Dissertation is dedicated to the implementation of several finite element formulations, based on the two-dimensional bi-linear quadrilateral element. The aim was to study and learn about different approaches available to deal with problems arising from the analysis of incompressible media, such as selective reduced integration, B-bar method, compatible and incompatible modes and mixed methods.

1.3 Methodology

The implementation of all the finite element formulations shown in this work was performed through the development of a in-house finite element code, from scratch. The chosen programming language was MATLAB, a modern high-level programming language specially designed for dealing with matrices and arrays. The vectorization capabilities make it particularly suitable for programming the FEM. The high-level nature of this language makes it easier to read and write, allowing the user to focus on the implementation rather than on the programming aspects. The integrated devolopment environment allows the user to type and execute commands at a command prompt, making program debugging easier [10, 14, 17]. MATLAB also offers sophisticated libraries for matrix operations, symbolic calculus, general numeric methods and data plotting. These reasons made this programming platform an excellent tool for the development of this work.

The decision of focusing on the finite element formulations did not allow for the implementation of a mesh generation algorithm. Therefore, it was decided that this step would be performed using Abaqus student version, a commercial general purpose finite element software available at the department. Abaqus allows the user to store the model parameters and export them to an input file. Given this possibility, the work started with the development of a module which would allow the program to import and read the contents of the input file related with mesh definition. Further attention was given to the development of simple user-interface elements to facilitate the introduction of other input data and definition of the boundary conditions.

Code development then proceeded with the implementation of sub-routines for analyses of plane stress/strain states, using the traditional formulation of the four-node bi-linear quadrilateral element. The necessary debug tests and model validations were conducted by comparing the program results with those obtained using Abaqus. Upon verifying the results agreeing between both softwares, development proceeded with the implementation of the necessary routines for incompressibility analysis. Further debug testing was performed to ensure the program was working correctly.

Finally, several finite element benchmarking problems were performed. The main purpose was to assess the quality and behavior of the implemented element formula-

tions under different circumstances. For comparison and validation purposes, the same benchmarks were also performed using different elements available in Abaqus software.

1.4 Reading guide

This Dissertation report is composed of six chapters.

In Chapter 2, the general concepts behind the FEM are presented and a review of the fundamental concepts is conducted regarding classical FEM formulations, applied to linear elasticity analyses. Focus is posed on the isoparametric four-node quadrilateral finite element formulation.

Chapter 3 presents a review on the problems behind the numerical analysis of incompressibility. The fundamental formulation of the incompressible problem is presented and a brief reference to the problem of locking, in this context, is done. A detailed literature review of several finite element formulations, aimed at the analysis of incompressible problems, is carried out.

In Chapter 4 computational implementation aspects are discussed, aimed at the development of a MATLAB-based, in-house finite element code. The scope of application and program architecture are reviewed with important software features and sub-routines being analyzed in detail, with the help of code examples.

In Chapter 5 a series of benchmark problems, available in the literature, are used in order to validate the developed code, with the results for the different implemented formulations being compared to those obtained with a commercial software.

In Chapter 6 a general overview of the Dissertation work and the obtained results is conducted. Some concluding remarks are given regarding future developments.

Intentionally blank page.

Chapter 2

The Finite Element Method

2.1 General concept

Mechanical problems are modelled using partial differential equations (PDEs) for the conservation of mass and momentum and additional material laws. These problems usually do not have analytical solution, requiring the use of approximate methods. One such method is the Finite Element Method (FEM), which consists of a piecewise application of a variational method. The term "variational" refers to the so called weak formulation, where a differential equation is rewritten as an equivalent integral form [27, 35].

The underlying concept of the FEM is to model a generic problem involving continuous media by means of analyzing discrete parts of it (called finite elements), interconnected by a set of points (called nodes), as depicted in the example from Fig. 2.1. An element is defined by nodes located at the border or inside the element itself. For two-dimensional analysis, quadrilateral or triangular elements are normally used [27, 36].

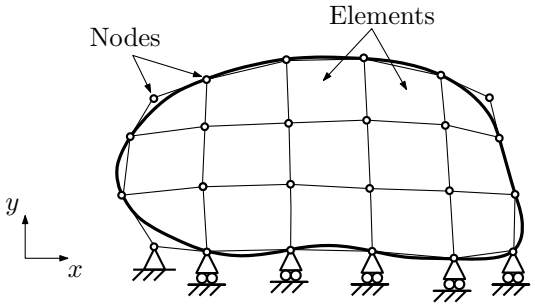


Figure 2.1: Spatial discretization of a given domain, using finite elements.

The calculation of a certain variable inside an element is performed by interpolating the corresponding nodal values. This is accomplished by approximation functions (known as shape functions), based on the idea that any continuous function can be represented by a linear combination of, for instance, Lagrangian polynomials. The order of the interpolation functions depends on the number of nodes in the element. The variational formulation describing the problem is applied to each individual element. At the end of the process, the effects of each element are properly combined in order for the discretization to represent the continuous medium as a whole [22, 27, 36].

2.2 Linear elasticity

Despite the nonlinear nature of continuum mechanics' laws governing mechanical problems, linearization is possible under the assumption of small deformations. The most common material law is the Hooke's law that states a linear dependence between stress $\boldsymbol{\sigma}$ and strain $\boldsymbol{\varepsilon}$ fields, using a variable as constant of proportionality, dependent on the material. For a one-dimensional relationship, it holds that [35, 36]:

$$\sigma = E\,\varepsilon\,, \tag{2.1}$$

where (E) stands for the elasticity modulus (Young's modulus) of the material.

Under linear elasticity, it is common to employ a pure displacement formulation where the stress tensor is eliminated, as it can be calculated using the displacement [3, 35]. Therefore, considering a three-dimensional body with an arbitrary geometry subjected to surface and body forces along its surface (Γ) and volume (Ω) respectively (Fig. 2.2), for the given set of boundary conditions and material laws the first variable of interest becomes the displacement field $\mathbf{u}(x, y, z)$ [23, 36].

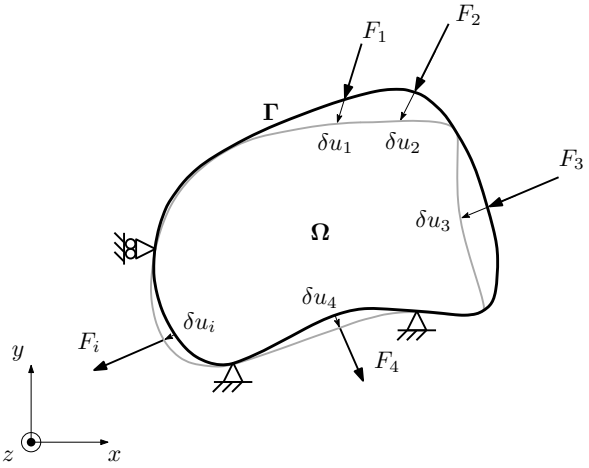


Figure 2.2: Solid elastic body of volume Ω and surface Γ, subjected to body and surface loads under static equilibrium.

The linear stress state of the three-dimensional body is based on Hooke's law from Eq. (2.1), but now rewritten using tensor notation in the form:

$$\boldsymbol{\sigma} = \mathbf{D} : \boldsymbol{\varepsilon}\,, \tag{2.2}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are the stress and strain tensors respectively, and $\mathbf{D}$ is the elasticity tensor for an isotropic material, related to the elastic Lamé constants (λ) and (μ) as:

$$\mathbf{D} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}, \tag{2.3}$$

which in turn can be written in terms of the Young's modulus (E) and the Poisson's ratio (ν), in the form [17, 23]:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{2.4}$$

Recallling that the strains are obtained by differentiating the displacements, it is possible to define a matrix differentiation operator $\nabla_{\mathbf{s}}$ to be applied to the displacement field vector $\mathbf{u}$, such that [23, 36]:

$$\boldsymbol{\varepsilon} = \nabla_{\mathbf{s}}\mathbf{u}\,, \tag{2.5}$$

thus, obtaining the linear strain field $\boldsymbol{\varepsilon}$ for the body.

2.2.1 Finite element approximation

Under the assumption of linear elasticity, the weak formulation is equivalent to find such displacement field that minimizes a quadratic functional, called the total potential energy (Π) defined as:

$$\Pi = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}\,\mathrm{d}\Omega - \int_{\Omega}\mathbf{u}^{\mathrm{T}}\mathbf{b}\,\mathrm{d}\Omega - \int_{\Gamma}\mathbf{u}^{\mathrm{T}}\,\bar{\mathbf{t}}\,\mathrm{d}\Gamma\,, \tag{2.6}$$

where $\mathbf{b}$ and $\bar{\mathbf{t}}$ are the external loads distributed along the domain (Ω) and the surface (Γ), respectively. Prescribed displacements are specified on the parts of the body surface which are not subjected to surface loads [23, 27, 36].

The weak formulation describing the problem is applied to each individual element. At the element level, the real displacement field is approximated by means of the the nodal displacements $\mathbf{d}^e$ and a set of shape functions, organized into matrix $\mathbf{N}^e$ [22, 36]:

$$\mathbf{u}^e = \mathbf{N}^e\mathbf{d}^e\,, \tag{2.7}$$

$$\mathbf{d}^e = \begin{Bmatrix} u_i & v_i & w_i & ... \end{Bmatrix}^{\mathrm{T}}, \quad i = 1, ..., n_{\text{nodes}}\,, \tag{2.8}$$

$$\mathbf{N}^e = \left[\begin{array}{c|ccc|c} \cdots & N_i(x,y,z) & 0 & 0 & \cdots \\ & 0 & N_i(x,y,z) & 0 & \\ & 0 & 0 & N_i(x,y,z) & \end{array}\right], \quad i = 1, ..., n_{\text{nodes}}\,. \tag{2.9}$$

The strain field described on Eq. (2.5) is now rewritten in the equivalent form [36]:

$$\boldsymbol{\varepsilon}^e = \nabla_{\mathbf{s}}\mathbf{u}^e = \nabla_{\mathbf{s}}(\mathbf{N}^e\mathbf{d}^e) = \mathbf{B}^e\mathbf{d}^e\,, \tag{2.10}$$

where $\mathbf{B}^e$ is the element strain-displacement matrix, given by [36]:

$$\mathbf{B}^e = \begin{bmatrix} \mathbf{B}_1^e & \mathbf{B}_2^e & \cdots & \mathbf{B}_{n_{\text{nodes}}}^e \end{bmatrix}, \tag{2.11}$$

with each component $\mathbf{B}_i^e$ defined as:

$$\mathbf{B}_i^e = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \end{bmatrix}, \quad i = 1, ..., n_{\text{nodes}} . \tag{2.12}$$

The principle of virtual work states that if a system is under static equilibrium, the work done by internal loads should be equal to that resulting from the external loads. In this case, the displacement and strain fields are calculated by means of considering a virtual displacement field $\delta\mathbf{d}^e$, such that its components are small enough for the whole set of loads to remain unchanged, thus obtaining [3, 36]:

$$\begin{cases} \delta\mathbf{u}^e = \mathbf{N}^e\delta\mathbf{d}^e \\ \delta\boldsymbol{\varepsilon}^e = \mathbf{B}^e\delta\mathbf{d}^e \end{cases} . \tag{2.13}$$

Therefore, the total work done by internal and external loads distributed along the element domain (Ω^e) and the surface (Γ^e), respectively, is given by [23, 36]:

$$W_{\text{int}}^e = (\delta\mathbf{d}^e)^{\mathrm{T}} \int_{\Omega^e} (\mathbf{B}^e)^T \boldsymbol{\sigma}^e \, \mathrm{d}\Omega^e , \tag{2.14}$$

$$W_{\text{ext}\Omega}^e = (\delta\mathbf{d}^e)^{\mathrm{T}} \int_{\Omega^e} (\mathbf{N}^e)^{\mathrm{T}} \mathbf{b}^e \, \mathrm{d}\Omega^e , \tag{2.15}$$

$$W_{\text{ext}\Gamma}^e = (\delta\mathbf{d}^e)^{\mathrm{T}} \int_{\Gamma^e} (\mathbf{N}_\Gamma^e) \bar{\mathbf{t}}^e \, \mathrm{d}\Gamma^e . \tag{2.16}$$

With virtual displacements being very small quantities, the total potential energy will experiment a small variation thus, substituting the above into Eq. (2.6), one can obtain [23, 36]:

$$\delta\Pi^e = (\delta\mathbf{d}^e)^{\mathrm{T}} \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \boldsymbol{\sigma}^e \, \mathrm{d}\Omega^e - (\delta\mathbf{d}^e)^{\mathrm{T}} \left[\int_{\Omega^e} (\mathbf{N}^e)^{\mathrm{T}} \mathbf{b}^e \, \mathrm{d}\Omega^e + \int_{\Gamma^e} (\mathbf{N}_\Gamma^e) \bar{\mathbf{t}}^e \, \mathrm{d}\Gamma^e \right] . \tag{2.17}$$

Given that the nodal displacements corresponding to the minimum of the total potential energy are determined by employing the condition: $\frac{\delta\Pi^e}{\delta\mathbf{d}^e} = 0$, the equilibrium for each element is described by the relation [3, 23, 36]:

$$\mathbf{k}^e\mathbf{d}^e = \mathbf{f}^e , \tag{2.18}$$

where $\mathbf{k}^e$ is the element stiffness matrix and $\mathbf{f}^e$ the nodal load vector, defined as:

$$\mathbf{k}^e = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}^e \mathbf{B}^e \, \mathrm{d}\Omega^e , \tag{2.19}$$

$$\mathbf{f}^e = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{b}^e \, \mathrm{d}\Omega^e + \int_{\Gamma^e} (\mathbf{N}^e)^{\mathrm{T}} \, \bar{\mathbf{t}}^e \, \mathrm{d}\Gamma^e . \tag{2.20}$$

Finally, the element stiffness matrices are assembled into a global stiffness matrix $\mathbf{K}$ and the nodal load vectors into a global load vector $\mathbf{f}$, such that [3, 36]:

$$\mathbf{K} = \bigwedge_{e=1}^{n_e} \mathbf{k}^e \,, \tag{2.21}$$

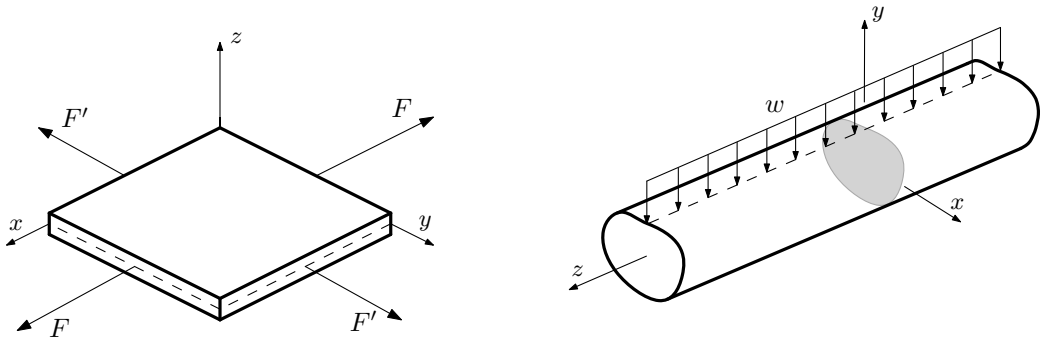
$$\mathbf{f} = \bigwedge_{e=1}^{n_e} \mathbf{f}^e \,, \tag{2.22}$$

resulting in a global system of equations of the form: $\mathbf{Kd} = \mathbf{f}$.

2.2.2 Plane stress and plane strain

The structural analysis of a three-dimensional body can be carried out in the two-dimensional space if the body is under a state of plane stress or plane strain [17, 36].

A solid with one dimension relatively small compared to the others, and loaded in its plane, can be analyzed using a plane stress approach, common examples can be seen in Fig. 2.3 (a). The surfaces ($z = \pm t/2$) are not under load, therefore one can reasonably assume that the stress components along the z-direction are equal to zero, while the other stress components remain constant. Nonetheless, it is important to note that under plane stress: $\varepsilon_{zz} \neq 0$ [17, 36, 38].



(a) (b)

Figure 2.3: Representation of states of (a) plane stress and (b) plane strain.

Consequently, the stress field only has three cartesian coordinates and is characterized by the relation [17, 36]:

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{Bmatrix} = \mathbf{D} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} , \tag{2.23}$$

with the elasticity matrix $\mathbf{D}$ being written in terms of the Young's modulus (E) and the Poisson's ratio (ν) as:

$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} . \tag{2.24}$$

States of plane strain usually refer to structures with one dimension significantly bigger than the others, such as the example from Fig. 2.3 (b). In this case, the strain components along the z-direction are equal to zero, however (σ_{zz}) is not negligible. As such, the strain field is formed by the three cartesian components obtained by inverting the relation from Eq. (2.23) and writing the elasticity matrix $\mathbf{D}$ as [17, 38]:

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}. \tag{2.25}$$

2.2.3 Isoparametric four-node quadrilateral element

Consider an analysis carried out in the two-dimensional space using quadrangular finite elements. The simplest available element is the Lagrangian bi-linear four-node quadrilateral, referred to as the Q4 element throughout this document. The Q4 element employs an isoparametric formulation which states that the nodal displacements are interpolated in the same way as the geometry. The concept is commonly used in the FEM implementation as it provides an efficient and systematic way to obtain higher-order elements, useful for the accurate representation of irregular domains [3, 27].

The domain of the Q4 element is defined by the coordinates (x_i, y_i) of its four corner nodes. It is assumed that the nodes are numbered in ascending order on the counter-clockwise direction as shown in Fig. 2.4 (a). Respecting an isoparametric approach, the coordinates (x, y) at any given point of the element are approximated as follows [3, 15, 27]:

$$\begin{cases} x = \sum_{i=1}^{n_{\text{nodes}}} N_i(\xi,\eta) x_i \\ y = \sum_{i=1}^{n_{\text{nodes}}} N_i(\xi,\eta) y_i \end{cases}, \tag{2.26}$$

where $N_i(\xi, \eta)$ are the element interpolation functions, defined in the natural coordinate system ($O\xi\eta$). In the global (Oxy) plane, the element has degrees of freedom (u) and (v), thus the displacement field can be obtained by interpolating the nodal displacements (u_i) and (v_i), such that [3, 15]:

$$\begin{cases} u(x,y) = \sum_{i=1}^{n_{\text{nodes}}} N_i(\xi,\eta) u_i \\ v(x,y) = \sum_{i=1}^{n_{\text{nodes}}} N_i(\xi,\eta) v_i \end{cases}. \tag{2.27}$$

The isoparametric shape functions for the Q4 element are based on the corresponding Lagrange polynomials, defined in the natural coordinate system ($O\xi\eta$) and written for

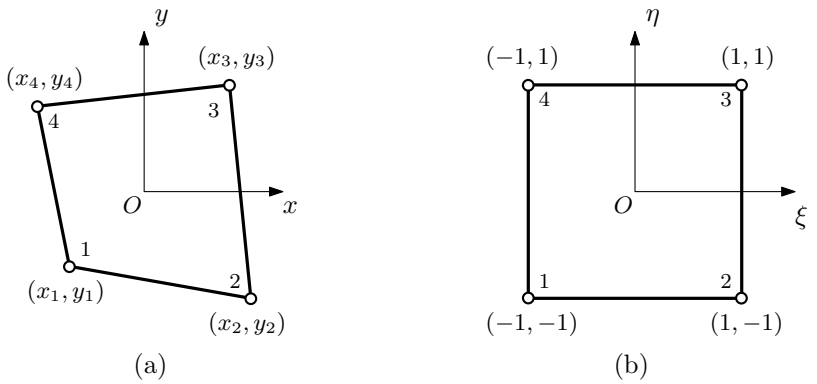


Figure 2.4: Configuration of the Q4 element in the (a) global Oxy coordinate system and (b) in the natural $O\xi\eta$ coordinate system.

each node as [3, 12, 15]:

$$\begin{cases} N_1 = \dfrac{1}{4}(1-\xi)(1-\eta) \\ N_2 = \dfrac{1}{4}(1+\xi)(1-\eta) \\ N_3 = \dfrac{1}{4}(1+\xi)(1+\eta) \\ N_4 = \dfrac{1}{4}(1-\xi)(1+\eta) \end{cases} . \tag{2.28}$$

The strain field of the element is determined by adapting Eq. (2.10) to a two-dimensional plane stress/strain problem. The strain-displacement matrix $\mathbf{B}^e$ looses the terms associated with the third dimension, being now defined as [15, 36]:

$$\mathbf{B}^e = \begin{bmatrix} \cdots & \left|\begin{matrix} \dfrac{\partial N_i}{\partial x} & 0 \\ 0 & \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x} \end{matrix}\right| & \cdots \end{bmatrix}, \quad i = 1, ..., n_{\text{nodes}} . \tag{2.29}$$

The strain field of the real distorted element is described by the infinitesimal variations of the displacements in the (Oxy) plane. This makes it difficult to compute the element equations in terms of the global (x) and (y) coordinates. Such limitation can be overcome by mapping the real distorted element to a reference element defined in the natural coordinate system $(O\xi\eta)$ with: $\xi, \eta \in [-1, 1]$. The mapping is defined by the coordinate transformation [3, 27, 38]:

$$x = x(\xi, \eta) \qquad y = y(\xi, \eta), \tag{2.30}$$

from which, using the chain rule of differentiation, holds:

$$\begin{Bmatrix} \dfrac{\partial N_i(\xi,\eta)}{\partial \xi} \\ \dfrac{\partial N_i(\xi,\eta)}{\partial \eta} \end{Bmatrix} = \mathbf{J}^e \begin{Bmatrix} \dfrac{\partial N_i(\xi,\eta)}{\partial x} \\ \dfrac{\partial N_i(\xi,\eta)}{\partial y} \end{Bmatrix}, \tag{2.31}$$

where $\mathbf{J}^e$ is the jacobian matrix of the transformation, obtained as:

$$\mathbf{J}^e = \sum_{i=1}^{n_{\text{nodes}}} \begin{bmatrix} \dfrac{\partial N_i}{\partial \xi} x_i & \dfrac{\partial N_i}{\partial \xi} y_i \\ \dfrac{\partial N_i}{\partial \eta} x_i & \dfrac{\partial N_i}{\partial \eta} y_i \end{bmatrix}. \tag{2.32}$$

In order to transform from natural coordinates to global coordinates, Eq. (2.31) needs to be inverted requiring the inverse of $\mathbf{J}^e$ to exist. A necessary condition for the jacobian matrix to be invertible is that its determinant be nonzero at every point (ξ, η) [27, 36].

Under linear elasticity and assuming small deformations, the principle of minimum potential energy is applied. Thus, the stiffness matrix for an element under a plane stress/strain state in global coordinates is given by [12, 36]:

$$\mathbf{k}^e = \iint_{A^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}^e \mathbf{B}^e t^e \, \mathrm{d}x\mathrm{d}y \,, \tag{2.33}$$

or, in natural coordinates as:

$$\mathbf{k}^e = \int_{-1}^{+1} \int_{-1}^{+1} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}^e \mathbf{B}^e t^e \det \mathbf{J}^e \, \mathrm{d}\xi\mathrm{d}\eta \,, \tag{2.34}$$

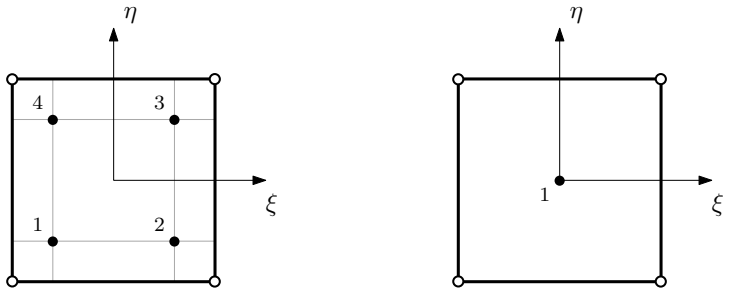
where (t^e) is the element thickness. The integration is numerically carried out using a recommended second order Gauss quadrature rule and replacing the integral by a double summation, such that [3, 27]:

$$\mathbf{k}^e = \sum_{r=1}^{n_r} \sum_{s=1}^{n_s} \left[(\mathbf{B}^e)^{\mathrm{T}}(\xi_r, \eta_s) \mathbf{D}^e \mathbf{B}^e(\xi_r, \eta_s) \det \mathbf{J}^e(\xi_r, \eta_s) t^e \right]_{r,s} w_r w_s \,, \tag{2.35}$$

where (n_r, n_s) are the number of integration points along each natural coordinate, (ξ_r, η_s) are the natural coordinates of the integration point and (w_r, w_s) are the corresponding weighting coefficients. The coordinates of the integration points are listed in Table 2.1, for first and second order quadrature rules, and their distribution inside the element are shown in Fig. 2.5. Regarding the four-node quadrilateral, if the integral is evaluated using a second order quadrature the solution will be exact. A one-point Gauss rule corresponds to a uniform reduced integration and will not yield in an exact solution for the integral.

Table 2.1: Natural coordinates and weights for Gauss rules of order 1×1 and 2×2.

Quadrature rule	Point	ξ_r	η_s	w_r	w_s
1×1	1	0	0	2	2
2×2	1	-0.57735	-0.57735	1	1
	2	$+0.57735$	-0.57735	1	1
	3	$+0.57735$	$+0.57735$	1	1
	4	-0.57735	$+0.57735$	1	1



(a) (b)

Figure 2.5: Natural coordinates of the integration points for Gauss rules of order (a) 2×2 and (b) 1×1.

Intentionally blank page.

Chapter 3

Incompressibility

3.1 Introduction

Many problems of physical relevance often involve displacement fields preserving the initial volume of the continuum being analyzed. Media that behave following this pattern are termed incompressible. Starting from the compressible isotropic case, these materials follow the equations for linear elasticity that can be described in terms of the displacements $\mathbf{u}$ and the stress tensor $\boldsymbol{\sigma}$, by the following constitutive relation [3, 4, 15]:

$$\boldsymbol{\sigma} = \underbrace{2\mu\,\boldsymbol{\varepsilon}}_{\text{deviatoric}} + \underbrace{\lambda(\operatorname{div}\mathbf{u})\mathbf{I}}_{\text{volumetric}}\,, \tag{3.1}$$

essentially dividing the stress tensor in its hydrostatic and deviatoric part, where $\mathbf{I}$ is the identity matrix and $(\operatorname{div}\mathbf{u})$ is the divergence operator applied to the displacement vector $\mathbf{u}$ with components (u, v, w), such that [7, 38]:

$$\operatorname{div}\mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} = \varepsilon_{kk}\,, \tag{3.2}$$

with (ε_{kk}) being the volumetric strain. If the given material is subjected to hydrostatic pressure, the relationship between the pressure (p) and volumetric strain (ε_{kk}) is linear, resulting in the definition of the bulk modulus (K) as follows [7]:

$$K = -\frac{p}{\varepsilon_{kk}} = \frac{E}{3(1-2\nu)}. \tag{3.3}$$

Mathematically, incompressibility is achieved by imposing a constraint enforcing the volumetric part of the strain field to be zero or extremely small, when compared to its deviatoric counterpart. Assuming only small deformations, the assumption that the volume of the body will remain constant is respected if the displacements normal to the body surface are considered to be zero, hence [1, 4, 15]:

$$\operatorname{div}\mathbf{u} = \varepsilon_{kk} = 0\,. \tag{3.4}$$

This condition causes the bulk modulus to grow towards infinity and the same effect will happen as the Poisson's ratio (ν) approaches 0.5. For elasticity problems the incompressibility conditon is ensured by the latter [15].

3.2 Volumetric locking

The displacement based FEM provides a robust approach to solve most problems, but encounters difficulties when incompressible materials are analyzed. Low-order elements may suffer from severe volumetric locking under incompressibility conditions, inasmuch as bi-linear elements are not able to ensure the nullity of the volumetric strain [1, 29].

According to the classical formulation of the FEM described in Section 2.2.1 the element stiffness matrix can be determined by the integral defined in Eq. (2.19) as follows:

$$\mathbf{k}^e = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}^e \mathbf{B}^e \, \mathrm{d}\Omega^e \, .$$

As the incompressibility limit is reached, the Lamé parameter (λ) increases towards infinity. As a result, some coefficients of the elasticity tensor $\mathbf{D}^{\mathrm{e}}$ become excessively high. This causes the element stiffness matrix $\mathbf{k}^e$ to be highly ill-conditioned, further contributing to the singularity of the global stiffness matrix. Naturally, if the coefficients of the global matrix are excessively high, the inverse will tend to zero when solving the global system of equations, inducing a near-zero displacement field and causing the stresses to be underestimated [22, 38]. The effect of locking can be explained as illustrated in Fig. 3.1, where a quadrangular domain is discretized using linear triangular elements. The structure is assumed to be under a plane strain state. In order for the elements deformation to be isocoric (thus respecting Eq. (3.2)) their area has to necessarily remain constant.

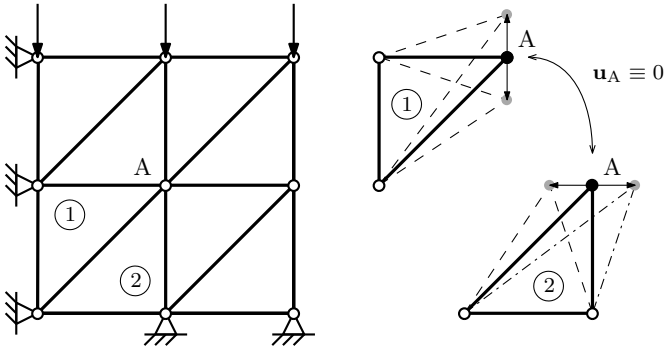


Figure 3.1: Locking of a mesh of linear triangles under incompressibility and possible displacements for elements 1 and 2.

Analyzing Fig. 3.1, it can be seen that element 1 respects this condition if node A only moves vertically. On the other hand, for element 2 the same node is only allowed to move horizontally. This conflict enforces the displacement at node A to be zero. The same conclusion may be drawn from analyzing the remaining elements of the mesh, resulting in a zero-displacement field for the structure. Since stresses are calculated from the displacements, the stress field will be underpredicted as well [15, 29, 38].

Low-order elements offer simpler implementation allowing for straightforward mesh operations and computational efficiency. For these reasons, it is of great importance to devise and implement strategies to avoid locking and improve the accuracy of the solutions provided by these elements.

3.3 Mixed (u/p) formulation

In the nearly-incompressible case a very small change in displacement can cause extremely large changes in pressure, rendering displacement-based solution too sensitive to be useful numerically. Therefore, it is desirable to devise a formulation of isotropic elasticity valid for both the compressible and incompressible cases. This purpose may be accomplished by the following constitutive relations [15, 34]:

$$\begin{cases} \boldsymbol{\sigma} = 2\mu\,\boldsymbol{\varepsilon} - p\,\mathbf{I} \\ \mathrm{div}\mathbf{u} + \dfrac{p}{\lambda} = 0 \end{cases}, \tag{3.5}$$

where the pressure (p) is treated as an independent variable. It can be seen that if ($\nu = 0,5$), the second equation becomes the incompressibility condition, with (p) being the hydrostatic pressure. On the other hand, if ($\nu < 0,5$), the pressure (p) may be eliminated from the relations to obtain the constitutive law from Eq. (3.1). The solution to element locking in the case of incompressibility is to break the strain field down to its deviatoric and volumetric parts, $\boldsymbol{\varepsilon}_{\mathrm{dev}}$ and $\boldsymbol{\varepsilon}_{\mathrm{vol}}$, such that [15, 29]

$$\begin{cases} \boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_{\mathrm{dev}} + \boldsymbol{\varepsilon}_{\mathrm{vol}} \\ \boldsymbol{\varepsilon}_{\mathrm{vol}} = \dfrac{1}{3}\varepsilon_{kk}\,\mathbf{I} \end{cases}. \tag{3.6}$$

The mixed displacement/pressure (u/p) approach determines the shape change from the deviatoric strains and the pressures from the volumetric strains, by employing the following system of equilibrium equations, at the element level [15, 29, 38]:

$$\begin{bmatrix} \mathbf{k}^e_{\mathrm{dd}} & \mathbf{k}^e_{\mathrm{dp}} \\ \mathbf{k}^e_{\mathrm{pd}} & \mathbf{k}^e_{\mathrm{pp}} \end{bmatrix} \begin{Bmatrix} \mathbf{d}^e \\ \mathbf{p}^e \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^e \\ \mathbf{0} \end{Bmatrix}, \tag{3.7}$$

with the corresponding stiffness matrices computed as:

$$\mathbf{k}^e_{\mathrm{dd}} = \int_{\Omega^e} (\mathbf{B}^e_{\mathrm{dev}})^{\mathrm{T}} \mathbf{D}^e_{\mu} \mathbf{B}^e_{\mathrm{dev}}\, \mathrm{d}\Omega^e\,, \tag{3.8}$$

$$\mathbf{k}^e_{\mathrm{dp}} = -\int_{\Omega^e} (\mathbf{B}^e_{\mathrm{vol}})^{\mathrm{T}} \mathbf{H}^e_{\mathrm{p}}\, \mathrm{d}\Omega^e\,, \tag{3.9}$$

$$\mathbf{k}^e_{\mathrm{pp}} = -\int_{\Omega^e} (\mathbf{H}^e_{\mathrm{p}})^{\mathrm{T}} \frac{1}{K} \mathbf{H}^e_{\mathrm{p}}\, \mathrm{d}\Omega^e\,, \tag{3.10}$$

where $\mathbf{B}^e_{\mathrm{dev}}$ and $\mathbf{B}^e_{\mathrm{vol}}$ are the deviatoric and volumetric strain-displacement matrices. Matrix $\mathbf{H}^e_{\mathrm{p}}$ incorporates a set of shape functions used to interpolate the pressure degrees of freedom $\mathbf{p}^e$ to the pressure field (p). Under full incompressibility the system is solved for $\mathbf{d}^e$ and $\mathbf{p}^e$. For the nearly-incompressible cases, there is no inter-element continuity requirement for the pressure terms and these may be eliminated at the element level, by static condensation of the system of equations which can be solved for $\mathbf{d}^e$, similarly to the described in Eq. (2.18), using [15, 29]:

$$\mathbf{k}^e = \mathbf{k}^e_{\mathrm{dd}} - \mathbf{k}^e_{\mathrm{dp}} (\mathbf{k}^e_{\mathrm{pp}})^{-1} \mathbf{k}^e_{\mathrm{pd}}\,. \tag{3.11}$$

3.4 Selective reduced integration

3.4.1 Overview

According to Malkus and Hughes [19], the concept of selective reduced integration was first employed by Doherty, Wilson and Taylor [8] to obtain improved bending behavior with the 4-node quadrilateral elements under plane stress/strain states. A Gauss quadrature rule with only one point was used on the shear strain term, with a rule of order (2×2) being used to integrate the remaining terms. Malkus and Hughes later extended this method to the analysis of incompressible problems, successfully attenuating the volumetric locking effects. The same authors also demonstrated the equivalence between mixed methods and reduced and selective integration [15, 19].

Uniform reduced and selective reduced integration techniques were the first successful forms of dealing with locking problems. The great advantage of uniform reduced integration is the computational cost-effectiveness of the element formulation. The disadvantage, on the other hand, is that it can cause the element stiffness matrix to be rank-deficient, leading to spurious deformation patterns. Selective reduced integration successfully minimizes the effects of locking, while keeping the global stiffness matrix rank-efficient. Moreover, this procedure is a very simple way of attaining the performance of the mixed formulation without having to deal with additional complications [1, 15, 18].

The key to the implementation of the selective reduced integration is the separation of the stress field in the volumetric and deviatoric parts. Thus, for a plane strain state, the constitutive relation from Eq. (3.1) can be rewritten as [13, 22]:

$$\boldsymbol{\sigma} = \mathbf{D}^e_\mu \, \boldsymbol{\varepsilon} + \mathbf{D}^e_\lambda \, \boldsymbol{\varepsilon} \,, \tag{3.12}$$

or in the equivalent matrix form:

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} = \underbrace{\mu \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{D}^e_\mu} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} + \underbrace{\lambda \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{D}^e_\lambda} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} , \tag{3.13}$$

with the elasticity tensors $\mathbf{D}^e_\mu$ and $\mathbf{D}^e_\lambda$ verifying the following relation [13, 15]:

$$\mathbf{D}^e = \mathbf{D}^e_\mu + \mathbf{D}^e_\lambda \,. \tag{3.14}$$

From the classical formulation of the FEM, one may recall that the element internal load vector is defined as [23, 36]:

$$\mathbf{f}^e_{\text{int}} = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \boldsymbol{\sigma}^e \, \mathrm{d}\Omega^e \,. \tag{3.15}$$

Given the stress decomposition stated in Eq. (3.12), its possible to write the internal load vector in terms of the elasticity tensors $\mathbf{D}^e_\mu$ and $\mathbf{D}^e_\lambda$:

$$\mathbf{f}^e_{\text{int}} = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} (\mathbf{D}^e_\mu \, \boldsymbol{\varepsilon} + \mathbf{D}^e_\lambda \, \boldsymbol{\varepsilon}) \, \mathrm{d}\Omega^e \,. \tag{3.16}$$

Provided that the strain-displacement matrix $\mathbf{B}^e$ remains constant, the element stiffness matrix can be established by the variation of the internal loads such that [13, 22]:

$$\delta \mathbf{f}_{\text{int}}^e = \left[\int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}_\mu^e \, \mathbf{B}^e \, \mathrm{d}\Omega^e + \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}_\lambda^e \, \mathbf{B}^e \, \mathrm{d}\Omega^e \right] \delta \mathbf{d}^e = (\mathbf{k}_\mu{}^e + \mathbf{k}_\lambda{}^e) \, \delta \mathbf{d}^e \,, \quad (3.17)$$

with the element stiffness matrices $\mathbf{k}_\mu{}^e$ and $\mathbf{k}_\lambda{}^e$ verifying the following relation [15]:

$$\mathbf{k}^e = \mathbf{k}_\mu{}^e + \mathbf{k}_\lambda{}^e \,. \quad (3.18)$$

Due to the growth of the parameter (λ) towards infinity in the incompressible limit, the coefficients of $\mathbf{k}_\lambda^e$ become dominant and the matrix will be singular, whereas $\mathbf{k}_\mu^e$ will not. However, integrating $\mathbf{k}_\lambda^e$ with one-point Gauss quadrature rule helps on alleviating the overstiff response of the volumetric term [15, 19].

3.4.2 Selective integration of hydrostatic component

The incompressible limit (where $\nu = 0.5$) creates problems in the equations of compressible elasticity. The Lamé parameter λ becomes unbounded, and an alternative formulation is needed. In this formulation the constitutive law from Eq. (3.1) is rewritten to obtain the stress field in terms of the hydrostatic pressure and the Lamé parameter (μ) as [15]:

$$\boldsymbol{\sigma} = 2\mu \, \boldsymbol{\varepsilon} - p \, \mathbf{I} \,. \quad (3.19)$$

The bulk modulus can be expressed in terms of the Lamé parameter (λ) and the Poisson's ratio (ν) as follows:

$$K = \frac{\lambda(1+\nu)}{3\nu}, \quad (3.20)$$

which yields ($K = \lambda$) for the incompressible case. Recalling the relation from Eq. (3.3) it means that the hydrostatic pressure (p) is now obtained by [15]:

$$p = -\lambda \varepsilon_{kk} \,. \quad (3.21)$$

It is worth noting that Eq. (3.19) only holds for ($\nu = 0.5$) and needs to be modified in order to represent the more general case. For this end, the deviatoric stress $\mathbf{s}$ may be introduced [15, 22]:

$$\mathbf{s} = \boldsymbol{\sigma} - p \, \mathbf{I} \,, \quad (3.22)$$

which results in:

$$\boldsymbol{\sigma} = p \, \mathbf{I} + \mathbf{s} \,, \quad (3.23)$$

therefore, obtaining the stress field in terms of the strain field as the sum of two parts [22]:

$$\boldsymbol{\sigma} = \mathbf{D}_p^e \, \boldsymbol{\varepsilon} + \mathbf{D}_s^e \, \boldsymbol{\varepsilon} \,, \quad (3.24)$$

where the elasticity tensors $\mathbf{D}_p^e$ and $\mathbf{D}_s^e$ relate the strain field with the hydrostatic pressure and the deviatoric stress, respectively. For a plane strain state these elastic tensors are given by [13, 22]:

$$\mathbf{D}_p^e = K \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.25}$$

$$\mathbf{D}_s^e = \frac{E}{3(1+\nu)} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & \frac{3}{2} \end{bmatrix}, \tag{3.26}$$

with both tensors respecting the relationship:

$$\mathbf{D}^e = \mathbf{D}_p^e + \mathbf{D}_s^e \,. \tag{3.27}$$

The element stiffness matrix may then obtained by the sum of the hydrostatic and deviatoric terms [13, 22]:

$$\mathbf{k}^e = \mathbf{k}_p^e + \mathbf{k}_s^e \,, \tag{3.28}$$

with $\mathbf{k}_p^e$ and $\mathbf{k}_s^e$ being written as:

$$\mathbf{k}_p^e = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}_p^e \mathbf{B}^e \, \mathrm{d}\Omega^e \,, \tag{3.29}$$

$$\mathbf{k}_s^e = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}} \mathbf{D}_s^e \mathbf{B}^e \, \mathrm{d}\Omega^e \,. \tag{3.30}$$

3.4.3 Selective integration of shear components

In his thesis, Natal Jorge [22] proposes to combine the selective integration of the shear strain term with the selective integration of the hydrostatic pressure. For this purpose, the elastic tensors $\mathbf{D}_p^e$ and $\mathbf{D}_s^e$ are reformulated as follows:

$$\mathbf{D}_p^e = \frac{E}{3} \begin{bmatrix} \frac{1}{(1-2\nu)} & \frac{1}{(1-2\nu)} & 0 \\ \frac{1}{(1-2\nu)} & \frac{1}{(1-2\nu)} & 0 \\ 0 & 0 & \frac{3}{2(1+\nu)} \end{bmatrix}, \tag{3.31}$$

$$\mathbf{D}_s^e = \frac{E}{3(1+\nu)} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.32}$$

The element stiffness matrices are calculated in the same way as in Eqs. (3.29) and (3.30), applying an uniform reduced integration to $\mathbf{k}_p^e$, while integrating exactly $\mathbf{k}_s^e$.

3.5 B-bar method

3.5.1 Overview

The ease of implementation toghether with a performance similar to that of mixed methods made the selective reduced integration a popular approach to deal with incompressible problems, albeit with some drawbacks [9].

The equivalence theorems with mixed formulations are not valid in the axisymmetric case and the extension to anisotropic and orthotropic materials is ambiguous due to the splitting of the stiffness matrix, which is not clear in these cases. Additionally, the computational implementation is harder to achieve in non-isotropic materials. Generalization of the mixed formulation to these cases also tends to be complicated [9, 15].

These difficulties were overcame by a strain projection approach introduced by Hughes, generalizing the selective integration and mean-dilational formulations to the anisotropic case [15], a technique that became known as the B-bar method. Simo and Hughes [31] established the equivalence between the B-bar method and the variational three-field principle of Hu-Washizu.

3.5.2 B-bar formulation

Recalling the finite element formulation from Section 2.2.1, the strain-displacement matrix $\mathbf{B}^e$ can be expanded in terms of sub-nodal matrices in the form:

$$\mathbf{B}^e = \begin{bmatrix} \mathbf{B}_1^e & \mathbf{B}_2^e & \cdots & \mathbf{B}_{n_{\text{nodes}}}^e \end{bmatrix} ,$$

where (n_{nodes}) is the number of element nodes. A sub-matrix $\mathbf{B}_i^e$ is written as shown in Eq. (2.12) however, for the sake of simplicity of representation, let us consider the following set of substitutions:

$$\begin{cases} B_1 = \dfrac{\partial N_i}{\partial x} \\ B_2 = \dfrac{\partial N_i}{\partial y} \\ B_3 = \dfrac{\partial N_i}{\partial z} \end{cases} \quad \Rightarrow \quad \mathbf{B}_i = \begin{bmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \\ B_2 & B_1 & 0 \\ B_3 & 0 & B_1 \\ 0 & B_3 & B_2 \end{bmatrix} .$$

The main idea in the B-bar method is to split the strain-displacement matrix into its deviatoric and dilatational (volumetric) parts such that [9, 33]:

$$\mathbf{B}_i = \mathbf{B}_{\text{i}}^{\text{dil}} + \mathbf{B}_{\text{i}}^{\text{dev}} , \tag{3.33}$$

where $\mathbf{B}_i^{\text{dil}}$ and $\mathbf{B}_i^{\text{dev}}$ are the dilational and deviatoric matrices, respectively, defined as follows:

$$\mathbf{B}_i^{\text{dil}} = \frac{1}{3} \begin{bmatrix} B_1 & B_2 & B_3 \\ B_1 & B_2 & B_3 \\ B_1 & B_2 & B_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} , \tag{3.34}$$

$$\mathbf{B}_i^{\text{dev}} = \frac{1}{3}\begin{bmatrix} 2B_1 & -B_2 & -B_3 \\ -B_1 & 2B_2 & -B_3 \\ -B_1 & -B_2 & 2B_3 \\ 3B_2 & 3B_1 & 0 \\ 3B_3 & 0 & 3B_1 \\ 0 & 3B_3 & 3B_2 \end{bmatrix} . \tag{3.35}$$

For nearly incompressible applications it is necessary to weaken the contribution of the volumetric part. To that end, Hughes proposed the replacement of matrix $\mathbf{B}_i^{\text{dil}}$ by an improved dilatational matrix $\bar{\mathbf{B}}_i^{\text{dil}}$ [15]:

$$\bar{\mathbf{B}}_i^{\text{dil}} = \frac{1}{3}\begin{bmatrix} \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} . \tag{3.36}$$

The sub-matrix defined in Eq. (3.33) is then replaced by the following matrix:

$$\bar{\mathbf{B}}_i = \bar{\mathbf{B}}_i^{\text{dil}} + \mathbf{B}_i^{\text{dev}} . \tag{3.37}$$

Concerning the four-node bi-linear quadrilateral, one way of weakening the contribution of the volumetric term is to employ a selective reduced integration technique. Hence, matrix $\bar{\mathbf{B}}_i^{\text{dil}}$ is integrated using a one-point Gauss quadrature rule and the deviatoric matrix $\mathbf{B}_i^{\text{dev}}$ is integrated using the recommended Gauss rule for the given element [15]. Based on the mean dilatation introduced by Nagtegaal, Parks and Rice [21], Hughes proposed another way of calculating the dilatational contribution using the mean value of B_i, such that [15, 33]:

$$\bar{B}_i = \frac{\displaystyle\int_{\Omega^e} B_i \, \mathrm{d}\Omega^e}{\displaystyle\int_{\Omega^e} \mathrm{d}\Omega^e} \tag{3.38}$$

3.5.3 Plane strain state

The B-bar method applied to plane strain problems originates a strain-displacement sub-matrix of dimensions (4×2), in the form [22]:

$$\bar{\mathbf{B}}_i = \frac{1}{3}\begin{bmatrix} 2B_1 + \bar{B}_1 & -B_2 + \bar{B}_2 \\ -B_1 + \bar{B}_1 & 2B_2 + \bar{B}_1 \\ -B_1 + \bar{B}_1 & -B_2 + \bar{B}_2 \\ B_2 & B_1 \end{bmatrix} . \tag{3.39}$$

Employing the generalized selective reduced integration, the matrix is calculated in the same way, but the coefficients $\bar{B}_i$ are then integrated using a one-point Gauss quadrature rule [22].

3.6 The enhanced strain method

3.6.1 Overview

Before the introduction of the enhanced strain method, an alternative approach to the development of low-order elements with enhanced performance in coarse meshes was the classical method of incompatible modes, introduced in 1973 by Wilson, Taylor, Doherty and Ghaboussi in the context of plane elasticity [32]. Usually, element formulations adopt shape functions that are continuous over the whole element, even at its boundary, although the same may not happen to their derivatives. The class of (C^0) continuity elements refers to elements whose first derivatives of the degree of freedom variables are not continuous. Violation of the (C^0) continuity does not ensure convergence, nonetheless the incompatible modes approach employed discontinuous shape functions at the element boundaries, generating the so-called nonconforming or incompatible elements [15, 36, 38]. In this context, Wilson and his co-workers proposed the incompatible modes Q6 element. Studies showed the element was capable of attaining good results in bending-dominated problems. However, these results only held for rectangular-shaped elements and, consequently, the Q6 element did not pass the patch test for arbitrarly shaped quadrilaterals. The element was later reformulated by Taylor, Beresford and Wilson in order to correct this defficiency and respect the patch test, resulting in the QM6 element, suitable for more general analyses [15, 20].

In 1986, Simo and Hughes [31] had established that the class of assumed strain finite element procedures could be systematically formulated within a three-field variational framework of Hu-Washizu. In this context, Simo and Hughes showed that the independent stress field could be eliminated from the finite element equations provided a certain orthogonality condition on the assumed strain field was satisfied. Hence, the three-field formulation collapsed to a two-field mixed method in terms of the displacements and the enhanced strain field. Based on this principle, Simo and Rifai [32] introduced, in 1990, the enhanced strain method. The technique became popular for providing a means of overcoming the poor performance of standard low-order elements in bending-dominated problems, using coarse meshes. Futhermore, these elements were able to circumvent the problems associated with locking in the near incompressibility range [1, 2]. Simo and Rifai also stated that the procedure could be extended to plasticity problems, with the capability of incorporating inelastic effects without modification of the strain-driven return mapping algorithms. The same authors demonstrated that the incompatible modes Q6 element of Wilson, and its extension by Taylor to arbitrarly shaped quadrilaterals, arise as special cases within the context of the enhanced strain formulation [24, 32].

The enhanced strain method consists in augmenting the strain field with the inclusion of extra internal field variables, resulting in additional deformation modes [1]. These variables, termed generalized displacements, are not associated with the element nodes and may be thought of as internal degrees of freedom. The generalized displacements have no physical meaning, despite being used to calculate the strain field [15]. The procedure is formulated based on the following three-field Hu-Washizu variational functional [32]:

$$\Pi(\mathbf{u}, \tilde{\boldsymbol{\varepsilon}}, \boldsymbol{\sigma}) = \frac{1}{2}\int_{\Omega} \boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{D}\boldsymbol{\varepsilon}\,\mathrm{d}\Omega - \int_{\Omega} \mathbf{u}^{\mathrm{T}}\mathbf{b}\,\mathrm{d}\Omega - \int_{\Omega} \boldsymbol{\sigma}^{\mathrm{T}}\tilde{\boldsymbol{\varepsilon}}\,\mathrm{d}\Omega - \int_{\Gamma} \mathbf{u}^{\mathrm{T}}\bar{\mathbf{t}}\,\mathrm{d}\Gamma\,, \tag{3.40}$$

where the strain tensor $\boldsymbol{\varepsilon}$ is expressed as the sum of the symmetric gradient of the

displacement vector $\nabla^{\mathrm{s}}\mathbf{u}$ and the enhanced strains $\tilde{\boldsymbol{\varepsilon}}$:

$$\boldsymbol{\varepsilon} = \nabla^{\mathrm{s}}\mathbf{u} + \tilde{\boldsymbol{\varepsilon}}\,. \tag{3.41}$$

Hence, at the element level, the strain tensor is written as:

$$\boldsymbol{\varepsilon}^e = \mathbf{B}^e\mathbf{d}^e + \tilde{\mathbf{B}}^e\boldsymbol{\alpha}^e\,, \tag{3.42}$$

where $\boldsymbol{\alpha}^e$ is the vector of the generalized displacements. The enhanced strain field $\tilde{\boldsymbol{\varepsilon}}$ is interpolated for a given point in the element by means of the generalized displacements and properly defined shape functions, organized into matrix $\tilde{\mathbf{B}}^e$. The admissible choices of shape functions must satisfy the orthogonality condition in order to successfully eliminate the assumed stresses. Considering the generalized displacements are constant over the element domain, this yields [31, 32]:

$$\boldsymbol{\alpha}^e \int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}}\boldsymbol{\sigma}^e \,\mathrm{d}\Omega^e = \mathbf{0} \;\Leftrightarrow\; \int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}}\mathbf{D}^e\left[\mathbf{B}^e\mathbf{d}^e + \tilde{\mathbf{B}}^e\boldsymbol{\alpha}^e\right] = \mathbf{0}\,. \tag{3.43}$$

Referring to the above and recalling that for a small variation of the nodal displacements, the internal loads will change, such that:

$$\delta\mathbf{f}^e_{\mathrm{int}} = \int_{\Omega^e} (\mathbf{B}^e)^{\mathrm{T}}\mathbf{D}^e\left[\mathbf{B}^e\delta\mathbf{d}^e + \tilde{\mathbf{B}}^e\delta\boldsymbol{\alpha}^e\right], \tag{3.44}$$

the equilibrium of the enhanced strains element, for the linear-elastic case, may be expressed in the following system of equations [32]:

$$\begin{bmatrix} \mathbf{k}^e_{\mathrm{dd}} & \mathbf{k}^e_{\mathrm{d}\alpha} \\ \mathbf{k}^e_{\alpha\mathrm{d}} & \mathbf{k}^e_{\alpha\alpha} \end{bmatrix} \begin{Bmatrix} \mathbf{d}^e \\ \boldsymbol{\alpha}^e \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^e \\ \mathbf{0} \end{Bmatrix}, \tag{3.45}$$

where $\mathbf{k}^e_{\mathrm{dd}}$ is the stiffness matrix defined in Eq. (2.19) and the remaining matrices are computed as follows [32]:

$$\mathbf{k}^e_{\alpha\alpha} = \int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}}\mathbf{D}\tilde{\mathbf{B}}^e \,\mathrm{d}\Omega^e\,, \tag{3.46}$$

$$\mathbf{k}^e_{\alpha\mathrm{d}} = \int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}}\mathbf{D}\mathbf{B}^e \,\mathrm{d}\Omega^e\,. \tag{3.47}$$

The generalized displacements are unique to each element, thus may be eliminated on the element level by way of applying the static condensation of the system of equations defined earlier [15, 32]:

$$\boldsymbol{\alpha}^e = -(\mathbf{k}^e_{\alpha\alpha})^{-1}\mathbf{k}^e_{\alpha\mathrm{d}}\mathbf{d}^e\,, \tag{3.48}$$

hence, obtaining the equilibrium at the element level in terms of the nodal displacements:

$$\tilde{\mathbf{k}}^e\mathbf{d}^e = \mathbf{f}^e\,, \tag{3.49}$$

with $\tilde{\mathbf{k}}^e$ given as:

$$\tilde{\mathbf{k}}^e = \mathbf{k}^e_{\mathrm{dd}} - \mathbf{k}^e_{\mathrm{d}\alpha}(\mathbf{k}^e_{\alpha\alpha})^{-1}\mathbf{k}^e_{\alpha\mathrm{d}}\,. \tag{3.50}$$

Assembly of the element stiffness matrices is performed analogously to the described in Eqs. (2.21) and (2.22) from Section 2.2.1.

According to Simo and Hughes [31], one drawback of the variational framework of Hu-Washizu is precisely the stress recovery. Since the stresses are eliminated from the mixed formulation, the method does not generate equations to compute the stress parameters. Simo and Rifai [32] propose a least squares projection to obtain the stress. In 2000, based on his earlier work with Taylor in 1995, Piltner [24] proposes a modified version of the enhanced strains method that provides a systematic way of obtaining the equations for stress computation. As an alternative, the stress field may be computed, at the element level, by employing the relation [15]:

$$\tilde{\boldsymbol{\sigma}}^e = \mathbf{D}^e(\mathbf{B}^e\mathbf{d}^e + \tilde{\mathbf{B}}^e\boldsymbol{\alpha}^e)\,. \tag{3.51}$$

3.6.2 The Q6 element formulation

The motivation that prompted Wilson and his co-workers to propose the incompatible modes method was not originally related with incompressibility problems. They noted the Q4 element would respond in shear rather than bending, when subjected to a bending moment (see Fig. 3.2). This spurious shear was responsible for an overly stiff behavior [15].

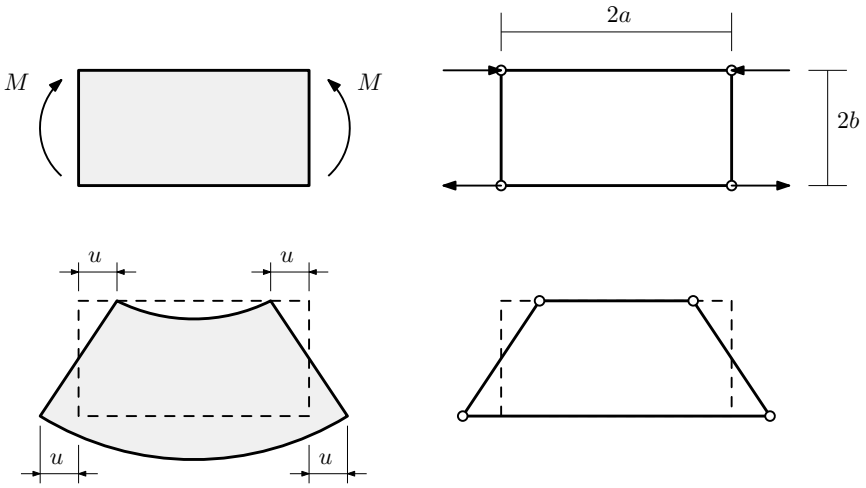


Figure 3.2: Discretization of a beam subjected to a bending moment and spurious shear response of the Q4 element.

The idea was to impose the element a certain curvature at its boundaries, without adding new nodes. The authors observed that one way of attaining this was to add quadratic modes of deformation. These observations led to the proposal of the Q6 element with the following displacement interpolation [22]:

$$\tilde{\mathbf{u}}^e = \left[\cdots \,\middle|\, \begin{matrix} N_i(\xi,\eta) & 0 \\ 0 & N_i(\xi,\eta) \end{matrix} \,\middle|\, \cdots\right]\mathbf{d}^e + \left[\cdots \,\middle|\, \begin{matrix} N_j(\xi,\eta) & 0 \\ 0 & N_j(\xi,\eta) \end{matrix} \,\middle|\, \cdots\right]\begin{Bmatrix}\boldsymbol{\alpha}_5 \\ \boldsymbol{\alpha}_6\end{Bmatrix}, \tag{3.52}$$

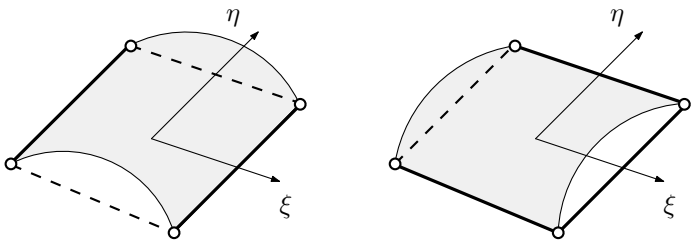
with:

$$\begin{cases} i = 1, n \\ j = n, n + n_{\text{modes}} \end{cases} .$$

The first part of Eq. (3.52) is the standard bi-linear interpolation of the nodal displacements and the second part is the new interpolation where the vectors $\boldsymbol{\alpha}_5$ and $\boldsymbol{\alpha}_6$ contain the generalized displacements, defined as [22]:

$$\boldsymbol{\alpha}_5^{\mathrm{T}} = \begin{Bmatrix} \alpha_{15} & \alpha_{25} \end{Bmatrix} , \qquad \boldsymbol{\alpha}_6^{\mathrm{T}} = \begin{Bmatrix} \alpha_{16} & \alpha_{26} \end{Bmatrix} . \tag{3.53}$$

The generalized displacements are interpolated with the shape functions (N_5) and (N_6) depicted in Fig. 3.3. These are the quadratic functions leading to the incompatible modes of deformation that disrupt the inter-element continuity, thus generating a discontinuous displacement field [15].



$N_5 = (1 - \xi^2)$ $N_6 = (1 - \eta^2)$

Figure 3.3: Shape functions for the enhanced strain field of the element Q6.

Considering only small deformations under plane strain, the Q6 formulation results in the following strain field [22]:

$$\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} = \left[\begin{array}{cc|c} N_{i,x} & 0 & \\ 0 & N_{i,y} & \cdots \\ N_{i,y} & N_{i,x} & i=1,n \end{array}\right] \begin{Bmatrix} u_i \\ v_i \\ \vdots \\ u_n \\ v_n \end{Bmatrix} + \begin{bmatrix} N_{5,x} & 0 & N_{6,x} & 0 \\ 0 & N_{5,y} & 0 & N_{6,y} \\ N_{5,y} & N_{5,x} & N_{6,y} & N_{6,x} \end{bmatrix} \begin{Bmatrix} \alpha_{15} \\ \alpha_{25} \\ \alpha_{16} \\ \alpha_{26} \end{Bmatrix} , \tag{3.54}$$

or in a more condensed form:

$$\boldsymbol{\varepsilon} = \mathbf{B}^e \mathbf{d}^e + \tilde{\mathbf{B}}^e \boldsymbol{\alpha}^e . \tag{3.55}$$

The indexes (i, x), for instance, indicate the first derivative of the shape function (i) in respect to the cartesian coordinate (x). The shape functions are first evaluated in the natural coordinate system and then mapped to the global coordinate system by means of the jacobian matrix, similarly to the described earlier in Section 2.2.3. The element stiffness and subsequent assembly process is analogous to the one described in Eqs. Equations (3.45) to (3.50) for the enhanced strain method.

With the use of quadratic shape functions to interpolate the enhanced strain field, the Q6 element was capable of attaining good results in bending-dominated problems but only for rectangular elements, thus failing the patch test for the more general case of arbitrarily shaped quadrilaterals [15].

3.6.3 The QM6 element formulation

In order to solve the incompatibility of the Q6 element formulation with the patch test, Taylor, Beresford and Wilson proposed a modified incompatible modes version, called QM6 element. They stated that if the displacements were set according to a given linear polynomial, the incompatible modes need not be activated, that is $\boldsymbol{\alpha}^e = \mathbf{0}$. From the first equation of (3.45), this condition only holds if [15]:

$$\mathbf{k}^e_{\alpha \mathrm{d}} \mathbf{d}^e = \mathbf{0}\,. \tag{3.56}$$

Recalling Eq. (3.47), one concludes that for a constant stress field the condition given above is equivalent to [15]:

$$\boldsymbol{\sigma}^e \int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}} \,\mathrm{d}\Omega^e = \mathbf{0}\,, \tag{3.57}$$

for which it is sufficient to prove that:

$$\int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}} \,\mathrm{d}\Omega^e = \mathbf{0}\,. \tag{3.58}$$

Recalling the definition of $\tilde{\mathbf{B}}^e$ from Eq. (3.54), and using a Gauss quadrature rule, one obtains:

$$\int_{\Omega^e} (\tilde{\mathbf{B}}^e)^{\mathrm{T}} \,\mathrm{d}\Omega^e = \frac{2}{\det \mathbf{J}^e} \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} -\xi y_{,\eta} & 0 & \xi x_{,\eta} \\ 0 & \xi x_{,\eta} & -\xi y_{,\eta} \\ \eta y_{,\xi} & 0 & -\eta x_{,\xi} \\ 0 & -\eta x_{,\xi} & \eta y_{,\xi} \end{bmatrix} \det \mathbf{J}^e \,\mathrm{d}\xi \,\mathrm{d}\eta\,, \tag{3.59}$$

The above equation will integrate to zero if the derivatives of (x) and (y) in respect to the natural coordinates are constants. This is true if the element is a rectangle or a parallelogram, but linear terms will appear for general quadrilaterals. Taylor and his co-workers proposed replacing the derivatives by their values at $(\xi = \eta = 0)$ [15]. Later Simo and Rifai [32] proposed an equivalent procedure for the enhanced strains element design, resulting in the following definition for the enhanced strain-displacement matrix $\tilde{\mathbf{B}}^e$:

$$\tilde{\mathbf{B}}^e = \frac{\det \mathbf{J}^e}{\det \mathbf{J}^e(\mathbf{0})} \mathbf{F}_0^{-\mathrm{T}} \begin{bmatrix} N_{5,x} & 0 & N_{6,x} & 0 \\ 0 & N_{5,y} & 0 & N_{6,y} \\ N_{5,y} & N_{5,x} & N_{6,y} & N_{6,x} \end{bmatrix}\,, \tag{3.60}$$

with matrix $\mathbf{F}$ defined as follows:

$$\mathbf{F}_0 = \begin{bmatrix} J_{11}^2 & J_{21}J_{12} & 2J_{11}J_{12} \\ J_{12}J_{21} & J_{22}^2 & 2J_{21}J_{22} \\ J_{11}J_{21} & J_{12}J_{22} & J_{11}J_{22} + J_{12}J_{21} \end{bmatrix}_{\xi=\eta=0}\,, \tag{3.61}$$

where the coefficients (J_{ij}) are the terms of the jacobian matrix of the transformation from natural to global coordinates, evaluated at the center of the element.

These modifications allow the element to pass the patch test, thus making it suitable for general use. The Q6 element, on the other hand, should be used only for rectangular or parallelogram-shaped elements [15].

3.6.4 The Q4/6I element formulation

Based on the incompatible modes element of Wilson and Taylor, Natal Jorge [22] proposes in his thesis the element Q4/6I with six extra deformation modes. The respective shape functions are presented in table 3.1.

Table 3.1: Shape functions for the six incompatible modes.

Modes	N_{α_i}
α_5	$\frac{1}{2}\,\eta(\eta-1)(1-\xi^2)$
α_6	$\frac{1}{2}\,\xi(\xi+1)(1-\eta^2)$
α_7	$\frac{1}{2}\,\eta(\eta+1)(1-\xi^2)$
α_8	$\frac{1}{2}\,\xi(\xi-1)(1-\eta^2)$

Following the enhanced strain procedure, the strain field at the element level is interpolated as:

$$\boldsymbol{\varepsilon} = \mathbf{B}^e\mathbf{d}^e + \tilde{\mathbf{B}}^e\boldsymbol{\alpha}^e\,, \tag{3.62}$$

where the first part is the standard bi-linear interpolation of the nodal displacements, already described in previous sections, whereas the second part is the augmented component of the strain field, with the vector of generalized displacements defined as:

$$\boldsymbol{\alpha}^e = \begin{Bmatrix}\alpha_{51} & \alpha_{57} & \alpha_{68} & \alpha_{62} & \alpha_{71} & \alpha_{82}\end{Bmatrix}. \tag{3.63}$$

The enhanced strain-displacement matrix $\tilde{\mathbf{B}}^e$ is written as follows:

$$\tilde{\mathbf{B}}^e = \begin{bmatrix} N_{5,\xi} & 0 & N_{6,\xi}+N_{8,\xi} & 0 & N_{7,\xi} & 0 \\ 0 & N_{5,\eta}+N_{7,\eta} & 0 & N_{6,\eta} & 0 & N_{8,\eta} \\ N_{5,\eta}+N_{7,\eta} & N_{5,\xi} & N_{6,\eta} & N_{6,\xi}+N_{8,\xi} & N_{8,\eta} & N_{7,\xi} \end{bmatrix}. \tag{3.64}$$

The respective shape function derivatives for the extra modes of deformation are presented in table 3.2. The author further demonstrates that the proposed modes of deformation agree with the orthogonality condition.

Table 3.2: Shape function derivatives for the six incompatible modes

Modes	$N_{\alpha_i,\xi}$	$N_{\alpha_i,\eta}$
α_5	$-\xi\eta(\eta-1)$	$\frac{1}{2}\,(2\eta-1)(1-\xi^2)$
α_6	$\frac{1}{2}\,(2\xi+1)(1-\eta^2)$	$-\xi\eta(\xi+1)$
α_7	$-\xi\eta(\eta+1)$	$\frac{1}{2}\,(2\eta+1)(1-\xi^2)$
α_8	$\frac{1}{2}\,(2\xi-1)(1-\eta^2)$	$-\xi\eta(\xi-1)$

3.6.5 The Q4/4I element formulation

In order to reduce the number of of incompatible modes of the element Q4/6I, Natal Jorge [22] proposes the following enhanced strain-displacement matrix $\tilde{\mathbf{B}}^e$:

$$\tilde{\mathbf{B}}^e = \begin{bmatrix} N_{5,\xi} + N_{7,\xi} & 0 & N_{6,\xi} + N_{8,\xi} & 0 \\ 0 & N_{5,\eta} + N_{7,\eta} & 0 & N_{6,\eta} + N_{8,\eta} \\ N_{5,\eta} + N_{7,\eta} & N_{5,\xi} + N_{7,\xi} & N_{6,\eta} + N_{8,\eta} & N_{6,\xi} + N_{8,\xi} \end{bmatrix} . \tag{3.65}$$

According to the author this configuration for the enhanced modes respects the orthogonality condition. The enhanced strain field is calculated as follows:

$$\tilde{\boldsymbol{\varepsilon}} = \tilde{\mathbf{B}}^e \begin{Bmatrix} \alpha_{51} \\ \alpha_{52} \\ \alpha_{61} \\ \alpha_{62} \end{Bmatrix} . \tag{3.66}$$

The element has four extra modes of deformation, the same number as the Q6 element, however Natal Jorge states that the modes proposed in his work are responsible for a linear variation of the strain.

3.6.6 The Qi5 element formulation

The compatible modes Qi5 element was proposed by César de Sá and Natal Jorge [6]. The element is constructed by the addition of two extra modes of deformation, similarly to the Wilson-Taylor element. The shape function leading to the compatible modes of deformation agrees with the displacement field and is written as:

$$N_\alpha = (1 - \xi^2)(1 - \eta^2) , \tag{3.67}$$

which leads to the following derivatives in respect to the local coordinates:

$$\begin{cases} N_{\alpha,\xi} = -2\xi(1 - \eta^2) \\ N_{\alpha,\eta} = -2\eta(1 - \xi^2) \end{cases} . \tag{3.68}$$

At the element level the strain field is defined as follows:

$$\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} = \left[\begin{array}{cc|c} N_{i,x} & 0 & \\ 0 & N_{i,y} & \cdots \\ N_{i,y} & N_{i,x} & {\scriptstyle i=1,n} \end{array}\right] \begin{Bmatrix} u_i \\ v_i \\ \vdots \\ u_n \\ v_n \end{Bmatrix} + \begin{bmatrix} N_{5,x} & 0 \\ 0 & N_{5,y} \\ N_{5,y} & N_{5,x} \end{bmatrix} \begin{Bmatrix} \alpha_{51} \\ \alpha_{52} \end{Bmatrix} . \tag{3.69}$$

The shape functions derivatives are first evaluated in the natural coordinates and then mapped to the global coordinates by means of the jacobian matrix.

The authors further state that the use of compatible modes leads the element to respect the orthogonality condition, without being necessary to evaluate the jacobian at the center of the element.

3.6.7 The Qi6 element formulation

Inspired by the works of Simo and Rifai [32] and Simo and Armero, César de Sá and Natal Jorge [6] propose the compatible modes Qi6 element based on their Qi5 element described in the previous section.

The element contains four compatible modes of deformation using the same shape functions as the Qi5 element. By respecting the orthogonality condition, the shape function derivatives need not be evaluated at the center of the element. The strain field, at the element level is defined such that:

$$\begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} = \left[\begin{array}{cc|c} N_{i,x} & 0 & \\ 0 & N_{i,y} & \cdots \\ N_{i,y} & N_{i,x} & i{=}1,n \end{array}\right] \begin{Bmatrix} u_i \\ v_i \\ \vdots \\ u_n \\ v_n \end{Bmatrix} - 2 \begin{bmatrix} N_{\alpha,x} & 0 & 0 & 0 \\ 0 & N_{\alpha,y} & 0 & 0 \\ 0 & 0 & N_{\alpha,y} & N_{\alpha,x} \end{bmatrix} \begin{Bmatrix} \alpha_{51} \\ \alpha_{52} \\ \alpha_{61} \\ \alpha_{62} \end{Bmatrix} \tag{3.70}$$

Chapter 4

Computational implementation

4.1 Overview

The importance of numerical simulation justifies the development of software solutions that take advantage of computational resources to solve distinct problems in various fields of engineering. There are several commercial solutions available in the market, for simulation and structural analysis, based on the FEM.

During undergraduate studies, the training given to engineering students on FEA often makes use of commercial software. However, these solutions are aimed at the professional practice of engineering, offering a set of features with capabilities far superior to those seen in classes. Moreover, the use of commercial software in this training environment leads the learning process to be heavily focused in the pre- and post-processing parts, overlooking the FEA itself.

For the undergraduate student an excellent alternative to gain a deeper understanding into the FEM is to develop its own finite element code. A successful implementation not only requires the student to understand and carefully review the classical FEM formulation, but also often confronts him with specific topics not approached in class, requiring further scientific literature review. This work-flow renders the student with the necessary knowledge about the specificities of the method, its advantages and limitations. Additionally, a deeper insight into the purpose and behavior of many element formulations may be gained through the benchmark tests used to validate the code.

Given this motivation, during this dissertation a software solution was developed for the study and implementation of several finite element formulations aimed at the analysis of incompressible problems. Although focused in this particular topic during this work, the development of this program was targeted at the higher purpose of serving as the base for a finite element analysis platform, open to further development inside our department. The idea was not only to create a means for students and/or researchers to implement and test different finite element formulations, but also allow them to contribute with added functionalities to the software, in subsequent years.

4.2 Program architecture

The constraints imposed by a short time period available to finish a dissertation work, forced the development to focus more on understanding and implementation of different finite element formulations and the necessary underlying infra-structure in order on make them work. Given this limitation, little attention was given to more pure programming and architecture aspects.

Nonetheless, due to the open-source nature of the software and to facilitate its growth over time with the introduction of code from different users, there was no doubt that a modular architecture should be privileged during development. This could be achieved by means of a structured programming approach, relying heavily on subroutines and structured control flow constructs. This philosophy was employed to a certain extent during this work, however, the code has room to be optimized in the future, by other students/researchers. For example, many functions are still too big or have duplicate code that could otherwise be transformed into smaller simpler functions.

Another important aspect to keep in mind is to maintain the code as dynamic as possible, avoiding hard-coded elements at all cost. Although this is achievable, it may be difficult to follow and presents some drawbacks. For example, in the current code the shape functions for the quadrilateral elements are generated dynamically and may be systematically obtained for higher-order serendipity elements. In order to facilitate this implementation, symbolic calculus had to be used, however, manipulation of symbolic expressions is slower than usual. An alternative is to hard code the matrices into the program as formulas. Although this may be faster in terms of program execution, it overpopulates the code and it is more prone to introduction errors.

At the current state of development, the program is composed of one main routine called `planeStress` and a total of seventeen sub-routines or functions. Some important sub-routines are:

- `readInput` - reads and parses model information from an Abaqus `.inp` file;
- `intPoints` - provides the natural coordinates of the Gaussian integration points and corresponding weights;
- `shapeFunctions` - provides the nodal shape functions for quadrilateral elements;
- `nodeSelection` - provides user interface elements for the user to define boundary conditions;
- `elementStiffnessMatrix` - computes element stiffness matrices and executes their assembly into the global matrix;
- `computeStress` - provides the stress parameters at the nodal points;
- `postProcessing` - provides user interface elements for the user to analyze the results.

These functions are reviewed in more detail over the next sections. In Appendix A, an unified modeling language (UML) diagram is shown, depicting the interactions between various sub-routines during a normal program execution.

4.3 Pre-processing

4.3.1 Model definition

Given the time constraints imposed for this work, automatic mesh generation algorithms were not implemented. Instead, the idea was to use a commercial software to define the model and export all the input data to a file, which could be imported into MATLAB and read by the program. To accomplish such task, the software had to be able to read the input file and parse the necessary data into appropriate variables. For the model definition, the choice fell on Abaqus student version, mainly because it is widely used in our department and during our course. Moreover, the CAD capabilities and the graphical user interface facilitate drawing the geometry and mesh generation. All these parameters can also be exported to an input file (.inp) that follows a specific format. An excerpt of an input file generated for a simple two-dimensional problem is shown in Appendix B.

Observing the example, one can immediately notice the data is organized into blocks inside labels. Such labels are identified by certain keywords, for example:

```
1 **PARTS
2 **ASSEMBLY
3 *Node
4 *Element
```

A mesh is defined by the nodal coordinates and the connectivity information. Thus, essentially we are interested in extracting the data inside the labels `*Nodes` and `*Element`. The nodal coordinates are presented inside the first label in the format:

```
1 *Node
2       1,           0.,           0.
3       2,           5.,           0.
4       3,          10.,           0.
5       4,           0.,           2.
6       5,           5.,           2.
7       6,          10.,           2.
```

where the first column contains the node labels and the second and third columns contain, respectively, the (x) and (y) coordinates of the nodal points. The connectivity information is listed inside the second label as follows:

```
1 *Element, type=CPS4
2 1, 1, 2, 5, 4
3 2, 2, 3, 6, 5
```

where the first column lists the element labels and the remaining columns list the nodes belonging to a given element, ordered in the counter-clockwise direction.

The function `readInput` provides the necessary algorithm to access the input file and read its contents in order to extract the information described above. When the function is called, a file explorer dialog is presented to the user to select the desired file to import. This is accomplished by the command `uigetfile` such that:

```
1 [fileName,pathName,~]=uigetfile({'*.inp';'*.txt'},'Select input file');
```

whose output variables are the file name and the file path. Essentially, this information is used for the program to know what file should be opened when executing the command `fopen`. Once the file is opened the routine uses the command `textscan` to read its content into a variable and closes the file afterwards, as follows:

```
1 fid=fopen(strcat(pathName,'/',fileName),'r');
2 fileContent=textscan(fid,'%[^\n]');
3 fclose(fid);
```

The file data is stored into the variable `fileContent`, which is a (1×1) cell array. Its contents need to be extracted to a different cell array, called `fileLines` which in turn will be a single column cell with as many lines as the file contents. At this point, the function searches for the labels `*Node` and `*Element` to know their respective line indexes, using the command `find`:

```
1 idxNode=find(contains(fileLines,'*Node'));
2 idxElement=find(contains(fileLines,'*Element'));
```

Based on the indexes, the routine proceeds to read the nodal coordinates, retrieving all the numerical data appearing after the label `*Node` until it reaches the next label, marked by an asterisk `*`:

```
1 i=idxElement+1;
2 while ~contains(fileLines(i,1),'*')
3 i=i+1;
4 end
5
6 elementRead=fileLines(idxElement+1:i-1,1);
```

With the nodal coordinates successfully extracted, the routine needs to parse the information into matrix. It can be seen that the raw data is stored in the form of numerical strings separated by commas. The command `regexp` is used to split the data at commas, obtaining a cell array with one column and as many lines as the number of nodes, which is later converted into a matrix by using the command `vertcat`, as follows:

```
1 parseNodes=regexp(nodeRead, ',', 'split');
2 nodes = str2double(vertcat(parseNodes{:}));
```

The same procedure is employed to extract the connectivity info. The code was further enhanced to deal with geometries composed of several parts. The end result is a structure array with the name `model` organized as follows:

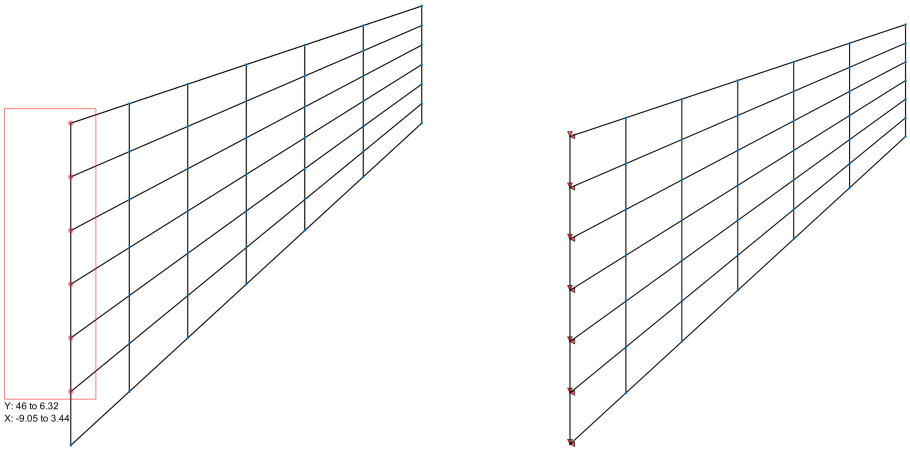
```
model
└─ Part
   ├─ Name: "Part-1"
   ├─ Node: [6x3 double]
   └─ Connect:
      ├─ Type: "quad"
      ├─ Nodes: 4
      └─ Elements: [2x5 double]
```

4.3.2 Boundary conditions

In order to facilitate the introduction of the boundary conditions, a simple graphical user interface was developed for the purpose. After the program execution, at certain point the user is prompted to define the boundary conditions of the model by means of the sub-routine `nodeSelection`. A figure window is launched and a representation of the meshed geometry with the nodal points is presented to the user via a plot object. The mesh representation is achieved by means of the `patch` command, as follows:

```
patch(ax,'faces',connectInfo(:,2:end),...
      'vertices',[xData(:) yData(:)],...
      'facecolor',[0.9 0.9 0.9],'facealpha',0.65);
```

where `xData` and `yData` are column vectors containing the cartesian coordinates of the nodal points. The `patch` command accepts nodal coordinates and connectivity info in the same format as the information extracted from the input file. For example, the code snippet above results in the following representation of (6×6) mesh for the cook's membrane shown in Fig. 4.1. The user is also allowed to select different nodes, using a brush tool, in order to define the boundary conditions.



(a) (b)

Figure 4.1: Meshed geometry representation with (a) node selection using data brush and (b) boundary condition representation.

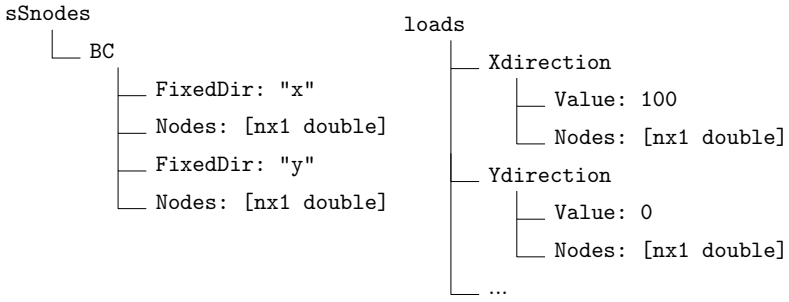
The data brush tool outputs a logical row vector of length equal to the number of nodes, with the value one at the indexes of the selected nodes. Using the command `find` is possible to use this information in order to obtain the node labels, as follows:

```
fixed=find(nodes.BrushData);
```

The interactions with the graphic interface are interpreted by nested callback functions defined within the main sub-routine. A nested function is invisible outside of its immediately enclosing function, but can access all local variables of the enclosing function. This particularity allowed to easily add these features to the software, while managing to convert the user graphical input into useful data variables. A callback is defined as

any normal function, the only difference is that it is pointed out to the object handler as the routine that will trigger upon certain interaction.

After defining the boundary conditions, the sub-routine originates three output variables: `fixedNodes`, `sSnodes` and `loads`. The first output is a simple row vector containing the labels of the pinned nodes. The second output is a structure array containing information in order to identify which degree of freedom is fixed for the simply supported nodes. The last output is also a structure array but contains information about which nodes have applied forces. The variable also holds data about the direction of the forces and their magnitudes. The last two data structures are defined as follows:



These structures are later handled by the functions `setNodalForces` and `setFreeNodes`. The first function outputs the external force vector to be used in the global system of equations. The second function outputs a vector containing the active degrees of freedom. The fixed degrees of freedom corresponding to the pinned nodes are stored in the vector `dofs` and are determined from the node labels in the vector `nodeOrder`:

```
1 nodeOrder=1:numNodes;
2 dofs=[];
3 for k=1:dOf
4     nodes=nodeOrder(fixedNodes);
5     dofs=[dofs,(nodes-1).*dOf+k];
6 end
```

where `dOf` is the number of nodal degrees of freedom. For the simply supported nodes, a similar process is conducted with the only difference being the fact that the code needs to navigate inside the structure array `sSnodes`. Near the end of execution, the routine defines a vector of global degrees of freedom and detects the active ones using the command `setdiff`, essentially setting the difference between the vectors and `globalDofs` and `dofs`:

```
1 globalDofs=1:dOf*numNodes;
2 activeDoF=setdiff(globalDofs,dofs);
```

The output is used as a mask in order to solve the global system of equations for the active degrees of freedom only.

4.4 Finite element formulas

4.4.1 Shape functions generation

The software is capable of dynamically generating the nodal shape functions for quadrilateral elements only. In theory, the algorithm is capable of generating correct expressions for two-dimensional serendipity rectangular elements of any order. Further tests need to be conducted in order to confirm this.

The sub-routine `shapeFunctions` literally follows the theoretical procedure described in [36] and [15], among others, to obtain the shape functions for the quadrilateral elements based on the one-dimensional Lagrange polynomial family in natural coordinates. For this end, the use of symbolic expressions was essential, allowing for the treatment of mathematical equations with unknown variables. The sub-routine starts by defining the necessary symbolic variables and matrices:

```
1 syms csi eta
2
3 nCsiI=zeros(dim,1);
4 nEtaI=zeros(dim,1);
5 nCsiI=sym(nCsiI);
6 nEtaI=sym(nEtaI);
```

The variable `dim` is related to the number of nodes per side of the element. This is used to determine the natural coordinates of the nodes along these segments such that:

```
1 nodeCoord=linspace(-1,1,dim);
```

The sub-routine then proceeds to calculate the shape functions of the one-dimensional segments of the quarilateral element, for both natural coordinates, such that [36]:

$$N_i(\xi) = \prod_{j=1(j\neq i)}^{n_{nodes}} \frac{(\xi - \xi_j)}{(\xi_i - \xi_j)}, \tag{4.1}$$

$$N_i(\eta) = \prod_{j=1(j\neq i)}^{n_{nodes}} \frac{(\eta - \eta_j)}{(\eta_i - \eta_j)}. \tag{4.2}$$

For that end, the sub-routine evaluates the one-dimensional segment at each node coordinate, along the ξ-direction and then creates the corresponding function along the η-direction by means of the command `subs` to substitute the variable `csi` by the variable `eta` in the symbolic expressions. This accomplished by the following loop:

```
1  for i=1:length(nodeCoord)
2      N=1;
3      csiI=setdiff(nodeCoord,nodeCoord(i));
4      %Constructing one-dimensional shape function over csi
5      for j=1:length(csiI)
6          N=N*(csi-csiI(j))/(nodeCoord(i)-csiI(j));
7      end
8
9      nCsiI(i,1)=N;
10     %Constructing one-dimensional shape function over eta
11     nEtaI=subs(nCsiI,csi,eta);
12 end
```

The end result for a four-node quadrilateral are the one-dimensional Lagrange polynomials defined as [36]:

$$\begin{cases} N_1(\xi) = \frac{1}{2}(1-\xi) \\ N_1(\eta) = \frac{1}{2}(1-\eta) \\ N_2(\xi) = \frac{1}{2}(1+\xi) \\ N_2(\eta) = \frac{1}{2}(1+\eta) \end{cases} . \tag{4.3}$$

Afterwards, the sub-routine proceeds to calculate the bi-dimensional expressions. The underlying concept is that a given node connects two one-dimensional segments, thus the corresponding shape function results from the combination of the expressions from both segments [36]. This is achieved by multiplying the one-dimensional shape functions defined in the expression above, such that:

$$\begin{cases} N_1(\xi,\eta) = N_1(\xi)N_1(\eta) \\ N_2(\xi,\eta) = N_2(\xi)N_1(\eta) \\ N_3(\xi,\eta) = N_2(\xi)N_2(\eta) \\ N_4(\xi,\eta) = N_1(\xi)N_2(\eta) \end{cases} , \tag{4.4}$$

which is accomplished by the sub-routine by the executing the following code:

```
%Constructing nodal bi-dimensional shape functions (csi,eta)
k=1;
for i=1:length(nodeCoord)-1
    nodeFun(k,1)=nCsiI(i,1)*nEtaI(1,1);
    k=k+1;
end

for i=1:length(nodeCoord)-1
    nodeFun(k,1)=nCsiI(end,1)*nEtaI(i,1);
    k=k+1;
end

for i=length(nodeCoord):-1:2
    nodeFun(k,1)=nCsiI(i,1)*nEtaI(end,1);
    k=k+1;
end

for i=length(nodeCoord):-1:2
    nodeFun(k,1)=nCsiI(1,1)*nEtaI(i,1);
    k=k+1;
end
```

The expressions are consecutively stored into the matrix `nodeFun` which is given as output. This matrix has one column and as many lines as the number of nodes of the element. For the case of a bi-linear quadrilateral element, this results essentially in the same structure of expressions of Eq. 2.28, but now in matrix form. These expressions are later used for computing the stiffness matrices and the strain and stress fields.

4.4.2 Gauss points generation

The numerical integration procedure employed by the program is based in the Gauss-Legendre quadrature. The number of quadrature points and their locations is defined by the element order, which is related to the number of nodes. Using a Gauss-Legendre quadrature rule, the integral of a given function $f(x)$ can be approximated as follows [15, 36]

$$\int_{-1}^{+1} f(x)\,\mathrm{d}x = \int_{-1}^{+1} g(x)W(x)\,\mathrm{d}x \approx \sum_{i=1}^{p} g(x_i)w(x_i)\,, \tag{4.5}$$

where $g(x)$ is a polynomial approximation of $f(x)$ and $W(x)$ is an appropriate weighting function, which in turn are approximated by a sum of function values at specific points (x_i) multiplied by some weights (w_i). The coordinates of the quadrature points and the values of the associated weights are determined to attain maximum accuracy while integrating the function numerically. The integration points are the roots of the n-th order Legendre polynomials, which may be defined by the recursively as [15, 28]:

$$P_n(x) = \frac{(2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x)}{n}. \tag{4.6}$$

The first derivative of the polynomial may also be defined recursively:

$$P'_n(x) = \frac{n}{x-1}\left[xP_n(x) - P_{n-1}(x)\right]\,. \tag{4.7}$$

The polynomial roots are not solvable analytically and have to numerically approximated. This can be achieved by the Newton-Raphson method, such that:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{4.8}$$

using the first-guess (x_0) for the i-th root of a n-order polynomial given by:

$$x_0 = \left(\pi\frac{i-0.25}{n+0.5}\right)\,. \tag{4.9}$$

The corresponding weights are computed by employing:

$$w_i = \frac{2}{\left(1-x_i^2\right)\left[P'_n(x_i)\right]^2}. \tag{4.10}$$

The procedure described above is available in [28] and is employed by the software by means of the sub-routine `gaussLegQ`. The code can be consulted in Appendix C. The output variable `q` is a matrix with as many lines as the number of roots of the polynomial and two columns. The first column contains the said polynomial roots and the second column contains the corresponding weights.

4.4.3 Assembly algorithm

The element stiffness matrices are calculated and assembled by executing the sub-routine `elementStiffnessMatrix`. The code follows the theoretical procedure already described in previous sections. The process starts by initializing the necessary matrices and symbolic variables. Afterwards, the routine computes the derivatives of the shape functions, in respect to the natural coordinates (ξ) and (η), applying the command `diff` to the symbolic expressions, as follows:

```
dNCsiEta=[diff(nodeShapeFun,csi).';diff(nodeShapeFun,eta).'];
```

At this point, the sub-routine iterates over the elements, starting by finding the degrees of freedom associated with the current element and storing them in the vector `indexB`. This is accomplished in the following inner-loop:

```
elConnect=connectInfo(i,:);      %Reading connectivity
idx=1;                           %Auxiliary counter

for x=1:length(elConnect)
    for y=1:dOf
        indexB(idx)=(elConnect(1,x)-1)*dOf+y;
        idx=idx+1;
    end
end
```

Following this operation, another inner-loop is initiated in order to iterate over the integration points. For each iteration the derivatives of the shape functions are computed at the natural coordinates of the current integration point using, once again, the command `subs`, such that:

```
for j=1:size(intPts,1)
    dN=dNCsiEta;
    %Computing derivatives in the integration point
    dN=subs(dN,[csi,eta],[intPts(j,1),intPts(j,2)]);

    %Converting from symbolic to double
    dN=double(dN);

    J=dN*elNodeCoord(:,:,i);     %Jacobian
    dNdXdY=J\dN;                 %Converting to global coordinates
```

The symbolic expressions are then converted to the numeric data type `double` and the execution then proceeds to compute the jacobian matrix based on the nodal coordinates of the element, stored in a multidimensional array called `elNodeCoord`, of dimensions ($n_{\text{nodes;elem}} \times 2 \times n_{\text{elem}}$). In subsequent operations, the derivatives are converted to global coordinates and the strain-displacement matrix `B` is constructed. In the end of the iteration, the element stiffness is calculated and directly assembled into the global stiffness matrix `K`, using the vector `indexB` as mask, such that:

```
K(indexB,indexB)=K(indexB,indexB)+B'*D*B*wCsi*wEta*det(J)*elThick;
```

where `elThick` is the element thickness, `wCsi` and `wEta` are the weighting coefficients of each natural coordinate. The vector `indexB` indicates which degrees of freedom along the lines and columns of the global matrix should be populated. The loop then proceeds to repeat the process for the subsequent integration points. Once all the points are evaluated, the outer-loop proceeds to the next element. The process can be easily visualized by referring to the flowchart shown in Appendix D.

4.5 Post-processing

4.5.1 Stress recovery

The stress recovery algorithm is employed in the sub-routine `computeStress`. Recalling the classical finite element formulation, the stress computation procedure begins with strain computations. Part of this routine is similar to the one employed for computing the stiffness matrices, described in the previous section, the only difference is that we are only interested in the necessary variables to compute the strains. This means reusing the most part of the `elementStiffness` routine, except the lines related with stiffness computation. As such, after the global displacement field is determined, the strain is computed at the element level, for each integration point, by means of:

```
1| e=B*U(indexB);
```

where `e` is the element strain field corresponding to the global degrees of freedom of the element stored in the vector `U(indexB)`. The stress parameters are then calculated at each integration point due to higher accuracy, applying the Hooke's law:

```
1| S(i,j,:)=D*e;
```

where the variable `S` is a multidimensional array with dimensions ($n_{\text{elem}} \times n_{\text{intpts}} \times 3$). The third dimension of the array refers to the number of stress components for a plane stress/strain problem, stored and ordered from the first stack to the third stack as follows: $\boldsymbol{\sigma}_{xx}$, $\boldsymbol{\sigma}_{yy}$ and $\boldsymbol{\tau}_{xy}$.

For visualization purposes, there is a special interest in representing the stress values in the nodal points because their representation is easier if they are associated to the nodal coordinates. However, the stresses computed at the same nodal point from adjacent elements are, generally, not the same because stresses are not required to be continuous in displacement-based finite elements. Therefore, some form of stress averaging has to be performed in order to improve the accuracy. The method used here is the extrapolation of the stress values from the integration points to the nodal points. The underlying concept is that the integration points are themselves the nodes of a inner-element, a so-called Gauss element, as shown in Fig. 4.2. The Gauss point numbering follows the element node numbering in the counterclockwise direction, with the point (i') being near the node (i) [11].

The advantage of this method is that the same shape functions that were used to interpolate a given value inside the element may now be used to extrapolate the values of stress components to outside the Gauss element. One only need to recall the Gauss

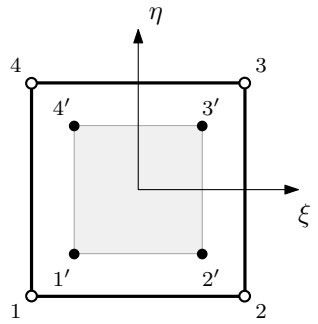


Figure 4.2: Idealized Gauss element for a bi-linear quadrilateral.

point coordinates from Table 2.1 and consider the following coordinate relations [11]:

$$\begin{cases} \xi = \dfrac{\xi'}{\sqrt{3}} \\ \eta = \dfrac{\eta'}{\sqrt{3}} \end{cases}, \qquad \begin{cases} \xi' = \xi\sqrt{3} \\ \eta' = \xi\sqrt{3} \end{cases}, \tag{4.11}$$

where (ξ',η') are the natural coordinates of the nodes of the Gauss element, which is also a four-node quadrilateral. Any given variable **w** whose values (w'_i) are known at the Gauss element nodes can be extrapolated to the element corners using the shape functions from Eq. (2.28), but now in terms of (ξ') and (η'), such that [11]:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1+0.5\sqrt{3} & -0.5 & 1-0.5\sqrt{3} & -0.5 \\ -0.5 & 1+0.5\sqrt{3} & -0.5 & 1-0.5\sqrt{3} \\ 1-0.5\sqrt{3} & -0.5 & 1+0.5\sqrt{3} & -0.5 \\ -0.5 & 1-0.5\sqrt{3} & -0.5 & 1+0.5\sqrt{3} \end{bmatrix} \begin{bmatrix} w'_1 \\ w'_2 \\ w'_3 \\ w'_4 \end{bmatrix}, \tag{4.12}$$

with the (w_i) being the values of **w** at the element corners. The sub-routine `computeStress` accomplishes this by first transposing the first two dimensions of the multidimensional array `S` by means of the command `permute` and applying Eq. (4.12) to each stack, as follows:

```
1  S=permute(S,[2 1 3]);
2
3  for i=1:size(S,3)
4
5     S(:,:,i)=[1+0.5*3^0.5 -0.5 1-0.5*3^0.5 -0.5;
6               -0.5 1+0.5*3^0.5 -0.5 1-0.5*3^0.5;
7               1-0.5*3^0.5 -0.5 1+0.5*3^0.5 -0.5;
8               -0.5 1-0.5*3^0.5 -0.5 1+0.5*3^0.5]*S(:,:,i);
9
10 end
```

Once the stress values are extrapolated there is the need to compute the mean value at each node. For the purpose, the routine needs to know what are the adjacent elements to a given node by applying the inverse connectivity in the form of a mask that can be applied to each stress component to determine its mean value from those points:

```
Sx=S(:,:,1)';
Sy=S(:,:,2)';
Sz=S(:,:,3)';
Svm=(Sx.^2+Sy.^2-Sx.*Sy+3.*Sxy.^2).^0.5;

meanStress=zeros(numNodes,4);

for i=1:numNodes

    mask=ismember(connectInfo,i);
    meanStress(i,:)=mean([Sx(mask) Sy(mask) Sz(mask) Svm(mask)],1);

end
```

The output is the `meanStress` variable, a matrix with as many lines as the total number of nodes of the problem with each column containing an averaged stress component, including the von Mises stress.

4.5.2 Results visualization

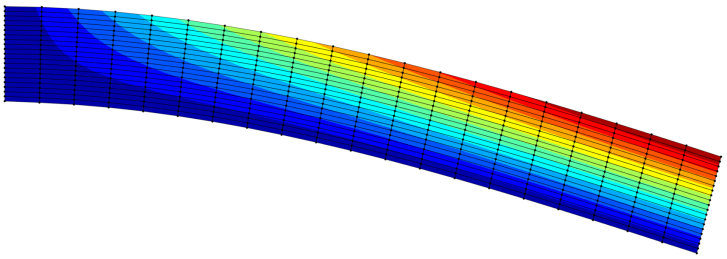
The post-processing part regarding results visualization and contour plotting is carried out by the sub-routine `postProcessing`. This function works in a way similar to the `nodeSelection` routine, but employs some different functionality. When an analysis is finished the user is presented with a representation of the deformed structure associated to a contour plot. This is achieved by means of the command `patch`, by summing the displacements to the nodal coordinates, in order to apply the deformation, and defining the parameter `'FaceVertexCData'` which will contain, in the following example, the horizontal displacement:

```
data=U(((1:numNodes)'-1)*2+1);
contour = patch(ax,'faces',connectInfo(:,2:end),...
             'vertices',[xData(:)+u1 yData(:)+u2],...
             'facecolor','interp','FaceVertexCData',data);
```

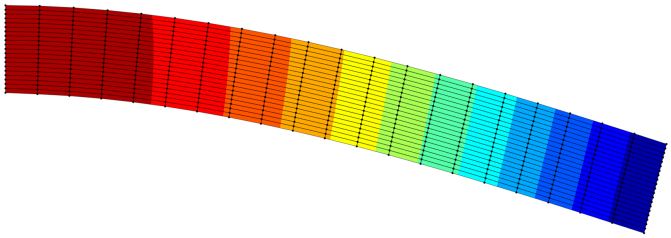
Before presenting the deformed structure there is the need to define a color map and associate it to a color bar, which will serve as the contour legend. This purpose is achieved with the commands `colormap` and `colorbar`:

```
c=colormap(ax,jet(12));
h=colorbar(ax,'Location','eastoutside');
```

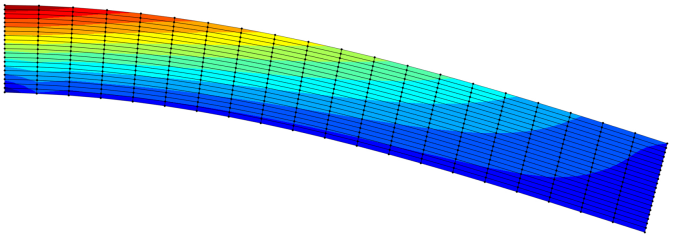
For instance, with these commands the sub-routine produces contour plots and presents deformed states as the ones shown in Fig. 4.3, for the case of the example from Section 5.5. The user also has a set of radiobuttons at his disposal to cycle through the desired contour plot. For the selected plot the user is allowed to probe the nodal values, by means of selecting the nodes with a brush, in a similar way as in the routine `nodeSelection`. When a set of nodes is selected, the parameter `'ActionPostCallBack'` of the brush object orders the execution of the callback function `onBrushData`. The routine extracts the brush data from the plot object called `nodes` and detects which plot contour the user had selected, by inspecting the corresponding radiobutton `'Tag'` parameter. The function then proceeds to populate a table with the variable `data` by means of the `uitable` command. In Listing 4.1 a code snippet that executes this sequence is shown.



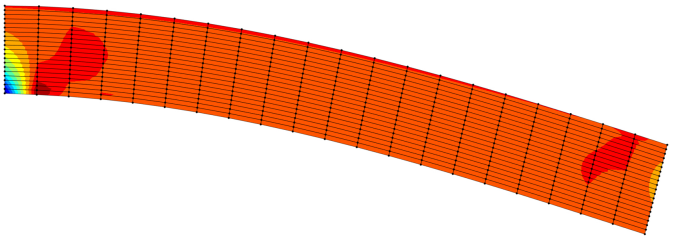
(a)



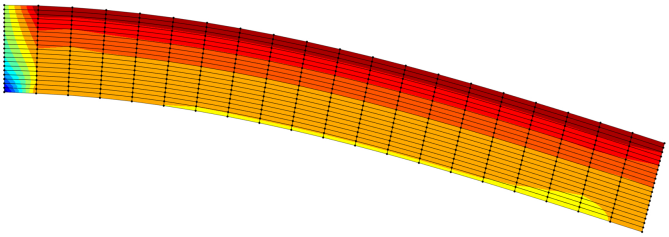
(b)



(c)

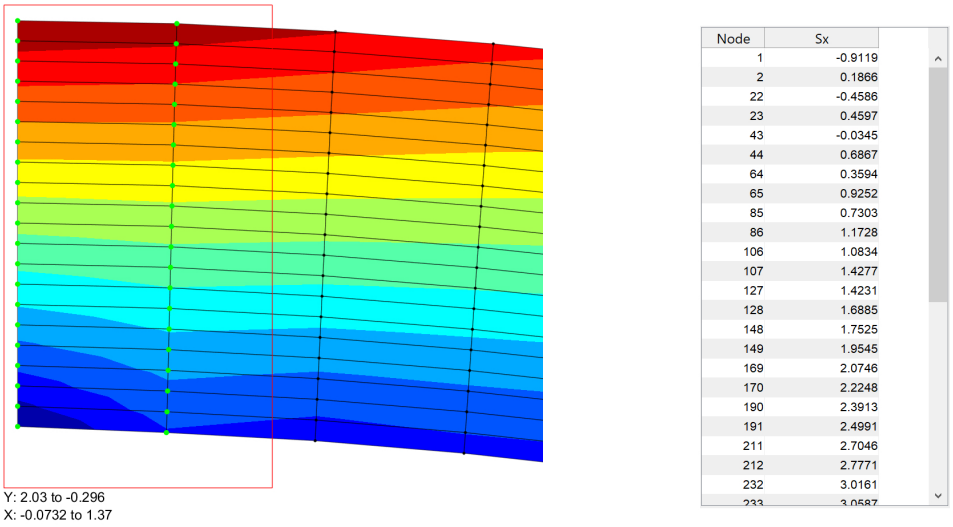


(d)



(e)

Figure 4.3: Visual representation of the deformed state for the example from Section 5.5. Countour plot representation for (a) the horizontal and (b) vertical displacement fields and the stress fields (c) $\boldsymbol{\sigma}_{xx}$, (d) $\boldsymbol{\sigma}_{yy}$ and (e) $\boldsymbol{\tau}_{xy}$.



(a) (b)

Figure 4.4: Probing node values by means of (a) the data brush tool and (b) table populated with the corresponding information, in this example the (σ_{xx}) stresses.

Listing 4.1: Code excerpt of the function *onBrushData*, providing node probing functionality and results representation in a table.

```
nodeLabels=(1:numNodes)';
selectedNodes=find(nodes.BrushData);

    switch bg.SelectedObject.Tag

        case '1'
                var=u1(selectedNodes);
            case '2'
                var=u2(selectedNodes);
            case '3'
                var=S(selectedNodes,1);
            case '4'
                var=S(selectedNodes,2);
            case '5'
                var=S(selectedNodes,3);
        case '6'
            var=S(selectedNodes,4);
    end

data=[nodeLabels(selectedNodes),var];
t =uitable(fPost,'Data',data);
```

Intentionally blank page.

Chapter 5

Benchmark analyses

In this chapter, a series of numerical examples is presented. All the examples were analyzed under states of plane stress or strain, assuming linear elastic and linear geometry behaviors. The mesh generation was limited to models with no more than one thousand nodes, a limit imposed by Abaqus student version used for pre-processing.

For problems in the incompressibility range, the hybrid versions of the elements available in Abaqus had to be used. According to Abaqus Theory Manual [34], these elements follow a mixed (u/p) formulation, where the displacement field is augmented with a pressure field. Still in the context of incompressibility, the Theory Manual also states that for the reduced integration formulation, Abaqus employs a selective integration, where reduced integration is used for the volumetric strain and full integration for the deviatoric strain, in what seems to be the formulation described in Section 3.4.1. In this case, the stresses and strains are calculated at the points that provide optimal accuracy, the so-called Barlow points that, according to Prathap [26], may or may not coincide with the Gauss points. The reduced integration elements also incorporate hourglass control mechanisms, whose formulation is described in more depth in Abaqus Theory Manual [34]. For all the analyses in this chapter, the reduced integration element from Abaqus was used with the default hourglass control option. The linear incompatible modes formulation used by Abaqus is based in the work of Simo and Rifai [32, 34].

The analyses were conducted with the goal of validating and assess the quality of implementation, by comparing the results coming from the developed program against the well-established Abaqus commercial software. The study was further extended to the evaluation of the performance of the finite element formulations presented in this work, for the compressible and incompressible cases.

Implemented finite element formulations

The finite element formulations implemented within the software developed during this Dissertation work:

- **Q4** - Classical four-node bi-linear quadrilateral finite element;
- **Q4SRI** - Selective reduced integration element, as described in Section 3.4.3;
- **Q6** - Incompatible modes element of Wilson [15];
- **QM6** - Generalized version of the incompatible modes Q6 element, as proposed by Taylor [15];
- **Q4/6I** - Incompatible modes element with six extra modes of deformation, as described in Section 3.6.4 [22];
- **Q4/4I** - Incompatible modes element with four extra modes of deformation, as described in Section 3.6.5 [22];
- **Qi5** - Compatible modes element with two extra modes of deformation, as described in Section 3.6.6 [6];
- **Qi6** - Compatible modes element with four extra modes of deformation, as described in Section 3.6.7 [6];

Abaqus finite element formulations

The finite element formulations available in Abaqus and used for the comparison analyses within the context of this work were:

- **CPS4** - Plane stress bi-linear quadrilateral from Abaqus;
- **CPE4H** - Abaqus hybrid plane strain four-node bi-linear quadrilateral with constant pressure;
- **CPE4RH** - Abaqus hybrid plane strain four-node bi-linear quadrilateral with reduced integration and constant pressure (allows hourglass control);
- **CPE4IH** - Abaqus hybrid incompatible modes plane strain four-node bi-linear quadrilateral with linear pressure.

In the following sections the results obtained using the implemented finite element formulations are presented and discussed, within the context of different numerical examples.

5.1 Rectangular plate with concentrated load

A rectangular thin-plate of thickness 1.0 mm is subjected to a concentrated in-plane load as depicted in Fig. 5.1. The structure is analyzed under plane stress or plane strain. It is considered that the concentrated load (F) acting on the plate has a magnitude of 100 N and the structure is made of a material with elastic properties: $E = 210$ GPa and $\nu = 0.3$.

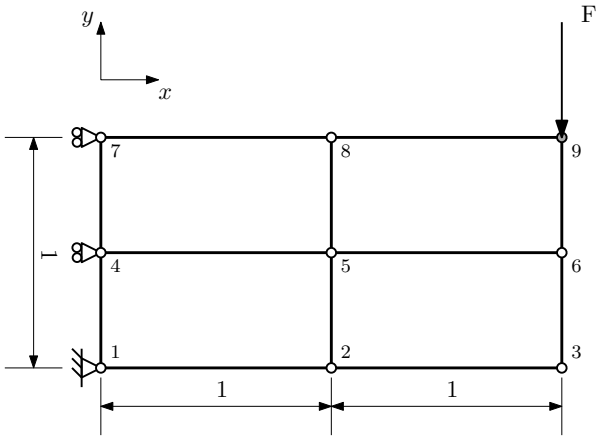
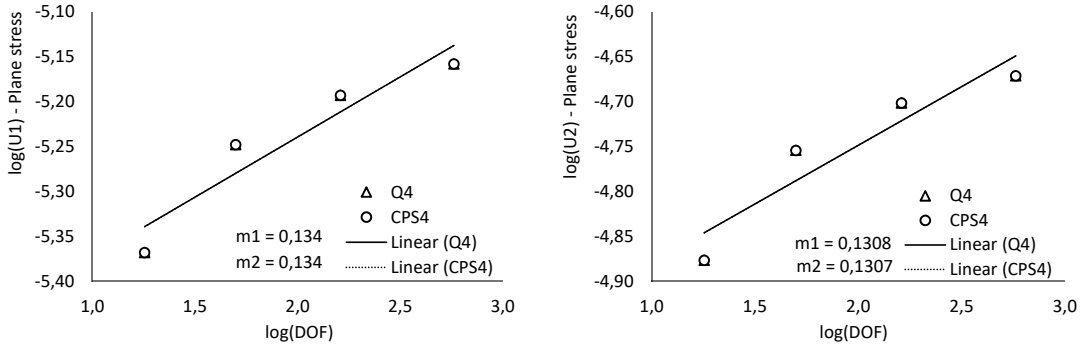


Figure 5.1: Thin rectangular plate subjected to concentrated load on its top-right corner. Structure dimensions are in meters.

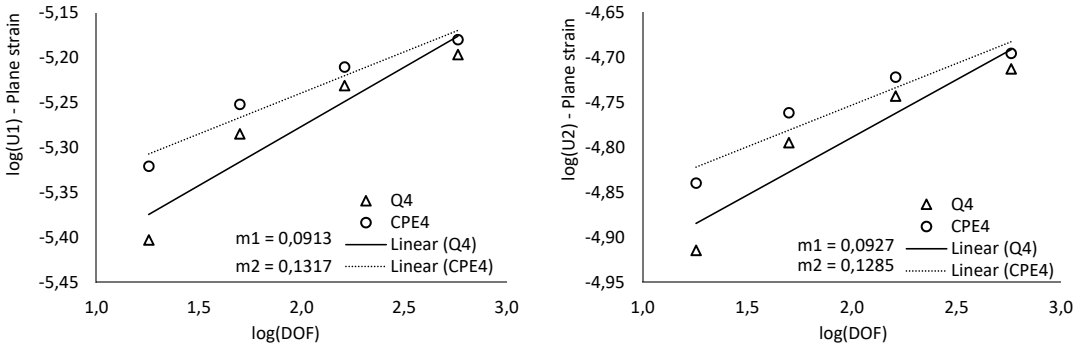
The original problem can be found in [36] using a finite element model based on a mesh of 2×1 elements. For this work, the problem was adapted to be analyzed using square meshes starting from two elements per side to a maximum of sixteen elements per side. The aim of this test was to evaluate the quality of the results regarding the displacements obtained with the developed program and validate them by taking the results obtained with Abaqus as reference. For this purpose, the horizontal and vertical displacements (U1, U2) of the top-right corner of the plate were evaluated. For ease of comparison, the values were logarithmized and plotted against the logarithm of the number of degrees of freedom (DOF) in order to quantify the mesh refinement. The results are shown in Figs. 5.2 and 5.3 for plane stress and strain, respectively.

Analyzing the data, we observe that the trendlines have similar slopes for the case of plane stress, indicating that our implementation of the Q4 element under plane stress was able to perform at the same level of the CPS4 element from Abaqus. However, a disparity is observed for the case of plane strain using coarse meshes. In this context, our implementation of the Q4 element revealed a stiffer behavior, when compared to the CPE4 element from Abaqus. Nonetheless, the results start to converge with the increase of degrees of freedom. From the data presented in Table 5.1 it can be seen that the relative deviation in respect to the CPE4 element rapidly decreases when doubling the number of elements per side, but finer meshes would be needed to confirm if the deviation would decrease. Nonetheless, for meshes using eight or sixteen elements per side, the deviation is already within an acceptable range. The disparity is attributed to the fact that Abaqus uses a different elasticity tensor for plane strain analysis, according to several discussions in some online scientific forums.



(a) (b)

Figure 5.2: Linearized results obtained for (a) the horizontal and (b) vertical displacements of the top-right corner node of the beam under plane stress.



(a) (b)

Figure 5.3: Linearized results obtained for (a) the horizontal and (b) vertical displacements of the top-right corner node of the beam under plane strain.

Table 5.1: Relative deviation of the results for the horizontal and vertical displacements using different mesh sizes.

Elements per side	**Relative error** (%)	
	U_1	U_2
2	17.23	15.84
4	7.29	7.39
8	4.63	4.74
16	3.75	3.92

5.2 Rectangular plate with distributed load

A rectangular thin-plate of thickness 25 mm is subjected to a distributed load in one side as depicted in Fig. 5.4. The structure is analyzed under a plane stress state and condidered to be made of a material with elastic properties: $E = 210$ GPa and $\nu = 0.3$. Considering linear elastic material and geometry, the configuration is equivalent to a uniaxial tensile test, where necking of the structure is expected to occur due to the Poisson's effect.

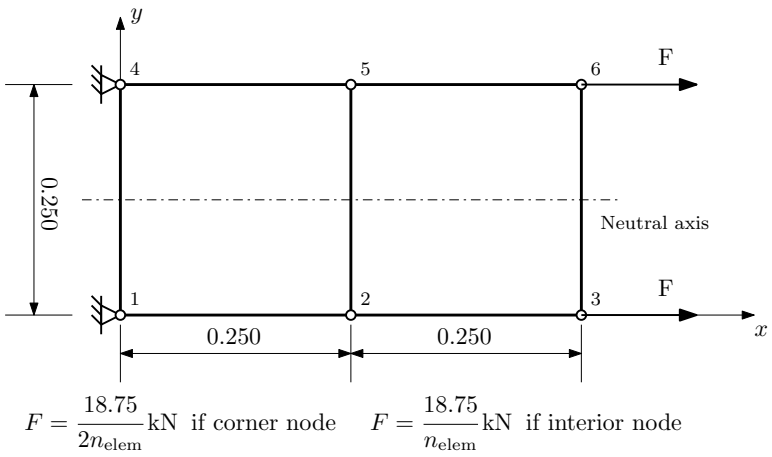
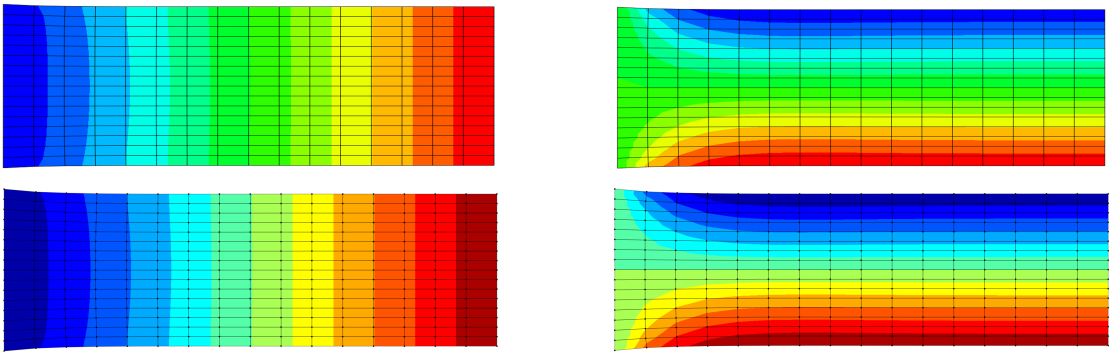


Figure 5.4: Thin rectangular plate subjected to distributed load on its right side (structure dimensions are in meters).

The original problem can be found in [16], using a mesh of 2×1 elements. However, the analysis was carried out using a square mesh of 16 elements per side, using the CPS4 and Q4 elements. The main purpose of this test was to assess the accuracy of the displacement and stress fields produced by the developed program. A visual comparison of the displacements and stress contour plots is established in Figs. 5.5 and 5.6, respectively.

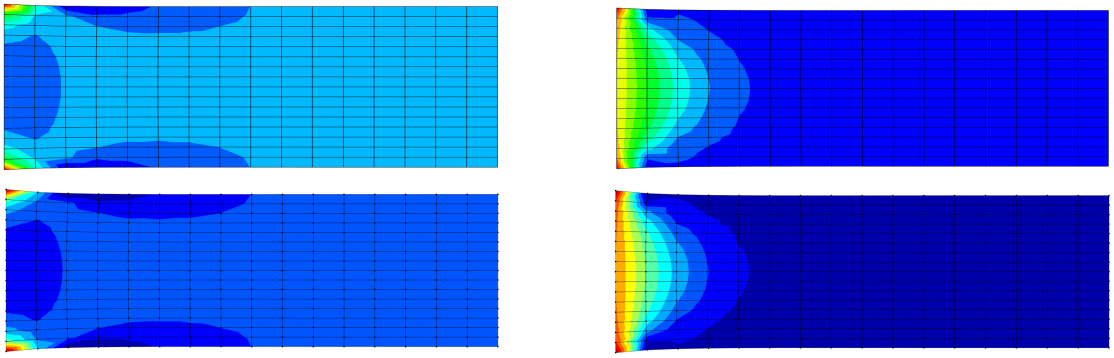


(a) (b)

Figure 5.5: Contour plots of (a) the horizontal and (b) vertical displacement fields (top images come from Abaqus, bottom images come from the developed program).

Regarding the displacement fields we observe the contours are very similar between the two programs, indicating that the displacements are calculated correctly and the

post-processing module is capable of reproducing the correct distribution. In the case of the vertical displacements (see Fig. 5.5b) the contour clearly depicts the Poisson's effect, which results in a symmetry of the displacement field in relation to the neutral axis.



(a) (b)

Figure 5.6: Contour plots of the normal stress fields (a) $\boldsymbol{\sigma}_{xx}$ and (b) $\boldsymbol{\sigma}_{yy}$ (top images come from Abaqus, bottom images come from the developed program).

Observing Fig. 5.6, we conclude that the developed program is capable of generating stress distributions similar to Abaqus. This means the stress recovery algorithm is well implemented with the extrapolation of the stress components computing the correct values. This is further validated by the excellent stress results presented in Table 5.2, measured along the bottom-half of the left side of the plate. Due to the symmetry of displacements the stress field is also symmetrical in relation to the neutral axis.

Table 5.2: Stress results, in MPa, measured along the left bottom-half of the plate.

Nodes	**Developed code**			**Abaqus**		
	σ_{xx}	σ_{yy}	τ_{xy}	σ_{xx}	σ_{yy}	τ_{xy}
1	3.972	1.192	0.789	3.972	1.192	0.789
18	3.340	1.002	0.590	3.339	1.002	0.590
35	3.124	0.937	0.453	3.124	0.937	0.453
52	3.023	0.907	0.350	3.023	0.907	0.350
69	2.970	0.891	0.265	2.970	0.891	0.265
86	2.940	0.882	0.191	2.940	0.882	0.191
103	2.923	0.877	0.124	2.923	0.877	0.124
120	2.914	0.874	0.061	2.914	0.874	0.061
137	2.912	0.873	0.000	2.911	0.873	0.000

For this case, the stress recovery algorithm needs to be revised in order to accommodate the necessary changes to compute the stress parameters based in the enhanced strain field, as described in Section 3.6.1.

5.3 Infinite plate with circular hole

An infinite plate with a circular hole is subjected to a uniform in-plane traction on both ends. The exact solution is obtained by employing the Kirsch equations such that [5]:

$$\sigma_{rr}(r,\theta) = \frac{T_x}{2}\left(1 - \frac{R^2}{r^2}\right) + \frac{T_x}{2}\left(1 - 4\frac{R^2}{r^2} + 3\frac{R^4}{r^4}\right)\cos 2\theta\,, \tag{5.1}$$

$$\sigma_{\theta\theta}(r,\theta) = \frac{T_x}{2}\left(1 + \frac{R^2}{r^2}\right) - \frac{T_x}{2}\left(1 + 3\frac{R^4}{r^4}\right)\cos 2\theta\,, \tag{5.2}$$

$$\tau_{r\theta}(r,\theta) = -\frac{T_x}{2}\left(1 + 2\frac{R^2}{r^2} - 3\frac{R^4}{r^4}\right)\sin 2\theta\,, \tag{5.3}$$

where (T_x) is the magnitude of the applied stress for the infinite plate case, (R) is the radius of the hole and (r) is the radial coordinate of a given arbitrary point $P(r,\theta)$ in the plate. The stress tensor can further be transformed from the polar coordinate system to the Cartesian coordinate system as follows [37]:

$$\begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \tau_{xy}\end{Bmatrix} = \begin{bmatrix}\cos^2\theta & \sin^2\theta & -\sin^2 2\theta\\ \sin^2\theta & \cos^2\theta & \sin 2\theta\\ \sin\theta\cos\theta & -\sin\theta\cos\theta & \cos 2\theta\end{bmatrix}\begin{Bmatrix}\sigma_{rr}\\ \sigma_{\theta\theta}\\ \tau_{r\theta}\end{Bmatrix}. \tag{5.4}$$

The stress (T_x) has magnitude 10 and the hole has a radius (R) measuring 1. The structure is considered to be made of linear elastic material with Young's modulus (E) of 10^5 and Poisson's ratio (ν) of 0.3. This problem may be simplified so that a quarter plate need only be analyzed, following the configuration shown in Fig. 5.7a [5, 37].

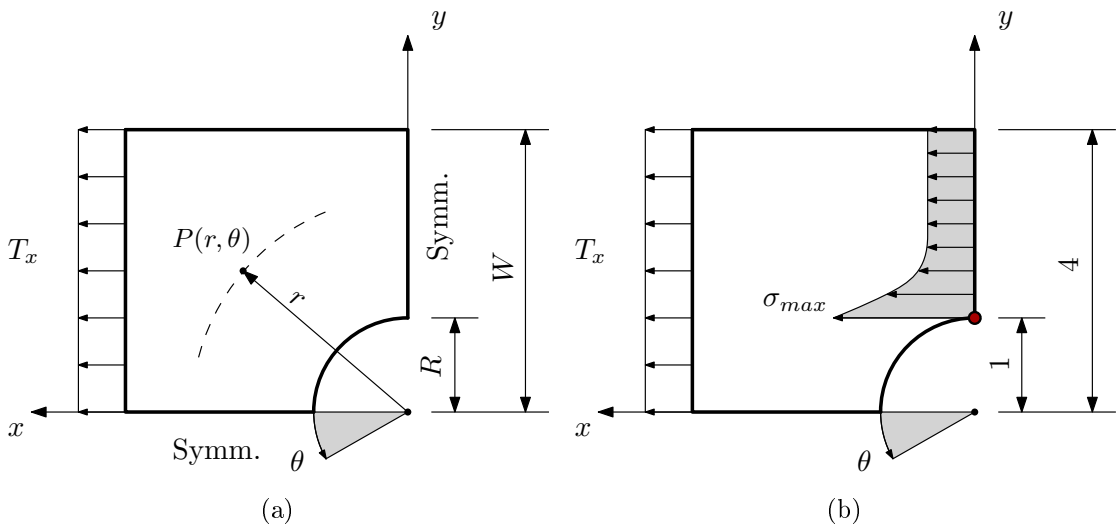


Figure 5.7: Elastic plate with circular hole: problem definition for $R/W = 0.25$.

The stress concentration is measured at the edge of the hole, where (see Fig. 5.7b):

$$r = R \;\wedge\; \theta = \frac{3}{2}\pi\,. \tag{5.5}$$

For this location, Kirsch's solution contains the well known factor-of-three stress concentration for the infinite plate under uni-axial loading, with the radial and shear stresses (σ_{rr}) and ($\tau_{r\theta}$) equal to zero and the hoop stress ($\sigma_{\theta\theta}$) respecting the following relation [5]:

$$\sigma_{\theta\theta} = \sigma_{xx} = \sigma_{\max} = 3\,T_x\,, \tag{5.6}$$

effectively demonstrating a stress concentration factor (K_t) of 3. However, the quarter plate has a finite width and an additional term needs to be considered: the nominal stress (σ_{nom}), which is the average stress at the hole due to the reduction in cross-section, related to (T_x) as follows [30]:

$$\sigma_{\text{nom}} = T_x\,\frac{W}{W-R}\,. \tag{5.7}$$

In this case, the stress concentration factor will be a fraction of the original analytical solution and is defined in terms of the width (W) of the plate and the radius (R) of the hole by the empirical relation [30]:

$$K_t = 3 - 3.14\left(\frac{R}{W}\right) + 3.667\left(\frac{R}{W}\right)^2 - 1.527\left(\frac{R}{W}\right)^3\,, \tag{5.8}$$

resulting in the graphical representation depicted in Fig. 5.8. The maximum stress at the edge of the hole may then be obtained as follows [30]:

$$\sigma_{\max} = K_t\,\sigma_{\text{nom}}\,. \tag{5.9}$$

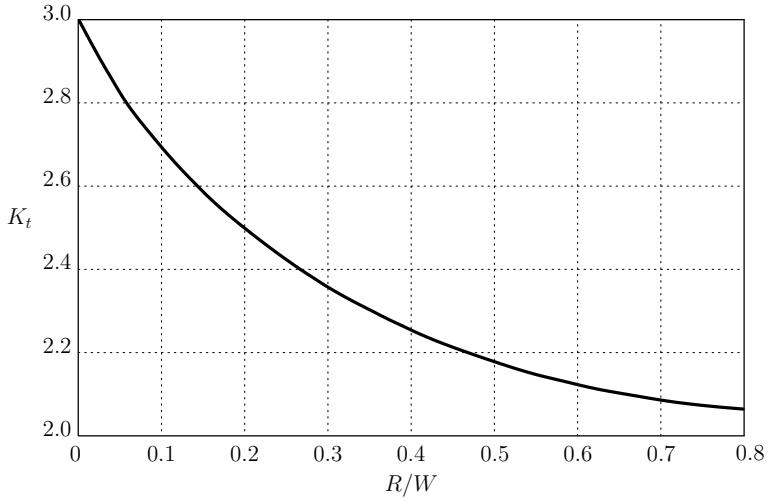


Figure 5.8: Stress concentration factor K_t in terms of the ratio R/W (adapted from [30]).

The quarter plate was analyzed under plane strain conditions, using the mesh shown in Fig. 5.9 for the case of R/W=0.25, with the necessary symmetry boundary conditions

being applied to the right and bottom edges. The values obtained for the normal stress (σ_{xx}) at the edge of the hole, the relative error and the analytical solution, according to [30], for this particular problem are shown in Table 5.3.

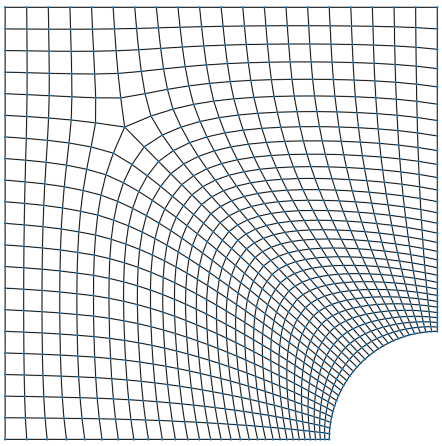


Figure 5.9: Elastic plate with circular hole: mesh definition for $R/W = 0.25$.

Table 5.3: Normal stress σ_{xx} at the edge of the hole and relative error with respect to the analytical solution for $R/W = 0.25$.

Element	$\boldsymbol{\sigma_{xx}}$	**Relative error**	**Analytical solution**
Q4	36.5554	13.27%	
Q4SRI	34.5313	7.00%	
Q6	35.1586	8.95%	$\sigma_{\text{nom}} = 13.333$
QM6	34.2934	6.27%	
Q4/4I	35.1586	8.95%	$\sigma_{\text{max}} = 32.271$
Qi5	32.4032	0.41%	
Qi6	32.0781	0.60%	$K_t = 2.42$
CPE4H	34.6948	7.51%	
CPE4RH	33.9036	5.06%	
CPE4IH	36.0201	11.62%	

Results show that the classical formulation produces the higher stress value corresponding to an error of 13.27%. The compatible mode elements Qi5 and Qi6 were capable of attaining excellent results here corresponding to an error less than 1%, with the first showing the smallest error. The remaining elements registered errors between 6% and and 9%, with the Q6 and Q4/4I elements obtaining exactly the same values. The element Q4/6I was excluded from the analysis due to low quality results. The deformed configuration of the quarter plate with the representation of the displacement and stress distributions, using the Qi5 element, are shown in Fig. 5.10. Regarding Abaqus' formulations, the incompatible mode element CPE4IH produces the highest stress value, corresponding to an error of 11.62%, on the other hand the selective integration element CPE4RH shows the smallest error.

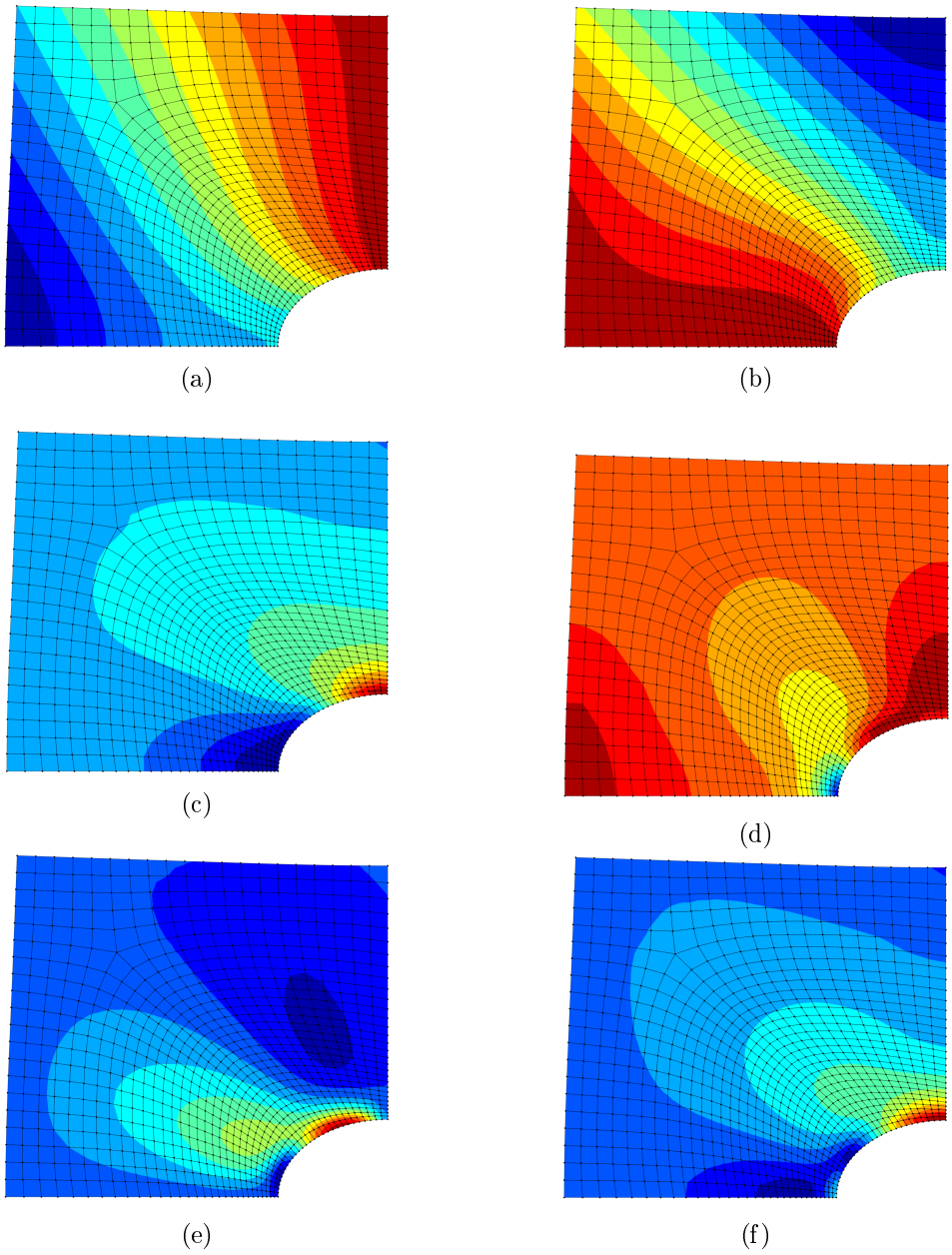


Figure 5.10: Deformed configuration of the quarter plate for $R/W = 0.25$ using the Qi5 element, with contour plots of: (a) horizontal displacement, (b) vertical displacement, (c) normal stress σ_{xx}, (d) normal stress σ_{yy}, (e) shear stress τ_{xy} and (f) von Mises stress.

5.4 One-element test

A squared structure with a side dimension of 2 units is subjected to two different configurations of loads and boundary conditions, as depicted in Fig. 5.11. The structure is discretized using a single-element mesh and the nodal forces have magnitude 1000. The material is considered linear elastic, with a Young's modulus of 5000 units. The analysis is carried out for both configurations, considering compressible ($\nu = 0.3$) and incompressible material ($\nu = 0.4999999$).

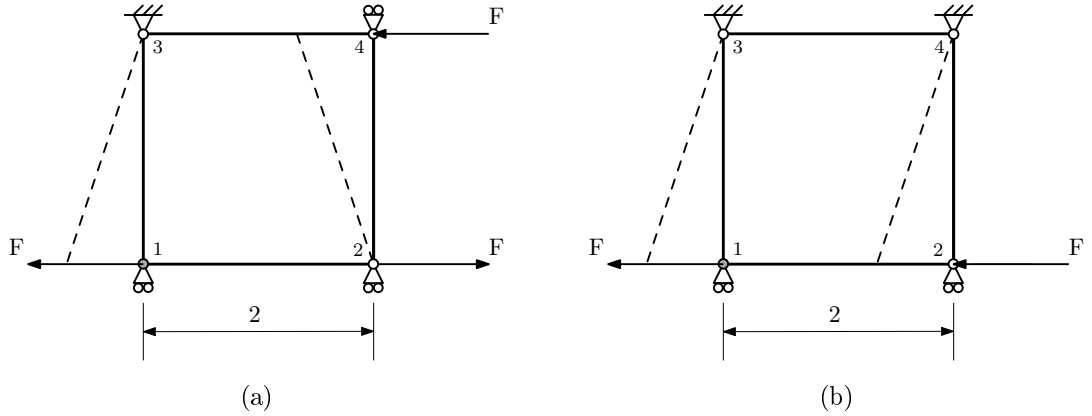


Figure 5.11: Different configurations of loads and boundary conditions.

The test is used by Natal Jorge [22] to show the effects of locking and effectively compare the stiffness of different element formulations. For configuration (a), it is expected that nodes 1 and 4 will have the same horizontal displacement, while node 2 will not move. Additionally, for configuration (b), nodes 1 and 2 should have the same horizontal displacement. The results are presented in Table 5.4 for all the element formulations implemented in this work and three elements from Abaqus. The element CPE4RH marked with (*) refers to the reduced integration element with hourglass control option set to "enhanced".

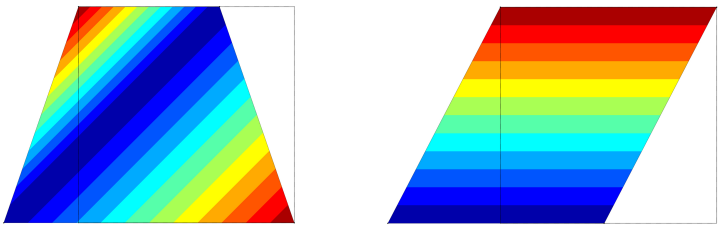
Referring to the compressible case, all the elements show the same results when employing configuration (b), except the selective integration element Q4SRI. However, the same does not apply for configuration (a). In this context, although the effects of locking are not noticeable, it can be seen that the Q4 and Q6 elements are stiffer than the rest. Between these two formulations, one can clearly see the advantageous effect of the incompatible modes on allowing the Q6 element to deform better than the classical bi-linear formulation. The deformed states of the Q4 element, in the compressible case, for both configurations are depicted in Fig. 5.12.

Still in the context of the compressible case, the square geometry effectively eliminates the effect of the coordinate mapping by the jacobian matrix. Therefore, it was expected for the Q6 and QM6 elements to originate the same results. The selective integration elements Q4SRI and CPE4RH are capable of sustaining the highest degrees of deformation, with the latter showing much larger values, which may indicate a tendency for the CPE4RH to originate spurious deformation. Switching the hourglass control option of the CPE4RH element to "enhanced" avoids this problem and allows the element to obtain the same results as the incompatible and compatible mode formulations.

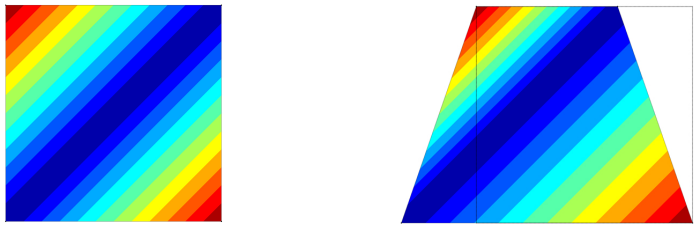
Table 5.4: Horizontal displacement at node 1.

Element	Configuration (a)		Configuration (b)	
	$\nu = 0.3$	$\nu = 0.4999999$	$\nu = 0.3$	$\nu = 0.4999999$
Q4	0.69333	7.2E-7	1.04	1.2
Q4SRI	2.2797	2.1294	1.1721	1.2
Q6	0.82588	0.6	1.04	1.2
QM6	1.092	0.9	1.04	1.2
Q4/6I	1.0833	0.9	1.04	1.2
Q4/4I	1.092	0.9	1.04	1.2
Qi5	1.0029	0.9	1.04	1.2
Qi6	1.092	0.9	1.04	1.2
CPE4H	1.56	1.8	1.04	1.2
CPE4RH	416	480	1.04	1.2
CPE4RH*	1.092	0.9	1.04	1.2
CPE4IH	1.092	0.9	1.04	1.2

Analyzing the results for the incompressible case, all the elements show the same values for configuration (b). Employing configuration (a), locking of the Q4 element is evident, effectively demonstrating the inability of the classical formulation to perform well in such restricted cases (see Fig. 5.12c). Both incompatible and compatible modes formulations have no issues in this case, although the Q6 element is slightly stiffer (see Fig. 5.12d). Worthy of reference is the difference between the Q4 and CPE4H elements, with the latter demonstrating the effectiveness of the mixed (u/p) formulation on avoiding locking.



(a) (b)



(c) (d)

Figure 5.12: Deformation using the Q4 element in the compressible case, for both test configurations (top); deformation using (c) the Q4 element and (d) the QM6 element, for $\nu = 0.4999999$ (configuration of Fig. 5.11a).

5.5 Beam under bending

This numerical example is used by Malkus and Hughes [15, 19] and also by César de Sá and Natal Jorge [6, 22], using a finite element mesh of (4×8) elements in order to study the performance of different element formulations under bending, in the compressible and incompressible cases.

A rectangular beam is subjected to a distributed load at the free-end. The structure has dimensions (16×4) and only one half need be modelled since the x-axis is a line of anti-symmetry. A representative discretization of the domain is depicted in Fig. 5.13 using a (4×4) elements model. The problem is analyzed in plane strain, respecting the following boundary conditions:

- displacements:

$$\begin{cases} u(0,0) = v(0,0) = 0 \\ u(0,\pm c) = 0 \end{cases} \tag{5.10}$$

- traction:

$$\sigma_{yy}(x,\pm c) = \tau_{xy}(x,\pm c), \qquad x \in [0, L] \tag{5.11}$$

$$\left.\begin{aligned} \sigma_{xx}(L,y) &= 0 \\ \tau_{xy}(L,y) &= \frac{3}{4c^3}(c^2 - y^2) \end{aligned}\right\}, \qquad y \in \,]-c, +c[\tag{5.12}$$

$$\left.\begin{aligned} \sigma_{xx}(0,y) &= -\frac{3L}{2c^3}y \\ \tau_{xy}(0,y) &= \frac{3}{4c^3}(c^2 - y^2) \end{aligned}\right\}, \qquad y \in \,]-c, 0[\cup]0, +c[\tag{5.13}$$

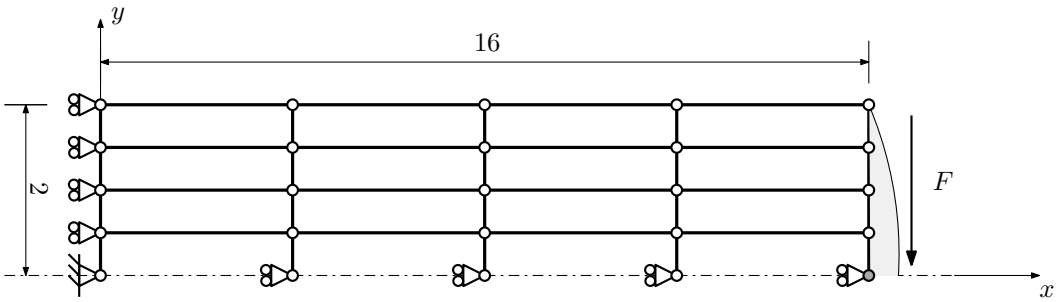


Figure 5.13: Finite element model of the beam under bending.

In this work, the problem was analyzed using square meshes starting from two elements per side to a maximum of thirty elements per side. Convergence of the vertical displacement at the tip bottom node was evaluated. Similar to Natal Jorge [22], the

vertical displacement at the reference node was also evaluated in terms of the following relation between the Lamé parameters:

$$\log\left(\frac{\lambda}{\mu}\right) = \log\left(\frac{2\nu}{1-2\nu}\right), \tag{5.14}$$

from where it's possible to retrieve the Poisson's ratio ν. Natal Jorge advances that the analytical solution for the vertical displacement at the reference node with coordinates $(16, 0)$ is given by:

$$v(16,0) = \frac{(1-\nu^2)L^3}{2Ec^3}\left(\frac{c^2}{2L^2}\left(4+5\frac{\nu}{1-\nu}\right)+1\right), \tag{5.15}$$

with the Young's modulus (E) of the material given by:

$$E = \frac{(1-\nu^2)L^3}{2c^3}\left(\frac{c^2}{2L^2}\left(4+5\frac{\nu}{1-\nu}\right)+1\right). \tag{5.16}$$

The results obtained for the vertical displacement at the reference node are presented in Fig. 5.14 for different mesh sizes and in Fig. 5.15, in terms of $\log(\lambda/\mu)$ for the elements Q4, QM6, Qi6, CPE4H and CPE4IH. For the compressible case, we observe that all formulations tend to converge to the analytical solution. The elements Q4SRI, Q4 and Qi5 show weak results using coarse meshes, with the selective integration element evidencing the same overrelaxed behavior from the previous example. Finer meshes contribute to stabilize the Q4SRI, however, it seems that a further mesh refinement would still be needed in order to confirm full convergence. Worthy of reference is the excellent behavior of the compatible and incompatible modes elements (Qi6, Q4/6I, Q4/4I, QM6 and Q6), effectively demonstrating the advantages of the enhanced strain formulation in a bending-dominated problem. The use of structured meshes of rectangular elements renders the Q6 and QM6 elements the same results, as it would be expected, contrarily to the previous example. The compatible modes Qi5 element does not behave well using coarse meshes, attaining the same results as the classical Q4 formulation, a rather poor performance for an enhanced strain formulation, albeit already predicted by Natal Jorge [22]. For the incompressible case, the Q4 element converges much slower than other formulations and shows the weakest results using coarse meshes, mainly due to the combined effects of locking and spurious shear response of bending. The enhanced strains formulations continue to reproduce the excellent results already verified for the compressible case, with the exception of the Qi5 element.

Concerning the data obtained with Abaqus, all the formulations converge to the solution, attaining essentially the same results for both cases. Here, the element CPE4H clearly outperforms the classical bi-linear quadrilateral, with the mixed (u–p) formulation playing an important role in eliminating locking in the incompressible case, although it still suffers from the same spurious shear response of its counterpart due to bending. It can be seen that the results for incompatible modes CPE4IH are equivalent to the implemented enhanced strain elements, as already confirmed in previous example. The CPE4RH effectively outperforms the Q4SRI in both cases by showing a surprisingly stable behavior, especially after the spurious response obtained in the previous example without hourglass control ("default" option enabled).

Analyzing the data from Fig. 5.15, a noticeable degration of the classical bi-linear quadrilateral starts to occur for values of $\log(\lambda/\mu)$ higher than 1. The enhanced strain elements QM6 and Qi6 are capable of sustaining the excellent results from previous tests through the most part of the incompressible range, for values of $\log(\lambda/\mu)$ less than 10. Beyond this limit, both formulations start to become unstable. These results are in line with the ones already obtained by Natal Jorge [22]. The CPE4H and CPE4IH show consistent results through the whole incompressibility range with no signs of instabilities, due to the effects of the mixed formulation.

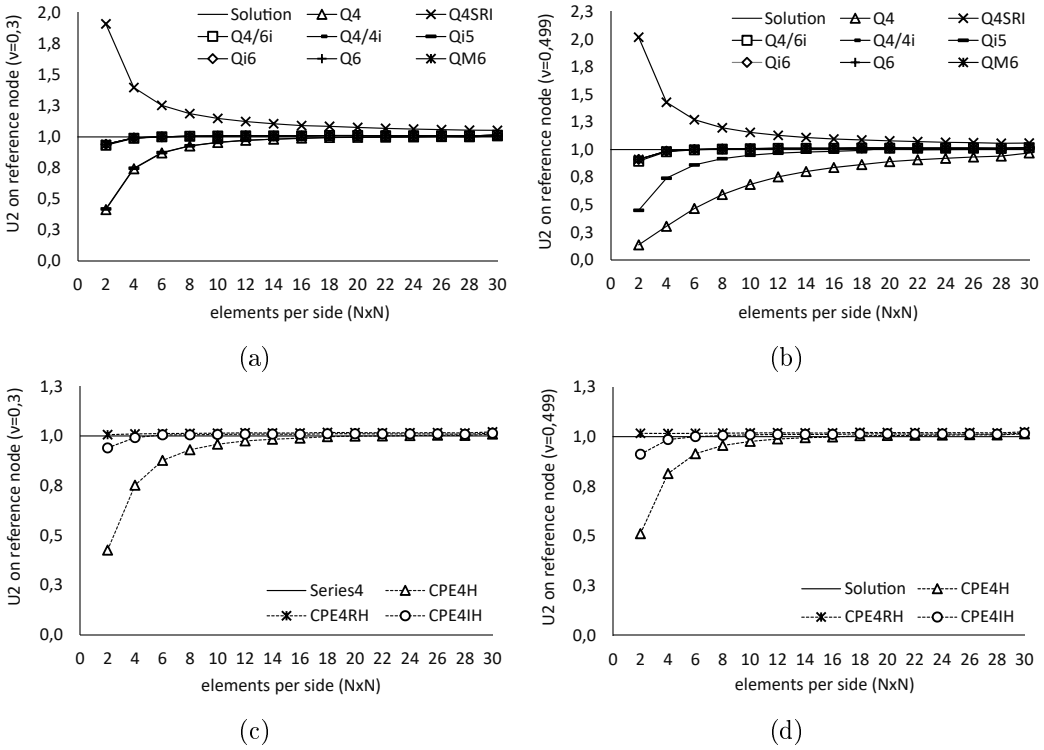


Figure 5.14: Results of the mesh convergence analysis for the bending problem, using the implemented formulations (top) and those from Abaqus (bottom).

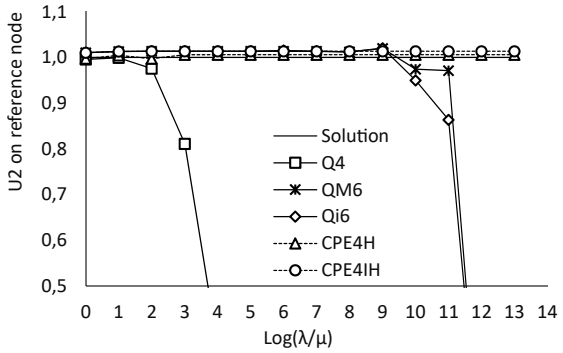


Figure 5.15: Vertical displacement at the reference node for different values of $\log(\lambda/\mu)$.

5.6 Cook's membrane

The Cook's membrane problem has been widely used in literature to test nearly incompressible formulations under combined bending and shear [2, 6, 9, 22, 24, 25, 29, 38]. All the authors use the same geometry, but consider different load conditions and material parameters. The variable of interest is the vertical displacement of the top right corner of the free-end. Convergence of this value as a function of the number of elements per side is usually used as a criterion to assess the robustness of a given finite element formulation against the skewed topology.

A trapezoidal panel is clamped on one side and subjected to a uniform in-plane shear load on the free-end side. The geometry, loading and boundary conditions are given in Fig. 5.16, for a representative (2×2) elements model. The problem was analyzed in plane strain using the same elastic properties as Natal Jorge [6, 22]: $E = 240.565$ and $\nu = 0.4999$. The results for the vertical displacement at the reference node are plotted in Fig. 5.17 with mesh sizes varying from two elements per side to thirty elements per side, for all formulations, except the incompatible modes Q4/6I element which demonstrated very weak results. In his thesis, Natal Jorge [22] presents results for this problem with increasing mesh sizes until a maximum size of (65×65) elements.

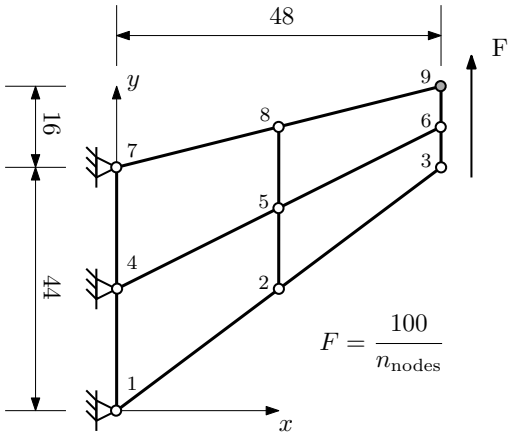
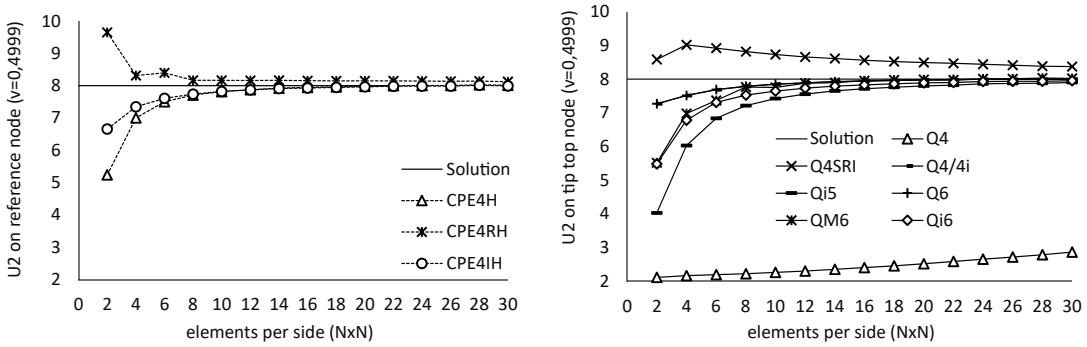


Figure 5.16: Cook's membrane problem: geometry, boundary and load conditions.

Analysing the data plotted in Fig. 5.17b, we observe that the classical bi-linear quadrilateral formulation does not converge to the solution, at least for the maximum mesh size considered in this work. Natal Jorge [22] also confirms the poor performance of the Q4 element using finer meshes. In this example, the incapacity of the Q4 to deform is aggravated because, apart from having to deal with the effects of locking due to bending and incompressibility, the effects of mesh distortion also come into play.

The selective integration element Q4SRI, once again, shows its overrelaxed behavior using coarse meshes, albeit in a smaller extent. The stabilizing effect of mesh refinement is not as evident in this case as in the previous example, further refinement would be needed to confirm converge of the Q4SRI element. Natal Jorge's [22] analysis of the Q4SRI element, shows higher quality results than the ones obtained in the present work, an indication that this implementation may have to be corrected. The compatible modes element Qi5 continues to show the weakest results from all the enhanced strain formulations, considering smaller mesh sizes. The QM6 and Qi6 elements manage to perform

just a little better than the Qi5. The elements Q4/4I and Q6 show excellent results here, although it was expected for the Wilson's formulation not to perform better than its counterpart revised by Taylor, the QM6, due to the skewed geometry. Distortion tests were performed to evaluate this issue.



(a) (b)

Figure 5.17: Vertical displacement at the reference node for increasing mesh sizes.

Concerning the formulations from Abaqus (Fig. 5.17a), unsurprisingly all the elements converge to the solution. The CPE4RH initially shows a spurious response, but rapidly stabilizes and manages to outperform the Q4SRI. The CPE4RH was used here with the default hourglass control option enabled. Both the CPE4H and CPE4IH have similar performances in this case, mainly due to the advantages of using a mixed formulation in a incompressible problem. The incompatible modes formulation shows slightly better results in coarse meshes due to the beneficial effect of the enhanced strain field in bending.

A computational cost analysis was performed for the Cook's membrane problem using three finite element formulations and increasing mesh sizes. The elapsed time of the main routine `planeStress` was measured by means of the `tic/toc` commands in MATLAB. Although this kind of analysis is not precise, the objective was demonstrate the difference in execution time between the classical formulations and the enhanced strain formulation. The results are plotted in Fig. 5.18 for the main routine and the machine specifications are presented in Table 5.5. It can be seen that increasing the number of elements, the elapsed time does not increase linearly. The performance is directly proportional to the square of the size of the input data set. Following the Big O notation used in Computer Science to describe the performance of an algorithm, this is common with algorithms that involve nested iterations over the data set. That is the case of the assembly and stress calculation algorithms. As expected, the enhanced strain formulation is computationally more demanding due to the additional degrees of freedom that require more stiffness matrices to be calculated and additional operations in order to calculate the stresses. The classical formulation and selective integration elements show similar results. The assembly and stress calculation routines were also evaluated and each of them consume approximately 50% of the total execution time.

Table 5.5: Specifications for the machine running the computational analysis.

Processor	Intel i5-4690K @ 3.50 GHz
Memory	2x8 GB DDR3 @ 1600MHz Dual-Channel
Graphics	GeForce GTX 980M 8GB
Storage	256GB SSD

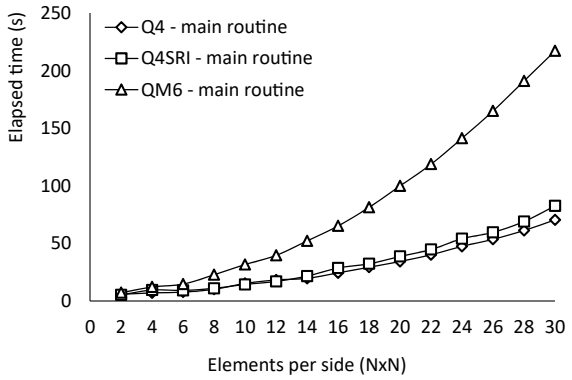


Figure 5.18: Execution time for the Cook's membrane analysis.

5.7 Distortion test

The example is used by Natal Jorge [22] to evaluate the robustness of a given finite element formulation to distortion. The problem was used here to compare the classical bi-linear quadrilateral with the compatible modes Qi6 element and to evaluate the quality of the implementation of the Wilson-Taylor incompatible modes element QM6 against its predecessor, the Q6 element.

A square structure is discretized using a (2×2) square-element mesh and is analyzed under a plane strain state using a material with elastic parameters: $E = 5000$ and $\nu = 0.3$. The geometry, load and boundary conditions are presented in Fig. 5.19. The distortion is imposed by translating the central node point (d) units into one of the four quadrants, as also depicted in Fig. 5.19. The horizontal displacement at nodes 1 and 9 is evaluated and the error for the corresponding values is calculated in relation to the values obtained for the undistorted configuration. The results are plotted in Fig. 5.20, where the error related to node 9 being represented in the negative axis for the ease of representation.

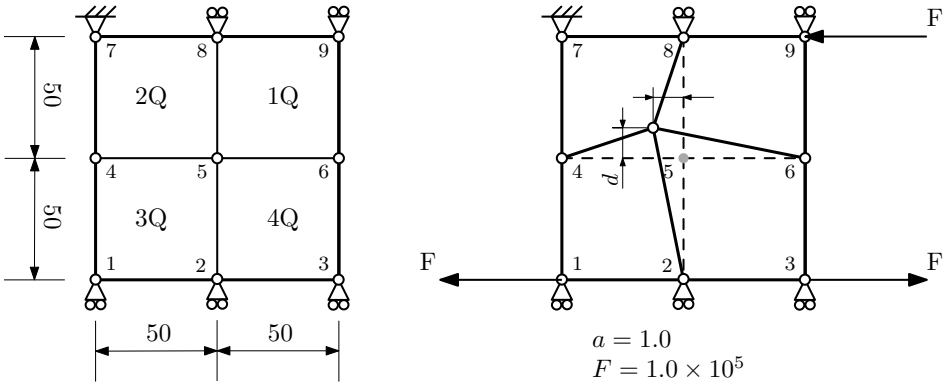


Figure 5.19: Finite element model used for the distortion test (left) and geometry, load and boundary conditions (right).

Analyzing the results plotted in Fig. 5.20, we conclude that the classical bi-linear quadrilateral element has a higher sensitivity to distortion than the compatible modes formulation. For the cases were the central nodal point is translated to the first or fourth quadrant, both formulations show similarly good results at node 1, however the performance of the Q4 element rapidly degrades for higher degrees of distortion. The Qi6 element behaves in a consistent manner through all the cases.

Concerning the incompatible modes formulations of Wilson and Taylor, we can see that the original Q6 formulation performs much better than its revised version for all cases. Additionally, the QM6 shows an erratic behavior with the error not increasing steadily through the analyses. The results obtained here for the QM6 element are not entirely in line with the ones obtained by Natal Jorge [22], indicating this implementation may have to be corrected.

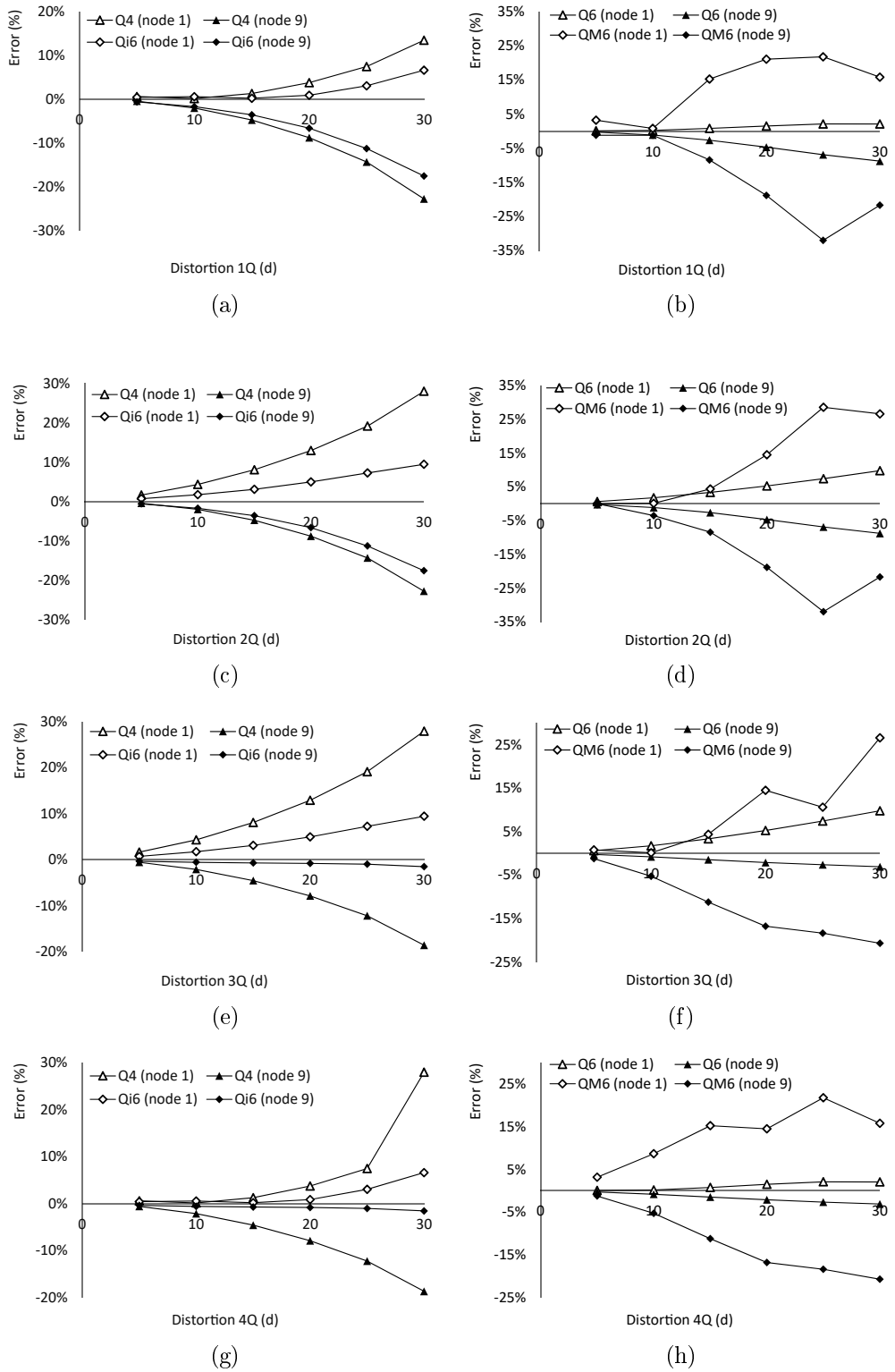


Figure 5.20: Error for the horizontal displacement at nodes 1 and 9 using elements Q4 and Qi6 (left column) and elements Q6 and QM6 (right column).

Chapter 6

Conclusions and future works

The main objective of this dissertation was to develop an in-house software based on the finite element method for pedagogical (teaching/training) and research purposes, and for the structural analysis of incompressible problems. To these ends, a detailed literature review was conducted in order to study the different finite element formulations available and assimilate the underlying concepts necessary for their correct implementation. In a first phase, a solid base platform for two-dimensional analysis was developed employing the classical bi-linear quadrilateral formulation. After proper testing, the development efforts were directed at the implementation of seven finite element formulations targeted at the treatment of numerical anomalies arising from the analysis of incompressible problems, such as volumetric locking. These formulations included a selective integration element, four incompatible mode elements and two compatible mode elements based on the enhanced strain method.

To validate the implementation, a series of benchmark tests was carried out, comparing the results coming from the in-house software with those from Abaqus software. The results proved the poor performance of the classical bi-linear quadrilateral, as already pointed out in some of the literature. In general, the incompatible mode Q6 element of Wilson and the compatible mode Qi6 element proposed by Natal Jorge had demonstrated excellent results, at the same level of the commercial software and effectively validating the implementation (with the latter also demonstrating a good response under distortion). In this context, surprisingly the QM6 element revised by Taylor registered a lower performance than its counterpart, the Q6 element. With the exception of the Q4 element, all the formulations somehow converged to the expected solutions, although finer meshes would be needed to achieve convergence of the selective integration element.

During this work we learned that the commercial software employs a mixed (u/p) formulation in its hybrid elements for incompressibility analysis. This successfully treats the volumetric locking effects but does not solve the shear locking for the CPE4H. In this context, the superior results of the CPE4IH have shown that the inclusion of incompatible modes can help in obtaining a better bending behavior, while the volumetric locking is minimized by the hybrid formulation.

The post-processing capabilities are yielding excellent stress values, generating stress distributions very similar to the commercial software, even after the extrapolation for the classical formulations. Stress computation for the enhanced strain elements was added based on the generalized displacements. Although the procedure is working well, the discontinuous nature of the generalized displacements shows a tendency to produce certain

irregularities resulting in stress distributions not as smooth as the classical formulations.

Regarding future developments, on the short term and without making deep chances to the software, it should be interesting to implement the B-bar and the mixed (u/p) formulations and carry out the necessary changes to accomodate routines allowing for the hourglass control of the selective reduced finite element Q4SRI. It should also be interesting to implement the least-squares projection method suggested by Simo and Rifai for stress computation of the enhanced strain formulations and compare the results with the procedure based on the generalized displacements. The implementation should also focus on routines for two-dimensional axissymetric analysis. Additional benchmark problems should also be performed to validate the implementation. In this context, concerning the example of the beam under bending, the force and displacement boundary conditions have to be revised. On a later phase and given that most of the finite element formulations reviewed and implemented in this work have direct application in non-linear analysis, it would make sense to implement an elasto-plastic model and the necessary iterative structure.

On the mid term and maintaining the scope of the software within the two-dimensional domain, development efforts should be directed to the implementation of routines targeted at the analysis of contact problems. At this point it would also be wise to initiate the implementation of automatic mesh generation algorithms and alternatively start evaluating the possibility of importing CAD geometry from an external program. At this point a unified user interface should be implemented extending the pre- and post-processing capabilities of the program. Another useful feature would be a test platform that would allow to automate repetitive analysis (batch execution).

On the long term, the effort should be directed to extend the implemented features to the 3D-space. In this context, the program would also be able to account for plates and shells formulations.

Bibliography

[1] Alves de Sousa, R. J., Natal Jorge, R. M., Fontes Valente, R. A., and César de Sá, J. M. A., *A New Volumetric and Shear Locking-Free 3D Enhanced Strain Element*, Engineering Computations, 20 (2003), pp. 896–925.

[2] Arunakirinathar, K. and Reddy, B. D., *Further Results for Enhanced Strain Methods with Isoparametric Elements*, Computer Methods in Applied Mechanics and Engineering, 127 (1995), pp. 127–143.

[3] Bathe, K. J., *Finite Element Procedures*, Prentice Hall, 2006.

[4] Boffi, D. and Stenberg R., *A Remark on Finite Element Schemes for Nearly Incompressible Elasticity*, Computers and Mathematics with Applications, 74 (2017), pp. 2047–2055.

[5] Cotrell, J. A., Hughes, T. J. R., and Bazilevs, Y., *Isogeometric Analysis - Toward Integration of CAD and FEA*, Wiley, 2009.

[6] César de Sá, J. M. A. and Natal Jorge, R. M., *New enhanced strain elements for incompressible problems*, International Journal for Numerical Methods in Engineering, 44 (1999), pp. 229–248.

[7] Dias da Silva, V., *Mecânica e Resistência dos Materiais*, ZUARI, 3rd ed., 2004.

[8] Doherty, W. P., Wilson, E. L., and Taylor, R. L., *Stress Analysis of Axisymmetric Solids Utilising Higher Order Quarilateral Finite Elements*, SESM Report 69-3, University of California, Barkeley, 1969.

[9] Elguedj, T., Bazilevs, Y., Calo, V. M., and Hughes, T. J. R., *B-bar and F-bar Projection Methods for Nearly Incompressible Linear and Non-Linear Elasticity and Plasticity Using Higher Order NURBS Elements*, Computer Methods in Applied Mechanics and Engineering, 197 (2008), pp. 2732–2762.

[10] Fanghorn, H., *A Comparison of C, MATLAB and Python as Teaching Languages in Engineering*, Lecture Notes in Computer Science, 3039 (2004), pp. 1210–1217.

[11] Felippa, Carlos A. , *Introduction to Finite Element Method*, Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado, 2004. Web: `https://goo.gl/nAx1rK`.

[12] Ferreira, A. J. M., *Problemas de Elementos Finitos em MATLAB*, Fundação Calouste Gulbenkian, 2010.

[13] Griffiths, D. and Mustoe, G. G. W., *Selective Reduced Integration of Four-Node Plane Element in Closed Form*, Journal of Engineering Mechanics, 121 (1995), pp. 725–729.

[14] Houcque, D., *Introduction to MATLAB for Engineering Students*, Lecture notes, Northwestern University, 2005.

[15] Hughes, T. J. R., *The Finite Element Method - Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, 1987.

[16] Kattan, P. I., *MATLAB Guide to Finite Elements*, Springer, 2nd ed., 2008.

[17] Khennane, A., *Finite Element Analysis Using MATLAB® and Abaqus*, CRC Press, 2013.

[18] Krysl, P., Novák, J., and Oberrecht, S., *Generalized Selective Reduced Integration and B-bar Finite Element Methods for Anisotropic Elasticity*, International Journal for Numerical Methods in Engineering, 98 (2014), pp. 92–104.

[19] Malkus, D. S. and Hughes, T. J. R., *Mixed Finite Element Methods - Reduced and Selective Integration Techniques: A Unification of Concepts*, Computer Methods in Applied Mechanics and Engineering, 15 (1978), pp. 63–81.

[20] Nagtegaal, J. C. and Fox, D. D., *Using Assumed Enhanced Strain Elements for Large Compressive Deformation*, International Journal of Solids and Structures, 33 (1996), pp. 3151–3159.

[21] Nagtegaal, J. C., Parks, D. M., and Rice, J. R., *On Numerical Accurate Finite Element Solutions in the Fully Plastic Range*, Computer Methods in Applied Mechanics and Engineering, 4 (1974), pp. 153–177.

[22] Natal Jorge, R. M., *Modelação de Problemas Incompressíveis pelo Método das Deformações Acrescentadas baseado em Modos Compatíveis*, PhD Thesis, Faculdade de Engenharia da Universidade do Porto, 1998.

[23] Nikishkov, G. P., *Introduction to the Finite Element Method*, Lecture notes, University of Aizu, 2004.

[24] Piltner, R., *An Implementation of Mixed Enhanced Finite Elements with Strains Assumed in Cartesian and Natural Element Coordinates Using Sparse $\bar{B}$-Matrices*, Engineering Computations, 17 (200), pp. 933–949.

[25] Piltner, R. and Taylor, R. L., *A Quadrilateral Mixed Finite Element with Two Enhanced Strain Modes*, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 1793–1808.

[26] Prathap, G., *Barlow Points and Gauss Points and the Aliasing and Best Fit Paradigms*, Computers & Structures, 58 (1996), pp. 321–325.

[27] Reddy, J. N., *An Introduction to the Finite Element Method*, McGraw-Hill, 1984.

[28] Rosetta Code, *Numerical integration/gauss-legendre quadrature*, 2018. Web: `https://goo.gl/Gqg3RU`.

[29] Rupp, C., Howard, M., and Weickum, G., *Incompressible Mixed (u/p) Elements for the CAS FEM Code*, 2006. Course material, Web: `https://goo.gl/FiFGpQ`.

[30] Shigley, J. E., Mischke, C. R., and Budynas, R. G., *Mechanical Engineering Design*, McGraw-Hill, 7th ed., 2004.

[31] Simo, J. C. and Hughes, T. J. R., *On the Variational Foundations of Assumed Strain Methods*, Journal of Applied Mechanics, 53 (1986), pp. 51–54.

[32] Simo, J. C. and Rifai, M. S., *A Class of Mixed Assumed Strain Methods and the Methods of Incompatible Modes*, International Journal of Numerical Methods in Engineering, 29 (1990), pp. 1595–1638.

[33] Simo, J. C., Taylor, R. L., and Pister, K. S., *Variational and projection methods for the volume constraint in finite deformation elasto-plasticity*, Computer Methods in Applied Mechanics and Engineering, 51 (1985), pp. 177–208.

[34] SIMULIA Dassault Systèmes, *Abaqus Theory Manual 6.14*, 2014. Web: `http://abaqus.software.polimi.it/v6.14/`.

[35] Sinwel, A. S., *A New Family of Mixed Finite Elements for Elasticity*, PhD Thesis, Institut für Numerische Mathematik, Johannes Kepler Universität, 2009.

[36] Teixeira-Dias, F., Pinho-da-Cruz, J., Fontes Valente, R. A., and Alves de Sousa, R., *Método dos Elementos Finitos - Técnicas de Simulação Numérica em Engenharia*, Editora ETEP, 2010.

[37] Zhang, W., Zhao, L., and Cai, S., *Shape Optimization of Dirichlet Boundaries based on Weighted B-Spline Finite Cell Method and Level-Set Function*, Computer Methods in Applied Mechanics and Engineering, 294 (2015), pp. 359–383.

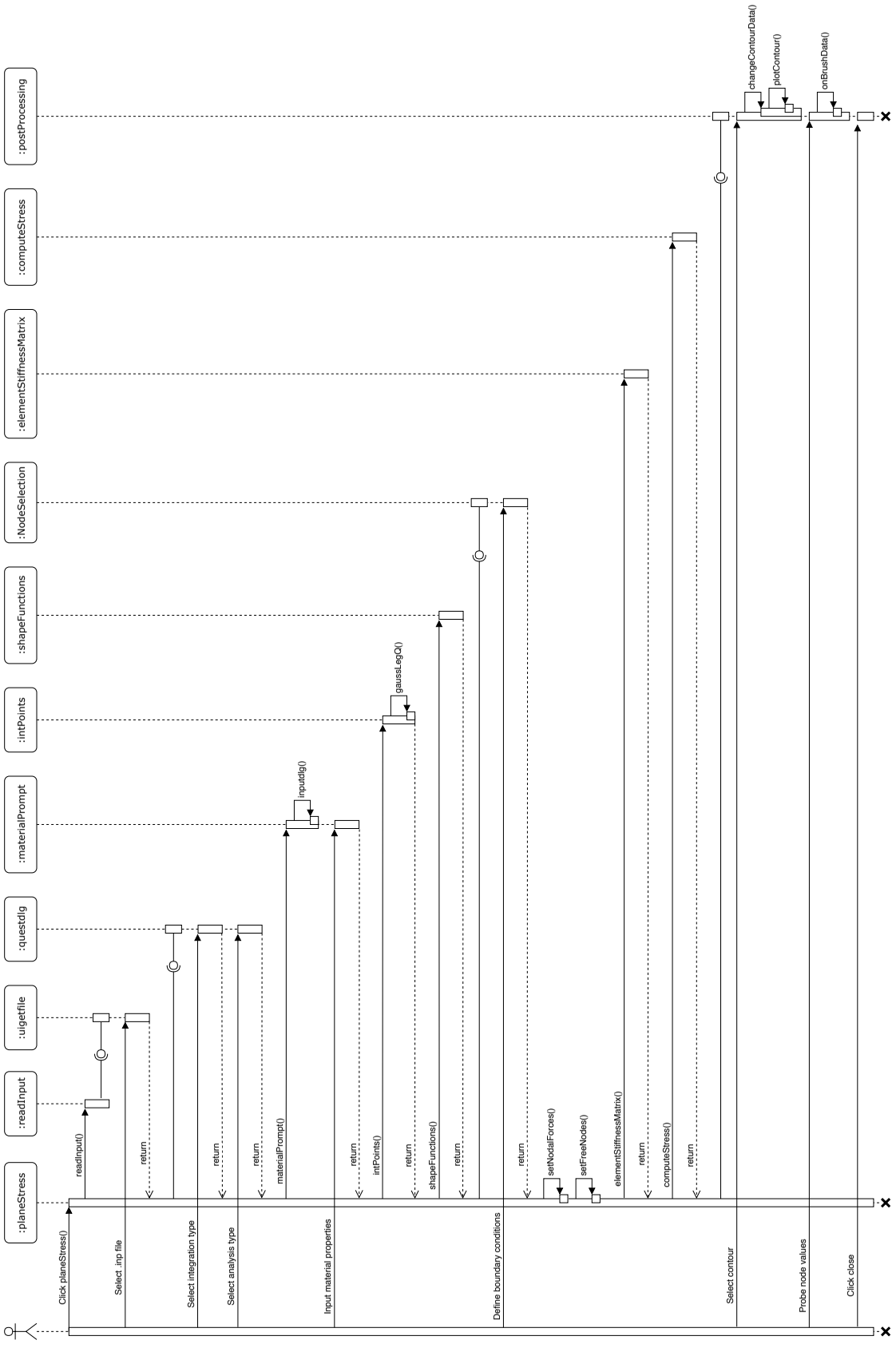
[38] Zienkiewicz, O.C., Taylor, R.L., and Zhu, J.Z., *The Finite Element Method: its Basis and Fundamentals*, Butterworth-Heinemann, 6th ed., 2013.

Intentionally blank page.

Appendix A

Program architecture

Intentionally blank page.


:planeStress
:readInput
:uigetfile
:questdlg
:materialPrompt
:intPoints
:shapeFunctions
:NodeSelection
:elementStiffnessMatrix
:computeStress
:postProcessing
Click planeStress()
Select .inp file
Select integration type
Select analysis type
Input material properties
Define boundary conditions
Select contour
Probe node values
Click close
readInput()
return
materialPrompt()
inputdlg()
intPoints()
gaussLegQ()
shapeFunctions()
setNodalForces()
setFreeNodes()
elementStiffnessMatrix()
computeStress()
changeContourData()
plotContour()
onBrushData()

Intentionally blank page.

Appendix B

Abaqus input file structure

Intentionally blank page.

Listing B.1: Abaqus input file example.

```
*Heading
** Job name: Job-1 Model name: Model-1
** Generated by: Abaqus/CAE Student Edition 2017
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Part-1
*Node
      1,           0.,           0.
      2,           5.,           0.
      3,          10.,           0.
      4,           0.,           2.
      5,           5.,           2.
      6,          10.,           2.
*Element, type=CPS4
1, 1, 2, 5, 4
2, 2, 3, 6, 5
*Nset, nset=Set-1, generate
 1,  6,  1
*Elset, elset=Set-1
 1, 2
*Nset, nset=fixed
 1,
*Nset, nset=ss
 4,
*Nset, nset=load1
 6,
*Nset, nset=load2
 3,
** Section: Section-1
*Solid Section, elset=Set-1, material=Material-1
1.,
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Part-1-1, part=Part-1
*End Instance
**
*End Assembly
**
** MATERIALS
**
*Material, name=Material-1
*Elastic
1500.,0.
(continues...)
```

Intentionally blank page.

Appendix C

Gauss-Legendre quadrature algorithm

Intentionally blank page.

Listing C.1: Algorithm employed for the generation of the Gauss quadrature points.

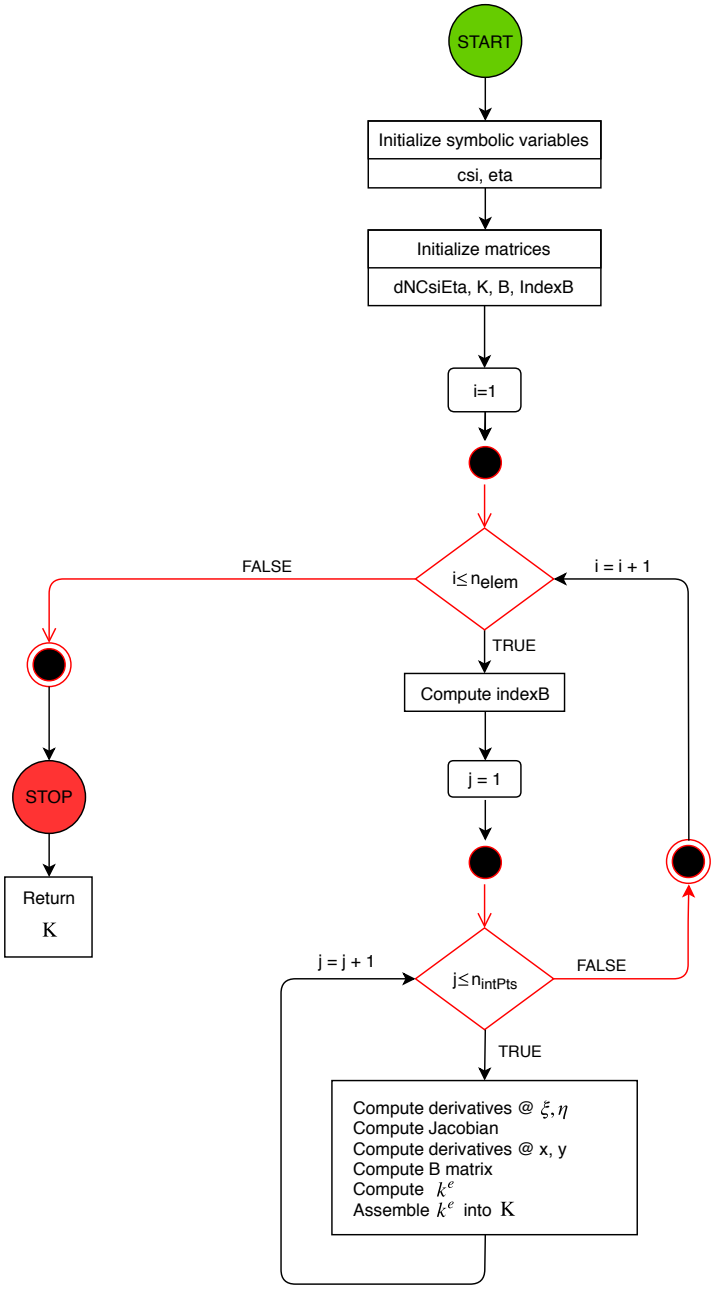
```
function [ q ] = gaussLegQ( N )

    % Initial guess
    x=cos(pi*((1:N)'-0.25)/(N+0.5));
    % Legendre-Gauss Matrix
    L=zeros(N,N+1);

    x0=2;
    while max(abs(x-x0))>eps
         L(:,1)=1;
         L(:,2)=x;

         for k=2:N
              % Legendre polynomial Pn defined by recursive rule
              L(:,k+1)=((2*k-1)*x.*L(:,k)-(k-1)*L(:,k-1))/k;
         end
         % Derivative Pn'
         Lp=N*(x.*L(:,N+1)-L(:,N))./(x.^2-1);
         % Newton-Raphson method for the roots
         x0=x;
         x=x0-L(:,N+1)./Lp;
    end
    %Computing weights
    w=2./((1-x.^2).*Lp.^2);
    q=[-x w];

end
```

Intentionally blank page.

Appendix D

Element stiffness assembly algorithm

Intentionally blank page.


START
Initialize symbolic variables
csi, eta
Initialize matrices
dNCsiEta, K, B, IndexB
i=1
i≤nelem
FALSE
i = i + 1
TRUE
Compute indexB
j = 1
STOP
Return
K
j = j + 1
j≤nintPts
FALSE
TRUE
Compute derivatives @ ξ,η
Compute Jacobian
Compute derivatives @ x, y
Compute B matrix
Compute k^e
Assemble k^e into K