Object-Oriented Nonlinear Finite Element Analysis Framework For Implementing Modified Cam Clay Model

A Thesis Submitted for the Degree of Doctor of Philosophy

By

Dipika Devi

Department of Civil Engineering
Indian Institute of Technology Guwahati
Guwahati-781039, India

September, 2011

Declaration

I hereby declare that the thesis entitled **"Object-oriented nonlinear finite element analysis framework for implementing modified Cam clay model"** has been carried out by me in the Department of Civil Engineering, Indian Institute of Technology Guwahati, India under the supervision of **Dr. Arbind Kumar Singh**.

In keeping with the scientific tradition, due acknowledgments have been made, wherever the work described is based on the findings of other investigators.

(**Dipika Devi**)

September, 2011
Guwahati.

Dedicated

to

My Father

Late Narayan Ch Barma

CERTIFICATE

It is to certify that the thesis entitled **"Object-oriented nonlinear finite element analysis framework for implementing modified Cam clay model"** being submitted by **Dipika Devi**, to the **Indian Institute of Technology Guwahati, India** for the award of the degree of **Doctor of Philosophy** is a record of bonafide research work carried out by her under my supervision.

The thesis work, in my opinion is worthy of consideration for the award of degree of **Doctor of Philosophy** in accordance with the regulations of the Institute.

(Arbind Kumar Singh)
Associate Professor
Department of Civil Engineering
Indian Institute of Technology Guwahati

September, 2011
Guwahati.

ACKNOWLEDGMENTS

It gives me immense pleasure to express my deep sense of gratitude and hearty thanks to my supervisor, Dr. Arbind Kumar Singh, for his invaluable guidance and constant encouragement and all out help throughout the progress of this work.

I would like to express my hearty thanks to Hods, Civil Engineering Department during my stay for providing all the facilities needed. I am also thankful to other faculty members who had played a vital role in bringing me to this level. I am thankful to the technical and non-technical staff of the Civil Engineering Department for their assistance whenever needed.

I am thankful to all my family members for their whole-hearted support and constant encouragement towards the fulfillment of the degree. I wish to record the help extended to me by my husband and my parents in-law in all possible ways. I specially thank my daughter Pammi and son Aadi.

Dipika Devi

September, 2011 IIT Guwahati

Contents

List of Figures

List of Tables

Nomenclature

$\boldsymbol{B}$	Strain displacement matrix
$\boldsymbol{C}^e$	Fourth order elasticity tensor
E	Modulas of elasticity
e	Void ratio
$\boldsymbol{e}$	Total deviatoric strain tensor
f and F	Yield surface
g	Plastic potential surface
G	Shear modulus
I	Identity tensor
I_1	First invariant of volumetric stress tensor
J or J_2	Second invariant of deviatoric stress tensor
J_3	Third invariant of deviatoric stress tensor
K	Bulk modulus
$\boldsymbol{K}_e$	Element stiffness matrix
$\boldsymbol{K}_t$	Tangential stiffness matrix
k	Hardening parameter
$\boldsymbol{l}$	Kronecker delta
M	Slope of critical state line in $p-q$ plane
p	Volumetric stress
p_c	Pre-consolidation pressure

q	Deviator stress
$\boldsymbol{R}$	Residual force vector
$\boldsymbol{S}$	Deviatoric stress tensor
U	Global displacement vector
v	Specific Volume
$\boldsymbol{\epsilon}$	Total strain tensor
$\boldsymbol{\epsilon}^e$	Elastic strain tensor
$\boldsymbol{\epsilon}^p$	Plastic strain tensor
ϵ_v	Total volumetric strain
$\boldsymbol{\epsilon}_d$	Total deviatoric strain tensor
ϵ_a	Axial strain
ϕ	plastic multiplier for Modified Cam clay model
Φ	Angle of internal friction of soil
μ	Poisson's ratio
$\boldsymbol{\sigma}$	Cauchy stress tensor
σ_r	Stress in radial direction
σ_a	Stress in axial direction
γ	Unit weight of soil
κ	Slope of swelling curve
λ	Slope of the virgin compression curve

List of abbreviations

ARC	Arclength method
AUTOITR	Automatic Iteration method
AUTOINC	Automatic Incrementation method
DAE	DIfferential algebraic equation
DIRK	Diagonally implicit Runge-Kutta
FE	Finite element
FEM	Finite element method
MCCM	Modified Cam clay model
MNR	Modified Newton Raphson method
MNRC	Modified Newton Raphson method with Chen accelerator
MNRT	Modified Newton Raphson method with modified Thomas accelerator
NR	Newton Raphson method
OCR	Over consolidation ratio
OOFEA	Object-oriented finite element analysis
OOP	Object-oriented programming
PDE	Partial differential equation
RK	Runge-Kutta

ABSTRACT

Cam clay and modified Cam clay models are most widely used plasticity based constitutive models in soil mechanics. Finite element method is a powerful numerical tool which can model many complex conditions and popularly used to analyze geotechnical engineering problems. Traditionally procedure oriented programing, particularly in FORTRAN or C, has been used but recently object-oriented programming has gained wide attention and used by many investigators to develop finite element framework. In this present work an object-oriented finite element analysis framework for modified Cam clay model has been developed using C++ as programming language. In this framework, the whole finite element analysis class is divided into separate component classes to perform various tasks. Extensions to the design are presented in terms of inclusion of new integration algorithms for an existing constitutive model, inclusion of new constitutive models and inclusion of different nonlinear solution techniques and load stepping schemes with an existing integration algorithm of a constitutive model.

In this study, firstly, an implicit integration algorithm for modified Cam clay model incorporating nonlinear elasticity is presented. The proposed implicit stress integration algorithm for the modified Cam clay model is based on the return mapping algorithm by making use of the both first order forward Euler scheme and the backward Euler scheme, considering associative flow rule. Secondly, some of the nonlinear solution techniques and load stepping schemes are presented and implemented with the modified Cam clay model. Their performances in some practical geomechanics problems are compared in terms of solution accuracy, number of global iteration and CPU time required to run the program. Thirdly, the context of interpreting the descretized form of the principle of virtual displacement and the ordinary differential equations of the constitutive model together as a system of differential

algebraic equation is discussed. The effectiveness of this treatment of the nonlinear finite element as differential algebraic equations is also discussed and the concept is implemented to modified Cam clay model. It is shown that the general case of diagonally implicit Runge-Kutta methods, which is stiffly accurate and has better stability property, can be applied to integrate the nonlinear modified Cam clay model equations. In this study, we have implemented two special cases of diagonally implicit Runge-Kutta method, namely, (i) the modified Euler method and (ii) the trapezoidal rule to integrate the modified Cam clay model using the concept of differential algebraic equation.

The integration algorithms presented for modified Cam clay model and the various load stepping schemes are implemented in this object-oriented finite element analysis framework. The functionality of the classes used in the framework are presented for solution of some practical geomechanics problems using modified Cam clay model, which are compared with established data. The types of numerical problems analyzed using the modified Cam clay model with various load stepping schemes to validate the proposed integration algorithms in the object-oriented programming framework include triaxial tests and plane strain analysis of strip footings.

In the subsequent chapters details of the integration algorithms, the nonlinear solution techniques and load stepping schemes, about the design of object-oriented finite element method and its implementation with modified Cam clay model is presented. For this purpose, the thesis has been organized in ten chapters. Chapter 1 highlights the general introduction with the motivation and objective of this work. The relevant literature review is discussed in Chapter 2 in the areas of (i) Object-oriented programming in finite element method, (ii) Soil constitutive models and numerical integration (iii) Nonlinear solution techniques and load stepping schemes, and (v) Interpretation of nonlinear finite element method as differential algebraic equations. A brief introduction about Cam clay models and various nonlinear solution techniques are presented in Chapter 3 and Chapter 4 respectively. The proposed implicit integration algorithm for modified Cam clay model is described in Chapter 5. Chapter 6 presents the application of diagonally implicit Runge-Kutta method to integrate the modified Cam clay model using the concept of differential algebraic equation. Chapter 7 and Chapter 8 present a new object-oriented finite element framework and its implementation for modified Cam clay model with the proposed

integration algorithms and different nonlinear solution techniques. In Chapter 9 few relevent problems are solved to illustrate the effectiveness of the new integration algorithms of modified Cam clay model along with the various load stepping schemes. Chapter 10 summarizes the main contribution of this research work and gives suggestions for future work.

Chapter 1

Introduction

1.1 General

Behaviour of soil is highly nonlinear and consistent with the elasto-plastic framework. Modified Cam clay model (MCCM) is developed from the application of plasticity to soil mechanics and is the most widely used soil constitutive model. The model is highly nonlinear in nature and a numerical tool is required for the application of this model to solve a problem in soil mechanics. Finite element method has gained wide acceptance for analyzing geomechanics problems and it has been shown that the finite element method can provide realistic results for practical engineering problems, when properly used. Traditional way of designing a finite element analysis software is by application of procedure-oriented languages like FORTRAN or C. The finite element software developed with procedure-oriented languages contain complex data structure and cannot be modified or extended without high degree of knowledge of the entire program. This makes the resulting code inflexible. Object-oriented programming (OOP) language has lots of advantages over procedure-oriented language. In OOP, more emphasis is given on data. The problem is decomposed into objects, and data and procedures are tied together in the data structure of objects. Bottom-up approach of design is used and any part of the program can be easily modified and extended without any ripple effect throughout the entire program. Therefore, an well designed OOP framework for finite element method can make the software flexible.

In the recent years a few object-oriented finite element frameworks have been presented which can be extended in some specific directions, e.g. the introduction of new element types or solving strategies. Much less support is available for task control, creation of new material models, creation of new integration algorithms or extension of the analysis model. Hence there is need of development of object-oriented software for nonlinear finite element analysis [144]. In this present study, an object-oriented nonlinear finite element analysis (FEA) framework has been developed using C++ as programming language and MCCM with an implicit stress integration algorithm has been implemented in this framework.

The present research work on object-oriented nonlinear finite element analysis for MCCM can be broadly divided into three parts. The first part is aimed at the integration of MCCM equations. An implicit integration algorithm for this model is presented by taking into account the nonlinearity in the elastic region of the model. In the second part implementation of various nonlinear solution techniques and load stepping schemes for elasto-plastic models are discussed and their effectiveness for MCCM is presented with comparison of these different techniques. The third part of the study is aimed at the interpretation of nonlinear finite element analysis as differential algebraic equations (DAEs) and its implementation to integrate the MCCM equations. The proposed framework is used to validate the new integration algorithms for MCCM and to compare the performance of different nonlinear solution techniques and load stepping schemes when used with MCCM. It is demonstrated that the framework can be easily extended to incorporate new constitutive models for soil with different integration algorithms and load stepping schemes.

1.2 Motivation of the research

1.2.1 Numerical integration of MCCM

The finite element method (FEM) was introduced to the geotechnical engineering profession by Clough and Woodward in 1967 [50]. This numerical tool is so powerful and general that it enables modeling of many complex conditions with a high degree of realism. It has been shown that the method can include in the analysis such things as nonlinear stress-strain behaviour, stress-dependent stress-strain behavior,

non-homogeneous conditions, changes in geometry during construction, dissipation of excess pore pressures following construction etc. The inclusion of all these generalities gave the finite element method great potential for the use in geotechnical engineering problems. Rapid advances in the field of computing also has brought revolution in the field of numerical analysis and in recent years there is marked increase in the use of FEM to analyze geotechnical engineering problems. It is understood from the experience gained till date that a finite element approach offers real benefits over the existing methods of analyzing any boundary value problem in geotechnical engineering.

The ability of a numerical approach like FEM to accurately model field conditions depends mainly on two conditions: (i) the constitutive model used in the analysis should accurately represent the real soil behaviour and (ii) the boundary conditions of the problem should be imposed correctly. An finite element analysis can simulate the field condition accurately and can predict the complete history of the boundary value problem. The analysis can also predict the failure mechanism or mode of behaviour of the problem and no postulation is to be made by the user. But for reliable numerical predictions of performance of practical geotechnical problems, one has to use a realistic soil model with appropriate information on the geometry, construction procedure, soil parameters and boundary conditions [133].

The most widely used generalized models for soil behaviour are Cam clay model and MCCM which are formulated using the conceptual framework of critical state soil mechanics. In order to implement these models in nonlinear finite element analysis for solution of boundary value problems, the constitutive equations must be integrated numerically over finite time steps. These methods used to integrate the constitutive equations have direct impact on the overall performance of the nonlinear finite element analysis and has been active area of research currently ([150], [151], [166], [118], [36]). There have been considerable advances in the methods of integration and superiority of one over the other lies in the context of convergence, accuracy, stability, consistency and simplicity of performance. In the present study an implicit integration algorithm is presented for MCCM and the same is implemented in an object-oriented finite element framework.

1.2.2 Nonlinear solution techniques and load stepping schemes

In finite element analysis, with a nonlinear material model, an iterative and incremental scheme is required to solve the resulting system of nonlinear equations. There are series of solution schemes developed for solving nonlinear equation system. In context to the solution of nonlinear finite element problems, the most conventional method is the Newton-Raphson method in any of its different versions as it has the property of quadratic convergence. However, Newton-Raphson method or any of its versions may not be always adequate and competent in solving certain problems involving nonlinearity. Hence, there may be a need of relatively advanced techniques of solving nonlinear finite element problems.

Variety of incremental and iterative schemes are developed in the past to integrate the load-displacement curve for nonlinear material models. However, the accuracy, efficiency and stability of these schemes is strongly influenced by the load increment size, when used with a highly nonlinear material model. Cam clay group of models are nonlinear in nature in the elastic as well as in the plastic parts of the constitutive relation. Therefore, the performance of different incremental and iterative schemes for finite element analysis particularly with critical state models is to be done and can be an active area of study.

Sheng and Sloan [147] presented and compared the performance of some of the load stepping schemes using an explicit sub-stepping integration algorithm for generalized Cam clay model. In the present study, these incremental and iterative load stepping schemes are used with an implicit integration algorithm for MCCM. They are implemented in the proposed object-oriented nonlinear finite element framework for MCCM. The performance of these schemes in terms of efficiency and accuracy is presented for some practical geomechanics problems.

1.2.3 Interpretation of nonlinear finite element analysis as DAEs and its implementation to MCCM

Interpretation of nonlinear finite element as DAEs facilitates to apply all the numerical algorithm of mathematics for its solution. This approach combines the time integration method of differential equation and a multi-level Newton-Raphson

method to solve nonlinear system of algebraic equations. In this study using the concept of DAEs, the MCCM equations are integrated using diagonally implicit Runge-Kuttta method (DIRK), which is stiffly accurate and has better stability property and the resulting coupled system of equations are solved using a multi-level Newton-Raphson method. For this purpose,the generalized trapezoidal numerical integration rule, which is also called as the α method, is selected, as this method is easy to incorporate in the general DIRK method. The α method is implemented to integrate the MCCM, with two special cases: (i) Backward Euler method and (ii) Trapezoidal rule. These integration algorithms for MCCM are then implemented in the proposed object-oriented finite element framework.

1.3 Objective

The general objective of this research work is to develop an object-oriented nonlinear finite element analysis framework for elasto-plastic models for use in analysis of practical geomechanics problems, which can be extended easily to suit a new task, in terms of inclusion of new material model, new integration algorithm and analysis type. It is observed that MCCM is the most widely used soil constitutive model for numerical analysis in the field of geotechnical engineering. So the system design and a prototype finite element implementation of MCCM for soil in the proposed object-oriented finite element framework are presented. The design will provide geotechnical engineering profession with a flexible tool that can be adapted to meet future requirement.

The specific objectives of this research work can be outlined as follows:

1. Developing an implicit integration algorithm for MCCM and implementing the same in the proposed object-oriented FEA framework.

2. Implementing the different solution techniques and load stepping schemes for MCCM in the proposed object-oriented FEA framework.

3. Application of more general DIRK method to integrate the MCCM using the concept of interpretation of finite element method as DAEs.

4. Comparison of the solutions for practical geomechanics problems obtained by MCCM with different integration algorithms and load stepping schemes.

5. Developing an object-oriented FEA framework for MCCM using C++ as programming language.

1.4 Layout of the thesis

The layout of the dissertation is described below.

Chapter 2 provides a survey of the current literature regarding object-oriented finite element analysis design and programming, soil constitutive models and numerical integration of MCCM, nonlinear solution techniques and load stepping schemes, and interpretation of nonlinear finite element as DAEs.

Chapter 3 gives a brief introduction of soil plasticity and development of critical state models.

Chapter 4 expounds an implicit integration algorithm developed for the integration of the rate constitutive equation for MCCM.

Chapter 5 provides the details of some of the load stepping schemes, which are implemented with MCCM in an object-oriented finite element framework, and their performances are compared in some practical problems.

Chapter 6 elucidates the interpretation of nonlinear finite element analysis as DAEs and explains the application of this approach of DAE to integrate the MCCM.

Chapter 7 gives the concept of object-oriented design and programming for FEM. An object-oriented nonlinear finite element framework for MCCM is developed with special emphasis on its application for practical geomechanics problems.

Chapter 8 explains the details of implementation of MCCM with the proposed integration algorithms and the various load stepping schemes in the proposed object-oriented FEA framework.

Chapter 9 gives the numerical examples used to validate the proposed integration algorithms in the proposed object-oriented FEA framework with the different load stepping schemes, and summaries the results obtained from different problems.

Chapter 10 concludes the thesis with a summary of the achievements, draws conclusions on findings and finally suggests ideas for future studies.

Chapter 2

Literature review

2.1 Introduction

In the last few decades considerable advances have made in the field of application of numerical techniques for solving practical geomechanics problems. Many universities and research institutes have developed their own FEM software for use in this field and as a results, many commercial FEM software are available today. In one hand, most of these software are written in procedure-oriented languages like FORTRAN or C, consisting of several thousands of lines of procedural codes with very complex data structure and on the other hand, a few object-oriented finite element architectures had been presented for use in this field.

A material model is a set of mathematical equations that describes the relationship between stress and strain. Material models are often expressed in a form in which infinitesimal increments of stress are related to infinitesimal increments of strain. In case of all elasto-plastic models, it is not possible to get the exact analytical solutions of these equations and they are to be implemented in the analysis using an integration algorithm. Different integration algorithms, both explicit and implicit, that apply directly to the critical state models have been developed in the past with one having some advantages over the other in some aspects. In the last few decades, there are continuing efforts in the improvement of these integration algorithms as they play a vital role in the finite element implementation of the constitutive equations.

Again the constitutive models which are mostly used for geotechnical problems solving are highly nonlinear in nature. There are different nonlinear solution techniques and load stepping schemes developed in the past for implementing these model in finite element codes. For successful implementation of these techniques, particularly for critical state models, a comparative study on their performance are to be made. In this chapter a review is made on the historical contribution made by different investigators in these subject areas, which are particularly relevant to this thesis. Keeping all these in view, the literature study is done in the following directions:

- Object-oriented programming in FEM,
- Soil constitutive models,
- Numerical integration of constitutive relations,
- Nonlinear solution techniques and load stepping schemes,
- Interpretation of nonlinear finite element as DAEs.

2.2 Object-oriented programming in FEM

The idea of implementing object-oriented languages in finite element programming has got its existence in late eighties with the development of several OOP languages like small-talk, Ada and C++. The proper use of these languages requires a different approach to design than the approach typically takes with languages such as FORTRAN, COBOL and even PASCAL. From that time many works have been done to replace the traditional procedure oriented programming style with the new concept of object-oriented design. In the following section a brief review of the literature available in this direction is presented.

Miller [111, 112] presented an object-oriented software architecture for use in nonlinear dynamic finite element analysis. The system is based on a coordinate-free geometry, which includes points, vectors and tensors in three dimensions. No mention is made of supporting numerical work. No specific solution schemes are given, but the intention of the work is to allow for iterative element-by-element solution

methods. Transformations are handled by the coordinate-free geometry. Since all elements and nodal properties are formulated without reference to a specific coordinate system, the elements and nodes do not participate in the transformation process. Elements report their properties in terms of geometric vectors and tensors. No information is given as to how these are transformed into numeric vectors and matrices for solution. Since the work intended for a element-by-element solution algorithm, no mapping between the model and analysis equations of equilibrium is provided. As a result of the coordinate free geometry, an element can be used in one, two or three dimensional problems, regardless of its internal representation.

Baugh and Rehak [19] presented an architecture for a linear static finite element analysis program. The bases of the system is the coordinate system objects, which permit the elements and nodes to describe themselves in terms of Cartesian, Cylindrical and Spherical systems. The coordinate systems are related to a global coordinate system by a 3x3 rotation matrix(direction cosines). These rotation matrices are used to perform transformations of load vectors and stiffness matrices from one coordinate system to another. Each node stores its position in its own coordinate system. The same coordinate system is used to describe the orientation of the degree of freedom at the node. Both rotations and translations use this same system. The displacements calculated by the analysis are stored at the nodes. The degree of freedom maintain their boundary condition and loading value. Only single point constrain are permitted. The nodes also stores the topography of the model, that is connectivity caused by the element.

Dubois-perlin et al. [46], Dubois-perlin and Zimmermann [47], and Zimmermann et al. [180] have demonstrated the usefulness of this alternative programming paradigm in the field of finite elements. They have developed a software architecture for linear dynamic finite element analysis. The system is similar to traditional finite element analysis programs in that all data are stored in terms of global degree of freedom, and the properties of a structure as a whole are assembled into a system of linear equations. Transformation are performed implicitly by the object that creates the information. No geometric classes are defined to aid in element geometry and vector based data manipulation. A basic numerical library is provided which contains full, banded and skyline matrices and full vectors. The concept of the object-oriented approach is discussed in the context of the implementation of a simple finite element

program in small-talk and C++ language.

Menetrey and Zimmermann [108] has applied the concept of object-oriented programming to the finite element method for nonlinear static analysis. The key aspects of the implementation are the Newton-Raphson and the stress integration algorithms. A procedural description of the algorithm is presented first, from which the necessary extensions to the existing classes are developed. These class extensions consist of the additional attributes and methods required to carry out the nonlinear analysis.

Zeglinski et al. [175] has presented a detailed discussion of OOP techniques in the creation of a generalized matrix library. The matrix library contains a number of matrix objects that are useful for specific types of matrix related problems. Different sparse storage schemes are implemented for each different type of matrix. A large number of functions are provided for each matrix type in order to implement many common matrix operations. They also explained how to set up a finite element hierarchy, material hierarchy and how to integrate this with the matrix hierarchy library.

Miller et al. [113] presented a computational approach in finite element method using the concept of object-oriented programming intended to support interactive modeling of geomechanics problems. It is reported that a system is constructed possessing several special features, including coordinate-free, abstract geometrical modeling, a dynamic problem formulation, encapsulated iterative solution techniques, and element-by-element implementation algorithms, and the basic architecture of the system is presented in terms of the various classes of objects which serve as the computational agents.

Adeli and Yu [3] presented a simple linear elastic finite element system that is closely related to the traditional finite element programs. All the nodes have three displacement degree of freedom. The orientation of these degree of freedoms coincide with the assumed global coordinate system. No information is provided as to how the equations of equilibrium are formed, or their relation to the degree of freedom. No facilities for geometric support or transformations are provided. A small numeric library is developed which includes full vector and full matrix classes. A separate class, named GlobalData, is created to make some model parameters, such as the

numbers of degrees of freedom, available to all objects in the system. Some classes use the GlobalData class as a superclass "to provide better service".

Lu et al. [97] presented an excellent architecture for a small, flexible, finite element analysis program for structural engineering computing. The heart of the system is an assembler object, which performs the transformation of element properties between the element coordinates and the structural degree of freedom. The Assembler object also assigns equation of equilibrium numbers to the structural degree of freedom and builds the structural property matrices. A large numerical library was developed by the authors. It includes banded, triangular, sparse symmetric, and profile matrices. Although no sets of geometric support objects are given, reference to coordinate systems and tensors indicate that at least a rudimentary system is implicit in the design. The elements provide their local coordinate systems, and the nodes provide the structural coordinate system, to the Assembler. Elements can provide their stiffness, mass, and damping matrices. No facilities for state update or resisting force calculations are provided. Element forces are communicated to the Assembler as equivalent nodal loads to be included in the right hand side of the equation of equilibrium. A material class is available to provide elements with a basic stress-strain law. The element is responsible for using this to derive the appropriate constitutive equations, such as moment-curvature, as needed. Nonlinear material objects are permitted, but no details are given.

Miki and Murotsu [110] has proposed object-oriented structural and geometric analysis methods for truss structures. The methods are based on the iterative relaxation of nodal unbalanced forces by moving the nodes. The entities appeared in truss analysis are modeled as classes, and their class hierarchy is established. The knowledge about a truss node, a truss member, a material, and a sectional shape of the member are extracted and constructed as a knowledge base in an object-oriented manner. The object-oriented structural analysis is found to be valid for material and geometric nonlinearity, and the object-oriented geometric analysis where a kinematic problem is converted in to a problem for a force equilibrium at the nodes is also found to be effective for determining the geometry of variable geometry trusses.

Kong [79] presented a design strategy for the data structure of object-oriented FEM program, enhancing data hiding and cohesion, decreasing data coupling, as well as managing finite element objects. A process of defining private data members and

arranging them into C+ + FEM class hierarchies is proposed.

Besson and Foerch [20] have discussed and examined the aspects of object-oriented finite element design particularly in the case of large project. A simple core library has been described which aids program development by isolating repetitive tasks into optimized classes. This library addresses particularly the mathematical entities which appear in mechanics.

Zimmermann and Eyheramendy [53, 54, 181] have systematically developed a methodology of coding finite elements, where, the emphasis is put on a symbolic environment. In the first volume of their paper, the key features of an interactive quasi-automatic environment for the development of a new numerical finite element model for the solution of an initial-boundary-value problem have been presented. In the second volume of the paper, the structure of the proposed environment was described. The key features of the automatic generation of finite element in a symbolic object-oriented environment are described with examples in the third volume of their paper.

Boris and Stein [29] has described a novel programming tool, *nDarray*, which is designed using an object-oriented programming paradigm and implemented in the C++ programming language. Finite element equations, represented in terms of multidimensional tensors are easily manipulated and programmed. The usual matrix form of the finite element equations are traditionally coded in FORTRAN, which makes it difficult to build and maintain complex program systems. On the other hand with the help of OOP concept they handled the complicated tensorial formulae associated with the numerical solution of various elastic and elasto-plastic problems in solid mechanics.

Dubois-Perlin and Pegon [48, 49] has shown flexibility of using OOP concept in the field of nonlinear finite element analysis, provided that the right type of objects with adequate capabilities are defined for handling nonlinear behaviours. In order to implement nonlinear models together with alternative methods for solution control, for nonlinear equation solving and for linear equation solving, a set of three classes are introduced: problem, nonlinear solver and linear solver.

Zimmermann et al. [182]presented a user-friendly environment, which combines

symbolic and numeric capabilities, includes reasoning and some advanced graphical tools. Most of the capabilities were tested under both Smalltalk and C++.

Archer et al. [7] presented a new architecture for object-oriented finite element analysis software, which is capable of the modeling and simulation of structural behavior, including the consideration of nonlinear static and dynamic effects.

Lichao and Ashok [92] presents the implementation of the object-oriented framework indicating some of the major benefits of using OOP for FEM software. It describes the implementation of the finite element methodology using a modular framework that allows easy extension of abstract classes to introduce new types of analysis, interpolation, loads or boundary conditions.

Sampath and Zabaras [143] has given the concepts of designing specific OOP software for a novel implementation of a class of adjoin optimization problems typical of the infinite-dimensional design and control of continuum systems. A typical design or control problem takes the form of a continuum system governed by one or more partial differential equations (PDEs) with insufficient or no boundary conditions are available in part of the boundary. In addition , over-specified boundary conditions are supplied in another part of the boundary or within the domain. These problems are generally ill-posed, so in simple terms the ill-posed problems are converted into a well-posed functional optimization problem.

Zimmermann and Eyheramendy [55] extended their work of the automatic generation of finite element in a symbolic object-oriented environment to nonlinear formulations with its application to the numerical solution of Navier-Stokes equations. The objectives of the paper is to develop the complete solution scheme for a simple nonlinear problem, to show its fast extendability and to extend the application of the concepts developed for a one dimensional model, to its multidimensional counterpart.

Jeremic and Yang [74] have presented an approach to computations in elasto-plastic geomechanics based on the object oriented design philosophy, which allows for creation of template material models. It is reported that the analysis of template material models allows for an easy implementation of other elasto-plastic models based on the object oriented design principles.

Modak and Sotelino [116] have addressed an object-oriented programming framework for the time history analysis of structures subjected to dynamic loads using high performance computing environments. The object-oriented approach is employed to facilitate extensibility, re-usability, maintainability and simplicity of the resulting software.

Martha [100] describes the development of a FEA system, called FEMOOP, based on the OOP paradigm. The system has been used as the computational platform for different researches, including the modeling of strain localization, the nonlinear analysis of concrete structures, the shape optimization of nonlinear continuum structures, the adaptive analysis and shape optimization of free-form shells, and the analysis of plates.

Mackerle [101] has given a bibliographical review of the object-oriented programming applied to the nite element method and the boundary element method. The bibliography contains the publications between 1990 and 2003, which includes journal papers, conference proceedings papers and theses/dissertations on the subjects that were published.

Kromer et al. [86, 87] have described an approach to the design and implementation of a multi-body systems analysis code using an object-oriented architecture with emphasis on the adequacy between object-oriented programming and the finite element method used for the treatment of three dimensional multi-body flexible mechanisms.

Boyan and Charles [30] describe aspects of the design and implementation of an OOP based FE code primarily for use in large geomechanics simulations. It is reported that the code allows various types of finite element analysis, as well as the modeling of multi-eld physical phenomena.

Franco et al. [58]have described and applied the finite element technique using OOP for the limit analysis of axisymmetrical pressure vessels.

Scott et al.[144] developed a modular, flexible and reusable software framework for simulating nonlinear response of beam-column members using the object-oriented concept of data encapsulation and polymorphism.

Heng and Mackie [66] have proposed the use of software design patterns to capture best practices in object-oriented FE programming. Five basic design patterns are presented and alternative solutions to each pattern are also reviewed.

Surendra Kumar [158] describes an implementation of object-oriented programming to the finite element method for engineering analysis using C++, and illustrates the advantages of this approach. It was reported that the object-oriented FEM code "FACS" can be used to solve a considerable range of general kinds of engineering problems in the fields of structural analysis, heat transfer and metal working.

McKenna et al.[106] presented a new object-oriented architecture for nonlinear finite element analysis in Structural Engineering. The goal of flexibility, extendability and portability of finite element software are achieved by using the concept of object composition.

These literature are presented in tabular form in the Tables 2.1 and Table 2.2.

2.3 Soil constitutive models

The successful use of finite element analysis in geotechnical engineering problems is highly dependent on the constitutive model chosen to represent soil behaviour. Constitutive models represent mathematical models that describe the macroscopic material behaviour that results from the internal constitution of a material. A constitutive model:

–relates stress with strain,

–allows the prediction of the mechanical behaviour of engineering materials,

–is formulated through mathematical equations,

–describes various kinds of ideal material response governed by controlling parameters.

Table 2.1: Literature review on OOP in FE Analysis

Author	Description
Miller [1991, 1992]	Structural engineering software using LISP extension Flavors [111, 112]
Baugh and Rehak [1992]	Architecture for linear static FEA. [19]
Zimmermann et al. [1992]	Software architecture for linear dynamic FEA using . Smalltalk [180, 46]
Dubois-perlin and Zimmermann [1993]	Software architecture for linear dynamic FEA, using C++ [47].
Menetrey and Zimmermann [1993]	Nonlinear static analysis with J2 plasticity, using C++. [108]
Zeglinski et al. [1994]	A generalized matrix library for FEA, using C++. [175]
Miller et al. [1995]	Framework for interactive modeling of geomechanics problems. [113]
Adeli and Yu. [1995]	Database management system for linear elastic FEA, using C++. [3]
Lu et al. [1995]	Architecture for FEA program for structural engineering computing, using C++. [97]
Miki and Murotsu [1995]	Structural and geometric analysis methods for truss structures, using Smalltalk. [110]
Kong [1996]	A design strategy for the data structure of FEM program, using C++. [79]
Zimmermann and Eyheramendy [1996]	FE environment for symbolic derivations and automatic programming in Smalltalk. [181, 53]
Besson and Foerch [1997]	A core library for mathematical operations, input file and string manipulation tasks, using C++. [20]
Boris and Stein [1998]	Programming tool for complicated tensorial formulae in solid mechanics problems, using C++. [29]
Dubois-Perlin and Pegon [1998]	A set of classes for linear and nonlinear models together with solution control, using C++. [48, 49]
Eyheramendy and Zimmermann [1998]	Automatic generation of FE in a symbolic object-oriented environment in Smalltalk. [54]
Zimmermann et al. [1998]	Environment for reasoning and advanced graphical tools, using C++ and Smalltalk. [182]

Table 2.2: Literature review on OOP in FE Analysis(cont.)

Author	Description
Archer et al. [1999]	Modeling and simulation of nonlinear static and dynamic structural behavior, using C++. [7]
Lichao and Ashok [2000]	A framework that allows extension for new analysis types, interpolation, constraints and equation solving algorithms, using C++. [92]
Sampath and Zabaras [2000]	Implementation of a class of adjoin optimization problems for infinite-dimensional design and control of continuum systems. [143]
Zimmermann and Eyheramendy [2001]	Extension of symbolic derivation concepts to nonlinear formulations with its application to solve Navier-Stokes equations. [55]
Jeremic and Yang [2002]	A framework for computations in geomechanics and creation of template material models. [74]
Modak and Sotelino [2002]	A framework for the time history analysis of structures under dynamic loads. [116]
Martha [2002]	An FEA system for modeling of strain localization, the nonlinear analysis of structure, etc., using C++. [100]
Mackerle [2004]	A bibliographical review of the OOP applied to the FEM and the boundary element method between 1990 and 2003. [101]
Kromer et al. [2004]	A multi-body systems analysis code for treatment of three dimensional multi-body flexible mechanisms. [86, 87]
Boyan and Charles [2005]	Design and implementation of an OOP based FE code for large geomechanics simulations. [30]
Franco et al. [2006].	FE technique using OOP for the limit analysis of axisymmetrical pressure vessels. [58]
Scott et al. [2008]	A software framework for simulating nonlinear response of beam-column members. [144]
Heng and Mackie [2009]	Use of software design patterns to capture best practices in object-oriented FE programming. [66]
Surendra Kumar [2010]	FEM code "FACS" for structural analysis, heat transfer and metal working, using C++. [158]
McKenna et al. [2010]	An OOP architecture for nonlinear FEA in Structural Engineering. [106]

Starting from simple linear elastic to more complex advance models, numerous formulations have been proposed in the field of soil constitutive modeling which are available in the soil mechanics literature. A good review of such works till 1996 can be found in [135]. In the recent past also many advanced constitutive models have been proposed. Brief review of a few recent constitutive models are presented below.

Yu [174] proposed a simple, unified critical state constitutive model for both clay and sand. The model is called as CASM (Clay And Sand Model) and it is formulated in terms of the state parameter that is defined as the vertical distance between current state (v, p) and the critical state line in v-$ln(p)$ space.

Callari et al. [34] presented a finite-strain plasticity model for soft clays. To motivate such a model, the infinitesimal-strain assumption is shown to be inadequate for the constitutive description of soft clays. Hence, assuming the multiplicative elasto-plastic decomposition of the deformation gradient, a finite-strain Cam clay model is presented. In particular, with respect to the original Cam clay formulation, this model improves the description of the isotropic compression behaviour as well as of the elastic shearing response. The constitutive laws are discussed and their implications are pointed out. The physical meaning of the model parameters is carefully addressed.

Gajo and Muir Wood [59] developed a simple kinematic hardening model to represent the general multiaxial stress-strain behaviour of granular materials over the full range of void ratios and stress level (neglecting grain crushing). The model is developed based on bounding surface and kinematic hardening plasticity, which is based on a single set of constitutive parameters, namely two for the elastic behaviour plus eight for the plastic behaviour, which all have a clear and easily understandable physical meaning.

Diaz et al. [44] introduced a system, which allows a time-dependent management of the individual stages of the construction by a dynamic model in consideration of the ground properties, the geotechnical construction elements and the construction processes. An automatic mesh generation enables the generation of hexahedron meshes for the finite-element representation. The implementation of the system was achieved using object oriented modeling techniques and implementation in C++.

The model proposed by Yin et al. [173] is an elastic visco-plastic model for time dependent behaviour of normally and over-consolidated clays. The model is essentially the Cam clay model if the viscous part is ignored. The model can be used for both normally consolidated and over consolidated soil. The model can incorporate the time dependent behaviour of soil and so it can be used to analyze the long term stability of cut slopes.

Simon et al. [152] proposed an anisotropic elasto-plastic model for soft clay (Normally to lightly over-consolidated). The model incorporates a rotational component of hardening to account for the influence of plastic anisotropy in soft clay. It was reported that the model is less suitable for analyzing problems such as excavations and also for problems involving cyclic loading.

The model proposed by McDowell et al. [105] is a non-associated three-surface kinematic hardening model which has been shown to be capable of modeling the effects of recent stress history in clays, and it can reproduce much of the anisotropic behaviour of soil with a simple isotropic state boundary surface. The new model can better predict the response of soil under cyclic loading. Hence there is a prospect for this model to be used in dynamic analysis of stability slopes.

The model proposed by Tamagnini [159] is an extended Cam clay model for unsaturated soils with hydraulic hysteresis. The model depicts the role of hydraulic hysteresis in the mechanical behaviour of unsaturated soils. Gallipoli et al. [60] also proposed a model for unsaturated soil to incorporate the effects of suction and degree of saturation on the mechanical behaviour of soils.

Georgiadis [62] proposed a new yield surface expression in the deviatoric plane which can equally be applied to loose or dense soils. The proposed expression also allows the yield surface to change its shape as shearing progresses and reduces to Mohr-Coulomb hexagon once critical state is reached. But the expression proposed should be used in conjunction with a constitutive model, which allows the variation of the triaxial compression angle of shearing resistance with the strain level. A model of this type was described and used by Potts et al. [133].

The model proposed by Sun et al. [157] is an anisotropic hardening elasto-plastic model for clays and sands. The model is capable of describing deformation and

strength characteristics of clays and sands with the initially anisotropic consolidation state in 3D stress. For contractive soils (showing negative dilatancy), the model parameters involved in the proposed model are same in the Cam clay model and for dilative soil, one more model parameter is to be added. But the approach to get this new model parameter is not available in the cited paper. Again, if the initial consolidation is done isotropically, the model is essentially the original Cam clay model.

Yamamuro and Kaliakin [171] have focused on existing, widely accepted soil constitutive models in this geotechnical special publication. Overall model framework, calibration procedures and possible implementation of different models are also presented. Features of different classes of soil models and their advantages and disadvantages are discussed in an overview paper of this publication.

Ling and Yang [93] have presented a unified generalized plasticity model using a nonlinear critical state line to simulate the stress-deformation behavior of sand of different densities and pressure levels, under both drained and undrained conditions and it has been validated against both monotonic and cyclic loading on sand.

Liu et al. [96] have developed a constitutive model for soil-structure interface using the concept of critical state soil mechanics. The model is capable of describing the basic features of sand-structure interface and is extended successfully to unify the interface behaviors with different roughness. The performance of the model is verified with various experimental results.

Abelev et al. [2] have presented an elasto-plastic single hardening model to capture the behavior of cross-anisotropic frictional materials. The model is constructed for cases where the principal stress and material symmetry axes are collinear and no significant rotation of principal stresses occur. To verify the capabilities of the proposed model, series of true triaxial and isotropic compression tests on dense Santa Monica Beach sand are done. The model incorporates fourteen parameters that can be determined from simple laboratory tests.

Zhang and Zhou[176] have developed a chemo-plastic constitutive model for partially saturated soils based on the existing models developed by Hueckel [70] and Gallipoli et al. [60]. The model also considered softening.

Lade et al. [89] have formulated a rotational kinematic hardening model to predict the behavior of soil during 3-D stress-reversals. The model is also capable of handling cross-anisotropic behavior along with the kinematic hardening mechanism to capture inherent anisotropy of sand in addition to the large stress-reversals. To verify the capabilities of the proposed model, series of true triaxial tests with large stress-reversals are performed on loose cross- anisotropic Santa Monica Beach sand. The model has thirteen parameters, which can be determined from simple laboratory tests.

Andrianopoulos et al. [6] have presented a bounding surface plasticity model for the simulation of monotonic and cyclic loading of non-cohesive soils using the framework of critical state soil mechanics.

Babu and Chouksey [16] have presented an analytical model for the analysis of fiber-reinforced soil in the framework of modified Cam clay model (MCCM). The model is verified using experimental results from the standard undrained triaxial tests with pore water pressure measurements. Tests have been conducted on clayey soil specimens reinforced with randomly oriented discrete coir fibers with different percentages of fiber contents. Numerical simulations of triaxial compression tests on fiber-reinforced clay specimens are also performed. The results presented demonstrate the applicability of proposed model in predicting the stress strain behavior of fiber-reinforced soil. The model is a modified form of the Cam clay model.

Chang and Yin [35] have presented a model to incorporate the inherent anisotropy of soil, based on micromechanics framework. A series of true triaxial tests on Santa Monica Beach Sand is presented to simulate and illustrate the formulation. The directional dependence of the strength characteristics for samples with inherent cross anisotropy and the orientations of failure planes between particles are also examined.

Hong-en et al. [68] have reviewed generalized plasticity theories and most recent developments of generalized plasticity models for geotechnical problems. It is reported that elasto-plastic constitutive models developed based on generalized plasticity have been successfully utilized in modeling the mechanical behavior of both saturated and unsaturated soils.

Uchida et al. [162] have proposed a new constitutive model for simulating the

mechanical behavior of methane hydrate-bearing soil based on the concept of critical state soil mechanics. The model is referred to as the Methane Hydrate Critical State (MHCS) model. It is reported that the model is capable of incorporating volumetric yielding and degradation of hydrate effects in soil, along with the stress relaxation due to hydrate dissociation. Five extra model parameters to the conventional critical state model are required in this model.

These literature are presented in tabular form in the following Table 2.3 and Table 2.4.

2.4 Numerical integration of constitutive relations

The numerical solution of any boundary value problem using an elasto-plastic constitutive model need to invoke an iterative and incremental schemes. For every load or time step, given a converged configuration at step n, the discretized momentum balance equations are to be solved to compute a new configuration for step (n+1) via an incremental motion [149].This generally gives the incremental strains $\Delta\epsilon$, at every stress point and corresponding to this strain increment, new values of the state variables at step (n+1) are obtained by integration of the local constitutive equations. Using this new computed state variables, which are nothing but the stresses, balance of momentum is checked and if violated, iterations are performed by returning to the computation of the incremental strain.

Here attention is to be focused on the integration of the local constitutive equations, as it plays the main role in the computations. Hence in the following section, a brief study on the various scheme of numerical integration of elasto-plastic model implementation in the field of geotechnical analysis is done with special emphasis on critical state models. In the subsequent sections integration algorithms for other soil constitutive models and general plasticity models are also presented.

Table 2.3: Literature review on soil constitutive models

Author	Description
Yu [1998]	A simple, unified critical state constitutive model (CASM) for both clay and sand. [174]
Callari et al. [1998]	A finite-strain Cam clay model for soft clays that improves the description of the isotropic compression behaviour as well as of the elastic shearing response. [34]
Gajo and Muir Wood [1999]	A kinematic hardening model to represent the general multiaxial stress-strain behaviour of granular materials. [59]
Diaz et al. [2000]	A dynamic model for time-dependent FE analysis of geotechnical engineering problems. [44]
Yin et al. [2002]	Elastic visco-plastic model for time dependent behaviour of normally and over-consolidated clays. [173]
Simon et al. [2003]	Anisotropic elasto-plastic model for soft clay incorporating a rotational component of hardening. [152]
McDowell et al. [2003]	Non-associated three-surface kinematic hardening model, capable of modeling the effects of recent stress history in clays [105]
Tamagnini [2004]	An extended Cam clay model for unsaturated soils with hydraulic hysteresis. [159]
Gallipoli et al. [2004]	A model for unsaturated soil to incorporate the effects of suction and degree of saturation on the mechanical behaviour of soils. [60]
Georgiadis [2004]	An yield surface expression in the deviatoric plane which can be applied to loose or dense soils. [62]
Sun et al. [2004]	Anisotropic hardening elasto-plastic model, capable of describing deformation and strength characteristics for clays and sands. [157]
Yamamuro and Kaliakin [2005]	Focused on existing, widely accepted soil constitutive models in context to overall model framework, calibration procedures, possible implementation, and their advantages and disadvantages. [171]
Ling and Yang [2006]	A unified generalized plasticity model using a nonlinear critical state line to simulate the stress-deformation behavior of sand. [93]
Liu et al. [2006]	A constitutive model for soil-structure interface using the concept of critical state soil mechanics. [96]

Table 2.4: Literature review on soil constitutive models (cont.)

Author	Description
Abelev et al. [2007]	An elasto-plastic single hardening model to capture the behavior of cross-anisotropic frictional materials. [2]
Zhang and Zhou [2008]	A chemo-plastic constitutive model for partially saturated soils based on the existing models developed by Hueckel [70] and Gallipoli et al. [60]. [176]
Lade et al. [2009]	A rotational kinematic hardening model to predict the behavior of soil during 3-D stress-reversals. [89]
Andrianop -oulos et al. [2010]	A bounding surface plasticity model for the simulation of monotonic and cyclic loading of non-cohesive soils using the framework of Critical State Soil Mechanics. [6]
Babu and Chouksey [2010]	A model for the analysis of fiber-reinforced soil in the framework of MCCM. [16]
Chang and Yin [2010]	A model to incorporate the inherent anisotropy of soil, based on micromechanics framework. [35]
Hong-en et al. [2011]	Reviews of generalized plasticity theories and most recent developments of generalized plasticity models for geotechnical problems. [68]
Uchida et al. [2012]	A model for simulating the mechanical behavior of methane hydrate-bearing soil based on the concept of critical state soil mechanics. [162]

The methods used to integrate the constitutive equations can be broadly divided into two category: explicit and implicit methods. Both explicit and implicit methods have been used to integrate many advanced constitutive models in soil mechanics. It is reported that [98] fully implicit integration schemes are very robust and efficient for nonlinear elasto-plastic and elastic-visco-plastic models and enjoy widespread use in finite element formulations.

2.4.1 Numerical integration: Cam clay group of models

Following are the literature review on numerical integration of constitutive relations for soil developed in the critical state framework.

A discussion of various implicit integration methods for critical state models can be found in Borja and Lee [24]. Numerical examples on the implementation of the

scheme for the MCCM were also presented. The integration rules used fall under the category of return mapping algorithm in which the return directions are computed by closest point projection for associated flow rule and by central return mapping for non-associated flow rule applied to the Cam clay ellipsoids. Stress updates takes place at the Gauss points upon the enforcement of the consistency condition in which the appropriate parameters are determined iteratively on the scalar level.

An implicit integration method of constitutive equations based on a nonlinear stress predictor was explained by Borja [25].

Hashash and Whittle [65] presented full details for implementing the MCCM equations in a nonlinear finite element analysis using an implicit integration scheme with consistent linearization of the tangent constitutive matrix. Numerical examples of some geotechnical boundary value problems were presented to illustrate the accuracy and robustness of the algorithm.

A brief description and a comparative study of the sub-stepping schemes and return mapping schemes at a fundamental level can be found in Potts and Ganendra [132].

Luccioni et al. [98] applied backward Euler integration scheme to Bear-clay model proposed for lightly consolidated cohesive materials. They used (BFGS method) quasi-Newton method to solve the global finite element equations. The algorithm required the iterative solution for both the predictor and corrector steps. They calculated the tangent matrix numerically using finite difference method. In the implementation, they calculated tangent matrix only once for each load step. Robustness, accuracy and efficiency of the presented algorithm were carried out with numerical simulation of single element test and strip footing.

Sheng et al. [146] has discussed some practical aspects of the finite element implementation of critical state models. They have presented a few automatic algorithm schemes for stress integration and load and time stepping, which are tested with two types of critical state models.

Borja et al. [27] presented a fully implicit integration algorithm for a class of anisotropic bounding surface plasticity models with ellipsoidal loading function. The plasticity model is coupled with a nonlinear hyperelastic model to ensure that the elastic component of the combined model is energy-conserving. A key feature of the

integration algorithm is a return mapping in strain space, which allows fully implicit integration and consistent linearization of the constitutive equations. For this class of bounding surface models the consistency condition on the bounding surface is shown to be mathematically equivalent to the consistency condition on the loading surface. Isoerror maps are generated describing the accuracy of the integration algorithm on the stress-point level. Finally, a boundary-value problem involving a strip footing on lightly overconsolidated clay is analyzed to demonstrate the robustness of the algorithm in a finite element setting.

Luccioni et al. [99] examined the performance of explicit integration scheme with sub-incrementation with automatic error control for Bear-clay model. They also studied the performance of global iterative scheme. The fully explicit scheme presented did not require the determination of yield surface and drift-correction from the yield surface. They presented single element test and rigid strip footing.

A refined explicit integration scheme for elasto-plastic models with automatic substepping and error control was described by Sloan et al. [156].

Rouainia and Wood [141] presented a fully implicit scheme for a simple kinematic hardening extension of the Cam clay soil model. The algorithm is based on the operator split methodology and the implicit Euler backward integration scheme is used to integrate the rate constitutive relations.

Hickman and Gutierrez [67] proposed an algorithm to integrate Cam clay and MCCM making use of linearity of the virgin isotropic compression curve and the parallel anisotropic consolidation line in $e - lnp$ space. They employed implicit method of integration. Hydrostatic compression test and undrained triaxial compression test simulation were carried out to validate the accuracy and efficiency of the proposed algorithm.

Zhao et al. [178] extended the earlier presented explicit integration scheme to two complex soil models. These models are the sub/super loading surface model(SSLM) and the kinematic hardening structures model(Khsm) which are employed for structured clays and sands. Explicit Euler method with automatic substepping and error control was used. Efficiency and accuracy of the proposed algorithm were verified by presenting triaxial compression test under drain condition and footing condition.

Peric [130] derived an analytical solution for three-invariant Cam clay model considering infinitesimal strain theory under fully saturated condition. Two different loading history namely proportional and circular load histories under drained condition were considered. They observed that the soil exhibited qualitatively similar behaviour under drained and undrained circular loads, but they are quantitatively different.

Amorosi et al. [5] presented operator split based integration scheme for Cam clay based plasticity model with isotropic and kinematic hardening rule for saturated clays. They also presented consistent tangent matrix. Accuracy and stability of the algorithm was examined by carrying out single element test and analysis of strip footing. These literature are presented in tabular form in the Table 2.5.

2.4.2 Numerical integration: constitutive relations for soil

Following are the literature review on numerical integration of general constitutive relations for soil.

Perez-Foguet et al. [124] applied numerical differentiation to obtain the jacobian matrix while using Newton-Raphson method in the integration of constitutive equations at each of the Gauss point. The same technique was also used for the calculation of tangent stiffness matrix. They calculated these matrices analytically first and obtain flow vector. The derivative of the flow vector were obtained by numerical differentiation and substituted in the derived matrices. They claimed that the approach required only marginal computational overhead$(1-2\%)$. The error analysis of these differentiation methods was discussed and superiority of the present algorithm for von Mises and modified Mohr-Coulomb method has been pointed out.

Perez-Fouget et al. [125] extended their method of numerical calculation of local and global matrices to MRL-Lade model. They achieved the quadratic rate of convergence at both local and global level. They solved many geotechnical problems to illustrate the efficiency and accuracy of the algorithm.

Table 2.5: Literature review on Numerical Integration of Cam clay group of models

Author	Description
Borja and Lee [1990]	Implicit integration methods based on return mapping algorithm. [24]
Borja [1991]	Implicit integration method based on a nonlinear stress predictor. [25]
Hashash and Whittle [1992]	Implicit integration scheme with consistent linearization of the tangent constitutive matrix. [65]
Potts and Ganendra [1994]	Comparative study of sub-stepping and return mapping scheme at fundamental level. [132]
Luccioni et al. [2000]	Implicit integration scheme for lightly consolidated cohesive materials. [98]
Sheng et al. [2000]	Explicit integration schemes with sub-incrementation.[146]
Borja et al. [2001]	Fully implicit integration algorithm for a class of anisotropic bounding surface plasticity models. [27]
Luccioni et al. [2001]	Explicit integration scheme with sub-incrementation with automatic error control [99]
Sloan et al. [2001]	Explicit integration scheme with automatic substepping and error control. [156]
Rouainia and Wood [2001]	Implicit integration scheme based on the operator split and the backward Euler scheme. [141]
Hickman and Gutierrez [2005]	Implicit algorithm based on linearity of the virgin isotropic compression curve and the parallel anisotropic consolidation line in $e - ln\ p$ space. [67]
Zhao et al. [2005]	Explicit Euler method with automatic substepping and error control [178]
Peric [2006]	Analytical solution for three-invariant Cam clay model considering infinitesimal strain theory under fully saturated condition. [130]
Amorosi et al. [2008]	Fully implicit operator split based integration scheme for structured clays. [5]

Perez-Foguet et al. [126] presented an expression for consistent tangent matrices considering substepping in the integration of constitutive relations. The expression consistent tangent is shown to have the same structure of the corresponding scheme without substepping. The various aspects of the implementation have been illustrated by considering hyperbolic Mohr-Coulomb model. In their implementation, substepping was activated only at those Gauss points where more than prescribed iterations were needed. They have provided consistent tangent matrices for generalized midpoint rule and backward Euler method with substepping.

Borja et al. [28] presented a numerical algorithm for the integration of isotropically hardening three-invariant elasto-plastic constitutive models and investigated the performance of the Mohr-Coulomb, Lade-Duncan and Matsuoka-Nakai models within the framework of the proposed algorithm. The algorithm is based on a spectral representation of stresses and strains for infinitesimal and finite deformation plasticity with return mapping in principal stress directions. Some specific features such as hardening/softening responses and the tapering of the yield surfaces toward the hydrostatic axis with increasing confining pressure are incorporated into the formulation. To investigate the performance of the integration algorithm a boundary-value problem involving loading of a strip foundation on a soil is analyzed with and without finite deformation effects.

An implicit return mapping scheme based on closest point projection was proposed by Wang et al. [168]. The algorithm has been developed for soil models which involve the stress invariants I_1, J_2 and J_3 and has been implemented for soil models such as Mohr-Coulomb and Matsuoka-Nakai. In this algorithm, two stage iteration scheme is adopted where the scalar consistency parameter ϕ is calculated from the first-stage iteration implicitly. The second-stage iteration is used to calculate the unknown stress components once ϕ is known. The corresponding tangential moduli, consistent with the return-mapping algorithm are then calculated by solving a group of linear equations, which are defined by the converged stresses and state variables. Problems such as sequential excavation subject to large strain, embankment construction with water and soils fully coupled, and propagation of shear band localization in a triaxial test, are solved using the algorithm.

Clausen et al. [36] obtained the closed form scheme for integration of linear isotropic yield criterion. They employed an implicit method of integration and illustrated the

method with Mohr-Coulomb yield criterion. They treated the algorithm in the principal stress space and derived the expression for consistent tangent modulus. The treatment of discontinuity of the yield surface was also examined.

Justino et al. [75] applied an operator split technique for porous sintered materials and performed comparative analysis of various models. The algorithms were implemented in the ABAQUS, and efficiency and accuracy of the algorithm were examined by presented uniaxial tension test and bar with a central hole.

Wang et al. [167] presented an algorithm for the integration of elasto-plastic constitutive relations with rotational hardening employing substepping. They used the substepping technique in both the implicit integration of constitutive relations and the calculation of consistent tangent matrix. The inverse of the local Jacobian of the converged step was used to obtain the consistent tangent matrix. The stability and efficiency of the proposed algorithm were examined by presenting a rigid footing problem and a plane strain compression test.

Yield surfaces with corners induce singularity in the formulation of elasto-plasticity problems. Krabbenhoft el al. [82] investigated the application of conic programming to plasticity problems. They have reviewed some of the more common cone and they demonstrated their applicability to the Drucker-Prager, Mohr-Coulomb and Neilsen criteria. It has been concluded that the three dimensional Mohr-Coulomb criterion could be cast as a set of conic constraints. The performance of the suitable conic programming algorithm for solving needs to be evaluated.

Pedroso et al. [127] presented an explicit algorithm based on the Runge-Kutta embedded method of second order accuracy for elasto-plastic model with nonconvex yield surfaces. The crossing of the yield surface is taken into account based on the Kronecker-Picard formula. Nonconvex yield surfaces considered are a yield surface with the cardioid shape, Barcelona basic model for unsaturated soils, Cam clay model with modification. The accuracy and efficiency of the presented algorithm is examined with the aid of isoerror map and solving boundary value problem. They found that the solution for nonconvex yield surface was much slower than that for similar models with convex yield surfaces mainly due to the higher order gradients required for the the computation of the number of intersections.

Sheng et al. [148] developed the modified Euler scheme with an automatic sub-incrementation scheme based on local error estimate for non-convex yield surfaces used for the modeling of unsaturated soil. They employed the Kronecker Picard formula for the intersection of yield surfaces.

Yang et al. [172] presented the generalized trapezoidal rule for middle surface concept(MSC) sand model. This model simulates different soil response characteristics using different pseudo-yield function. They also derived the consistent tangent matrix. The global integration using consistent tangent operator performed better in convergence when $\alpha \geq 0$. But integration of the constitutive equation performed with higher rate of convergence when $\alpha < 0$. The accuracy was found to be the highest at $\alpha = 0.5$.

Zhang and Zhou[176] presented backward Euler integration algorithm to a chemo-plastic constitutive model for partially saturated soils. The model considered the softening. They also derived the consistent tangent matrix. The efficiency and accuracy of the algorithm were examined by carrying out numerically isotropic compression test with suction increasing, constrained compression test with suction decreasing, suction decreasing with chemical effect and finite element analysis of consolidation problem.

Mira et al. [114] presented a generalized midpoint algorithm for the integration of a generalized plasticity model for sand(SandPZ model) where hyper-elasticity based formulation was used for elastic part. The consistent tangent matrix was also derived. Here iteration is carried out in the elastic strain space instead of stress space. Thus the present algorithm iterates in elastic strain, plastic modulus and plastic multiplier. Drained triaxial test and drained cyclic loading were carried out to test the efficiency and accuracy of the presented algorithm.

Andrianopoulos et al.[6] implemented explicit scheme with automatic error control and substepping for the stress integration of a two surface bounding surface plasticity model for the monotonic and cyclic behaviour of non-cohesive soils. The algorithm was implemented in a commercial software FLAC via userdefined model. They pointed that the use of two-step modified Euler scheme led to integration error lower than those using the classical form of one-step forward Euler scheme. To show the accuracy and efficiency of the algorithm, single element triaxial test under cyclic

loading along complex stress path and centrifuge experiment system response due to earthquake induced liquefaction were carried out.

Ngygen and Einav [119] developed an implicit integration algorithm for nonlocal inelastic models. They considered two nonlocal models; plasticity coupled with damage and plasticity-breakage model. Importance of regularization of constitutive relations for softening material in the implementation was demonstrated. It was pointed out that the stress update process required the construction of a nonlocal constitutive matrix. The effectiveness of the algorithm was provided by considering granular crushing of sand sample subjected to compressive loading.They have also stressed that the form of the loading function plays an important role in regularizing the constitutive equations using nonlocal theory.

Krabbenhoft and Lyamin [84] extended the work of Krabbennhoft et al. [83] to consider the implantation of modified Cam clay plasticity. They presented an incremental variational formulation whose discrete approximation provides a second order cone programming. The advantages of this approach are that there is no need of special technique to treat yield surface singularity. The efficiency and robustness of the method was verified by solving (a) triaxial tests with the drained tests, undrained tests (b) biaxial test (c) strip footing and (d) excavation. They obtained that the global Newton iterations required was higher than the conventional methods. The treatment of softening plasticity problem was also discussed.

These literature are presented in tabular form in the following Table 2.6 and Table 2.7.

2.4.3 Numerical integration: general plasticity models

Following are the literature review on numerical integration of general plasticity models.

Lee and Fenves [90] presented return-mapping algorithm for plastic damage model in the principal stress space. Scalar damage model was considered. As most of the yield criteria are written in the principal stress or stress invariant, it is convenient to work stress integration algorithm also in the principal stress. From the

spectral decomposition of stress, transformation matrix was obtained. Consistent tangent matrix was derived. Numerical examples were carried out to demonstrate the performance of the algorithm. They also discussed the plane stress case.

Table 2.6: Literature review on numerical integration of soil constitutive relations (implicit)

Author	Description
Perez-Foguet et al. [2000]	Applied numerical differentiation to obtain the jacobian matrix and tangent stiffness matrix while using Newton-Raphson method in the integration of constitutive equations at each of the Gauss point. [124]
Perez-Fouget et al. [2000]	Extended their method of numerical calculation of local and global matrices to MRL-Lade model. [125]
Borja et al. [2003]	Integration scheme for three-invariant elasto-plastic constitutive models in principal stress space and extended to finite deformation plasticity. [28]
Wang et al. [2004]	Implicit return mapping scheme based on closest point projection for soil models and implemented for Mohr-Coulomb and Matsuoka-Nakai models. [168]
Clausen et al. [2006]	Implicit method of integration for linear isotropic yield criterion. [36]
Sheng et al. [2008]	Modified Euler scheme with an automatic sub-incrementation scheme based on local error estimate for non-convex yield surfaces used for the modeling of unsaturated soil. [148]
Yang et al. [2008]	Generalized trapezoidal rule for middle surface concept(MSC) sand model and also derived the consistent tangent matrix. [172]
Zhang and Zhou [2008]	A fully implicit backward-Euler integration algorithm for chemo-plastic constitutive model for partially saturated soils. [176]
Ngygen and Einav [2010]	An implicit integration algorithm for nonlocal inelastic models. [119]

Table 2.7: Literature review on Numerical Integration of soil constitutive relations (explicit)

Author	Description
Perez-Foguet et al. [2001]	Consistent tangent matrices considering substepping in the integration of constitutive relations. [126]
Justino et al. [2006]	An operator split technique for porous sintered materials with comparative study of various models. [75]
Wang et al. [2006]	Algorithm for the integration of elasto-plastic constitutive relations with rotational hardening employing substepping. [167]
Krabbenhoft et al. [2007]	Application of conic programming to plasticity problems and three dimensional Mohr-Coulomb criterion was casted as a set of conic constraints. [82]
Pedroso et al. [2008]	An explicit algorithm based on the Runge-Kutta embedded method of second order accuracy for elasto-plastic model with nonconvex yield surfaces. [127]
Mira et al. [2009]	A generalized midpoint algorithm for the integration of a generalized plasticity model for sand, where hyper-elasticity based formulation was used for elastic part. [114]
Andrianopoulos et al. [2010]	An explicit integration scheme with automatic error control and substepping, and was applied for a bounding surface plasticity model. [6]
Krabbenhoft and Lyamin [2012]	An incremental variational formulation whose discrete approximation provides a second order cone programming. [84]

Ritto-Correa and Camotion [140] extended the return mapping algorithm for plane stress proposed by Simo and Taylor to the general case of mixed stress-strain control. They employed backward Euler scheme and derived consistent tangent matrix for J_2 elasto-plasticity. Accuracy of the implementation was examined by providing isoerror map and numerical simulation based on beam theory.

Wallin and Ristinmaa [164] presented a stress updating algorithm based on the assumption of constant strain rate increment during the time step. They used explicit Runge-Kutta method for integration and derived corresponding algorithmic tangent stiffness matrix. In the derivation of tangent stiffness matrix they employed approx-

imation in the calculation of derivatives. The algorithm applied to two models; one with isotropic hardening and second one with isotropic and kinematic hardening as well as damage evolution. They observed that damage evolution is very sensitive to size of load step for the usual case of closest point projection method.

Armero and Perez-Foguest [8, 129] discussed the variation structure and issue related to global convergence of Backward Euler(Closest point projection method) in their two part papers. Firstly they obtained the variation structure behind closest point projection algorithm and discussed the primal-dual problem. These structure facilitated them to develop a new closest point projection method. They considered both small strain as well as large strain elasto-plasticity and visco-plasticity.

Buttner and Simeon [33] presented implicit Runge-Kutta methods for integration of constitutive relations. The constitutive relations were treated as differential algebraic equations. Although formulations presented for general constitutive relations, the implementation is restricted to J_2 plasticity. The challenge of effective step-size control was discussed. The contractivity property of general constitutive relations and its numerical solution by Runge-Kutta method were studied. Several problems were solved to examine the efficiency and accuracy of the algorithm.

Keavey [76] first converted the elasto-plasticity based constitutive equations to a set of first order differential equation in the canonical form $\dot{x} = f(X)$. These equations were integrated and resulting integrated equation together with consistency algebraic equation was solved using iterative method. He also derived the general procedure to derive consistent tangent matrix. They used numerical techniques. This algorithm was claimed to be general for any elasto-plastic constitutive relation, but it was implemented only for von Mises plasticity model. The efficiency and accuracy of the algorithm was verified by solving thin tube in tension and torsion, and cylinder under internal pressure.

Auricchio and Da Veiga [15] presented a new integration scheme based on the computation of a model integration factor for von Mises plasticity with linear hardening. They also compared the performance of the presented model with radial return map algorithm. They found that the presented method showed a quadratic error convergence in relation to the discrete time interval length which the radial return map algorithm showed a linear behaviour. The comparison was carried out with the ex-

amples, pure tension,pure torsion and plate with hole. In this approach they first reduced the plasticity based constitutive relation to $\dot{X} = AX$ form where X is a vector and A is a matrix.

Chocchetti and Perego [37] presented method to obtain the error involved in backward difference time integration scheme for elasto-plasticity based models. The proposed error indicator was the non-negative difference between the exact total, work increment calculated by a backward difference integration. The algorithm was examined with perfectly plastic associated constitutive models. They also investigated the possibility of using the proposed error estimate in adaptive times step control. They solved problem of plate with hole under cyclic loading to examine the algorithm. The proposed error estimate is based on the work of Ortiz and Martin.

A non-uniform model with transient and steady strain rates plays an important role in forming process. Transient and steady state strain rates are dependent on both time and temperature.Backward Euler implicit integration scheme was employed to integrate such non-uniform constitutive model by Kobayashi et al. [77]. They also derived the consistent tangent matrix. The discussed the difficulties in implementation. Accuracy and efficiency were investigated by solving problems and comparing the results.

Eckert et al. [51], treating displacement as well as internal variables as global variable in finite element analysis, obtained differential algebraic equations. They employed multi-step backward difference method (BDF2) to integrate the differential equations. In this algorithm, proper meaning of consistent tangent matrix is explained. They used predictor-corrector method to solve the global differential algebraic equation. A plate with hole under plane stress condition was presented to investigate the proposed method.

Liu [94] presented an integral formulation and group theoretic formulation for von Mises plasticity model with mixed hardening. He first established the internal symmetry group of the mixed hardening elasto-plastic model and consequently derived a quasilinear system with only three dimensions. The concept of response subspace was introduced in which a Minkowski space time could be endowed. Two integration schemes derived were based on Volterra integral equation and on the a group symmetry.

Montans [117] extended the backward Euler scheme for the integration of multilayer plasticity based constitutive relations for the plane stress case. They also explicitly derived the consistent tangent modulus. The plane stress formulation adopted is based on the plane stress projection algorithm of Simo and Taylor. Multilayer J_2 Plasticity with Prager-Ziegler transition rule was considered for implementation. He examined the algorithm with stress point integration and solving flexure of beam.

Artioli et al. [9] compared the three exponential map algorithm for von Mises plasticity with linear hardening with radial return map algorithm. Two of these algorithm were the previously presented algorithm and modified version of that. The third one was based on a formulation of the continuous plasticity model which automatically satisfied the yield consistency. They found the behaviour of the second method to be superior to other two.

Liu and Li [95] extended the method of integral representation and geometric integrator to the plasticity model with the Armstrong-Frederick kinematic hardening rule. They presented two geometric integrator for strain control and stress control respectively. The methods are more accurate, more stable and effective than the radial return method because they are sharing the geometric structures and invariance of plasticity equations. The integrator used employed a discretization of integral representation and an exponential approximation of the quasi-linear differential equations system for relative stress.

Artioli et al. [10] modified their earlier presented exponential based integration algorithm for von Mises plasticity with linear hardening. They also presented the accuracy and stability analysis of the new algorithm. They called new algorithm as optimal exponential symmetric consistent method(ESC^2). They examined the numerical accuracy of the method with isoerror map in pure shear and pure tension and solving boundary value problem of plate with hole under plane stress condition.

Kulkarni et al. [88] presented a Newton-Schur approach which were claimed to be an alternative to consistent tangent approach in Newton method. They also showedthat the structure of the consistent tangent approach was preserved in the proposed method. In the Newton-Schur approach, the local constitutive iterations are not required.

Artioli et al. [11] discussed generalized midpoint integration algorithm for von Mises plasticity with linear hardening. Two of the algorithm are single step whereas other two are double step. They also derived the corresponding consistent tangent matrix. They discussed the B-stability of the algorithm and symmetry of the tangent matrix. Accuracy and efficiency of these algorithm were analyzed by carrying out point-wise stress strain analysis and solving boundary value problem of plate with hole.

Artioli et al. [12] compared two second order numerical scheme namely generalized midpoint scheme and exponential based scheme(ESC^2) presented earlier. They consider linear isotropic hardening and nonlinear Armstrong-Frederick kinematic hardening rule. They concluded that both the scheme has similar numerical behaviour. They pointed the higher precision of exponential based method wherever there was crossing of the yield surface.

Ding et al. [45] compared the accuracy and efficiency of various explicit method with implicit method (backward Euler) for sheet metal forming considering von Mises yield criterion and Hill's anisotropic yield criterion. They presented two substepping schemes, modified Euler and Runge-Kutta Dormand Prince scheme with error control enhanced by the stress correction procedure. They also implemented forward Euler algorithm with subincrementation. These algorithms were implemented in ABAQUS. Higher order Runge-Kutta method was found to be superior to backward Euler scheme and modified Euler scheme when tight tolerance was employed. In most cases, they found that explicit method within stress correction performed with the same level of accuracy as backward Euler scheme.

Krabbenhoft et al. [83] employed convex programming theory and algorithms to treat the problem of small deformation rate independent plasticity. To achieve this, the problem has been formulated in the finite step variational framework. Their finite element implementation is based on a mixed stress-displacement element which, they claimed, yields more accurate result. An interior point based algorithm was employed for small deformation rate-independent elasto-plasticity. They have compared the algorithm with existing implicit method in terms of efficiency, robustness and quadratic rate of convergence.

Rezaiee-Pajanal and Nasirai [139] presented two exponential map algorithm for integrating Drucker-Prager's elasto-plastic models. They have also interpreted the

work of Lice in the simplified way. The algorithm proposed is semi-implicit having good accuracy and rapid convergence. Accuracy was examined with isoerror maps.

This work is the extension of their earlier work [164]. The assumption of constant strain rate was made by Wallin and Ristinmaa [165] and constitutive equations were formulated as a set of ordinary differential equations(ODE). These equations were solved using explicit Runge-Kutta method(R-K Method). They have also derived the algorithmic tangent stiffness matrix for R-K scheme. In the derivation of the algorithmic tangent matrix, they obtained a set of ordinary differential equations which they solved using the same R-K Method which was employed for constitutive relations. The algorithm has been implemented for von Mises yield function with nonlinear isotropic hardening and isotropic damage evolution.

Crouch and Askes [42] developed an analytical method by transforming stress to another space called energy mapped stress space. In this space, closest point projection has been shown as the closest point in Eucleadian sense whereas untransformed stress space the distance between the trial stress and the return stress is minimum in the energy matrix. This transformation of stress to the energy mapped stress facilitated to obtain efficient analytical solution. The present algorithm is applied to modified Drucker-Prager yield surface with perfect plasticity. Superiority of the method has been pointed with respect to conventional closest point projection method. Possibility of extension of this method has also been stated.

Vrh et al. [163] applied the forward difference method to integrate the constitutive relation. But they satisfied consistency condition at the the end of the increment by expanding the consistency condition using Taylor's series expansion and retaining only linear term. Thus they obtained an explicit integration scheme. They pointed out that the proposed algorithm maintained explicit nature of the forward difference and satisfied yield criterion approximately as in backward difference method. They implemented the algorithm into ABAQUS via user material subroutine(VUMAT) for von Mises and Gurson-Tvergaard-Needleman(GTN) Model. They also discussed the difficulty in the application of the algorithm to shell theory. Accuracy of the present method was reported to be comparable with the backward Euler integration scheme.

Betteieb et al. [21] presented Backward Euler integration scheme for advanced Gur-

son model which considered kinematic hardening. They also derived the consistent tangent modulus. Efficiency and accuracy of the algorithm were carried out by single element test and tension test with the notched specimen.

Peng and Chen [128] developed backward Euler integration scheme for elasto-plasticity in principal stress. They also derived the consistent tangent modulus. They employed a set of three mutually orthogonal unit base vector in conjunction with a new set of three invariant for the representation of arbitrary isotropic tensor value function and scalar valued function of stress tensor. The integration was carried out in three dimensional space of stress. The proposed algorithm was implemented through UMAT in ABAQUS. They considered the Drucker-Prager yield criterion.

These literature are presented in tabular form in the Table 2.8 and Table 2.9.

2.5 Nonlinear solution techniques and load stepping schemes for critical state models

Nonlinearity in a problem may be broadly divided into three groups: (a) geometric nonlinearity, (b) material nonlinearity and (c) interface nonlinearity. In geometric nonlinearity some of the geometrical parameters of the model can change during the loading process. Geometric nonlinearity needs to be accounted for when the structure deforms to such an extent that the original geometry and/or position and direction of the loads significantly affect the structural behaviour. In case of material nonlinearity some of the constitutive parameters of the model undergo changes during the loading process. Nonlinear elastic, elasto-plastic or visco-plastic material behaviour are some typical examples of this type. Interface nonlinearity is also called as boundary nonlinearity and one typical example is a contact problem where the boundary conditions change during the loading process. Other examples include problems involving friction, impact/energy absorption and metal forming processes. Several literature are available which cover this topic extensively [18, 41].

Table 2.8: Literature review on numerical integration of general plasticity models (implicit)

Author	Description
Lee and Fenves [2001]	A return-mapping algorithm for plastic damage model in the principal stress space. [90]
Ritto-Correa and Camotion [2001]	A return mapping algorithm based on backward Euler scheme for the general case of mixed stress-strain control. [140]
Armero and Perez-Foguest [2002]	Discussed the variation structure and issue related to global convergence of Backward Euler(Closest point projection method) scheme. [8, 129]
Buttner and Simeon [2002]	An implicit Runge-Kutta methods for integration of constitutive relations. [33]
Keavey [2002]	A general integration algorithm, by converting the constitutive equations to a set of first order differential equation. [76]
Auricchio and Da Veiga [2003]	An integration scheme based on the computation of a model integration factor. [15]
Kobayashi et al. [2003]	Backward Euler implicit integration scheme for non-uniform constitutive model in forming process. [77]
Eckert et al. [2004]	Employed multi-step backward difference method (BDF2) and predictor-corrector method. [51]
Artioli et al. [2005]	Compared the three exponential map algorithm for von Mises plasticity with linear hardening with radial return map algorithm. [9]
Liu and Li [2005]	Extended the method of integral representation and geometric integrator to the plasticity model with the Armstrong-Frederick kinematic hardening rule. [95]
Artioli et al. [2007]	Compared two second order numerical scheme namely generalized midpoint scheme and exponential based scheme(ESC^2). [12]
Crouch and Askes [2009]	Developed an analytical method by transforming stress to another space called energy mapped stress space. [42]
Betteieb et al. [2011]	Backward Euler integration scheme for advanced Gurson model which considered kinematic hardening. [21]

Table 2.9: Literature review on numerical integration of general plasticity models (explicit)

Author	Description
Wallin and Ristinmaa [2001]	A stress updating algorithm using explicit Runge-Kutta method. [164]
Chocchetti and Perego [2003]	A method to obtain the error involved in backward difference time integration scheme for elasto-plasticity based models. [37]
Liu [2004]	An integral formulation and group theoretic formulation for von Mises plasticity model with mixed hardening. [94]
Montans [2004]	Extended the backward Euler scheme for multilayer plasticity based constitutive relations for the plane stress case. [117]
Artioli et al. [2006]	Modified exponential based integration algorithm for von Mises plasticity with linear hardening. [10]
Kulkarni et al. [2006]	A Newton-Schur approach as an alternative to consistent tangent approach in Newton method. [88]
Artioli et al. [2007]	Generalized midpoint integration algorithm for von Mises plasticity with linear hardening. [11]
Ding et al. [2007]	Compared the accuracy and efficiency of various explicit method with implicit method (backward Euler) for sheet metal forming. [45]
Krabbenhoft et al. [2007]	Employed convex programming theory and algorithms to treat the problem of small deformation rate independent plasticity. [83]
Rezaiee-Pajanal and Nasirai [2008]	Two exponential map algorithm for integrating Drucker-Prager's models. [139]
Wallin and Ristinmaa [2008]	Extended their earlier work [164] by assuming constant strain rate. and employed explicit Runge-Kutta method. [165]
Vrh et al. [2010]	An explicit integration scheme, using forward difference method to integrate the constitutive relation. [163]
Peng and Chen [2012]	Backward Euler integration scheme for elasto-plasticity in principal stress and applied for Drucker-Prager yield criterion. [128]

Performance of various load stepping schemes and nonlinear solution techniques while used with critical state models was reported to be very sensitive to the load step size and iteration tolerance as well. Gens and Pots [61] compared a number of incremental and iterative schemes and concluded that critical state models are vulnerable to numerical breakdown and may need different solution strategies for different problems.

Abbo and Sloan [1] developed automatic incremental schemes for solving a general class of elasto-plastic problems, which do not involve any iteration. In these methods, the trial load step may be sub-incremented, if necessary, to keep the local load path error below a specified tolerance.

Sheng and Sloan [147] reported that in spite of strong influence of the load stepping schemes on the solution of nonlinear finite element problems, a few comparative studies of them are done while applied to critical state models. They investigated the performance of various load stepping schemes for critical state models in finite element analysis and reported some of the difficulties that can be encountered while applying with these models in practice. The efficiency and robustness of some conventional iterative schemes, such as the Newton-Raphson and modified Newton-Raphson methods was discussed along with some accelerators. The performance of these schemes was reported to be very sensitive to the load step size and iteration tolerance as well. Investigation on the performance of two automatic load stepping schemes presented by Abbo and Sloan [1] was also done and reported to be robust, efficient, accurate, and reliable as compared to other iterative schemes.

Sheng and Sloan [147] also described an automatic incremental-iterative scheme, in which within each sub-increment, an iterative scheme like Newton-Raphson is invoked to dissipate the unbalanced force. These two automatic schemes have proved to be robust when applied with conventional elasto-plastic models, but their performance yet to be studied systematically when applied to critical state models.

These literature are shown in tabular form in the Table 2.10.

Table 2.10: Literature review on Nonlinear solution techniques and load stepping schemes for critical state models

Author	Description
Gens and Pots [1988]	Compared a number of incremental and iterative schemes and concluded that critical state models are vulnerable to numerical breakdown and may need different solution strategies for different problems. [61]
Abbo and Sloan [1996]	Developed automatic incremental schemes for solving a general class of elasto-plastic problems, which do not involve any iteration. In these methods, the trial load step may be sub-incremented, if necessary, to keep the local load path error below a specified tolerance. [1]
Sheng and Sloan [2001]	Carried out comparative study of various load stepping schemes employed in nonlinear finite element analysis for generalized Cam clay model. [147]

2.6 Interpretation of nonlinear finite element as DAEs

Very less literature is available in this direction of study.

Buttner and Simeon [33] treated the constitutive relations as differential algebraic equations and used implicit Runge-Kutta methods for integration of the same. Although formulations presented for general constitutive relations, the implementation is restricted to J_2 plasticity.

Eckert et al. [51] obtained differential algebraic equations treating displacement as well as internal variables as global variable in FEA and then employed multi-step backward difference method (BDF2) to integrate the differential equations. They used predictor-corrector method to solve the global differential algebraic equation. A plate with hole under plane stress condition was presented to investigate the proposed method.

Ellsiepen and Hartmaan [52] and Hartmann [64] interpreted the discretized form of the principle of virtual displacement together with the ODEs of the constitutive model as a system of DAEs and used implicit Runge-Kutta method for an elasto-plastic model.

2.7 Summary

A brief literature review in the relevant directions is presented. Following points can be drawn from this study:

- There are many finite element software developed in the past using the framework of OOP and most of them are directed towards structural engineering applications. A few number of literature are available for application of this programming paradigm for solving geotechnical engineering problems.

- The most of the soil constitutive models are developed in the framework of elasto-plasticity with inclusion of visco-plasticity to some of them. Again most of the models are developed from the back ground of critical state models, that is Cam clay group of models.

- The Cam clay group of models has appeared to be sufficiently accurate for most of the application in the numerical analysis of many of the practical geomechanics problems requiring realistic soil models. Again these models are simple and need a few input parameters, which can be determined from standard laboratory tests.

- Mita et al. [115] reported that some form of MCCM, is by far, the most widely used model in computational applications.

- Various integration schemes are developed in the recent past to integrate the Cam clay group of models as well as other elasto-plastic models of soil.

- Both explicit and implicit methods have been used to integrate many advanced constitutive models in soil mechanics. While some authors have demonstrated

the effectiveness of the implicit integration algorithms [98], others have reported the disadvantages of the same while using with increasingly complex nonlinear elasto-plastic constitutive models [99].

- Within the last decade, implicit algorithms have drawn growing attentions [73, 78, 81, 142, 149, 153] and now-a-days, return mapping schemes dominate the whole implicit algorithm category [4, 36, 145, 150, 151, 177].
- Very less literature is available which study the influence of the nonlinear solution techniques and load stepping schemes on the solution of nonlinear finite element problems while applied to critical state models.
- Interpretation of nonlinear finite element as DAEs allows the application of all mathematical integration methods to be applied to integrate the advance constitutive models. Only a few work is done in this direction.

The present study is directed towards the object-oriented finite element implementation of MCCM with some integration algorithms and load stepping schemes. Also the concept of the interpretation of nonlinear finite element as DAEs is used to integrate the equations of MCCM. The literature study in these directions reveal that there is scope of development of object-oriented finite element framework for implementation of soil constitutive models which allows easy extension of inclusion of new analysis types, material models, integration algorithms, nonlinear solution techniques and load stepping schemes.

Chapter 3

Soil plasticity and critical state models

3.1 Introduction

The soil model that is selected for the present study is the modified Cam clay model (MCCM) under the group of critical state models. These models have been widely used and discussed in details in the literature [32, 61, 133, 134, 135, 146, 147, 170]. This chapter is aimed at to give a comprehensive review of soil plasticity and development of critical state models for sake of ready reference and continuity of notations.

When applied to geomechanics, the traditional equations of continuum mechanics need some modification. For example, the sign convention for stresses and strains in soil mechanics are opposite to the one used in most engineering materials. In soil mechanics, compressive stresses and strains are normally considered to be positive. To avoid any possible confusion and also for the shake of completeness, basic definitions and equations of stresses and strains are presented in the first section of this chapter. In the subsequent sections, brief introduction of critical state soil mechanics and formulation of critical state models in triaxial stress space is presented.

For notations in the subsequent sections and chapters, bold face is used for vectors and tensors. The tensor product of two tensors is denoted by the notation $\otimes$ such that $(\boldsymbol{a}\otimes\boldsymbol{b})_{ijkl} = (\boldsymbol{a})_{ij}(\boldsymbol{b})_{kl}$ and ':' denotes vector contraction such that $\boldsymbol{a} : \boldsymbol{b} = a_{ij}b_{ij}$,

for two second order tensor $\boldsymbol{a}$ and $\boldsymbol{b}$. Fourth order identity tensor is denoted by $\boldsymbol{I}$, such that $\boldsymbol{I} = \frac{1}{2}(\boldsymbol{\delta}_{ik}\boldsymbol{\delta}_{jl} + \boldsymbol{\delta}_{il}\boldsymbol{\delta}_{jk})$, $\boldsymbol{\delta_{ij}}$ is the Kronecker delta and is denoted by $\boldsymbol{l}$.

3.2 State of stress and strain in soil mechanics

We denote Cauchy stress tensor as $\boldsymbol{\sigma}$. According to Terzaghi's principle, stresses in the soil are divided into effective stresses, $\boldsymbol{\sigma'}$, and pore pressures, σ_w, such that total stress can be written as

$$\boldsymbol{\sigma} = \boldsymbol{\sigma'} + \sigma_w \boldsymbol{l} \tag{3.2.1}$$

In case of critical state models, invariants of stresses are used rather than cartesian stress components when formulating the material models. Therefore, it is useful to define invariants of stress, which are stress measures that are independent of the orientation of the coordinate system. Two useful stress invariants, which are used in critical state models are the volumetric stress p' and deviator stress q. They are given as

$$\begin{aligned} p' &= \frac{1}{3}(\sigma'_{11} + \sigma'_{22} + \sigma'_{33}) = \frac{1}{3}(\sigma'_{ii}) \\ &\Rightarrow p' = \frac{1}{3} tr(\boldsymbol{\sigma'}) \end{aligned} \tag{3.2.2}$$

$$q = \sqrt{\frac{1}{2}\left[(\sigma'_{11} - \sigma'_{22})^2 + (\sigma'_{22} - \sigma'_{33})^2 + (\sigma'_{33} - \sigma'_{11})^2 + 6(\sigma^2_{12} + \sigma^2_{23} + \sigma^2_{31})\right]}$$

$$\Rightarrow q = \sqrt{3J} = \sqrt{\frac{3}{2}\boldsymbol{S} : \boldsymbol{S}} = \sqrt{\frac{3}{2}} \parallel \boldsymbol{S} \parallel \tag{3.2.3}$$

where J is the second invariant of deviatoric stress tensor and $\boldsymbol{S}$ is the deviatoric stress tensor given as $\boldsymbol{S} = \boldsymbol{\sigma'} - p'\boldsymbol{l}$.

Infinitesimal strains are considered in this work which are related with displacement as given below.

$$\boldsymbol{\varepsilon} = (\varepsilon_{11} \quad \varepsilon_{22} \quad \varepsilon_{33} \quad \varepsilon_{12} \quad \varepsilon_{23} \quad \varepsilon_{31})^T$$

where

$$\varepsilon_{11} = \frac{\partial u_1}{\partial x_1} , \qquad \varepsilon_{22} = \frac{\partial u_2}{\partial x_2} , \qquad \varepsilon_{33} = \frac{\partial u_3}{\partial x_3} ,$$

$$\varepsilon_{12} = \tfrac{1}{2}(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}) , \qquad \varepsilon_{23} = \tfrac{1}{2}(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}) , \qquad \varepsilon_{31} = \tfrac{1}{2}(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3})$$

Here u_1, u_2 and u_3 are the displacement components in x_1, x_2 and x_3 directions respectively. For elasto-plastic material models, strains are decomposed into elastic and plastic components

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p$$

In analogy to the invariants of stress, it is also useful to define invariants of strain. A strain invariant that is often used is the volumetric strain, ε_v, which is defined as the sum of all normal strain components

$$\begin{aligned} \varepsilon_v &= \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} = \varepsilon_{ii} \\ \Rightarrow \varepsilon_v &= tr(\boldsymbol{\varepsilon}) \end{aligned} \tag{3.2.4}$$

The total strain ($\boldsymbol{\varepsilon}$) can be thus divided into two components as volumetric strain (ε_v) and deviatoric strain ($\boldsymbol{\varepsilon}_d$)

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}_d + \frac{1}{3}\varepsilon_v \boldsymbol{l}$$

Each of the volumetric and deviatoric strains can be divided into plastic and elastic parts. They are expressed as

$$\varepsilon_v^e = tr(\boldsymbol{\varepsilon}^e) , \qquad \varepsilon_v^p = tr(\boldsymbol{\varepsilon}^p) \tag{3.2.5}$$

$$\boldsymbol{\varepsilon}_d^e = \boldsymbol{\varepsilon}^e - \frac{1}{3}\varepsilon_v^e \boldsymbol{l} , \quad \boldsymbol{\varepsilon}_d^p = \boldsymbol{\varepsilon}^p - \frac{1}{3}\varepsilon_v^p \boldsymbol{l} \tag{3.2.6}$$

Throughout this work, the superscript e will be used to denote elastic strains and the superscript p will be used to denote plastic strains. Similarly the subscript v will used to denote volumetric strains and subscript d will be used to denote deviatoric strains. The relationship between infinitesimal increments of effective stress ('effective stress rates') and infinitesimal increments of elastic strain ('strain rates') can be expressed in the form

$$\dot{\boldsymbol{\sigma}}' = \boldsymbol{C}^e : \dot{\boldsymbol{\varepsilon}}^e \tag{3.2.7}$$

where $\boldsymbol{C}^e$ is elastic stiffness matrix of the material. For isotropic linear elastic behaviour $\boldsymbol{C}^e$ contains only two parameters, the effective Young's modulus, E', and the effective Poisson's ratio, μ'.

The relationship between Young's modulus E' and other stiffness moduli, such as the shear modulus G', and the bulk modulus K' is given by

$$G' = \frac{E'}{2(1+\mu')} \ , \qquad K' = \frac{E'}{3(1-2\mu')} \ , \quad \text{and so} \quad G' = 3K'\frac{(1-2\mu')}{2(1+\mu')}$$

The pore pressures, σ_w in Equation (3.2.1) are zero for a drained analysis. To determine the incremental change of pore pressure $\Delta\sigma_w$ corresponding to an incremental strain $\Delta\boldsymbol{\varepsilon}$ during an undrained analysis, it is assumed that volumetric strain experienced by the soil is due entirely to a change in the volume of pore water. The volumetric strain experienced by the pore water can be given as $[(1+e)/e]\Delta\varepsilon_v$, where e is the current void ratio. Then the change in pore pressure can be given as

$$\Delta\sigma_w = K_w[(1+e)/e]\Delta\varepsilon_v \tag{3.2.8}$$

where K_w is the bulk modulus of pore water and is defined as mK'. In an undrained analysis, K_w is normally set to a value between 50 to 500 times of K', which is equivalent to the use of a value of Poisson's ratio in the range 0.49 to 0.499 [32].

3.3 Critical state soil mechanics

It is observed that the features of soil behaviour are consistent with the elasto-plastic framework. Hence the use of plasticity in the prediction of the behaviour of soils allows a rational treatment of many of the soil mechanics problems. The theories of soil behaviour known as critical state soil mechanics were developed from the application of the theory of plasticity to soil mechanics. Following points can be drawn for critical state soil mechanics:

∗ At its critical state, soil continues to distort at constant effective stress and at constant volume. This applies for turbulent flow of the particles. If the flow becomes laminar, as in clays at large strain, the strength falls to the residual.

∗ When soil is at its critical state, there is a unique relationship between shear stress,

effective normal stress and water content (or void ratio).

∗ Critical states are unique and do not depend on initial state or stress path.

∗ Critical shear stress (critical state strength) increases with increasing effective normal stress and with decreasing water content.

In order to use the theory of plasticity to predict the behaviour of any engineering structures, an appropriate idealization of plasticity is required (such as linear elastic and perfectly plastic, linear elastic and linear strain hardening, etc.). As soil exhibits a rather more complex behaviour, more appropriate idealization is required while using the plastic theory of deformation to soil. Early attempts of taking into account the frictional characteristics of soil by generalizing the Mohr-Coulomb failure envelope failed to model adequately many basic features of soil behaviour. This led towards the formulation of the first critical state models, i.e. Cam clay group of models. In the following sections, the development of Cam clay group of models in triaxial stress space is discussed.

3.4 Basic formulation of critical state models in triaxial stress space

The Cam clay models were originally developed for triaxial loading conditions. They are essentially based on the following assumptions, which are explained in brief in following subsections. For more detailed explanation, [32] and [170] can be referred.

3.4.1 Critical state parameters

The three parameters, which describe the state of a sample of soil during a triaxial test, are volumetric stress p', deviator stress q and specific volume v. These three parameters are called as critical state parameters. They are also basic parameters for the critical state models. For triaxial loading condition, the parameters p' and q are obtained from Equation (3.2.2) and Euation (3.2.3). They are defined as

$$p' = (\sigma'_a + 2\sigma'_r)/3 = (\sigma_a + 2\sigma_r)/3 - \sigma_w \tag{3.4.9}$$

$$q = (\sigma'_a - \sigma'_r) = (\sigma_a - \sigma_r) \tag{3.4.10}$$

$$v = 1 + e \tag{3.4.11}$$

where $\sigma'_a = \sigma'_{11}$ is the effective pressure applied in the axial direction, $\sigma'_r = \sigma'_{22} = \sigma'_{33}$ is the effective pressure applied in the radial direction of the soil sample, σ_w is the pore water pressure, and e is the void ratio. Here p' is often called as mean normal effective pressure. The progress of a soil sample during a triaxial test can be represented by a series of points describing a line in three dimensional (p', v, q) space.

3.4.2 Volume-pressure relations

If a sample of soil is subjected to slow, perfectly drained, isotropic compression and swelling tests, then the paths followed by the sample in the $(ln(p'), v)$ space is known as isotropic normal consolidation line and swelling line (Figure 3.1), which is basically similar to the ($log(\sigma'_v)$, e) plot obtained from oedometer tests.

In critical state theory, the isotropic normal consolidation line (or virgin compression line), swelling and re-compression lines are assumed to be straight in $(ln(p'), v)$ plots, the slope of which are known as λ and κ respectively. The virgin compression line (λ-line) and the swelling lines (κ-line) are given by the following equations

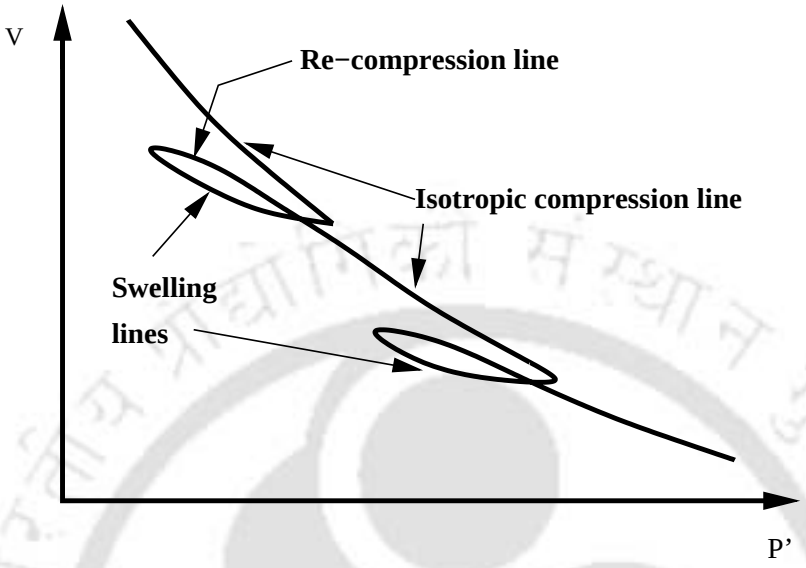
$$v = N - \lambda\ ln(p') \tag{3.4.12}$$

$$v = v_\kappa - \kappa\ ln(p') \tag{3.4.13}$$

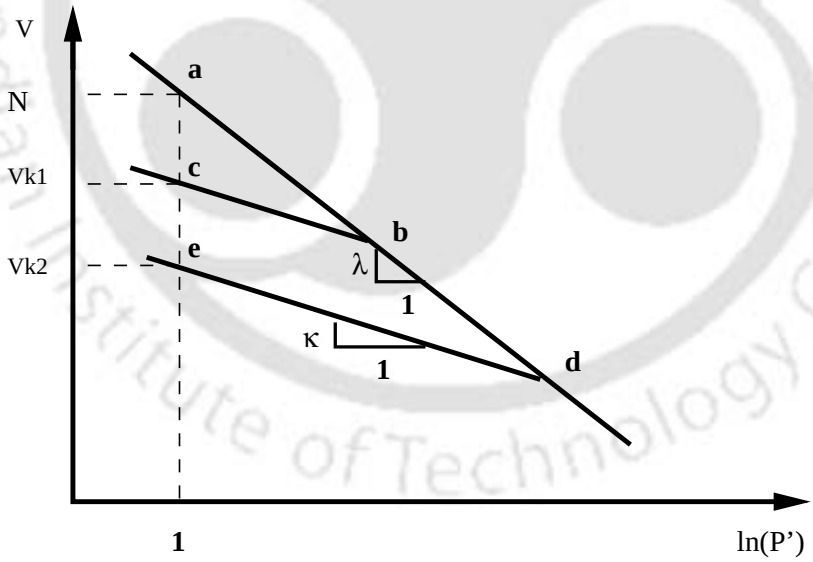
where N is the value of the effective volume v corresponding to $ln(p') = 0$, which is a constant for a particular soil and can be related to other parameters as ($N = \Gamma + \lambda - \kappa$). The parameter Γ defines the location of the critical state line in the (p', v) space. Volume change along the virgin compression line is mainly irreversible or plastic, while volume change along a swelling line is reversible or elastic.

3.4.3 Critical state line

When the soil samples are sheared, they approach the critical state line (CSL), which is a straight line in the (p', q) space and a curved line in the three dimensional



(a)Typical Experimental behaviour under Isotropic Compression



(b) Idealized plot in critical state theory

Figure 3.1: Isotropic loading in triaxial space

$(p',\ v,\ q)$ space. The parameter M is the slope of the critical state line in the (p', q) space. The equations of the critical state line are

$$q = Mp' \tag{3.4.14}$$

and

$$v = \Gamma - \lambda\ ln(p') \tag{3.4.15}$$

The critical state line represents the final state of soil samples in triaxial tests when it is possible to continue to shear the soil sample with no change in the imposed stress or volume of the soil.

3.4.4 Yielding of Cam clay

Before reaching the CSL, the material undergoes successive states of yielding. Yielding of a soil sample (in triaxial stress space) starts at a point when

$$q = [N - v - \lambda\ \ln(p')]\frac{Mp'}{(\lambda - \kappa)} \tag{3.4.16}$$

Equation(3.4.16) describes a surface in the (p', v, q) space. It is called as stable state boundary surface(SSBS) and the CSL lies on this surface. Behavior of soil at any point below this surface is elastic. Soil state on the surface indicates yielding and it is impossible to obtain a stress state above this surface. Figure 3.2 shows the SSBS, virgin compression line, swelling line and CSL.

The projection of the SSBS forms a series of moving caps in the (p', q) space. Elastic straining underneath the SSBS corresponds to movement along a κ-line, with a corresponding change in volume, v. Thus, when an elastic sample is brought to the point of yield, it must simultaneously lie both on the κ-line and on the SSBS. Therefore, the intersection of the SSBS with the κ-line equation gives the current yield surface as

$$q = Mp'\ ln(p'_c/p) \tag{3.4.17}$$

or

$$q/(Mp') + ln(p'/p'_c) = 0 \tag{3.4.18}$$

If the sample is continued to loading after reaching the first yielding, it will move on the SSBS with successive state of yielding during which it hardens or softens. Finally

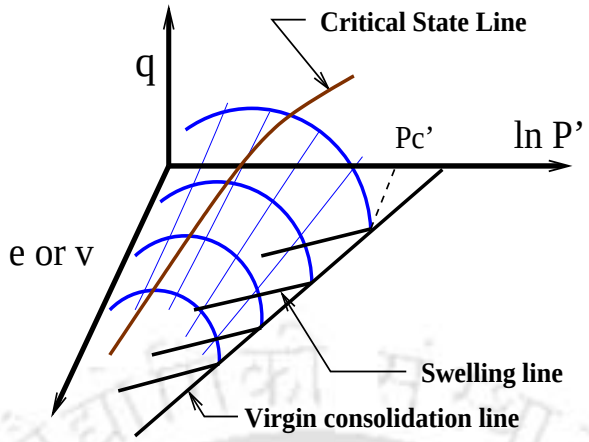


Figure 3.2: Stable state boundary surface

the soil sample will fail on reaching the critical state line. The form of the yield function defined by the above Equation(3.4.17) or (3.4.18) is as shown in Figure 3.3 and it consists of the critical state line and the projection of the SSBS in the (p', q) space. Here p'_c is the isotropic normal consolidation pressure for a soil sample lying on a particular swelling line which determines the intercept of the yield locus in the direction of axis p'.

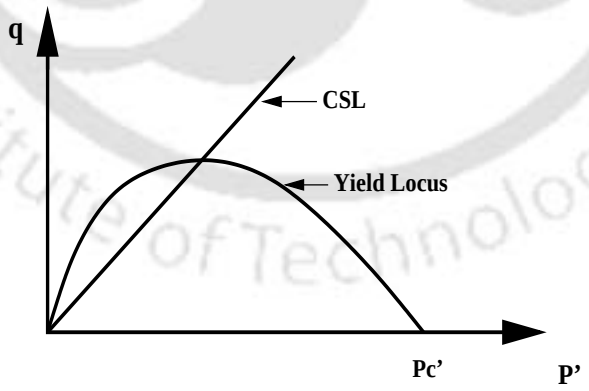


Figure 3.3: Yield locus of Cam clay model

3.4.5 Hardening rule

Isotropic hardening/softening is controlled by the parameter p'_c which is related to the volumetric plastic strain, ϵ_v^p, by

$$\frac{dp'_c}{p'_c} = d\varepsilon_v^p \frac{v}{\lambda - \kappa} \tag{3.4.19}$$

which provides the hardening rule.

As the behaviour along a swelling line is elastic, the elastic volumetric strain, ϵ_v^e, can be determined from the equation of the swelling line

$$d\epsilon_v^e = \frac{dv}{v} = \frac{\kappa}{v}\frac{dp'}{p'} \tag{3.4.20}$$

This gives the elastic bulk modulus K', as

$$K' = \frac{dp'}{d\epsilon_v^e} = \frac{vp'}{\kappa} \tag{3.4.21}$$

It is seen from Equation(3.4.21) that the elastic modulus K' depends linearly on the hydrostatic pressure p' in Cam clay model. Therefore in Cam clay group of models the elastic moduli K' and G' (or μ') are state variables exhibiting nonlinear elastic response. This is one of the important features of soil behaviour and it makes the Cam clay group of models different from conventional plasticity models.

3.4.6 Volumetric response of Cam clay

Cam clay models are capable of predicting different volumetric response of soil depending on its stress history. The projection of the CSL on the $q = 0$ plane divides the possible stress states into two regions, referred as the wet side and the dry side of the CSL. Soil yielding on the wet side will experience volumetric hardening and on the dry side will cause softening. The state boundary surface is called as Roscoe and Hvorslev surface on the wet and dry side respectively and the two surfaces merge at the CSL.

Generally various shapes of yield surface such as spherical bullet and elliptical have been assumed and the slope of the yield surface will depend on the type of material. But the point of intersection of the yield cap with the p'-axis determines the

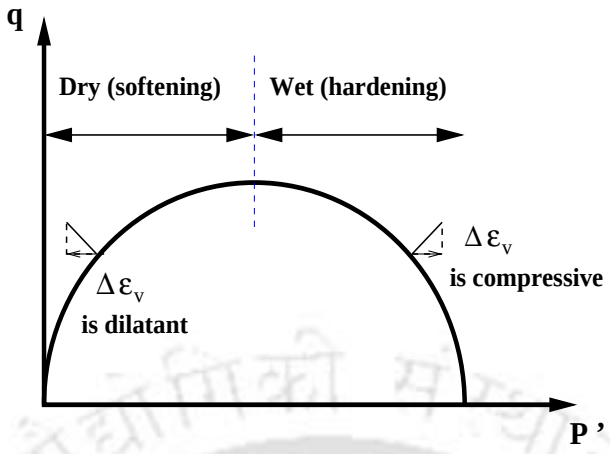


Figure 3.4: Volumetric response in triaxial stress space

hardening behaviour and the tangent drawn at the point of intersection of the yield cap with the CSL is horizontal. This means that on the CSL, with associated flow rule, the plastic flow occurs with constant volume. Subsequently modified Cam clay model was introduced by Roscoe and his co-workers in 1968 which has an elliptical yield surface [32].

3.4.7 Strain for triaxial stress space

Following are the assumptions made regarding the strain for triaxial stress space in the development of Cam clay models.

Elastic strain: Elastic volumetric strain ε_v^e is calculated from κ-line equation and the elastic deviatoric strain ε_d^e is assumed to be zero.

Plastic strain: Plastic volumetric strain is given as $\varepsilon_v^p = \delta v_k / v$ and plastic deviatoric strain ε_d^p is calculated from the flow rule as $\varepsilon_v^p / \varepsilon_d^p = M - q/p'$

Cam clay models are thus far the most widely used plasticity models for characterizing the stress-strain behaviour of cohesive soils. The advantages of these models lie in their apparent simplicity and their capability to represent the stress-strain behaviour of soil realistically. Cam clay models include features such as pressure sensitivity, hardening and softening responses typical in soils, and require few parameters which can be directly obtained from conventional laboratory tests.

3.5 Modified Cam clay model

It is found that the Cam clay model is deficient in some aspects of stress-strain modeling, which lead to the development of modified Cam clay model. Firstly, there is a point or stress discontinuity at $p' = p'_c$ in the yield locus of Cam clay model and so normality does not hold at this point. Secondly, due to the shape of the yield locus, the shear strains predicted by Cam clay are too high at low stress ratios (q/p') as compared to the one obtained from experiments. Thirdly, the value of k_o (the coefficient of earth pressure at rest) predicted by Cam clay model is 1.0 for normally consolidated soil, where measured values are in the range of 0.5 to 0.7. To overcome these drawbacks, the original Cam clay model was modified by modifying the flow rule. The yield locus for modified Cam clay model is obtained by integrating the flow rule

$$F(p', q, p'_c) = \left(\frac{q}{M}\right)^2 + p'(p' - p'_c) \leq 0 \tag{3.5.22}$$

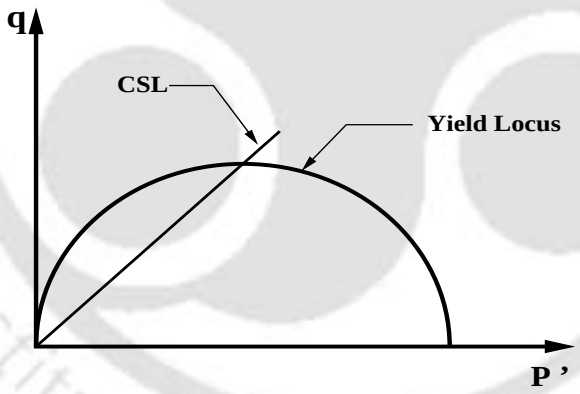


Figure 3.5: Yield locus of modified Cam clay model

Figure 3.5 shows the yield locus for modified Cam clay model. The isotropic hardening/softening of Cam clay group of models is controlled by the parameter p'_c which is related to the plastic volumetric strain by Equation (3.4.19).

The modified Cam clay yield locus is elliptical in shape and this is the main difference between modified Cam clay model and Cam clay model. Because of this difference in shape of the yield locus, the vertical distance between the isotropic normal consolidation line and the CSL becomes $(\lambda - \kappa)ln(2)$ rather than $(\lambda - \kappa)$.

3.6 Constitutive relations used in the present study

In present work, modified Cam clay model has been employed to develop object-oriented finite element software. Modified Cam clay model is developed from the application of concept of plasticity to soil mechanics [170]. The behavior of an elasto-plastic material model can be characterized by (1) existence of a yield function, (2) stress-strain relation, (3) flow rule for the plastic strain increment, and (4) hardening rule for the evolution of the yield surface with plastic straining [32]. Table 3.1 compares these fundamental statements for modified Cam clay model with general plasticity models.

Table 3.1: Comparison of the fundamentals of modified Cam clay model with general plasticity model

Sl No.	General Plasticity models	Modified Cam clay model
1	Yield function: $F(\boldsymbol{\sigma}, k) = 0$	$F(p', q, p'_c) = \left(\frac{q}{Mp'}\right)^2 - \left(\frac{p'_c}{p'} - 1\right) = 0$
2	Stress-strain relation: $\dot{\boldsymbol{\sigma}} = \boldsymbol{C}^e : \dot{\boldsymbol{\varepsilon}}^e$	$\dot{\boldsymbol{\sigma}}' = \boldsymbol{C}'^e : \dot{\boldsymbol{\varepsilon}}^e$
3	Plastic strain $\dot{\boldsymbol{\varepsilon}}^p = \dot{\phi}\frac{\partial g(\boldsymbol{\sigma},\ k)}{\partial \boldsymbol{\sigma}}$	$\dot{\boldsymbol{\varepsilon}}^p = \dot{\phi}\frac{\partial F(\boldsymbol{\sigma}',\ k)}{\partial \boldsymbol{\sigma}'}$
4	Hardening parameters $\dot{k} = f_1(\dot{\varepsilon}^p_{eff})$ $\dot{\varepsilon}^p_{eff} = \sqrt{\frac{2}{3}\dot{\boldsymbol{\varepsilon}}^p : \dot{\boldsymbol{\varepsilon}}^p}$ $\dot{\varepsilon}^p_{eff} = \dot{\phi}\sqrt{\frac{2}{3}\frac{\partial g}{\partial \boldsymbol{\sigma}} : \frac{\partial g}{\partial \boldsymbol{\sigma}}}$	$\dot{k} = \dot{p}'_c = \vartheta\ p'_c\dot{\varepsilon}^p_v$ $\dot{\varepsilon}^p_v = \dot{\phi}\frac{\partial F}{\partial p'}$ $\vartheta = \frac{v}{(\lambda - \kappa)}$ $\dot{p}'_c = \vartheta\ p'_c\dot{\phi}\frac{\partial F}{\partial p'}$

The yield function, constitutive relations, flow rule (evolution of plastic strain), and hardening parameters for modified Cam clay model are as depicted in Table 3.1. Here $\dot{\boldsymbol{\varepsilon}}$, $\dot{\boldsymbol{\varepsilon}}^e$ and $\dot{\boldsymbol{\varepsilon}}^p$ are the total, elastic and plastic strain rate tensors respectively. $\boldsymbol{C}^e$ is the fourth order elasticity tensor, $\dot{\varepsilon}_v$ is the total volumetric strain rate, and $\dot{\phi}$ is the plastic multiplier. The plastic multiplier $\dot{\phi}$ is related to the magnitude of

plastic strain by using Kuhn-Tucker form as follows:

$$F(p', q, p'_c) \leq 0 \tag{3.6.23}$$

$$\dot{\phi} \geq 0 \tag{3.6.24}$$

$$F(p', q, p'_c)\dot{\phi} = 0 \tag{3.6.25}$$

The above Equations (3.6.23)−(3.6.25) are statements of the loading-unloading criterion for elasto-plasticity, which must be satisfied simultaneously. The first criterion represents that the current stress state must always lie on or below the current yield surface and the behaviour of the material must be elastic when the current stress state lies below the current yield surface. If the material deforms plastically, then the stress state must lie on the yield surface. The last statement is the consistency requirement, which states that the final stress state must be consistent with and so lie on the current yield surface.

As Cam clay group of models are essentially derived using effective stress concept, the stress states considered in the subsequent chapters are also effective stresses. The notation '′' is associated with the effective stress values and in the remaining part of this report effective parameters are denoted without dash (′), unless a different meaning is explicitly stated. One important aspect of geotechnical engineering that the compressive stresses are treated as positive, while the tensile stresses are treated as negative. Again, all positive stresses referred here in the subsequent sections and chapters are compressive stresses, unless otherwise mentioned.

3.7 Summary

This chapter briefly explains the fundamentals of modified Cam clay model. Modified Cam clay model is developed by applying the concept of plasticity to soil mechanics. This model is widely used for numerical analysis of many practical geomechanics problems. In the present work, modified Cam clay model is considered for study of integration of constitutive relation and its effect on incremental-iterative process.

Chapter 4

An implicit integration algorithm for MCCM

4.1 Introduction

The purpose of the numerical integration of the elasto-plastic constitutive equations is to return the stress history corresponding to a given deformation history, at each stress point. As it is not possible to get the exact analytical solutions for the elasto-plastic evolution problems except for the simplest ones, all elasto-plastic models are to be implemented in the analysis using a numerical integration algorithm. Therefore, the accuracy and stability of the solutions and also the cost of a numerical analysis is strongly affected by the accuracy, stability and efficiency of the integration algorithm.

It is observed from the literature survey that various integration algorithms have been developed in past that apply directly to the critical state models and both implicit and explicit methods have been used to integrate advanced constitutive relations in soil mechanics [156]. These algorithms are implemented using either substepping scheme or return mapping scheme such as radial return mapping, central return mapping, closest point projection. In an explicit integration scheme, the yield surface gradient, plastic potential gradients and hardening rule are all evaluated at the known stress state and so it does not strictly require any iteration. But this may lead to drift of the final stress state from the yield surface and in some cases may

have considerable affect on the results. In a fully implicit method, the gradients and hardening rule are evaluated at unknown stress states. This results in a system of nonlinear equations, which must be solved iteratively. Wang et al. [168] reported that within the last decade, implicit algorithms have drawn growing attentions and return mapping schemes have dominated the whole implicit algorithm category.

In this research, a new implicit algorithm is presented for the integration of the rate constitutive equation for MCCM. The algorithm is based on the idea of secant elastic moduli to integrate the nonlinear elastic constitutive equation. The elasto-plastic constitutive relations are integrated using return mapping algorithms in which the return directions are computed by closest point projection, considering associative flow rule. In the present algorithm the elastic secant moduli are treated implicitly in the elasto-plastic region which improves the convergence at global level. The generality and applicability of the algorithm are tested through numerical examples, which demonstrate its local as well as global accuracy and stability.

4.2 Integration algorithm for MCCM

MCCM exhibits nonlinearity in the elastic regime. Hence the constitutive equations have to be integrated in the elastic region also. In most of the implicit stress integration algorithms for MCCM, the constitutive relations are considered to be linear over a finite time interval. The integration algorithm proposed by Borja and Lee [24] also considered the stress-strain behaviour to be linear in the elastic region.

Again the elastic moduli K and G, which linearly depend on the volumetric stress level, are also treated explicitly over a finite time interval. In the present algorithm, the constitutive equations are treated nonlinearly and are incorporated into the algorithm proposed by Borja and Lee, along with implicit treatment of the elastic moduli in the elasto-plastic region. The new implicit integration algorithms proposed for MCCM using closest point return mapping scheme are outlined in the subsequent sections.

4.2.1 Preliminaries

The rate constitutive equations, which need to be integrated can be given as

$$\dot{\boldsymbol{\sigma}} = \boldsymbol{C}^e : \dot{\boldsymbol{\varepsilon}}^e = K\hat{\boldsymbol{C}}^e : \dot{\boldsymbol{\varepsilon}}^e \tag{4.2.1}$$

where

$$\boldsymbol{C}^e = K\boldsymbol{l}\otimes\boldsymbol{l} + 2G(\boldsymbol{I} - \frac{1}{3}\boldsymbol{l}\otimes\boldsymbol{l})$$

$$\hat{\boldsymbol{C}}^e = \boldsymbol{l}\otimes\boldsymbol{l} + 2r(\boldsymbol{I} - \frac{1}{3}\boldsymbol{l}\otimes\boldsymbol{l})$$

$$r = \frac{3(1-2\mu)}{2(1+\mu)}$$

The yield function, flow rule, and hardening rule of the MCCM are as given in Table 3.1.

$$F = \frac{q^2}{M^2} + p(p - p_c) = 0, \tag{4.2.2}$$

$$\dot{\boldsymbol{\varepsilon}}^p = \phi\frac{\partial F(\boldsymbol{\sigma},\ k)}{\partial\boldsymbol{\sigma}} \tag{4.2.3}$$

$$\dot{p}_c = \vartheta p_c\dot{\varepsilon}_v^p \tag{4.2.4}$$

where $\vartheta = v/(\lambda-\kappa)$. The specific volume of soil v, and so the parameter ϑ involved in the hardening rule are state variables. However, variation of v is usually small for large load increment. Hence while integrating the evolution equations, both v and ϑ can be treated explicitly over the time interval $[t_n, t_{n+1}]$. The hardening rule is in rate form and needs to be integrated either analytically or by using numerical integration technique.

Using associated flow rule, plastic strain rate can be given as

$$\dot{\boldsymbol{\varepsilon}}^p = \phi\frac{\partial F}{\partial\boldsymbol{\sigma}}\ , \quad Hence \quad \dot{\varepsilon}_v^p = tr(\dot{\boldsymbol{\varepsilon}}^p) = \phi\frac{\partial F}{\partial p}\ , \quad and \quad \dot{\boldsymbol{\varepsilon}}_d^p = \phi\frac{\partial F}{\partial\boldsymbol{S}} \tag{4.2.5}$$

The derivative $\frac{\partial F}{\partial\boldsymbol{\sigma}}$ in Equation(4.2.5) can be obtained by applying chain rule of differentiation to the yield function as follows:

$$\frac{\partial F}{\partial\boldsymbol{\sigma}} = \frac{\partial F}{\partial p}\frac{\partial p}{\partial\boldsymbol{\sigma}} + \frac{\partial F}{\partial q}\frac{\partial q}{\partial\boldsymbol{\sigma}} = \frac{1}{3}\frac{\partial F}{\partial p}\boldsymbol{l} + \sqrt{\frac{3}{2}}\frac{\partial F}{\partial q}\boldsymbol{n}, \tag{4.2.6}$$

where $\boldsymbol{n} = \frac{\boldsymbol{S}}{\|\boldsymbol{S}\|}$, unit vector along the direction of $\boldsymbol{S}$. The derivatives in Equation(4.2.6) can be directly obtained from the yield function F as

$$\frac{\partial F}{\partial p} = 2p - p_c \ , \qquad \frac{\partial F}{\partial q} = \frac{2q}{M^2} \ , \qquad \frac{\partial F}{\partial p_c} = -p \tag{4.2.7}$$

4.2.2 Stress integration algorithm in the elastic region

In the nonlinear elastic region, Equation(4.2.1) is integrated over a finite time interval $[t_n,\ t_{n+1}]$ as

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}_n + \int_{t_{n+1}}^{t_n} K\hat{\boldsymbol{C}}^e : d\boldsymbol{\varepsilon}^e \tag{4.2.8}$$

Considering the volumetric part of Equation (4.2.1) we have $\dot{p} = K\dot{\boldsymbol{\varepsilon}}_v^e$, which gives

$$K = \frac{\dot{p}}{\dot{\boldsymbol{\varepsilon}}_v^e} \tag{4.2.9}$$

Let us consider an average bulk modulus over the time interval (t_n, t_{n+1}) as $\bar{K}$, such that

$$\bar{K} = \frac{\Delta p}{\Delta \varepsilon_v^e} = \frac{p_{n+1} - p_n}{\varepsilon_{v\ n+1}^e - \varepsilon_{v\ n}^e} \tag{4.2.10}$$

Using this average bulk modulus in Equation (4.2.8), we get

$$\boldsymbol{\sigma}_{n+1} \simeq \boldsymbol{\sigma}_n + \bar{K}\hat{\boldsymbol{C}}^e : \Delta\boldsymbol{\varepsilon}^e \tag{4.2.11}$$

Now substituting the instantaneous value of $K = \dfrac{1+e}{\kappa}p$ in Equation (4.2.9) we get

$$\dot{p} = \frac{1+e}{\kappa} p\ \dot{\boldsymbol{\varepsilon}}_v^e \tag{4.2.12}$$

Integrating Equation (4.2.12) over the time interval (t_n, t_{n+1}), we get

$$\begin{aligned} \left[\int \frac{dp}{p} = \frac{1+e}{\kappa} \int d\varepsilon_v^e\right]_{t_{n+1}}^{t_n} \\ \Rightarrow ln(\frac{p_{n+1}}{p_n}) = \frac{1+e}{\kappa}(\varepsilon_{v\ n+1}^e - \varepsilon_{v\ n}^e) \\ \Rightarrow p_{n+1} = p_n\ exp\left[\frac{1+e}{\kappa}\Delta\varepsilon_v^e\right] \end{aligned} \tag{4.2.13}$$

Using this value in Equation (4.2.10), we get the expression for $\bar{K}$ as

$$\bar{K} = \frac{p_n}{\Delta\varepsilon_v^e}\left[exp(\frac{1+e}{\kappa}\Delta\varepsilon_v^e) - 1\right] \tag{4.2.14}$$

The tangent stress-strain tensor $\boldsymbol{C}_{n+1}^e$ for an elastic process at time step $(n+1)$ after k^{th} iteration consistent with the Equation (4.2.11) is obtained by deriving it with respect to the strain tensor $\boldsymbol{\varepsilon}_{n+1}^e$

$$\boldsymbol{C}_{n+1}^e = \frac{\partial\sigma_{n+1}}{\partial\varepsilon_{n+1}^e} = \frac{\partial}{\partial\varepsilon_{n+1}^e}\left(\bar{K}\hat{\boldsymbol{C}}^e : \Delta\varepsilon^e\right) = \bar{K}\hat{\boldsymbol{C}}^e + \hat{\boldsymbol{C}}^e : \Delta\varepsilon^e \otimes \frac{\partial\bar{K}}{\partial\varepsilon_{n+1}^e}$$

The variation of $\bar{K}$ with respect to ε_{n+1}^e is obtained as

$$\frac{\partial\bar{K}}{\partial\varepsilon_{n+1}^e} = \frac{\partial\bar{K}}{\partial\varepsilon_{v\ n+1}^e}\frac{\partial\varepsilon_{v\ n+1}^e}{\partial\varepsilon_{n+1}^e} = \frac{\frac{1+e}{\kappa}p_{n+1} - \bar{K}}{\Delta\varepsilon_{v\ n+1}^e}\boldsymbol{l}$$

So we have

$$\left(\boldsymbol{C}_{n+1}^e\right)^k = \bar{K}\hat{\boldsymbol{C}}^e + \frac{K_{n+1}^k - \bar{K}}{\Delta\varepsilon_v^e}\left(\hat{\boldsymbol{C}}^e : \Delta\boldsymbol{\varepsilon}^e\right) \otimes \boldsymbol{l}, \tag{4.2.15}$$

where $K_{n+1}^k = \frac{1+e}{\kappa}p_{n+1}^k$, and k is the iteration number.

4.2.3 Stress integration algorithm in the elasto-plastic region

For integrating the elasto-plastic rate constitutive equation, the algorithm proposed by Borja and Lee [24] with closest point projection method is modified for implicit treatment of elastic moduli K and G. The rate constitutive Equation (4.2.1) can be integrated over the finite time interval $[t_n, t_{n+1}]$, to yield the stress tensor as follows:

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}_n + K_{n+\alpha}\ \Delta\varepsilon_{v\ n+1}^e\ \boldsymbol{l} + 2G_{n+\alpha}\ \Delta\boldsymbol{\varepsilon}_{d\ n+1}^e \tag{4.2.16}$$

Decomposing the stress tensor into deviatoric and volumetric parts at stages t_n and t_{n+1}

$$\boldsymbol{\sigma}_n = p_n\boldsymbol{l} + \boldsymbol{S}_n\ , \qquad \boldsymbol{\sigma}_{n+1} = p_{n+1}\boldsymbol{l} + \boldsymbol{S}_{n+1} \tag{4.2.17}$$

From Equation(4.2.16) and (4.2.17) we can write

$$p_{n+1} = p_n + K_{n+\alpha}\left(\Delta\varepsilon_v - \Delta\varepsilon_v^p\right)_{n+1} \tag{4.2.18}$$

$$\boldsymbol{S}_{n+1} = \boldsymbol{S}_n + 2G_{n+\alpha}\left(\Delta\boldsymbol{\varepsilon}_d - \Delta\boldsymbol{\varepsilon}_d^p\right)_{n+1} \tag{4.2.19}$$

$$q_{n+1} = \sqrt{\frac{3}{2}}\|\boldsymbol{S}_{n+1}\| \tag{4.2.20}$$

The hardening rule can be integrated over a finite time increment analytically with respect to the variable p_c and is obtained as

$$(p_c)_{n+1} = (p_c)_n exp\left[\vartheta\left(\Delta\varepsilon_v - \Delta\varepsilon_v^p\right)_{n+1}\right] \tag{4.2.21}$$

Let us assume a trial stress state, $\boldsymbol{\sigma}_{n+1}^{tr} = \boldsymbol{\sigma}_n + \boldsymbol{C}_n^e : \Delta\boldsymbol{\varepsilon}_{n+1}$, where $\boldsymbol{C}_n^e$ can be calculated using Equation (4.2.15) with $\Delta\boldsymbol{\varepsilon}^p = 0$. Using this in above Equations (4.2.18), (4.2.19), (4.2.21) and applying normality rule over a finite strain increment $\Delta\boldsymbol{\varepsilon}^p$, we get from generalized trapezoidal rule as

$$p_{n+1}(\phi) = p_{n+1}^{tr} - K_{n+\alpha}\ \phi(2p - p_c)_{n+1} \tag{4.2.22}$$

$$\boldsymbol{S}_{n+1}(\phi) = \boldsymbol{S}_{n+1}^{tr}\left(1 + 6\frac{G_{n+\alpha}\ \phi}{M^2}\right)^{-1} \tag{4.2.23}$$

$$p_{c\ n+1}(\phi) = p_{c\ n}\ exp\left[\vartheta\phi\left(2\ p - p_c\right)_{n+1}\right] \tag{4.2.24}$$

where $\vartheta_{n+1} = \vartheta_n = \vartheta$.

From Equation (3.2.3)

$$q_{n+1}(\phi) = \sqrt{\frac{3}{2}}\ \|\ \boldsymbol{S}_{n+1}(\phi)\ \|$$

It can be proofed that the directions of $\Delta\boldsymbol{\varepsilon}_{d\ n+1}^p$, $\boldsymbol{S}_{n+1}^{tr}$ and $\boldsymbol{S}_{n+1}$ are same, i.e.

$$\boldsymbol{n} = \frac{\boldsymbol{S}_{n+1}}{\|\ \boldsymbol{S}_{n+1}\ \|} = \frac{\boldsymbol{S}_{n+1}^{tr}}{\|\ \boldsymbol{S}_{n+1}^{tr}\ \|} = \frac{\Delta\boldsymbol{\varepsilon}_{d\ n+1}^p}{\|\ \Delta\boldsymbol{\varepsilon}_{d\ n+1}^p\ \|}$$

Hence we can write

$$\|\ \boldsymbol{S}_{n+1}^{tr} + \Delta\boldsymbol{\varepsilon}_{d\ n+1}^p\ \| = \|\ \boldsymbol{S}_{n+1}^{tr}\ \| + \|\ \Delta\boldsymbol{\varepsilon}_{d\ n+1}^p\ \|$$

Therefore

$$q_{n+1} = \sqrt{\frac{3}{2}}\ \|\ \boldsymbol{S}_{n+1}^{tr}\ \| - \sqrt{\frac{3}{2}}2G_{n+\alpha}\ \|\ \Delta\boldsymbol{\varepsilon}_{d\ n+1}^p\ \|$$

$$\Rightarrow q_{n+1} = q_{n+1}^{tr} - \sqrt{6}G_{n+\alpha}\ \|\ \Delta\boldsymbol{\varepsilon}_{d\ n+1}^p\ \|$$

Using $\| \Delta\boldsymbol{\varepsilon}^p_{d\ n+1} \| = \| \phi \frac{\partial F}{\partial \boldsymbol{S}} \| = \phi \sqrt{\frac{3}{2}} \frac{\partial F}{\partial q}$ in the above equation, we get after simplification

$$q_{n+1} = q^{tr}_{n+1} \left(1 + 6 \frac{G_{n+\alpha}\ \phi}{M^2} \right)^{-1} \tag{4.2.25}$$

The value of α is 0 for first order forward Euler scheme, 1 for backward Euler scheme, and 0.5 for mid-point rule. Considering the first order Euler scheme for the elastic moduli K and G (i.e. $\alpha = 0$) in Equations (4.2.22), (4.2.24) and (4.2.25) we have

$${}^1p_{n+1}(\phi) = p^{tr}_{n+1} - K_n \phi \left(2\ {}^1p - {}^1p_c \right)_{n+1} \tag{4.2.26}$$

$${}^1q_{n+1}(\phi) = q^{tr}_{n+1} \left(1 + 6 \frac{G_n \phi}{M^2} \right)^{-1} \tag{4.2.27}$$

$${}^1p_{c\ n+1}(\phi) = p_{c\ n}\ exp \left[\vartheta \phi \left(2\ {}^1p - {}^1p_c \right)_{n+1} \right] \tag{4.2.28}$$

The notation ^{1*} is used for the stage values obtained from first order Euler scheme. By applying the consistency requirement, $F = 0$, we get

$${}^1F_{n+1}(\phi) = \left[\frac{{}^1q^2}{M^2} + {}^1p({}^1p - {}^1p_c) \right]_{n+1} = 0 \tag{4.2.29}$$

The nonlinear Equations (4.2.26), (4.2.27), (4.2.28) and (4.2.29) are to be solved iteratively to get the four unknowns, ${}^1p_{n+1}$, ${}^1q_{n+1}$, ${}^1p_{c\ n+1}$ and ϕ. But the variables p and p_c are coupled via Equations (4.2.26) and (4.2.28), and so one cannot evaluate the function $F(\phi)$ explicitly. Consequently, a multi-stage Newton-Raphson method has to be used, where in the first stage of iteration we can solve for ϕ and p_c iteratively. For this purpose we rewrite the Equation (4.2.26) in the following form

$${}^1p_{n+1}(\phi) = \frac{p^{tr}_{n+1} + K_n \phi\ {}^1p_{cn+1}}{1 + 2K_n \phi} \tag{4.2.30}$$

and substitute in Equation (4.2.28) to give

$$R = R({}^1p_{cn+1}) = (p_c)_n exp[\vartheta \phi \frac{2p^{tr}_{n+1} - {}^1p_{cn+1}}{1 + 2K_n \phi}] - {}^1p_{cn+1} = 0 \tag{4.2.31}$$

We can then determine the root $({}^1p_c)_{n+1}$ of Equation (4.2.31) which gives the zero of R iteratively by using a sub-local Newton-Raphson scheme with initial estimates of ${}^1p_{c\ n+1} = p^{tr}_{cn+1} = p_{cn}$ and $\phi = 0$. In the second stage of Newton-Raphson iteration we consider the initial estimates of ${}^1p_{n+1}$ and ${}^1q_{n+1}$ as the ones calculated from Equations (4.2.26) and (4.2.27) using estimated values of ${}^1p_{c\ n+1}$ and ϕ obtained

from the first stage of the iteration. The procedure in the multi-stage Newton-Raphson method used is shown in the Table 4.1.

Next, the proposed integration scheme requires the constitutive equations to be reintegrated with a second order method. The backward Euler method is selected for this purpose, which uses the converged stress state achieved from forward Euler method as base values. Then the elastic moduli K and G are to be calculated at the converged stress state obtained from forward Euler procedure, i.e. from ${}^1p_{n+1}$. Then the integrated equations of p, q and p_c are modified to give

$$p_{n+1}(\phi) = p^{tr}_{n+1} - {}^1K_{n+1}\phi\,(2p - p_c)_{n+1} \tag{4.2.32}$$

$$q_{n+1}(\phi) = q^{tr}_{n+1}\left(1 + 6\frac{{}^1G_{n+1}\phi}{M^2}\right)^{-1} \tag{4.2.33}$$

$$p_{c\ n+1}(\phi) = p_{c\ n}\ exp\left[\vartheta\phi\,(2p - p_c)_{n+1}\right] \tag{4.2.34}$$

$$F_{n+1}(\phi) = \left[\frac{q^2}{M^2} + p(p - p_c)\right]_{n+1} = 0 \tag{4.2.35}$$

Equations (4.2.32), (4.2.33), (4.2.34), and (4.2.35) are solved in the same manner as Equations (4.2.26), (4.2.27), (4.2.28) and (4.2.29), which gives the stress state at the time step t_{n+1} and which are of second order accuracy. Multi-stage Newton-Raphson method is used to solve these system of equations at local level. It is observed that integrating the constitutive equation in this manner reduces the number of iteration required to solve the global equations considerably.

In the N-R iteration scheme, the derivative of F can be obtained by employing the chain rule on F as follows:

$$F'(\phi) = \frac{\partial F}{\partial p}\frac{\partial p}{\partial \phi} + \frac{\partial F}{\partial q}\frac{\partial q}{\partial \phi} + \frac{\partial F}{\partial p_c}\frac{\partial p_c}{\partial \phi} \tag{4.2.36}$$

where $\partial F/\partial p$, $\partial F/\partial q$ and $\partial F/\partial p_c$ are given by Equation (4.2.7). Differentiating Equations (4.2.32), (4.2.33) and (4.2.34) implicitly results in

$$\frac{\partial p}{\partial \phi} = -K\frac{(2p - p_c)}{1 + (2K + \vartheta p_c)\phi} \tag{4.2.37}$$

$$\frac{\partial q}{\partial \phi} = -\frac{q}{\phi + M^2/6G} \tag{4.2.38}$$

$$\frac{\partial p_c}{\partial \phi} = \vartheta p_c\frac{(2p - p_c)}{1 + (2K + \vartheta p_c)\phi} \tag{4.2.39}$$

Table 4.1: Multi-stage Newton-Raphson iteration at local level of time step $n \rightarrow n+1$

Step No.	Description
1	$\phi = 0, \quad p_{c\ n+1} = p_{c\ n}$
2	Compute $R = R(p_{c\ n+1})$
3	If $\mid R \mid \leq$ RTOL, go to step 5 Else go to step 4
4	Compute: $p_{c\ n+1} = p_{c\ n+1} - \frac{R}{R'(p_{c\ n+1})}$ Go to step 2
5	Compute P_{n+1} , $\quad q_{n+1}$ from ϕ and $p_{c\ n+1}$ known
6	Compute $F = F(\phi, \quad p_{n+1}, \quad q_{n+1}, \quad p_{c\ n+1})$
7	If $F \leq FTOL$, EXIT Else go to step 8
8	$\phi = \phi - \frac{F}{F'(\phi)}$ Go to step 2

Calculation of consistent tangent operator plays a vital role in the convergence on Newton-Raphson iteration[150]. The next step is to get the tangential stress-strain tensor consistent with the Equations (4.2.32 - 4.2.35), which can be determined by taking the strain derivative of the incremental response function $\boldsymbol{\sigma}^i_{n+1}$ at time step $(n+1)$ after iteration i. For this purpose, the incremental response function corresponding to the strain tensor increment $\boldsymbol{\varepsilon}^i_{n+1}$ can be written as

$$\boldsymbol{\sigma}^i_{n+1} = \frac{1}{3} tr(\boldsymbol{\sigma}^i_{n+1})\boldsymbol{l} + \parallel \boldsymbol{S}^i_{n+1} \parallel \boldsymbol{n} = p^i_{n+1}\boldsymbol{l} + \sqrt{\frac{2}{3}} q^i_{n+1}\boldsymbol{n}. \tag{4.2.40}$$

The consistent tangential moduli can then be obtained by directly evaluating the variation

$$\boldsymbol{C}^i_{n+1} = \frac{\partial \boldsymbol{\sigma}^i_{n+1}}{\partial \boldsymbol{\varepsilon}^i_{n+1}} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} = l \otimes \frac{\partial p}{\partial \boldsymbol{\varepsilon}} + \sqrt{\frac{2}{3}} q \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\varepsilon}} + \sqrt{\frac{2}{3}} \boldsymbol{n} \otimes \frac{\partial q}{\partial \boldsymbol{\varepsilon}} \tag{4.2.41}$$

The subscript $(n+1)$ representing the time step and the superscript (i) representing the iteration counter are removed in the Equation (4.2.41) and also in the subsequent equations and expressions for brevity. The desired consistent tangential moduli can thus be obtained as follows. For relevant calculations of the derivatives, Appendix A can be referred.

$$\boldsymbol{C}_{n+1} = t_0 \boldsymbol{I} + t_1 \boldsymbol{l} \otimes \boldsymbol{l} + t_2 \boldsymbol{l} \otimes \boldsymbol{n} + t_3 \boldsymbol{n} \otimes \boldsymbol{l} + t_4 \boldsymbol{n} \otimes \boldsymbol{n} \tag{4.2.42}$$

where

$t_0 = 2G\xi\ ,\qquad t_1 = K(a_1 + a_2 b_1) - \frac{2}{3}G\xi\ ,\qquad t_2 = K a_2 b_2\ ,$

$t_3 = 2G\sqrt{\frac{2}{3}}(a_6 b_1)\ ,\quad t_4 = 2G\left(\sqrt{\frac{2}{3}}(a_5 + a_6 b_2) - \xi\right)\ ,\quad \xi = \frac{\|\boldsymbol{S}^k_{n+1}\|}{\|\boldsymbol{S}^{tr}_{n+1}\|}\ ,\quad K = {}^1K_{n+1}\ ,$

$G = {}^1G_{n+1},\qquad a_1 = (1 + p_c\vartheta\phi)/a,\qquad a_2 = -(2p - p_c)/a,$

$a_3 = 2p_c\vartheta\phi/a,\qquad a_4 = \vartheta\frac{p_c}{K}(2p - p_c)/a,\qquad a_5 = \sqrt{\frac{3}{2}}\left(1 + 6\frac{G\phi}{M^2}\right)^{-1},$

$a_6 = -\frac{3q}{M^2}\left(1 + 6\frac{G\phi}{M^2}\right)^{-1},\qquad a = 1 + 2K\phi + p_c\vartheta\phi,$

$b_1 = -K\left[(a_3 - 2a_1)p + a_1 p_c\right]/b,\qquad b_2 = 2G\frac{2q}{M^2}a_5/b,$

$b = -2G\frac{2q}{M^2}a_6 - K\left[(2a_2 - a_4)p - a_2 p_c\right].$

The tangential moduli presented here is consistence with the linearization of $\boldsymbol{\sigma}^k_{n+1}$ and generally yields a non-symmetric tangent operator.

4.3 Summary

A new implicit stress integration algorithm incorporating nonlinear elasticity for MCCM is presented and implemented in an object-oriented framework. In the integration algorithm the elastic moduli are treated implicitly in the elasto-plastic region, which improves the convergence in the global iteration. The integration algorithm is based on the use of secant elastic moduli to integrate the nonlinear elastic constitutive equation. The elasto-plastic constitutive relations are integrated using return mapping algorithms by making use of the both first order forward Euler scheme and the second ordered backward Euler scheme, considering associative flow rule. Numerical simulations, namely triaxial tests and a solution of boundary value problems are presented to demonstrate the accuracy and stability of the proposed algorithm. The implementation and the results of various tests are shown in Chapter 9.

Chapter 5

Nonlinear solution techniques and load stepping schemes

5.1 Introduction

In general all problems in geomechanics are significantly nonlinear. Linear numerical models may be used to roughly approximate the solution of such problems. But the error may be significant in some cases, which can not be ignored, as the results so obtained may considerably deviate from the actual solution. Therefore nonlinear material models are to be used in such situation To reach a solution in an efficient manner with a nonlinear material model, an iterative or incremental scheme is required to solve the resulting system of nonlinear equations in finite element method.

To integrate the load-displacement curve for nonlinear material models, variety of incremental and iterative schemes are developed in the past. In incremental schemes, the governing equilibrium equation, which is a set of ordinary differential equations, are solved using a series of piece-wise linear steps. In these load-stepping schemes no iteration involved over any load step and thus they are explicit in nature. Although they are robust, their solutions tend to drift from the equilibrium in a cumulative manner as the unbalanced forces generated by the numerical approximation are not being taken care properly. Moreover, minimum number of load increments required to reach a solution of acceptable accuracy are to be determined by trial and error. In iterative schemes, equilibrium is tried to maintained by iteratively finding the

solution at a given load level and hence they are implicit in nature. For highly nonlinear problems, these methods also may not reach convergence and appropriate load step size may need to be determined by trial and error only.

It can be perhaps mentioned here that starting from a simple incremental scheme to some classical iterative schemes, the accuracy, efficiency and stability of such scheme is strongly influenced by the load increment size. Critical state models are nonlinear in nature in the elastic as well as in the plastic parts of the constitutive relation. Analysis with such models using fixed increment sizes may lead to non-convergence behaviour or inaccurate solutions, if large load steps are adopted. Therefore, the performance of different load-stepping schemes for finite element analysis particularly with critical state models is to be done and can be an active area of study.

In the present study some of the load stepping schemes discussed by Sheng and Sloan [147] are presented. While implementing these schemes by Sheng and Sloan [147], an explicit integration algorithm was used to integrate the generalized Cam clay model. In the present study, an implicit integration algorithm is used to integrate the MCCM and they are implemented in a new object-oriented finite element template. The nonlinear solution techniques and load stepping schemes which are included in the present study are

1. Newton-Raphson scheme,
2. Modified Newton-Raphson scheme,
3. Arclength method.
4. Accelerated modified Newton-Raphson scheme,
5. Automatic scheme.

5.2 Newton-Raphson method

To solve a solid mechanic problem defined in the domain Ω through displacement formulation of FEM (based on small displacement theory), the objective of the

problem is to find a displacement vector **u** such that following set of equations is satisfied

$$\sum \left[\int_{\Omega} \boldsymbol{B}^T \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{u})) d\Omega \right] - \boldsymbol{Q} = 0 \tag{5.2.1}$$

where $\sum$ is the assembly operator to obtain global variables from element variables, $\boldsymbol{Q}$ is the external load vector, $\boldsymbol{B}$ is the strain displacement matrix, and $\boldsymbol{\sigma}$ is the Cauchy stress tensor. In more compact form Equation (5.2.1) may be written as

$$\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{Q} = \boldsymbol{R}(\boldsymbol{u}) = 0 \tag{5.2.2}$$

where $\boldsymbol{F}(\boldsymbol{u})$ is the internal force vector, and $\boldsymbol{R}(\boldsymbol{u})$ is the residual force vector. Equation (5.2.2) is a nonlinear equation in $\boldsymbol{u}$, which needs to be solved using an iterative procedure. If Newton-Raphson method is to be used to solved Equation (5.2.2), in ith iteration we look for the unknown displacement $\boldsymbol{u}^i = \boldsymbol{u}^{i-1} + \delta\boldsymbol{u}^i$, such that $\boldsymbol{R}^i = \boldsymbol{R}(\boldsymbol{u}^i) = 0$, where we assume the displacement $\boldsymbol{u}^{i-1}$ to exist such that $\boldsymbol{R}(\boldsymbol{u}^{i-1}) = \boldsymbol{R}^{i-1} \neq 0$, . Using a two terms Taylor series expansion for $\boldsymbol{R}(\boldsymbol{u}^i)$

$$\boldsymbol{R}(\boldsymbol{u}^i) = \boldsymbol{R}(\boldsymbol{u}^{i-1}) + \frac{\partial \boldsymbol{R}(\boldsymbol{u})}{\partial \boldsymbol{u}} |_{\boldsymbol{u}=\boldsymbol{u}^{i-1}} \delta\boldsymbol{u}^i = 0$$

$$\Rightarrow \boldsymbol{R}^i = \boldsymbol{R}^{i-1} + \boldsymbol{K}_t^{i-1} \delta\boldsymbol{u}^i = 0$$

$$\Rightarrow \boldsymbol{K}_t^{i-1} \delta\boldsymbol{u}^i = -\boldsymbol{R}^{i-1} \tag{5.2.3}$$

where

$$\boldsymbol{K}_t^{i-1} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} |_{\boldsymbol{u}=\boldsymbol{u}^{i-1}} = \sum [\int_{\Omega} \boldsymbol{B}^T \frac{\partial \boldsymbol{\sigma}(\boldsymbol{\varepsilon})}{\partial \boldsymbol{u}} d\Omega] |_{\boldsymbol{u}=\boldsymbol{u}^{i-1}}$$

$$= \sum [\int_{\Omega} \boldsymbol{B}^T \frac{\partial \boldsymbol{\sigma}(\boldsymbol{\varepsilon})}{\partial \boldsymbol{\varepsilon}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{u}} d\Omega] |_{\boldsymbol{u}=\boldsymbol{u}^{i-1}} = \sum [\int_{\Omega} \boldsymbol{B}^T \boldsymbol{C}^{ep} \boldsymbol{B} d\Omega] |_{\boldsymbol{u}=\boldsymbol{u}^{i-1}} \tag{5.2.4}$$

Equation (5.2.3) is solved for $\delta\boldsymbol{u}^i$, the displacement is updated as $\boldsymbol{u}^i = \boldsymbol{u}^{i-1} + \delta\boldsymbol{u}^i$ and the internal force vector $\boldsymbol{F}(\boldsymbol{u}^i) = \boldsymbol{F}^i$ corresponding to this displacement vector is evaluated. In Newton-Raphson method, the load is applied in an incremental fashion, i.e., the external load vector $\boldsymbol{Q}$ is divided into number of increments as $\boldsymbol{Q}^1$, $\boldsymbol{Q}^2$, $\boldsymbol{Q}^3$, ..etc, such as $\sum \boldsymbol{Q}^j = \boldsymbol{Q}$, where j stands for increment number. This

incremental external load applied to the nonlinear FEM is fixed at the beginning of each step or increment. The corresponding displacement at each node are to be found iteratively by calculating $\delta \boldsymbol{u}^i$ from Equation (5.2.3), as shown in Figure 5.1. The iteration process is stopped when the forces due to internal stresses become equal to the external fixed load. The convergence criterion to terminate the iteration is applied as follows:

$$\frac{\|\boldsymbol{R}^i\|}{\|\boldsymbol{Q}^j\|} = \frac{\|\boldsymbol{Q}^j - \boldsymbol{F}^i\|}{\|\boldsymbol{Q}^j\|} \leq tol \tag{5.2.5}$$

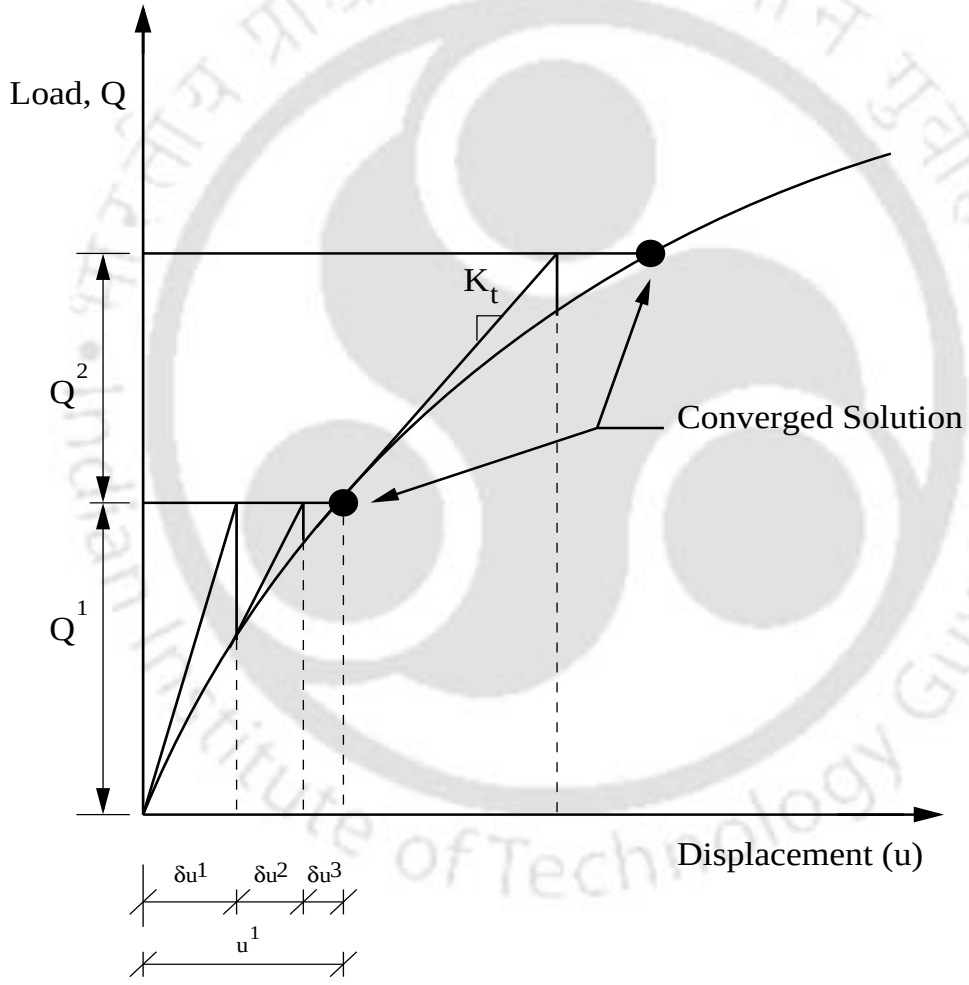


Figure 5.1: Newton-Raphson iterative solution

If Equation (5.2.5) is not satisfied, a new stiffness matrix $\boldsymbol{K}_t^i$ will have to be obtained based on the new displacement vector and Equation (5.2.3) has to be solved for next $\delta \boldsymbol{u}^{i+1}$ corresponding to the residual load $(\boldsymbol{Q}^j - \boldsymbol{F}^i)$.

5.3 Modified Newton-Raphson method

In Newton-Raphson method the stiffness matrix in Equation (5.2.3) is updated after each iteration for a given load increment. To reduce the cost of computation the stiffness matrix can be updated only once at the beginning of a load increment and kept constant throughout the iteration process of the specified increment.This version of Newton-Raphson method is called as modified Newton-Raphson method. The graphical representation of the method is shown in Figure 5.2. As $\boldsymbol{K}_t$ remains constant throughout a load increment, the number of iterations required per increment will increase in this case. But the computation cost is reduced from the point of view of computation of element stiffness matrices and inversion of the global stiffness matrix. Modified Newton-Raphson method does not possess the quadratic convergence. Thus there is a trade off between calculation of stiffness matrix and number of iterations.

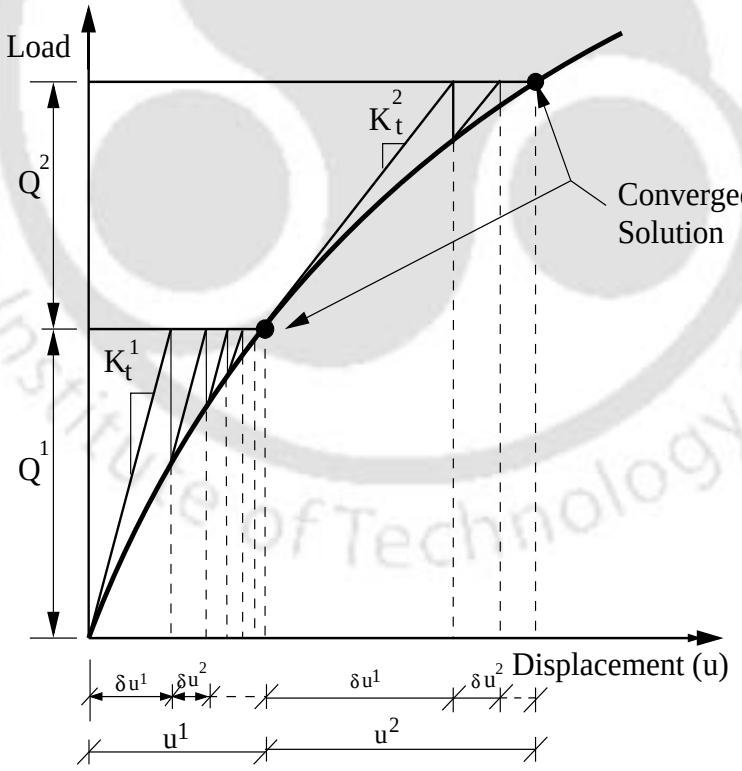


Figure 5.2: Modified Newton-Raphson iterative solution

5.4 Arclength method

Arclength method is one of the solution procedures used to trace equilibrium paths beyond limit points. The method was originally proposed by Riks and Wempner and was modified by Crisfield and Ramm [39]. The method can be easily implemented in a standard finite element program. Crisfield [39] improved this method by incorporating line searchers and accelerations, which can lead to a substantial improvement in the convergence characteristics in nonlinear analysis.

The basic features of the standard arclength method are given in number of literature. Bashir-Ahmed and Xiao-zu [107] reviewed recent developments of arclength during the last two decades and discussed related key issues. In any load controlled method (e.g. Newton-Raphson method) the external load is fixed during a time step, and similarly in a displacement controlled method the displacement is fixed for each iteration within a load or time step. In contrast to that, in arclength method, the load level is treated to be variable at each step. At each iteration within a time step, the load level is modified so that the solution follows some specified path until convergence.

In arclength method, the governing equilibrium equation for a nonlinear system is written as

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{F}(\boldsymbol{u}) - \lambda \boldsymbol{Q} \tag{5.4.6}$$

where λ is the load level (variable), $\boldsymbol{Q}$ is the fixed total load vector, $\boldsymbol{F}$ is the internal force and $\boldsymbol{R}$ is the vector of out of balance forces. As $\boldsymbol{F}$ is a function of current displacement $\boldsymbol{u}$, hence so is $\boldsymbol{R}$. To handle the variable λ one extra governing equation is required and is given by the so-called spherical constraint relationship of the following form [39]

$$\Delta \boldsymbol{u}^{i+1^T} \Delta \boldsymbol{u}^{i+1} + b(\Delta\lambda^{i+1})^2 \boldsymbol{Q}^T \boldsymbol{Q} = \Delta \boldsymbol{u}^{i^T} \Delta \boldsymbol{u}^{i} + b(\Delta\lambda^{i})^2 \boldsymbol{Q}^T \boldsymbol{Q} = ...\Delta l^2 \tag{5.4.7}$$

where $\Delta \boldsymbol{u}^i$ are incremental displacements after $(i-1)$th iteration, $\Delta\lambda^i$ is the incremental load change after this iteration, and Δl is a prescribed incremental length, called as arclength. Crisfield [39] used the spherical constraint of Equation (5.4.7) with $b = 0$, and the following initial values of the adopted notations

$\boldsymbol{u}^0$ =displacement at the end of previous increment, $\Delta \boldsymbol{u}^0 = 0$

$\boldsymbol{F}^0(\boldsymbol{u}^0)$ = internal force vector at end of last increment

$\boldsymbol{F}^0(\boldsymbol{u}^0) = \lambda^0 \boldsymbol{Q}$, with perfect convergence, where λ^0 = load level at the end of last increment.

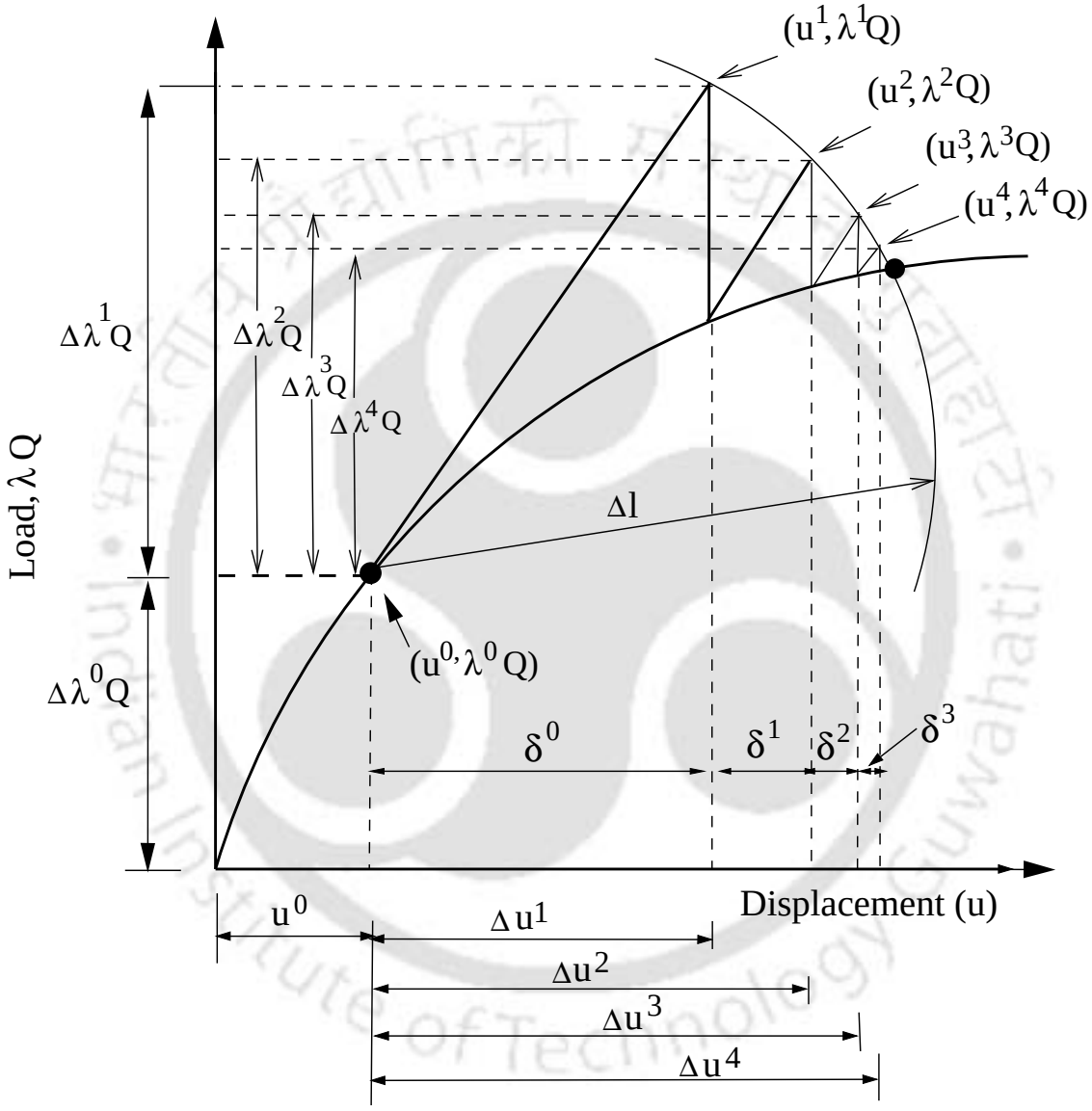


Figure 5.3: Basic arclength procedure

At the beginning of the solution procedure, we specify one arclength from the last converged point, say Δl. The x-coordinate of point of intersection of the tangent at previous converged point and arc gives the next approximation of increment of u, say $\Delta \boldsymbol{u}^{i+1}$, as shown in Figure 5.3. The iterative procedure starts from i=0 onwards

with

$$\lambda^{i+1} = \lambda^0 + \Delta\lambda^{i+1} = \lambda^i + \Delta\lambda^i \tag{5.4.8}$$

$$\boldsymbol{u}^{i+1} = \boldsymbol{u}^0 + \Delta\boldsymbol{u}^{i+1} = \boldsymbol{u}^i + \eta^i\delta^i \tag{5.4.9}$$

where

$$\Delta\boldsymbol{u}^{i+1} = \Delta\boldsymbol{u}^i + \eta^i\boldsymbol{\delta}^i \tag{5.4.10}$$

Here $\Delta\boldsymbol{u}^i$ and $\boldsymbol{\delta}^i$ stand for incremental and iterative displacement vectors respectively, and η^i is the step length, which is assumed unity as in Figure 5.3. The tangent stiffness matrix $\boldsymbol{K}_t$ is formed at the beginning of each increment and kept fixed for all iterations within an increment like modified Newton-Raphson method. Using the indirect solution procedure given by Crisfield and Ramm [39], the ith iterative displacement vector of an increment is given by

$$\boldsymbol{\delta}^i = -\boldsymbol{K}_t^{-1}[\boldsymbol{F}(\boldsymbol{u}^i) - \lambda^{i+1}\boldsymbol{Q}] = \bar{\boldsymbol{\delta}}^i + \lambda^{i+1}\boldsymbol{\delta}^T \tag{5.4.11}$$

where

$$\bar{\boldsymbol{\delta}}^i = -\boldsymbol{K}_t^{-1}\boldsymbol{F}(\boldsymbol{u}^i)\ , \quad \boldsymbol{\delta}^T = \boldsymbol{K}_t^{-1}\boldsymbol{Q} \tag{5.4.12}$$

The vector $\boldsymbol{\delta}^T$, called as tangential displacement vector, is fixed for all iterations within an increment and is formed at the beginning of an increment. Assuming perfect convergence at the last increment, i.e. $\boldsymbol{F}(\boldsymbol{u}^0) = \lambda^0\boldsymbol{Q}$, the iterative displacement vector $\boldsymbol{\delta}^0$ can be formed at the beginning of an increment from

$$\boldsymbol{\delta}^0 = -\boldsymbol{K}_t^{-1}[\boldsymbol{F}(\boldsymbol{u}^0) - \lambda^1\boldsymbol{Q}] = (\lambda^1 - \lambda^0)\boldsymbol{K}_t^{-1}\boldsymbol{Q} = \Delta\lambda^1\boldsymbol{\delta}^T \tag{5.4.13}$$

Here we see that the iterative displacement vector $\boldsymbol{\delta}^i$ is dependent on λ^{i+1}. To find λ^{i+1} we substitute Equation (5.4.10) and Equation (5.4.11) into the constraint Equation (5.4.7) with b=0, which gives

$$a_1(\lambda^{i+1})^2 + a_2\lambda^{i+1} + a_3 = 0 \tag{5.4.14}$$

where

$$\left.\begin{aligned} a_1 &= \eta^i({\boldsymbol{\delta}^T}^T\boldsymbol{\delta}^T) = \eta^i c_1 \\ a_2 &= 2({\boldsymbol{\delta}^T}^T\Delta\boldsymbol{u}^i) + 2\eta^i({\boldsymbol{\delta}^T}^T\bar{\boldsymbol{\delta}}^i) = 2d_1 + 2\eta^i d_2 \\ a_3 &= \eta^i({\bar{\boldsymbol{\delta}}^i}^T\bar{\boldsymbol{\delta}}^i) + 2({\bar{\boldsymbol{\delta}}^i}^T\Delta\boldsymbol{u}^i) + ({\Delta\boldsymbol{u}^i}^T\Delta\boldsymbol{u}^i - \Delta l^2) \\ &= \eta^i d_3 + 2d_4 + ({\Delta\boldsymbol{u}^i}^T\Delta\boldsymbol{u}^i - \Delta l^2) \end{aligned}\right\} \quad (A)$$

Here $\boldsymbol{\delta}^T$, $\Delta\boldsymbol{u}^i$ and $\bar{\boldsymbol{\delta}}^i$ are all known. With step length $\eta^i = 1$, Equation (5.4.14) can be easily solved for λ^{i+1}. If we assume that the constraint Equation (5.4.7) (with b=1) is fully satisfied at the previous iteration, the terms in square brackets in Equation (A) will become zero and can be omitted. Equation (5.4.14) will have two roots and appropriate root is selected by ensuring an acute angle 'θ' between $\Delta\boldsymbol{u}^i$ and $\Delta\boldsymbol{u}^{i+1}$, for which $cos\,\theta$ will be positive and is given by

$$cos\,\theta = 1 + \frac{\eta^i}{\Delta l^2}(d_4 + \lambda^{i+1} d_1) \tag{5.4.15}$$

If both the roots of λ^{i+1} give positive value of $cos\,\theta$, then the appropriate root of λ^{i+1} may be selected as the root closest to the linear solution of Equation (5.4.14), which is

$$\lambda^{i+1,lin} = -a_3/a_2 \tag{5.4.16}$$

The value of $\Delta\lambda$ is suitably assumed for the first load increment. Knowing the tangential displacement vector $\boldsymbol{\delta}^T$, the corresponding arclength Δl with $i = 0$ and $b = 0$ is calculated as follows:

From Equation (5.4.7), $\quad \Delta\boldsymbol{u}^{1^T}\Delta\boldsymbol{u}^1 = \Delta l^2$ (i)

Using Equation (5.4.10) in (i), $\quad \Delta l^2 = (\Delta\boldsymbol{u}^0 + \eta^0\boldsymbol{\delta}^0)^T(\Delta\boldsymbol{u}^0 + \eta^0\boldsymbol{\delta}^0)$ (ii)

Using $\eta^0 = 1$ *and* $\Delta\boldsymbol{u}^0 = 0$ *in* (*ii*), $\quad \Delta l^2 = \boldsymbol{\delta}^{0^T}\boldsymbol{\delta}^0$ (iii)

Using Equation (5.4.13) for $\boldsymbol{\delta}^0$ in (iii) we get

$$\Delta l = \Delta\lambda^1\sqrt{[\boldsymbol{\delta}^{T^T}\boldsymbol{\delta}^T]} \tag{5.4.17}$$

For all iterations within an increment, Δl is kept constant and its magnitude for the next load increment (say jth) is given by

$$\Delta l_j = \sqrt{(\frac{I_{j-1}}{I_d})}\Delta l_{j-1} \tag{5.4.18}$$

where I_{j-1} is the number of iteration required for the $(j-1)$th increment and I_d is the desired number of iterations. To start a new increment, the first estimate of $\Delta\lambda^1$ can be obtained from Equation (5.4.17)

$$\Delta\lambda^1 = \sqrt{[\frac{\Delta l^2}{\boldsymbol{\delta}^{T^T}\boldsymbol{\delta}^T}]} = \frac{\pm\Delta l}{\sqrt{[\boldsymbol{\delta}^{T^T}\boldsymbol{\delta}_T]}} = \frac{a(\Delta l)}{\sqrt{[\boldsymbol{\delta}^{T^T}\boldsymbol{\delta}^T]}} \tag{5.4.19}$$

where $\quad a = sign(r)$ and $r = \boldsymbol{\delta}^{T^T}\boldsymbol{Q} = \boldsymbol{\delta}^{T^T}\boldsymbol{K}_t\boldsymbol{\delta}^T$.

The scalar r is called an unscaled version of Bergan's current stiffness parameter.

The arclength method explain above is implemented with the MCCM in an object-oriented template. The results are shown in Chapter 9.

5.5 Accelerated modified Newton-Raphson method

In modified Newton-Raphson method the stiffness matrix is computed only once at the beginning of a load increment and is kept constant for each iteration within that load increment. This reduces the computing cost, but the rate of convergence is also reduced accordingly. To accelerate the convergence rate of modified Newton-Raphson method, the displacement at each iteration can be scaled with some acceleration factors. When this technique is used, the method can be called as accelerated modified Newton-Raphson method. In one family of such accelerated modified Newton-Raphson scheme, a single parameter α is used to scale the iterative displacement and the steps involved are discussed below.

The global equilibrium equation for displacement finite element method can be given by Equation(5.4.6) and at the end of $(i-1)$th iteration within a load step it can be written as

$$\boldsymbol{R}(\boldsymbol{u}^{i-1}) = \boldsymbol{R}^{i-1} = \boldsymbol{Q}_n - \int_\Omega \boldsymbol{B}^T \boldsymbol{\sigma}(\boldsymbol{u}^{i-1}) d\Omega \tag{5.5.20}$$

where $\boldsymbol{Q}_n = \boldsymbol{Q}_0 + \Delta\boldsymbol{Q}$, $\boldsymbol{Q}_0$ is the total external load at the end of last converged step, $\Delta\boldsymbol{Q}$ is the load increment during the current load step, $\boldsymbol{R}^{i-1}$ is the residual load at the end of $(i-1)$th iteration, and $\boldsymbol{\sigma}(\boldsymbol{u}^{i-1})$ is the total internal stress at the end of $(i-1)$th iteration of the current load step. Adopting the forward Euler load stepping method, the iterative displacement vector for ith iteration corresponding to the residual load $\boldsymbol{R}^{i-1}$ is given as

$$\delta\boldsymbol{u}_0^i = [\boldsymbol{K}_t(\boldsymbol{u}_0)]^{-1}\boldsymbol{R}^{i-1} \tag{5.5.21}$$

Thus the displacement vector at the end of ith iteration is given as

$$\boldsymbol{u}^i = \boldsymbol{u}^{i-1} + \alpha^i \delta\boldsymbol{u}_0^i \tag{5.5.22}$$

where α^i is the acceleration factor for ith iteration, $\boldsymbol{K}_t(\boldsymbol{u}_0)$ is the tangent stiffness matrix, $\delta\boldsymbol{u}_0^i$ is the iterative displacement at ith iteration and $\boldsymbol{u}_0$ is the displacement vector at the end of last converged step. There are different acceleration methods

developed based on the methods of computing the value of α. Examples of such methods are Thomas accelerator and Chen accelerator, which were initially developed for accelerating initial stiffness scheme. Sheng and Sloan [147] reported that a good review of these methods along with their comparison in terms of performance while applying to conventional plasticity models can be found in Abbo and Sloan [1] and Sloan et al. [146] and a modified form of Thomas procedure for Tresca and Mohr-Coulomb soil models was also proposed by these investigators. All these acceleration procedures were developed for conventional initial stiffness scheme, but they can be effectively extended to modified Newton-Raphson scheme. In the following sections the modified Newton-Raphson method with Chen accelerator and modified Thomas accelerator as adopted by Sheng and Sloan [147] is presented.

5.5.1 Modified Newton-Raphson method with Chen accelerator

In this algorithm the scaling factor α is to be determined so as to minimize the unbalance forces at the end of a given iteration. For ith iteration, its value α^i is determined by assuming that the unbalance forces $\boldsymbol{R}^i$ at the end of ith iteration varies linearly in accordance with the following equation

$$\boldsymbol{R}^i = \boldsymbol{R}^{i-1} - \alpha^i(\boldsymbol{R}_0^i - \boldsymbol{R}^{i-1}) \tag{5.5.23}$$

where

$$\boldsymbol{R}^i = \boldsymbol{R}(\boldsymbol{u}^i) = \boldsymbol{R}(\boldsymbol{u}^{i-1} + \alpha^i \delta \boldsymbol{u}_0^i) \ , \quad \boldsymbol{R}_0^i = \boldsymbol{R}(\boldsymbol{u}^{i-1} + \delta \boldsymbol{u}_0^i)$$

And $\delta \boldsymbol{u}_0^i$ is given by Equation (5.5.21). For least square minimization of the unbalanced forces $\boldsymbol{R}^i$, it is required that

$$\frac{d}{d\alpha^i}[\{\boldsymbol{R}^i\}^T\{\boldsymbol{R}^i\}] = 0 \tag{5.5.24}$$

Substituting Equation (5.5.23) in Equation (5.5.24) and then solving for α^i, we get

$$\alpha^i = \frac{\{\boldsymbol{R}_0^i - \boldsymbol{R}^{i-1}\}^T\{\boldsymbol{R}^{i-1}\}}{\{\boldsymbol{R}_0^i - \boldsymbol{R}^{i-1}\}^T\{\boldsymbol{R}_0^i - \boldsymbol{R}^{i-1}\}} \tag{5.5.25}$$

Knowing the value of α^i, the displacement for the current load step at the end of ith iteration is found by Equation (5.5.22). In this algorithm, we need to perform

one back substitution to compute $\delta \boldsymbol{u}_0^i$, and two unbalance forces evaluation, i.e., $\boldsymbol{R}^{i-1}$ and $\boldsymbol{R}_0^i$. While evaluating the unbalanced forces, the constitutive laws should be integrated using the incremental strain and not the iterative strain, i.e., the stress increments are always evaluated with respect to the last converged state of equilibrium. In this algorithm, no restriction is placed on the value of α. If the value of α is greater than unity, it may be viewed as accelerated iteration, if its value is less than unity, it may be viewed as stabilizing iteration, and if its value is equal to unity, it corresponds to the standard modified Newton-Raphson iteration (Sheng and Sloan [147]).

From the present study it is found that while this algorithm is applied to MCCM, the minimum value of α should be kept equal to or more than 0.1, so as to reduce the number of iteration effectively.

5.5.2 Modified Newton-Raphson method with modified Thomas accelerator

The Thomas accelerator, which was originally developed for conventional elasto-plastic models with initial stiffness method, was modified by Sloan et al. [146] to implement it in finite element code. The same was again modified by Sheng and Sloan [147] to incorporate it into critical state models.

In this algorithm also to determine the scaling factor α, least square minimization technique is used in the similar way as in the Chen acceleration scheme. But in this case, the unbalance forces $\boldsymbol{R}^i$ at the end of ith iteration within a load step is given by

$$\boldsymbol{R}(\boldsymbol{u}^i) = \boldsymbol{R}^i = \boldsymbol{R}^{i-1} - \alpha^i \boldsymbol{K}_t(\boldsymbol{u}^{i-1}) \delta \boldsymbol{u}_0^i \tag{5.5.26}$$

where $\delta \boldsymbol{u}_0^i$ is given by the Equation (5.5.21).

In Equation (5.5.26), the tangent stiffness matrix $\boldsymbol{K}_t(\boldsymbol{u}^{i-1})$ is determined from the previous iteration and can be decomposed into two parts in accordance with the following equation

$$\boldsymbol{K}_t(\boldsymbol{u}^{i-1}) = \boldsymbol{K}_t^{i-1} = \boldsymbol{K}_t(\boldsymbol{u}_0) - \Delta \boldsymbol{K}_t(\boldsymbol{u}^{i-1}) \ = \boldsymbol{K}_t^0 - \Delta \boldsymbol{K}_t^{i-1} \tag{5.5.27}$$

Here $\boldsymbol{K}_t^0$ is the tangent stiffness at the end of last converged load step and is a

function of displacement $\boldsymbol{u}_0$ at that stage and $\Delta \boldsymbol{K}_t^{i-1}$ is the change in tangent stiffness from the last converged step to the $(i-1)$th iteration of the current load increment and is unknown. Substituting Equation (5.5.27) in Equation (5.5.26) we get

$$\boldsymbol{R}^i = \boldsymbol{R}^{i-1} - \alpha^i [\boldsymbol{K}_t^0 - \Delta \boldsymbol{K}_t^{i-1}] \delta \boldsymbol{u}_0^i \tag{5.5.28}$$

Using the approximation $\alpha^i \Delta \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i \simeq \alpha^{i-1} \Delta \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i$ in Equation (5.5.28) we get

$$\begin{aligned} \boldsymbol{R}^i &= \boldsymbol{R}^{i-1} - \alpha^i \boldsymbol{K}_t^0 \delta \boldsymbol{u}_0^i + \alpha^{i-1} \Delta \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i \\ \Rightarrow \boldsymbol{R}^i &= \boldsymbol{R}^{i-1} + (\alpha^{i-1} - \alpha^i) \boldsymbol{K}_t^0 \delta \boldsymbol{u}_0^i - \alpha^{i-1} (\boldsymbol{K}_t^0 - \Delta \boldsymbol{K}_t^{i-1}) \delta \boldsymbol{u}_0^i \\ \Rightarrow \boldsymbol{R}^i &= \boldsymbol{R}^{i-1} + (\alpha^{i-1} - \alpha^i) \boldsymbol{K}_t^0 \delta \boldsymbol{u}_0^i - \alpha^{i-1} \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i \end{aligned} \tag{5.5.29}$$

Now using Equation (5.5.22) in Equation (5.5.26) we can write

$$\boldsymbol{R}(\boldsymbol{u}^i) = \boldsymbol{R}(\boldsymbol{u}^{i-1} + \alpha^i \delta \boldsymbol{u}_0^i) = \boldsymbol{R}^{i-1} - \alpha^i \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i$$

Similarly we can write

$$\boldsymbol{R}(\boldsymbol{u}^i) = \boldsymbol{R}(\boldsymbol{u}^{i-1} + \alpha^{i-1} \delta \boldsymbol{u}_0^i) = \boldsymbol{R}^{i-1} - \alpha^{i-1} \boldsymbol{K}_t^{i-1} \delta \boldsymbol{u}_0^i$$

Using the above relation in Equation (5.5.29) we get

$$\boldsymbol{R}^i = (\alpha^{i-1} - \alpha^i) \boldsymbol{K}_t^0 \delta \boldsymbol{u}_0^i + \boldsymbol{R}(\boldsymbol{u}^{i-1} + \alpha^{i-1} \delta \boldsymbol{u}_0^i)$$

or

$$[\boldsymbol{K}_t^0]^{-1} \boldsymbol{R}^i = (\alpha^{i-1} - \alpha^i) \delta \boldsymbol{u}_0^i + \delta \tilde{\boldsymbol{u}}_0^i \tag{5.5.30}$$

where

$$\delta \tilde{\boldsymbol{u}}_0^i = [\boldsymbol{K}_t^0]^{-1} \boldsymbol{R}(\boldsymbol{u}^{i-1} + \alpha^{i-1} \delta \boldsymbol{u}_0^i)$$

To get the value of α^i, we need to perform least square minimization of the displacement $[\boldsymbol{K}_t^0]^{-1} \boldsymbol{R}^i$, from Equation (5.5.30)

$$\frac{d}{d\alpha^i} \{[\boldsymbol{K}_t^0]^{-1} \boldsymbol{R}^i\}^T \{[\boldsymbol{K}_t^0]^{-1} \boldsymbol{R}^i\} = 0$$

which gives

$$\alpha^i = \alpha^{i-1} + \frac{\{\delta \boldsymbol{u}_0^i\}^T \{\delta \tilde{\boldsymbol{u}}_0^i\}}{\{\delta \boldsymbol{u}_0^i\}^T \{\delta \boldsymbol{u}_0^i\}} \tag{5.5.31}$$

In the Chen accelerator, α^i for the ith iteration is determined by minimizing the unbalanced forces at the end of the iteration. But Thomas accelerator attempts

to minimize the iterative displacement corresponding to the next iteration, i.e., $[\boldsymbol{K}_t^o]^{-1}\boldsymbol{R}^i = \delta\boldsymbol{u}_0^{i+1}$, to get the value of α^i for the current ith iteration. Once α^i is known, the displacement $\boldsymbol{u}^i$ may me updated using Equation (5.5.22). But this only provides moderate acceleration. Thomas adopted a second option to update u^i, which was also used by Sheng and Sloan [147]. It is based on the least square minimization of the displacement vector $[\boldsymbol{K}_t^0]^{-1}\boldsymbol{R}^i$ and is given by the following equation

$$\boldsymbol{u}^i = \boldsymbol{u}^{i-1} + \alpha^{i-1}\delta\boldsymbol{u}_0^i + \alpha^i\delta\tilde{\boldsymbol{u}}_0^i \tag{5.5.32}$$

From Equation (5.5.32), it can be observed that the acceleration factors used to update displacements are both from previous iteration as well as from current iteration. In the beginning of each load increment, the value of α^{i-1} for first iteration can be assumed as unity, i.e., $\alpha^0 = 1$.

In this algorithm, for each iteration, we need to perform two back substitution to find out two iterative displacements $\delta\boldsymbol{u}_0^i$ and $\delta\tilde{\boldsymbol{u}}_0^i$, and two unbalance forces evaluation. This is double the work required for a standard modified Newton-Raphson method. Again there is no restriction placed on the value of α^i within an increment. Like Chen accelerator, in this method also while evaluating the unbalanced forces, incremental strain should be used to integrate the constitutive laws to evaluate the stress increments with respect to the last converged state of equilibrium. From the present study it is found that while using this algorithm to MCCM, the maximum value of α should be kept equal to or less than 15, so as to reduce the number of iteration effectively.

5.6 Automatic schemes

In all the iterative schemes so far discussed, i.e., Newton-Raphson scheme, modified Newton-Raphson scheme, accelerated modified Newton-Raphson scheme, the magnitude of load increments within each increment are fixed. While keeping the load increment fixed, the size of each load increment are generally to be kept equal in magnitude. Approximate number of such equal load increments for a particular problem are determined by trial and error. This approach of using equal size load increments are seen to be successful when large number of load increments are used, which may be time consuming and costly for large scale problems. Again there may

arise convergence problem if appropriate size of load increments are not used for highly nonlinear material models. In case of critical state models analysis with finite load increments may lead to some load path error in the resulting displacement and stresses, and this error will go on increasing with the increase in the size of the load increment.

In the subsequent section, two automatic incremental schemes, namely (i) Automatic incremental scheme and (ii) Automatic iterative scheme, developed by Abbo and Sloan [1] are discussed, where the size of the load increments can be automatically adjusted to reduce the load path error. Out of these two schemes one is explicit without any iteration and the other is implicit in nature, which involves iterations. In both the methods a series of trial load load steps are assumed first and they may be sub-incremented if needed to keep the local load path error bellow a prescribed tolerance. Both the methods are reported to be robust and efficient while applying with conventional plasticity models and have been implemented for a generalized Cam clay model by Sheng and Sloan [147] using an explicit integration algorithm.

5.6.1 Automatic incremental scheme

The automatic increment scheme presented by Abbo and Sloan [1] can choose small load increments automatically wherever required and is described here. In this scheme, sub-incrementation of a given load increment is done automatically while computing the load displacement response to control the local load path error in the displacement. Incremental solution for displacement is obtained by first order accurate Euler scheme and second order accurate modified Euler scheme. Then the error in the displacement is obtained from the difference between these two solutions. This scheme is explicit in nature involving no iteration, and therefore to prevent the accumulation of global error an unbalance force correction can be included.

In displacement finite element analysis of rate independent plasticity, the global stiffness Equations (5.5.21) for a given load increment $\Delta \boldsymbol{Q}$, without unbalance force, can be written as

$$\Delta \boldsymbol{u} = [\boldsymbol{K}_t(\boldsymbol{u})]^{-1} \Delta \boldsymbol{Q} \tag{5.6.33}$$

In the automatic load increment scheme, Equation (5.6.33) is solved by employing

the concept of pseudo time sub-step in the range of $0 \leq \Delta T_i \leq 1$, for a sub-incremental load vector of $\Delta \boldsymbol{Q}_i = \Delta T_i \Delta \boldsymbol{Q}$. Ignoring the unbalanced forces at the start of each sub-increment, the Euler and modified Euler solutions of the Equation (5.6.33) at the pseudo time steps ΔT_{i-1} and ΔT_i may be written as

$$\boldsymbol{u}_i = \boldsymbol{u}_{i-1} + \Delta \boldsymbol{u}_1 \tag{5.6.34}$$

$$\tilde{\boldsymbol{u}}_i = \boldsymbol{u}_{i-1} + \frac{1}{2}(\Delta \boldsymbol{u}_1 + \Delta \boldsymbol{u}_2) \tag{5.6.35}$$

where

$$\Delta \boldsymbol{u}_1 = [\boldsymbol{K}_t(\boldsymbol{u}_{i-1})]^{-1}\{\Delta \boldsymbol{Q}_i\} \tag{5.6.36}$$

$$\Delta \boldsymbol{u}_2 = [\boldsymbol{K}_t(\boldsymbol{u}_{i-1} + \Delta \boldsymbol{u}_1)]^{-1}\{\Delta \boldsymbol{Q}_i\} \tag{5.6.37}$$

The error in $\boldsymbol{u}_i$ can be estimated by taking any convenient norm of difference of the lower-order solution and the higher-order solution, and then by dividing it by the displacement norm to convert it to a dimensionless parameter given as

$$R_i \simeq \frac{\frac{1}{2} \parallel (\Delta \boldsymbol{u}_2 - \Delta \boldsymbol{u}_1) \parallel}{\parallel \boldsymbol{u}_i \parallel} \tag{5.6.38}$$

R_i is called as relative error measure and can be used to control the size of the load sub-increments. The current load sub-increment is accepted if R_i is less than a user specific tolerance, $DTOL$, otherwise rejected. Then the next sub-incremental load is calculated according to the following equations

$$q = .7\sqrt{DTOL/R_i} \tag{5.6.39}$$

$$\Delta T_{i+1} = q \Delta T_i \tag{5.6.40}$$

$$\Delta \boldsymbol{Q}_i = \Delta T_{i+1} \Delta \boldsymbol{Q} \tag{5.6.41}$$

In case a sub-increment fails, the value of q is limited to 0.1 and in case of a successful sub-increment, q is limited to 1.1. These limits of q was suggested by Abbo and Sloan [1] to limit the number of rejected sub-steps. In the above interpretation, the unbalance force is not included. To incorporate the effect of unbalance force in this scheme, Equations (5.6.34) and (5.6.37) can be replace with

$$\boldsymbol{u}_i = \boldsymbol{u}_{i-1} + \Delta \boldsymbol{u}_1 + \Delta \boldsymbol{u}_1^{unb} \tag{5.6.42}$$

$$\Delta \boldsymbol{u}_2 = [\boldsymbol{K}_t(\boldsymbol{u}_{i-1} + \Delta \boldsymbol{u}_1 + \Delta \boldsymbol{u}_1^{unb})]^{-1}\{\Delta \boldsymbol{Q}_i\} \tag{5.6.43}$$

where

$$\Delta \boldsymbol{u}_1^{unb} = [\boldsymbol{K}_t(\boldsymbol{u}_{i-1})]^{-1}\{\boldsymbol{R}(\boldsymbol{u}_{i-1})\} \tag{5.6.44}$$

The step by step implementation of the scheme can given as follows:

(1) Start the integration scheme with present stresses $\boldsymbol{\sigma}_0$, displacements $\boldsymbol{u}_0$, external load increment $\Delta \boldsymbol{Q}$, previous pseudo time sub-step ΔT_{last} and displacement error tolerance $DTOL$.

(2) Initialize: $T = 0; \quad i = 1; \quad \Delta T_i = \min\{\Delta T_{last}, 1\}$

(3) While $T > 1$ go to step 10.

(4) Compute: $\Delta \boldsymbol{Q}_i = \Delta T_i \Delta \boldsymbol{Q}; \quad \Delta \boldsymbol{u}_1 = [\boldsymbol{K}(\boldsymbol{u}_{i-1})]^{-1} \Delta \boldsymbol{Q}_i$

(5) Compute first order displacement update: $\boldsymbol{u}_i = \boldsymbol{u}_0 + \Delta \boldsymbol{u}_1$

Then integrate the constitutive law to find corresponding state of stress $\boldsymbol{\sigma_i}$

(6) Compute: $\Delta \boldsymbol{u}_2 = [\boldsymbol{K}(\boldsymbol{u}_i)]^{-1} \Delta \boldsymbol{Q}_i$

(7) Compute error: $R_i = \dfrac{\frac{1}{2} \parallel (\Delta \boldsymbol{u}_2 - \Delta \boldsymbol{u}_1) \parallel}{\parallel \boldsymbol{u}_i \parallel}$

(8) If $R_i \leq DTOL$ go to step 9. Else current sub-increment has failed. So estimate a smaller pseudo time step using

$$q = max\{.7\sqrt{DTOL/R_i},\ 0.1\}$$

Set $\Delta T_i = q \Delta T_i$, and return to step 4.

(9) Current load step is successful. Update pseudo time, displacements, and stresses according to

$$T = T + \Delta T_i; \quad \boldsymbol{u}_0 = \boldsymbol{u}_i; \quad \boldsymbol{\sigma}_0 = \boldsymbol{\sigma}_i; \quad i = i + 1;$$

$$q = max\{.7\sqrt{DTOL/R_i},\ 1.1,\ (1 - T)/\Delta T\}$$

Set $\Delta T_i = q \Delta T_{i-1}$, and return to step 3.

(10) Save last sub-increment: $\Delta T_{last} = \Delta T_i$, record total number of successful sub-

increments as i, continue from step 1 for another load increment.

5.6.2 Automatic iterative scheme

In automatic incremental scheme, the effect of unbalance force at the end of a sub-increment is incorporated simply by adding the displacement due to that unbalance force with the current incremental displacement. By doing so the global equilibrium equation may not be satisfied exactly and there may always be a drifting of the solution from the exact solution. In automatic iterative scheme, the global equilibrium equation is satisfied by solving for the unbalance force iteratively during each sub-increment. Except this all other criteria in an automatic iterative scheme is same as that in an automatic incremental scheme. For example, selection of size and number of sub-increments, estimation of relative error, acceptance and rejection of a sub-increment, everything is done in the same way as in incremental scheme. In this scheme, to iterate the Euler solution, Newton-Raphson method is used as follows:

$$\boldsymbol{u}_i^k = \boldsymbol{u}_{i-1} + \Delta\boldsymbol{u}_1^k$$

where

$$\Delta\boldsymbol{u}_1^k = \Delta\boldsymbol{u}_1^{k-1} + \delta\boldsymbol{u}^k$$
$$\delta\boldsymbol{u}^k = [\boldsymbol{K}_t(\boldsymbol{u}_i^{k-1})]^{-1}\boldsymbol{R}(u_i^{k-1})$$

Here $\delta(\boldsymbol{u}^k)$ is the iterative displacement correction and k is the iteration number. To start the iteration, for $k = 0$, we can have $\Delta\boldsymbol{u}_1^0 = \Delta\boldsymbol{u}_1$, with $\Delta\boldsymbol{u}_1$ given by Equation (5.6.36). The constitutive laws are then integrated to find corresponding state of stress $\boldsymbol{\sigma}_i^k$ and checked for convergence. An iteration tolerance of $ITOL = DTOL/10$ is shown to be near optimal setting for minimizing the load path error by Abbo and Sloan [1]. After the iterations have converged, the pseudo time step and next load sub-increment are calculated in the same way as in the automatic incremental scheme. To find the relative error, the iterative displacement $\Delta\boldsymbol{u}_2^k$ is calculated as

$$\Delta\boldsymbol{u}_2^k = [\boldsymbol{K}_t(\boldsymbol{u}_i^k)]^{-1}\Delta\boldsymbol{Q}_i$$

If the relative error exceeds the the tolerance $DTOL$, the solution for the current

sub-increment is discarded and a smaller sub-step is calculated using Equations (5.6.39)-(5.6.41). Iterative displacements and error are again calculated and the process is repeated until a successful sub-step size is obtained (Sheng and Sloan [147]).

5.7 Summary

In this chapter some of the solution techniques and load stepping schemes are presented. They are implemented for MCCM in a new object-oriented finite element template, which is explained in Chapter 8. The automatic incremental scheme is excluded from the study as the results obtained with this scheme deviated considerably from the established one and kept for future study. The implicit integration algorithm presented in the Chapter 4 and the DIRK method explained in Chapter 6 are used to integrate the MCCM equations. In the various iterative and load stepping schemes, the path independent strategy is used for stresses and strains. The performances of these schemes are compared in some practical geomechanics problems. The numerical examples considered and the results obtained from different schemes are presented and discussed in the Chapter 9.

.

Chapter 6

Interpretation of nonlinear finite element analysis as DAEs and its application to MCCM

6.1 Introduction

Application of the variational principle to the equilibrium equations and their spatial discretization lead to a system of nonlinear algebraic equations. The solution of these resulting equations is generally computed on the global level. The evolution equations, which are ordinary differential equations (ODEs) of first order, are included in the solution by integrating them on element level, which is carried out by performing time integration at quadrature points of the numerical integration. Furthermore, the yield condition has to be incorporated in the solution which defines an algebraic equations occurring in case of elasto-plasticity ([40], [149]). The resulting nonlinear system of algebraic equations of the discretized variational principle has to be solved iteratively and consistent tangent operator has to be derived, which is necessary to achieve quadratic convergence in the context of Newtons method. However, it is only possible to derive consistent tangent operator effectively in case of simple implicit time-integration procedures, which are usually of low order [52].

The combination of the integration procedure with the discretization of variational principle and the notion of the consistent tangent operator are incoherent in most of the older proposals [52]. In this study we have addressed these issues in the context of

interpretation of the discretized form of the principle of virtual displacement together with the ODEs of the constitutive model defined locally at an integration point as a system of DAEs. This approach is extension of the work of Ellsiepen and Hartmaan [52] to MCCM equations. Treating nonlinear problem of Cam clay plasticity in this way facilitates to apply all the numerical algorithms of mathematics for the solution of algebraic differential equations. Numerical analysis of this approach gives the better proof for convergence and stability. This approach combines the time integration method of DAEs and a multi-level Newton-Raphson method to solve nonlinear system of algebraic equations. Consistent tangent operator can be better explained which provides the quadratic rate of convergence of multi-level Newton-Raphson method. The general case of diagonally implicit Runge-Kuttta (DIRK) method can be applied which is stiffly accurate and has better stability property. Also, adaptive time steps can be better implemented which is more efficient than backward Euler method. In the following sections the application of this approach of DAEs to MCCM equations is explained in brief. Description of the MCCM is given in Chapter 3.

6.2 Global algorithm

The equilibrium equation in any mathematical description of an elasto-plastic body of reference configuration $\Omega \subset \Re$, which is obtained from the balance of momentum is given by

$$div\ \boldsymbol{\sigma}(\boldsymbol{x},\ t) + \boldsymbol{\rho}(\boldsymbol{x})\ \boldsymbol{b}(\boldsymbol{x},\ t) = 0 \tag{6.2.1}$$

where $\boldsymbol{\sigma}$ denotes the Cauchy stress tensor of a particular material point $\boldsymbol{x} \in \Omega$ at time t, and $\boldsymbol{\rho}\,\boldsymbol{b}$ is the specific volume force of the same point. The Cauchy stress tensor $\boldsymbol{\sigma}$ for most of the constitutive models are defined by the elasticity relation as

$$\boldsymbol{\sigma} = f(\boldsymbol{\varepsilon},\ \boldsymbol{q}_m) \tag{6.2.2}$$

where $\boldsymbol{\varepsilon} = \frac{1}{2}(grad\ \boldsymbol{u} + grad^T\ \boldsymbol{u})$, is the linearized Green strain tensor depending on the displacement field $\boldsymbol{u}(\boldsymbol{x},\ t)$, and $\boldsymbol{q}_m = \boldsymbol{q}_m(\boldsymbol{x},\ t) = \{q_1,\ q_2, .., q_{n_q}\}^T$, are some internal variables. For any elasto-plastic material model, these variables can be expressed by local evolution equations for all times $t \in [t_0,\ t_n]$ in a more general form as follows:

$$\boldsymbol{A}\dot{\boldsymbol{q}}_m - \boldsymbol{r}(\boldsymbol{u},\ \boldsymbol{q}_m) = 0 \tag{6.2.3}$$

The dot denotes differentiation with respect to time t. The general constitutive Equation (6.2.3) vary for different material behaviour. For example, in case of rate-independent plasticity, which are based on a yield function, the matrix $\boldsymbol{A}$ is not invertible and $\boldsymbol{r}$ may have different switches. Now Equations (6.2.1)-(6.2.3) represent a system of initial-boundary-value problem with primary variable $\boldsymbol{u}$ and n_q number of internal variables $\boldsymbol{q}_m$. We need to investigate this system of equations in order to calculate these variables.

The variational formulation of the basic momentum balance Equation (6.2.1) is given by

$$\pi(\boldsymbol{u},\ \delta\boldsymbol{u},\ \boldsymbol{q}_m) = \int_\Omega grad\ \delta\boldsymbol{u}\ \boldsymbol{\sigma}\ d\Omega - \int_\Omega \delta\boldsymbol{u}\ \boldsymbol{\rho}\ \boldsymbol{b}\ d\Omega - \int_\Gamma \delta\boldsymbol{u}\ \boldsymbol{t}\ d\Gamma = 0 \qquad (6.2.4)$$

Here $\boldsymbol{t}$ is the external surface traction vector applied at the boundary Γ. By applying Galerkin's method in context to finite element, the unknown functions $\boldsymbol{u}$ and $\delta\boldsymbol{u}$ in Equation (6.2.4) are spatially discretized by shape function N_j with corresponding nodal degrees of freedom u_j and δu_j

$$\boldsymbol{u}^h(\boldsymbol{x},t) = \sum_{j=1}^{n_u} N_j(\boldsymbol{x})u_j(t) = \boldsymbol{N}(\boldsymbol{x})\boldsymbol{u}(t)\ ,\quad \delta\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{i=1}^{n_u} N_i(\boldsymbol{x})\delta u_i = \boldsymbol{N}(\boldsymbol{x})\delta\boldsymbol{u} \qquad (6.2.5)$$

Here $\boldsymbol{N}(\boldsymbol{x})$ is the global matrix of all shape functions and $\boldsymbol{u}(t)$ is the global vector of all nodal displacements. In Equation (6.2.5), the shape function $N_j(\boldsymbol{x})$ is a function of spatial variables $\boldsymbol{x}$, and nodal displacement $u_j(t)$ is a function of time only. Hence the finite element discretization is called as semi-discretization in space [52]. Now the spatially discrete strain and stress can be expressed as $\boldsymbol{\varepsilon}^h(\boldsymbol{x},\ t) = \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}(t)$ and $\boldsymbol{\sigma}^h(\boldsymbol{x},\ t) = f\left(\boldsymbol{\varepsilon}^h(\boldsymbol{x},\ t),\ \boldsymbol{q}^h(\boldsymbol{x},\ t)\right)$ respectively, where $\boldsymbol{B}(\boldsymbol{x})$ is the global strain-displacement matrix. Inserting the spatially discretized displacements into the evolution Equation (6.2.3), and collocating them at the Gaussian quadrature points $\boldsymbol{x}_k$ (numerical integration points), we arrive at a system of nonlinear ordinary differential equations of the spatially discretized internal variables $\boldsymbol{q}_{mk}(t) = \boldsymbol{q}_m^h(x_k,\ t)$

$$\boldsymbol{A}\dot{\boldsymbol{q}}_{mk}(t) - \boldsymbol{r}\left(\boldsymbol{u}^h(x_k,t),\ \boldsymbol{q}_{mk}(t)\right) = 0\ ,\qquad k = 1,2,3....n_i \qquad (6.2.6)$$

Here n_i is the total number of integration points in the structure. Now applying quadrature rule to the spatially discretized variational principle of momentum bal-

ance Equation (6.2.4) we get

$$\boldsymbol{g}\left(t, \boldsymbol{u}(t), \boldsymbol{q}_m(t)\right) \equiv \sum_{k=1}^{ni} \left[w_k \; \boldsymbol{B}^T(x_k) \; \boldsymbol{f}(\boldsymbol{B}(x_k)\boldsymbol{u}(t), \boldsymbol{q}_{mk}(t))\right] - \boldsymbol{T}(t) = 0 \qquad (6.2.7)$$

Here $\boldsymbol{T}(t)$ is the external nodal load vector explicitly depending on time, and w_k is the weighing factors. Equation (6.2.6) and Equation (6.2.7), which define a system of semi-explicit nonlinear differential algebraic equations of order first, can be written in the following form

$$\begin{aligned} \boldsymbol{F}_t(t,\ \boldsymbol{y}(t),\ \dot{\boldsymbol{y}}(t)) \equiv & \\ & \boldsymbol{g}\left(t,\ \boldsymbol{u}(t),\ \boldsymbol{q}_m(t)\right) = 0 \\ & \boldsymbol{A}\dot{\boldsymbol{q}}_m(t) - \boldsymbol{r}\left(\boldsymbol{u}(t),\ \boldsymbol{q}_m(t)\right) = 0 \end{aligned} \qquad (6.2.8)$$

where the function $\boldsymbol{y}(t)$ and its initial conditions at time t_0 are given by

$$\boldsymbol{y}(t) \ = \ \left\{ \begin{array}{l} \boldsymbol{u}(t) \\ \boldsymbol{q}_m(t) \end{array} \right\} \qquad (i)$$

(6.2.9)

$$\boldsymbol{y}(t_0) \equiv \ \left\{ \begin{array}{l} \boldsymbol{u}_0(t) = \boldsymbol{u}_0 \\ \boldsymbol{q}_m(t_0) = \boldsymbol{q}_{m0} \end{array} \right\} \ \equiv \boldsymbol{y}_0 \qquad (ii)$$

In Equation (6.2.8) and Equation (6.2.9), the index k for numerical integration points, is removed for brevity. Different methods of solving a system of DAE of type Equation (6.2.8) have been proposed. One of such example is stiffly accurate diagonally implicit Runge-Kutta method. If this method is applied to solve Equation (6.2.8), we can apply higher ordered methods and very efficient time-adaptive procedure [52], [64].

6.2.1 DIRK method

Consider an ordinary differential equation

$$\dot{\boldsymbol{y}}(t) = f_t(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \qquad \boldsymbol{y} \in \Re, \qquad t \in [t_0, T] \qquad (6.2.10)$$

Let the time step size be Δt_n such that $\Delta t_n = t_{n+1} - t_n$ and the solution of $\boldsymbol{y}$ at time $t_n = t_0 + n\Delta t_n$ be approximated by $\boldsymbol{y}_n$. The step size Δt_n is further divided into number of sub-steps called as stages as shown in figure 6.2.1. These points in a time step corresponding to each stage are defined as $T_{ni} = t_n + c_i \Delta t_n,\ i = 1, .., s$, with $T_{ns} = t_{n+1}$ and $0 \leq ci \leq 1$. We assume that the exact solution at time t_n is known and we are looking for the solution at t_{n+1}. According to an s-stage implicit Runge-Kutta method, this is defined as

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \dot{\boldsymbol{Y}}_{ni} \tag{6.2.11}$$

with internal stages

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{s} a_{ij} \dot{\boldsymbol{Y}}_{nj} \qquad i = 1, ..., s. \tag{6.2.12}$$

where the expression $\dot{\boldsymbol{Y}}_{nj}$ are called stage derivatives representing the discrete right-hand side of the ODE (6.2.10) under study and are defined as

$$\dot{\boldsymbol{Y}}_{ni} = f_{ni}(T_{ni}, \boldsymbol{Y}_{ni}),\ i = 1, ..., s.$$

The weighing factors a_{ij} and b_j and the values c_i are summarized in the Butcher array to distinguish different classes of implicit Runge-Kutta methods, as shown in figure 6.2.1.

In case of a stiffly accurate Runge-Kutta method, $a_{si} = b_i$ and hence solution $\boldsymbol{Y}_{ns}$ on the last stage coincides with the the new solution $\boldsymbol{y}_{n+1}$. This property guarantees that the algebraic constraints are fulfilled at the new time step t_{n+1}, which is preferable in the DAE context. In case of DIRK method, $a_{ij} = 0$ for $i < j$ and this nullifies the coupling of the stages so that the sparse structure of the linearized finite element system is not destroyed. Therefore, DIRK method is applied.

In the case of DIRK method, the integration step (6.2.12) reduces to

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i} a_{ij} \dot{\boldsymbol{Y}}_{nj} \ = \ {}^{s}\boldsymbol{Y}_{ni} + \Delta t_n a_{ii} \dot{\boldsymbol{Y}}_{ni} \tag{6.2.13}$$

with the starting value

$${}^{s}\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{nj} \tag{6.2.14}$$

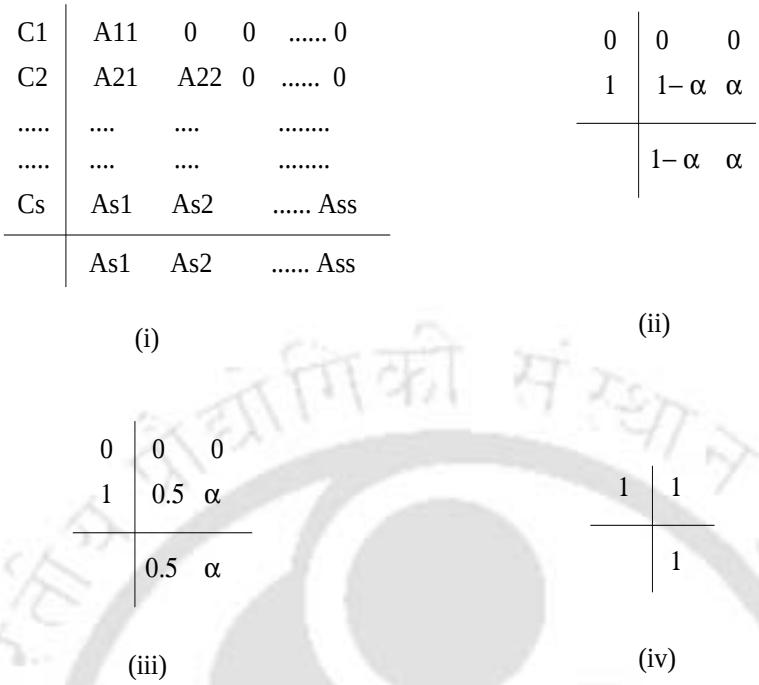


Figure 6.1: Butcher arrays for (i) DIRK method (ii) Generalized trapezoidal rule (α method) (iii) Backward Euler method (iv) Trapezoidal rule

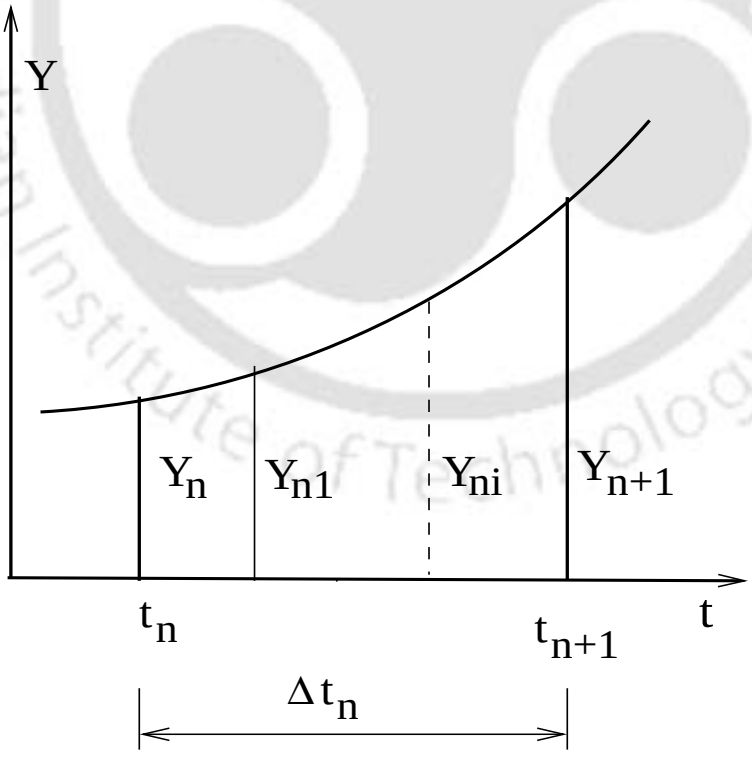


Figure 6.2: Time-step and stages in DIRK method

where ${}^{s}\boldsymbol{Y}_{n1} = \boldsymbol{y}_n$. The starting value ${}^{s}\boldsymbol{Y}_{ni}$ at stage i depends only on the stage derivatives which are already calculated at the previous stages $i = 1, ...i-1$. From Equation (6.2.13), the stage derivative at stage i is given as

$$\dot{\boldsymbol{Y}}_{ni} = \frac{\boldsymbol{Y}_{ni} - {}^{s}\boldsymbol{Y}_{ni}}{\Delta t_n a_{ii}} \tag{6.2.15}$$

The DAEs (6.2.8) resulting from the finite element discretization now can be written in terms of the stage quantities T_{ni} and $\boldsymbol{Y}_{ni}$ as

$$\begin{aligned} \boldsymbol{R}_{ni}(\boldsymbol{Y}_{ni}) = \boldsymbol{F}_{ni}(\boldsymbol{T}_{ni},\ \boldsymbol{Y}_{ni}\ \dot{\boldsymbol{Y}}_{ni}) = \boldsymbol{F}_{ni}(\boldsymbol{T}_{ni},\ \boldsymbol{Y}_{ni},\ \frac{\boldsymbol{Y}_{ni} - {}^{s}\boldsymbol{Y}_{ni}}{\Delta t_n a_{ii}}) = \\ \boldsymbol{G}_{ni}(\boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) = 0 \\ \boldsymbol{L}_{ni}(\boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) = 0 \end{aligned} \tag{6.2.16}$$

where $\boldsymbol{G}_{ni}$ represents the global system of equations which results from inserting stage quantities into the spatially discretized weak formulation (6.2.7) and $\boldsymbol{L}_{ni}$ represents the nonlinear system of equation resulting from inserting the stage quantities into the spatially discretized form of local evolution Equation (6.2.6) and can be written as

$$\boldsymbol{G}_{ni}(\boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) \equiv \boldsymbol{g}(\boldsymbol{T}_{ni},\ \boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) \quad \text{and} \tag{6.2.17}$$

$$\boldsymbol{L}_{ni}(\boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) \equiv (\frac{\boldsymbol{Q}_{ni} - {}^{S}\boldsymbol{Q}_{ni}}{\Delta t_n a_{ii}}) - \boldsymbol{r}(\boldsymbol{U}_{ni},\ \boldsymbol{Q}_{ni}) \tag{6.2.18}$$

In Equation (6.2.16) we have only one set of unknowns, namely the stage solutions $\boldsymbol{Y}_{ni}$, and thus we have obtained a decoupling of the stage variable calculation. So, in each time-step we have to consecutively solve s nonlinear system of equations for the vector $\boldsymbol{Y}_{ni}$ and due to stiff accuracy, we have $\boldsymbol{y}_{n+1} = \boldsymbol{Y}_{ns}$. The whole solution procedure is shown in Table 6.1.

6.2.2 Solution of coupled system of equations using multi-level Newton-Raphson method

Different solution schemes have been developed for solving system of nonlinear equations, such as in Equation (6.2.16), and Newton-Raphson method is the one mostly used so far. In this method the notion of the consistent tangent operator has been proved by purely numerical considerations. Again, by applying this method to solve the system of Equation (6.2.16), the advantage of band-structured sparse tangential

Table 6.1: Global step-size controlled embedded DIRK method [52]

Given

- Coefficients $c_i,\ a_{ij},\ b_j,\ \ (i,\ j = 1, ...s)$ of a DIRK method
- Initial time t_0, initial step-size Δt_0
- Initial conditions:

$$\boldsymbol{y}(t_0) \equiv \left\{ \begin{array}{c} \boldsymbol{u}(t_0) \\ \boldsymbol{q}_m(t_0) \end{array} \right\} \equiv \left\{ \begin{array}{c} \boldsymbol{u}_0 \\ \boldsymbol{q}_{m0} \end{array} \right\} \equiv \boldsymbol{y}_0$$

- n=0

Loop over time-steps: $n = 0, ...N$

Repeat

Loop over stages: $i = 1, ...s$

$$T_{ni} = t_n + c_i \Delta t_n$$

$${}^s\boldsymbol{Y}_{ni} = \boldsymbol{Y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{nj}$$

Solve: $F_{ni}(T_{ni},\ \boldsymbol{Y}_{ni},\ \dfrac{\boldsymbol{Y}_{ni} - {}^s\boldsymbol{Y}_{ni}}{\Delta t_n a_{ii}}) = 0$ for $\boldsymbol{Y}_{ni}$

Store $\dot{\boldsymbol{Y}}_{ni} = \dfrac{\boldsymbol{Y}_{ni} - {}^s\boldsymbol{Y}_{ni}}{\Delta t_n a_{ii}}$

Estimate error for time-step
Compute new time-step size Δt_{new} if required
If step is not accepted set $\Delta t_n = \Delta t_{new}$

Until step is accepted
Update:

$$\Delta t_{n+1} = \Delta t_n,\ \Delta t_{n+1} = \Delta t_{new},\ \boldsymbol{y}_{n+1} = \boldsymbol{Y}_{ns}$$

stiffness matrix is lost. To overcome these, multi-level Newton-Raphson method can be applied, which is based on the implicit function theorem.

To apply this method, let us rewrite the Equation (6.2.16) in the following form by removing the indices symbolizing time steps or stages

$$\boldsymbol{G}(\boldsymbol{U},\ \boldsymbol{Q}) = 0 \tag{6.2.19}$$

$$\boldsymbol{L}(\boldsymbol{U},\ \boldsymbol{Q}) = 0 \tag{6.2.20}$$

By applying implicit function theorem to Equation (6.2.19) we solve for the function $\boldsymbol{Q}(\boldsymbol{U})$ near the solution of $\boldsymbol{L}$ whenever it is sufficiently smooth. If we insert this function $\boldsymbol{Q}(\boldsymbol{U})$ into Equation (6.2.20), we get an equation, which solely depends on

the vector $\boldsymbol{U}$

$$\boldsymbol{G}(\boldsymbol{U},\ \boldsymbol{Q}(\boldsymbol{U})) = 0 \tag{6.2.21}$$

We need to solve this Equation (6.2.21) for the vector $\boldsymbol{U}$ iteratively. For this we can apply classical Newton-Raphson method, which converts Equation (6.2.21) to a system of linear equations as

$$\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{Q}}\frac{d\boldsymbol{Q}}{d\boldsymbol{U}}\right]^{(k)} \Delta \boldsymbol{U} = -\boldsymbol{G}(\boldsymbol{U}^{(k)}, \boldsymbol{Q}^{(k)}) \tag{6.2.22}$$

where the index k symbolizes iteration number. Equation (6.2.22) is to be solved for $\Delta \boldsymbol{U} = \boldsymbol{U}^{k+1} - \boldsymbol{U}^{k}$, which is the increment of the quantity $\boldsymbol{U}$. For a given value of $\boldsymbol{U}^{(k)}$, we can determine $\boldsymbol{Q}^{(k)} = \boldsymbol{Q}(\boldsymbol{U}^{(k)})$ on element level at each integration point by solving the equation

$$\boldsymbol{L}\left(\boldsymbol{U}^{(k)}, \boldsymbol{Q}^{(k)}\right) = 0 \tag{6.2.23}$$

The solution of Equation (6.2.23) for a given value of $\boldsymbol{U}^{(k)}$ represents the time integration of the internal variables in order to obtain stresses and hence called as "stress integration". The coefficient matrix of $\Delta \boldsymbol{U}$ in Equation (6.2.22) is called as the tangential stiffness matrix. This functional matrix depends on the partial derivative of the function $\boldsymbol{G}$ with respect to $\boldsymbol{U}$ as well as the derivative of the unknown function $\boldsymbol{Q}(\boldsymbol{U})$ with respect to $\boldsymbol{U}$. This derivative can be determined by applying chain rule of differentiation to the function $\boldsymbol{L}(\boldsymbol{U},\ \boldsymbol{Q}(\boldsymbol{U})) = 0$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{Q}}\frac{d\boldsymbol{Q}}{d\boldsymbol{U}} = 0 \ \Rightarrow\ \left[\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{Q}}\right]\frac{d\boldsymbol{Q}}{d\boldsymbol{U}} = -\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{U}} \tag{6.2.24}$$

The linear system of Equation (6.2.24) is solved for $\dfrac{d\boldsymbol{Q}}{d\boldsymbol{U}}$ for given values of $\boldsymbol{U}^{(k)}$ and $\boldsymbol{Q}^{(k)}$ on element level at each integration point. The solution procedure explained above is shown step-wise in Table 6.2.

The tangential stiffness matrix in Equation (6.2.22) can be derived at local level at each integration point from the differentiation of Equation (6.2.7) with $\boldsymbol{\sigma} = \boldsymbol{f}(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^{p})$, i.e., $\boldsymbol{q}_m = \boldsymbol{\varepsilon}^{p}$ as follows: (for brevity the indices symbolizing the time-step and space-discretization are removed)

$$\begin{aligned}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{Q}}\frac{d\boldsymbol{Q}}{d\boldsymbol{U}} &= \sum_{k=1}^{m} w_k \left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\sigma}}\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\sigma}}\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{q}_m}\frac{d\boldsymbol{q}_m}{d\boldsymbol{\varepsilon}}\frac{\partial \boldsymbol{\varepsilon}}{\partial \boldsymbol{U}}\right]_{x=x_k} \\ &= \sum_{k=1}^{m} w_k \left[\boldsymbol{B}^{T}\left[\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} + \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{q}_m}\frac{d\boldsymbol{q}_m}{d\boldsymbol{\varepsilon}}\right]\boldsymbol{B}\right]_{x=x_k}\end{aligned} \tag{6.2.25}$$

Table 6.2: Multi-level Newton-Raphson iteration at stage i of time-step $n \rightarrow n+1$ [52]

	Iteration step (k) of global Newton-Raphson iteration	
1	Local Newton-Raphson iteration (given: fixed $\boldsymbol{U}_{ni}^{(k)}$): Solve iteratively $\boldsymbol{L}(\boldsymbol{U}_{ni}^{(k)}, \boldsymbol{Q}_{ni}^{(k)}) = 0$	$\rightarrow \boldsymbol{Q}_{ni}^{(k)}$
2	Consistent linearization with argument vector $z = \left(\boldsymbol{U}_{ni}^{(k)}, \boldsymbol{Q}_{ni}^{(k)}\right)$: $\left[\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{Q}}\right]_z \frac{d\boldsymbol{Q}}{d\boldsymbol{U}} = \left[\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{U}}\right]_z$	$\rightarrow \left[\frac{d\boldsymbol{Q}}{d\boldsymbol{U}}\right]_z$
3	Global Newton-Raphson step: $\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{U}} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{Q}}\frac{d\boldsymbol{Q}}{d\boldsymbol{U}}\right]_z \Delta \boldsymbol{U} = -\boldsymbol{G}(z)$	$\rightarrow \Delta \boldsymbol{U}_{ni}$
4	Update global variables: $\boldsymbol{U}_{ni}^{(k+1)} \leftarrow \boldsymbol{U}_{ni}^{(k)} + \Delta \boldsymbol{U}_{ni}$	$\rightarrow \boldsymbol{U}_{ni}^{(k+1)}$
5	Convergence Check Find residual and/or update criterion with relative/absolute tolerance(s)	

In Equation (6.2.25), $\boldsymbol{\sigma} = f(\boldsymbol{\varepsilon}, \boldsymbol{q}_m)$ and $\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}}$ represents the elasticity matrix, which can be determined by applying the implicit function theorem only.

6.3 Application to MCCM

In the following, we apply the integration steps depicted in Table 6.2 to the MCCM equations. The structure of this model is given in Table 3.1. As this model exhibits hypo-elasticity, we need to integrate Equation (4.2.1) in the elastic region too. Now we have to integrate the stress-strain relation (4.2.1) along with the evolution Equation (4.2.3) and Equation (4.2.4), which is called as stress integration algorithm. According to the multi-level Newton-Raphson method, the displacements $\boldsymbol{U}_{ni}^{(k)}$ of the current stage ni are known and thus, the total strain $\boldsymbol{\varepsilon}_i^{(k)}$ at the integration points are also known. For brevity we write the incremental stress-strain relation (4.2.1) in the following form

$$\dot{\boldsymbol{\sigma}} = K\hat{\boldsymbol{C}}^e : \dot{\boldsymbol{\varepsilon}}^e \tag{6.3.26}$$

where

$$\hat{\boldsymbol{C}}^e = \boldsymbol{l} \otimes \boldsymbol{l} + 2r(\boldsymbol{I} - \frac{1}{3}\boldsymbol{l} \otimes \boldsymbol{l}) \quad \text{and} \quad r = \frac{3(1-2\mu)}{2(1+\mu)} \tag{6.3.27}$$

In Equation (6.3.27) the Poissons ratio μ and the tensor $\hat{\boldsymbol{C}}^e$ is constant. In MCCM, the elastic moduli K and hence G are not constant, and K is a function of the effective volumetric stress p via the relation.

$$K = \frac{\partial p}{\partial \varepsilon_v^e} = (\frac{v}{\kappa})p \tag{6.3.28}$$

The stage value of K, i.e., K_{ni} is obtained by integrating Equation (6.3.28) for p and ε_v^e, which represents an 'average' bulk stiffness of soil in time interval $(t_n - T_{ni})$

$$K_{ni} = \frac{p_n}{\Delta\varepsilon_{vni}^e}[exp[\frac{v}{\kappa}(\varepsilon_{vni}^e - \varepsilon_{vn}^e)] - 1] \tag{6.3.29}$$

Now integrating Equation (6.3.26) and Equation (4.2.3) using DIRK method with their known starting values as ${}^S(*)_{ni}$, we arrive at the following equations

$$\boldsymbol{\sigma}_{ni} = {}^S\boldsymbol{\sigma}_{ni} + \Delta t_n a_{ii}[K_{ni}\hat{\boldsymbol{C}}^e : (\Delta\boldsymbol{\varepsilon}_{ni} - \Delta\boldsymbol{\varepsilon}_{ni}^p)] \tag{6.3.30}$$

$$\boldsymbol{\varepsilon}_{ni}^p = {}^S\boldsymbol{\varepsilon}_{ni}^p + \Delta t_n a_{ii}\phi_{ni}\boldsymbol{N}_{ni} \quad \text{where} \quad \boldsymbol{N}_{ni} = \frac{\partial F}{\partial \boldsymbol{\sigma}} \tag{6.3.31}$$

And from analytical integration of Equation (4.2.4) of $\dot{p}_c$ from ${}^S p_{cni}$ to p_{cni}, we get

$$p_{cni} = {}^S p_{cni}\ exp\ [\frac{v}{(\lambda - \kappa)}\ (\boldsymbol{\varepsilon}_{vni}^p - {}^S\boldsymbol{\varepsilon}_{vni}^p)] \tag{6.3.32}$$

where $\Delta(*)_{ni} = (*)_{ni} - (*)_n$, and the starting values are given by the Equation (6.2.14). Now let us consider a trial value of stress, called as elastic predictor

$${}^T\boldsymbol{\sigma}_{ni} = {}^S\boldsymbol{\sigma}_{ni} + \Delta t_n a_{ii}[K_{ni}\hat{\boldsymbol{C}}^e : (\Delta\boldsymbol{\varepsilon}_{ni} - \Delta\,{}^S\boldsymbol{\varepsilon}_{ni}^p)] \tag{6.3.33}$$

with $\Delta\,{}^S\boldsymbol{\varepsilon}_{ni}^p = {}^S\boldsymbol{\varepsilon}_{ni}^p - \boldsymbol{\varepsilon}_n^p$. From ${}^T\boldsymbol{\sigma}_{ni}$ we can get ${}^T p_{ni}$ and ${}^T q_{ni}$. In Equation (6.3.33) all quantities are known where K_{ni} is to be calculated from Equation (6.3.29), using

$$\Delta\varepsilon_{vni}^e = \Delta\ {}^S\varepsilon_{vni}^e = {}^S\varepsilon_{vni}^e - \varepsilon_{vn}^e$$

Now using this trial value of stresses we need to check for the condition of yielding. If $F({}^T p_{ni},\ {}^T q_{ni},\ {}^S p_{cni}) \leq 0$ holds, the stress state lies inside the yield surface. Then the new stresses $\sigma_{ni} = {}^T\sigma_{ni}$, plastic strains $\varepsilon_{ni}^p = {}^S\varepsilon_{ni}^p$, and pre-consolidation pressure

$p'_{cni} = {}^{S}p_{cni}$ are known. Otherwise if $F({}^{T}p_{ni},\ {}^{T}q_{ni},\ {}^{S}p_{cni}) > 0$, we have to solve Equations (6.3.30), (6.3.31) and (6.3.32) for $\boldsymbol{\sigma}_{ni}$, $\boldsymbol{\varepsilon}_{ni}$ and p_{cni} with the constraint equation

$$F(p_{ni},\ q_{ni},\ p_{cni}) = 0 \tag{6.3.34}$$

In this case we rewrite the trial stress-strain relation at stage i in the following form

$${}^{T}\boldsymbol{\sigma}_{ni} = K_{ni}\ {}^{S}\varepsilon^{e}_{vni}\ \boldsymbol{l} + 2G_{ni}\ {}^{S}\varepsilon^{e}_{dni} = {}^{S}p_{ni}\ \boldsymbol{l} + {}^{S}\boldsymbol{S}_{ni} \tag{6.3.35}$$

There are some special procedures which fall in the class of DIRK methods. Examples are backwards Euler method, implicit midpoint rule and trapezoidal rule. Out of these, backward Euler method is the simplest DIRK method.

In this study the generalized trapezoidal rule, which is also called as α method, is implemented, with two special cases. They are
(i) Backward Euler method and
(ii) Trapezoidal rule

6.3.1 Application of α method

To incorporate generalized trapezoidal rule or α method in the general DIRK method, a little modification is required. In case of α method, the upper left elements of the Butcher array are 0, which is not allowed in the usual DIRK notation used before. To include this method, modification is required in the Butcher array. An extra row with index $j = 0$ is added to the matrix $\boldsymbol{a}_{ij}$, with additional stage $i = 0$ (with $a_{00} = 0$). This represents the previous time-step, such that $T_{n0} = t_n$, $c_0 = 0$, $\boldsymbol{Y}_{n0} = \boldsymbol{y}_n$, ${}^{s}\boldsymbol{Y}_{n0} = \boldsymbol{y}_n$, $\dot{\boldsymbol{Y}}_{n0} = \dot{\boldsymbol{y}}_n$. Therefore, no additional computation is required for this extra stage. Only the sum in the equation of ${}^{S}\boldsymbol{Y}_{ni}$ will start from $j = 0$. Again in DIRK method, $\boldsymbol{Y}_{ns} = \boldsymbol{y}_{n+1}$ and $\dot{\boldsymbol{Y}}_{ns} = \dot{\boldsymbol{y}}_{n+1}$. Therefore we have

$$\dot{\boldsymbol{y}}_n = \dot{\boldsymbol{Y}}_{(n-1)s} \tag{6.3.36}$$

In the first stage with i=0, all the unknowns are previously calculated at the last converged time-stage Δt_n. So the nonlinear equations are always fulfilled in the first

step. In the second step (stage i=1), we have to calculate $\boldsymbol{Y}_{n1} = \boldsymbol{y}_{n+1}$ at $T_{n1} = t_{n+1}$, according to the equation

$$F(t_{n+1},\ \boldsymbol{y}_{n+1},\ \frac{\boldsymbol{y}_{n+1} - {}^{S}\boldsymbol{Y}_{n1}}{\alpha\ \Delta t_n}) = 0, \qquad \alpha > 0 \tag{6.3.37}$$

with $\boldsymbol{y}_{n+1}$ and ${}^{S}\boldsymbol{y}_{n1}$ given by DIRK method as

$$\boldsymbol{y}_{n+1} = {}^{S}\boldsymbol{Y}_{n1} + \Delta t_n \alpha \dot{\boldsymbol{Y}}_{n1} \tag{6.3.38}$$

$${}^{S}\boldsymbol{Y}_{n1} = \boldsymbol{y}_n + (1-\alpha)\Delta t_n \dot{\boldsymbol{Y}}_{n0} \tag{6.3.39}$$

So we have to evaluate only one nonlinear system of Equation (6.3.37)

6.3.1.1 Integration of nonlinear elastic constitutive equation

For the MCCM, the equation, which needs to be integrated in the elastic region is

$$\dot{\boldsymbol{\sigma}} = K\hat{\boldsymbol{C}}^{e} : \dot{\boldsymbol{\varepsilon}}^{e} \tag{6.3.40}$$

The elastic moduli K is not constant, and is a function of p through $K = (v/\kappa)p$. By integrating Equation (6.3.40) in time interval $(t_n - T_{n1})$

$$\boldsymbol{\sigma}_{n1} = \boldsymbol{\sigma}_n + K_{n1}\ \hat{\boldsymbol{C}}^{e} : \Delta\boldsymbol{\varepsilon}^{e}_{n1} \tag{6.3.41}$$

where

$$K_{n1} = \frac{p_n}{\Delta\varepsilon^{e}_{vn1}}[exp(\frac{v}{\kappa}\Delta\varepsilon^{e}_{vn1}) - 1]\ , \qquad \Delta\boldsymbol{\varepsilon}^{e}_{n1} = \boldsymbol{\varepsilon}^{e}_{n1} - \Delta\boldsymbol{\varepsilon}^{e}_{n} \tag{6.3.42}$$

6.3.1.2 Integration of nonlinear elasto-plastic constitutive equation

In the elasto-plastic region, along with Equation (6.3.40), we need to integrate

$$\dot{\boldsymbol{\epsilon}}^{p} = \phi(\frac{\partial f(\boldsymbol{\sigma}, k)}{\partial \boldsymbol{\sigma}}) \tag{6.3.43}$$

$$\dot{k} = \dot{p}_c = \frac{v}{\lambda - \kappa}(p_c \dot{\epsilon}^{p}_{v}) \tag{6.3.44}$$

For this purpose we write the stress-strain relation in terms of volumetric and deviatoric stresses, in the following form

$$\boldsymbol{\sigma}_{n1} = p_{n1}\boldsymbol{l} + \boldsymbol{q}_{n1} \tag{6.3.45}$$

where

$$p_{n1} = K_{n1}\varepsilon^e_{vn1} \tag{6.3.46}$$

$$q_{n1} = \sqrt{\frac{3}{2}}\|\boldsymbol{S}_{n1} : \boldsymbol{S}_{n1}\| \tag{6.3.47}$$

$$\boldsymbol{S}_{n1} = 2G_{n1}\boldsymbol{\varepsilon}^e_{dn1} \tag{6.3.48}$$

Equation (6.3.43) is integrated using Equation (6.3.38) as

$$\boldsymbol{\varepsilon}^p_{n1} = {}^S\boldsymbol{\varepsilon}^p_{n1} + \Delta t_n \phi_{n1} \alpha \frac{\partial F}{\partial \boldsymbol{\sigma}_{n1}} \tag{6.3.49}$$

with starting values as

$${}^S\boldsymbol{\varepsilon}_{n1} = \boldsymbol{\varepsilon}_n + \Delta t_n (1-\alpha)\dot{\boldsymbol{\varepsilon}}_{n0} \tag{6.3.50}$$

where $\dot{\boldsymbol{\varepsilon}}_{n0} = \dot{\boldsymbol{\varepsilon}}_n$ and $\Delta\boldsymbol{\varepsilon}_{ni} = \boldsymbol{\varepsilon}_{ni} - \boldsymbol{\varepsilon}_n$.

Using ($\varepsilon^e_{vn1} = \varepsilon_{vn1} - \varepsilon^p_{vn1}$) and Equation (6.3.49) for volumetric part in Equation (6.3.46) we get

$$\begin{aligned} p_{n1} &= K_{n1}\left[\varepsilon_{vn1} - {}^S\varepsilon^p_{vn1} - \Delta t_n \phi_{n1}\alpha \frac{\partial F}{\partial p_{n1}}\right] \\ &= {}^S p_{n1} - K_{n1}\Delta t_n \phi_{n1}\alpha(2p_{n1} - p_{cn1}) \\ \Rightarrow p_{n1} &= \frac{{}^S p_{n1} + K_{n1}\Delta t_n \phi_{n1}\alpha p_{cn1}}{1 + 2K_{n1}\Delta t_n \phi_{n1}\alpha} \end{aligned} \tag{6.3.51}$$

Similarly, using ($\boldsymbol{\varepsilon}^e_{dn1} = \boldsymbol{\varepsilon}_{dn1} - \boldsymbol{\varepsilon}^p_{dn1}$) and Equation (6.3.49) for deviatoric part in Equation (6.3.48) we get

$$\begin{aligned} \boldsymbol{S}_{n1} &= 2G_{n1}\left[\boldsymbol{\varepsilon}_{dn1} - {}^S\boldsymbol{\varepsilon}^p_{dn1} - \Delta t_n \phi_{n1}\alpha \frac{\partial F}{\partial \boldsymbol{S}_{n1}}\right] \\ &= {}^S\boldsymbol{S}_{n1} - 2G_{n1}\Delta t_n \phi_{n1}\alpha \frac{\partial F}{\partial \boldsymbol{S}_{n1}} \\ &= {}^S\boldsymbol{S}_{n1} - 2G_{n1}\Delta t_n \phi_{n1}\alpha \frac{3}{M^2}\boldsymbol{S}_{n1} \end{aligned} \tag{6.3.52}$$

Now using Equation (6.3.52) in Equation (6.3.47) we get

$$q_{n1} = \sqrt{\frac{3}{2}}\| \ {}^{S}\boldsymbol{S}_{n1} : \ {}^{S}\boldsymbol{S}_{n1}\| - \sqrt{\frac{3}{2}}\ 2G_{n1}\Delta t_n\phi_{n1}\alpha\frac{3}{M^2}\| \quad \boldsymbol{S}_{n1} : \boldsymbol{S}_{n1}\|$$

$$= \ {}^{S}q_{n1} - 6\frac{G_{n1}}{M^2}\Delta t_n\phi_{n1}\alpha q_{n1}$$

$$\Rightarrow q_{n1} = \frac{{}^{S}q_{n1}}{(1+6G_{n1}\Delta t_n\phi_{n1}\alpha/M^2)} \tag{6.3.53}$$

Now to get p_{cn1}, we integrate Equation (6.3.44) analytically from p_{cn} to p_{cn1} as

$$\int \frac{\dot{p_c}}{p_c} = \vartheta \int \dot{\varepsilon}_v^p$$

$$\Rightarrow p_{cn1} = p_{cn}\ exp\left[\vartheta\left(\varepsilon_{vn1}^p - \varepsilon_{vn}^p\right)\right]$$

Using Equation (6.3.49) for volumetric part

$$p_{cn1} = p_{cn}\ exp\left[\vartheta\left\{\left(\ {}^{S}\varepsilon_{vn1}^p + \Delta t_n\phi_{n1}\alpha\frac{\partial F}{\partial p_{n1}}\right) - \varepsilon_{vn}^p\right\}\right]$$

$$= p_{cn}\ exp\left[\vartheta\left(\ {}^{S}\varepsilon_{vn1}^p - \varepsilon_{vn}^p\right)\right]\ exp\left[\vartheta\Delta t_n\phi_{n1}\alpha(2p_{n1} - p_{cn1})\right]$$

$$\Rightarrow p_{cn1} = \ {}^{S}p_{cn1}\ exp\left[\vartheta\Delta t_n\phi_{n1}\alpha(2p_{n1} - p_{cn1})\right] \tag{6.3.54}$$

Using Equation(6.3.51) in Equation (6.3.54), we get

$$p_{cn1} = \ {}^{S}p_{cn1}\ exp\left[\vartheta\Delta t_n\phi_{n1}\alpha\frac{2\ {}^{S}p_{n1} - p_{cn1}}{1+2K_{n1}\Delta t_n\phi_{n1}\alpha}\right]$$

$$\Rightarrow R(p_{cn1}) = \ {}^{S}p_{cn1}\ exp\left[\vartheta\Delta t_n\phi_{n1}\alpha\frac{2\ {}^{S}p_{n1} - p_{cn1}}{1+2K_{n1}\Delta t_n\phi_{n1}\alpha}\right] - p_{cn1} = 0 \tag{6.3.55}$$

Equation (6.3.55) is solved iteratively for a known value of $\xi = \Delta t_n\phi_{n1}$ to get p_{cn1}. To start the iteration, initially we assume $\xi = 0$ and $K_{n1} = K_n$. Knowing these, we can calculate p_{n1} and $\boldsymbol{q}_{n1}$ from equation (6.3.51) and equation (6.3.53). Next we check for consistency requirement of $F(\boldsymbol{\sigma}_{ni},\ p_{cni}) = 0$. A multi-level Newton-Raphson method can be used to solve these equations simultaneously.

6.3.2 Backward Euler method

The MCCM equations are already integrated using backward Euler method and resulting equations are shown before. In case of α method, it become identical to

the backward Euler method for $\alpha = 1$. In the above equations, if we put $\alpha = 1$, we get ${}^{S}\boldsymbol{\varepsilon}_{n1} = \boldsymbol{\varepsilon}_n$.Using this and putting $\alpha = 1$ in the equations of ${}^{S}p_{n1}$, ${}^{S}\boldsymbol{q}_{n1}$, and ${}^{S}p_{cn1}$, we get

$$ {}^{S}p_{n1} = p^{tr}_{n+1} \qquad {}^{S}q_{n1} = q^{tr}_{n+1} \quad \text{and} \quad {}^{S}p_{cn1} = p_{cn} \tag{6.3.56} $$

Using these in Equations (6.3.51, 6.3.53, and 6.3.54), we get

$$ p_{n+1}(\xi) = p^{tr}_{n+1} - K_{n+\alpha}\xi(2p - p_c)_{n+1} \tag{6.3.57} $$

$$ q_{n+1}(\xi) = q^{tr}_{n+1}(1 + 6\, G_{n+\alpha}\xi/M^2)^{-1} \tag{6.3.58} $$

$$ p_{cn+1}(\xi) = p_{cn}\ exp[\vartheta\xi(2p - p_c)_{n+1}] \tag{6.3.59} $$

$$ F_{n+1}(\xi) = [\frac{q^2}{M^2} + p(p - p_c)]_{n+1} \tag{6.3.60} $$

The Equations (6.3.57-6.3.60) are similar to the Equations (4.2.32-4.2.35), so they are solved iteratively in the similar manner.

The consistent tangent operator is same as the Equation (4.2.42), except ϕ is replaced with ξ here.

6.3.3 Trapezoidal rule

In case of α method, if we put $\alpha = 0.5$ we arrive at the trapezoidal rule. Using $\alpha = 0.5$, and $\Delta t_n \phi_{n1} = \xi$, the resulting equations for trapezoidal rule are

$$ \Rightarrow p_{n1} = \frac{{}^{S}p_{n1} + 0.5K_{n1}\xi p_{cn1}}{1 + K_{n1}\xi} \tag{6.3.61} $$

$$ \Rightarrow q_{n1} = \frac{{}^{S}q_{n1}}{(1 + 3G_{n1}\xi/M^2)} \tag{6.3.62} $$

$$ \Rightarrow p_{cn1} = {}^{S}p_{cn1}\ exp\,[0.5\vartheta\xi(2p_{n1} - p_{cn1})] \tag{6.3.63} $$

Equations (6.3.61, 6.3.62, 6.3.63) and the constraint equation $F(\boldsymbol{\sigma}_{ni},\ p_{cni}) = 0$ are solved simultaneously using multi-level Newton-Raphson method.

6.4 Summary

The concept of interpretation of the discretization form of the principle of virtual displacement together with the ODEs of the constitutive model as a system of

DAE is implemented to integrate the MCCM equations. This treatment allows the application of all the numerical algorithm of mathematics for the solution of nonlinear constitutive equations. In this study, the DIRK method is employed to integrate the MCCM equations. In the algorithm, α method with two special cases, namely (i) backward Euler method and (ii) trapezoidal rule, are incorporated. The algorithms are implemented in an object-oriented finite element framework.

.

Chapter 7

Object-oriented finite element: fundamental design

7.1 Introduction

The finite element method has gained wide acceptance for analyzing geomechanics problems recently. It has been shown that the finite element method (FEM) can provide realistic results for practical engineering problems, when properly used. In FEM software, there is always increasing demand on new analytic capabilities, constitutive models, solving strategies and easy-to-handle graphical user interfaces. But most of the available software have been written in FORTRAN or C language consisting several hundred and thousand lines of procedural code, which contains many complex data structures and which are accessed throughout the program. As such, modification or extension of the portion of the code requires a high degree of knowledge of the entire program. Thus the resulting code is inflexible and presents a barrier to practicing engineers and researchers for new uses, models and solution procedures.

Therefore, a new design approach is needed, where the code becomes flexible, reusable and extendable with minimal effort. A finite element framework which should provide extendability, re-usability, maintainability would serve such need. This requires the introduction of new approaches or methodologies in the design of an finite element software. Object-oriented programming paradigm is one of

such promising approaches to program design that attempts to eliminate the above-mentioned pitfalls of conventional programming methods and leads to programs which are easy to understand and modify. In this paradigm, it is assumed that the underlying theory is known. In the object-oriented programming approach, programs are divided into what are known as objects and the relationships between the objects are explicitly defined. Data structures are designed such that they characterize the objects, and functions that operate on the data of an object are tied together in the data structure. The implicit reliance on another component's data does not occur. New data and functions can be easily added whenever necessary and thus, the design can be extended with minimal effort [43].

In the recent years a few object-oriented finite element frameworks have been presented which are partially extendable. However, the extendability is limited to a few specific directions, e.g. the introduction of new element types or solving strategies. Much less support is available for task control, creation of new material models, creation of new integration algorithms or extensions of the analysis model. No framework is available which is especially designed to cover the problems encountered when dealing with geotechnical engineering and which can be easily modified. In the present study a new object-oriented finite element framework is developed with special emphasis given on its application for practical geomechanics problems and thus we try to close these gaps.

In this chapter, first we discuss the concept of object-oriented design and programming for finite element method. Next the main class abstractions identified in the new object-oriented finite element framework is presented.

7.2 Object-oriented programming paradigm

Object-oriented programming paradigm forms the basis for a very large part of the software development industry, particularly in the area of computer graphics, databases, graphical interfaces and operating system development. Object-oriented finite element packages, particularly those written in C++, have been shown [46, 111] to have comparable performance to their procedure-oriented language (FORTRAN, C) counterparts. This is due to their support of many advanced features, such

as, *inheritance*, *abstraction*, *encapsulation*, *polymorphism*, *modularity*, and *abstract data types* (which allow the programmer to work at higher levels of abstraction). Larger object-oriented applications end up with numbers of layers of frameworks that cooperate with each other.

To promote code reuse, object-oriented programming languages support the notion of *class hierarchies*, with data and methods of an ancestor class being inherited by descendant classes. The *inheritance* features allows the programmer to define the common function and data used by several classes at the highest possible level in the hierarchy, which avoids the duplication of code at lower levels. Inheritance allows an object of the descendant class to be treated as an object of an ancestor class[111]. It improves the adaptability and extendability of the current project. The descendant classes may add additional attributes and methods, and can redefine the method of an ancestor class, termed as *operator overloading*. *Abstraction* allows localization of data to facilitates maintenance of the software without side effects and *encapsulation* keeps the essential aspects of the data hidden from the user of the data. *Polymorphism* means ability of a single message to activate different methods, which reduces the number of function names to remember. Decomposition into modular framework called as *modularity*, and subsequently identification of objects play a significant role in the object-oriented programming. An object-oriented program is composed of objects [92], each with a number of attributes, that define the state of the object. The behavior of an object is defined by methods, which are procedures for changing or returning the state of the object. An objects method is invoked when another sends a message to that object. The function of an object-oriented program can be viewed as the interaction between the program's object by the sending of messages. To aid in program development, objects with similar attributes and behavior are grouped into classes. The classes are implemented in a programming language. For an implementation the following are provided for each class

- *a class interface* (defining the methods that can be invoked on each object of a class),
- *private data* (defining the attributes held privately by each object),
- *method of implementation* (code defining the sequence of operations that object of the class perform in order to complete the method invoked on the object).

A well designed object-oriented programming system enables programmers to independently develop and validate new code, to maintain and revise existing code, and to be able to introduce new code into existing programs. Fenves [56] identifies the three essential steps in the development of the object-oriented system as: identification of classes, specification of the class interfaces, and implementation. A detail discussion of object-oriented program design can be found in the popular books of object-oriented programming by Cohoon and Davidson [38]. For the shake of completeness, definitions of key terms of object-oriented programming are discussed briefly in the next section.

7.3 Key concepts of object-oriented programming

Object-oriented programming involves a few key concept [22, 109]. The most basic of these is abstraction which makes writing large programs simpler. Another is encapsulation which makes it easier to change and maintain a program. Most importantly, there is the concept of class hierarchies, that other languages do not have at all, and a powerful classification tool that can make a program easily extendible. Different from procedure-oriented programming, in the object-oriented view of programming, a program describes a system of object interaction, where an object is defined as an instance of class. As stated above one can apply the concepts of abstraction and encapsulation by using any languages; only object-oriented languages supports these concepts explicitly. The various key concepts of object-oriented language have been described bellow.

Abstraction

Abstraction is the property of extracting the relevant properties of an object while ignoring nonessential details [38]. The extracted properties define a view of the object. The relevant properties are defined by how we use or manipulate the object. By focusing on the relevant properties and ignoring irrelevant details, the complexity of dealing with an object is reduced. Abstraction is essential for managing the complexity of designing and writing software.

Encapsulation and information hiding

It is the process of separating the aspects of an object into external and internal aspects. The external aspects of an object need to be visible, or known to the other objects in the system. The internal aspects are the details that should not effect other parts of the system. Hiding the internal aspects of an object means that they can be changed without effecting other parts of the system. A benefit of information hiding is that it helps us to make changes to complex system.

Modularity

It refers to the process of dividing an object into smaller pieces or modules so that some goal is easier to attain. A complex object may be structured into components so that each component can be tested individually. Most complex systems are modular. They are constructed by combining simpler working component or packages. Proper modularization of a complex system also helps manage complexity. Breaking things into smaller makes it easier to understand.

Class

A class is an abstract data type definition. It provides implementation details for the data structure used and their operations. Consequently, a class describes a group of objects with similar properties, common behavior, common relationships to other objects, and common semantics. Each object is an instance of a class. The class characterizes its instances by the number, the name, the type of their attributes and the set of methods they implement.

Object

An object is a device capable of performing predefined actions such as sorting information, doing work or providing access to another object. It can be uniquely identified by its name and it defines a state which is represented by the values of its attributes at a particular time. An object has the responsibility to know and the responsibility to do. The state of the object changes according to the methods which are applied to it. We refer to these possible sequences of state changes as the behavior of the object. An object is in fact a new type of variable. Classical types are: integer, real etc. The new variable type is object. It is characterized by loose

typing, which means that an object can be of any type or even a mixture of types. An object is described by its attributes and the methods which drive its behaviour [109]. Objects differ only by the name and values of their attributes (instance variables). One can therefore distinguish two portions in an objects implementation: *a private portion*, which is unique to each instance and a *shared portion* which is common to all instances of an object and defines its class.

Message

A running program is a collection of objects, where objects are created, destroyed and interacting. Objects interact through messages. These messages are requested to activate a method of the receiver object. Objects react when they receive messages by applying methods on themselves. They also may deny the execution of a method, for example if the calling object is not allowed to execute the requested method.

Inheritance

Inheritance is the mechanism which allows a class (descendant class) to inherit properties of another class (ancestor class). Objects of the descendant class thus have access to attributes and methods of the ancestor class without the need to redefine them. Ancestor class is also called as parent class or base class or superclass. Similarly the descendant class can be called as child class or derived class or subclass. Object-oriented programming also allows notion of multiple inheritance. Multiple inheritance means that one subclass can have more than one superclass, which enables the subclass to inherit properties of more than one superclass and to merge their properties.

Hierarchy

A ranking or ordering of objects based on some relationship between them is a hierarchy. Hierarchy helps us understand complex organizations and systems. Object-oriented software design promotes thinking about software in away that more closely models the way we think about and interact with the real world. Such interactions are so routine that it is easy to overlook how amazing this activity is. We are able to make two objects interact and perform a complex activity without understanding the internal activity of either object. We are able to do so because we have appropriate abstraction of each object.

Polymorphism

Object-oriented language supports polymorphism by allowing member functions defined in the classes to be overridden with member functions having the same names(operator overloading), but different implementations, in derived classes (subclasses). In selecting the appropriate member function to call in response to a function invocation, object-oriented language distinguishes between the static type of a reference and the dynamic type of the object it refers to at a given point. The dynamic type must be descendant of the static type. The invocation is type-checked based on the static type of reference. If the function called is a virtual function, the member function associated with the actual object pointed to is called dynamically at run time. If the function is non-virtual, the call will have been statically bound to the member function of the reference's class at compile time.

Binding

Binding is the address management related to control transfer (subroutine calls, messages). If a variable of the type T is explicitly associated with its name N by declaration, we say that N is statically bound to T and this process is called as static binding. With static binding, the same code would be executed at each run. On the other hand, if the variable of type T with name N is implicitly associated by its content, then the association process is called as dynamic binding. Dynamic binding is necessary to support Polymorphism. With dynamic binding, the method invoked will depend on the receiver of the message, as required to support polymorphism.

If we are going to take an object-oriented approach for developing software, it makes sense to use a programming language that supports thinking and implementing solution in terms of objects. A language that has features to support thinking about and implementing solution in terms of object is an object-oriented programming language. Using an object-oriented programming language to implement an object-oriented design is called object-oriented programming.

In this present work, object-oriented programming language C++ has been employed for finite element analysis. In the next section identification of various classes for finite element analysis has been discussed in detail.

7.4 Modeling finite element through object-oriented programming concept

The following approach describes how an object-oriented finite element system is designed, which begins with identifying the classes, then arranging the candidate classes into hierarchies. Emphasis is placed on the possibility of maximizing code reuse and inheritance. Next is the implementation of class interfaces and member functions. Virtual functions are declared on the top, then the polymorphism provides appropriate versions of these foundations in the derived classes. The classes that are responsible for performing the FEM elementary calculations are on the bottom and have most of their codes in their base classes.

The program architecture for implementing the object-oriented framework for finite element analysis is depicted using an object model in Figure 7.4. The frame-

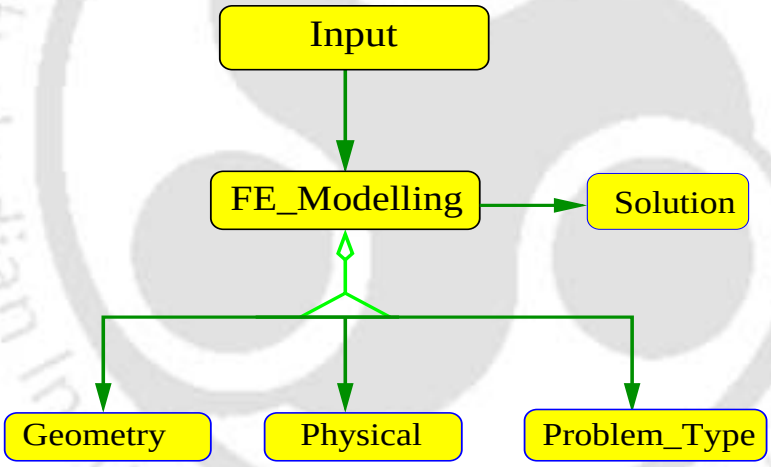


Figure 7.1: Object-oriented framework for FEM [43].

work is designed as an assembly of modules that represent the building blocks of the finite element methodology. The modules were implemented as abstract classes where member functions serve as the programming interface for the framework. The **FE_Modelling** class represents the overall framework and is the basis of handling all the events of a finite element problems. The **FE_Modelling** class inherits the property of **Input** class which is at the top level of the hierarchical order of the classes. The **FE_Modelling** class serves as the interface of the input file and the FEM programs itself, which tells the program what event it will handle, where an

event corresponds to a particular FEM functionality. The class **FE_Modelling** is decomposed into abstract subclasses, **Problem_Type**, **Physical**, **Geometry**, and **Solution**. The class **Problem_Type** is responsible for defining different kinds of problems such as plane stress problem, plane strain problem, plate bending problem etc. The class **Physical** is responsible for defining physical properties of a problem such as elastic or inelastic type. The class **Geometry** is an abstract representation of the analysis module that contains information about the geometric properties of the element, the gauss integration points and the interpolation functions. The class **Solution** contains the information about the constitutive model and integration algorithm employed which is further categorized by subclasses that implement specific equation solving algorithm. The classes, mentioned above have been discussed separately in detail in the next section.

7.5 Designing finite element classes

The first step of the object-oriented programming is to identify classes. This is done on the basis of a detailed analysis about the relationship among the functionality of the modules in a typical FEM package. In this work, some class abstraction have been identified, with different class interfaces depending on the type of the problem being solved. The class that have been presented can be grouped into three broad categories as

1. Modeling classes : classes used to create the finite element model for a given problem.
2. Finite element model classes: classes used to describe finite element model.
3. Analysis classes: classes used to perform the analysis of the finite element model i.e. form and solve the governing equations.

In the following sections, the classes that have been identified under those categories have been discussed. In addition, class interfaces for the basic classes that have been implemented are presented using class model and pseudo C++ codes. The pseudo codes are similar to the codes used in the implementation, the only difference being that private data are not shown.

7.5.1 Modeling classes

These classes are used to create the finite element model for a given problem, i.e. to create the nodes, elements, loads and constraints. For large problems, it is important that the analyst be able to create the model in a simple and descriptive manner. The class, which comes under this category, is the **Input** class.

7.5.1.1 Input class

An Input object is a container responsible for reading all the information for modeling a finite element problem. This class provides information to the other classes of the system. **Input** class has been defined at the top most level of the hierarchical order. It reads and holds the modeling information such as no of elements, type of element, problem type, loading type, boundary conditions etc. Separate input files are created which is made open in the constructor of the **Input** class. The programming interface of the **Input** class has been shown in Figure 7.2.

Seven methods have been declared to perform various tasks. The method *Get_mesh_para()* reads the data containing the information about the parameters like no of elements, type of element, node per element etc. and stores them in proper data member. The function *Get_Int_Type()* is responsible for reading and holding the information about the type and the order of integration to be performed. The function *Get_connectivity()* reads and stores the element connectivity in an array *con*. The function *Get_coordinate()* reads the data regarding position coordinate of the element. Methods *Get_T1_load()* and *Get_Dispbc()* collect information regarding loading and boundary conditions respectively. The method *Get_material()* reads the information regarding the type of the material and the constitutive model used in the analysis from the input file.

7.5.2 Finite element model classes

The instances of the finite element model classes are objects which describe the finite element model and provide information to the other classes of the FE system when necessary. The classes which have been identified in this category are *FE_Modelling*

```
class Input()
 {
   public:
     Input();
      void Get_mesh_para();
      void Get_Int_Type();
      void Get_connectivity();
      void Get_coordinate();
      void Get_T1_load();
      void Get_Dispbc();
      void Get_material();
  } ;
```

Figure 7.2: Interface of the *Input* class

class, *Geometry class*, and *Physical class*. They are described below.

7.5.2.1 FE_Modelling class

This class serves as the interface of the input file and the FEM program itself, which tells the program what event it will handle. This is the base class for modeling an entire FEM problem. It serves as the parent class for the other classes of the system and provides a polymorphic interface to the element classes which are at the last level of hierarchy. In this class all the methods have been declared as pure virtual functions. The interface of the **FE_Modelling** class is shown in the Figure 7.3.

Six steps have been identified in complete modeling and analysis of a finite element problem. These are

1. Calculation of element stiffness matrix and element load vectors.
2. Assembling element stiffness matrix and element load vector into global coordinate and solving the system of equations.
3. Computing displacements at nodes.
4. Computing strains at specific points.

5. Computing stresses by employing a specific stress integration algorithm.

6. Printing out the results to an output file.

```
class FE_Modelling :  public Input
    {
      protected:
        double  *Q, tolar, ....;
        int  ...;
      public:
        virtual void Global_f_preload () {};
        virtual void Global_f () {};
        virtual void Global_K () {};
        virtual void FE_Solver () {};
        virtual void Gravity_Loading () {};
        virtual void Get_OCR () {};
        virtual void Get_Stress_Strain () {};
        virtual void Global_Iter_NR () {};
        virtual void  Glogal_Iter_ARC () {};
        virtual void  Glogal_Iter_MNRC () {};
        virtual void  Glogal_Iter_MNRT () {};
        virtual void  Glogal_Iter_AUTOITR () {};
        virtual void  Glogal_Iter_RK () {};
        virtual void  Output () {};
    };
```

Figure 7.3: Interface of *FE_Modelling* class

Accordingly the programming interface of the **FE_Modelling** class consists of pure virtual functions like: *Global_f_preload ()*, *Global_f ()*, *Global_K ()*, *FE_Solver ()*, *Gravity_Loading ()*, *Get_OCR ()*, *Get_Stress_Strain ()*, *Global_Iter_NR ()*, *Global_Iter_ARC ()*, *Global_Iter_MNRC ()*, *Global_Iter_MNRT ()*, *Global_Iter_RK()*, *Global_Iter_AUTOITR()* and *Output()*. All these functions must be defined in the element classes (which are at the last level of the class hierarchy). The method *Global_f_preload ()* is responsible for calling other class interfaces to assemble element load vector for gravity loading into global load vector. The method *Global_f ()* is responsible for assembling the element load vector for external load other than gravity load into global vector. The method *Global_K ()* assembles the element

stiffness matrix into the global stiffness matrix. The method *FE_Solver ()* tells the object of a *Solver* class to solve the system of equations by some specific equation solving algorithm and is responsible for directing an event to calculate displacements at some specific points. The method *Gravity_Loading ()* is responsible for imposition of gravity loads and calculation of corresponding stresses and strains at some specific points. The method *Get_OCR ()* calculates the over-consolidation ratio (OCR) for a given preloading and present body force. The *Get_Stress_Strain ()* method is responsible for directing an event to calculate strains and then stresses corresponding to the externally applied load at some specific points using a specific stress integration algorithm for a given material model. The methods *Global_Iter_NR ()*, *Global_Iter_ARC ()*, etc. are responsible to employ some specific iterative schemes for solution of nonlinear finite element problems. The method *Output ()* directs to an object of the *Output* class that prints out the analysis output of a problem to output files.

Thus the **FE_Modelling** class performs the task of handling the events through the polymorphic interface to the abstract element classes of the system.

7.5.2.2 Geometry class

The class **Geometry** is an abstract representation of the finite element model that contains information about the geometric property of an element. This class provides the element class with the information about the coordinates, interpolation function and integration algorithm over an element. The implementation of the class **Geometry** is illustrated in the Figure 7.4.

The **Geometry** class has been categorized into two dimensional and three dimensional problems using the derived classes *Geo_Two_D* and *Geo_Three_D* respectively. Both these classes can be again categorized using subclasses to represent different types of elements. In the present framework only the two dimensional interpolation has been implemented but this can be extended for the three dimensional case just by adding subclasses in the category of respective problem type and kept for future extension.

The class *Geo_Two_D* is categorized into rectangular and triangular elements using

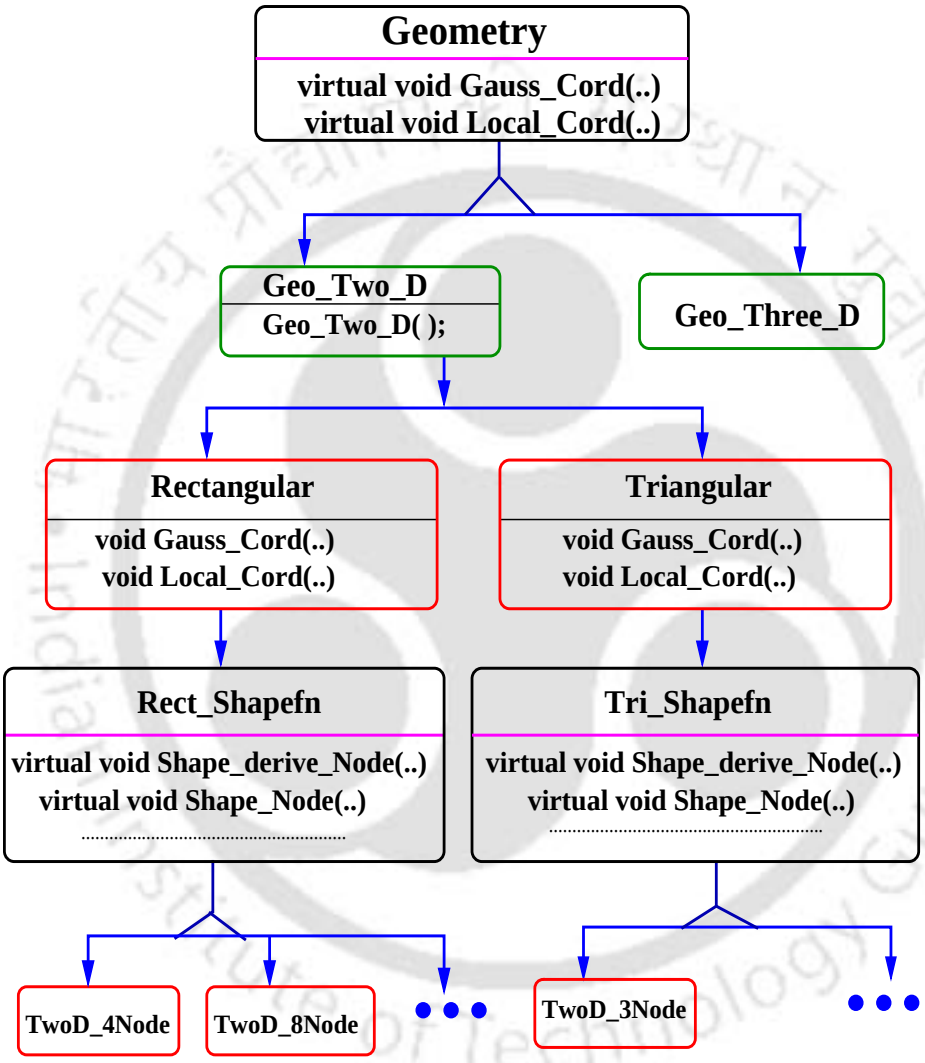


Figure 7.4: *Geometry* class tree of object-oriented finite element analysis.

derived classes *Rectangular* and *Triangular* respectively. The abstract classes for interpolation functions *Rect_Shapefn* and *Tri_Shapefn* has been derived from the *Rectangular* and *Triangular* class respectively. The interpolation classes *Rect_Shapefn* and *Tri_Shapefn* have been further categorized into subclasses like *TwoD_4Node* and *TwoD_8Node* in the case of rectangular element, and *TwoD_3Node* and *TwoD_6Node* for triangular element.

The programming interface of the **Geometry** class consists of two pure virtual functions: *Gauss_Cord()* and *Local_Cord()*. The *Gauss_Cord()* and *Local_Cord()* functions are defined in the classes *Rectangular* and *Triangular* etc. (the third level of the inheritance hierarchy). The *Gauss_Cord()* function numerically integrates the interpolation functions over the length (in the case of 1D element), over the area (in the case of 2D element), or over the volume (in the case of 3D element) of an element. The integration is accomplished using gauss quadrature algorithm. The *Local_Cord()* function is responsible for creating the geometric coordinates of a local element.

The abstract classes *Rect_Shapefn* and *Tri_Shapefn* consist of two pure virtual functions: *Shape_Node()* and *Shape_deriv_Node()*. The *Shape_Node()* function computes the interpolation functions (or shape functions) and is only defined in the last hierarchical level because the shape functions depends on the specific interpolation to be used. The *Shape_deriv_Node()* computes the partial derivative of the shape functions at a point $\mathbf{x}$ in an element. For an example these functions when declared in the class *TwoD_4Node* computes the shape functions and their derivatives for the four nodded element and the element type is determined by the parent class from which the subclass *TwoD_4Node* has been derived. The interface of the class *TwoD_4Node* has been shown in the Figure 7.5.

The primary data members of the **Geometry** class are the order of integration (*Int_order*), number of nodes per element (*Node_elm*) and geometric coordinate array (*cord*). The order of integration is used to determine appropriate gauss quadrature procedure to be used in *Gauss_Cord()* function.

```
class TwoD_4Node: public Rect_Shapefn
 {
   public:
   void Shape_Node (double, double, double*);
   void Shape_deriv_Node (double, double, double**) ;
 }
```

Figure 7.5: Interface of the *TwoD_4Node* class.

7.5.2.3 Physical class

The **Physical** class represents the constitutive relations for different types of analysis such as elastic , elasto-plastic, etc. A graphical representation of the **Physical** class is illustrated in the Figure 7.6.

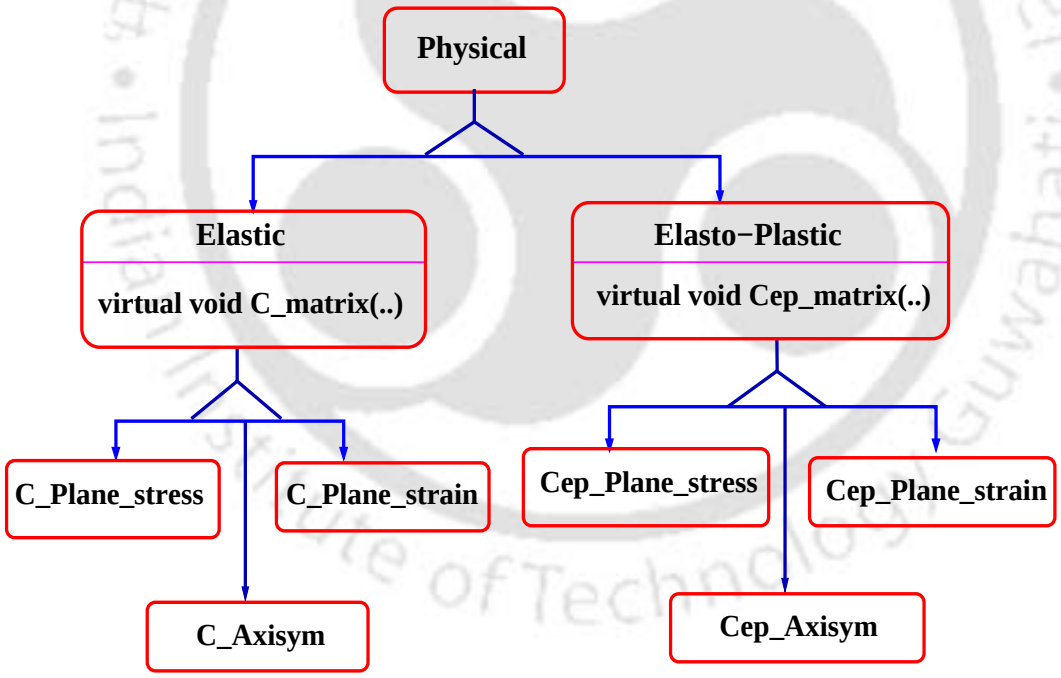


Figure 7.6: Physical class tree of object-oriented finite element analysis.

The **physical** class has been categorized into type of problems such as elastic, elasto-plastic using derived classes (or subclasses) *Elastic*, *Elasto-Plastic* respectively. The class *Elastic* is further categorized into plane stress, plane strain and axisymmetric

problems using derived subclasses as *C_Plane_stress*, *C_Plane_strain* and *C_Axisym*. Similarly the class *Elasto-Plastic* is further categorized into plane stress, plane strain, axisymmetric problems using derived subclasses *Cep_Plane_stress*, *Cep_Plane_strain* and *Cep_Axisym* respectively.

The programming interface of the *Physical* class contains one pure virtual function : *C_matrix(..)*. This function is again kept as pure virtual function in the derived classes *Elastic* and *Elasto-Plastic* (which form the second level of inheritance hierarchy). The function *C_matrix()* is defined in the classes *C_Plane_stress*, *C_Plane_strain*, *C_Axisym*, *Cep_Plane_stress*, *Cep_Plane_strain* and *Cep_Axisym*, which form the last level of the hierarchical order of *Physical* class. The function *C_matrix* defines the constitutive relations of the material, which remain constant for linear elastic case. It changes with loading and has to be updated continuously in case of nonlinear elastic and elasto-plastic cases. Therefore, it makes sense to define the elastic and elasto-plastic constitutive relations separately.

7.5.3 Analysis classes

Each type of finite element analysis has its own governing and constitutive equations. These equations describe the physical problem that is to be analyzed (such as plane stress, plane strain, axisymmetric, etc.). It can be expressed in general using the following type of weak form of a variational principle.

$$\int_V \delta \boldsymbol{X}^T \; \boldsymbol{B}^T \; \boldsymbol{\sigma} \; \boldsymbol{X} \; dV = \int_S \delta \boldsymbol{X}^T \; \boldsymbol{N}^T \; \boldsymbol{f} \; dS + \int_V \delta \boldsymbol{X}^T \; \boldsymbol{N}^T \; \boldsymbol{b} \; dV \qquad (7.5.1)$$

The above equation represents the weak form applied to an element, where V is the volume of the element, S is the surface area of the element that lies on the boundary of the object, $\boldsymbol{f}$ is the vector of the traction force acting on per unit area of these boundaries, $\boldsymbol{b}$ is the body force vector acting on per unit volume of the element and $\boldsymbol{N}$ is the vector of the shape functions of the element. The vector $\boldsymbol{X}$ contains the nodal variables while the $\delta\boldsymbol{X}$ contains the corresponding virtual variables. The matrix $\boldsymbol{B}$ denotes an operator that, when applied to the nodal variable vector, yields some physical quantities associated with the analysis such as the gradient of the strain. The object of the *Analysis* class forms and solves the governing equations for the finite element model. The type of the analysis that

can be performed by the analyst depends on the *Analysis* classes provided and the interface of the component of the *Finite element model* class. The classes identified in this category are described below.

7.5.3.1 Problem_Type class

The **Problem_Type** class is a representation of the analysis module that contains information about the nature of the analysis, corresponding strain displacement relations and the associated data. Specific analysis type may be defined by inheriting from the **Problem_Type** class. Figure 7.7 shows the object model of the **Problem_Type** class architecture.

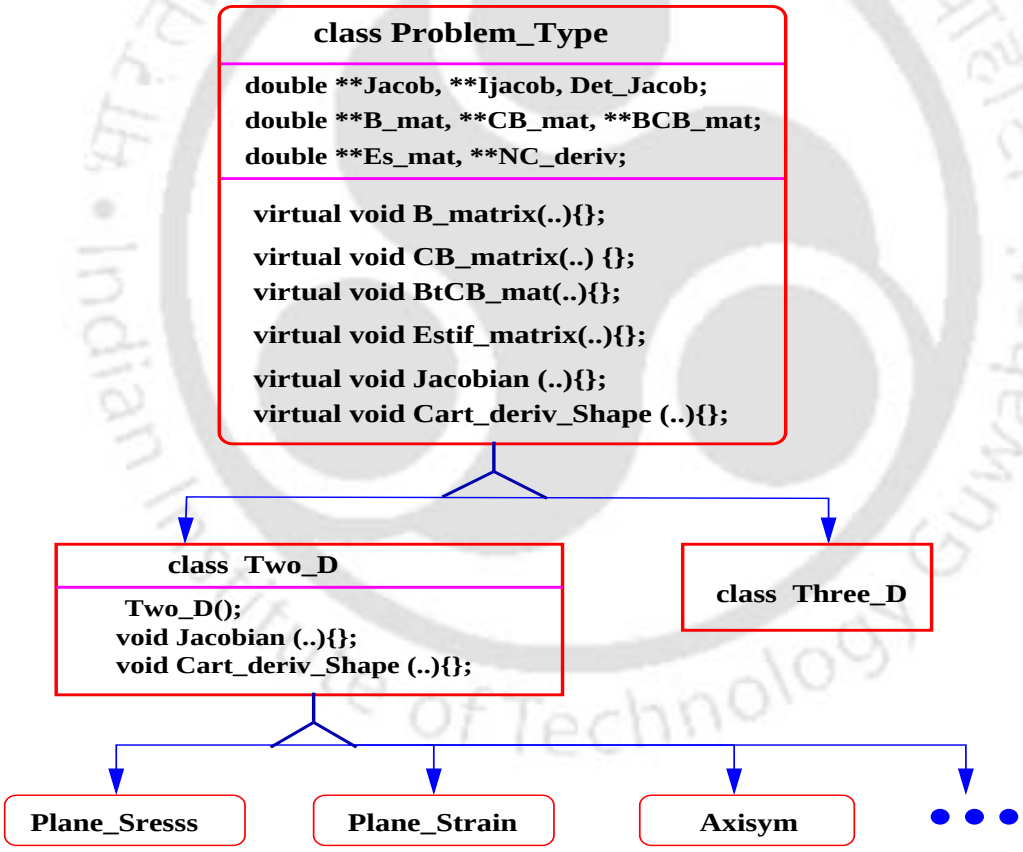


Figure 7.7: Problem_Type class tree of object-oriented finite element analysis.

The data stored in the **Problem_Type** class are the information required for describing equations of the physical problem to be analyzed. The **Problem_Type**

class has been categorized into types of analysis such as two dimensional and three dimensional using derived classes (or subclasses) *Two_D* and *Three_D* respectively.

The class *Two_D* is divided into abstract subclasses such as *Plane_Stress*, *Plane_Strain*, *Axisym*, etc. Each of these classes is responsible for performing specific type of analysis. The abstract classes *Plane_Stress*, *Plane_Strain*, and *Axisym* represent the analysis module for plane stress, plane strain and axisymmetric problems. These classes can be further subdivided according to the necessity of the problem.

The programming interface of the **Problem_Type** consists of six pure virtual functions: *B_matrix()*, *CB_matrix()*, *BtCB_matrix()*, *Estif_matrix()*, *Jacobian()* and *Cart_deriv_Shape()*. The functions *Jacobian()* and *Cart_deriv_Shape()* are defined in the derived classes *Two_D* and *Three_D*, which are at the second level of hierarchy. The function *Jacobian()* computes the jacobian matrix, its determinate and inverse. The function *Cart_deriv_Shape()* is responsible for transforming the derivative of the shape functions from natural coordinate to cartesian coordinate. The data members include the array of jacobian matrix *Jacob*, the variable to store its determinate *Det_Jacob*, the array to store its inverse *IJacob*, and the array of the derivative of the shape function with respect to cartesian coordinate *NC_deriv*.

The function *B_matrix()* computes the strain displacement matrix at a point within the element. The function *CB_matrix()* is responsible for computing the product of constitutive matrix and strain displacement matrix ($\boldsymbol{CB}$), where $\boldsymbol{C}$ may be elastic or elasto-plastic depending on the type of the problem and loading stage. The function *BtCB_matrix()* computes the matrix ($\boldsymbol{B^T CB}$), and the function *Estif_matrix()* computes the element stiffness matrix at a point within the element respectively. These functions are defined in the derived classes of *Problem_Type* such as *Plane_Stress*, *Plane_Strain* and *Axisym* (which are at the third level of hierarchy). The input to these functions are element number *n*, the array of shape function values *N_vec*, the array of derivative of shape functions *N_deriv*, and the constitutive matrix *C_mat* for nonlinear elastic and elasto-plastic analysis. The data members include the array of the strain displacement matrix *B_mat*, the array of the product of constitutive matrix and strain displacement matrix *CB_mat*, and the array *Es_mat* to store the calculated value of the stiffness matrix at element level.

7.5.3.2 Solution class

The class *Solution* is a representation of the analysis module that contains information about different equation solving algorithms, available material models, and corresponding stress integration algorithms along with the associated data. Specific equation solving algorithm and material model may be defined by inheriting from the *Solution* class. Figure 7.8 graphically depicts the implementation of the *Solution* class.

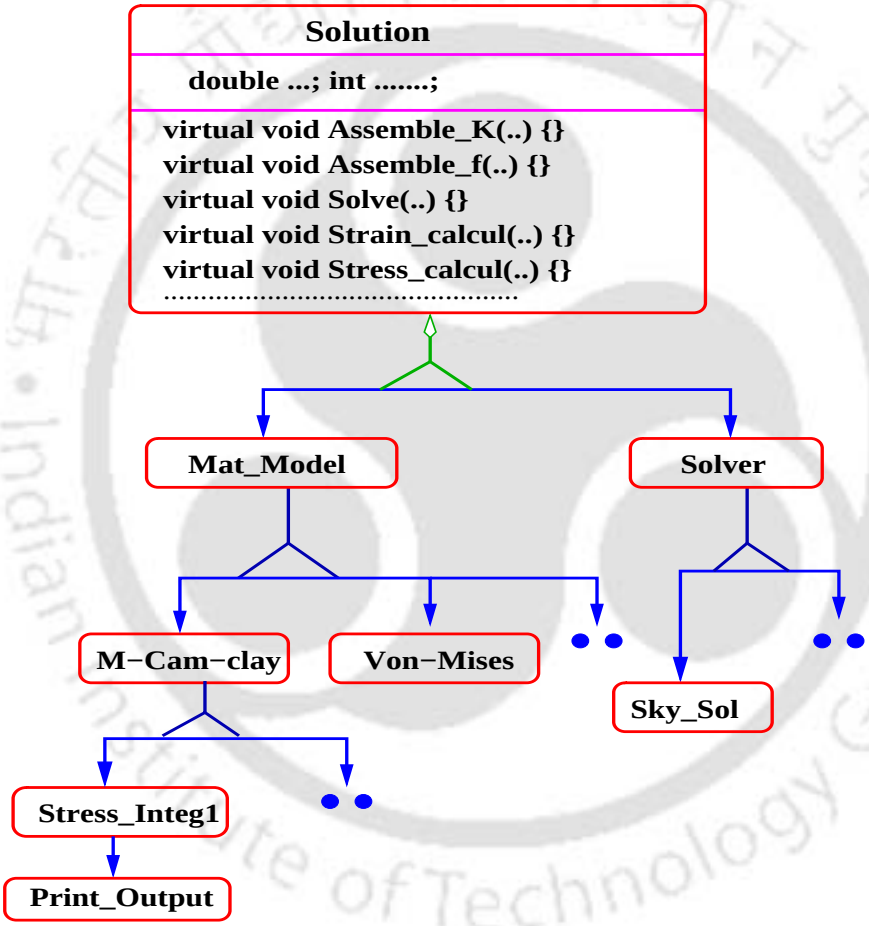


Figure 7.8: Solution class tree of object-oriented finite element analysis.

The class *Solution* is divided into two abstract subclasses as *Mat_Model* and *Solver*. The class *Mat_Model* has been further categorized into types of material models used for the analysis, such as MCCM, von Mises model, etc., using derived subclasses *M-Cam-clay*, *Von-Mises*, etc. respectively. The subclasses *M-Cam-clay* can be again

divided into different classes depending on the stress integration algorithms used to integrate the rate constitutive equations of a specific material model. Figure 7.9 shows the interface of the class *M-Cam-clay*. The class *M-Cam-clay* contains virtual functions like *Update_stress()*, *Tstress_Update()*, *Elastic_moduli_update()*, etc. and these functions are defined in the subclasses of the *M-Cam-clay* class. The class *Print_Output* is defined at the last level of the hierarchy, which print the required results into an output file. The class *Solver* can be divided into abstract subclasses such as *Skyline_Solver*, *Band_Solver*, etc., which are responsible for employing specific equation solving algorithms.

The *Solution* class contains virtual functions like: *Assemble_K()*, *Assemble_f()*, *Solve()*, *Strain_calcul()*, and *Stress_calcul()*. The *Assemble_K()* function assembles the global stiffness matrix in the appropriate form. The *Assemble_f()* function assembles the global forcing vector which is the right hand side of the global equations. The *Solve()* function defines the equation solving algorithm used to solve the global set of linear equations. These functions must be defined by the subclasses of the *Solver* class. The functions *Strain_calcul()*, and *Stress_calcul()* calculate the strains and stresses corresponding to a known displacement and they must be defined in the subclasses of the *Mat_Model* class. The main data members of the *Solver* class include the arrays *K_mat* and *T_load*, which stores the global stiffness and global forcing vectors respectively. The format of storing stiffness matrix in *K_mat* and the forcing vector in *T_load* depends on the solution algorithm employed to solve the resulting algebraic equations.

7.5.4 Abstraction of elements from the framework

An element can be categorized by the type of analysis it is used for, by its geometry and interpolation, and by the material model and its method of integration. Therefore, it makes sense to define the element in the framework as the composition of *Problem_Type* class, *Geometry* class and *Mat_Model* class. The separation of these three aspects of an element, significantly promotes software reuse and enhances the extendability of the finite element code.

An analysis module and interpolations can be reused in many different elements so that multiple elements can be easily added to the software by combining it with

```
class M−Cam−clay : public Mat_Model
 {
  public:
    double .....; int .......;
   virtual void   Update_stress (){};
   virtual void   Tstress_update (){};
   virtual void   Elastic_moduli_update (){};
   ....................................
   void  Strain_calcul (){};
   void  Stress_calcul (){};
   ....................................
 } ;
```

Figure 7.9: Interface of class *M-Cam-clay*.

```
class Sky_sol :public Solver
{
   public:
    int .....; double .....;
    ....................................
    Sky_sol();
    void  Profile();
    void Assemble_K(in, double **);
    void  Assemble_f(int, double *);
    void  Solve();
    void  Load_Input();
    void  Clear_T_load();
    void   Clear_Kmat();
     ....................................
  } ;
```

Figure 7.10: Interface of class *Sky_Sol*.

available material models and its stress integration algorithms. Similarly, upon adding a new material model or new integration algorithm for an existing material model, many new elements can be immediately created by combining it with all the compatible analysis modules and interpolation types available. For example, if a four nodded quadrilateral element has been defined for performing plane strain analysis using MCCM with a specific integration algorithm, an eight nodded quadrilateral element can be defined for the same analysis type using the same material model and integration algorithm by simply defining a eight nodded interpolation class and combining it with the previously created plane strain analysis module.

Similarly, for the same material model and integration type, by defining an axisymmetric analysis type and combining it with previously created interpolation type, a four node quadrilateral axisymmetric element can be created. The abstraction of the element is quite simple, suppose, we want to solve a plane stress problem using eight nodded rectangular element, MCCM with specific integration algorithm, then we will be creating a module for the element that will inherit the property of *Plane_Stress* class, *Rectangular* class, and *Mat_Model* class. Thus this element module automatically becomes an abstract subclass of *FE_Modelling* class. Now an object of this element module will be responsible for solving the entire finite element problem.

7.5.5 Analysis flowchart

Figure 7.11 gives a summary of the steps involved in using the modules in the framework to perform a finite element analysis.

Firstly, the constructor of the Input class is created which reads the data from an input file and store the data in appropriate arrays. Once all the data representing the finite element model are read and stored, the object of the Geometry and Physical class is created and constitutive matrix and Gauss coordinates are stored. Then the object of interpolation type subclass is created which stores the shape functions and their derivatives. Then the derivatives of shape functions are passed to the object of the Problem_Type class which computes strain displacement matrix, [B] and the element stiffness matrix $[K]_e$. After calculating the element stiffness matrix an object of Solver class is created and the assembly of global stiffness matrix is

initiated.

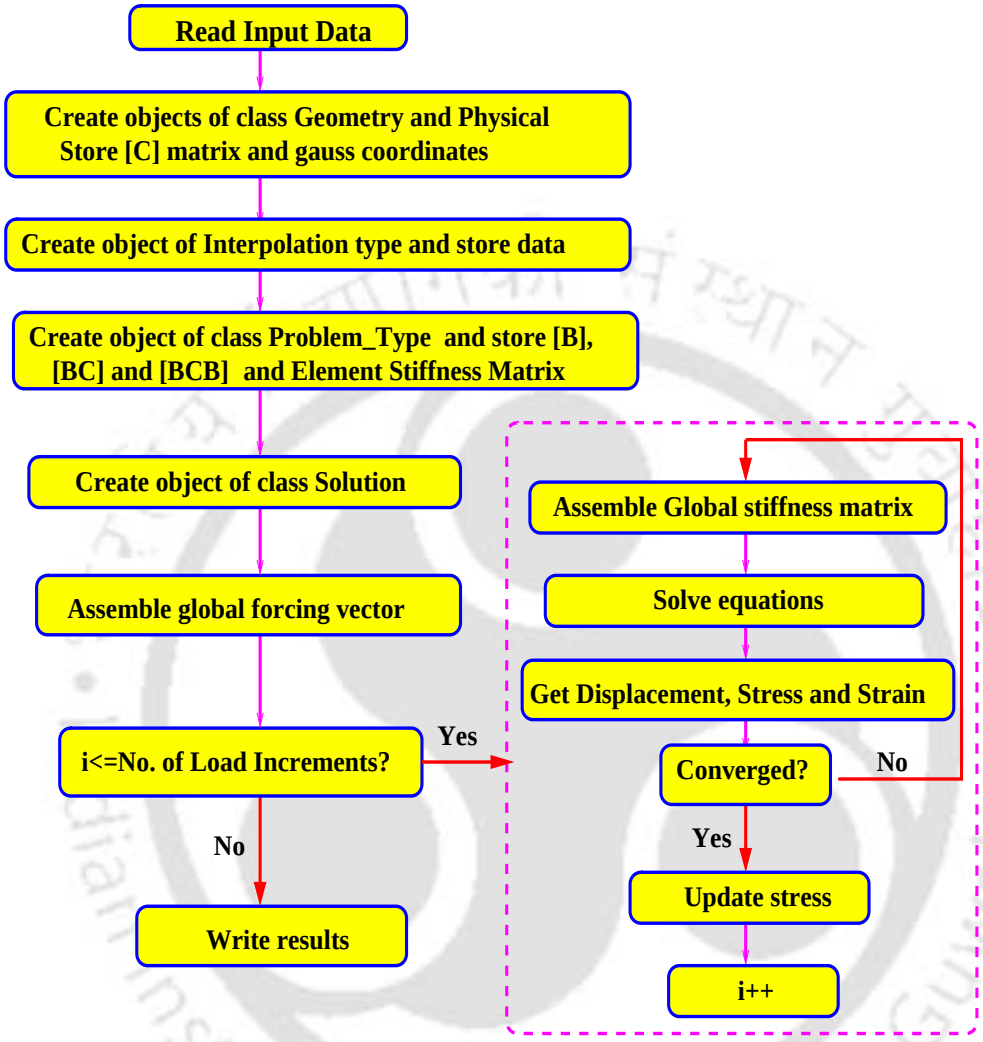


Figure 7.11: Finite element analysis flowchart.

The *Estif_matrix()* function of the *Problem_Type* class is used to compute the stiffness matrix for every element in the abstract Element class. The element stiffness matrix is then assembled into the correct location in the global stiffness matrix. Clearly, the assembly algorithm does not have to be modified when new elements are added to extend the software. After assembling the stiffness matrix, the right hand side of the global equation is computed using *Assemble_f()* function before the equation are solved using the member function *solve()*. The stress and strains are calculated by the member function of the *Solution* class, and finally, the member

functions of the *Print_Output* class are used to write the results of the analysis to an output file.

7.6 Summary

In this chapter, basic concepts related with the object-oriented finite element program design have been explained. The emphasis has been given to the design of finite element codes in the realm of basic object-oriented concepts, like, abstraction, encapsulation, class hierarchy and polymorphism. The overall framework and the modules that take up the finite element analysis are introduced. These modules were implemented as abstract classes so that one could consider the framework to be an abstract implementation of the finite element methodology. The details of the modularization, interaction between the modules and their implementation in FEM are described. Detail discussion on the classes that have been implemented and identified in designing finite element problems are presented. New class interfaces are provided both to demonstrate the functionality of the classes and for the use in subsequent chapters.

Implementation of the identified classes and abstraction of elements for analyzing plane strain and axisymmetric problems using different load stepping schemes explained in Chapter 5, are discussed in Chapter 8. The MCCM with new integration algorithms discussed in Chapter 4 and Chapter 6 is implemented and used to solve axisymmetric and plane strain problems in geomechanics.

.

Chapter 8

Implementation of MCCM in object-oriented finite element framework

8.1 Introduction

In this chapter the object-oriented implementation of the MCCM has been discussed. The integration algorithms and load stepping schemes discussed in Chapter 4, Chapter 5, and Chapter 6, are implemented for MCCM in the proposed object-oriented finite element framework. Different elements have been abstracted to serve these purposes as explained in Chapter 7.

The overall framework of the object-oriented finite element is presented in the Figure 8.1. As shown in the figure, the type of interpolations implemented in the object-oriented programming framework are for four nodded and eight nodded rectangular, and three nodded triangular elements in two dimension. Three dimensional interpolation can be easily added to extend it for three dimensional analysis, if required. However, in many practical situations, the geometry and loading will be such that the problems can be reduced to two dimensional problems without much loss of accuracy. Plane stress and Plane strain are such two dimensional idealizations of stress analysis, encountered in the field of geotechnical engineering. Axisymmetric problems are arising in some special cases, such as laboratory testing. Keeping that in view, the types of analysis incorporated in the framework are for plane stress,

plane strain and axisymmetric problems only. Different types of material models can be used to simulate the behaviour of soil. Keeping the wide spread benefits of Cam clay group of models, as explained in Chapter 2, the MCCM is selected for the present study and implemented in the proposed framework. Different integration algorithms are developed in the past to integrate the rate constitutive equations of MCCM. The new integration algorithm presented in the Chapter 4 is implemented in the object-oriented programming framework. The class *Stress_Integ1* is derived from the *M-Cam-clay* class to incorporate this integration algorithm. Another class *Stress_Integ2* (not shown in the figure) is derived from the same *M-Cam-clay* class to incorporate the integration algorithm by Runge-Kutta method, which is explained in Chapter 6.

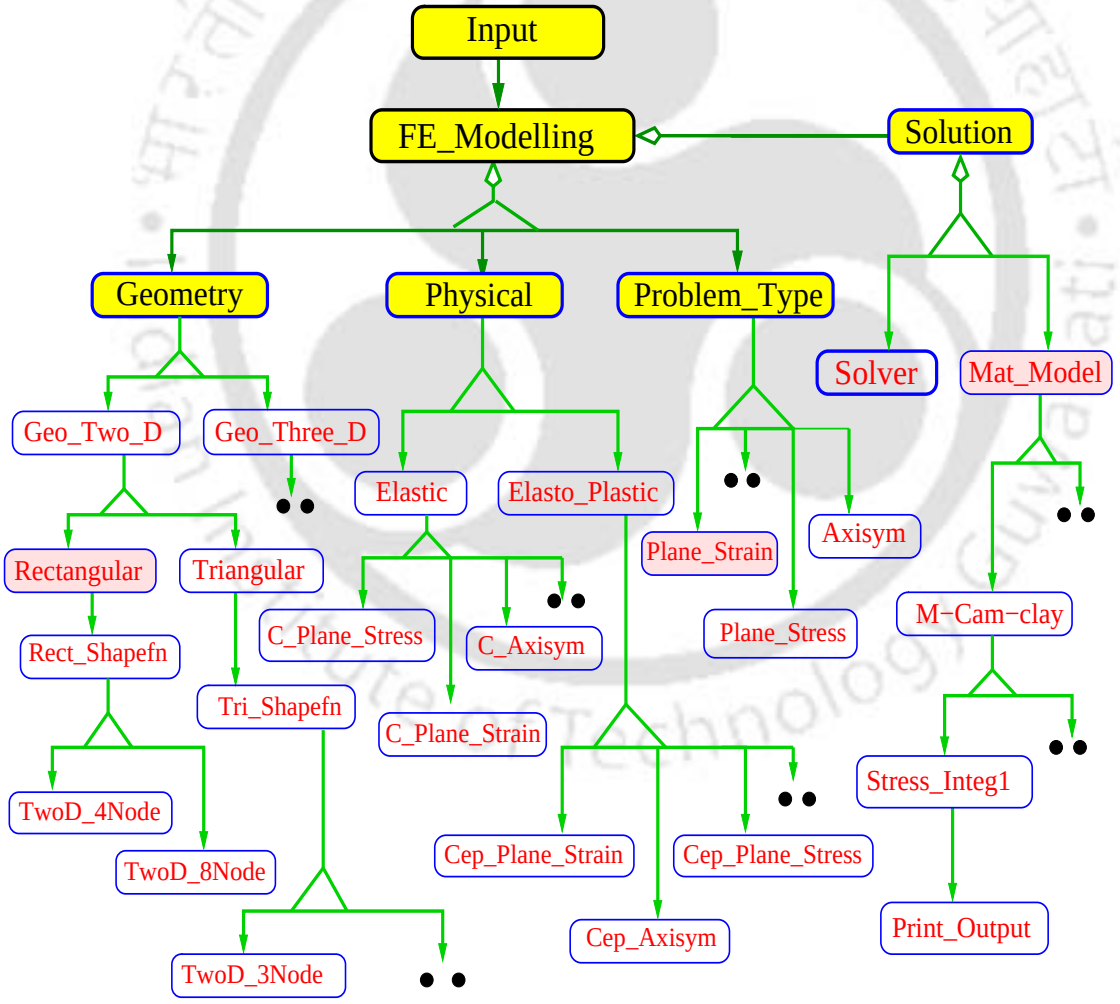


Figure 8.1: Overall framework of the object-oriented finite element analysis

8.2 Object-oriented approach of modeling plane strain analysis

In this section an object-oriented approach has been discussed for modelling plane strain analysis problems. The plane strain condition is characterized by very large dimension in one direction (say z-coordinate direction), in comparison with the dimensions in other two directions (x- and y-coordinatedirections) of the structure. The applied loads are uniformly distributed with respect to the large dimension and act perpendicular to it. Some important practical applications of this representation in geotechnical works occur in the analysis of dams, tunnels, strip footings, etc. In these cases, the strain components ε_z ε_{xz} and ε_{yz} are zero and the state of strain is then specified by ε_x ε_y and ε_{xy} only (functions of x and y).

As discussed in Chapter 7 several classes has been defined and placed into proper hierarchical order to develop the framework for the plane strain analysis algorithm. For analyzing plane strain problems a separate class *Plane_Strain* have been created which is an abstract representation of the plane strain analysis algorithm. This class is responsible for analyzing all types of plane strain problems. The interface of the class *Plane_Strain* is shown in Figure 8.2.

```
class Plane_Strain: virtual public Two_D
{
   public:
    Plane_Strain (){} ;
    void B_matrix(..) ;
    void CB_matrix(..) ;
    void BtCB_mat(..) ;
    void Estif_matrix(..) ;
};
```

Figure 8.2: Interface of the class Plane_Strain

The class *Plane_Strain* consists of four functions : *B_matrix()*, *CB_matrix()*, *BtCB_mat()* and *Estif_matrix()*. The function *B_matrix()* calculates the strain displacement matrix for each gauss points. Function *CB_matrix()* computes the product of constitutive matrix and strain displacement matrix. Finally the function *Estif_matrix()*

calculates the stiffness matrix for an element after getting the value of $\boldsymbol{B}^T\boldsymbol{C}\boldsymbol{B}$ from the function *BtCB_mat()*.

8.2.1 Plane strain analysis with eight nodded isoparametric quadrilateral element

Though a number of finite element models are available, a simple, efficient, higher order eight nodded isoparametric element is implemented using object-oriented fundamental. Several class interfaces have been created for analyzing different types of plane strain problems, several Element classes have been defined, which forms the last level in the hierarchy. Table 8.1 shows the Elements classes (**Element01**-**Element06**) abstracted from the framework for analyzing different types of plane strain problems using eight nodded isoparametric quadrilateral interpolation.

The abstraction of the element classes (only **Element01** and **Element06** are shown) for Plane_strain analysis is depicted in Figure 8.3. The classes (**Element01** through **Element05**) are derived from *Rectangular*, *Plane_Strain* and *Print_Output* classes, thus inherits the property of these classes. The *Rectangular* class provides the information about interpolation functions and integration procedure. The type of analysis is provided by the class *Plane_Strain*. The information regarding the type of material model and integration algorithm is provided by the class *Print_Output*. The class **Element06** is derived from *Rectangular*, *Plane_Strain* and *Print_Output1* classes, thus it inherits the property of different stress integration algorithm through the class *Print_Output1*.

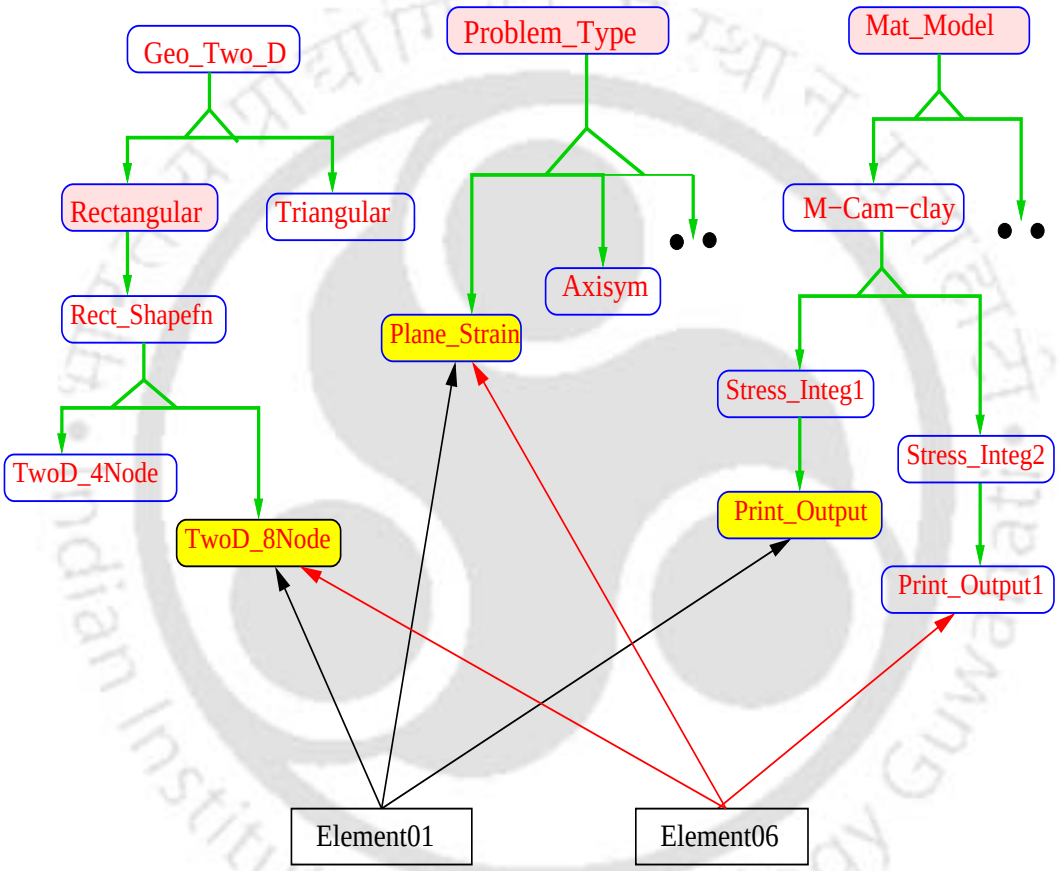


Figure 8.3: Abstraction of plane strain element classes

Table 8.1: List of Elements abstracted from the object-oriented programming framework for plane strain analysis

Sl No.	Element No.	Interpolation Type	Analysis Type	Solution Scheme	Integration Scheme
1	Element01	Iso_Quad8N	Plane_strain	NR	Integ1
2	Element02	Iso_Quad8N	Plane_strain	ARC	Integ1
3	Element03	Iso_Quad8N	Plane_strain	MNRC	Integ1
4	Element04	Iso_Quad8N	Plane_strain	MNRT	Integ1
5	Element05	Iso_Quad8N	Plane_strain	AUTOITR	Integ1
6	Element06	Iso_Quad8N	Plane_strain	NR	RK

Note:

Iso_Quad8N: 8-nodded isoparametric quadrilateral elements,

NR: Newton-Raphson method,

ARC: Arclength method,

MNRC: Modified NR method with Chen accelerator,

MNRT: Modified NR method with modified Thomas accelerator,

AUTOITR: Automatic iteration method,

Integ1: Integration method explained in Chapter 4,

RK: Runge_Kutta method.

A class interface for the eight nodded isoparametric quadrilateral plane strain element with Newton-Raphson method using the new integration algorithm for MCCM (**Element01**) is shown in Figure 8.4. The interface of the class **Element01** consists of eight functions, which were declared as pure virtual functions in the *FE_Modelling* class. The function *Global_f_Preload()* assembles the global load vector for preloading-loading if an over-consolidated soil layer is to be established. The function *Get_OCR()* calculates the initial stresses and over-consolidation ratio at each gauss point of the soil profile. The functions *Global_f()* and *Global_K()* assemble and store the global forcing vector and global stiffness matrix. The function *FE_Solver()* solves the governing finite element equations employing a specific equation solving algorithm and calculates the displacements at each node of the finite element mesh. The function *Get_Stress_Strain()* calculates stresses by employing a specific integration algorithm for the constitutive model under consideration. This function needs modification for different types of material models and integration algorithms. Function *Global_Iter_NR()* are written to perform the Newton-Raphson iteration at global level.

Figure 8.6 through Figure 8.10 show the interface of the other five elements. In *Element02*, *Element03*, *Element04* and *Element05*, modification is required only in the function for global iteration and the other functions remain unchanged, as same material model with the same integration algorithm is used with different iterative schemes. In case of *Element06*, modification is required in the function for global iteration as well as in the function *Get_Stress_Strain()* to incorporate different integration algorithm.

Figure 8.11 through Figure 8.14 show the pseudo codes for the functions *Global_f()*, *Global_K()*, *FE_Solver()*, and *Get_Stress_Strain()*. The functions *Stress_In(..)* and *Elastic_Moduli_input(..)* are invoked in the function *Global_f()* to supply the information regarding the initial stresses, strains, void ratio and elastic moduli at eachg gauss point of the finite element mesh. The functions *Tangent_Moduli1(..)* and *Tangent_Moduli2(..)* are called in the function *Global_K()* to calculate the tangential modulus by deriving the stress tensor with respect to strain tensor for a specific stress integration algorithm and are defined as virtual functions in the class *M-Cam-clay*. They are to be defined in the subclass of the *M-Cam-clay* class for a specific stress integration algorithm. The functions *Assemble_f()*, *Assemble_K()*, *Decom-*

position(), *Load_reduction()*, *Backsubstitution()*, and *Disp_calculation()* are defined in the subclass of the *Solver* class. Functions *Assemble_f()* and *Assemble_K()* are invoked to assemble the forcing vector and stiffness matrix at global level. Functions *Decomposition()*, *Load_reduction()*, *Backsubstitution()*, and *Disp_calculation()* are invoked to calculate the displacement by using a specific equation solving algorithm. The functions *Update_stress(..)*, *Effective_pl_strain(..)*, *Tstress_update(..)*, *Elastic_moduli_update(..)*, *Elastic_Tstress_update(..)* are invoked in the function *Get_Stress_Strain()* to calculate the stresses and strains at gauss point by employing a specific stress integration algorithm for MCCM equations. They are defined in the subclass of the *M-Cam-clay* class.

A constructor is defined (Figure 8.5) in which the reference of the objects of the interpolation class *TwoD_8Node* is saved as the array of the object of *Rect_Shapefn* class. The constructor also saves the reference of the objects of the classes *Rectangular*, *Sky_sol* and *Print_Output* as the data members of the objects of the classes *Geometry*, *Solver* and *Stress_Integ1* respectively. The class *Rectangular* provides the **Element01** class with information about geometric coordinates of the element, the class *Sky_sol* provides information about type of solver to be used in the analysis, the class *Print_Output* provide the **Element01** class with information about the material model and its corresponding integration algorithm.

Another six numbers of elements are abstracted to perform axisymmetric type of analysis with the different integration algorithms and iteration schemes. For this purpose, the elements are derived from *Rectangular*, *Axisym* and *Print_Output* classes, thus inherits the property of these classes. There will be minor changes in the interfaces of these classes. For example, in place of the object of the class *Plane_Strain*, we need to define the object of the class *Axisym*.

In the same way discussed in the previous sections we can abstract many new elements from the existing codes or can add new elements to the finite element system. In the Chapter 9 the elements that have been created is examined with the aid of some numerical examples.

The pseudo codes for some of the important functions of the abstraction are presented here.

```
class Element01 :public Rectangular, public Plane_Strain, public Mat_Model
        {
          protected:
            int       N, K, i, j, a;   double   weight, thick;
            Geometry            *Geo;
            Rectangular         Rn;
            Rect_Shapefn        *Rec[20];
            TwoD_8Node          N8;
            Cep_Plane_strain    E1;
            Plane_Strain        ps1;
            Solver              *Solv;
            Sky_sol             Sky;
            Stress_Integ1       *Str1;
            Print_Output        Prnt;
          public:
            Element01();
            void  Global_f_Preload();
            void  Global_f();
            void  Global_K();
            void   FE_Solver();
            void   Get_OCR();
            void  Get_Stress_Strain();
            void  Global_Iter_NR();
            void   Output();
         } ;
```

Figure 8.4: Interface of class *Element01*

```
Element01 :: Element01()
  {
     a=Int_order;
     Rec[8]=&N8;
     Geo=&Rn;
     Solv=&Sky;
     Str1=&Prnt;
     cout<<"\n In Element01 constructor \n";
  }
```

Figure 8.5: Pseudo-code for constructor of class *Element01*

```
class Element02 :public Rectangular, public Plane_Strain, public Mat_Model
   {
     protected:
        int        N, K, i, j, a;
        double   weight, thick;
        Geometry          *Geo;
     public:
        Element02();
        void  Global_f_Preload();
        void  Global_f();
        .....................
        .....................
        void  Get_Stress_Strain();
        void   Global_Iter_ARC();
        void   Output();

   } ;
```

Figure 8.6: Interface of class *Element02*

```
class Element03 :public Rectangular, public Plane_Strain, public Mat_Model
        {
          protected:
            int       N, K, i, j, a;
            Geometry          *Geo;
            ....................
            ....................
          public:
            Element03();
            void  Global_f_Preload();
            void  Global_f();
            ....................
            ....................
            void  Get_Stress_Strain();
            void  Global_Iter_MNRC();
            void   Output();
        } ;
```

Figure 8.7: Interface of class *Element03*

```
class Element04 :public Rectangular, public Plane_Strain, public Mat_Model
        {
          protected:
            int       N, K, i, j, a;
            Geometry          *Geo;
            ....................
            ....................
          public:
            Element04();
            void  Global_f_Preload();
            void  Global_f();
            ....................
            ....................
            void  Get_Stress_Strain();
            void  Global_Iter_MNRT();
            void   Output();
        } ;
```

Figure 8.8: Interface of class *Element04*

```
class Element05 :public Rectangular, public Plane_Strain, public Mat_Model
        {
          protected:
            int       N, K, i, j, a;
            Geometry          *Geo;
            ....................
            ....................
          public:
            Element05();
            void  Global_f_Preload();
            void  Global_f();
            ....................
            ....................
            void  Get_Stress_Strain();
            void  Global_Iter_AUTOITR();
            void   Output();
        } ;
```

Figure 8.9: Interface of class *Element05*

```
class Element06 :public Rectangular, public Plane_Strain, public Mat_Model
        {
          protected:
            int       N, K, i, j, a;
            Geometry          *Geo;
            ....................
            Print_output1     Prnt1;
          public:
            Element06();
            void  Global_f_Preload();
            void  Global_f();
            ....................
            ....................
            void  Get_Stress_Strain();
            void  Global_Iter_RK();
            void   Output();
        } ;
```

Figure 8.10: Interface of class *Element06*

```
void Element01 :: Global_f()   //Assemble the Global forcing vector
 {
    Solv->Profile2(Height);
    Str1->Clear_T_load();
    int count1=0;
    Iter=1;
    for(K=0; K<Num_elm; K++)
    {
       ps1.Clear_Estiff();
       for(N=0; N<Int_order*Int_order; N++)
       {
          Geo->Gauss_Cord(a, Gx, Gy, w);
          eta=Gx[N]; zeta=Gy[N]; weight=w[N];
          Geo->Local_Cord(K, cordL);
          Rec[8]->Shape_deriv_Node(eta, zeta, N_deriv);
          Rec[8]->Shape_Node(eta, zeta, N_vec);
          Str1->Lcord_Stress_In(K);
          Rec[8]->Cart_Gauss_Cord(cordL, N_vec, dcount, Gcord);
          Str1->Stress_In(Gcord, count1);
          Str1->Elastic_moduli_input(count1, En);
          E1.C_Matrix(count1, En, C_Mat);
          ps1.Jacobian(cordL, N_detriv, weight, Djwt);
          ps1.Cart_deriv_shape(N_deriv, N_vec);
          ps1.B_matrix(Gcord, count1, B_mat);
          ps1.Load_Mat(Q);
          Str1->Gravity_Load(count1, B_mat, Djwt);
          count1++;
       }
       Solv->Assemble_f(K, Q, T_load);
    }
 }
```

Figure 8.11: Pseudo codes for function Global_f()

```
void Element01 :: Global_K()   //Assemble the Global Stiffness Matrix
 {
    Solv->Clear_Kmat();
    int dcount=0;
    for(K=0; K<Num_elm; K++)
    {
       ps1.Clear_Estiff();
       Geo->Local_Cord(K, cordL);
       for(N=0; N<Int_order*Int_order; N++)
       {
          Geo->Gauss_Cord(a, Gx, Gy, w);
          eta=Gx[N]; zeta=Gy[N]; weight=w[N];
          Rec[8]->Shape_deriv_Node(eta, zeta, N_deriv);
          Rec[8]->Shape_Node(eta, zeta, N_vec);
          E1.C_Matrix(scount, En, C_Mat);
          Str1->Iden_tensor();
          Str1->Tangent_Moduli1(dcount);
          Str1->Tangent_Moduli2(dcount);
          Rec[8]->Cart_Gauss_Cord(cordL, N_vec, dcount, Gcord);
          ps1.Jacobian(cordL, N_detriv, weight, Djwt);
          ps1.Cart_deriv_shape(N_deriv, N_vec);
          ps1.B_matrix(Gcord, dcount, B_mat);
          ps1.CB_matrix(C_Mat, CB_mat);
          ps1.BtCB_matrix(CB_mat);
          ps1.Estif_matrix(Es_mat);
          dcount++;
       }
      Solv->Assemble_K(K, Es_mat);
    }
 }
```

Figure 8.12: Pseudo codes for function Global_K()

```
void Element01 :: FE_Solver()
  {
      switch(IterType)
      {
          case 1:
              Solv->Decomposition();
          break;
          case 2:
              if(Iter==1)    //For Modified Newton_Raphson Method
                  Solv->Decomposition();
           break;
      }
      Solv->Load_reduction();
      Solv->Backsubstitution();
      Solv->Disp_calculation();
  }
```

Figure 8.13: Pseudo codes for function FE_Solver()

```
void Element01 :: Get_Stress_Strain()
 { int scount=0;
   for(K=0; K<Num_elm; K++) {
     for(N=0; N<Int_order*Int_order; N++) {
       Geo->Gauss_Cord(a, Gx, Gy, w);
       E1.C_Matrix(scount, En, C_Mat);
       eta=Gx[N]; zeta=Gy[N]; weight=w[N];
       Geo->Local_Cord(K, cordL);
       Rec[8]->Shape_deriv_Node(eta, zeta, N_deriv);
       Rec[8]->Shape_Node(eta, zeta, N_vec);
       ps1.Jacobian(cordL, N_detriv, weight, Djwt);
       //Djwt is the Determinant of Jacobian matrix
       Rec[8]->Cart_Gauss_Cord(cordL, N_vec, scount, Gcord);
       ps1.Cart_deriv_shape(N_deriv, N_vec);
       ps1.B_matrix(Gcord, Scount, B_mat);
       Str1->Disp_local(K);     Str1->Strain_Calcul(scount,a , B_Mat);
       Str1->Stress_Calcul(scount, a, C_Mat);        //a is the Int_order
       Str1->Check_stress_state(scount, &ust, &lst);
       if(ust>0.00000001) { if(lst<=-0.00000001) {
         cout<<"\n FROM ELASTIC TO PLASTIC \n";
         Str1->Update_stress(scount);     Str1->Effective_pl_strain(scount);
         Str1->Tstress_update(scount);    Str1->Elastic_moduli_update(scount, En); }
       else { if(fabs(lst)<=0.00000001) {      cout<<"\nPURE PLASTIC\n";
         goto step8; }
         else { cout<<"\nSTRESS STATE LIES OUTSIDE YIEL LOCUS"\n
         goto step8; }
step8:   Str1->Update_stress(scount);    St1->Effective pl_strain(scount);
         Str1->Tstress_update(scount);
         Str1->Elastic_moduli_update(scount, En); } }
      else { if(lst>=-0.00000001) {     cout<<"\nELASTIC UNLOADING\n";
         Str1->Elastic_Tstress_update(scount);     else { cout<<"\nPURE ELASTIC"\n
         Str1->Elastic_Tstress_update(scount);      Str1->Elastic_moduli_update(scount, En); }
         Str1->Load_R(scount, Gcord, B_mat, Djwt);
      scount++; }    Str1->assemble_load(K); }
      Str1->Residual_force();
}
```

Figure 8.14: Pseudo codes for function Stress_Strain()

```
void Element01 :: Global_Iter_NR()   //Global Ieration for NR method
 {
    for(INCRE=0; INCRE<NINCRE; INCRE++)
    {
         Iter=1;
         Str1->Load_Input(INCRE);
         ks=Toler_pres + 0.1;
         while(ks>Tolr_pres)
         {
              switch(IterType)
             {
              case 1:
                  Global_K();
              break;
              case 2:
                   if(Iter==1)
                      Global_K();
               break;
              }
              FE_Solver();
              Get_Stress_Strain();
              Solv->Convergence(&ks);
              Iter++;
         }
         for(K=0; K<Num_elm*Int_order*Int_order; K++)
         {
              Str1->Pcn_update(K);
              Str1->Elastic_moduli_update2(K, En);
         }
         Str1->Plot_Stress(INCRE);
    }
    Str1->Load_update();
    Str1->Print_Stress(INCRE);
    Str1->Nodal_stress();
 }
```

Figure 8.15: Pseudo codes for function Global_Iter_NR()

8.3 Summary

An object-oriented finite element framework for MCCM for analyzing geomechanics problems is developed using c++ as programming language. The integration algorithms presented for MCCM and the various nonlinear solution techniques and load stepping schemes are implemented in this object-oriented programming framework. The framework is developed for two dimensional problems, where plane stress, plane strain and axisymmetric analysis is implemented. The framework can be extended in many directions, such as, inclusion of new analysis type, new interpolation type, new material models, new integration algorithms for an existing models, inclusion of various load stepping schemes with an existing integration algorithm, etc.

Chapter 9

Numerical examples

9.1 Introduction

Numerical examples used to validate the proposed integration algorithms include (i) one element, and eight elements triaxial tests in both drained and undrained conditions, and (ii) plane strain drained analysis of strip footings using modified Cam clay model (MCCM). The new integration algorithm and Runge-Kutta method for MCCM are implemented in the object-oriented finite element framework discussed in the Chapter 7. Various iteration schemes and load stepping schemes explained in Chapter 5 are implemented for the MCCM with the new integration algorithm. Iterations are terminated according to the residual force convergence criterion for force control method and residual displacement criterion for arclength method. They are given as follows:

(i) Residual force convergence criterion

$$\|\boldsymbol{r}^k\| \leq TOL\|\boldsymbol{r}^0\| \tag{9.1.1}$$

where $\boldsymbol{r}^k = (\boldsymbol{F}_{ext})_{n+1} - \boldsymbol{F}_{int}(\boldsymbol{\sigma}^k)$, $\boldsymbol{F}_{ext}$ is the external force vector, $\boldsymbol{F}_{int}$ is the internal force vector, which is a function of the internal stress at the end of $(n+1)$th load step, $\|.\|$ denotes an L_2-vector norm, TOL is the force threshold value set at $TOL = 10^{-6}$, and k is the iteration counter.

(ii) Residual displacement convergence criterion

$$\|\boldsymbol{u}^k\| \leq TOL\|\boldsymbol{u}^1\| \tag{9.1.2}$$

where $\boldsymbol{u}^k$ is the iterative displacement vector, $\boldsymbol{u}^1$ is the iterative displacement vector for $k = 1$, k is the iteration counter, $\|.\|$ denotes an L_2-vector norm, TOL is the displacement threshold value set at $TOL = 10^{-6}$.

9.2 Triaxial tests

Numerical examples, which investigate the integration algorithm at local as well as global levels consist of uncoupled elasto-plastic analysis to simulate drained and undrained triaxial compression tests. To investigate the local algorithm, a quarter of the cylindrical specimen of 0.5 unit in diameter and 1.0 unit in length is used as a single, eight nodded, isoparametric, quadrilateral element, which is shown in Figure 9.1. To validate the integration algorithm at its global level the same cylindrical specimen is discretized into 4 numbers, 8 numbers and 16 numbers of eight nodded isoparametric elements, which are shown in the Figure 9.2, Figure 9.3 and Figure 9.4. For all of these cases two types of initial conditions are considered. In the first initial condition, the soil is considered to be lightly over-consolidated with an OCR of 1.2, which can be designated as normally consolidated clay (NCC). The second initial condition represents a heavily over-consolidated clay (OCC) with an OCR of 6. It is found that the results of the triaxial tests with discretization of 4 number of elements, 8 number of elements and 16 number of elements are identical in all respects. Therefore, the results of only one discretization, ie. of the 8 number of elements are presented in the report.

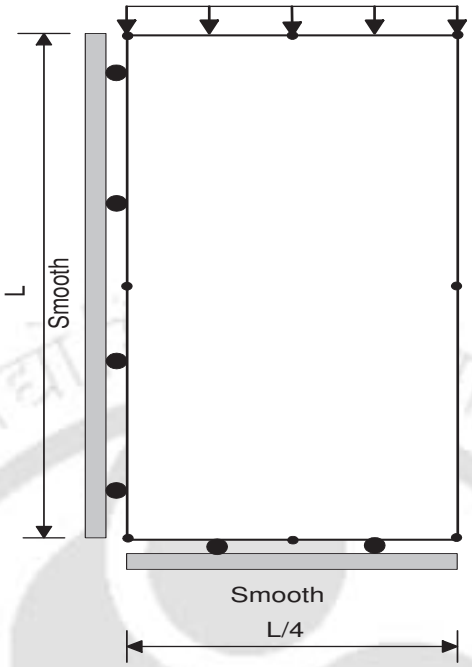


Figure 9.1: Axisymmetric finite element mesh for one element (eight nodded isoparametric element)

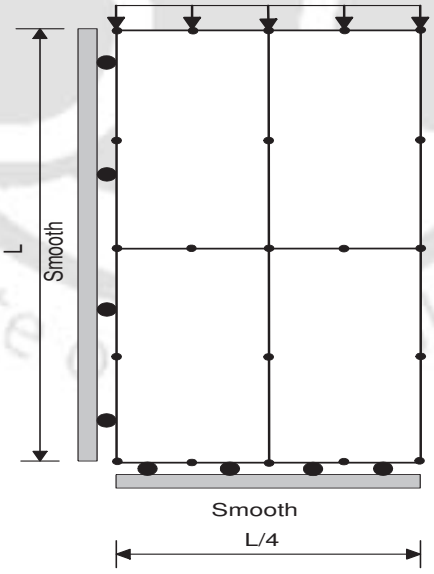


Figure 9.2: Axisymmetric finite element mesh for 4 elements (eight nodded isoparametric elements)

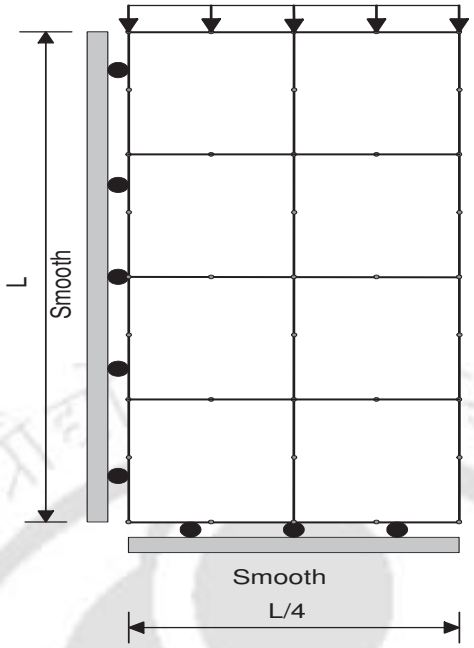


Figure 9.3: Axisymmetric finite element mesh for 8 elements (eight nodded isoparametric elements)

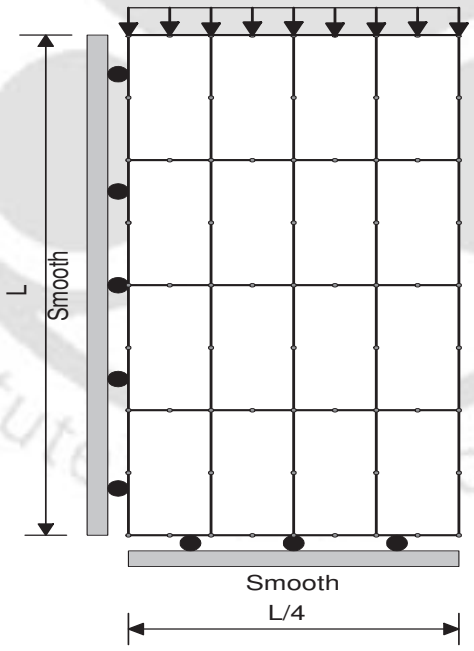


Figure 9.4: Axisymmetric finite element mesh for 16 elements (eight nodded isoparametric elements)

9.2.1 Sample preparation for triaxial tests

The sample for triaxial tests is prepared by isotropic normal consolidation to $p = p_c$, followed by swelling to $p = p_o$ as shown in Figure 9.5. For both NC and OC clay the soil is first consolidated to p_c=60 kN/m^2. This is then followed by swelling to p_o=50 kN/m^2 to give OCR=1.2 for NC clay and p_o=10 kN/m^2 to give OCR=6.0 for OC clay. The initial radial stress ($\boldsymbol{\sigma}_{ro}$), axial stress ($\boldsymbol{\sigma}_{ao}$), and void ratio (e_0) for these conditions are as follows:

$$\text{For NCC:} \quad \boldsymbol{\sigma}_{ro} = \boldsymbol{\sigma}_{ao} = 50\ kN/m^2,\ e_0 = 1.50$$

$$\text{For OCC:} \quad \boldsymbol{\sigma}_{ro} = \boldsymbol{\sigma}_{ao} = 10\ kN/m^2,\ e_0 = 1.53$$

The MCCM parameters considered in both the cases are (Sheng et al. [146])

$$\kappa = 0.02,\ \lambda = 0.2,\ M = 1.2, \text{ and } \mu = 0.3$$

For an undrained analysis, the bulk modulus of water K_w is required. In the analysis presented here K_w is taken as $65K'$ for both NC and OC clay.

9.2.2 Drained compression tests

During the drained compression test, the radial stress $\boldsymbol{\sigma}_r$ is kept constant while the axial stress $\boldsymbol{\sigma}_a$ is increased by applying prescribed axial loads in convenient steps, as axial strain is applied in a conventional triaxial compression test. In this study it is assumed that the pore pressure is maintained at a back pressure of zero (i.e. atmospheric). Thus the effective stress paths (ESP) always corresponds to the total stress paths (TSP) and hence the effective stress can be determined by considering the total stress acting on the soil sample. Figure 9.6 shows the typical (p , q) plot and (p , v) plot for a conventional drained triaxial test on a normally consolidated or lightly over consolidated clay. In this case the initial stress state of the sample will lie in the right side of the critical state line in the (p , v) plot. During the initial part of the test, before the EPS intersects the current yield locus at point B, the soil behaviour is elastic. After reaching point B, the soil starts yielding and any point on BF is associated with a new yield locus. In the p , v plot, the soil follows the swelling line while its behaviour is elastic and then changes its direction to meet

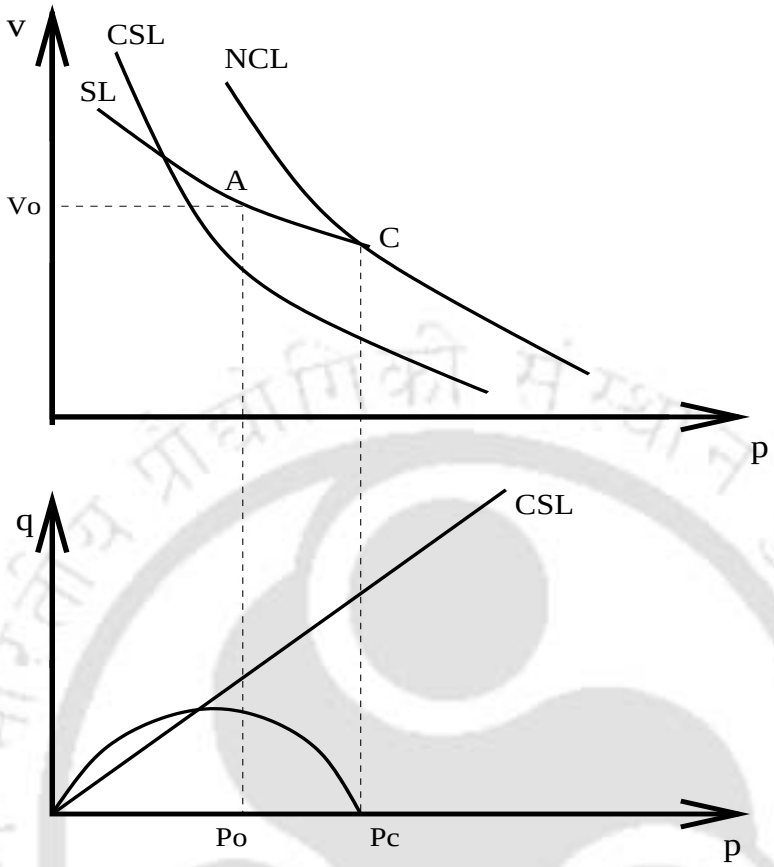


Figure 9.5: Preparing the sample for Triaxial tests [32]

the CSL at failure.

In case of highly over-consolidated (OC) clay, the initial stress state of the sample will lie in the left side of the critical state line in a (p , v) plot. Figure 9.7 shows the progress in a drained compression test, which is typical for an OC clay. The sample follows the effective stress path AB before it starts yielding. Then it moves back down the EPS to point F on the critical state line after yielding, which is accompanied by the shrinking of the initial yield locus.

The typical stress-strain response of these two types of soils (NC clay and OC clay) are shown in the Figure 9.8. It is clear from the figure that the NC clay undergoes strain-hardening after yielding (q increasing with increased p) and exhibits compressive plastic volumetric strains. On the other hand, the OC clay strain-softens (q decreasing with increased p) and exhibits expansive volumetric strains.

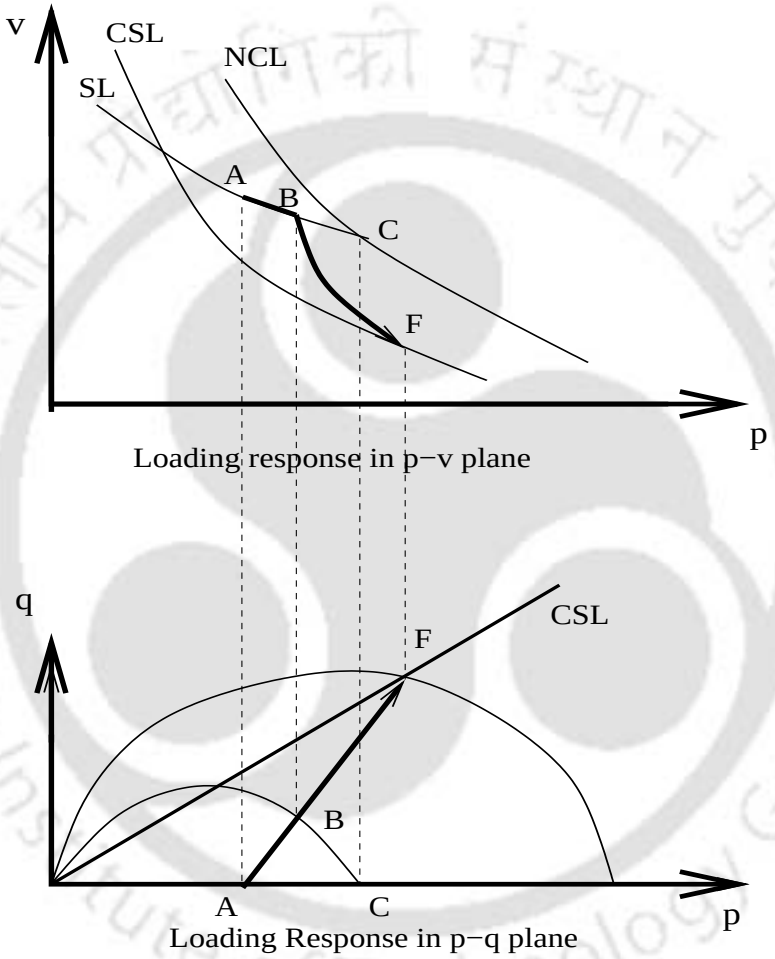


Figure 9.6: Triaxial drained compression test for NC clay [32]

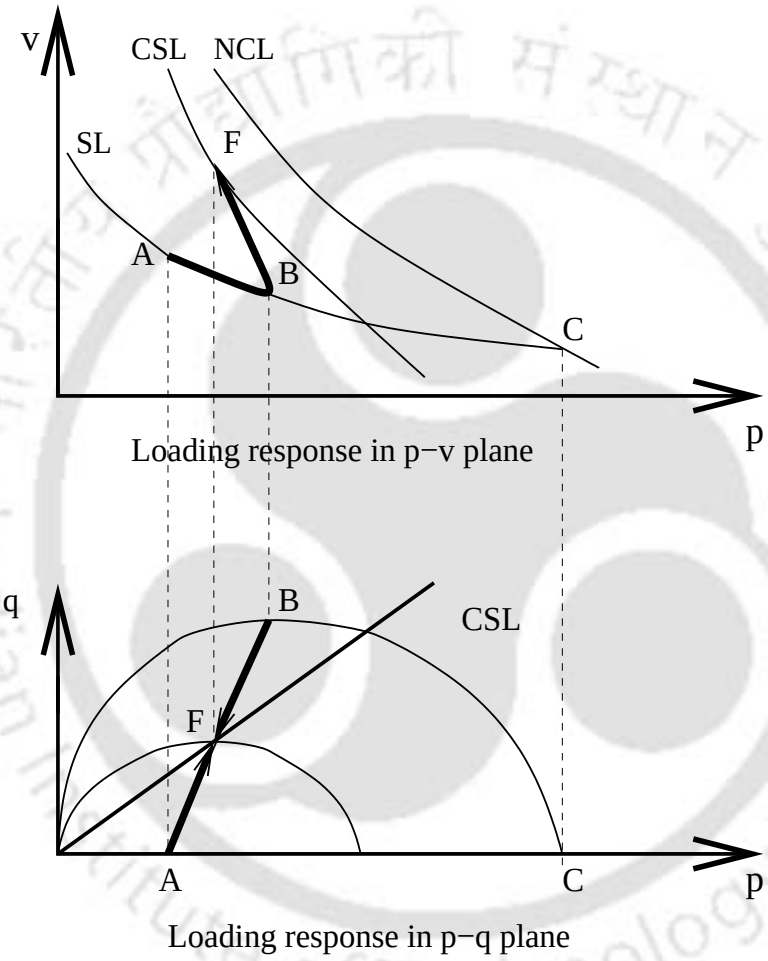


Figure 9.7: Triaxial drained compression test for OC clay [32]

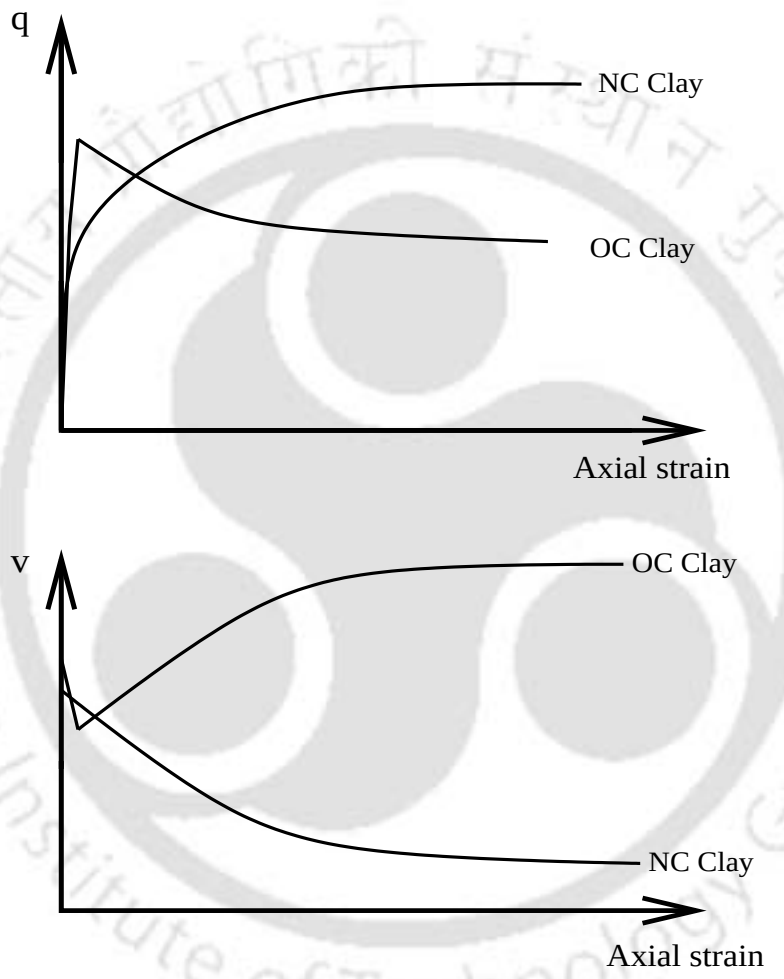


Figure 9.8: Stress-strain response for triaxial drained compression test [32]

9.2.2.1 Results from drained compression tests

The simulated results for uncoupled elasto-plastic drained triaxial compression tests (both one element and eight elements tests) using MCCM with the new implicit integration algorithm are presented here. The test results presented to show the loading response and stress-strain response of the sample include: (a) Axial strain vs deviator stress (q); (b) Effective mean stress vs deviator stress; (c) Axial strain vs specific volume (v); (d) Effective mean stress vs specific volume.

Figure 9.9 through Figure 9.12 show the loading response and stress-strain response for both NC and OC clay, where arclength method is used as iterative scheme. The results have been compared with the ones reported in Sheng et al. [146], and thus the results are typical for NCC and OCC subjected to elasto-plastic analysis under drained conditions. The effective stress paths depicted by the program moves towards the critical state line (CSL) and the specimen fails when it reaches the CSL, as it should be for a soil sample tested under triaxial testing condition.

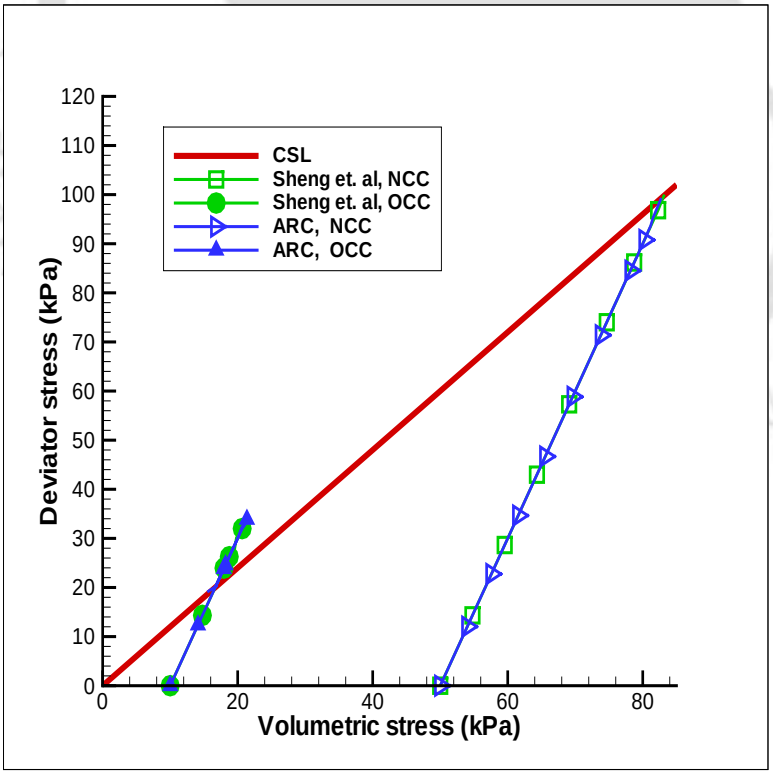


Figure 9.9: Comparison of drained triaxial tests results by arclength method with the results of Sheng et al. [147]: Volumetric stress (p) vs Deviator stress (q)

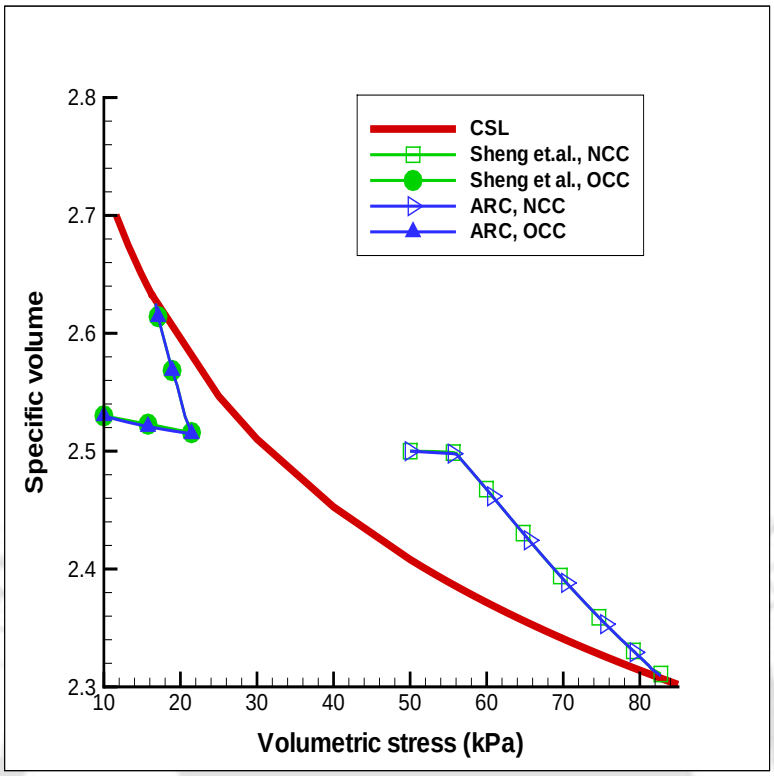


Figure 9.10: Comparison of drained triaxial tests results by arclength method with the results of Sheng et al. [147]: Volumetric stress (p) vs Specific volume (v)

Figure 9.13 through Figure 9.16 show the results of drained triaxial tests using other iterative schemes, which include Newton-Raphson method, modified Newton-Raphson method (with Chen accelerator and modified Thomas accelerator), and Automatic iteration method. Here also MCCM with the new implicit integration algorithm is used. The results obtained with Runge-Kutta method are also shown in the same figures. The results with Automatic incrementation method is not shown as there is considerable deviation of the results with the others.

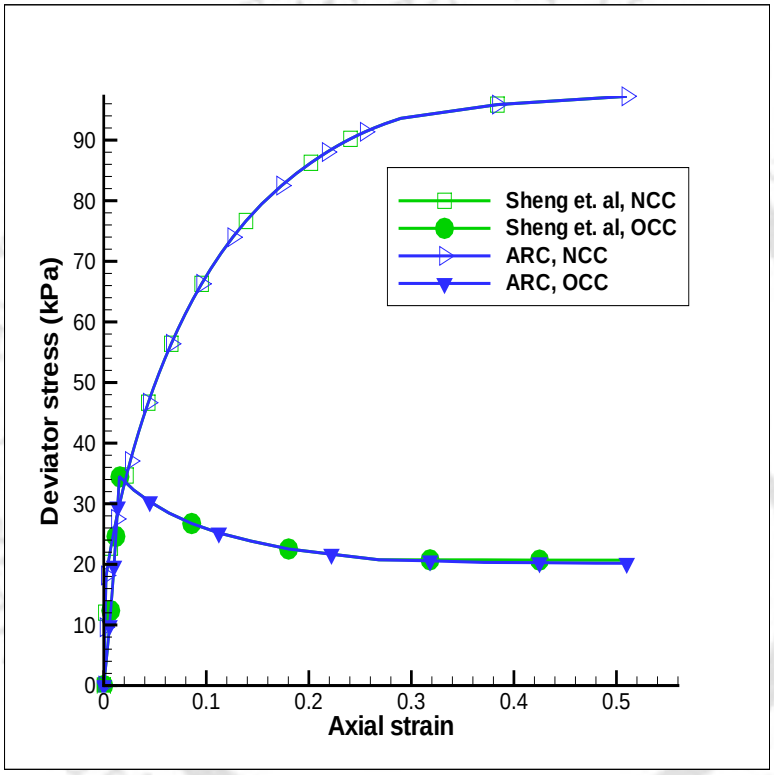


Figure 9.11: Comparison of drained triaxial tests results by arclength method with the results of Sheng et al. [146]: Axial strain (ε_a) vs Deviator stress(q)

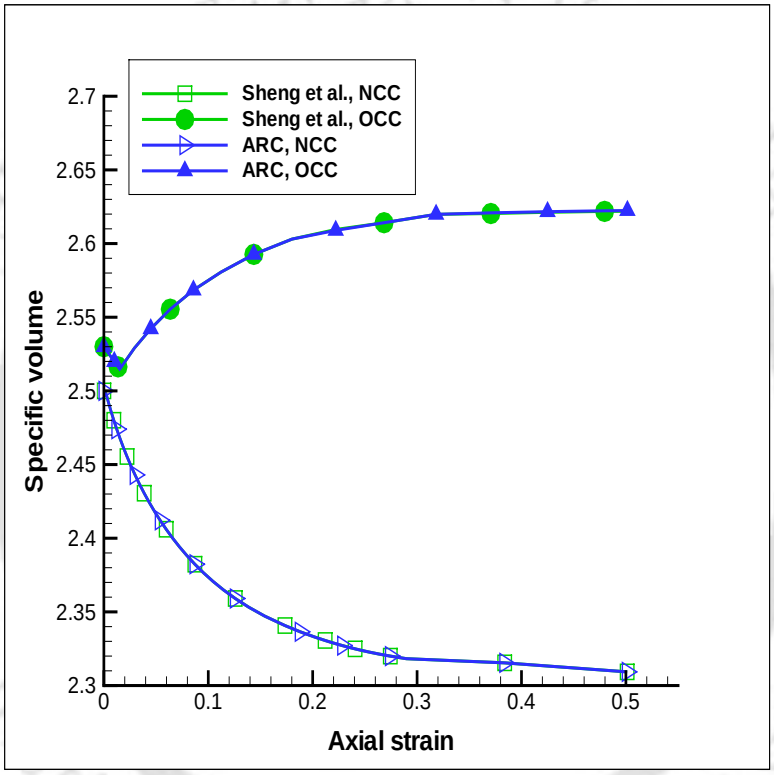


Figure 9.12: Comparison of drained triaxial tests results by arclength method with the results of Sheng et al. [146]: Axial strain (ε_a) vs Specific volume (v)

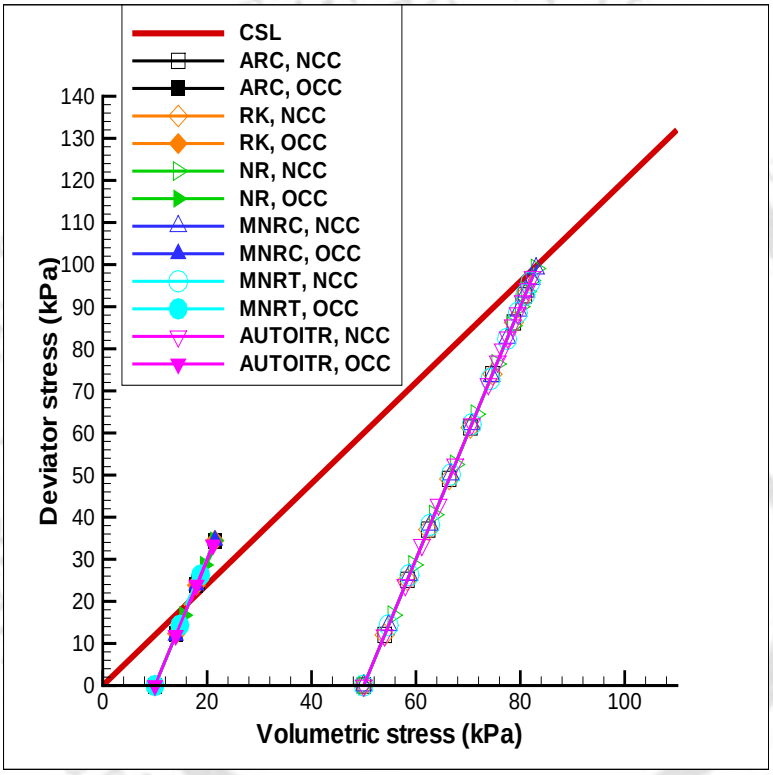


Figure 9.13: Comparison of drained triaxial tests results by other methods(Drained): Volumetric stress (p) vs Deviator stress (q)

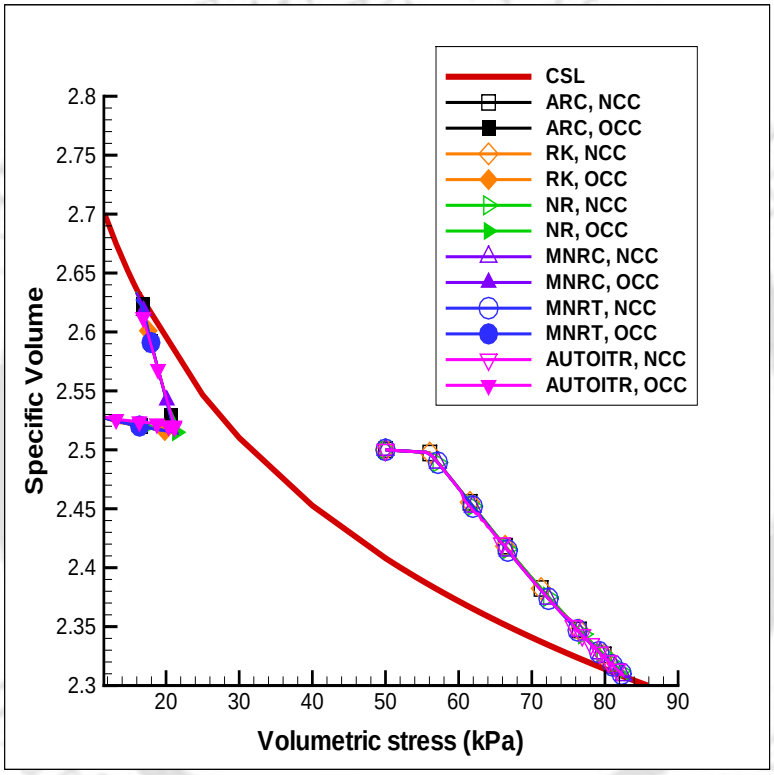


Figure 9.14: Comparison of results of drained triaxial tests by other methods(Drained): Volumetric stress (p) vs Specific volume (v)

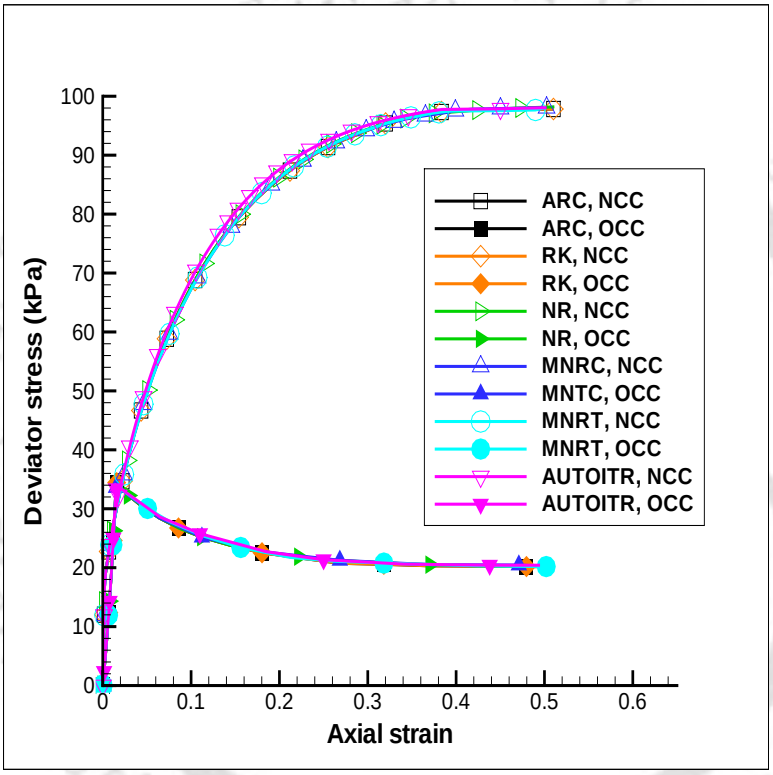


Figure 9.15: Comparison of drained triaxial tests results by other methods(Drained): Axial strain (ε_a) vs Deviator stress(q)

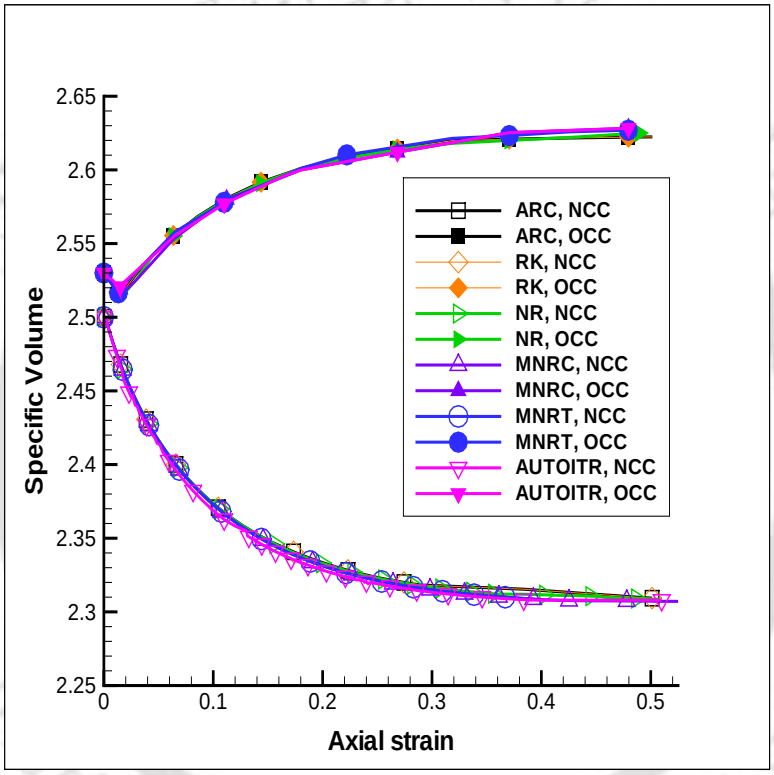


Figure 9.16: Comparison of drained triaxial tests results by other methods(Drained): Axial strain (ε_a) vs Specific volume (v)

9.2.3 Undrained compression tests

The undrained compression test is carried out in the same way like drained compression test, except that no water is allowed to flow into or out of the sample during this test. Therefore, during the whole of the undrained test, the specific volume must remain constant and hence the total volumetric strain must be zero. Before the sample starts yielding, the plastic volumetric strain is zero. Therefore the elastic volumetric strain also must be zero. Hence the volumetric stress p must remain constant before the sample yields. Figure 9.17 and Figure 9.18 show the loading response of NC and OC clay under undrained triaxial compression tests. In case of NC clay the yield surface expands and hence the soil undergoes strain-hardening after yielding. In case of OC clay, the sample undergoes strain-softening with contraction of the yield locus. It is also clear from the figures that, in case of NC clay q increases before failure, while in OC clay q decreases or remains constant.

The stress-strain response of both NC and OC clay are shown in Figure 9.19. For both the soils pore water pressure increases linearly in the initial elastic part of the tests. During yielding of the sample, the NC clay generates positive pore water pressure, while the OC clay generates negative pore water pressure.

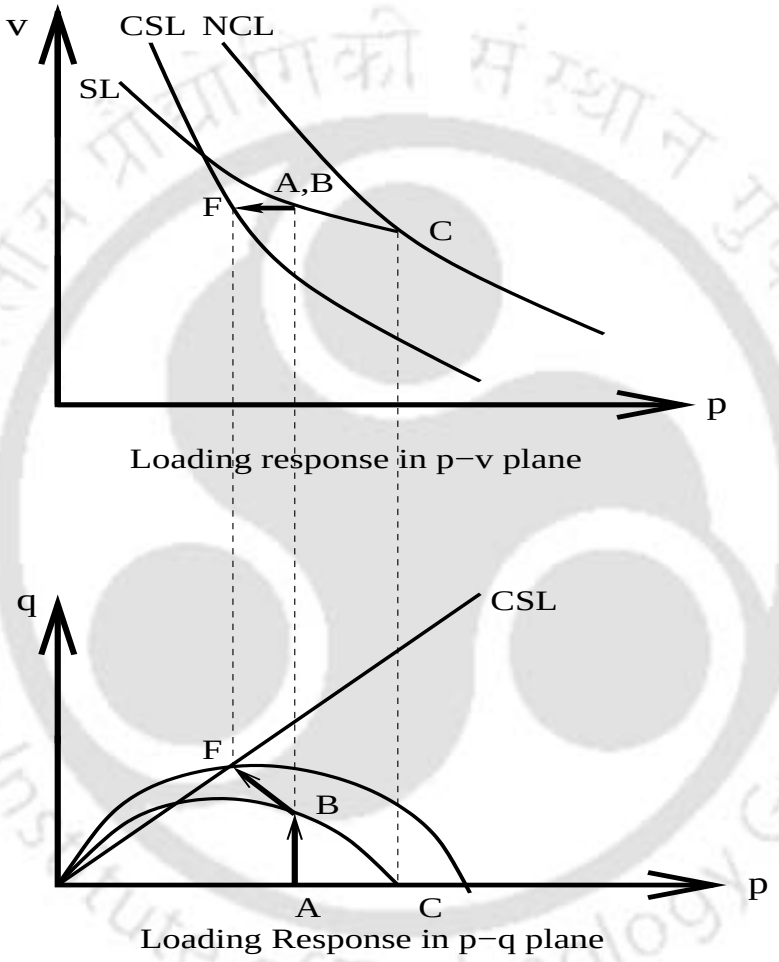


Figure 9.17: Triaxial undrained compression test for NC clay [32]

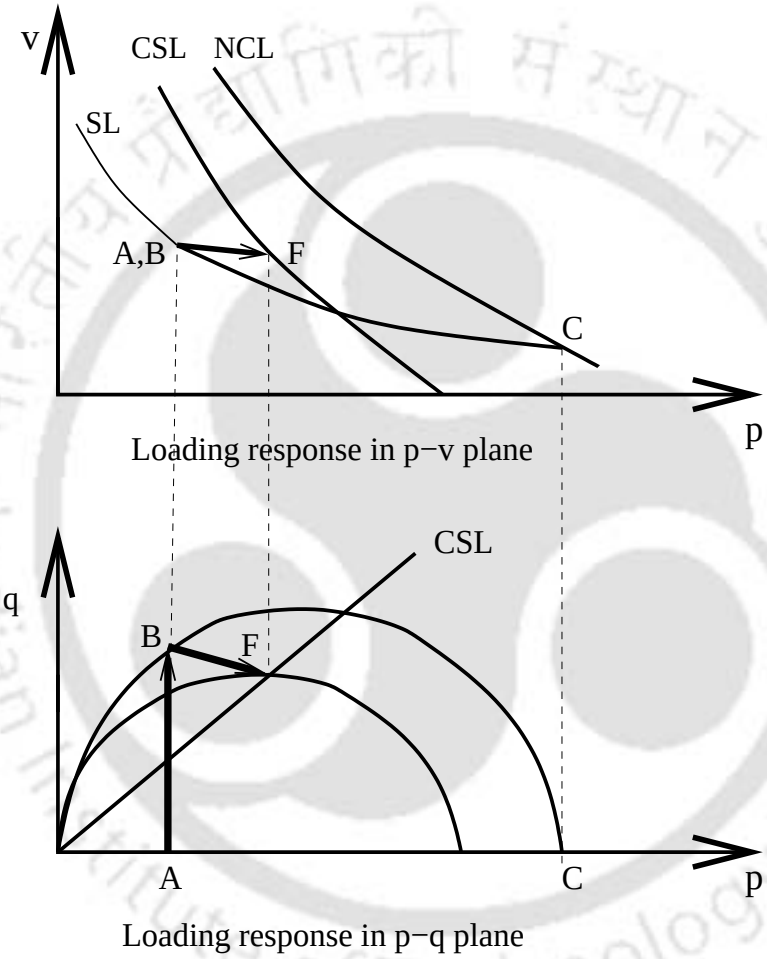


Figure 9.18: Triaxial undrained compression test for OC clay [32]

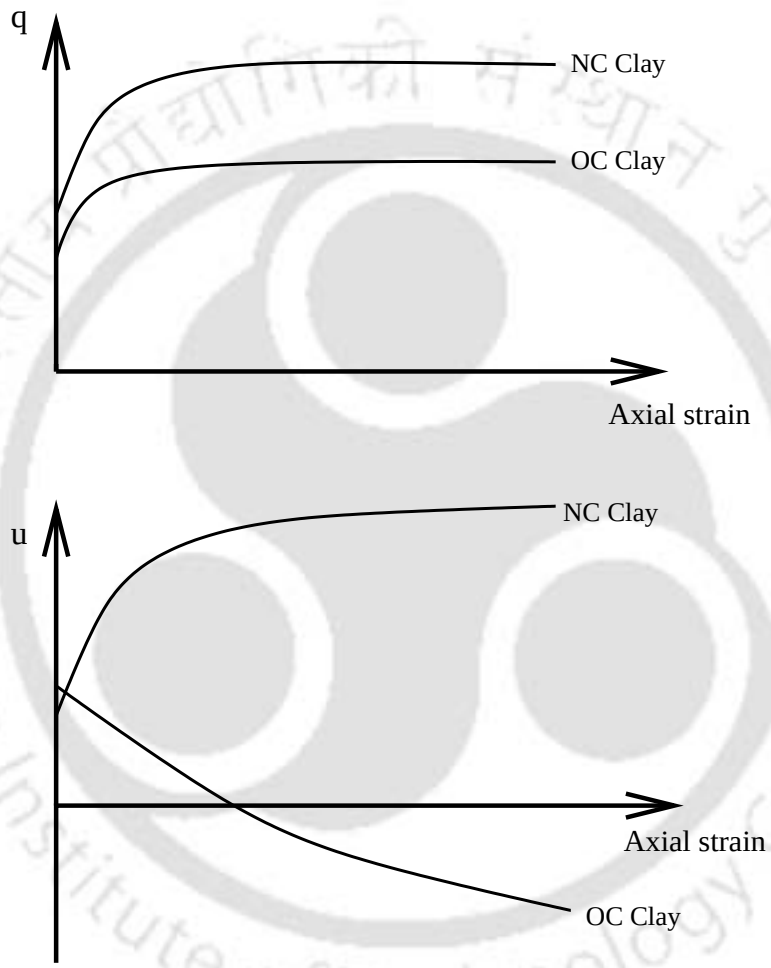


Figure 9.19: Stress-strain response for Triaxial undrained compression test [32]

9.2.3.1 Results from undrained compression tests

The simulated results for uncoupled elasto-plastic undrained triaxial tests (both one element and eight elements tests) using MCCM with the new implicit integration algorithm are presented here. The Figure 9.20 through Figure 9.23 show the loading response and stress-strain response for both NC and OC clay, where arclength method is used as iterative scheme. The results from undrained tests also have been compared with the ones reported in Sheng et al. [146], and thus the results are typical for NCC and OCC subjected to elasto-plastic analysis under undrained conditions. The effective stress paths depicted by the program moves towards the critical state line (CSL) and the specimen fails when it reaches the CSL, as it should be for a soil sample tested under undrained triaxial testing condition. Figure 9.24 through Figure 9.28 show the results of undrained triaxial tests using all other iterative schemes, which include Newton-Raphson method, modified Newton-Raphson method (with Chen accelerator and modified Thomas accelerator), and Automatic iteration method. The results with Runge-Kutta method is also shown in the same figures.

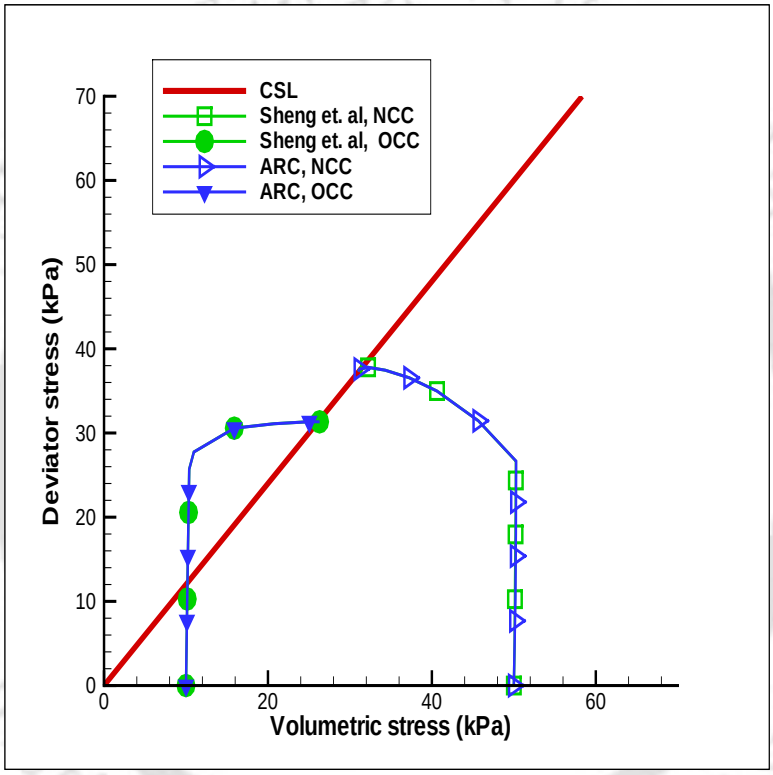


Figure 9.20: Comparison of undrained triaxial tests results by arclength method with the results of Sheng et al. [146]: Volumetric stress (p) vs Deviator stress (q)

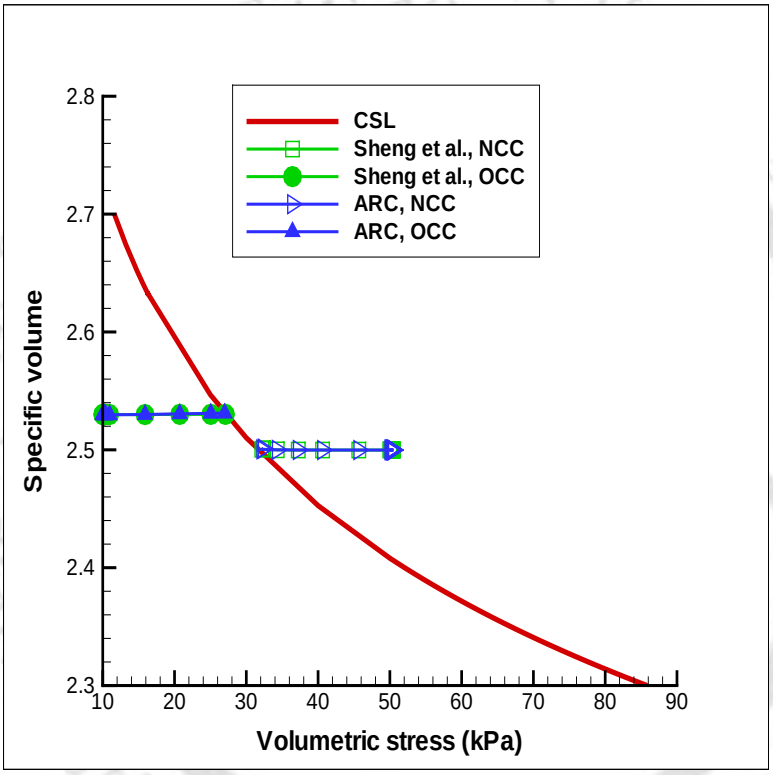


Figure 9.21: Comparison of undrained triaxial tests results by arclength method with the results of Sheng et al. [146]: Volumetric stress (p) vs Specific volume (v)

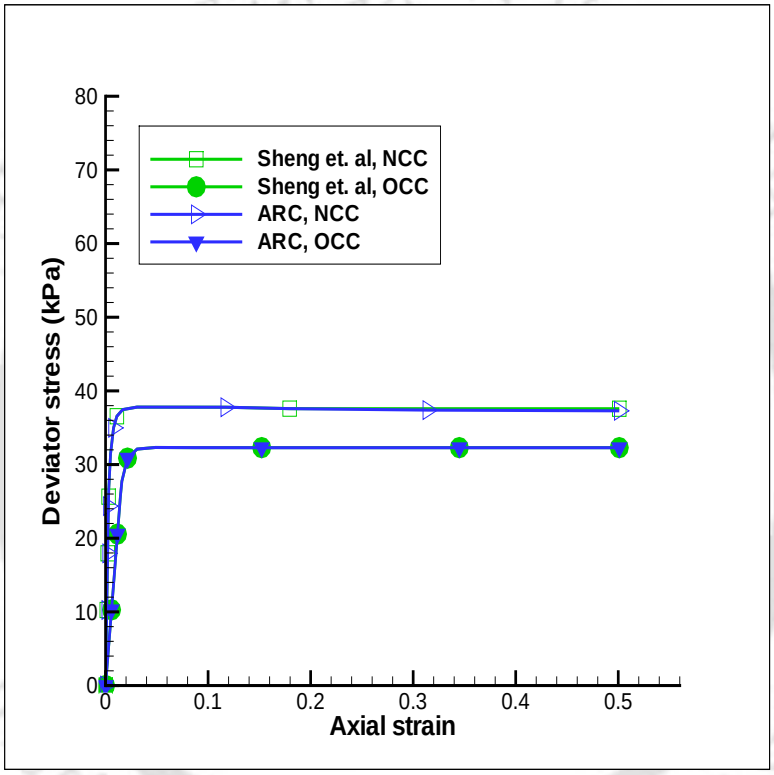


Figure 9.22: Comparison of undrained triaxial tests results by arclength method with the results of Sheng et al. [146]: Axial strain (ε_a) vs Deviator stress(q)

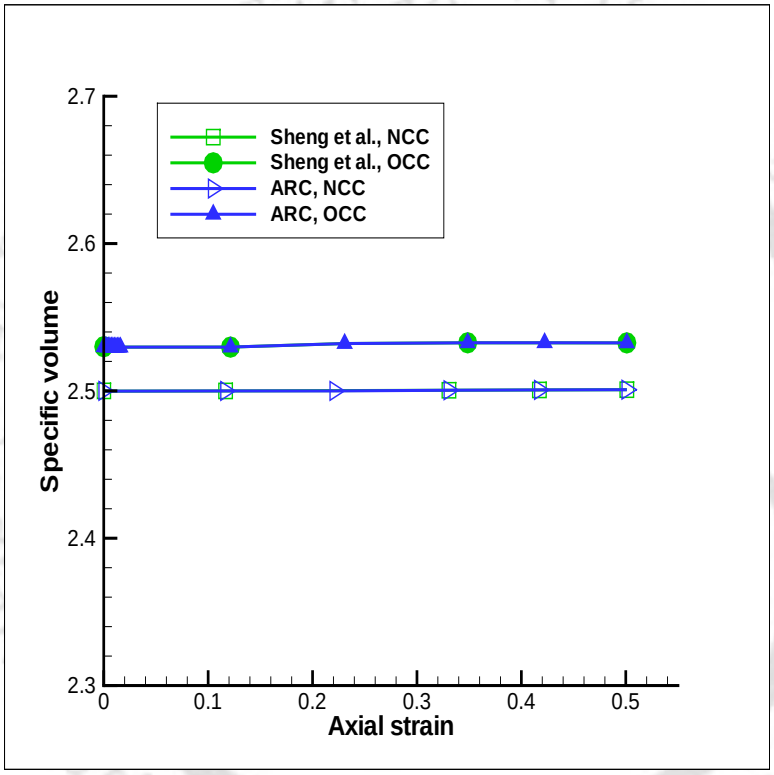


Figure 9.23: Comparison of undrained triaxial tests results by arclength method with the results of Sheng et al. [146]: Axial strain (ε_a) vs Specific volume (v)

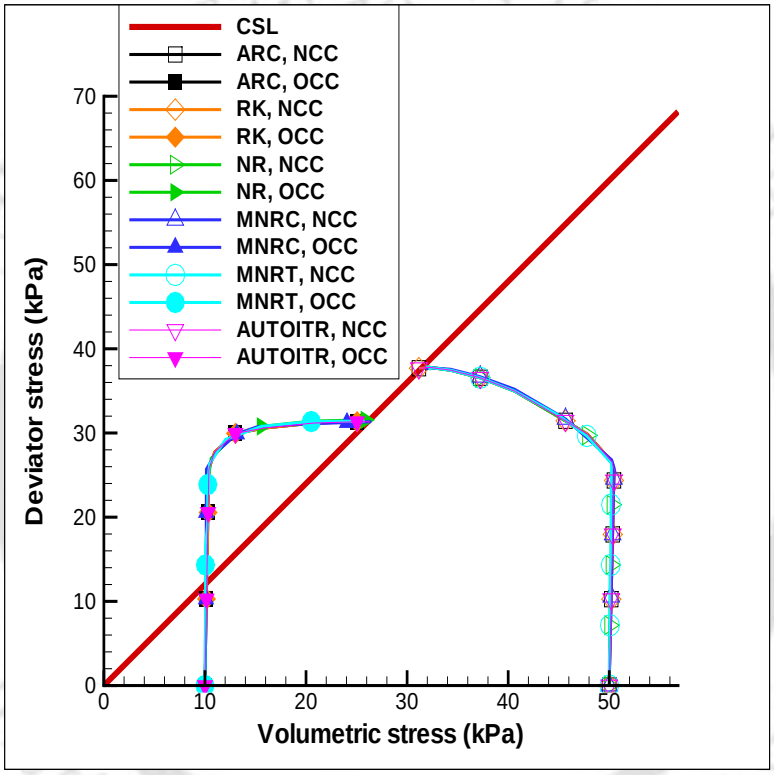


Figure 9.24: Comparison of undrained triaxial tests results by other methods(Undrained): Volumetric stress (p) vs Deviator stress (q)

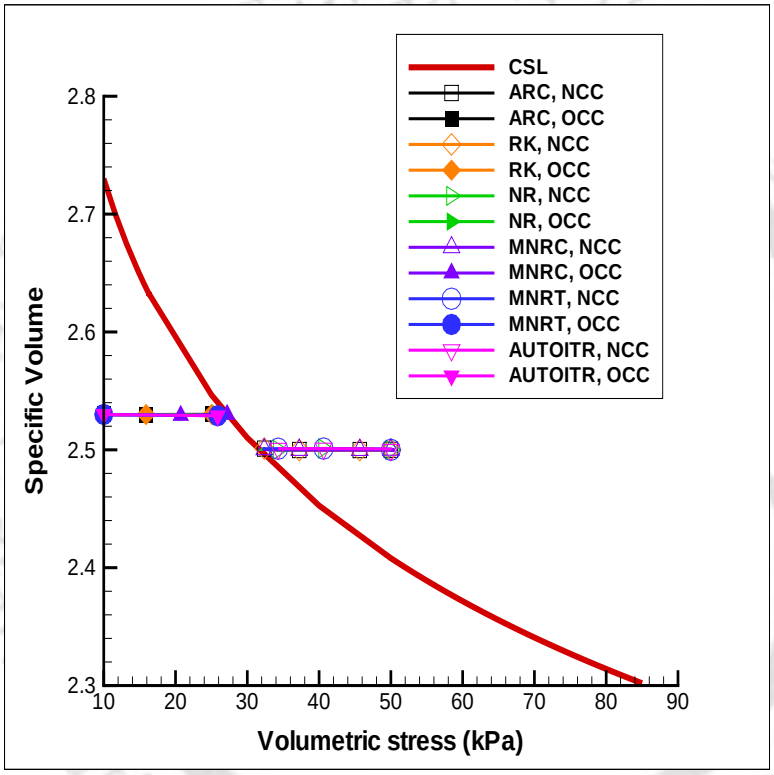


Figure 9.25: Comparison of undrained triaxial tests results by other methods(Undrained): Volumetric stress (p) vs Specific volume (v)

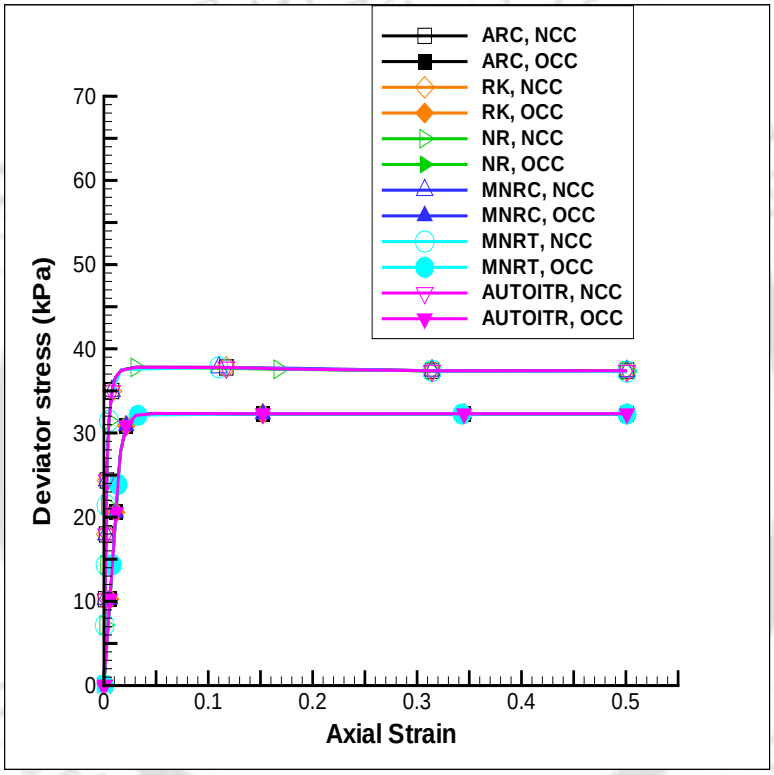


Figure 9.26: Comparison of undrained triaxial tests results by other methods(Undrained): Axial strain (ε_a) vs Deviator stress(q)

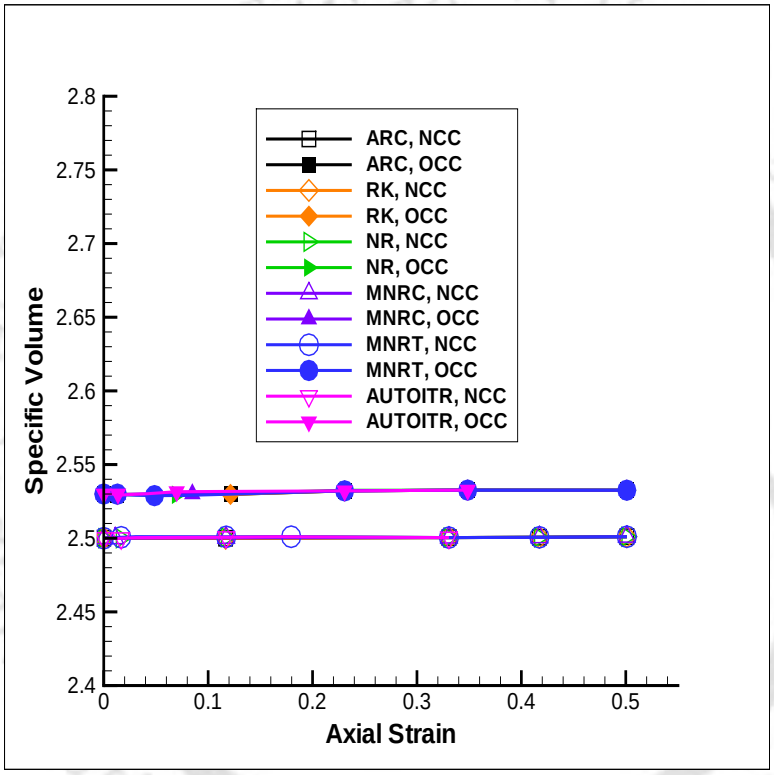


Figure 9.27: Comparison of undrained triaxial tests results by other methods(Undrained): Axial strain(ε_a) vs Specific volume (v)

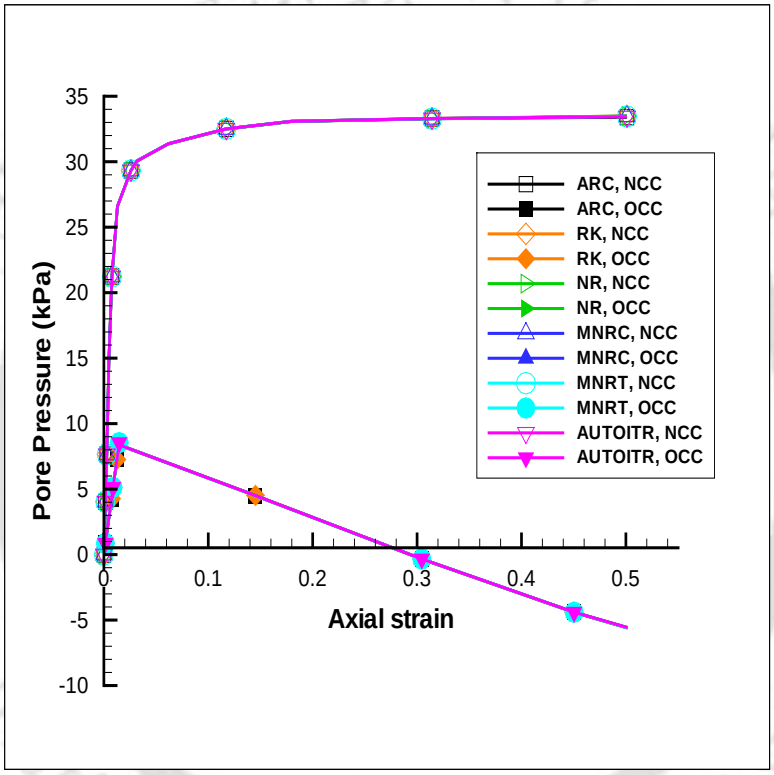


Figure 9.28: Comparison of undrained triaxial tests results by other methods(Undrained): Axial strain(ε_a) vs Pore water pressure (σ_w)

9.2.4 Summary of triaxial compression tests results

The results obtained by various iterative schemes with the two integration algorithms for MCCM are presented in the previous sections. There is good agreement of these results with the one reported by Sheng et al. [146]. It is also observed that the results are identical for one element and eight elements test, as expected. In case of both drained and undrained tests, it is clear from the figures that the NC clay undergoes strain-hardening after yielding and exhibits compressive plastic volumetric strains. On the other hand, the OC clay strain-softens and exhibits expansive volumetric strains. These types of behaviour are typical for NC and OC clay and established from experimental observations. This demonstrates the stability and accuracy of the proposed integration algorithms and the proposed object-oriented finite element framework.

The different iterative methods implemented here are compared in terms of the number of iterations required to perform different triaxial tests. Table 9.1 shows the total number of iterations required by different methods for different drainage conditions and soil type for triaxial compression tests. It is observed from the table that the number of iterations required by arclength method (ARC) is less as compared to all other methods. Again, the Automatic iteration method ($AUTOITR$) is taking the highest number of iterations, where Runge-Kutta method is in the second highest position. Therefore, arclength method can be considered to be more efficient than all other methods without loss of accuracy and stability.

Table 9.1: Total number of iterations required by different methods for different drainage conditions and soil type for triaxial tests.

Soil Type	Test Type	Element No.	Methods					
			ARC	NR	MNRC	MNRT	AUTOITR	RK
NCC	CD	1	51	215	220	193	245	225
		8	26	134	189	103	198	134
OCC	CD	1	25	75	76	76	83	77
		8	19	27	31	23	34	32
NCC	CU	1	50	225	242	243	254	245
		8	45	162	165	172	191	176
OCC	CU	1	74	242	256	264	346	328
		8	68	214	268	263	341	316

9.3 Boundary value problems

The second numerical examples focus on the performance of the overall implementation technique into the nonlinear finite element codes. To allow a direct comparison, first an application problem is presented, which is identical to that shown by Sheng et al.[146]. In this example, we consider the plane strain analysis of a vertically loaded rigid strip footing of width B on a 'finite' deposit of clay. The loading in the footing is assumed to be kept for sufficient duration, which corresponds to a drained analysis and hence no excess hydrostatic pressure developed during each increment of loading. The finite element mesh used in the analysis is shown in Figure 9.29, where half width of the footing ($B/2$) is kept as 1 unit. The domain of the foundation soil ($5B$ x $5B$) is divided into 225 eight nodded, isoparametric, quadrilateral elements with a 2 x 2 gaussian integration rule employed for each element. The soil profile is assumed to be underlain by rigid and rough bedrock and so restrained in both the directions at the bottom, while left and right boundaries are restrained in horizontal direction only. The load is applied vertically in 10 and 50 equal steps, but a converged solution was not achieved with any of the methods with 10 number of load steps.

The second boundary value problem consists of study of the effect of the distance between two identical strip footings on its bearing capacity. In this analysis also plain strain condition is considered. For this purpose two strip footings of same width $'B'$ are placed at different distances apart on an uniform bed of clay having the same properties as the first problem. The finite element meshes considered for this study are shown in the Figure 9.35 where half width of both the footings (B/2) is kept as 1 unit. The domain of the foundation soil bounded by the footings is divided into 450 eight nodded, isoparametric, quadrilateral elements with a 2 x 2 gaussian integration rule employed for each element. Other relevant soil conditions and soil properties are kept same as the first boundary value problem. The load is applied vertically in 10 and 50 equal steps, but a converged solution was not achieved with any of the methods with 10 number of load steps for this problem too.

9.3.1 Soil properties and initial data

9.3.1.1 Soil properties

The simulated soil corresponds to one considered by Sheng et al. [146]. The soil is assumed to be heavily over-consolidated at the ground surface and nearly normally consolidated at greater depth. The MCCM parameters for this soil are listed below. The soil is considered to be fully saturated and the unit weight of the soil listed below corresponds to the submerged unit weight.

$$M = 0.898, \quad \lambda = 0.25, \quad \kappa = 0.05, \quad e_{cr} = \Gamma - 1 = 1.6, \quad \mu = 0.35, \quad \gamma{=}6\ kN/m^3$$

The specific volume, v_o $(= 1 + e_o)$, at the start of the test can be given from the following equation

$$v_o = N - \lambda\ ln(p_c) + \kappa\ ln(p_c/p_o)$$

where $N = \Gamma + (\lambda - \kappa)\ ln(2)$ is the specific volume of the soil corresponding to unit isotropic normal consolidation pressure p_c. N is constant for a particular soil, but depends on the unit used to measure the pressure. In this study unit adopted for pressure is kN/m^2 [32].

9.3.1.2 Initial stresses

For the plane strain analysis of strip footing, initial stresses were generated directly at each Gauss point using the following formula

$$\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_v = \gamma h \tag{9.3.3}$$

$$\boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_h = k_0 \boldsymbol{\sigma}_v \tag{9.3.4}$$

where γ is the unit weight of soil mass, h is the depth of the Gauss point considered from the ground surface, and K_o is the Co-efficient of earth pressure at rest given as $K_o \simeq 1 - \sin\Phi$. Here Φ is the angle of internal friction of soil and is calculated from the equation $M = \frac{6\sin\Phi}{3-\sin\Phi}$. With these initial values of input stresses, the initial Elastic Moduli K and G are to be calculated.

9.3.1.3 Over consolidation ratio (OCR)

The over-consolidation ratio (OCR) at any Gauss point of the soil mass under consideration is defined as OCR=p_c/p_o, where p_c is the past consolidation pressure representing the current size of the yield locus at the Gauss point in question and p_o is the equivalent consolidation pressure corresponding to the current state of stress (p, q), at this same point [24], i.e., $p_o = p + q^2/(M^2 p)$.

To get a realistic soil layer, the elements are preloaded with a uniform surcharge of 50 kPa at the ground surface in one step, and subsequently removed, thereby establishing a soil profile that is heavily over-consolidated at top and nearly normally consolidated at greater depth.

9.3.2 Analysis of results of single strip footing

The load-settlement curves of this problem are plotted for the node lying directly below the footing at its centre line. The plotted load is the total vertical load applied in kN/m. The plotted settlement is the settlement per unit width of the footing. The following Figure 9.31 compares the results obtained by different load stepping and iterative schemes applied to MCCM with the proposed integration algorithms, with that of Sheng et al. [146].

Table 9.2 shows the total number of iteration and time required by different methods for the footing analysis. The same is shown graphically in Figure 9.32. The accuracy of different methods are compared by plotting the relative errors of displacements at the centre line of the footing during the loading history of the footing. This is shown graphically in Figure 9.33. Relative error in Figure 9.33 is calculated using the following formula

$$Er = \frac{\|R - Rr\|}{Rr} \times 100$$

Where R is the displacement from the present study, Rr is the reference displacement which is taken as the displacement from Sheng et al. [146].

Figure 9.32 shows that the relative error in the case of Runge-Kutta method is less as compared to other methods. Hence a more refined mesh containing 484 elements

Table 9.2: Total number of Iterations and time required by different methods for Bearing capacity analysis of a rigid strip footing

Problem type	Scheme type	Total no of iteration	Time (sec)
Plane strain	ARC	75	4.475
Plane strain	NR	262	20.345
Plane strain	MNRC	191	17.758
Plane strain	MNRT	137	12.5
Plane strain	AUTOITR	306	26.97
Plane strain	RK	300	23.451
Plane strain	RK, mesh2	500	39.085

is used with Runge-Kutta method and shown in the Figure 9.30 (named as mesh2 or new mesh in the figures). It is found that the relative error in the case of this refined mesh is the least in the initial portion of the loading as well at the failure load. Figure 9.34 shows the initial and final OCR throughout the foundation soil of the footing for the Runge-Kutta method with the new mesh..

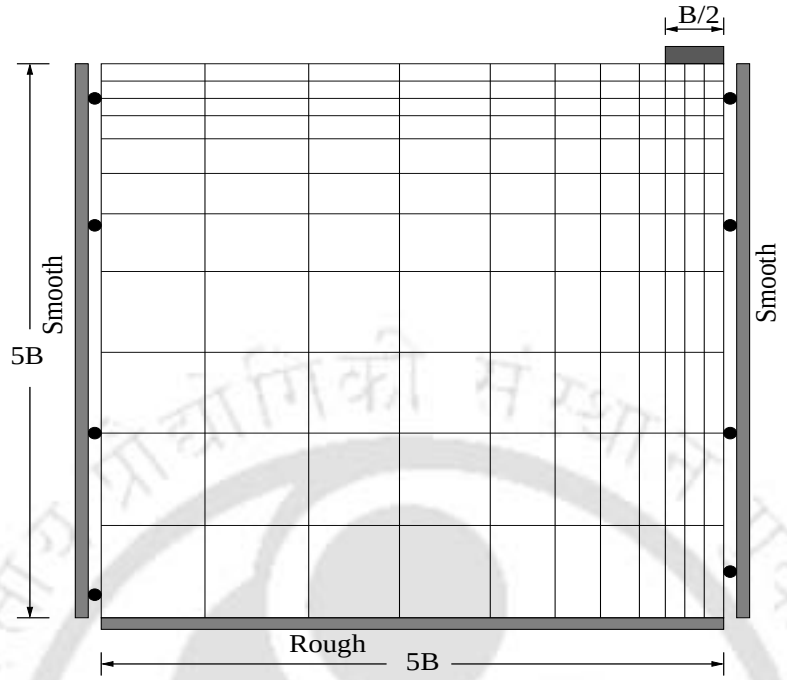


Figure 9.29: Finite element mesh for single footing analysis

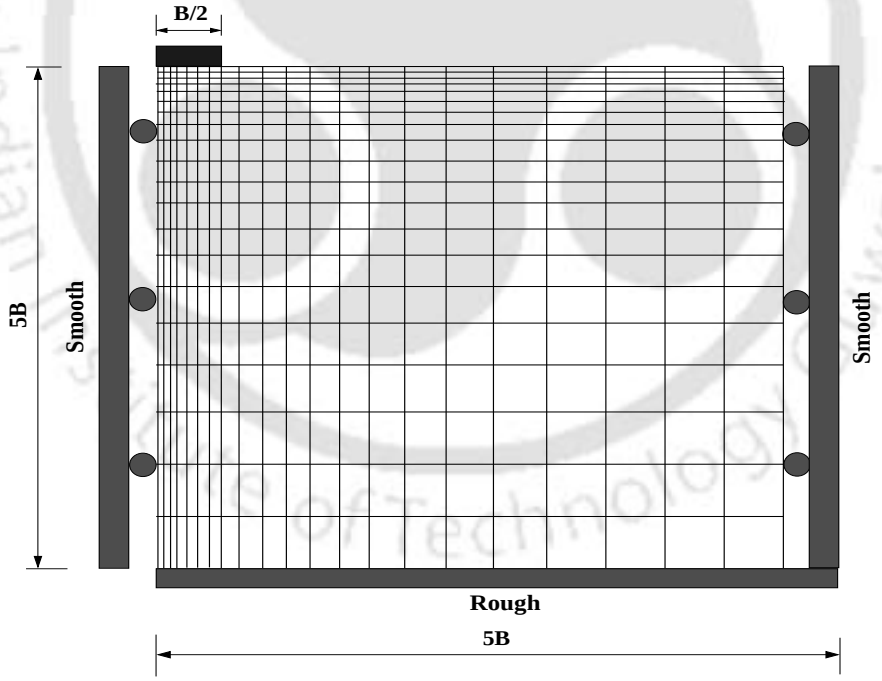


Figure 9.30: Finite element mesh(Mesh2) for single footing analysis by RK method

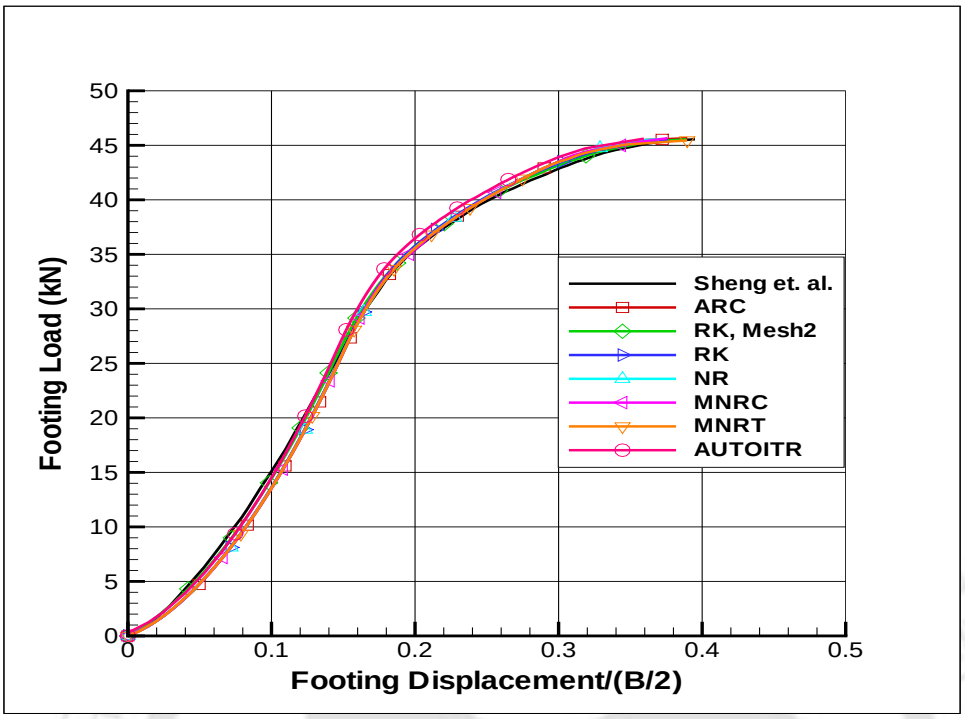


Figure 9.31: Comparison of all methods, drained: Footing displacement vs Footing load

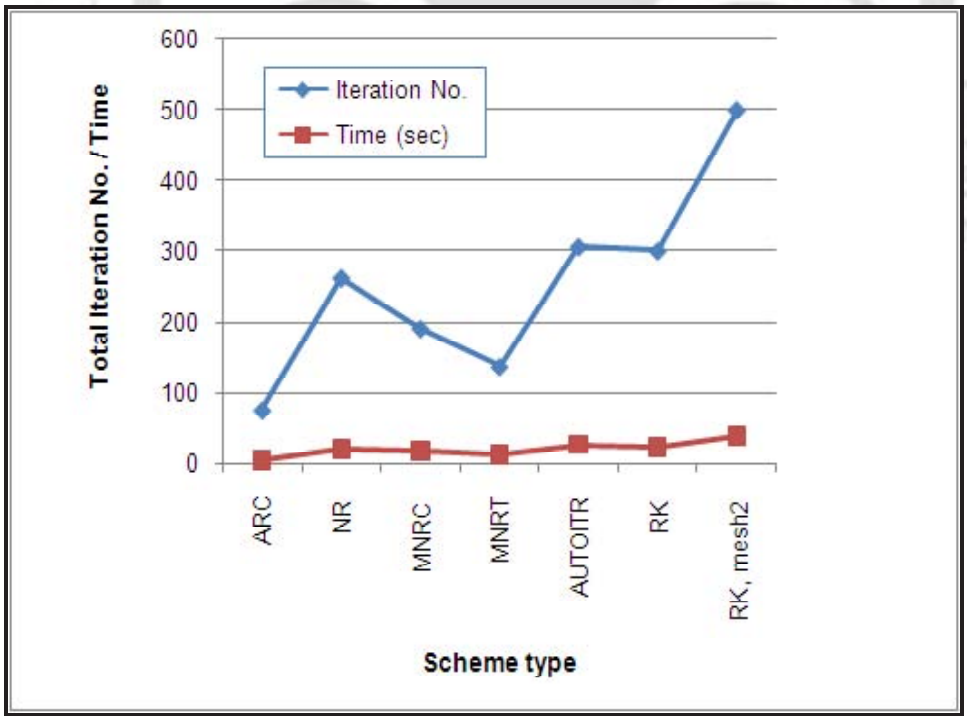


Figure 9.32: Total number of iteration and time required for different methods

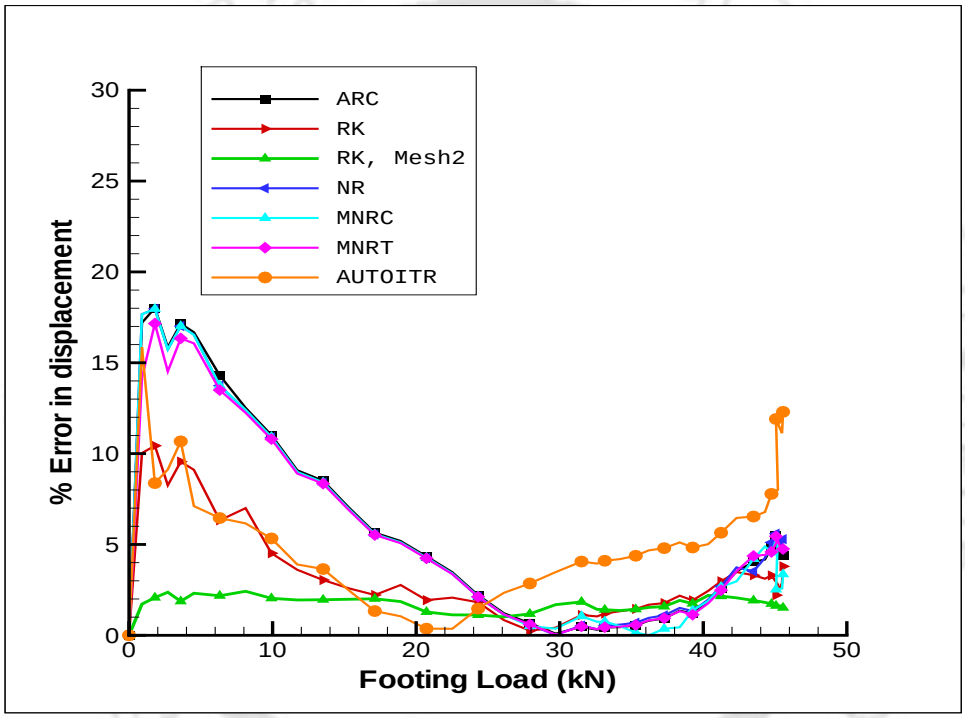


Figure 9.33: Percentage error in displacement for different methods

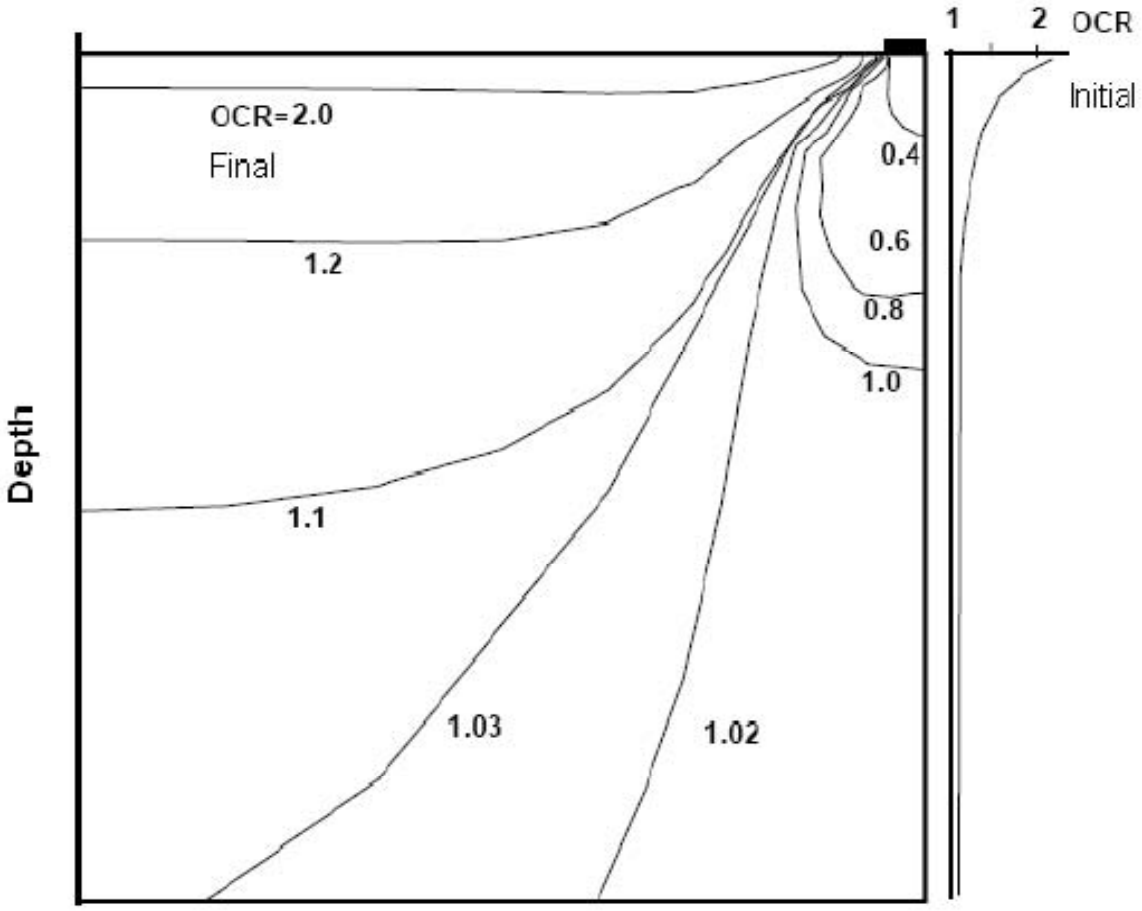


Figure 9.34: Initial and final OCR from RK method

9.3.3 Analysis of results of two identical strip footings

To study of the effect of the distance between two identical strip footings on its bearing capacity, the analysis is done for four different distances. It is assumed that the soil within a zone of 5Bx5B is effected by the foundation loading. So, initially the footings are kept at a distance of 12B apart, so that their zones of influence are not overlapped. Then it is reduced to 10B (Figure 9.35), 8B and 5B. In the analysis of a single strip footing, it is found that the relative error in the case of Runge-Kutta method is less as compared to other methods. Hence Runge-Kutta method is used

for this analysis.

The load-settlement curves of this problem are plotted for the node lying directly below the footings at the centre line. As the problem is symmetric, the load settlement curves for both the footings are identical. The plotted load is the total vertical load applied in kN/m. The plotted settlement is the settlement per unit width of the footing. The following Figure 9.36 compares the load settlement curves obtained for different distances between the footings with Runge-Kutta method applied to MCCM with the proposed integration algorithms. The load settlement curve for a single footing is also shown in the same plot to have a direct comparison.

Table 9.3 shows the failure load and settlement for these three cases. The total number of iteration and time required by Runge-Kutta method for each analysis are also shown.

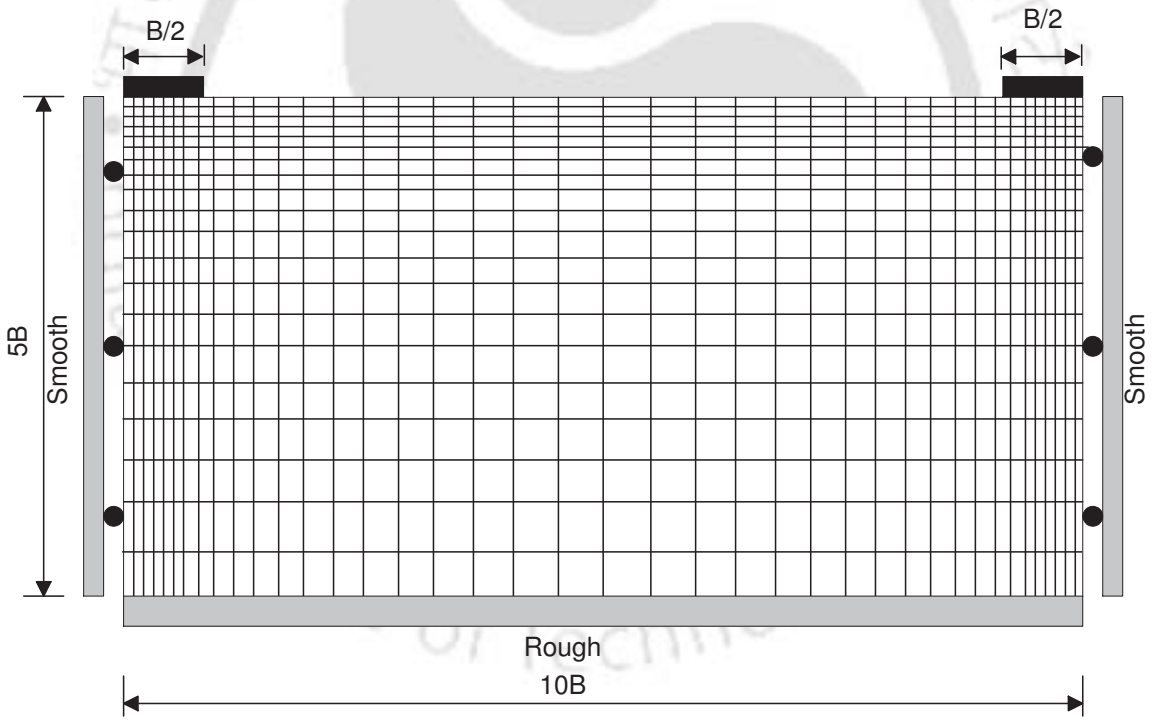


Figure 9.35: FE mesh for two footings spaced at different distance apart.

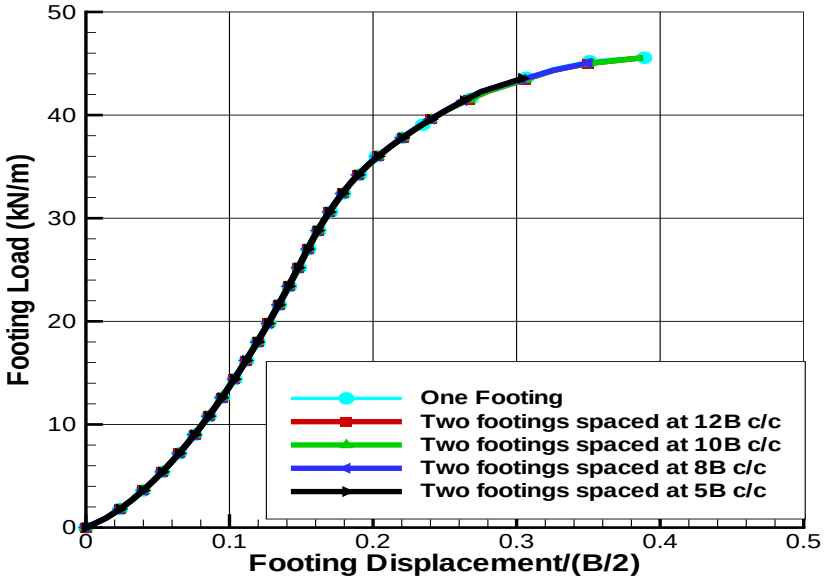


Figure 9.36: Load-displacement curve for two footings spaced at different distance apart.

Table 9.3: Plane strain analysis of two identical strip footings placed at different distances apart: comparison of results

Distance between the footings	Load at failure	Displacement at failure	No of iteration
12B	45.56	0.388	729
10B	45.56	0.388	732
8B	45.06	0.350	721
5B	43.56	0.304	708
Single footing	45.61	0.389	500

9.3.4 Summary of boundary value problems

The results of the bearing capacity test of a rigid strip footing obtained by various iterative schemes with the two integration algorithms for MCCM are presented in the previous sections. There is good agreement of these results near the failure load with the one reported by Sheng et al. [146], except the Automatic iteration method. The relative error presented in Figure 9.33 is found to be more in the initial portion of the loading with a maximum value of 18 percent and is of negligible order of about 5 percent near the failure load. The reason behind this small amount of deviation of result in the initial portion of loading may be due the type of constitutive model and the integration algorithms used in the study. In the analysis of Sheng et al., Generalized Cam-clay model with an explicit integration algorithm was used. In the present study MCCM is used with an implicit integration algorithm, where the nonlinearity in the elastic region is given due consideration.

It is found that the relative error in the case of Runge-Kutta method with a more refined mesh is the least as compared to other methods. But the number of iteration and hence the time required by Runge-Kutta method for this new refined mesh is increased by about seventy percent of that of the previous mesh. On the other hand, from the point of view of number of iterations and time required to run the program, arclength method is found to be more efficient than the others. The percentage of error as well as iteration number and time required by the automatic iteration method is the highest. The same conclusion is drawn from the results of the triaxial tests also. Therefor arclength method can be considered as an efficient iterative tool and can be preferred over other methods for MCCM, and Runge-Kutta method can be utilized where more accuracy in the analysis is required.

In case of the second boundary value problem, the load settlement curve of the footings are not effected when the distance between them are more or equal to about 10B, where B is the width of the footings. Both load and settlement at failure of the footings are effected when the distance between them are further reduced. It is seen that failure load is reduced by 4.5 percentage with respect to the failure load for single footing when the distance between the footings are reduced to 5B.

Chapter 10

Conclusions

10.1 Introduction

In this thesis some of the issues associated with object-oriented finite element implementation for geomechanics problems have been studied and an object-oriented nonlinear finite element framework for modified Cam clay model(MCCM) is developed using c++ as programming language. The framework can be extended in many directions, such as, inclusion of new analysis type, new interpolation type, new material models, new integration algorithms for an existing models, inclusion of different load stepping schemes with an existing integration algorithm and material model.

An implicit integration algorithms for MCCM is presented. The more general diagonally implicit Runge-Kutta (DIRK) method is applied to integrate the MCCM using the concept of differential algebraic equations (DAEs). Some nonlinear solution techniques and load stepping schemes are implemented with MCCM in this object-oriented finite element framework.

The thesis starts with the general overview of the works done in this study, the motivation and general objective of the present study in Chapter 1. In Chapter 2 a brief literature review in the relevant fields is given. Chapter 3 and Chapter 4 present a general introduction of the Cam clay group of models and nonlinear solution techniques in FEM.

The main contribution of the present work have been described in chapters from Chapter 5 to Chapter 9. The implicit integration algorithm presented for MCCM has been described in Chapter 5. Using the concept of DAEs for the interpretation of nonlinear FE analysis, the general case of DIRK method has been applied for integrating MCCM, which is described in Chapter 6. Two special cases of DIRK methods, namely (i) backward Euler method and (ii) trapezoidal rule are implemented for MCCM. An object-oriented finite element framework for nonlinear analysis using MCCM is developed and described in Chapter 7. The implementation of these integration algorithms for MCCM and various nonlinear solution techniques in the proposed object-oriented FE framework is described in Chapter 8. Some relevant problems have been solved using these integration algorithms with different nonlinear solution techniques and have been presented in Chapter 9. In the next sections, main contributions of this research work have been summarized. The thesis concludes with the scope for future work.

10.2 Main contributions

10.2.1 General highlights

The major contribution of this research is focused on the areas of integration of MCCM equations and development of an object-oriented finite element analysis framework for MCCM. The contributions are listed as below:

1. Successful development of an implicit integration algorithm for MCCM. In this integration algorithm, nonlinearity of this model in the elastic region is taken into account and the elastic moduli are treated implicitly in the elasto-plastic region, which improves the convergence at the global level.

2. The comparative study of various nonlinear load stepping schemes for MCCM with implicit integration algorithm in terms of efficiency and accuracy is done.

3. Successful implementation of the concept of DAEs to integrate the MCCM using DIRK method and implementation of the α method.

4. Development of an object-oriented nonlinear finite element analysis framework for analyzing geomechanics problems. The MCCM with different integration algorithms and load stepping schemes are implemented in the framework and the performance is studied by analyzing some practical geomechanics problems.

10.2.2 Implicit integration algorithm for MCCM

An implicit integration algorithm is presented to integrate the rate constitutive equation for MCCM and the corresponding consistent tangent modulus has been derived. The following points can be drawn regarding this algorithm.

- The nonlinear elastic constitutive equations of MCCM is integrated using the idea of secant elastic moduli. The elastic tangential modulus is obtained by deriving the stress tensor with respect to the strain tensor at elastic level. The Newton-Raphson method is used to solve the resulting constitutive equation at local level.
- The elasto-plastic constitutive relations are integrated using return mapping algorithms in which the return directions are computed by closest point projection, considering associative flow rule.
- In the present algorithm the elastic secant moduli are treated implicitly by using both first-order forward Euler scheme and second-order backward Euler scheme. It is found that treating the elastic secant modulus implicitly improves the convergence at global level.
- The generality and applicability of the algorithm are tested through numerical examples, which demonstrate its local as well as global accuracy and stability.

10.2.3 Nonlinear solution techniques and load stepping schemes

Newton-Raphson method, modified Newton-Raphson method with modified Thomas accelerator, modified Newton-Raphson method with Chen accelerator, Automatic

iterative scheme, and arclength method have been implemented with the proposed integration algorithm for MCCM. The following points can be drawn regarding these nonlinear solution procedure.

- All the nonlinear solution techniques successfully trace the equilibrium path when used with the proposed integration algorithm for MCCM.
- Their performances in some practical geomechanics problems are compared in terms of solution accuracy, number of global iteration and CPU time required to run the program.
- While comparing these results obtained with these schemes with established data, it can be seen that the solution obtained by arclength method is giving less error as compared to the other methods. At the same time, the total number of global iterations and the time required to run the program are also less in the case of arclength method. Therefore arclength method can be preferred over the other methods when more accurate result is required with economy.

10.2.4 Implementation of the concept of DAEs to integrate the MCCM

The context of interpretation of the discretized form of the principle of virtual displacement together with the ODEs of the constitutive model as a system of DAEs is discussed and the concept is implemented to MCCM. Treating the nonlinear finite element as DAEs facilitates to apply all the numerical algorithms of mathematics for its solution. Based on the study, the following points can be drawn regarding the implementation of the concept of DAEs to integrate the MCCM.

- The general case of DIRK method is applied to integrate the MCCM, which is stiffly accurate and has better stability property.
- A multi-level Newton-Raphson method is employed to solve the resulting nonlinear equations.

- In this study, we have implemented the α method to integrate the MCCM with two special cases (i) $\alpha = 0$, which gives modified Euler method and (ii) $\alpha = 0.5$, which gives trapezoidal rule.
- By putting $\alpha = 0$, we get the similar expressions as was obtained by the proposed implicit integration algorithm of MCCM where, forward and backward Euler methods were employed.
- The results of a boundary value problem obtained by the trapezoidal rule while applied to MCCM is compared with the established data along with the results obtained by the proposed implicit integration algorithms with different nonlinear solution techniques. It is seen that the result of trapezoidal rule is having the least error as compared to the other methods. But the number of iteration and time required to run the program was more in the case of trapezoidal rule.
- A more refined mesh was used to solve the same boundary value problem by trapezoidal rule. In this case, though the error incorporated is reduced by some extent, the number of iteration and the time required to run the program were also increased accordingly.

10.3 Scope for future works

The present work concentrates mainly on the development of an object-oriented finite element framework, development of different integration algorithms for MCCM and its implementation in the proposed object-oriented finite element framework. More general DIRK method is employed to integrate the MCCM equation by using the concept of DAEs. Emphasis is also given on the performance of some of the nonlinear solution techniques and load stepping schemes while using with MCCM. Other aspects which are required to be considered for future study in the direction of the present study include the following.

- A more details study can be performed on the effect of the various nonlinear solution techniques when employed with implicit integration algorithms for other soil models.

- The object-oriented framework presented here can be extended for inclusion of other material models and integration algorithms.
- The object-oriented framework presented here can be extended for inclusion of large deformation.
- Other higher ordered DIRK methods can be implemented for MCCM using the concept of DAEs presented in this study.

.

Bibliography

[1] Abbo A. J., Sloan S. W.;(1996). "An automatic load stepping algorithm with error control"; *International Journal for Numerical Methods in Engineering*; Vol. 39; p: 1737-1759.

[2] Abelev A. V., Gutta S. K., Lade P. V., Yamamuro J. A.; (2007). Modeling Cross Anisotropy in Granular Materials. ;*J. Eng. Mech.*; Vol. 133(8); p: 919-932; doi: 10.1061/(ASCE)0733-9399(2007)133:8(919)

[3] Adeli H., Yu G.; (1995). "An Integrated Coupling Environment of Complex Engineering Problems Using the Object-oriented Programming Paradigm and a Black Board architecture" ; *Computers and Structures* ; Vol. 54 ; No. 2.

[4] Akamatsu M., Nakane K., Ohno N.; (2005). " An implicit integration scheme for nonisothermal viscoplastic, nonlinear kinematic hardening model"; *CMES: Computer Modeling in Engineering and Sciences*; Vol. 10; No. 3; p: 217-228.

[5] Amorosi A., Boldini D., Germano V.; (2008). "Implicit integration of a mixed isotropic-kinematic hardening plasticity model for structured clays"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 32; p: 1173-1203.

[6] Andrianopoulos K. I., Papadimitriou A. G., Bouckovalas G. D.; (2010). "Explicit integration of bounding surface model for the analysis of earthquake soil liquefaction"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 34; p: 1586-1614.

[7] Archer G.C., Fenves G., Thewalt C.; (1999). "A new object-oriented finite element analysis program architecture"; *Computers and Structures*; Vol. 70; p: 63-75.

[8] Armero F., Perez-Foguest A.; (2002). "On the formulation of closest point projection in elasto-plasticity Part I The variation structure"; *International journal for Numerical methods in Engineering*; Vol. 53; p: 297-329.

[9] Artioli E., Auricchio F., Da Veiga L. B.; (2005). "Integration scheme for von Mises plasticity model based on exponential map : Numerical investigations and theoretical considerations"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 64; p: 1133-1165.

[10] Artioli E., Auricchio F., Da Veiga L. B.; (2006). "A novel optimal exponential based integration algorithm for von Mises plasticity with linear hardening : Theoretical analysis on yield consistency accuracy, convergence and numerical investigations"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 67; p: 449-495.

[11] Artioli E., Auricchio F., Da Veiga L. B.; (2007). "Generalized midpoint integration algorithm for J_2 plasticity with linear hardening"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 72; p: 422-463.

[12] Artioli E., Auricchio F., Da Veiga L. B.; (2007). "Second order accurate integration algorithm for von Mises plasticity with a nonlinear kinematic hardening mechanism"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 196; p: 1827-1846.

[13] Atluri S. N.; (1984a). "On constitutive relations at finite strain: hypo-elasticity with isotropic or kinematic hardening"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 43; p: 137-171.

[14] Atluri S. N.; (1984b). "Alternate stress and conjugate strain measures, and mixed variational formulations involving rigid rotations, for computational analysis of finitely deformed plates and shells. Part-I Theory"; *Computers and Structures*; Vol. 18; No. 1; p: 93-116.

[15] Auricchio F., Da Veiga L. B.; (2003). "On a new integration scheme for von Mises plasticity with linear hardening"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 56; p: 1375-1396.

[16] Babu G. L. S., Chouksey S. K.; (2010); "Model for analysis of fiber-reinforced clayey soil"; *Geomechanics and Geoengineering: An International Journal*; Vol. 5(4); p: 277285.

[17] Balagurusamy E.; (2006). "Object Oriented Programming with C++"; The McGraw-Hill Companies, Third Edition.

[18] Bathe K. J.; (1996). "Finite Element Procedures"; *Prentice Hall*

[19] Baugh J. W., Rehak D. E.; (1992). "Data abstraction in engineering software development"; *Journal of Computing in Civil Engineering* ; Vol. 6 ; No. 3.

[20] Besson J., Foerch R.; (1997). "Large scale object-oriented finite element code design"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 142; p: 165-187.

[21] Betteieb M. B., Lemoine X., Duchene L., Habraken A. M.; (2011). "On the numerical integration of an advanced Gurson model"; *International journal for numerical methods in engineering*; Vol. 85; p: 1049-1072.

[22] Booch G.; (1994). "Object-Oriented Analysis and Design"; *Clifornia: Benjamin.*

[23] Borja R. I., Lee R. S., Seed R. B.; (1989). "Numerical simulation of excavation in elastoplastic soils"; *International Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 13(3); p: 231-249.

[24] Borja R. I., Lee R. S.; (1990). "Cam-clay plasticity, part I: Implicit integration of elasto-plastic constitutive relations"; *Computer Methods in Applied Mechanics and Enginnering*; Vol. 78; p: 49-72.

[25] Borja R. I.; (1991). "Cam-clay plasticity, Part II, Implicit integration of constitutive equations based on a nonlinear stress predictor"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 88; p: 225-240.

[26] Borja R. I., Tamagnini C.; (1998). "Cam-clay plasticity. Part III: Extension of the innitesimal model to include nite strains"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 155; p: 73-95.

[27] Borja R. I., Lin C., Montáns F. J.; (2001). "Cam-clay plasticity, Part IV: Implicit integration of anisotropic bounding surface model with nonlinear hyperelasticity and ellipsoidal loading function"; *Computer Methods in Applied Mechanics and Enginnering*; Vol. 190; p: 3293-3323.

[28] Borja R. I., Sama K. M., Sanz P. F.; (2003). "On the numerical integration of three-invariant elstoplastic constitutive models"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 192; p: 1227-1258.

[29] Boris J., Stein S.; (1998). "Tensor Objects in Finite Element Programming"; *International Journal for Numerical Methods in Engineering*; Vol. 41; No. 5.

[30] Boyan L., Charles A.; (2005). "Object Oriented Design of Parallel and Sequential Finite Element Codes"; *13th ACME Conference: University of Sheffield*.

[31] Brandt J., Nilsson L.; (1999); "A constitutive model for compaction of granular media, with account for deformation induced anisotropy"; *Mechanics of Cohesive-Frictional Materials*; Vol. 4(4); p: 391-418.

[32] Britto A. M., Gunn M. J.; (1987). "Critical State Soil Mechanics via Finite Elements"; *Ellis Horwood Limited, England*.

[33] Buttner J., Semeon B.; (2002). "Runge-Kutta methods for elasto-plasticity"; *Applied Numerical Mathematics*; Vol. 41; p: 443-458.

[34] Callari C., Auricchio F., Sacco E.; (1998). "A nite-strain Cam-clay model in the framework of multiplicative elasto-plasticity"; *International Journal of Plasticity*; Vol. 14(12); p: 1155-1187.

[35] Chang C., Yin Z.; (2010). "Micromechanical Modeling for Inherent Anisotropy in Granular Materials.; *J. Eng. Mech.*; Vol. 136(7); p: 830-839; doi: 10.1061/(ASCE)EM.1943-7889.0000125

[36] Clausen J., Damkilde L., Anderson L.; (2006). "Efficient return algorithms for associated plasticity with multiple yield planes"; *International Journal for Numerical Methods in Engineering*; Vol. 66; p: 1036-1059.

[37] Cocchetti G., Perego V.; (2003). "A rigorous bound on error in backward-difference elasto-plastic time integration"; *Computer Methods Applied Mechanics and Engineering*; Vol. 192; p: 4909-4927.

[38] Cohoon J. P., Davidson J. W.; (2000). "C++ Programming Design: An Introduction to Programming and Object-oriented Design"; *Tata McGraw Hill* ; 2nd Edition.

[39] Crisfield M. A.; (1983). "An Arc-Length Method Including Line Searches and Accelerations"; *International Journal for Numerical Methods in Engineering*; Vol. 19; p: 1269-1289.

[40] Crisfield M. A.; (1991). "Non-Linear Finite Element Analysis of Solids and Structures: Essentials"; *John Wiley and Sons, Inc. New York, NY, USA*.

[41] Crisfield M. A.; (1997). "Non-Linear Finite Element Analysis of Solids and Structures: Advanced Topics"; *John Wiley and Sons, Inc. New York, NY, USA*.

[42] Crouch R. S., Askes H.; (2009). "Analytical CPP in energy mapped stress space : Application to a modified Drucker-Prager yield surface"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 198; p: 853-859.

[43] Das S.; (2002). "Object-oriented finite element programming: A framework for implementation of plane stress and plate bending"; *M. Tech. Project Report, Indian Institute of Technology Guwahati, India.*

[44] Diaz J., Meissner U., Burghardt M.; (2000). "Time-dependent three-dimensional finite-element ground model for geotechnical engineering problems."; *Computing in Civil and Building Engineering, ASCE*; doi: 10.1061/40513(279)190; p: 1458-1465.

[45] Ding K. Z., Qin Q. H., Cardew-Hall M.; (2007). "Substepping algorithm with stress correction for the simulation of sheet metal forming process"; *International journal of mechanical sciences*; Vol. 49; p: 1289-1308.

[46] Dubois-perlin Y., Zimmermann T., Bomme P.; (1992). "Object-oriented finite element programming II. A prototype program in Smalltalk" ; *Computer Methods in Applied Mechanics and Engineering*; Vol. 98; p: 361-397.

[47] Dubois-perlin Y., Zimmermann T.; (1993). "Object-oriented Finite Element Programming: III An Efficient Implementation in C++"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 108; p: 165-183.

[48] Dubois-perlin Y., Pegon P.; (1998). "Linear constraints in object-oriented finite element programming";*Computer Methods in Applied Mechanics and Engineering*; Vol. 154; p: 31-39.

[49] Dubois-perlin Y., Pegon P.; (1998). "Object-oriented Programming in Nonlinear Finite Element Analysis"; *Computers and Structures*; Vol. 67; p: 225-241.

[50] Duncan James Michael; (1996). "State of the Art: Limit Equilibrium and Finite-Element Analysis of Slopes"; *Journal of geotechnical Engineering*; Vol. 122; No. 7; p: 577-596.

[51] Eckert S., Baeser H., Gross D., Scherf O.; (2004). "BDF2 integration method with step size control for elasto-plasticity"; *Computational Mechanics*; Vol. 34; p: 377-386.

[52] Ellsiepen P., Hartmann S.; (2001). "Remarks on the interpretation of current non-linear finite element analysis as differential-algebraic equations"; *International Journal for Numerical Methods in Engineering*; Vol. 51; p: 679-707.

[53] Eyheramendy D., Zimmermann T.; (1996). "Object-oriented Finite Elements II. A symbolic environment for automatic programming" ; *Computer Methods in Applied Mechanics and Engineering*; Vol. 132; p: 277-304.

[54] Eyheramendy D., Zimmermann T.; (1998). "Object-oriented finite elements III. Theory and application of automatic programming"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 154; p: 41-68.

[55] Eyheramendy D., Zimmermann T.; (2001). "Object-oriented Finite elements. IV. Symbolic derivations and automatic programming of nonlinear formulations"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 190 p: 2729-2751.

[56] Fenves G. L.; (1990). "Object-oriented programming for engineering software development"; *Engineering with Computers*; Vol. 6; No. 1; p: 1-15.

[57] Forde B. W. R., Foschi R. O., Stiemer S. F.; (1990). "Object-oriented finite element analysis"; *Computers and Structures*; Vol. 34; No. 3; p: 355-374.

[58] Franco J. R. Q., Barros F. B., Malard F. P., Balabram A.; (2006). "Object oriented programming applied to a finite element technique for the limit analysis

of axisymmetrical pressure vessels"; *Advances in Engineering Software*; Vol. 37; p: 195-204.

[59] Gajo A., Muir Wood D.; (1999). "A kinematic hardening constitutive model for sands: the multiaxial formulation"; *International Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 23(9); p: 925965.

[60] Gallipoli D., Gens A., Sharma R., Vaunat J.; (2003). "An elasto-plastic model for unsaturated soil incorporating the effects of suction and degree of saturation on mechanical behaviour"; *Geotechnique*; Vol. 53; No. 1; p: 123-135.

[61] Gens A., Potts D. M.; (1988) "Critical state models in computational geomechanics"; *Engineering Computations*; Vol. 5; p: 178-197.

[62] Georgiadis K., Potts D. M., Zdravkovic L.; (2004). "Modeling the shear strength of soils in the general stress space"; *Computers and Geotechnics*; Vol. 31; p: 357-364.

[63] Gopal Ranjan, Rao A.S.R.; (2002). "Basic and Applied Soil Mechanics"; *New Age International Publishers*; 2nd Edition.

[64] Hartmann S.; (2005). "A remark on the application of the Newton-Raphson method in non-linear finite element analysis"; *Computational Mechanics*; Vol. 36; p: 100-116.

[65] Hashash Y. M. A., Whittle A. J.; (1992). "Integration of the modified cam-clay model in non-linear finite element analysis"; *Computers and Geotechnics*; Vol. 14; No. 2; p: 59-83.

[66] Heng B. C. P., Mackie R. I.; (2009). Using Design Patterns in Object-Oriented Finite Element Programming; *Computers and Structures*; Vol. 87; No. 15-16; p: 952-961.

[67] Hickman R. J., Gutierrez M.; (2005). "An internally consistent integration method for critical state models"; *International Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 29(3); p: 227-248.

[68] Hong-en LI, Yong-jun HE, Guang-ya FAN, Tong-chun LI, Manuel PASTOR; (2011). "Recent developments of generalized plasticity models for saturated and unsaturated soils"; *Water Science and Engineering*; Vol. 4(3); p: 329-344.

[69] Hovis J. K., Oliphant D., Dubois-Perlin Y., Pegon P.; (1998). "Object-oriented Programming in Nonlinear Finite Element Analysis"; *Computers and Structures*; Vol. 67; No. 4; p: 225-241.

[70] Hueckel T.; (1997). "Chemo-plasticity of clays subjected to stress and flow of a single contaminant"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 21, p: 43-72

[71] Hughes J. R., Taylor R. L., Kanoknukulchai W.; (1977). "A Simple and Efficient Finite Element for Plate Bending"; *International Journal for Numerical Methods in Engineering* ; Vol. 11; p: 1529-1543.

[72] Jakobsen K.P., Lade P.V.; (2002). "Implementation algorithm for a single hardening constitutive model for geomaterials"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 26; p: 661-681.

[73] Jeremic B., Sture S.; (1997). "Implicit Integrations in Elasto-Plastic Geotechnics"; *International Journal for Mechanics of Cohesive-Frictional Materials and Structures*; Vol. 2; p: 165-183.

[74] Jeremic B., Yang Z.; (2002). "Template elasto-plastic computations in geomechanics"; *International Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 26(14); p: 1407-1427.

[75] Justino J. G., Alves M. K., Klein A. N., Al Qureshi H. A.; (2006). "A comparative analysis of elasto-plastic constitutive models for porous sintered materials"; *Journal of material processing technology*; Vol. 179; p: 44-49.

[76] Keavey M. A.; (2002). "Canonical form return mapping algorithm for rate independent plasticity"; *International journal for Numerical methods in Engineering*; Vol. 53; p: 1491-1510.

[77] Kobayashi M., Mukai M., Takahashi H., Ohno N., Kawakami T., Ishikawa T.; (2003). "Implicit integration and consistent tangent modulus of a time dependent non-uniform constitutive model"; *International journal for Numerical methods in Engineering*; Vol. 58; p: 1523-1543.

[78] Kojic M., Vlastelica I., Zivkovic M.; (2002). "Implicit stress integration procedure for small and large strains of the Gurson material model"; *Int. J. Numer. Methods Eng.*; Vol. 53; p: 2701-2720.

[79] Kong X. A.; (1996). "A data design approach for object-oriented fem programs"; *Compurers and Strucrures*; Vol. 61(3); p: 503-513.

[80] Kong X. A. , Chen D. P. ; (1995). "An Object-oriented Design of FEM Programs" ; *Computers and Structures* ; Vol.57 ; No. 1 ; p:157-166.

[81] Kouhia R., Marjamki P., Kivilahti J; (2004). "On the implicit integration of ratedependent inelastic constitutive models"; *Int. J. Numer. Meth. Engng*; Vol. xx; p: 1-25.

[82] Krabbenhoft K., Lyamin A. V., Sloan S. W.; (2007). "Formulation and the solution of some plasticity problems with a conic programs"; *International journal of solids and structures*; Vol. 44; p: 1533-1549.

[83] Krabbenhoft K., Lyamin A. V., Sloan S. W., Wriggers P.; (2007). "An interior point algorithm for elasto-palsticity"; *International journal for numerical methods in engineering*; Vol. 69; p: 592-626.

[84] Krabbenhoft K., Lyamin A. V.; (2012). "Computational Cam clay plasticity using second order cone programming, Computer methods in Applied mechanics and engineering, vol 209-212, pp239-249, 2012.

[85] Krishnamoorthy C. S.; (2000). "Finite Element Analysis: Theory and Programming"; *Tata McGraw-Hill Publishing Company Limited*; 2nd Edition.

[86] Kromer V., Dufoss F., Gueury M.; (2004). "An object-oriented design of a finite element code: application to multibody systems analysis"; *Advances in Engineering Software*; Vol. 35(5); p: 273-287.

[87] Kromer V., Dufoss F., Gueury M.; (2005). "On the implementation of object-oriented philosophy for the design of a finite element code dedicated to multibody systems"; *Finite Elements in Analysis and Design*; Vol. 41(5); p: 493-520.

[88] Kulkarni D. V., Tortorelli D. A., Wallin M.; (2006). "A Newton-Schur alternative to consistent tangent approach in computational plasticity"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 196; p: 1169-1177.

[89] Lade P., Gutta S., Yamamuro J.; (2009). "Kinematic hardening predictions of large stress-reversals in 3-D test on loose sand."; *Computers and Geotechnics*; Vol. 36(8); p: 1285-1297.

[90] Lee J., Fenves G. L.; (2001). "A return-mapping algorithm for elastic-damage model : 3D and plane stress formulation"; *International journal for Numerical methods in Engineering*; Vol. 50; p: 487-506.

[91] Li Q., Sun S., Xue Y.; (2010). "Analyses and development of a hierarchy of frozen soil models for cold region study"; *Journal of Geophysical Research*; Vol. 115; D03107, doi:10.1029/2009JD012530.

[92] Lichao Y., Ashok V. K.; (2000). "An Object-oriented Modular Framework for Implementing the Finite Element Method"; *Computers and Structures*; Vol. 79; p: 918-928.

[93] Ling HI, Yang S.; (2006). "Unied sand model based on the critical state and generalized plasticity"; *Journal of Engineering Mechanics (ASCE)*; Vol. 132(12); p: 1380-1391.

[94] Liu C. S.; (2004). "A consistent numerical scheme for the von Mises mixed hardening constitutive equations"; *International Journal of plasticity*; Vol. 20; p: 663-704.

[95] Liu C. S., Li C. F.; (2005). "Geometrical numerical algorithms for plasticity model with Armstrong-Frederick kinematic hardening rule and strain and stress control"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 63; p: 1396-1423.

[96] Liu H., Song E., Ling H. I.; (2006). Constitutive modeling of soil-structure interface through the concept of critical state soil mechanics; *Mechanics Research Communications*; Vol. 33; p: 515-531.

[97] Lu J., White D. W., Chen W., Dunsmore H. E.; (1995). "A Matrix Class Library in C++ for Structural engineering computing"; *Computers and Structures*; Vol. 55; p: 95-111.

[98] Luccioni L. X., Pestana J. M., Rodriguez-Marek A.; (2000). "An implicit integration algorithm for the finite element implementation of a nonlinear anisotropic material model including hysteretic nonlinearity"; *Comput. Meth. Appl. Mech. Eng.*; Vol. 190; p: 1827-1844.

[99] Luccioni L. X., Pestana J. M., Taylor R. L.; (2001). "Finite element implementation of non-linear elastoplastic constitutive laws using local and global explicit algorithms with automatic error control"; *Int. J. Numer. Methods Eng.*; Vol. 50; p: 1191-1212.

[100] Luiz Fernando Martha; (2002). "An Object-Oriented Framework for Finite Element Programming"; *Fifth World Congress on Computational Mechanics, Vienna, Austria- WCCM V*; Vol. V; Eds.: H.A. Mang, F.G. Rammerstorfer, J. Eberhardsteiner.

[101] Mackerle J.; (2004). "Object-oriented programming in FEM and BEM: a bibliography (1990-2003)"; *Advances in Engineering Software*; Vol. 35; p: 325-336.

[102] Mackie R. I.; (1992). "Object-oriented Programming of the Finite Element Method"; *International Journal for Numerical Methods in Engineering*; Vol. 35; No. 2; p: 425-436.

[103] Mackie R. I.; (1998). "Object-oriented finite element programming-the importance of data modeling"; *Advances in Engineering Software*; Vol. 30; p: 775-782.

[104] Madan A. ; (2004). "Object-oriented paradigm in programming for computer-aided analysis"; *Journal of Computing in Civil Engineering*; Vol. 18; No. 3; p: 226-236.

[105] McDowell G. R., Hau K. W.; (2003). "A simple non-associated three-surface kinematic hardening model"; *Geotechnique*; Vol. 53; No. 4; p: 433-437.

[106] McKenna F., Scott M. H., Fenves G. L.; (2010). "Nonlinear Finite-Element Analysis Software Arcitecture Using Object Composition"; *Journal of Computing in Civil Engineering*; Vol. 24; No. 1; p: 95-107.

[107] MEMON Bashir-Ahmed, SU Xiao-zu; (2004). "Arc-length technique for nonlinear finite element analysis"; *Journal of Zhejiang Univerrsity SCIENCE* ; Vol. 5; No. 5; p: 618-628.

[108] Menetrey P. H., Zimmermann T.; (1993). "Object-oriented Nonlinear Finite Element Analysis: An Application to J2 Plasticity"; *Computers and Structures* ; Vol. 49 ; No. 5; p: 767-777.

[109] "C++ Language Reference"; *Microsoft Corporation, 1991.*

[110] Miki M., Murotsu Y.; (1995). "Object-oriented Approach to Modeling and Analysis of Truss structures"; *AIAA Journal*; Vol. 33; p: 348-354.

[111] Miller G. R.; (1991). "Object-oriented Approach to Structural Analysis and Design" ; *Computers and Structures* ; Vol. 40; p: 75-82.

[112] Miller G. R.; (1993). "Coordinate Free Isoparametric Elements" ; *Computers and Structures* ; Vol. 49; p: 1027-1035.

[113] Miller G. R., Banerjee S., Sribalaskandarajah K.; (1995). "A framework for interactive computational analysis in geomechanics"; *Computers and Geotechnics*; Vol. 17(1); p: 17-37

[114] Mira P., Tonni L., Pastor M., Ferrandes Merodo J. A.; (2009). "A generalized midpoint algorithm for the integration of a generalized plasticity model for sands"; *International Journal for Numerical method in applied mechanics and Engineering*; Vol. 77; p: 1201-1233.

[115] Mita K. A., Dasari G. R., Lo K. W.; (2004). "Performance of a Three-Dimensional Hvorslev-Modified Cam Clay Model for Overconsolidated Clay"; *Internaional Journal of Geomechanics*; Vol. 4(4); p: 296-309.

[116] Modak S., Sotelino E. D.; (2002). "An object-oriented programming framework for the parallel dynamic analysis of structures"; *Computers and Structures*; Vol. 80(1); p: 77-84.

[117] Montans F. I.; (2004). "Implicit plane stress algorithm for multilayer J_2 plasticity using the Prager-Ziegler transition rule"; *International journal for Numerical methods in Engineering*; Vol. 59; p: 409-418.

[118] Nazem M., Sheng D., Carter J. P.; (2006). "Stress integration and mesh refinement for large deformation in geomechanics"; *International Journal for Numerical Methods in Engineering*; Vol. 65; p: 1002-1027.

[119] Nguyen C. D., Einav I.; (2010). "A stress return algorithm for nonlocal constitutive models of softening materials"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 82; p: 637-670.

[120] Nikishkov G. P.; (2006). "Object oriented design of a finite element code in Java"; *CMES: Computer Modeling in Engineering and Sciences*; Vol. 11(2); p: 81-90.

[121] Nikishkov G. P., Atluri S.N.; (1993). "An analytical-numerical alternating method for elastic-plastic analysis of cracks"; *Computational Mechanics*; Vol. 13(6); p: 427-442.

[122] Nylor D.J.; (1999). "On the Use of F.E.M for Assessing the Stability of Cuts and Fills"; *Numerical Models in Geomechanics-NUMOG VII*; Vol. VII; A.A.Balkema/Rotterdam/Brookfield/1999, ISBN 90 5809 0957.

[123] Ortiz M., Simo J. C.; (1986). "An analysis of a new class of integration algorithms for elastoplastic constitutive relations"; *International Journal for Numerical Methods in Engineering*; Vol. 23; p: 353-366.

[124] Perez-Foguet A., Rodrigues A., Huerta A.; (2000). "Numerical differentiation for local tangent operators in computational plasticity"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 189, p: 277-296.

[125] Perez-Foguet A., Rodrigues A., Huerta A.; (2000). "Numerical differentiation for nontrivial consistent tangent matrices : an application to MRL-Lade model"; *International journal for Numerical methods in Engineering*; Vol. 48; p: 159-184.

[126] Parez-Foguet A., Rodrigues A., Huerta A.; (2001). "Consistent tangent matrices for substepping schemes"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 190, p: 4627-4647.

[127] Pedroso D. M., Sheng D., Sloan S. W.; (2008). "Stress update algorithm for elsto-plastic models with nonconvex yield surfaces"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 76; p: 2029-2062.

[128] Peng Q., Chen M. X.; (2012). "An efficient return mapping algorithm for generalized isotropic elasto-plasticity"; *Computers and Structures*; Vol. 92-93; p: 173-184.

[129] Perez-Foguest A., Armero F.; (2002). "On the formulation of closest point projection in elasto-plasticity Part II globally convergent scheme"; *International journal for Numerical methods in Engineering*; Vol. 53; p: 331-374.

[130] Peric D.; (2006). "Analytical solutions for a three invariant Cam-clay model subjected to drained loading histories"; *International journal for numerical and analytical methods in geomechanics*; Vol. 30; p: 363-387.

[131] Pidaparti R. M. V., Hudli A. V.; (1993). "Dynamic analysis of structures using object-oriented techniques"; *Computers and Structures*; Vol. 49(1); p: 149-156.

[132] Potts D. M., Ganendra D.; (1994). "An evaluation of substepping and implicit stress point algorithms"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 189; p: 277-296.

[133] Potts David M., Zdravkovic Lidija; (1999). "Finite element analysis in geotechnical engineering, Theory"; *Thomas Telford: London.*

[134] Potts David M., Zdravkovic Lidija; (1999). "Finite element analysis in geotechnical engineering, Application"; *Thomas Telford :London.*

[135] Prevost Jean H., Popescu Radu; (2002). "Constitutive Relations for Soil Materials"; *Electronic Journal of Geotechnical Engineering, EJGE.*

[136] Reddy J. N.; (1997). "Mechanics of Laminated Composite Plates: Theory and Analysis" ; *CRC Press, Inc*; Florida-33431.

[137] Reddy J. N.; (2000). "An Introduction to Finite element Method"; *McGraw Hill Inc.*; 2nd Edition.

[138] Reissner E. ; (1945). "The Effects of Transverse Shear Deformation on the Bending of Elastic Plates" ; *Journal of Applied Mechanics* ; Vol.12 ; p: 69-77.

[139] Rezaiee-Pajand M., Nasirai C.; (2008). "On the integration scheme for Drucker-Prager's elasto-plastic models based on exponential maps"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 7; p: 799-826.

[140] Ritto-Correa M., Camotion D.; (2001). "Integration algorithm for J_2 elasto-plasticity under arbitrary mixed stress-strain control"; *International journal for Numerical methods in Engineering*; Vol. 50; p: 1213-1232.

[141] Rouainia M., Wood D. M.; (2001). "Implicit numerical integration for a kinematic hardening soil plasticity model"; *nternational Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 25(13); p: 1305-1325.

[142] Saleeb A. F., Wilt T. E., Li W.; (1998). "An Implicit integration scheme for generalized viscoplasticity with dynamic recovery"; *Computational Mechanics*; Vol. 21(6); p: 429-440.

[143] Sampath R., Zabaras N.; (2000). "An Object-oriented Framework for the Implementaion of adjoining Techniques in the Design and Control of a Complex Continuum System"; *International Journal for Numerical Methods in Engineering*; Vol. 48; p: 239-266.

[144] Scott M. H., Fenves G. L., McKenna F., Filippou F. C.; (2008). "Software Patterns for Nonlinear Beam-Column Models"; *Journal of Structural Engineering*; Vol. 134; No. 4; p: 562-571.

[145] Seifert T., Schmidt I.; (2008). "Line-search methods in general return mapping algorithms with application to porous plasticity"; *Int. J. Numer. Meth. Engng*; Vol. 73; p: 1468-1495.

[146] Sheng D., Sloan S. W., Yu H. S.; (2000). "Aspects of finite element implementation of critical state models"; *Computational Mechanics*; Vol. 26; p: 185-196.

[147] Sheng D., Sloan S. W.; (2001). "Load stepping schemes for critical state models"; *International Journal for Numerical Methods in Engineering*; Vol. 50; p: 67-93.

[148] Sheng D., Pedroso D. M., Abbo A. J.; (2008). "Non convexity and stress path dependency of unsaturated soil models".; *Computational Mechanics*; Vol. 42; p: 685-695.

[149] Simo J. C., Hughes T. J. R.; (1998). "Computational Inelasticy"; *Springer Verlag, Heidelberg.*

[150] Simo J. C., Taylor R. L.; (1985). "Consistent tangent operators for rate-independent elastoplasticity"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 48; p: 101-118.

[151] Simo J. C., Taylor R. L.; (1986). "A return mapping algorithm for plane stress elastoplasticity"; *International Journal for Numerical Methods in Engineering*; Vol. 22; p: 649-670.

[152] Simon J. Wheeler, Anu Näätänen, Minna Karstunen, Matti Lojander; (2003). "An anisotropic elastoplastic model for soft clays"; *Canadian Geotechnical Journal*; Vol. 40; p: 403-418.

[153] Singh A. K., Pandey P. C.; (1999). "An implicit integration algorithm for plane stress damage coupled elastoplastcity"; *Mechanics Research Communications*; Vol. 26; No. 6; p: 693-700.

[154] Sloan S. W.; (1987). "Substepping schemes for the numerical integration of elastoplastic stress-strain relations"; *International Journal for Numerical Methods in Engineering*; Vol. 24; p: 893-911.

[155] Sloan S. W., Booker J. R.; (1992). "Integration of Tresca and Mohr-Coulomb constitutive relations in plane strain elastoplasticity"; *International Journal for Numerical Methods in Engineering*; Vol. 33; p: 163-196.

[156] Sloan S. W., Abbo A. J., Sheng D.; (2001). "Refined explicit integration of elastoplastic models with automatic error control"; *Engineering Computations*; Vol. 18; p: 121-154.

[157] Sun D. A., Mastuoka H., Yao Y. P., Ishii H.; (2004). "An anisotropic hardening elastoplastic model for clays and sands and its application to FE analysis"; *Computers and Geotechnics*; Vol. 31; p: 37-46.

[158] Surendra Kumar; (2010). "Object-Oriented Finite Element Programming for Engineering Analysis in C++"; *Journal of Software*; Vol. 5(7); p: 689-696.

[159] Tamagnini R.; (2004). "An extended cam-Clay model for unsaturated soils with hydraulic hysteresis"; *Geotechnique*; Vol. 54; No. 3; p: 223-228.

[160] Terzaghi K., Peck R. B.; (1967). "Soil Mechanics in engineering Practice"; *Johan Wiley and Sons, New York*; 1st Edition.

[161] Timoshenko S., Goodier J. N.; (1970). "Theory of Elasticity"; *McGraw-Hill Co., Newyork*.

[162] Uchida S., Soga K., Yamamoto K.; (2012). "Critical state soil constitutive model for methane hydrate soil"; *Journal of Geophysical Research*; Vol. 117; B03209, doi:10.1029/2011JB008661.

[163] Vrh M., Halilovic M., Stok B.; (2010) "Improved explicit integration in Plasticity"; *Journal for Numerical method in applied mechanics and Engineering*; Vol. 81, p: 910-938.

[164] Wallin M., Ristinmaa M.; (2001). "Accurate stress updating algorithm based on constant stress rate assumption"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 190; p: 5583-5601.

[165] Wallin M., Ristinmaa M.; (2008). "An alternative method for the integration of constitutive damage evolution laws"; *Computational Mechanics*; Vol. 41; p: 347-359.

[166] Wang L. H., Atluri S. N.; (1994). "An analysis of an explicit algorithm and the radial return algorithm, and a proposed modification, in finite plasticity"; *Computational Mechanics*; Vol. 3; p: 380-389.

[167] Wang W., Datcheva D., Schanz T., Kolditz O.; (2006). "A substepping approach for a elasto-plasticity with rotational hardening"; *Computational Mechanics*; Vol. 37; p: 266-278.

[168] Wang X., Wang L. B., Xu L. M.; (2004). "Formulation of the return mapping algorithm for elastplastic soil models"; *Computers and Geotechnics*; Vol. 31; p: 315-338.

[169] Wilson E. L., Taylor R. L., Doherty W. P., Ghabussi T., (1973). "Incompatible Displacement Models"; *Numerical and Computer Methods in Structural Mechanics*; p: 43-57.

[170] Wood D. M.; (1994). "Soil Behaviour and Critical State Soil Mechanics"; *Cambridge University Press: Cambridge.*

[171] Yamamuro J., Kaliakin V.; (2005). "Soil Constitutive Models : Evaluation, Selection, and Calibration"; *Geo-Frontiers Congress 2005; Austin, Texas, United States; January 24-26*; Publisher: American Society of Civil Engineers.

[172] Yang Y. , Muraleetharan K. K., Yu H. S.; (2008). "Generalized trapezoidal numerical integration of an advanced soil model"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 32; p: 43-64.

[173] Yin Jean-Hua, Zhu Jun-Gao, Graham James; (2002). "A new elastic viscoplastic model for time-dependent behaviour of normally and overconsolidated clays: theory and verification"; *Canadian Geotechnical Journal*; Vol. 39; p: 157-173.

[174] Yu H. S.; (1998). "CASM: a unified state parameter model for clay and sand"; *International Journal for Numerical and Analytical Methods in Geomechanics*; Vol. 22(8); p: 621-653.

[175] Zeglinski G. W., Han R. P. S., Aitchison P.; (1994). "Object-oriented Matrix Classes for use in a Finite Element Code Using C++"; *International Journal for Numerical Methods in Engineering*; Vol. 37; p: 3921-3937.

[176] Zhang H. W., Zhou L.; (2008). "Implicit integration of a chemo-plastic constitutive model for partially saturated soils"; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 32; p: 1715-1735.

[177] Zhang Z. L.; (1995). "Explicit consistent tangent moduli with a return mapping algorithm for pressure-dependent elastoplasticity models"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 121; p: 29-44.

[178] Zhao J., Sheng D., Rouainia M., Sloan S. W.; (2005). "Explicit stress integration of complex soil models."; *Int. J. Numer. Anal. Meth. Geomech.*; Vol. 29(12); p: 1209-1229.

[179] Zienkiewicz O. C., Taylor R. L.; (1991). "The Finite Element Method"; *McGraw-Hill International Editions*; Fourth Edition; Vol. 2.

[180] Zimmermann T., Dubois-perlin Y., Bomme P.; (1992). "Object-oriented Finite Element Programming I. Governing Principles"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 98; p: 291-303.

[181] Zimmermann T., Eyheramendy D.; (1996). "Object-oriented Finite Elements I. Principles of Symbolic Derivations and Automatic Programming"; *Computer Methods in Applied Mechanics and Engineering*; Vol. 132; p: 259-376.

[182] Zimmermann T., Bomme P., Eyheramendy D., Vernier L., Commend S.; (1998). "Aspects of an object-oriented finite element environment"; *Computers and Structures*; Vol. 68; p: 1-16.

.

Appendix A

Determination of the consistent tangential moduli:

The derivatives in equation(4.2.41) can be derived implicitly from the equations (4.2.32), (4.2.33) and (4.2.34). The derivative of p w.r.t. $\boldsymbol{\varepsilon}$ is:

$$\frac{\partial p}{\partial \boldsymbol{\varepsilon}} = a_1 K \boldsymbol{l} + a_2 K \frac{\partial \phi}{\partial \boldsymbol{\varepsilon}}, \tag{.0.1}$$

where

$a_1 = (1 + p_c \vartheta \phi)/a$,

$a_2 = -(2p - p_c)/a$ and

$a = 1 + 2K\phi + p_c \vartheta \phi$.

The derivative of p_c w.r.t. $\boldsymbol{\varepsilon}$ is

$$\frac{\partial p_c}{\partial \boldsymbol{\varepsilon}} = a_3 K \boldsymbol{l} + a_4 K \frac{\partial \phi}{\partial \boldsymbol{\varepsilon}}, \tag{.0.2}$$

where

$a_3 = 2p_c \vartheta \phi / a$ and

$a_4 = \vartheta \frac{p_c}{K}(2p - p_c)/a$.

Finally, the derivative of q with respect to $\boldsymbol{\varepsilon}$ is

$$\frac{\partial q}{\partial \boldsymbol{\varepsilon}} = 2G\ a_5 \boldsymbol{n} + 2G\ a_6 \frac{\partial \phi}{\partial \boldsymbol{\varepsilon}}. \tag{.0.3}$$

where

$a_5 = \sqrt{\frac{3}{2}}(1 + 6G\frac{\phi}{M^2})^{-1}$

and $a_6 = -\frac{3q}{M^2}(1 + 6G\frac{\phi}{M^2})^{-1}$

The derivative $\partial\phi/\partial\boldsymbol{\varepsilon}$ can be determined by imposing the scalar consistency requirement (4.2.35). Differentiating $F = F(\phi(\boldsymbol{\varepsilon}))$ with respect to $\boldsymbol{\varepsilon}$ results in

$$\frac{\partial F}{\partial \boldsymbol{\varepsilon}} = \frac{2q}{M^2}\frac{\partial q}{\partial \boldsymbol{\varepsilon}} + (2p - p_c)\frac{\partial p}{\partial \boldsymbol{\varepsilon}} - p\frac{\partial p_c}{\partial \boldsymbol{\varepsilon}} \equiv 0. \tag{.0.4}$$

Substituting equations(.0.1), (.0.2) and (.0.3) in (.0.4) gives

$$\frac{\partial \phi}{\partial \boldsymbol{\varepsilon}} = b_1\boldsymbol{l} + b_2\boldsymbol{n}, \tag{.0.5}$$

where

$b_1 = -K[(a_3 - 2a_1)p + a_1p_c]/b,$

$b_2 = 2G\frac{2q}{M^2}a_5/b,$

$b = -2G\frac{2q}{M^2}a_6 - K[(2a_2 - a_4)p - a_2p_c].$

Substituting equation(.0.5) in equations (.0.1), (.0.2) and (.0.3) results in

$$\frac{\partial p}{\partial \boldsymbol{\varepsilon}} = K(a_1 + a_2b_1)\boldsymbol{l} + K(a_2b_2)\boldsymbol{n}, \tag{.0.6}$$

$$\frac{\partial q}{\partial \boldsymbol{\varepsilon}} = 2G(a_6b_1)\boldsymbol{l} + 2G(a_5 + a_6b_2)\boldsymbol{n}, \tag{.0.7}$$

$$\frac{\partial p_c}{\partial \boldsymbol{\varepsilon}} = K(a_3 + a_4b_1)\boldsymbol{l} + K(a_4b_2)\boldsymbol{n}. \tag{.0.8}$$

Again the derivative $\partial\boldsymbol{n}/\partial\boldsymbol{\varepsilon}$ can be obtained as

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\varepsilon}} = \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{S}^{tr}} : \frac{\partial \boldsymbol{S}^{tr}}{\partial \boldsymbol{\varepsilon}} = 2G\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{S}^{tr}} : \frac{\partial \Delta\boldsymbol{\varepsilon}_d}{\partial \boldsymbol{\varepsilon}} = \frac{2G}{\parallel \boldsymbol{S}^{tr} \parallel}(I - \frac{1}{3}\boldsymbol{l} \otimes \boldsymbol{l} - \boldsymbol{n} \otimes \boldsymbol{n}) \tag{.0.9}$$

where

$\Delta\boldsymbol{\varepsilon}_d = \Delta\boldsymbol{\varepsilon} - \frac{1}{3}(\Delta\boldsymbol{\varepsilon})\boldsymbol{l},$

and $I_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}).$

Substituting equations(.0.6), (.0.7), (.0.8) and (.0.9) in equation (4.2.41) produces the desired consistent tangential moduli:

$$\boldsymbol{C}_{n+1} = 2G\boldsymbol{S}I + [K(a_1 + a_2 b_1) - \frac{1}{3}(2G\boldsymbol{S})]\boldsymbol{l} \otimes \boldsymbol{l} + K(a_2 b_2)\boldsymbol{l} \otimes \boldsymbol{n} + \qquad (.0.10)$$

$$2G\sqrt{\frac{2}{3}}(a_6 b_1)\boldsymbol{n} \otimes \boldsymbol{l} + 2G[\sqrt{\frac{2}{3}}(a_5 + a_6 b_2) - \boldsymbol{S}]\boldsymbol{n} \otimes \boldsymbol{n} \qquad (.0.11)$$

where $S = \sqrt{\frac{2}{3}}q / \parallel \boldsymbol{S}^{tr}_{n+1} \parallel = \parallel \boldsymbol{S}_{n+1} \parallel / \parallel \boldsymbol{S}^{tr}_{n+1} \parallel$

Equation(.0.10) is consistence with the linearization of $\boldsymbol{\sigma}_{n+1}$ and preserves the asymptotic rate of quadratic global convergence of Newton's method.