DISCRETE ELEMENT METHOD

FOR

SLOPE STABILITY ANALYSIS

By

DAWIT KEBEDE ABERRA

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN CIVIL ENGINEERING

ADDIS ABABA UNIVERSITY
Faculty of Technology
Department of Civil Engineering
July 2003

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

DISCRETE ELEMENT METHOD FOR SLOPE STABILITY ANALYSIS

BY DAWIT KEBEDE ABERRA

APPROVED BY BOARD OF EXAMINERS

_____	_____
ADVISOR	
_____	_____
CHAIR PERSON	
_____	_____
EXTERNAL EXAMINER	
_____	_____
INTERNAL EXAMINER	

DECLARATION

"The thesis is my original work, has not been presented for a degree in any other university and all of sources of material used for the thesis have been duly acknowledged."

Candidate's

Name: _____

Signature: _____

Acknowledgements

I would like to express my deepest gratitude to my advisor ,Dr.Ing. Girma Boled, for his invaluable advice and support from start to finish of this research work. I would also like to thank Adrian Rodriguez-Marek(PhD at Washington State University, Pullman, WA,USA) for sharing his papers on the method presented here.

I would like to thank my friends here at our Faculty for their support and encouragement. My biggest thanks go to Sisay Denboba for his assistance during writing of the computer code. Special thanks go to SB Consult staff and to Ato Seifu Birke for their patience and logistic support.

Lastly, I would like to thank my very close friends for their emotional support during my academic endeavor.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

DEM: Discrete Element Method

SSDEM: Slope Stability Analysis using Discrete Element Method

FEM: Finite Element Method

FS: Factor of Safety

LEM: Limit Equilibrium Method

Abstract

A static DEM has been presented to solve slope stability analyses. The method consists of discretizing the soil mass into blocks which are joined by Winkler springs. From relation of forces with stiffness and displacement, sets of simultaneous equations are solved to find displacements and stresses. The local stresses are used to calculate factor of safety while the displacement fields are used for predicting the critical failure surface.

Example problems have been solved using DEM and the results have shown the potential of the method for slope stability analysis. Comparisons of the present model between limit equilibrium methods, FEM and with DEM models by other authors are made .The results suggest that DEM methods can be used both to predict the critical slip surface and give a factor of safety and that these outputs closely agree with the conventional limit equilibrium methods and the more rigorous FEMs.

CHAPTER ONE

1.1 Introduction

Slopes can be man made or natural. The study and quantification of their safety has been recognized as very essential for the economical prevention of life and property loss. Civil engineers and in particular Geotechnical engineers have devoted much effort and study to the understanding of the mechanisms leading to failure of slopes.

Any slope can fail because of relative displacement of two block masses. However, there is usually one surface along which the probability of failure is the highest. This surface is termed critical surface.

There are many methods of finding the critical surface. Some investigators have prepared tables for homogeneous materials and simple geometry. Commercial softwares, such as Geo-Slope and XSTABL have routines for the search of the critical surface. However, good engineering judgment is essential for the interpretation of output of such packages. Optimization techniques using calculus of variations and iterative approaches have also been used in engineering practice.

The traditional way of analyzing slopes assumes simultaneous failure along the failure surface and associates a single factor of safety for it. This assumption is not acceptable because it doesn't take into account the ability of soils to distribute stress. The other shortcoming of these methods is the fact that effects of initial state of stress and stress/strain history, soil stress-strain relations, stress/strain path to failure, differing constraints on boundary

displacements are not accounted for in modeling the potential range in distribution of interslice forces.

State of the art finite element methods can avoid these drawbacks but their implementation requires accurate determination of geotechnical parameters. In addition, because of the longer computer time involved their use is still considered expensive.

Recent research efforts have focused on describing the behavior of a soil mass using discrete elements (E.g. Chang 1992 and Rodriguez-Marek et al 1996). The discrete element method is based on simple equilibrium equations and a Mohr-Coulomb failure criterion. The method is attractive because of its simplicity and relative ease of application over the finite element approach.

1.2 Objective and Scope

The objective of this thesis is to formulate a theoretically sound method for analyzing slopes, a theoretically sound method being a method satisfying both statics and deformation compatibility. For ease in using DEM, a computer program, SSDEM, is developed. This computer algorithm is used to do comparative study with other currently used methods for slope stability analyses, namely limit equilibrium methods and FEM.

In Chapter 2, traditional limit equilibrium methods and finite-element analysis of slopes are concisely reviewed. A brief literature review is also presented on factor of safety and critical surface search. In addition, discrete element methods are introduced and the basis for static discrete element method with its application in geotechnical engineering is discussed.

Chapter 3 discusses the modeling aspects of DEM, particularly the mechanics of block interaction.

The first part of Chapter 4 deals with algorithm design and program organization. The second part studies three slopes that have been analyzed by other researchers and the output from the new program, SSDEM, is compared with that of other authors.

Chapter 5 summarizes the results in this study and forwards some recommendations for further study.

CHAPTER TWO

Literature Review

2.1 Introduction

This chapter presents a review of the limit equilibrium methods that are currently in common use. A brief discussion on factor of safety and critical surface is given. A concise note on state of the art finite element modeling of slopes is also presented. This is followed by a focused review of discrete element method for slope stability analysis.

2.2 Limit Equilibrium Methods

2.2.1 Introduction

The first works on slope stability analysis date back to 1840's. Later in the 1920's, the Swedish engineer Fellenius gave the slope stability problem an analytic solution, which became the basis for the current limit equilibrium analyses.

Closed form solutions are possible for slopes with high shear strength backfill, which are underlain by weak soils. Infinite slopes, planar, circular failure surfaces also render slope stability problem determinate.

Limit equilibrium methods start with an assumption of a failure surface. Then, using method of slices, equilibrium of each slice and overall equilibrium of the failure mass is considered. Since the problem is statically indeterminate, certain assumptions are made regarding interslice forces. These assumptions are the points that differentiate between different limit equilibrium methods.

2.2.2 Analysis Methods

Limit equilibrium methods divide a sliding mass into smaller slices as shown in fig 2.1. Each slice is acted upon by a system of forces as shown in Fig. 2.2

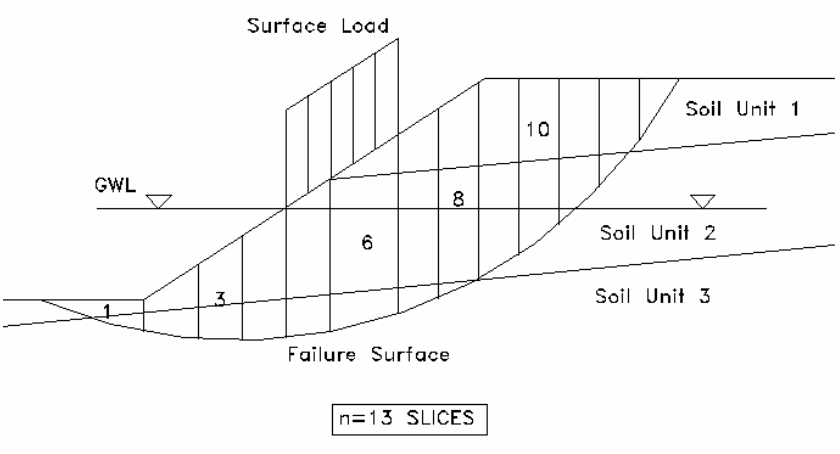
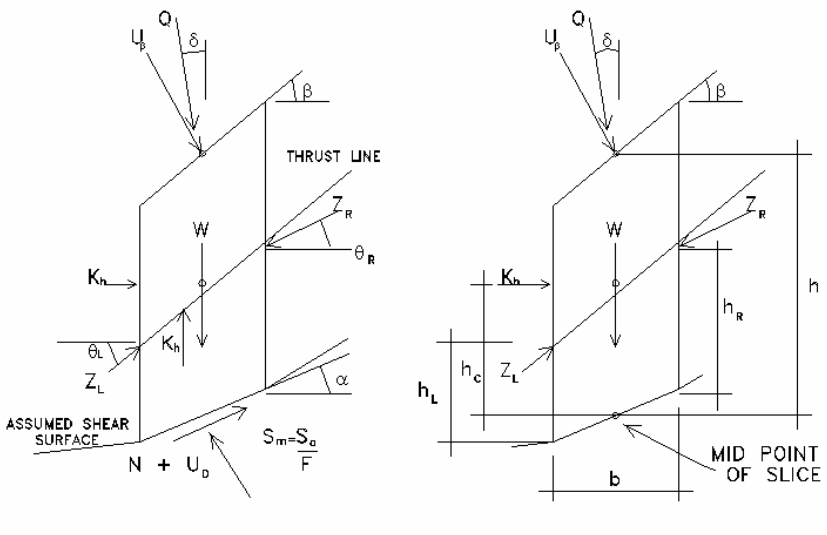


Figure 2.1 Division of sliding mass into slices

The thrust line (line connecting points of application of force) can be assumed or rigorously determined so as to satisfy complete equilibrium. However, popular simplified methods neglect the inter slice force location indicating that complete equilibrium of failure mass is not satisfied.

As shown in Table 2.1 , there are 4n equations and 6n-2 unknowns. A solution is possible by reducing the number of unknowns. One of the most common assumptions is that the normal force on the base of the slice acts at the midpoint. This reduces the number of unknowns to (5n-2) and the degree of indeteminacy to (n-2).



F= factor of safety

Sa= Available strength

= C + N' tan ∅

Sm = mobilized strength

U_d= pore water force

W= Weight of slice

N'= Effective normal force

Q= External surcharge

Kv= Vertical seismic coefficient

Kn= Horizontal seismic coefficient

Z_L= left interslice force

Zr= right interslice force

θ_L=left interslice force angle

θ_R = right interslice force angle

h_L = height of force Z_L

h_R= inclination of slice base

α = inclination of slice base

β=inclination of slice slope

b = width of slice

h_c= height to centroid of slice

Fig. 2.2 Forces acting on a typical slice

Table 2.1 Equations and unknowns associated with the method of slices.

Equations	**Condition**
n	Moment equilibrium for each slice
2n	Force equilibrium in two directions (for each slice)
n	Mohr-Coulumb relationship between shear strength and normal effective stress
4n	Total number of equations
Unknowns	**Variable**
1	FOS
n	Normal force at base of each slice, N'
n	Location of normal
n	Shear force at base of each slice, S_m
n - 1	Interslice force,Z
n - 1	Inclination of interslice force, θ
n - 1	Location of interslice force (line of thrust)
6n - 2	Total number of unknowns

Based on their assumptions, the common methods of analysis are divided. Table 2.2 lists the common methods of analysis and the conditions of static equilibrium that are satisfied in determining the factor of safety. Factor of safety is defined as the ratio of available strength to strength mobilized.

Table 2.2 Static equilibrium conditions satisfied by limit equilibrium methods

Method	Force-Equilibrium		Moment Equilibrium
	x	y	
Ordinary method of slice (OMS)	No	No	Yes
Bishop's simplified	Yes	No	Yes
Janbu's simplified	Yes	Yes	No
Corps of Engineers	Yes	Yes	No
Lowe and Karafiath	Yes	Yes	No
Janbu's generalized	Yes	Yes	No
Bishop's rigorous	Yes	Yes	Yes
Spencer's	Yes	Yes	Yes
Sarma's	Yes	Yes	Yes
Morgenstern-Price	Yes	Yes	Yes

The assumptions made by each of the methods to arrive at statical determinacy are summarized as follows.

Ordinary method of slices (OMS), Fellenius method is one of the simplest methods but it neglects all interslice forces and fails to satisfy force equilibrium for individual slices and slide mass.

Bishop's simplified method reduces the number of unknowns to (n-1) by assuming all interslice shear forces to be zero. The solution is over determined with (4n-1) unknowns, thus horizontal force equilibrium is not satisfied for a slice. (Abramson et. al.(1996))

Janbu's simplified method (1954, 1957, 1973) makes the same assumptions leading to an over determined solution that will not completely satisfy moment equilibrium conditions. A correction is introduced to account for the inadequacy.

Lowe and Karafiath's method (1960) assumes the interslice forces to be inclined at $\theta = \frac{1}{2}(\alpha + \beta)$ .See figure 2.2. Moment equilibrium is not satisfied and (4n-1) unknowns result.

Corps of Engineers' (1982) method considers interslice forces are either

(a) parallel to ground surface ($\theta=\beta$) or

(b) equal to the average slope angle between the left and right end points of the failure surface.

These result in an over-determined system with moment equilibrium not being satisfied.

Spencer's method (1967, 1973) satisfies equilibrium by assuming ((n-1) assumptions) that the resultant has an unknown constant inclination. This presents (4n-1) unknowns without counting the unknown inclination.

Bishop's Rigorous method (1955) assumes (n-1) interslice shear forces resulting in (4n-1) unknowns. This can give factor of safety without satisfying moment equilibrium. Therefore,

Bishop introduced an unknown. This unknown is unique distribution of interslice resultant force, which will rigorously satisfy the equilibrium equations.

Janbu's Generalized method (1954, 1957, 1973) assumes a location of the thrust line. This results in (4n-1) unknowns. Since this analysis does not satisfy, static equilibrium Janbu further suggested a unique thrust line that rigorously satisfies equilibrium.

Morgenstern and Price Method (1965) is similar to Spencer's method, except for the assumption that the inclination of the interslice resultant force varies according to a "portion" of an arbitrary function. Hence, 4n unknowns and 4n equations result.

Sarma's method (1973) calculates horizontal seismic coefficient needed to bring failure mass into limiting equilibrium using method of slices. A relationship between the seismic coefficient and presumed factor of safety is developed. Zero seismic coefficient corresponds to static factor of safety. An interslice force distribution function, which is similar to that of Morgenstern and Price, is used to calculate the seismic coefficient from the presumed factor of safety. Though this method satisfied all equilibrium conditions, the critical surface corresponding to static factor of safety is usually different from that which is determined by the more conventional approaches with unknown factor of safeties.

2.2.3 *Unified approaches*

A new procedure that uses Morgenstern-Price Method, called "Best-fit regression" has been developed. Its solution satisfies the same static equilibrium

conditions as that of the original solution by Morgenstern and Price which uses Newton-Raphson method. The solution procedure, however, is different. The numerical technique consists of an interactive approach in which initial values of shear forces are set to zero. Subsequent interactions use values of shear forces,

which are computed using assumed value for λ and a predetermined side force

function in equation 2.1

$$\frac{Z_s(X)}{Z_N(X)} = \lambda f(X) \text{ ---------- } 2.1$$

where

$Z_s(X)$ is interslice shear force function

Zn(X) is interslice normal force function

λ is presumed constant

f(X) is interslice force function

X is x-coordinate from toe to crest

The factors of safety relative to moment and force equilibrium are solved for a range of X values. The graph of factor of safety versus X is fit by a second order polynomial regression. The point of intersection identifies factor of safety satisfying both force and moment equilibrium.

The other methods fall as a special case of the "best fit method" (Rodriguez-Marek 1996). Bishop's method satisfies equilibrium with $\lambda = 0$, Spencer's with λ equal to the tangent of the assumed angle between the horizontal and the resultant interslice force. Janbu's factor of safety can be compared to other methods by observing for what value λ the force equilibrium equation gives the resulting factors of safety.

Factor of safety varies slightly for methods satisfying moment equilibrium. Methods that satisfy only force equilibrium, however, have a larger variation in resulting FS.

Espinoza, Repetto, and Muhunthan(1992) presented a unified framework for slope stability analysis for circular failure surfaces. Espinoza, Bordeau and Muhunthan (1994) extended it for general surfaces. The main advantage of this framework is that all intreslice shear forces are grouped in a single term, allowing an easier assessment of the influence of the internal forces on the final value of the factor of safety.

Different methods are included in the unified formulation by means of different assumptions on interslice forces (Rodriguez-Marek (1996))

Hypothesis I: A relationship between internal shear and normal force distributions Zs(X) and Zn(X) is assumed.

Hypothesis II: The relationship between Zs(X) and Zn(X) is derived by considering moment equilibrium of a single slice. This equation relates Zs(x) to internal moment M(x), which in turn can be related to the height of the line of thrust by

$$M(X) = Z_n(X)\ h(X) \quad \text{---------- (2.2)}$$

where h(X) is introduced as an assumption. In this hypothesis, both moment and force equilibrium are satisfied.

Hypothesis III: The shape of the internal shear force distribution $Z_s(X)$ is assumed but its magnitude is not. Correia (1988) made such an assumption. Sarma (1973) developed an equation based on soil strength parameters and geometric characteristics of the slope that also renders a distribution of $Z_s(X)$ independent of $Z_n(X)$

The unified formulation has a significant advantage for computer implementation of different techniques (Rodriguez-Marek (1996)). Varying the type of hypothesis and using a computer program developed for this method Espinoza et al (1994) performed a comparative study. They found that factor of safety is not affected by type of hypothesis for circular surfaces. For such instances, simplified methods accounting only for moment equilibrium (E.g. Bishop 1955) result in reasonable values of factor of safety while for non-circular surfaces variations in factor of safety are large. (Rodriguez-Marek (1996)).

Different researchers performed an extensive study on convergence problems in limit equilibrium using hypothesis similar to Hypothesis II. These researchers include Morgenstern and Price (1965), Spencer (1967), Ching and Fredlund (1983), Sharma (1994). Three of the most commonly encountered problems related to the method of slices are:

(a) An unreasonably large and/or negative normal force at the base of the slice. The problem is related to the parameter (Fredlund & Krahn 1977)

$$m_\alpha = \frac{\sec\alpha}{1 + \frac{\tan\alpha\tan\phi}{FS}} \qquad (2.3)$$

where α= inclination of failure surface at base of slice

$\varnothing$= Friction angle

FS= Factor of safety

The problem occurs when m_α approaches zero and/or goes negative. This results from the inclination of the failure surface at the ends. Classical earth pressure theory suggests that α, in the passive zone, should be limited by (Rodriguez-Marek (1996)):

$$\alpha = \frac{\varnothing}{2} - 45^0 \quad \text{----------(2.4)}$$

and in the active zone by

$$\alpha = \frac{\varnothing}{2} + 45^0 \quad \text{---------- (2.5)}$$

With these limitations, the problem is avoided.

(b) A negative normal force may be developed in a soil with high cohesive strength. Negative stresses develop in highly cohesive soils, because the high cohesive component of soil is much larger than other resistance forces. Therefore, for the slice to move downward along with the rest of the slope, the friction resistance must act in the direction of sliding. Hence, to satisfy equilibrium, the normal force becomes negative (Rodriguez-Marek (1996)).

This indicates that limit equilibrium alone is not sufficient in determining stability for highly cohesive soils. When tension develops at the top of a cohesive slope, the soil will likely develop tension cracks. Reducing the length of the slip surface to the bottom of the tension zone can eliminate the problem of negative normal stresses.

In some cases, for example in shallow slopes with high cohesion and unsaturated slopes with matrix suction, negative stresses develop below the tension zone. For such cases, using the calculated negative stresses in the analysis results in a more reasonable solution than changing all negative stresses to zero. This second option results in a solution that does not satisfy equilibrium.

(c) Problems are also encountered when unreal assumptions on the force distribution are introduced, which usually lead to a lack of convergence to a solution. Difficulties arise because boundary conditions (the side force must be zero or some prescribed value at end slice) cannot be satisfied with certain assumptions. As mentioned before, convergence related problems are usually associated with Hypothesis II (e.g. Janbu's method).

2.3 Factor of safety and Critical surface search

2.3.1 Introduction

The purpose of slope stability analysis is to determine the stability of a soil mass. Factor of safety has been the traditional quantitative measure of this stability. This element of slope stability analysis is considered somewhat archaic because of the introduction of new concepts such as progressive failure. However, the definition of FS is a key step in the development of slope stability analysis. Thus, the current definition of FS needs to be treated.

The FS is tied to the determination of the critical failure surface. Most analysis methods are restricted by the requirement of a predetermined failure surface. Therefore, they are applied with algorithms that search for minimum FS.

The following sections review various measures of stability and determination of critical surface.

2.3.2 The Factor of Safety

For most limit equilibrium methods, FS is defined as the ratio of available shear strength to the shear stress mobilized for maintaining equilibrium (Morgenstern and Price (1965) & Duncan (1996))

There are also other definitions of FS although the one mentioned above is very common. FS can be related to surcharge loads, slope geometry or as the factor by which the soil unit weight must be increased to cause failure (Naylor 1981), according to Rodriguez-Marek (1996)

Rodriguez-Marek (1996) reported the use of probability of failure by Gudehus (1988) but its practicality is strongly questioned by practicing engineers.

Since FS is not constant, a global factor of safety can be determined by using the distribution of failure strength and mobilized stress along the slope.(Rodriguez-Marek (1996))

$$FS = \frac{\int_L \tau_f \, dL}{\int_L \tau_n \, dL} \qquad (2.6)$$

where τ_f = failure shear strength

τ_n = shear stress in slope

L = length along failure surface

The stress distribution in equation (2.6) above can be obtained from FEM solution (Duncan 1996).

2.3.3 Critical Surface Search

Different methods are available for the determination of the critical surface. They are either based on optimization techniques in combination with limit equilibrium methods or using more elaborate analysis that yield failure surfaces for which FS can be found.

According to Greco (1996), nonlinear programming has been frequently used for locating the critical slip surface in slope-stability analysis by means of safety-factor minimization. Following this approach, another investigator proposed the use of dynamic programming, while other investigators utilized the alternating variable method. Successively, still other authors used the conjugate-gradient method. Other methods employed by different authors include the simplex method and the sequential unconstrained minimization technique.

Until the early 1960s, when the first deterministic methods of nonlinear programming began to emerge, the grid method or Monte Carlo methods were the only means of searching for the minimum of function of several variables. The Monte Carlo methods are techniques of random search that are very simple in structure. In fact, they are based on a random generation of trial solutions.

According to Zhu (2001), in the presence of weak layers or discontinuities, the implementation of these methods into computer programs is less efficient and the results are largely dependent on the initially assumed position of the slip surface. The random search

method has been occasionally used in determining critical sip surface (Boutrup and Lovell 1980; Greco 1996). Though effective in some cases, this method can only be considered as an ad hoc method due to its poor theoretical background.

Several researchers attempted to apply variational calculus to the determination of the position of the critical slip surface. Zhu (2001) further commented that this method is mathematically complex and limited to very simple slopes. According to him, De Josselin De Jong (1980) has found that the slip surfaces computed by the calculus of variations corresponds to no extremum at all because of the inadequate formulation of slope stability problems.

Zhu (2001) presented a new concept referred to as the critical slip field (CSF). It combines the simplified Janbu method with the principle of optimality. The CSF method is used for locating the critical slip surface with respect to the method of slices satisfying both force and moment equilibrium.

2.4 Finite Element Methods for Slope Stability Analysis

2.4.1 Introduction

The finite-element method was introduced to the geotechnical engineering profession by Clough and Woodward in 1967), in a paper written on the stability and performance of slopes for the first Berkley conference. The most outstanding feature of their paper was the use of nonlinear stress-strain relationships in their analyses of an embankment dam.

Their finding made it apparent that the ability to consider nonlinear stress-strain behavior gave the FEM great potential for use in geotechnical engineering problems.

2.4.2 Advantages and Limitations of FEM

FEM is a general purpose method and has been used to:

A. calculate stresses, movements and pore pressures in embankments and slopes.

B. analyze conditions during construction, such as progress of consolidation or swelling and dissipation of excess pore water pressures.

C. investigate the likelihood of cracking, hydraulic fracturing, local failure, and overall stability of slopes.

The method is so general that it is possible to model many complex conditions with high degree of realism, including in the analyses such things as nonlinear stress-strain behavior, non-homogeneous conditions, and changes in geometry during construction of an embankment or any excavation.

The generality and power does not come without a compromise. The effort and the cost of finite-element analyses are high. Each analysis takes a considerable amount of engineering time to develop property values, to perform the computer analyses, and to evaluate the results. The amount of engineering time required has been reduced through development of graphical preprocessors and postprocessors, but still is very significant .In addition, a considerable amount of time is required to learn to use the method effectively.

The cost of computer time for finite-element analyses has been reduced greatly by the fact that they can be performed on the powerful new microcomputers that are now available. This has not had a very significant effect on the overall cost of performing analyses, however, because even when computer time was an order of magnitude more expensive than it is now, its cost was typically no more than about 10% of the total cost of an analysis.

Although some types of finite-element analyses can now be performed on microcomputers, mainframe computers are required for problems that involve large numbers of elements, three-dimensional analyses, and iterative techniques that involve extremely large number of calculations for accurate simulation of nonlinear behavior.

Comparison of the results of FEM analyses with field measurements have shown that there is a tendency for calculated deformations to be larger than measured deformations (Duncan 1996). The reasons for this difference include:

A. soils in the field tend to be stiffer than soils at the same density and water content in the lab because of aging effects,

B. average field densities are higher than the specified minimum dry density, which is often used for preparing lab triaxial test specimens,

C. samples of insitu materials suffer disturbance during sampling and are less stiff as a result and

D. many field conditions approximate plane strain, whereas triaxial tests are almost always used to evaluate stress-strain behavior and strength.

2.4.3 Incremental Analyses

Incremental analyses techniques have been very crucial in realistic modeling of slopes using FEM. These involve simulating the overall problem as a series of event, and analyzing each event as a simple linear problem.

Incremental analyses are advantageous in modeling of slopes because they allow analyses with changes in geometry (e.g. construction of an embankment, the increment stages being modeled as addition of layer of elements to the mesh) and more realistic simulation of

nonlinear stress-strain behavior of soil by changing the stiffness values assigned to each element during each increment of the analysis.

2.4.4 Information Required for Analyses

Finite-element analyses require definition of initial conditions, stress-strain properties and the construction or loading sequence.

Initial stresses are needed for three reasons. First, in incremental analyses, the changes in stress calculated during each increment are added to the stresses at the beginning of the increment to evaluate the stresses at the end. Second, the stiffness of the soil depends on the stress in the soil. Third, in analyses of excavation, the forces that are applied to simulate excavation of the soil are calculated using the stresses before excavation on the boundary of the excavation. Initial stresses are usually approximated. One simple procedure that has been used is to perform a gravity turn-on analysis (applying vertical forces representing the weight of the material to an initially unstressed mesh), and then change the horizontal stresses to be equal to K_o (coefficient of earth pressure at rest) times the calculated vertical stresses, using values of K_o based on the best information available.

The reference state of stresses must satisfy equilibrium. The reference state for displacements and strains are arbitrary. Therefore, it is preferable to relate the values of stress-strain parameters used in nonlinear stress-strain relationships to stresses rather than strains.

The stress-strain properties of the soils play a critical role in finite-element analyses. In some simple cases where soils are not stressed close to failure, and where strains are small, it is possible to represent soils as linear-elastic materials. More often, it is necessary to use stress-

strain relationships that account for nonlinear behavior, and that account for the fact that soil modulus values vary with confining pressure.

A finite element model of a slope should simulate the actual construction or loading sequence of the problem being analyzed. Adding elements to simulate fill placement, removing elements to simulate excavation, and applying loads in increments can be modeled realistically. Other processes that can be modeled include raising or lowering water levels within a soil mass, changing the temperature of structures and consolidation.

In designing a finite-element mesh, it is important to consider the number of steps to be used in the analysis as well as the significant geometric features that need to be included. In the problems that model nonlinear behavior or changes in geometry, the accuracy of the analyses is affected by the number of steps used. Studies have shown that about eight layers are enough to model construction of an embankment or an excavation. Acceptable accuracy can sometimes be achieved with fewer layers. The number of steps needed for accuracy is independent of the embankment height or the excavation depth.

2.4.5 Stress-strain Relationships Used in Practice

Selecting an appropriate soil stress-strain relationship is primarily involved with balancing simplicity and accuracy. Which type of relationship is most suitable for a given case depends on the conditions being analyzed and the purpose of the analysis. While it seems reasonable that more complex stress-strain relationships should be able to model the behavior of soils

more accurately, there is no benefit in using a very complex relationship to analyze a problem where the simplest representation or the stress-strain behavior of the soil would result in acceptable accuracy. In order of increasing complexity, the choices of stress-strain relationships include linear elastic, multilinear elastic, hyperbolic (elastic), elastoplastic, and elastoviscoplastic and perfect plastic with an associated flow rule.

The associated flow rule defines the plastic strain rate by assuming the yield function to coincide with plastic potential function. This function is in turn used to arrive at the plastic strain rate, which in turn is used in virtual work equations. Kim, Salgado and Yu(1999) used the principle of virtual work in conjunction with lower- and upper-bound theorems to compute lower and upper bounds to stability factors of slopes subject to pore water pressures.

Each of the stress-strain relationships has its own advantages and limitations. Linear elastic stress-strain relationships have the advantage of simplicity, and the limitation that they only model the behavior of real soils well at low stress levels and small strains. Multilinear elastic stress-strain relationships have the advantage that they can be used to model any shape of stress-strain curve for ductile materials, and the limitation that they must be developed on a case-by-case basis to approximate the particular stress-strain characteristics of the soils under consideration. Hyperbolic stress-strain relationships have the advantages that they model nonlinear behavior, and that the parameters involved have physical significance and can be evaluated using the results of conventional triaxial tests. They have the limitation that they are inherently elastic, and do not model plastic deformations in a fully logical way. Elastoplastic and elastoviscoplastic stress-strain relationships have the advantage that they model more realistically the behavior of soils close to failure, at failure and after failure. Perfect plastic with associated flow rule is a reasonable assumption if the loadings associated with very large displacements are of concern. In addition, theoretical studies show that the collapse loads for

earth slopes, where soils are not heavily constrained, are quite insensitive to whether the flow rule is associated or nonassociated. The last three relationships have the limitation that they are more complex.

2.5 Discrete Element Methods

2.5.1 Introduction

The Discrete Element Method (DEM) was developed by Cundall (1971, 1974) for the study of Rock Mechanics problems. The method was enhanced later (Cundall and Strack 1979). The DEM has been used extensively to study physical and geotechnical phenomena such as mechanisms of deformation, constitutive relations for soil, stability of rock masses, flow of granular media, ground collapse, and other types of geotechnical phenomena.

In the DEM, the medium under study is divided into discrete elements with arbitrary shapes. The interaction of these elements is viewed as a transient problem with states of equilibrium developing whenever the internal forces are balanced (Cundall and Strack (1979)). The calculation cycle alternates between the sum of forces acting on an element resulting from a force-displacement law at the contacts, and the application of Newton's law to find the incremental displacements, velocities, and accelerations of the element. The time step is chosen small enough so that disturbances do not propagate to more than the adjacent particles on each time step, and accelerations can be assumed constant during that time step (Cundall and Strack 1979).

There are a series of computer codes available in the literature for the application of the DEM. Most follow the pioneering work by Cundall (1974, 1978), but vary either in the modeling of the contact forces or in the solution algorithm. The first DEM code

was developed for the study of rock mass behavior by Cundall (1971, 1974). Cundall also developed the Universal Discrete Element Code (UDEC) to model jointed rock mass and BALL and TRUEBALL to study granular media.

The UDEC code was later enhanced by other authors to include dynamical inputs. DISC, a model similar to BALL, was developed in the mid 1980s for the study of applications of DEM to geotechnical engineering. Extensive research on DEM-like models has also been performed in the same decade by other researchers. Three-dimensional DEMs have been developed in the early 1990s.

The previously described methods are essentially dynamic in nature. Static variations of the DEM were developed by Chang (1992) for slope stability analysis. The traditional (dynamic) DEM is not discussed here. However, the static DEM is described in detail in the following sections.

2.5.2 Static DEMs

As indicated before, traditional DEMs are dynamic by nature. This characteristic is desirable in problems that involve large displacements or time-dependent processes, such as rapid granular flows. Their disadvantages appear when equilibrium is needed. Since these models neglect the restricting effect of neighboring grains and the total structure when calculating the displacement of individual grains, displacements are usually over-estimated and the calculated state "oscillates" around the exact solution.

Chang (1992) presented a static DEM for the study of slope stability. The method overcomes many limitations of traditional slope stability analysis, and yet is simple to understand and easy to implement. In this method, a failure surface is chosen and the failure mass is divided into slices. A force-displacement relationship is then specified at the failure surface and inter-slice boundaries. The force-displacement relationship is governed by elastoplastic springs that yield according to the Mohr-Coulomb criterion. The solution proceeds by applying equilibrium and displacement compatibility, and solving for the resulting interslice forces. A more detailed derivation of the method will be presented in Chapter 3.

Chang's method replaces the usual assumptions of interslice force distribution and constant factor of safety used in the LEM by force-displacement equations for the soil. This results in much more realistic force distributions along the failure surface which, in turn, allows for a realistic determination of a local factor of safety. The correct evaluation of localized failures permits the study of the mechanism of progressive failure. The method allows for easy manipulation of soil parameters and lends itself to the study of the behavior of brittle soil slopes, in which the progressive failure mechanism is a critical factor.

A comparison of Chang's method with other slope stability methods is presented by Chang (1992). The factors of safety obtained with this model are generally lower than those from LEMs, due to the consideration of progressive failure. The stress distributions obtained with these methods compare well with the finite element studies of Lo and Lee(1973).

2.5.3 Geotechnical Applications

The model of a granular assembly described by Cundall and Strack (1979) was compared qualitatively with an experimental setup with photoelastic discs by De Josseling De Jong and Verruijt (1969).The authors concluded that the DEM is a valid tool for research into the behavior of granular assemblies.

Other researchers used their adapted version of the algorithm developed by Cundall for the study of jointed rock masses under seismic loading. The DEM has also been used extensively for the study of granular assemblies. Still other authors used a variation of Cundall's DEM in which they introduced a rebound coefficient as a means to dissipate the system's energy in place of a viscous dashpot in the interparticle contact. The authors used this model for various simulations such as slope failure resulting from seismic loading, dynamic response of a structural foundation, analyses of dynamic earth pressure of loose ground under dynamical excitation, and soil behavior surrounding a pile penetrating into the ground. Slope failure simulation, analysis of dry rock avalanche, detailed granular shear flow study have also been carried out using DEM. Cundall and Strack(1979) recommended the DEM as a modeling tool to study constitutive laws for soils.

The feasibility of using DEM for numerically simulating the behavior of cohesive soils has been studied by Anandrajah (1994). Particle bending and physiochemical forces and especially the double-layer repulsive force and Vander Waals' forces were studied. He found that the use of DEM for study of cohesive particles resulted in reasonable qualitative trends and patterns. (E.g. relation between e and σ_v)

2.6 Summary

LEMs are the most widely used methods for slope stability analysis. Results of

these methods are backed by long experience in design and analysis of slopes. On the other hand, LEMs make many simplifying assumptions regarding stress distributions. This shortcoming makes them unreliable for the study of progressive failure.

FEMs have been used successfully for the deformation analyses of slopes. However, FEMs are computationally intensive, and they do not provide a single value for the measure of safety. In view of this, different methods to quantify slope safety have been discussed.

DEMs are methods in which the medium under study is divided into discrete blocks. Most DEMs are dynamic in nature, but static variations of the DEM apply better to geotechnical phenomena. The method developed by Chang (1992) is highlighted as important in the study of progressive failure in slope stability.

The concepts reviewed in this chapter present a necessary overview of the state of the art in slope stability analyses and DEMs. In the next chapter a static DEM for the study of slope stability will be introduced. The method generalizes Chang's method to a series of blocks rather than slices. The method presented is not constrained by a failure surface, thus it represents a large improvement for geotechnical practice. The concepts reviewed in this chapter provide a strong basis for evaluating the performance of the method presented in chapter 3.

CHAPTER THREE

DEM Modeling of Slopes

3.1 Introduction

Stability analysis of slopes has been traditionally performed using LEMs. The assumptions used in these methods do not always result in realistic analysis. The assumption of a constant FS throughout the slope (i.e., that the slope reaches failure simultaneously along the whole failure surface) is unrealistic. These constraints render LEMs useless for the analysis of progressive failure. On the other hand, the FEM is useful to

examine the development of the failure surface. However, FEMs are computationally intensive and costly for routine applications.

In the proposed model, the slope is discretized into a number of discrete blocks. A stiffness matrix is derived from block boundary behavior, equilibrium considerations, and compatibility. These relationships replace traditional assumptions of LEMs. The stiffness matrix is used to find a statically acceptable solution for block displacements from the loads applied to the blocks. The resulting displacements are then used to recalculate the stiffness matrix for the local yielding cases. Convergence occurs when no change in the stiffness matrix occurs from one iteration to the next. For unstable slopes, blocks in the failure mass have large displacements. The failure surface can be found from the formation of a displacement discontinuity between safe and failed blocks. For stable slopes, the rotations of principal stresses can be used as indicators of likely failure. surfaces. The method is simple and computationally inexpensive. Chang (1992) developed a special case of the proposed method.

The model is derived accounting for the interaction of two blocks. The interface force-displacement relationships are first introduced to develop the local stiffness matrix for the boundary between the two blocks. Consideration of block equilibrium is then used to find the relationship between displacements of each block to the forces applied to its centroid. Displacement compatibility is then used to assemble the global stiffness matrix for a system of blocks. Finally, the application of boundary conditions is presented. The techniques used to handle numerical instability in the computation are also discussed.

3.2 Mechanics of Block Interaction

3.2.1 Development of the global stiffness matrix

A. Interface behavior.

The slope is discretized into a series of blocks joined together by Winkler springs as shown in Figure 3.1. Load deformation behavior is modeled by elasto-plastic normal and shear stiffness (Kn and Ks) Winkler springs. The normal springs do not yield in compression, but they do in tension, beyond the tensile capacity of the soil.

This "tension cutoff" is provided for cohesive soils but not for cohesionless soils. After reaching the tension cut-off, the normal soil resistance is assumed to drop immediately to zero (Figure 3.2a).

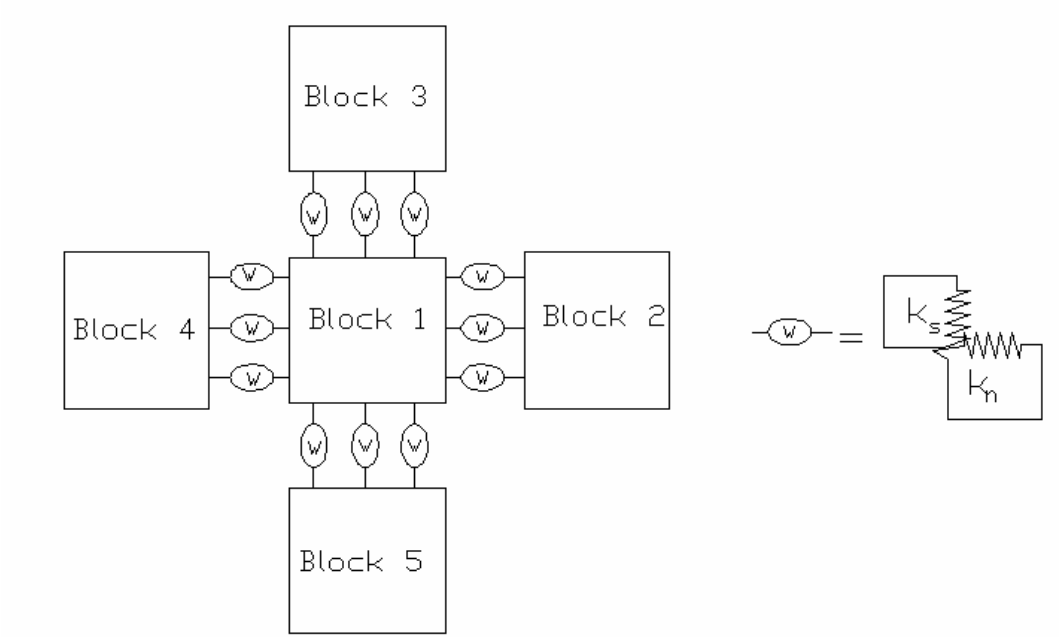


Figure 3.1 Block interface(Schematic diagram)

The shear springs yield when the peak shear strength is reached. For soils, the peak strength is governed by the Mohr-Coulomb criterion:

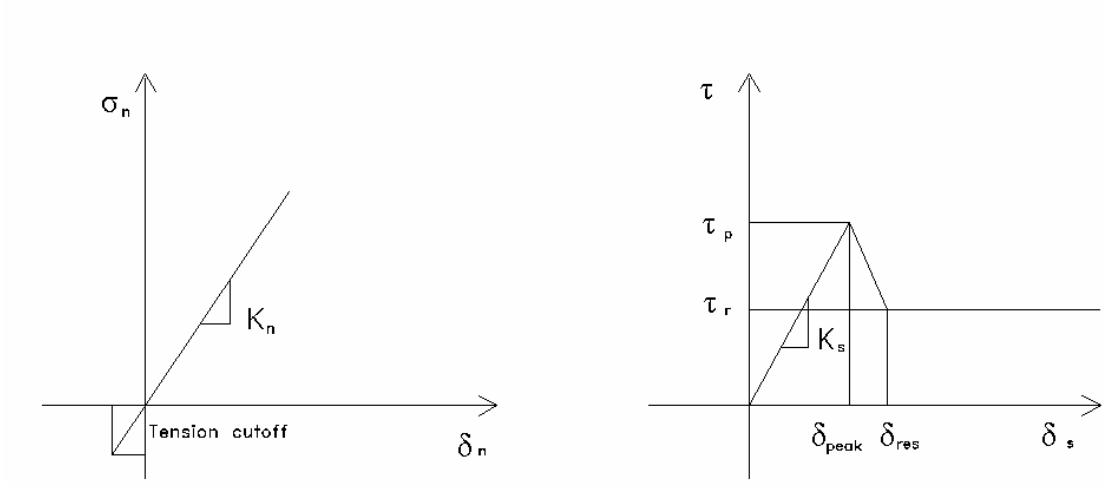
$$\tau_p = c'_p + \sigma'_n \tan \phi'_p \quad \text{----------} \quad (3.1)$$

where τ_p is the peak strength, C_p is the effective peak cohesion, σ'_n is the effective normal stress, and ϕ'_p is the effective peak friction angle. The interface shear behavior of the soil is shown in Figure 3.2b.

Beyond peak strength, the soil reaches a residual state in which strength is governed by:

$$\tau_r = c'_r + \sigma'_n \tan \phi'_r \quad \text{----------(3.2)}$$

where τ_r, c'_r and ϕ'_r are the effective residual strength, cohesion, and friction angle, respectively.



(a) (b)

Figure 3.2. Behavior of normal and shear Winkler springs (a) normal Winkler spring.(b) shear Winkler spring

For plastic non strain-softening soils, the residual and peak strength parameters are the same. When displacements exceed peak displacement, δ_{peak}, the stiffness is changed to the secant stiffness at that point. Details of the secant stiffness calculation are

given in Chapter 4. The input parameters needed to define the model shown in Figure 3.2 are the Mohr-Coulomb peak and residual parameters (c'_p, c'_r, ϕ'_p, and ϕ'_r), the shear and normal initial stiffness (K_n and K_s), the displacement needed for the onset of residual strength δ_{res} and displacement at peak strength δ_{peak}. The residual strength is achieved only at large deformations .

B. Development of the local stiffness matrix.

The equilibrium and compatibility equations are derived for two interactive blocks A and B (Figure 3.3). P is the midpoint of the interface between these two blocks, and $\vec{r}^{ip}$ is a vector pointing from the centroid of block i to the point P. The vectors representing centroid displacements of blocks A and B are $\vec{u}^{a}$ and $\vec{u}^{b}$, respectively. These vectors include the block displacements in global x and y directions (u_x, and u_y), and the block rotation (u_ϖ). $\vec{\Delta P}$ represents displacements of point P, with components in the global x and y directions (ΔP_x and ΔP_y), and a rotational component ΔP_ϖ.

It is assumed that the block displaces as a rigid body, and all displacements are in the x-y plane (i.e., the slope is modeled in 2 dimensions). Then, the displacement of point P resulting from the rotation of block A can be expressed as (Figure 3.4):

$$\left(\Delta \vec{P}^{a}\right)_{\omega} = \vec{\omega}^{a} \times \vec{r}^{ap} \quad \text{----------3.3}$$

where,

$$\vec{\omega}^{a} = u_{\omega}^{a}\vec{k} \quad \text{----------(3.4)}$$

and $\vec{k}$ is a unit vector perpendicular to the x and y axis (right-hand rule).

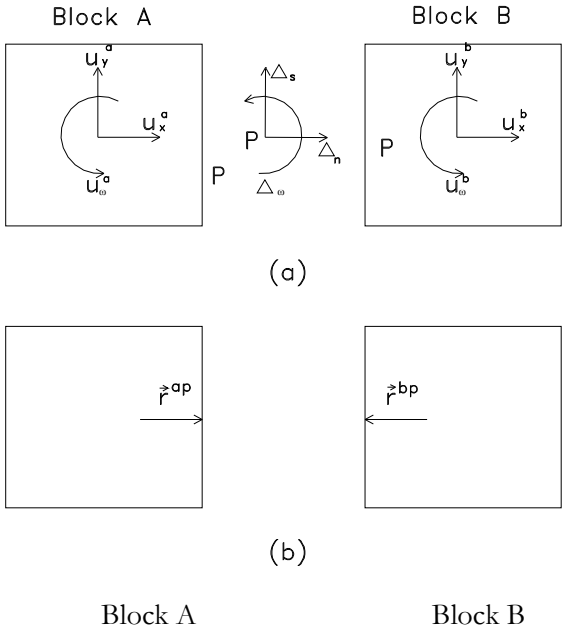


Figure 3.3 Block displacements and position vectors. (a) centroid and P point displacements (b) position vectors $\vec{r}^{ap}$ and $\vec{r}^{bp}$

Let $\vec{r}^{\,ap}_{\,x}$ and $\vec{r}^{\,bp}_{\,y}$ denote the x and y components of $\vec{r}^{\,ap}$. Expanding Equation (3.3),and making use of Equation (3.4), results in:

$$\left(\Delta \vec{P}^{\,a}\right)_{\omega} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ 0 & 0 & u^a_\omega \\ r^{ap}_x & r^{ap}_y & 0 \end{vmatrix} = -r^{ap}_y u^a_\omega \vec{i} + r^{ap}_x u^a_\omega \vec{j} \text{ ----------(3.5)}$$

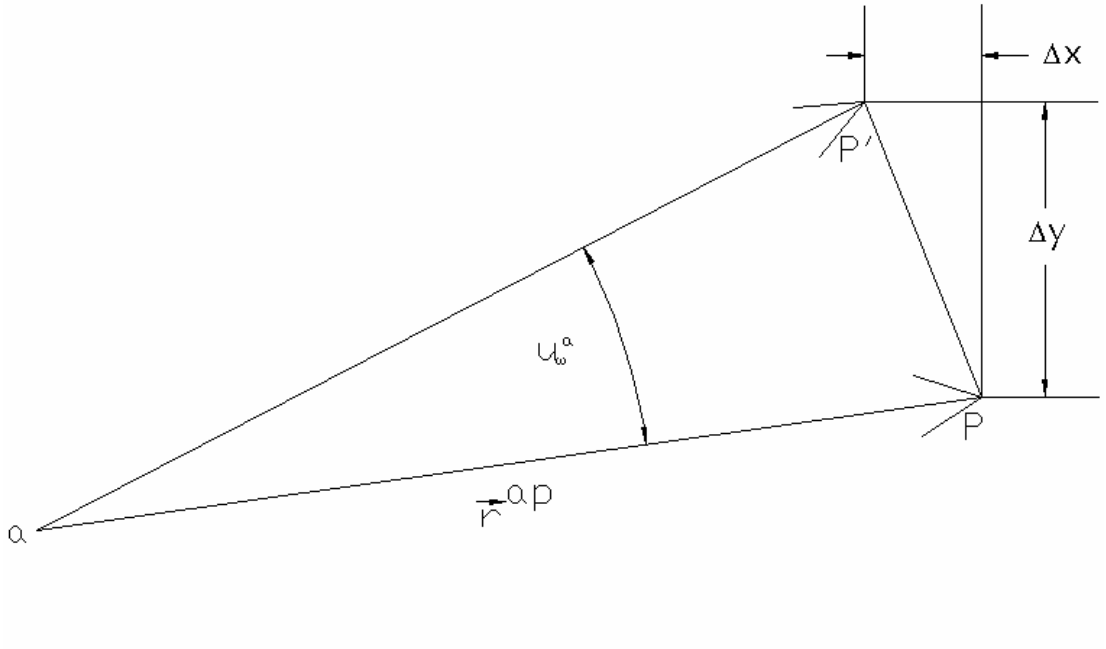


Figure 3.4. Vector diagram for block rotation.

where $\vec{i}$, $\vec{j}$, and $\vec{k}$ are coordinate unit vectors. Similarly, the displacement of point P due to the rotation of block B is given by:

$$\left(\Delta \vec{P}^{b}\right)_{\omega} = -r_y^{bp} u_\omega^b \vec{i} + r_x^{bp} u_\omega^b \vec{j} \quad \text{---------- (3.6)}$$

Because of the rigid body assumption, the displacement of point P resulting from the translational displacements of block A in the x and y global directions is simply given by:

$$\left(\Delta \vec{P}^{a}\right)_{disp} = u_x^a \vec{i} + u_y^a \vec{j} \qquad \text{----------(3.7)}$$

The same applies for the displacements of point P due to the translational displacements of block B. The total displacement of point P from the motion of block A results from the combination of the effects of rotation (Equation 3.5) and displacement (Equation 3.7). Now, let the vector $\vec{\Delta}^{p}$ denote the displacement of block B relative to block A at point P. This vector has displacement components in the x and y directions, $\vec{\Delta}^{p}_{x}$ and $\vec{\Delta}^{p}_{y}$, as well as a rotational component, $\vec{\Delta}^{p}_{\omega}$. Combining Equations (3.5), (3.6), and (3.7) results in:

$$\left(\vec{\Delta}^{P}\right) = \Delta \vec{P}^{b} - \Delta \vec{P}^{a} = \left(\Delta \vec{P}^{b} - \Delta \vec{P}^{a}\right)_{disp} + \left(\Delta \vec{P}^{b} - \Delta \vec{P}^{a}\right)_{\omega} \quad \text{----------(3.8)}$$

or, in matrix form (see also Chang 1992).

$$\begin{Bmatrix} \Delta_x^p \\ \Delta_y^p \\ \Delta_\omega^p \end{Bmatrix} = \begin{bmatrix} 1 & 0 & -r^{bp} \\ 0 & 1 & r^{bp} \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u_x^b \\ u_y^b \\ u_\omega^b \end{Bmatrix} - \begin{bmatrix} 1 & 0 & -r^{ap} \\ 0 & 1 & r^{ap} \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u_x^a \\ u_y^a \\ u_\omega^a \end{Bmatrix} \quad \text{----------(3.9)}$$

A local coordinate system is useful to characterize the interface behavior. The

unit vectors ($\vec{n}$ and $\vec{s}$) in local coordinate axes point in a direction perpendicular and parallel to the block interface (Figure 3.5). The transformation matrix λ from global to local axes is then expressed by:

$$\lambda = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{----------(3.10)}$$

where α is the angle between the x and n axes (see Figure 3.5). Note that λ is orthogonal, therefore $\lambda^T = \lambda^{-1}$

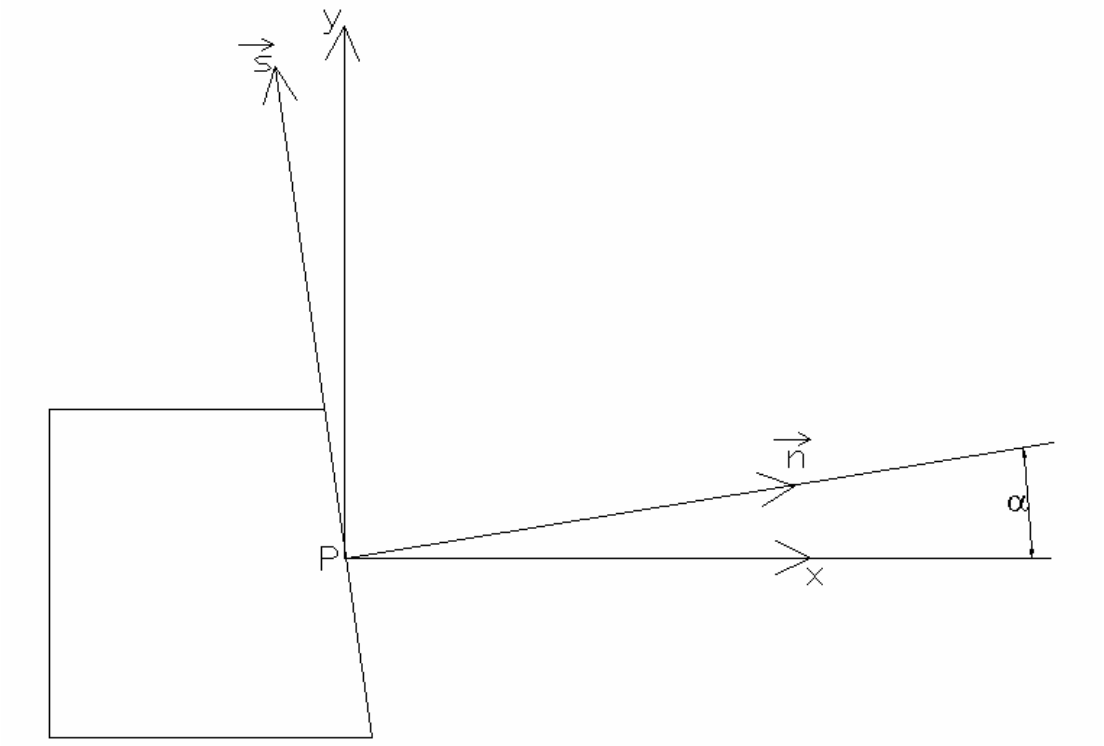


Figure 3.5. Local and global axes.

From now on, forces and displacement vectors are given either in the global or local coordinate system. Subscripts x and y on vector components indicate that the vector

is expressed in global coordinates, while subscripts n and s indicate that the vector is expressed in the local coordinate system. Similarly, the subscript L on a stiffness matrix denotes local coordinates, while a subscript G denotes global coordinates.

If P' is any point along the block interface, located a distance *l* from point P (Figure 3.6), the relative movements of the two interacting blocks at point P' on the interface result in spring displacements in the normal direction, δ_n, and in the shear direction, δ_s, given by:

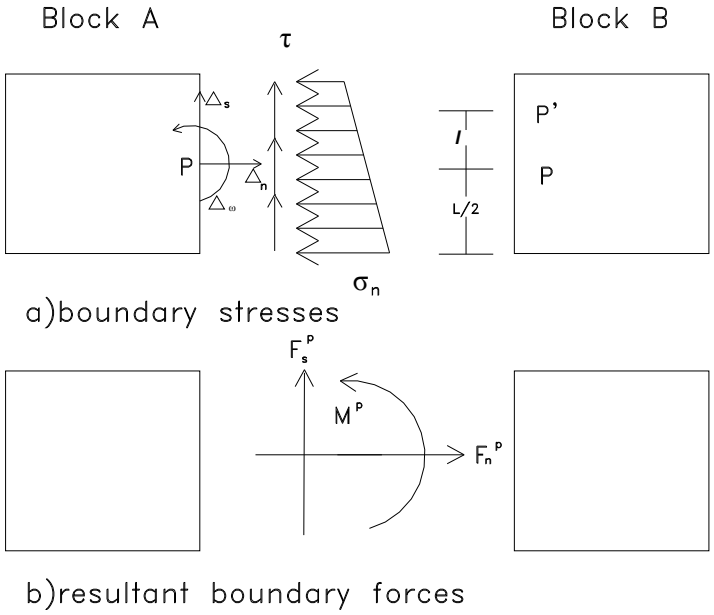


Figure 3.6. Stresses and forces in block boundaries.

$$\delta_n = \Delta_n^p + \Delta_\omega^p l \quad \text{----------} \quad (3.11a)$$

$$\delta_s = \Delta_s^p \quad \text{----------} \quad (3.11b)$$

where *l* is the distance from the center point P to point P'.

The spring displacements, in turn, generate normal and shear stresses on the block interface as shown in Figure 3.6. The stresses are governed by the force-displacement relationships in Figure 3.2, and are given by:

$$\tau_s = k_s \delta_s \text{ ---------- (3.12a)}$$

and

$$\sigma_n = k_n \delta_n \text{ ----------(3.12b)}$$

where τ_s and σ_n are the shear and normal stresses, respectively. Since the blocks are assumed to rotate as rigid bodies, Equations (3.12a) and (3.12b) result in a linear stress distribution along the block boundary. The resultant forces on point P are obtained by integrating the stresses on the block boundary (Chang 1992):

$$F_n^P = \int_{-L/2}^{L/2} k_n \delta_n dl = \int_{-L/2}^{L/2} k_n \Delta_n dl + \int_{-L/2}^{L/2} k_n \Delta_\omega d \text{ ----------(3.13)}$$

$$F_s^P = \int_{-L/2}^{L/2} k_s \delta_s dl = \int_{-L/2}^{L/2} k_s \Delta_s dl \text{ ----------(3.14)}$$

$$M^P = \int_{-L/2}^{L/2} k_n l \delta_n dl = \int_{-L/2}^{L/2} k_n l \Delta_n dl + \int_{-L/2}^{L/2} k_n l^2 \Delta_\omega dl \text{ ----------(3.15)}$$

where F_n^P, F_s^P, and M^P are components in the local coordinate system of the force-moment vector $\vec{F}^P$ (Figure 3.6). This vector represents the forces and moments at point P due to the displacements of adjacent blocks A and B. Evaluating the integrals in Equations (3.13)-(3.15) results in:

$$\begin{Bmatrix} F_n^P \\ F_s^P \\ M^P \end{Bmatrix} = \begin{bmatrix} K_n & 0 & 0 \\ 0 & K_s & 0 \\ 0 & 0 & K\omega \end{bmatrix} \begin{Bmatrix} \Delta_n^P \\ \Delta_s^P \\ \Delta_\omega^P \end{Bmatrix} \text{ ----------(3.16a)}$$

or

$$\left\{\vec{F}_L^{\,P}\right\} = [K_L]\left\{\vec{\Delta}_L^{\,P}\right\} \text{----------(3.16}b\text{)}$$

where $K_n=k_nL$, $K_s=k_sL$, $K_\omega=(k_nL^3)/12$, and $[K_L]$ is the local stiffness matrix.

*C. Equilibrium of block*s.

Now to consider force and moment equilibrium of each block, Equation (3.16b) is rewritten in global coordinates. Note that:

$$\left\{\vec{F}_G^{\,P}\right\} = [\lambda]^T\left\{\vec{F}_L^{\,P}\right\} \text{----------(3.17)}$$

and

$$\left\{\vec{\Delta}_L^{\,P}\right\} = [\lambda]\left\{\vec{\Delta}_G^{\,P}\right\} \text{----------(3.18)}$$

Substituting for $\vec{F}_L^{\,p}$ in Equation (3.17) using Equation (3.16b), and for $\vec{\Delta}_L^{\,p}$ of Equation (3.16b) using Equation (3.18) results in:

$$\left\{\vec{F}_G^{\,P}\right\} = [K_G]\left\{\vec{\Delta}_G^{\,P}\right\} \text{----------(3.19)}$$

where K_G is a stiffness matrix in the global coordinate system that relates the

displacements of point P to the resulting forces at that point, given by:

$$\{K_G\} = [\lambda^T][K_L][\lambda] \text{ ---------- (3.20)}$$

Consider the case in which block A is in contact with other blocks along all its boundaries, as shown in Figure 3.1. The boundary forces resulting from Equation (3.19) act at P in each boundary, and are denoted by $\vec{F}^{p}$, where the superscript p indicates the point at which the force is applied. For ease of notation, the subscript G from force and displacement vectors in the global coordinate system will be omitted from this point on. Let the force-moment vector f^a represent the forces applied to the centroid (point a) of the block, with components in x and y global directions (f_x^a and f_y^a), and a moment component represented by f_ω^a. The free body diagram of this block is shown in Figure 3.7.

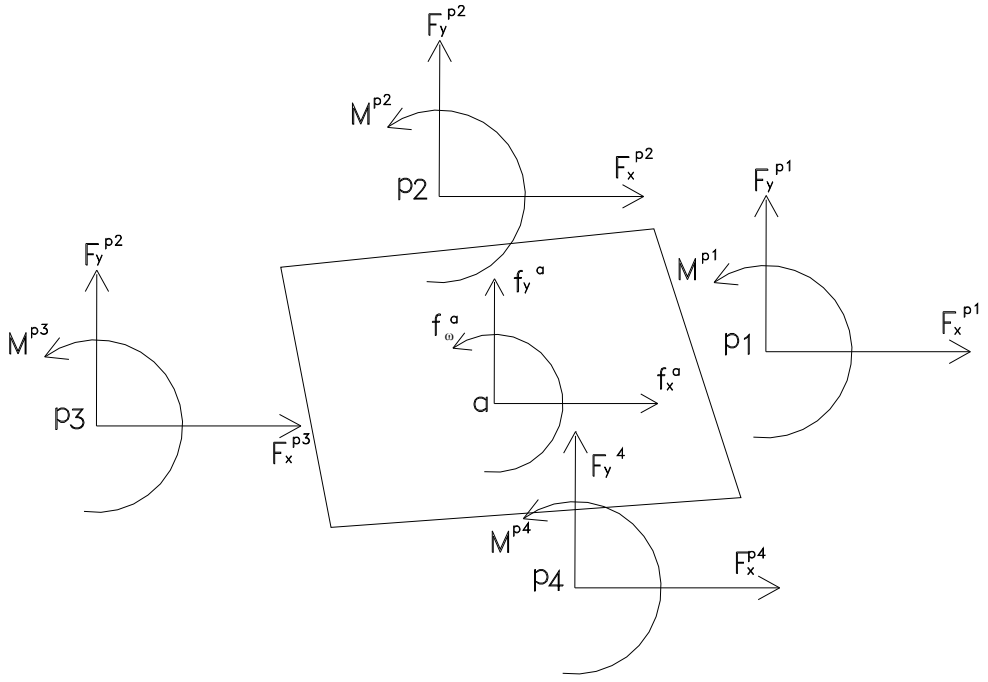


Figure 3.7 Free body diagram of a typical block A.

Applying force equilibrium in the x and y directions for the block results in:

$$\sum F_x = f_x^a + \sum_P^N F_x^p = 0 \quad \text{----------} \quad (3.21)$$

$$\sum F_y = f_y^a + \sum_P^N F_y^p = 0 \quad \text{----------} \quad (3.22)$$

and, applying moment equilibrium about the centroid of the block results in:

$$\sum M_a = f_\omega^a + \sum_P^N M^p - \sum_P^N r_y^{ap} F_x^p + \sum_P^N r_y^{ap} F_y^p \quad \text{----------} \quad (3.23)$$

In Equations (3.21)-(3.23), N is the total number of sides of block A, and p is a dummy summation index corresponding to each point P in block A. Rewriting Equations (3.21)-(3.23) in matrix form:

$$\begin{pmatrix} f_x^a \\ f_y^a \\ f_\omega^a \end{pmatrix} = \sum_P^N \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ r_y^{ap} & -r_x^{ap} & -1 \end{bmatrix} \begin{Bmatrix} F_x^p \\ F_y^p \\ M^p \end{Bmatrix} \quad \text{----------} \quad (3.24)$$

Substituting for the vector $\vec{F}^p$ from Equation (3.19) results in:

$$\left\{ \vec{f}^{\,a} \right\} = \sum_P^N \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ r_y^{ap} & r_x^{ap} & -1 \end{bmatrix} [K] \left\{ \vec{\Delta}^{\,p} \right\} \quad \text{----------} \quad (3.25a)$$

Note that the displacement vector, $\vec{\Delta}^p$, in the last equation is given by Equation (3.9).

D. Application of displacement compatibility.

Consider the compatibility between adjacent blocks A and B (Figures 3.3 or 3.6). Equilibrium consideration of block B results in:

$$\left\{ \vec{f}^{\,b} \right\} = \sum_{P}^{N} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ r_y^{bp} & r_x^{bp} & -1 \end{bmatrix} [K] \left\{ \vec{\Delta}^{\,p} \right\} \quad ---------- (3.25b)$$

Equations (3.25a) and (3.25b), in conjunction with Equation (3.9), relate the forces applied to blocks A and B to the displacements of their centroids. Recall that Equation (3.9) expresses $\vec{\Delta}^{\,p}$ in terms of the displacements of the centroids connected through point p. Equations (3.25a) and (3.25b) are re-written as follows:

$$f^a = C^i_{aa} u^a + C^i_{ab} u^b \quad ---------- (3.26a)$$

$$f^b = C^i_{ba} u^a + C^i_{bb} u^b \quad ---------- (3.26b)$$

or in matrix form,

$$\begin{bmatrix} f^a \\ f^b \end{bmatrix} = \begin{bmatrix} C^i_{aa} & C^i_{ab} \\ C^i_{ba} & C^i_{bb} \end{bmatrix} \begin{bmatrix} u^a \\ u^b \end{bmatrix} \quad ---------- (3.26c)$$

where each C^i_{jk} sub-matrix is a 3x3 matrix relating forces applied to centroid j to displacements of centroid k, where centroids j and k are the centroids adjacent to the point i. The sub-matrices are given by:

$$\left[C^i_{aa}\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -r_y^{ai} & r_x^{ai} & 1 \end{bmatrix} \left[K^i_G\right] \begin{bmatrix} 1 & 0 & -r_y^{ai} \\ 0 & 1 & r_x^{ai} \\ 0 & 0 & 1 \end{bmatrix} \quad ---------- (3.27)$$

$$\left[C^i_{ab}\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ r_y^{ai} & -r_x^{ai} & -1 \end{bmatrix} \left[K^i_G\right] \begin{bmatrix} 1 & 0 & -r_y^{bi} \\ 0 & 1 & r_x^{bi} \\ 0 & 0 & -1 \end{bmatrix} \quad ---------- (3.28)$$

$$\left[C^i_{ba}\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ r_y^{bi} & -r_x^{bi} & -1 \end{bmatrix} \left[K^i_G\right] \begin{bmatrix} 1 & 0 & -r_y^{ai} \\ 0 & 1 & r_x^{ai} \\ 0 & 0 & 1 \end{bmatrix} \quad ---------- (3.29)$$

$$\left[C^i_{bb}\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -r_y^{bi} & r_x^{bi} & 1 \end{bmatrix} \left[K^i_G\right] \begin{bmatrix} 1 & 0 & -r_y^{bi} \\ 0 & 1 & r_x^{bi} \\ 0 & 0 & 1 \end{bmatrix} \quad ---------- (3.30)$$

E. Assembly of stiffness matrix

The assembly of a stiffness matrix for the group of blocks shown in Figure 3.7, will now be treated. Each block in the assembly is denoted by a number. The total number of blocks in the assembly is n. Similarly, each point P in the block boundary is numbered. Consider block i that is connected on all four sides to blocks j, k, l, and m, by boundaries having points pj, pk, pl, and pm, respectively. The "incidence pair" of point P shall be defined as (a, b). The first number in the incidence pair corresponds to block A in Equation (3.26), and the second to block B. By convention, the first block in an incidence pair will always be the left-most or upper-most block. For example, the incidence of point pj is (i, j), pk is (k, i), pl is (l, i), and pm is (i,m). This convention is not necessary but is helpful for the programming aspects discussed in Chapter 4. Applying equilibrium to block i by using Equation (3.25) with N= 4, and letting the dummy summation index p take the values pj, pk, pl, and pm, results

$$\vec{f}^{\,i} = C_{aa}^{pj}\vec{u}^{\,i} + C_{ab}^{pj}\vec{u}^{\,j} + C_{bb}^{pk}\vec{u}^{\,i} + C_{ba}^{pk}\vec{u}^{\,k} + C_{bb}^{pl}\vec{u}^{\,i} + C_{ba}^{pl}\vec{u}^{\,l} + C_{aa}^{pm}\vec{u}^{\,i} + C_{aa}^{pm}\vec{u}^{\,m} \quad ----(3.31)$$

The matrices C are given by Equations (3.27) to (3.30). The superscript indicates the point it belongs to. The subscripts a and b refer to the first and second block from the incidence pair. As was indicated before, matrix C^{p}_{aa} relates the influence of displacements of block A from the C^{p}_{ab} relates the displacements of block B to the forces of block A on that incidence pair.

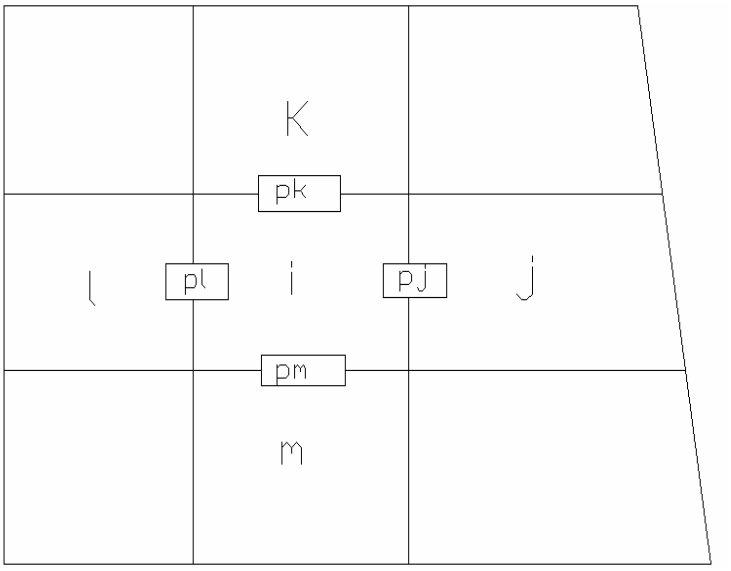


Figure 3.8. Typical block assembly

Vector $\vec{f}$ is a column vector composed of all $\vec{f}_i$ global force vectors,

$$\left\{\vec{f}\right\} = \begin{Bmatrix} \vec{f}^{\,1} \\ \vec{f}^{\,2} \\ M \\ M \\ \vec{f}^{\,n} \end{Bmatrix} \quad (3.32)$$

Similarly, vector $\vec{u}$ is composed of the global displacements vectors of each centroid.

Note that the size of $\vec{f}$ and $\vec{u}$ is n*3, which is equal to the number of degrees of freedom (DOF) in the system of blocks. Recall that each vector $\vec{f}^{\,i}$ represents the forces applied to the centroid of each block. In an usual slope stability problem, it corresponds to the weight of the blocks, or to any externally applied surcharge in the boundary blocks.

F. Rule of assembly

If K^* is a matrix composed of n x n sub-matrices then K^* relates f to u such that,

$$\left\{ \vec{f} \right\} = \left[K^* \right] \left\{ \vec{u} \right\} \quad (3.33)$$

Note that the i, j element of K^* relates the displacement vector $\vec{u}_j$ to the force vector $\vec{f}^{\,i}$. Inserting the coefficients of Equation (3.31) into the matrix K^*, a rule of assembly can be inferred: For a boundary point p, with adjacent blocks i and j, and with an incidence pair (i, j), the sub-matrices C^p_{aa}, C^p_{ab}, C^p_{ba}, and C^p_{bb} defined by Equations (3. 27), (3.28), (3.29), and (3.30), respectively, are inserted to the K^* matrix as follows:

C^p_{aa} goes to the element i, i;

C^p_{ab} goes to the element i, j;

C^p_{ba} goes to the element j, i;

C^p_{bb} goes to the element j, j.

When a sub-matrix is inserted into K^*, it is added to the previous value of the element to which it belongs. This reflects the fact that elements with a common variable in Equation (3.31) can be combined.

G. Characteristics of the stiffness matrix

Since the stiffness matrix, K^*, is composed of n x n sub-matrices, and each sub-matrix is 3 x 3, K^* is an (n*3) x (n*3) matrix, where n*3 is the number of DOFs of the system (3 degrees of freedom per block). Note that the size of the matrix corresponds to the size of vectors $\vec{f}$ and $\vec{u}$ in Equation (3.33). By virtue of Betti's Law , K^* is symmetric. This is proven by first considering a case in which a unit displacement is applied to the degree of freedom i, while all other displacements are zero, and then applying a unit displacement to degree of freedom j, keeping all others zero. This results in,

$f_i=K_{ij}u_j=K_{ij}$----------(3.34)

$f_j=K_{ji}u_i=K_{ji}$----------(3.35)

Betti's law states that:

$f_iu_j=f_ju_i$ ----------(3.36)

Substituting in Equations (3.36) from Equations (3.34) and (3.35), with $u_i=uj= 1$, results in $K_{ij}=K_{ji}$, thus the matrix is symmetric.

The stiffness matrix is also banded with a bandwidth (B) equal to:

$B=(|max(i-j)|+1)*3$ ----------(3.37)

where i, j are block numbers in an incidence pair, and the number 3 is the number of degrees of freedom per block. This can be easily shown by invoking the rule of assembly, and considering a stiffness matrix equal to 0. The bandwidth resulting from inserting the sub-matrices corresponding to a point P with incidence pair (i, j) will be $[|i-j|+1]*3$. Thus the maximum bandwidth correspond to the P point that has the highest difference in the incidence pair (i, j).

H. Application of boundary conditions

If the matrix K^* is unconstrained, it is singular. This is intuitively obvious because without the application of boundary conditions, any force applied to the system of blocks in Figure (3.7) will result in rigid body motion of the whole system. The matrix K^* is constrained by the application of boundary conditions. The application of homogeneous constraints, i.e., setting the displacement components of selected blocks to zero, is considered. In most slope stability applications, the constraints will be applied to the motion of boundary blocks. The constraints will be applied by observing that for $u_i = 0$, that displacement will have no influence on any of the forces. Therefore, the i^{th} column of the stiffness matrix can be eliminated. Also, for $u_i = 0$, any force applied to centroid i will only cause a

reaction force on that centroid, and will have no effect on the displacements of any other centroids. Therefore, the row corresponding to f_i in the stiffness matrix can be eliminated. Thus, the constraint is applied by eliminating the row and column in the stiffness matrix of the DOF for which the constraint is applied. Similarly, since u_i is fixed to zero, the constraint can be applied by setting all the i^{th} row and column of the matrix to zero, and the diagonal element to 1.

3.2.2 Treatment of instability.

Numerical instability occurs when the constrained matrix K^* is singular. Physically, this indicates that there exists a system of blocks that will undergo unrestrained motion upon the application of a force. The objective of the present model is to observe the failure mechanism, and there is no interest in observing the flow mechanism that may follow a slope stability failure. Thus, instabilities can be removed by ignoring the blocks that are causing them.

Mathematically, this can be achieved in two different ways: a) an imaginary spring can be attached to the degree of freedom causing instability, or b) the degree of freedom causing instability can be removed. The present method makes use of the latter option, following the observations in the previous paragraph. Removal of a DOF is obtained simply by removing the row and column corresponding to the DOF in the stiffness matrix. Since the displacement of a DOF corresponding to an unstable block is not zero, the boundaries connected to the DOF are also removed. The method of removal will be discussed in Chapter 4.

The solution of the method presented consists simply in the solution of linear

simultaneous equations. This makes the model easy to code. The computer implementation of the method will be discussed in the next chapter.

CHAPTER 4

Computer Implementation and Testing

4.1 Computer Implementation

4.1.1 Introduction

This chapter presents the computer implementation of the model presented in the previous chapter. The program is written using Visual Basic 6.0. It is named Slope Stability Analysis using Discrete Element Method (SSDEM).

SSDEM performs stability analysis using the mesh input by the user. The slice geometry, therefore, can be input readily by the user. Description of SSDEM is given for an arbitrary

mesh. Factor of safety, similar to that of LEMs, is calculated and is used as a measure of the stability of the slope.

The input to SSDEM are slope and mesh geometry, force vector and soil parameters. These can be input using text files or filling out descriptive interface fields. The program builds local stiffness matrices and assembles them into a global stiffness matrix. The stiffness matrix is used to find centroid displacements. From these displacements stresses along boundaries are calculated and compared with Mohr-Coulomb failure criterion (Eq. 3.1). If stresses exceed this criterion (yielding in soil), then stiffness values are re-evaluated and the process is repeated until the number of iteration is reached.

The program then calculates the factor of safety and centroidal displacement. The stiffness matrix, yielding block IDs and unstable block IDs are readily available from the user interface on SSDEM. All these output are stored as text files which can be opened by any type of windows based text editors.

Section 4.1.2 describes input to SSDEM. The section that follows presents the solution algorithm. Section 4.1.4 discusses the iteration procedure. In section 4.1.5 post processing of output is given coverage. Section 4.1.6 presents the overall program organization with a flow chart of the code. Finally, section 4.2 gives examples of using SSDEM and its verification.

4.1.2 Input to SSDEM

Block and boundary data can be input using the graphic user interface (See Fig. 4.1). These inputs will be saved as data file when the program is run by clicking the 'Analyze' button. A

second alternative for input is using a text file. The format of input files is given in Appendix A.

A four-sided block mesh is input by the user by means of the coordinates of the corners. Triangular elements may be entered by combining the coordinates of two of the points. Point P is identified by the program and connectivity is checked. The convention (left-right & top-bottom) mentioned in Chapter 3 is incorporated in the program.

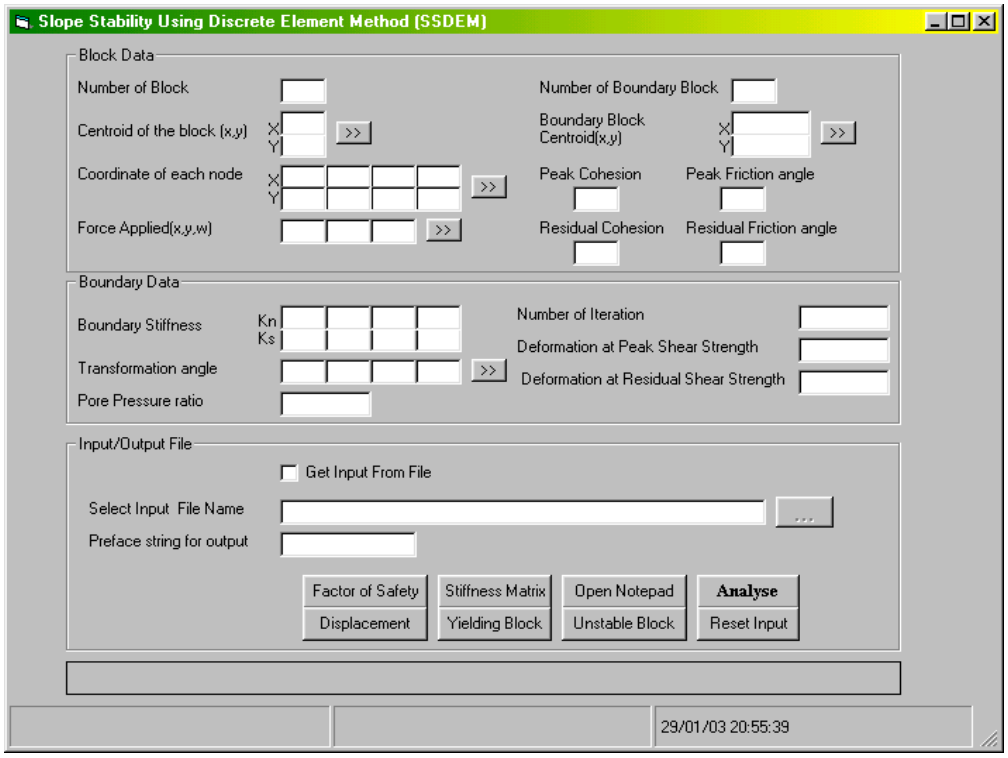


Figure 4.1 User interface of SSDEM.

The program SSDEM is given in appendix B with relevant comments incorporated.

4.1.3 Solution Algorithm

Displacements of the centroid of each block are obtained from the solution equation 3.3.

For most slope stability analysis the force vector f is gravity loading. Other types of loading include surcharge forces, seepage forces and pseudo-static seismic forces. These can be input by giving appropriate values of the force vector.

The solution algorithm used is Gauss-Jordan. The banded nature of the stiffness matrix simplifies the problem.

Numerical problem arise if the stiffness matrix is singular. For a slope that is properly constrained, numerical instability will occur in the case of local or global failure. To avoid this problem SSDEM eliminates the blocks at which instability is encountered. These blocks are given in text format in a file called 'unstable blocks'. Subsequent calculations are carried out by removing the corresponding degree of freedom. This can help for better understanding of block failure propagation.

4.1.4 Iteration Procedure

The solution of Equation (3.33) yields the displacements of the blocks' centroids for each iteration. These displacements are used to find forces (Equation 3.16) and stresses (Equation 3.12a, b) along boundaries. The average shear and normal stresses on a surface are compared with the stress-displacement relations dictated by Equations (3.1) and (3.2), and described in Figure 3.2. If normal stresses exceed the tensile capacity of the soil, a tension crack is created. Tension cracks are modeled by turning the stiffness of the boundaries where they occur to zero. It is noted that in case of large block displacements, some of the tension cracks may close. This type of geometric rearrangement can be a later improvement on the program.

When peak shear strength is achieved, the shear stiffness changes to the secant stiffness corresponding to the boundary displacement δ_s (Equation 3.11b, Figure 4.2).The displacement at peak strength, δ_{peak}, is obtained from the peak strength τ_p (Equation 3.1) and the initial stiffness, k_s:

$$\delta_{peak} = \frac{\tau_p}{k_s} \qquad (4.2)$$

The displacement at the onset of residual strength (δ_{res}) is entered by the user as

input to the program. If the displacement, δ_s, exceeds δ_{res}, then the shear strength, τ, is governed by Equation (3.2). If the displacement is in between δ_{peak} and δ_{res}, τ is interpolated from Equations (3.1) and (3.2). Once τ is determined, the new stiffness value is evaluated from:

$$(k_s)_{new} = \frac{abs(\tau)}{\delta_s} \quad \text{(4.3)}$$

When the shear stiffness increases, it is an indication of unloading in the soil, and the stiffness is not changed. Thus, a check is provided to ensure that the value of the secant stiffness does not increase from one iteration to the other.

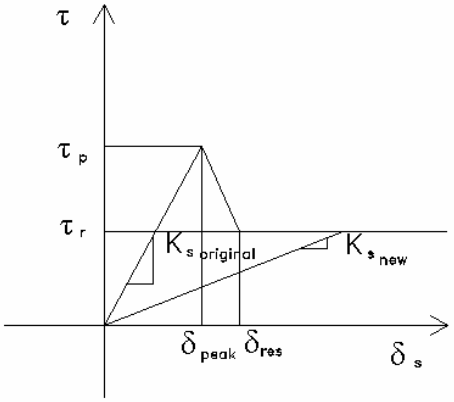


Figure 4.2. Secant stiffness evaluation

The model in Figure 4.2 is intended to represent soil behavior. When initial load increments produce large displacements, the peak in the stress-displacement curve (Figure 4.2) is "missed", and the resistance drops immediately to residual strength. To avoid this, the load must be applied in very small increments around this peak region. The iteration process is stopped when both the following conditions are satisfied:

i) changes in boundary stiffness is less than a user defined value, and,

ii) no local instabilities are generated in the stiffness matrix. For ease of programming, the process is stopped when the number of iterations (input by user) is reached.

4.1.5 Post processing of Output

The standard output of SSDEM are the centroid displacements and factor of safety at each boundary when a soil mass is unstable. The failure surface is easily determined from the displacement discontinuity formed between safe and critical blocks. Displacement values of each block allow the user to follow the failure mechanism. The average of the factor of safeties at each relevant boundary can be used to compare DEM with LEM and FEM methods.

Detailed descriptions of results are presented in section 4.2.

4.1.6 Program organization

The program SSDEM is comprised of a main program and more than thirty subroutines. A flow chart of SSDEM is given in fig 4.3. The main program manages these subroutines while each subroutine serves to carry out specific tasks. This feature is meant to make future improvement efficient.

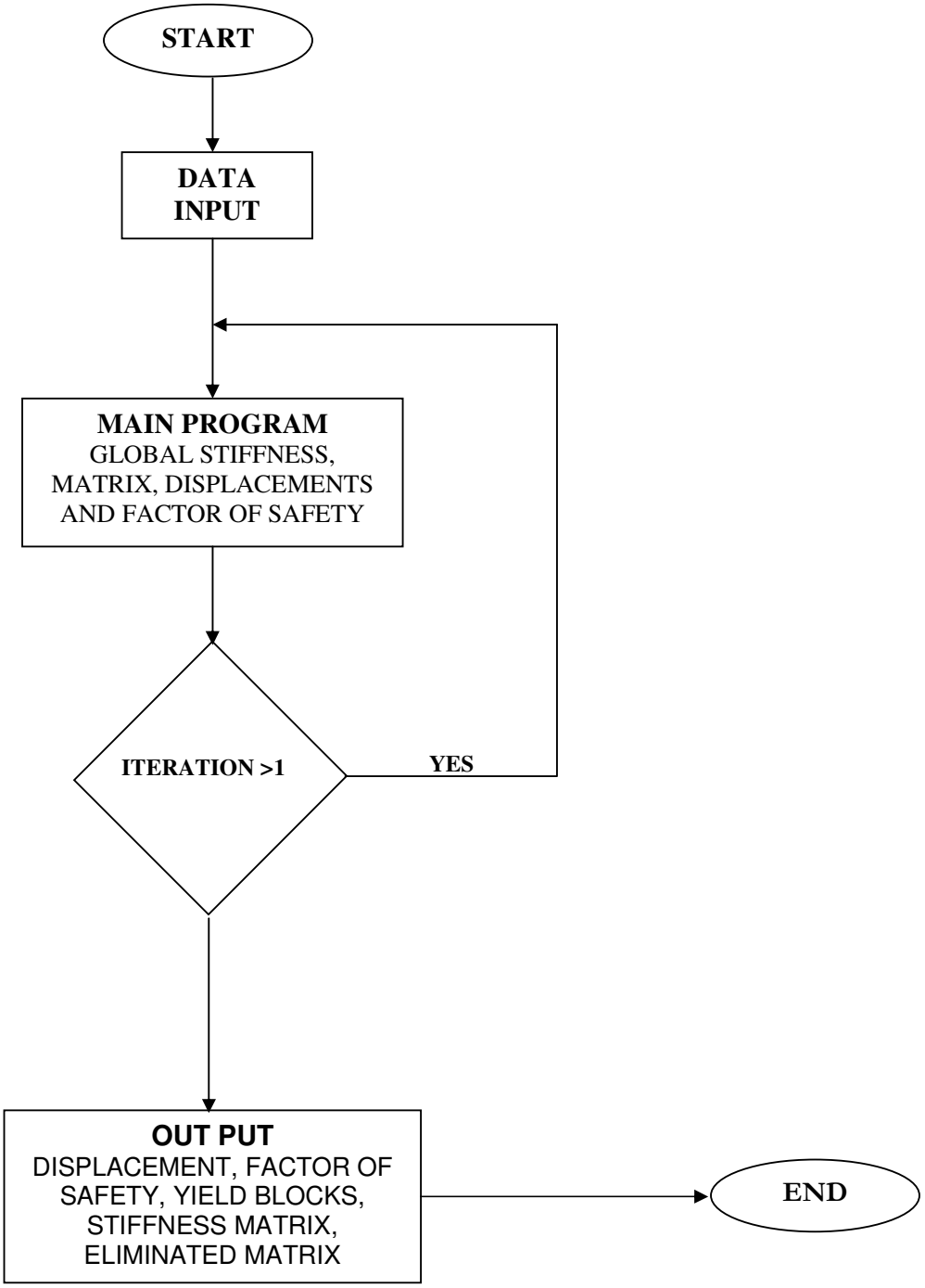


Figure 4.3 Flow chart of SSDEM

4.2 Testing

4.2.1 Introduction

The input parameters to SSDEM can be divided into three groups:

(1) Modeling parameters include mesh type, mesh density, size of the region of analysis, loading rate (controlled by number of iteration) and boundary conditions.

(2) Soil strength and geometric parameters include peak and residual Mohr-Coulomb parameters peak and residual displacements needed for onset of the respective strengths and geometric parameters of the slope.

(3) The normal and shear stiffness parameter (k_n &k_s) for evaluation of results, 'yield' is quantified as (Rodriguez -Marek 1996)

$$\text{Yield} = \left| \frac{k_{new} - k_{old}}{k_{old}} \right| \times 100\,\%$$

Yield levels of 100% indicate block boundaries with zero stiffness, hence, can be used to locate the failure surface. Displacement values are useful to locate the critical surface.

To demonstrate the potential of SSDEM, three examples which were previously studied by other researchers are presented in this chapter. Section 4.2.2 demonstrates the use of SSDEM for factor of safety calculations on an artificial slope. Section 4.2.3 studies a natural slope stability problem with particular emphasis on location of critical slip surface. Section 4.2.4 presents an example dam with seepage.

4.2.2 Example One

Chang (1992) used DEM to analyze an artificial slope (2 on 1), 12.2m high,$\phi_p=\phi_r=20^o$, $c_p=c_r=28.73KN/m^2$ and $\gamma=19.2KM/m^3$.See figure 4.4. Safety factors are computed for two conditions:

(a) the drained condition and (b) with pore pressure taken into consideration (the ratio of pore pressure to over burden stress r_u = 0.25). Since the objective of this example is to compare factor of safeties, the critical surface determined by Chang is used. Factor of safeties are compared with SSDEM results.

Table 4.1 Factor of safety comparison

Case No.	Pore Pressure Ratio r_u	Simplified Bishop's method	Specer's method			Janbu's Simplified method	MongensternPrice method F(x) = constant		Chang (1992)	Pre-sent study
			Fs	θ	λ		Fs (range)	X (range)		
1	r_u = 0	2.08	2.07	14.81	0.237	2.04	2.076 – 2.085	0.254 – 0.318	1.92	1.93
2	r_u = 0.25	1.77	1.76	14.33	0.255	1.73	1.765 – 19.77	0.244 – 0.432	1.68	1.66

The present method compares well with Chang's DEM calculation. It should be noted that FS values predicted by DEM tend to be lower than most limit equilibrium methods.

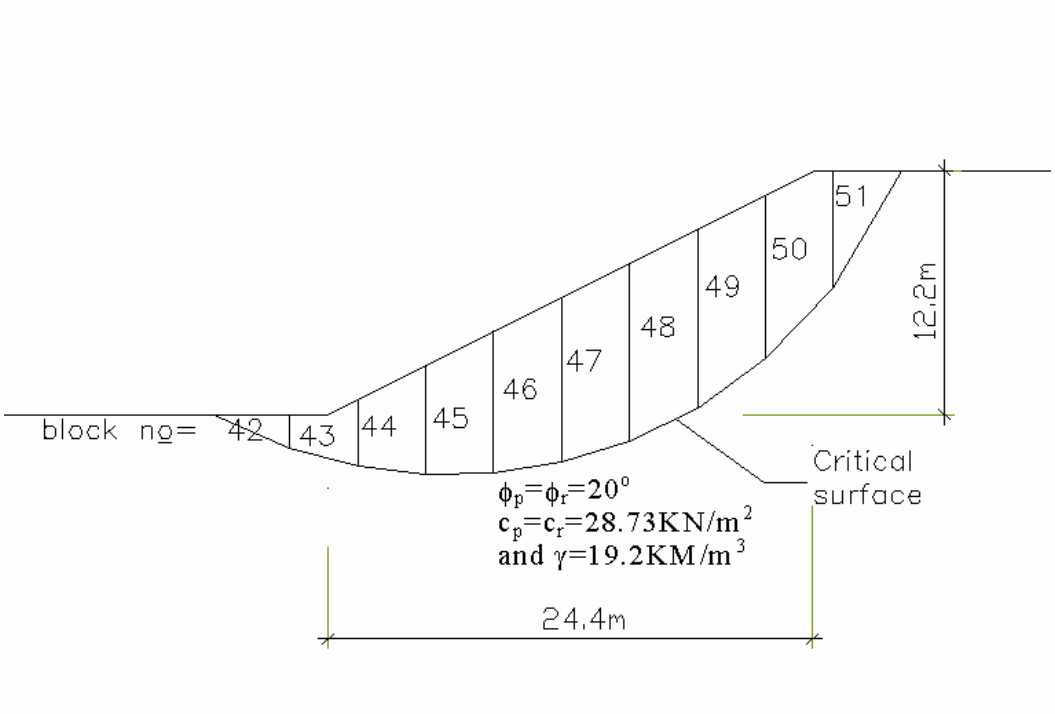


Figure 4.4 Geometry of artificial slope

4.2.3 Example Two

A land slide in the south slope of the River Luna Valley upstream of Selset at the head of Grassholme Reservoir in England is presented. It has been previously studied by Skempton and Brown (1961), Law and Lumb 1978, Chang 1992, Rodriguez–Marek (1996) and Rodriguez et al (1996).

The slope is 12.8m high with an inclination of 28^0. Peak soil strength parameters are $C`_p$ = 8.6Kpa, $\phi`_p = 32^0$. Residual strength parameters are taken as $\phi`_p = \phi`_r$ and $c`_r = 0$, unit weight of the soil is 21.8 KN/m^3. A pore pressure ratio of 0.45 was used in the analyses. See fig 4.5 the critical surface was obtained using random surface generation (Rodriguez-Marek(1996)).

The critical surfaces are shown on Figure 4.5 .The blocks that resulted in displacement incompatibility from the present study are shown shaded. Joining the mid point of sides of these blocks gave the critical surface.

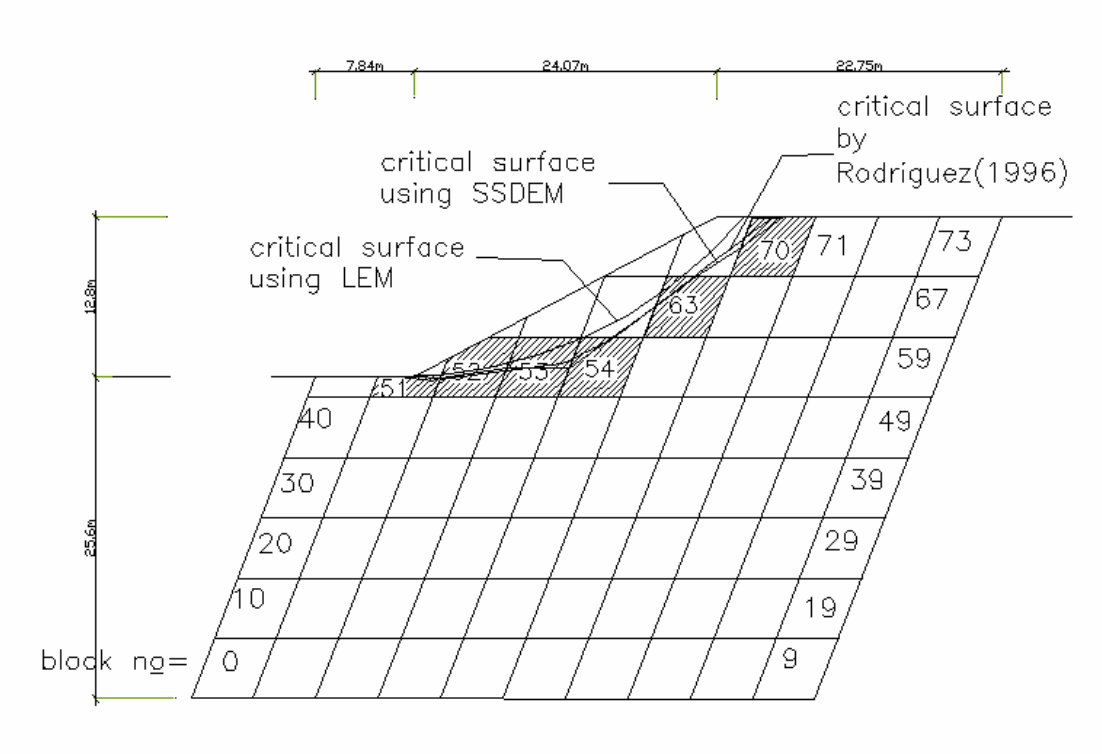


Figure 4.5 Critical failure surfaces, SSDEM mesh and critical blocks (Selset)

From Figure 4.5, it is apparent that:(a)the slip circle from LEM is consistently shallower than that found using DEM,(b) the critical surface using SSDEM closely agrees with that of Rodriguez-Marek's(1996) DEM.

4.2.4 Example Three

J. Kim, R. Salgado and H. S. Yu (1999) studied a dam with conditions shown in Figure 4.6. The slope is 10m high with an inclination of 45^0. Peak soil strength parameters are C^1_p = 20Kpa, ϕ^1_p = 15^0. Residual strength parameters are taken as $\phi`_p = \phi`_r$ and

$C^{1}_{r} = C^{1}_{p}$,. Unit weight of the soil is 18 KN/m^3. The dam is underlain with an impervious firm base extending infinitely. For use in the present study and upper bound finite element analysis by the authors, pore water effects (seepage and buoyancy) are incorporated as external work done by pore water pressures. In their finite element lower-bound analysis, pore water pressures were assigned to each submerged node and unit weight of soil is maximized using linear programming subject to equilibrium and linearized Mohr-Coulomb yield condition. In upper-bound finite element analysis traction forces and unit weight were minimized with the associated flow rule within the element, flow rule along discontinuities and velocity boundary conditions used as constraints.

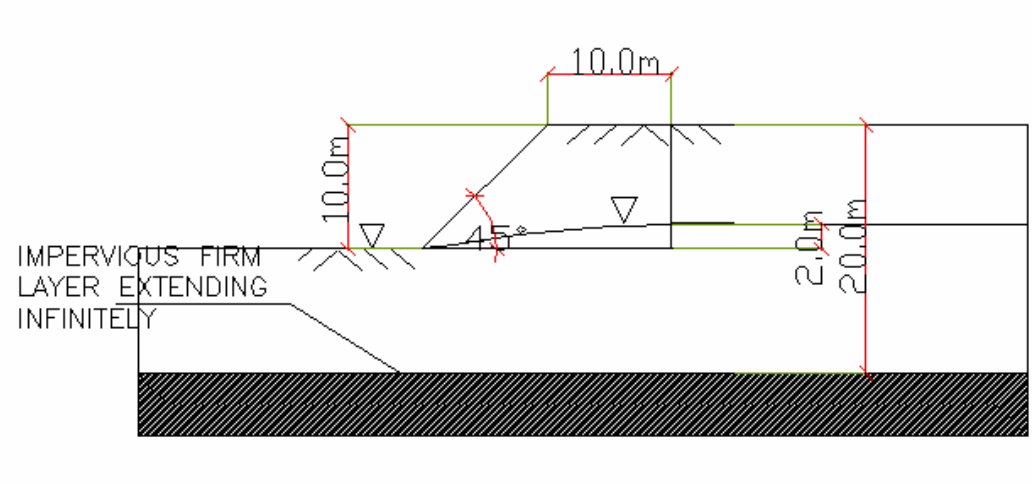
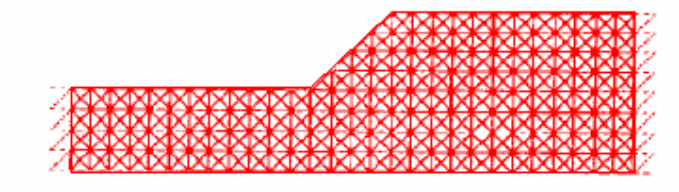


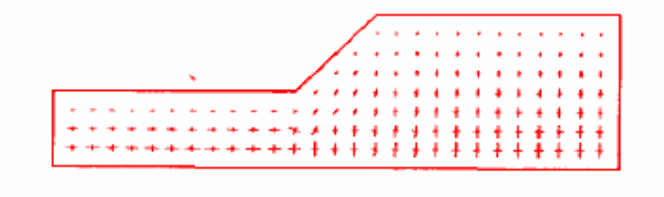
Figure 4.6 Geometry of slope used in the analysis of dam

Their analysis used a linearized yield criterion and stress-strain relation that is perfectly plastic with associated flow rule.The typical finite-element mesh used in Kim, Salgado and Yu (1999) is shown in Fig 4.7(a). For a given slope geometry, the researchers used identical finite-element meshes for lower- and upper-bound analyses. The only exception was that the extension elements (dotted lines in fig 4.7(a)) were used for lower-bound analysis to extend the solution over a semi-infinite domain. In the study, three nodded linear triangular elements were analyzed

under the condition of plane strain. In upper-bound analysis, extension elements are not needed because rigid boundaries are assumed. Figs 4.7(b)-(d) show examples of a stress field from lower-bound analysis and plastic zone and velocity field from upper bound analysis.

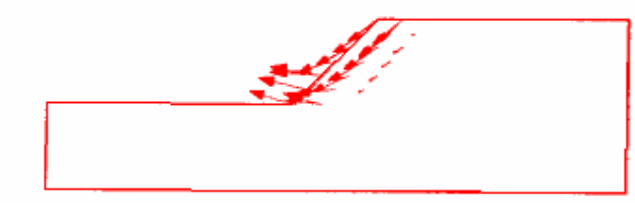


(a)Finite element mesh



(b)Principal stresses from lower bound analysis

(c)Plastic zone from upper bound analysis



(d)Velocity vectors from upper bound analysis

Fig 4.7 Finite element mesh, principal stress field, plastic zone and velocity field

The authors have also done limit-equilibrium analysis using the slope stability program PCSTABL5M with Bishop's simplified method, which is commonly used in practice. For the comparison of limit analysis (FEM) solutions with limit-equilibrium solutions, they used two

dimensionless numbers, namely, stability number ($N_F=\gamma HF/c^h$) and extension of Janbu's dimensionless number ($\lambda c\emptyset w=\alpha(\gamma H-\gamma_w H_w)\tan\emptyset^l/C^h$) which is used to account for the effects of pore water pressure. Kim, Salgado and Yu (1999) developed relationships between these numbers, which they in turn used to arrive at the factor of safety.

Figure 4.8 shows the SSDEM mesh used in the analysis, critical failure surface using SSDEM and blocks resulting in displacement incompatibility in DEM analysis (shown shaded).

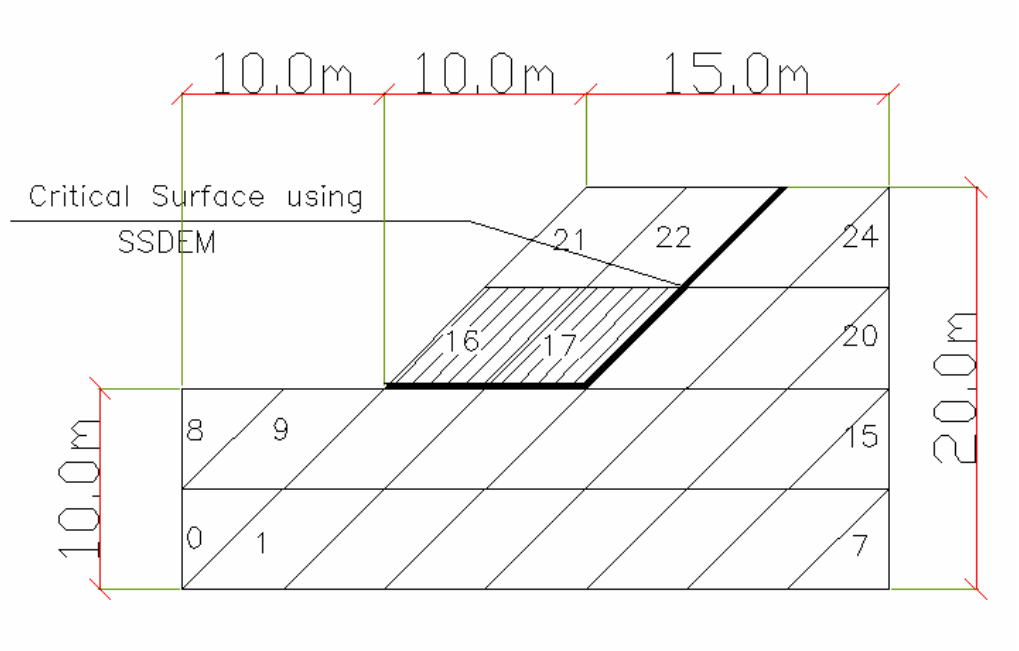


Figure 4.8 Critical failure surface and critical blocks(Dam)

Table 4.2 shows factor of safety values calculated using DEM against those calculated by Kim,Salgado and Yu(1999).

Table 4.2 Factor of safety results

Kim,Salgado,Yu (1999)						Present Study (FS using SSDEM)
Lower		Bishop		Upper		
$\lambda_{c\emptyset w}$	FS	$\lambda_{c\emptyset w}$	FS	$\lambda_{c\emptyset w}$	FS	

	2.211	1.10	2.254	1.17	2.211	1.23	1.32

The higher factor of safety from SSDEM can be partly due to sources of error in calculation of seepage forces.

4.3 Summary

The three examples presented above show the tremendous potential of SSDEM. Factor of safety and critical surface search can be determined more realistically using SSDEM. The fact that the results from SSDEM are in good agreement with other researchers' suggests the accuracy of SSDEM.

CHAPTER FIVE

Conclusions and Recommendations

5.1 Conclusions

A static DEM has been presented to solve slope stability analyses. The method consists of discretizing the soil mass into blocks which are joined by Winkler springs. From relation of forces with stiffness and displacement, a set of simultaneous equations is solved to find displacements and stresses. The local stresses are used to calculate factor of safety while the displacement fields are used for predicting the critical failure surface.

In conclusion, the method used has shown a strong potential for use in stability analyses of slopes. This study shows that:

1) The artificial slope example showed good agreement with prior research results with regards to factor of safety values indicating that DEM is an accurate method for calculation of safety factor with predetermined critical slip surface.

2) The Selset slope example model shows the strong potential of DEM for identification of critical slip surface.

3) The analysis of the dam with seepage shows that the DEM can be used for other types of external forces. In the particular example, seepage forces were treated as external forces on the blocks. The discrepancy in the value of factor of safety can be partly attributed to the likely source of error in calculation of seepage forces using flow net.

4) Unlike limit equilibrium methods for slope stability analyses, DEM is an 'advanced' method that satisfies all conditions of equilibrium.

5) DEM has an advantage over FEM because it is simpler to use. In addition, its simplicity is complimented by the fact that it is computationally cheaper in comparison with FEM. The method presented requires only the solution of simultaneous equations and thus is easy to implement at a very low cost.

5.2 Recommendations

DEM is a good way of modeling soil behavior. SSDEM is a step in the direction of effectively using DEM. Here are some recommendations and suggestions.

1) Parametric study: SSDEM can be used to study the influence of modeling parameters, which are listed in section 4.2.1.

2) Inclusion of graphic user interface: the input to SSDEM can be greatly simplified if a user graphic interface is developed. The output will also be enhanced with such an improvement.

3) Modeling non-homogeneous slopes: This will only require minimum modification to the code. Including an input procedure to include number of layers and different fields to be filled for each layer type is recommendable.

4) Pore water pressure generation: More accurate modeling of slope stability problem is possible by using piezometric data (specification of pore pressure at discrete points within a slope) along with a flow net appropriate for the model.

5) Allowing for change in mesh geometry due to deformations: Such a step can be incorporated at each iteration. This would permit closing of tension cracks and would result in a more realistic model.

6) Introducing K_o effects: The model does not generate lateral stresses from vertically applied loads. The model can be improved by introducing a lateral pressure coefficient.

7) Validation using finite element methods: By using local stresses calculated by the program, principal stresses can be estimated. These stresses and the displacements can be used to compare DEM and FEM

8) Slope stabilization: The study of slope stabilization mechanisms can be studied without much further modification. E.g., soil nailing, soil anchors or micropiles.

9) Bearing capacity analysis: This can be another area of numerical modeling application of the presented method.

Bibliography

Abramson, L.W. et al (1996) "Slope stability and stabilization methods" John Wiley and Sons Inc, Canada

Anandarajah, A. (1994) " discrete element method for simulating behavior of cohesive soil" Journal of Geotechnical Engineering, ASCE, Vol. 120, No. 9, 1593-1613

Chang,C.S. (1992) "Discrete element method for slope stability analyses" Journal of Geotechnical Engineering, ASCE, Vol.118, No. 12, 1889-1905

Chugh A.K. (2002) "A method for locating critical slip surfaces in slope stability analysis: discussion." Canadian Geotechnical Journal, 39,765-770

Cundall, P.A. and Strack, O.D.L (1979). "A discrete numerical model for granular assemblies" Geotechnique, 29, No. 1, 47-65

Duncan, J.M. (1996) "State of the art: limit equilibrium and finite-element analyses of slopes" Journal of Geotechnical engineering, ASCE, vol. 122, 577-596

Greco,V.R. (1996). " Efficient Monte Carlo technique for locating critical surface." Journal of Geotechnical Engineering, ASCE, Vol. 122, No. 7, 514-525

Janbu, N. (1973) " Slope stability computations", Embankment Dam Engineering, John Wiley and Sons

Kim, J. et al(1999) "Limit analysis of soil slopes subjected to pore water pressures" Journal of Geotechnical and Geoenviromental Engineering, ASCE,49-58

Leshchinsky, D. and Huang C.C.(1992) " Generalized three-dimensional slope-stability analysis." Journal of Geotechnical Engineering, ASCE, Vol. 118, No. 11, 1748-1763

Morgenstern, N.R. and Price, V.E. (1965) " The analyses of stability of general slip surfaces." Geotechnique,London,England,15,No1, 79-93

Nguyen, V.U. (1985) "Determination of critical slope failure surfaces" Journal of Geotechnical Engineering, ASCE, Vol. 111,No. 2, 238-248

Ortigao,J.A.R. et al.(1997) " Slope failures in tertiary expansive OC clays" Journal of Geothechnical and Geoenviromental Engineering, ASCE, vol. 123, no. 9, 812-817

Rodriguez-Marek,(1996) "Discrete element method for slope stability analyses" Master's Thesis, Washington State University, Pullman, WA.

Rodriguez-Marek, A. et al (1996) "Discrete element method for slope stability analyses" 7th International Symposium on Land slides, Trondheim, Norway, 1345-1350

Sarma, S.K. (1973) "Stability analyses of embankments and slopes" Geotechnique, 23, No. 3, 423-433

Sarma, S.K.. (1979) " Stability analysis of embankments and slope." Journal of Geotechnical Engineering Div., ASCE, Vol. 105, No. GT12, 1511-1524

Spencer,E.(1967) " A method of analysis of the stability of embankments assuming parallel inter-slice forces" Geotechnique,17,11-26

Spencer, E. (1973) "Thrust line criterion in embankment stability analyses", Geotechnique, No. 1, 85-100

Stark, T.D and Eid, H.T. (1997) "Slope stability analyses in stiff fissured clays." Journal of Geotechnical and Geoenvironmental Engineering, ASCE, Vol. 123, No. 4, 335-343

Zhu, D.Y. (2001) " A method for locating critical slip surfaces in slope stability analyses." Canadian Geotechnical Journal, 38,328-337

Appendix A: Format of Input Files

Input files can be entered to SSDEM using text format. Data in a line can be separated with 'Tab' or space. The sequence of reading data by SSDEM is as follows.

1.Total number of blocks

2.Centroid of each block

3.Nodes or vertices of blocks

4.External forces on each block

5.Number of boundary blocks

6.Centroid of boundary blocks

7.Peak cohesion, peak friction angle, residual cohesion and residual friction angle

8.Winkler's spring constants (Kn & Ks, respectively) for each side, starting from right side and proceeding counter clockwise.

9.Transformation angle in degrees with same order as in No. 8 above

10.Total number of iterations to full loading

11.Deformation for onset of peak strength

12.Deformation for onset of residual strength

13.Pore pressure ratio

Appendix B

Subroutines

```
Option Explicit
'Declaration of Global variables
Private NoBlock              As Integer
Private YieldBlock(1000, 1)     As Single
Private UnstableBlock(1000)   As Single
Private Displacement()        As Single

Private Centroid()           As Single
Private Node()               As Single
Private Force()              As Single
Private RefCent()            As Single
Private Stiffness()          As Single
Private angle()              As Single
Private Cp, Pp, Cr, Pr       As Single

Private o, p, q, u, t        As Integer          'for loop
Private c, n, f, a           As Integer          'input
Private r, s, g              As Integer          'input
Private NOIteration          As Integer
Private Del_r                As Single
Private Del_p                As Single
Private NoOfRefBlock         As Integer
Private Ru                   As Single

Private midpoint()           As Single
Private length()             As Single
Private AssStifMatrix()      As Single
Private FactorOfSafety()     As Single
Private strpreface           As String
Private strInput             As String
Private objMatrix            As CMatrix
'*****************************************************
'Procedure Name   BlockCal
'Usage            Calculate stiffness matrix,
'                 Displacement vector and factor of safety
'*****************************************************
Private Sub BlockCal()
 'Declaration of procedure level varaibles
 ReDim AssStifMatrix(NoBlock * 3, NoBlock * 3) As Single
  Dim Caa_j(2, 2)          As Single
 Dim Cbb_k(2, 2)           As Single
 Dim Cbb_l(2, 2)           As Single
 Dim Caa_m(2, 2)           As Single
 Dim Cab_j(2, 2)           As Single
 Dim Cba_k(2, 2)           As Single
 Dim Cba_l(2, 2)           As Single
 Dim Cab_m(2, 2)           As Single
 Dim K_j(2, 2)             As Single
 Dim K_k(2, 2)             As Single
 Dim K_l(2, 2)             As Single
 Dim K_m(2, 2)             As Single
```

```
 Dim Da_j(2, 2)          As Single
 Dim Db_k(2, 2)          As Single
 Dim Db_l(2, 2)          As Single
 Dim Da_m(2, 2)          As Single

 Dim Dat_j(2, 2)         As Single
 Dim Dbt_k(2, 2)         As Single
 Dim Dbt_l(2, 2)         As Single
 Dim Dat_m(2, 2)         As Single

 Dim Dbt_j(2, 2)         As Single
 Dim Dat_k(2, 2)         As Single
 Dim Dat_l(2, 2)         As Single
 Dim Dbt_m(2, 2)         As Single

 Dim temp(2, 2)          As Single
 Dim j, K, l, m          As Integer
 Dim count               As Integer
 Dim yield               As Integer
 Dim iterate             As Integer
 count = 0
 yield = 0

'Full iteration start point
For iterate = 1 To NOIteration
   SBar.Panels(1).Text = "Iteration = " & iterate

   For t = 0 To NoBlock - 1
   SBar.Panels(2).Text = "Processing assembly of stiffness matrix"
   pbar.Value = (t / (NoBlock - 1)) * 100

      'j,k,l and m are the adjesent block of block i depending on the value of t

        j = -1: K = -1: l = -1: m = -1
        j = adjbound(midpoint(t, 0), midpoint(t, 1), t)
        K = adjbound(midpoint(t, 2), midpoint(t, 3), t)
        l = adjbound(midpoint(t, 4), midpoint(t, 5), t)
        m = adjbound(midpoint(t, 6), midpoint(t, 7), t)
    If Not IsRefBlock(Centroid(t, 0), Centroid(t, 1)) Then
       For q = 0 To 2
         For p = 0 To 2
          temp(q, p) = 0
          Cab_j(q, p) = 0
          Cba_k(q, p) = 0
          Cba_l(q, p) = 0
          Cab_m(q, p) = 0
        Next
        Next
      'calculate Global Stiffness matrix at each boundary j,k,l and m.
```

```
        Call solveK(Centroid(t, 0), Centroid(t, 1), length(t, 0), angle(t, 0), Stiffness(t, 0), Stiffness(t, 1), K_j())
        Call solveK(Centroid(t, 0), Centroid(t, 1), length(t, 1), angle(t, 1), Stiffness(t, 2), Stiffness(t, 3), K_k())
        Call solveK(Centroid(t, 0), Centroid(t, 1), length(t, 2), angle(t, 2), Stiffness(t, 4), Stiffness(t, 5), K_l())
        Call solveK(Centroid(t, 0), Centroid(t, 1), length(t, 3), angle(t, 3), Stiffness(t, 6), Stiffness(t, 7), K_m())

      'calculate Displacement vector for boundary j,k,l and m
      'Caa_j + Cbb_k + Cbb_l + Caa_m  all are block i

        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 0), midpoint(t, 1), True, Da_j(), Dat_j())
        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 2), midpoint(t, 3), True, Db_k(), Dbt_k())
        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 4), midpoint(t, 5), True, Db_l(), Dbt_l())
        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 6), midpoint(t, 7), True, Da_m(), Dat_m())


    '   Caa_j = Da_j * K_j * Dat_j

        For q = 0 To 2
        For p = 0 To 2
         temp(q, p) = 0
        Next
        Next

        Call objMatrix.mult(Da_j(), K_j(), temp())
        Call objMatrix.mult(temp(), Dat_j(), Caa_j())

    '   Cbb_k = Db_k * K_k * Dbt_k

        For q = 0 To 2
        For p = 0 To 2
         temp(q, p) = 0
        Next
        Next


        Call objMatrix.mult(Db_k(), K_k(), temp())
        Call objMatrix.mult(temp(), Dbt_k(), Cbb_k())

    '   Cbb_l = Db_l * K_l * Dbt_l

        For q = 0 To 2
        For p = 0 To 2
         temp(q, p) = 0
        Next
```

```
        Next


        Call objMatrix.mult(Db_l(), K_l(), temp())
        Call objMatrix.mult(temp(), Dbt_l(), Cbb_l())

'    Caa_m = Da_m * K_m * Dat_m

        For q = 0 To 2
        For p = 0 To 2
         temp(q, p) = 0
        Next
        Next


        Call objMatrix.mult(Da_m(), K_m(), temp())
        Call objMatrix.mult(temp(), Dat_m(), Caa_m())


    'calculate common stiffness matrix for boundary j,k,l and m
    'Cab_j + Cba_k + Cba_l + Cab_m  all are block i

        For q = 0 To 2
        For p = 0 To 2
         temp(q, p) = 0
         Da_j(q, p) = 0
         Db_k(q, p) = 0
         Db_l(q, p) = 0
         Da_m(q, p) = 0
        Next
        Next

       Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 0), midpoint(t, 1), False, Da_j(),
temp())
       If j <> -1 Then
       If Not IsRefBlock(Centroid(j, 0), Centroid(j, 1)) Then
        Call DisVector(Centroid(j, 0), Centroid(j, 1), midpoint(t, 0), midpoint(t, 1), False, temp(),
Dbt_j())
       End If
       End If

       'Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 2), midpoint(t, 3), False, temp(),
Dat_k())
       Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 2), midpoint(t, 3), False, Db_k(),
temp())
       If K <> -1 Then
       If Not IsRefBlock(Centroid(K, 0), Centroid(K, 1)) Then
        Call DisVector(Centroid(K, 0), Centroid(K, 1), midpoint(t, 2), midpoint(t, 3), False,
temp(), Dat_k())
        'Call DisVector(Centroid(K, 0), Centroid(K, 1), midpoint(t, 2), midpoint(t, 3), False,
Db_k(), temp())
```

```
        End If
        End If

        'Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 4), midpoint(t, 5), False, temp(),
Dat_l())
        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 4), midpoint(t, 5), False, Db_l(),
temp())
        If l <> -1 Then
        If Not IsRefBlock(Centroid(l, 0), Centroid(l, 1)) Then
        'Call DisVector(Centroid(l, 0), Centroid(l, 1), midpoint(t, 4), midpoint(t, 5), False, Db_l(),
temp())
        Call DisVector(Centroid(l, 0), Centroid(l, 1), midpoint(t, 4), midpoint(t, 5), False, temp(),
Dat_l())
        End If
        End If



        Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, 6), midpoint(t, 7), False, Da_m(),
temp())
        If m <> -1 Then
        If Not IsRefBlock(Centroid(m, 0), Centroid(m, 1)) Then
        Call DisVector(Centroid(m, 0), Centroid(m, 1), midpoint(t, 6), midpoint(t, 7), False,
temp(), Dbt_m())
        End If
        End If

         For q = 0 To 2
         For p = 0 To 2
          temp(q, p) = 0
         Next
         Next

    '   Cab_j = Da_j * K_j * Dbt_j
        If j <> -1 Then
        If Not IsRefBlock(Centroid(j, 0), Centroid(j, 1)) Then

         For q = 0 To 2
         For p = 0 To 2
          temp(q, p) = 0
         Next
         Next


         Call objMatrix.mult(Da_j(), K_j(), temp())
         Call objMatrix.mult(temp(), Dbt_j(), Cab_j())
        End If
        End If
    '   Cba_k = Db_k * K_k * Dat_k
         If K <> -1 Then
         If Not IsRefBlock(Centroid(K, 0), Centroid(K, 1)) Then
```

```
      For q = 0 To 2
      For p = 0 To 2
       temp(q, p) = 0
      Next
      Next


      Call objMatrix.mult(Db_k(), K_k(), temp())
      Call objMatrix.mult(temp(), Dat_k(), Cba_k())
     End If
     End If
'Cba_l = Db_l * K_l * Dat_l
     If l <> -1 Then
     If Not IsRefBlock(Centroid(l, 0), Centroid(l, 1)) Then
      For q = 0 To 2
      For p = 0 To 2
       temp(q, p) = 0
      Next
      Next

      Call objMatrix.mult(Db_l(), K_l(), temp())
      Call objMatrix.mult(temp(), Dat_l(), Cba_l())
     End If
     End If
'    Cab_m = Da_m * K_m * Dbt_m
     If m <> -1 Then
     If Not IsRefBlock(Centroid(m, 0), Centroid(m, 1)) Then
      For q = 0 To 2
      For p = 0 To 2
       temp(q, p) = 0
      Next
      Next

      Call objMatrix.mult(Da_m(), K_m(), temp())
      Call objMatrix.mult(temp(), Dbt_m(), Cab_m())
     End If
     End If
   'Caa_j(u, v) + Cbb_k(u, v) + Cbb_l(u, v) + Caa_m(u, v)

      For q = 0 To 2
      For p = 0 To 2
       temp(q, p) = 0
      Next
      Next

     For q = 0 To 2
     For p = 0 To 2
        If j <> -1 Then temp(q, p) = temp(q, p) + Caa_j(q, p)
        If K <> -1 Then temp(q, p) = temp(q, p) + Cbb_k(q, p)
        If l <> -1 Then temp(q, p) = temp(q, p) + Cbb_l(q, p)
        If m <> -1 Then temp(q, p) = temp(q, p) + Caa_m(q, p)
```

```
            Next
            Next
        Else
          'temp = 0,Cab_j=0,Cba_k=0,Cba_l=0,Cab_m=0
            For q = 0 To 2
            For p = 0 To 2
             temp(q, p) = 0
             Cab_j(q, p) = 0
             Cba_k(q, p) = 0
             Cba_l(q, p) = 0
             Cab_m(q, p) = 0
            Next
            Next
            For q = 0 To 2
            temp(q, q) = 1
            Next

        End If
    'Assemble stiffness matrix
          For q = 0 To 2
             For p = 0 To 2
                          AssStifMatrix(t * 3 + q, t * 3 + p) = Round(temp(q, p), 3)
                If j <> -1 Then AssStifMatrix(t * 3 + q, j * 3 + p) = Round(Cab_j(q, p), 3)
                If K <> -1 Then AssStifMatrix(t * 3 + q, K * 3 + p) = Round(Cba_k(q, p), 3)
                If l <> -1 Then AssStifMatrix(t * 3 + q, l * 3 + p) = Round(Cba_l(q, p), 3)
                If m <> -1 Then AssStifMatrix(t * 3 + q, m * 3 + p) = Round(Cab_m(q, p), 3)
             Next
          Next
      Next t
'output stiffness matrix to file AssStifMat.txt
Open strpreface & "AssStifMat.txt" For Output As #1
Dim str As String
    For q = 0 To NoBlock * 3 - 1
    str = ""
    For p = 0 To NoBlock * 3 - 1

      str = str & AssStifMatrix(q, p) & "  "

    Next
      Print #1, str
    Next

Close #1

     Dim Force_new() As Single
     ReDim Displacement(NoBlock * 3 - 1)
     ReDim Force_new(NoBlock * 3 - 1)

     For q = 0 To NoBlock - 1
     For p = 0 To 2
       Force_new(q * 3 + p) = Force(q, p) * iterate / NOIteration
```

```
    Next
    Next

    Dim singular() As Integer
    ReDim singular(NoBlock * 3)

    For p = 0 To NoBlock * 3
    singular(p) = -1
    Next
    'Eliminate assembled matrix using Gauss-Jordan method

    objMatrix.Guass_Jordan  AssStifMatrix(),  Force_new(),  NoBlock  *  3  -  1,  Displacement(),
singular()

   Dim strtest As String
   'output eliminated matrix to file Gauss_Jordan.txt
   strtest = "Gauss_Jordan.txt"
   Open strtest For Output As #1

         For q = 0 To NoBlock * 3 - 1
          str = ""
          For p = 0 To NoBlock * 3 - 1

            str = str & AssStifMatrix(q, p) & vbTab

          Next
            Print #1, str
          Next

       Close #1


    For q = 0 To UBound(singular())
    If singular(q) <> -1 Then
       UnstableBlock(count) = singular(q) \ 3
       count = count + 1
    End If
    Next
      pbar.Value = 0
      SBar.Panels(2).Text = "Iterating"
    ReDim FactorOfSafety(NoBlock, 3)
    For q = 0 To NoBlock
    For p = 0 To 3
    FactorOfSafety(q, p) = -1
    Next
    Next
    Dim z As Integer
    For t = 0 To NoBlock - 1
      For z = 0 To 3

          pbar.Value = ((t * 4 + z) / (NoBlock * 4 - 1)) * 100
```

```
Dim RelDelG(2) As Single
Dim RelDelL(2) As Single
Dim temp1(2) As Single
Dim temp2(2) As Single
Dim Lamda(2, 2) As Single
Dim Disp_i(2)  As Single
Dim Disp_j(2)  As Single
Dim Del_n As Single
Dim Del_s As Single
Dim Sigma_n As Single
Dim Sigma_s As Single
Dim Taw_s As Single
Dim Taw_n As Single
Dim Taw_p As Single
Dim Taw_r As Single
Dim Ks_new As Single
Dim Ks_old As Single

j = -1
j = adjbound(midpoint(t, z * 2), midpoint(t, z * 2 + 1), t)

If j <> -1 Then
  If z = 0 Or z = 3 Then
    Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, z * 2), midpoint(t, z * 2 + 1),
True, temp(), Dat_j())
    Call DisVector(Centroid(j, 0), Centroid(j, 1), midpoint(t, z * 2), midpoint(t, z * 2 + 1),
True, temp(), Dbt_j())
  Else
    Call DisVector(Centroid(j, 0), Centroid(j, 1), midpoint(t, z * 2), midpoint(t, z * 2 + 1),
True, temp(), Dat_j())
    Call DisVector(Centroid(t, 0), Centroid(t, 1), midpoint(t, z * 2), midpoint(t, z * 2 + 1),
True, temp(), Dbt_j())
  End If
   For p = 0 To 2
    Disp_i(p) = Displacement(t * 3 + p)
    Disp_j(p) = Displacement(j * 3 + p)
   Next

  If z = 0 Or z = 3 Then
   objMatrix.mult1 Dbt_j(), Disp_j(), temp1()
   objMatrix.mult1 Dat_j(), Disp_i(), temp2()
  Else
   objMatrix.mult1 Dbt_j(), Disp_i(), temp1()
   objMatrix.mult1 Dat_j(), Disp_j(), temp2()
  End If

   objMatrix.subt1 temp1(), temp2(), RelDelG()

   Cal_Lamda angle(t, z), Lamda()
   objMatrix.mult1 Lamda(), RelDelG(), RelDelL()
```

```
            Del_n = RelDelL(0) + RelDelL(2) * length(t, z)
            Del_s = RelDelL(1)

            Taw_s = Stiffness(t, z * 2 + 1) * Del_s
            Sigma_n = Stiffness(t, z * 2) * Del_n

          Taw_p = Cp + Sigma_n * ((1 - Ru * iterate / NOIteration)) * Tan((Pp * (22 / 7)) / 180)
            Taw_r = Cr + Sigma_n * ((1 - Ru * iterate / NOIteration)) * Tan((Pr * (22 / 7)) / 180)



'           If Del_n <= 0 Then
'            Stiffness(t, z * 2) = 0
'            'Stiffness(t, z * 2 + 1) = 0
'           End If

           If Stiffness(t, z * 2) <> 0 Then
            If Taw_s <> 0 Then
              'Del_p = Taw_p / Stiffness(t, z * 2 + 1)

              If Del_s < Del_p Then
                Ks_new = Stiffness(t, z * 2 + 1)
                FactorOfSafety(t, z) = Round(Taw_p / Taw_s, 2)
              ElseIf Del_p < Del_s And Del_s < Del_r Then
                Ks_new = Abs(Taw_r - ((Taw_p - Taw_r) * (Del_r - Del_s) / (Del_p - Del_r))) /
Del_s
                FactorOfSafety(t, z) = Round(Taw_r / Taw_s, 2)
              ElseIf Del_s > Del_r Then
                Ks_new = Taw_r / Del_s
                FactorOfSafety(t, z) = Round(Taw_r / Taw_s, 2)
              End If

              Ks_old = Stiffness(t, z * 2 + 1)

              'if (ks_new-Ks_old)/ks_old *100 =100 then plot

              If (Ks_new - Ks_old) / Ks_old = 1 Then
               YieldBlock(yield, 0) = midpoint(t, z)
               YieldBlock(yield, 1) = midpoint(t, z + 1)
               yield = yield + 1
              End If
              If Ks_new < Stiffness(t, z * 2 + 1) Then
                Stiffness(t, z * 2 + 1) = Ks_new
              'Else
              '  MsgBox "Unloding In the soil."
              End If
            End If
          End If
        End If
```

```
      Next z
   Next t
Next iterate
 SBar.Panels(2).Text = "Done"
End Sub

'*****************************************************
'Procedure Name   Disvector
'Usage            Calculate matrix R(D)
'
'*****************************************************

Private Sub DisVector(Xc As Single, Yc As Single, Xm As Single, Ym As Single, similar As
Boolean, ByRef DisVec() As Single, ByRef DisVecT() As Single)

    For q = 0 To 2
    For p = 0 To 2
     DisVec(q, p) = 0
     DisVecT(q, p) = 0
    Next
    Next

    If similar Then
       DisVec(0, 0) = 1
       DisVec(1, 1) = 1
       DisVec(2, 2) = 1
       DisVec(2, 0) = -1 * (Ym - Yc)
       DisVec(2, 1) = Xm - Xc

       DisVecT(0, 0) = 1
       DisVecT(1, 1) = 1
       DisVecT(2, 2) = 1
       DisVecT(0, 2) = -1 * (Ym - Yc)
       DisVecT(1, 2) = Xm - Xc

    Else
       DisVec(0, 0) = -1
       DisVec(1, 1) = -1
       DisVec(2, 2) = -1
       DisVec(2, 0) = (Ym - Yc)
       DisVec(2, 1) = -1 * (Xm - Xc)

       DisVecT(0, 0) = 1
       DisVecT(1, 1) = 1
       DisVecT(2, 2) = -1
       DisVecT(0, 2) = -1 * (Ym - Yc)
       DisVecT(1, 2) = Xm - Xc
    End If
End Sub
```

```
'*****************************************************
'Procedure Name   MidPoint()
'Usage            Calculate Midpoint of each side of the blocks
'
'*****************************************************

Private Sub Cal_midpoint()

   For u = 0 To NoBlock - 1
      For o = 0 To 5
        midpoint(u, o) = (Node(u, o) + Node(u, o + 2)) / 2
      Next
        midpoint(u, 6) = (Node(u, 6) + Node(u, 0)) / 2
        midpoint(u, 7) = (Node(u, 7) + Node(u, 1)) / 2
   Next
End Sub
'*****************************************************
'Procedure Name   adjbound()
'Usage            Check whether the blocks are neighbours
'
'*****************************************************

Private Function adjbound(X As Single, Y As Single, blockno As Integer) As Integer
   adjbound = -1
   For u = 0 To NoBlock - 1
      If u <> blockno Then
        For o = 0 To 6 Step 2
          If midpoint(u, o) = X And midpoint(u, o + 1) = Y Then
            adjbound = u
          End If
        Next
      End If
   Next
End Function

'*****************************************************
'Procedure Name   SolveK()
'Usage            Calculate stiffness matrix using Global coordinate system
'
'*****************************************************

Private Sub solveK(Xc As Single, Yc As Single, length As Single, angle As Single, ks As Single,
Kn As Single, ByRef K() As Single)
    For q = 0 To 2
   For p = 0 To 2
    K(q, p) = 0
   Next
   Next
```

```
    Dim Lamda(2, 2)      As Single
    Dim LamdaT(2, 2)     As Single
    Dim Vector(2, 2)     As Single
    Dim temp3(2, 2)      As Single
    Dim Kl(2, 2)         As Single

    Call Cal_Lamda(angle, Lamda())
    objMatrix.tran Lamda(), LamdaT()
    Call Cal_Kl(length, ks, Kn, Kl())
    objMatrix.mult LamdaT, Kl, temp3
    objMatrix.mult temp3, Lamda, K
End Sub
'*****************************************************
'Procedure Name   MidPoint()
'Usage            Calculate transformation matrix
'
'*****************************************************

Private Sub Cal_Lamda(angle As Single, ByRef Lamda() As Single)

    For q = 0 To 2
    For p = 0 To 2
     Lamda(q, p) = 0
    Next
    Next
    angle = (angle * (22 / 7)) / 180
    Lamda(0, 0) = Round(Cos(angle), 2)
    Lamda(0, 1) = Round(Sin(angle), 2)
    Lamda(1, 0) = Round(-1 * Sin(angle), 2)
    Lamda(1, 1) = Round(Cos(angle), 2)
    Lamda(2, 2) = 1

End Sub

'*****************************************************
'Procedure Name   Cal_length()
'Usage            Calculate length for each side
'
'*****************************************************

Private Function Cal_length()

    For q = 0 To NoBlock - 1
    For p = 0 To 3
     length(q, p) = 0
    Next
    Next
    For t = 0 To NoBlock - 1
     length(t, 0) = ((Node(t, 2) - Node(t, 0)) ^ 2 + (Node(t, 3) - Node(t, 1)) ^ 2) ^ 0.5
     length(t, 1) = ((Node(t, 4) - Node(t, 2)) ^ 2 + (Node(t, 5) - Node(t, 3)) ^ 2) ^ 0.5
     length(t, 2) = ((Node(t, 6) - Node(t, 4)) ^ 2 + (Node(t, 7) - Node(t, 5)) ^ 2) ^ 0.5
```

```
    length(t, 3) = ((Node(t, 0) - Node(t, 6)) ^ 2 + (Node(t, 1) - Node(t, 7)) ^ 2) ^ 0.5

    Next
End Function

'*****************************************************
'Procedure Name   Cal_Kl()
'Usage            Calculate local stiffness matrix elements
'                (Result of integration over length)
'*****************************************************

Private Sub Cal_Kl(length As Single, ks As Single, Kn As Single, ByRef Kl() As Single)
   For q = 0 To 2
   For p = 0 To 2
    Kl(q, p) = 0
   Next
   Next

   Kl(0, 0) = Kn * length
   Kl(1, 1) = ks * length
   Kl(2, 2) = (Kn * (length ^ 3)) / 12
End Sub

'*****************************************************
'Procedure Name   IsRefBlock()
'Usage            Check whether the blocks are boundary blocks
'
'*****************************************************

Private Function IsRefBlock(Xc As Single, Yc As Single) As Boolean
 IsRefBlock = False
 For p = 0 To NoOfRefBlock - 1
  If Xc = RefCent(p, 0) And Yc = RefCent(p, 1) Then
   IsRefBlock = True
   Exit For
  End If
 Next
End Function




'*****************************************************
'Usage         set whether the input is from file or interface
'
'*****************************************************


Private Sub ckInput_Click()
```

```
If ckInput.Value = vbChecked Then
 txtinput.Enabled = True
 cmdInput.Enabled = True
Else
 txtinput.Enabled = True
 cmdInput.Enabled = True
End If
End Sub

'*****************************************************
'Usage         output displacement vector
'
'*****************************************************


Private Sub cmdDisplacement_Click()
On Error GoTo ErrHandler
'open the file Displacement with notepad
OpenfilewithNotepad (strpreface & "Displacement.txt")
   Exit Sub
ErrHandler:
    MsgBox Err.Description
End Sub

'*****************************************************
'Usage         output factor of safety
'
'*****************************************************

Private Sub cmdFactor_Click()
On Error GoTo ErrHandler
'open the file Factor of safety with notepad
 OpenfilewithNotepad (strpreface & "FactorOfSafety.txt")
   Exit Sub
ErrHandler:
   MsgBox Err.Description

End Sub




'*****************************************************
'Usage         output assembled stiffness matrix
'
'*****************************************************


Private Sub cmdAssStifMat_Click()
On Error GoTo ErrHandler
'open the file Factor of safety with notepad
```

```
 OpenfilewithNotepad (strpreface & "AssStifMat.txt")
    Exit Sub
ErrHandler:
    MsgBox Err.Description

End Sub

'*****************************************************
'Usage         output unstable blocks
'
'*****************************************************

Private Sub cmdUnstable_Click()
On Error GoTo ErrHandler
'open the file Displacement with notepad
OpenfilewithNotepad (strpreface & "Unstable.txt")
    Exit Sub
ErrHandler:
     MsgBox Err.Description
End Sub

'*****************************************************
'Usage         output yeilded blocks
'
'*****************************************************


Private Sub cmdYield_Click()
On Error GoTo ErrHandler
'open the file Displacement with notepad
OpenfilewithNotepad (strpreface & "Yield.txt")
    Exit Sub
ErrHandler:
     MsgBox Err.Description
End Sub
'interface operations follow


'*****************************************************
'Usage         input initializing
'
'*****************************************************


Private Sub Form_Load()
    Set objMatrix = New CMatrix
    o = 0: p = 0: q = 0: u = 0: t = 0
    c = 0: n = 0: f = 0: a = 0
    r = 0: s = 0: g = 0
    SBar.Panels(3).Text = Now
End Sub
```

```
'*****************************************************
'Usage          output error if input causes error
'
'*****************************************************
Private Sub txtNoBlock_LostFocus()
On Error GoTo ErrHandler
 NoBlock = Val(txtNoBlock)
 If NoBlock <> 0 Then
  Call ReDim_Variables
  txtNoBlock.Enabled = False
 End If
 Exit Sub

ErrHandler:
   MsgBox Err.Description
   Exit Sub
End Sub

'*****************************************************
'Usage          input boundary blocks
'
'*****************************************************


Private Sub txtNoRefBlock_LostFocus()
 NoOfRefBlock = Val(txtNoRefBlock)
 ReDim RefCent(NoOfRefBlock, 1) As Single
End Sub

'*****************************************************
'Usage          input centroid of blocks
'
'*****************************************************
Private Sub cmdCentroid_Click()
    If c < NoBlock Then
        Centroid(c, 0) = CSng(txtCentX)
        Centroid(c, 1) = CSng(txtCentY)
        c = c + 1
        txtCentX = ""
        txtCentY = ""
        txtCentX.SetFocus
    End If
    If c = NoBlock Then
        txtCentX.Enabled = False
        txtCentY.Enabled = False
    End If
End Sub
```

```
'*****************************************************
'Usage          input centroid of blocks
'
'*****************************************************
Private Sub cmdNode_Click()
   If n < NoBlock Then
      For q = 0 To 7
         Node(n, q) = CSng(txtNode(q))
         txtNode(q) = ""
      Next
         n = n + 1
         txtNode(0).SetFocus
   End If
   If n = NoBlock Then
      For q = 0 To 8 - 1
      txtNode(q).Enabled = False
      Next
   End If
End Sub

'*****************************************************
'Usage          input force
'
'*****************************************************

Private Sub cmdForce_Click()
   If f < NoBlock Then
      Force(f, 0) = CSng(txtForcex)
      Force(f, 1) = CSng(txtForcey)
      Force(f, 2) = CSng(txtForcew)
      f = f + 1
      txtForcex = ""
      txtForcey = ""
      txtForcew = ""
      txtForcex.SetFocus
   End If
   If f = NoBlock Then
      txtForcex.Enabled = False
      txtForcey.Enabled = False
      txtForcew.Enabled = False
   End If
End Sub

'*****************************************************
'Usage          input boundary block
'
'*****************************************************
Private Sub cmdRefCent_Click()
   If r < NoOfRefBlock Then
      RefCent(r, 0) = CSng(txtRefCentx)
      RefCent(r, 1) = CSng(txtRefCenty)
```

```
        txtRefCentx = ""
        txtRefCenty = ""
            r = r + 1
        txtRefCentx.SetFocus
    End If
    If r = NoOfRefBlock Then
        txtRefCentx.Enabled = False
        txtRefCenty.Enabled = False
    End If
End Sub

'*****************************************************
'Usage          input transformation angle
'
'*****************************************************

Private Sub cmdangle_Click()
    If g < NoBlock Then
        For q = 0 To 3
            angle(g, q) = CSng(txtangle(q).Text)
            txtangle(q) = ""
        Next
            g = g + 1
            txtangle(0).SetFocus
    End If
    If g = NoBlock Then
        For q = 0 To 3
        txtangle(q).Enabled = False
        Next
    End If
End Sub

'*****************************************************
'Usage          command to restart operation
'
'*****************************************************

Private Sub cmdReset_Click()
    Unload Me
    Dim frm As Form
    Set frm = New frminput
    frm.Show
End Sub
'*****************************************************
'Usage          runs the program(analyse)
'
'*****************************************************

Private Sub cmdCalculate_Click()
On Error GoTo ErrHandler
    Dim i As Integer
```

```
    For i = 0 To 1000
      YieldBlock(i, 0) = -1
      YieldBlock(i, 1) = -1
      UnstableBlock(i) = -1
    Next
    If ckInput.Value = vbChecked Then
       Call ReadFromFile
    Else
       Call Read_Input
       Call WriteToFile
    End If
    Call Cal_midpoint
    Call Cal_length
    Call BlockCal

    Call OutputDisplacement      'output displacement
    Call OutputFactorOfSafety    'output factor of safty
    Call outputUnstable          'output displacement
    Call OutputYield             'output factor of safty

    Exit Sub
ErrHandler:
  If Err.Number = 13 Then
    MsgBox Err.Description & vbCr & " You may not fill some of the input OR You may feed
wrong input. ", vbCritical, "Input Error"
  Else
    MsgBox Err.Description
  End If
End Sub

'*****************************************************
'Usage          select file path
'
'*****************************************************

Private Sub cmdInput_Click()
    With cdinput
       .DialogTitle = "open"
       .CancelError = False

       .Filter = "All Files (*.txt)|*.txt*"
       .ShowOpen
       If Len(.FileName) = 0 Then
          Exit Sub
       End If
       Me.txtinput = .FileName
    End With
    strInput = Me.txtinput
End Sub
```

```
'*****************************************************
'Usage          open notepad
'
'*****************************************************

Private Sub cmdNotepad_Click()
 Shell "Notepad", vbNormalFocus
End Sub

'end of interface

'*****************************************************
'Usage          output displacement to file
'
'*****************************************************


Private Sub OutputDisplacement()
   ReDim dispvec(NoBlock - 1, 3)

   For q = 0 To NoBlock - 1
   For p = 0 To 3
   dispvec(q, p) = -1
   Next
   Next

   For q = 0 To NoBlock - 1
    For p = 0 To 2
      dispvec(q, p) = Displacement(q * 3 + p)
    Next
   Next
   Dim message As String
   Open strpreface & "Displacement.txt" For Output As #1
      Print #1, " Block No     x              y              w" & vbCr
      For q = 0 To NoBlock - 1
       message = q & vbTab
       For p = 0 To 2
          message = message & dispvec(q, p) & vbTab
       Next
          message = message & vbCr
          Print #1, message
      Next
   Close #1
End Sub
```

```
'*****************************************************
'Usage          open files with notepad
'
'*****************************************************

Private Sub OpenfilewithNotepad(str As String)
    str = "Notepad " & App.Path & "/" & str
    Shell str, vbNormalFocus
End Sub

'*****************************************************
'Usage          read input from interface
'
'*****************************************************

Private Sub Read_Input()
On Error GoTo ErrHandler

 For q = 0 To NoBlock - 1
   For p = 0 To 7
    Stiffness(q, p) = txtStifness(p)
   Next
 Next

 Cp = CSng(txtCp)
 Pp = CSng(txtPp)
 Cr = CSng(txtCr)
 Pr = CSng(txtPr)
 Ru = CSng(txtRu)

 NOIteration = CInt(txtNoIteration)
 Del_p = CSng(txtDel_r)
 Del_r = CSng(txtDel_r)
 Exit Sub
ErrHandler:
   Err.Raise Err.Number
End Sub

'*****************************************************
'Usage          write input to file
'
'*****************************************************


Private Sub WriteToFile()
Dim strval As String
    Open strInput For Output As #1

        Print #1, NoBlock

        For q = 0 To NoBlock - 1
```

```
    strval = ""
    For p = 0 To 1
        strval = strval & Centroid(q, p) & vbTab
    Next
    Print #1, strval
Next

For q = 0 To NoBlock - 1
    strval = ""
    For p = 0 To 7
        strval = strval & Node(q, p) & vbTab
    Next
    Print #1, strval
Next

For q = 0 To NoBlock - 1
    strval = ""
    For p = 0 To 2
        strval = strval & Force(q, p) & vbTab
    Next
    Print #1, strval
Next

Print #1, NoOfRefBlock

For q = 0 To NoOfRefBlock - 1
    strval = ""
    For p = 0 To 1
        strval = strval & RefCent(q, p) & vbTab
    Next
    Print #1, strval
Next

Print #1, Cp & vbTab & Pp & vbTab & Cr & vbTab & Pr

For q = 0 To NoBlock - 1
    strval = ""
    For p = 0 To 7
      strval = strval & Stiffness(q, p) & vbTab
    Next
Next
For q = 0 To NoBlock - 1
    strval = ""
    For p = 0 To 3
        strval = strval & angle(q, p) & vbTab
    Next
    Print #1, strval
Next

Print #1, NOIteration
Print #1, Del_p
```

```
        Print #1, Del_r
        Print #1, Ru

    Close #1
End Sub

'*****************************************************
'Usage          read input from file
'
'*****************************************************

Private Sub ReadFromFile()
    Open strInput For Input As #1
        Input #1, NoBlock
        Call ReDim_Variables
        For q = 0 To NoBlock - 1
            For p = 0 To 1
                Input #1, Centroid(q, p)
            Next
        Next

        For q = 0 To NoBlock - 1
            For p = 0 To 7
                Input #1, Node(q, p)
            Next
        Next

        For q = 0 To NoBlock - 1
            For p = 0 To 2
                Input #1, Force(q, p)
            Next
        Next

        Input #1, NoOfRefBlock

        ReDim RefCent(NoOfRefBlock - 1, 1)

        For q = 0 To NoOfRefBlock - 1
            For p = 0 To 1
                Input #1, RefCent(q, p)
            Next
        Next

        Input #1, Cp, Pp, Cr, Pr

        For q = 0 To NoBlock - 1
            For p = 0 To 7
              Input #1, Stiffness(q, p)
            Next
        Next
```

```
        For q = 0 To NoBlock - 1
           For p = 0 To 3
              Input #1, angle(q, p)
           Next
        Next

        Input #1, NOIteration
        Input #1, Del_p
        Input #1, Del_r
        Input #1, Ru

    Close #1
End Sub

'*****************************************************
'Usage          redimension variables
'
'*****************************************************

Private Sub ReDim_Variables()
    ReDim Centroid(NoBlock - 1, 1) As Single
    ReDim Node(NoBlock - 1, 7) As Single
    ReDim Force(NoBlock - 1, 2) As Single
    ReDim Area(NoBlock - 1) As Single
    ReDim BoundP(NoBlock - 1, 7) As Single
    ReDim Stiffness(NoBlock - 1, 7) As Single
    ReDim angle(NoBlock - 1, 3) As Single
    ReDim length(NoBlock - 1, 3) As Single
    ReDim midpoint(NoBlock - 1, 7) As Single
End Sub

'*****************************************************
'Usage          output values of factor of safety
'
'*****************************************************

Private Sub OutputFactorOfSafety()
Dim message As String
Open strpreface & "FactorOfSafety.txt" For Output As #1
    Print #1, "The Factor of Safety for each block are:" & vbCr
    Print #1, "B.No   Side    Factor of safety" & vbCr
    For t = 0 To NoBlock - 1
    For q = 0 To 3
     If FactorOfSafety(t, q) <> -1 Then
     Print #1, t & vbTab & q & vbTab & FactorOfSafety(t, q) & vbCr
     End If
    Next
    Next
Close #1
End Sub
```

```
'*****************************************************
'Usage          output yield block
'
'*****************************************************

Private Sub OutputYield()
Dim message As String
Open strpreface & "Yield.txt" For Output As #1
    Print #1, "Yield plot for each block are:" & vbCr
    Print #1, "Block No  side x    side y" & vbCr
    For t = 0 To 1000
     If YieldBlock(t, 0) <> -1 Then
        Print #1, t & vbTab & YieldBlock(t, 0) & vbTab & YieldBlock(t, 1) & vbCr
     End If
    Next
Close #1
End Sub

'*****************************************************
'Usage          output unstable blocks
'
'*****************************************************

Private Sub outputUnstable()
Dim message As String
Open strpreface & "Unstable.txt" For Output As #1
    Print #1, "Unstable Blocks are:" & vbCr
    Print #1, "Block No" & vbCr
    For t = 0 To 1000
     If UnstableBlock(t) <> -1 Then
     Print #1, UnstableBlock(t) & vbCr
     End If
    Next
Close #1
End Sub


'*****************************************************
'Usage          read prefix text from text box
'
'*****************************************************

Private Sub txtpreface_Change()
strpreface = txtpreface.Text
End Sub
```

Appendix C

Relevant output of SSDEM for example slopes

Factor of safety of example one
Case 1

Block No	Side name	FS
42	3	7.37
43	3	4.42
44	3	1.15
45	3	0.69
46	3	1.07
47	3	0.26
48	3	1.35
49	3	0.26
50	3	0.79
51	3	1.99
Average FS		**1.93**

Factor of safety of example one
Case 2

Block No	Side name	FS
42	3	7.02
43	3	3.73
44	3	1.07
45	3	0.69
46	3	0.96
47	3	0.03
48	3	1.2
49	3	0.01
50	3	0.83
51	3	1.05
Average FS		**1.66**

Global Displacements for example two

Block No	x	y	w
0	0	-520.1	0
1	0	-520.1	0
2	0	-520.1	0
3	0	-520.1	0
4	0	-520.1	0
5	0	-520.1	0
6	0	-520.1	0
7	0	-520.1	0
8	0	-520.1	0
9	0	-520.1	0
10	5.10E-03	-2.35E-02	-2.15E-03
11	1.33E-02	-5.59E-02	-1.20E-02
12	-1.23E-02	-5.46E-02	4.39E-03
13	-4.67E-03	-7.13E-02	-0.01238
14	-3.88E-02	-8.48E-02	-2.02E-03
15	-7.29E-02	-9.32E-02	3.61E-03
16	-5.26E-03	-6.39E-02	7.31E-03
17	0.1363661	-0.11723	-3.81E-02
18	8.03E-02	-0.12002	-5.13E-03
19	-3.54E-03	5.14E-02	6.00E-02
20	4.47E-02	-7.52E-02	-3.10E-02

Block No	x	y	w
21	0.0219759	-8.78E-02	1.99E-02
22	2.76E-02	-0.12635	-3.37E-02
23	-1.57E-02	-0.11302	2.66E-02
24	-2.69E-02	-0.16669	-3.92E-02
25	-6.13E-02	-0.15245	3.63E-02
26	-5.47E-03	-0.20709	-5.06E-02
27	-2.71E-02	-0.22431	4.97E-02
28	-1.02E-02	-0.03171	0.106073
29	-6.20E-02	7.39E-02	-3.76E-02
30	7.23E-02	-8.09E-02	2.57E-02
31	8.68E-02	-0.16791	-5.47E-02
32	2.92E-02	-0.14273	4.60E-02
33	1.84E-03	-0.1654	-4.73E-02
34	-5.65E-02	-0.2099	2.84E-02
35	1.25E-02	-0.271	-5.85E-02
36	1.62E-02	-0.23022	5.92E-02
37	-0.206098	-0.12949	3.09E-03
38	-0.140687	-0.20035	-5.52E-02
39	-2.13E-02	5.81E-02	2.06E-02
40	3.83E-02	-0.27318	-1.53E-02
41	4.82E-02	-0.23503	2.49E-02
42	0.0321866	-0.37985	-6.93E-02
43	3.56E-02	-0.3589	6.92E-02

Block No	X	Y	w
44	3.42E-02	-0.32835	-5.62E-02
45	-3.21E-02	-0.30661	5.89E-02
46	-4.07E-02	-0.14952	1.18E-02
47	-2.59E-02	-0.23729	-5.16E-02
48	5.03E-02	-0.327	1.77E-02
49	-6.46E-02	-0.13255	6.12E-02
50	2.93E-02	-0.25965	4.43E-02
51	3.22E-02	-0.31145	-4.43E-02
52	5.32E-02	-0.36933	0.053961
53	-6.99E-03	-0.37707	-4.93E-02
54	-4.92E-02	-0.35415	4.99E-02
55	-9.53E-02	-0.33998	-3.08E-02
56	-0.179803	-0.28476	0.05546
57	-0.217327	-0.44515	-6.84E-02
58	-0.331869	-0.28172	0.121529
59	-0.216994	-0.13851	-4.73E-02
60	5.01E-03	-0.36934	-0.1434
61	4.86E-02	-0.38718	3.48E-02
62	-3.17E-02	-0.38342	-0.05501
63	-0.091526	-0.30979	7.39E-02
64	-0.130809	-0.38089	-9.30E-02
65	-0.217662	-0.35117	9.86E-02
66	-0.261565	-0.4144	-0.12531

Block No	X	y	w
67	-0.252577	-0.46511	9.36E-02
68	-2.90E-02	-0.2809	9.39E-02
69	-9.54E-02	-0.40413	-9.00E-02
70	-0.122093	-0.34449	9.28E-02
71	-0.163983	-0.41692	-0.10945
72	-0.179167	-0.42193	9.37E-02
73	-0.188153	-0.5257	-0.1166

Factor of safety of example three

Block No	Side name	FS
16	3	0.70
17	0	0.39
17	3	0.01
22	0	4.19

Average FS **1.32**

Block No	x	y	w
0	-40	-72	0
1	-40	-297	0
2	-40	-603	0
3	-40	-603	0
4	-40	-450	0
5	-40	-450	0
6	-40	-450	0
7	-40	-225	0
8	-2.39E-03	-1.73E-02	3.74E-03
9	2.60E-03	-1.92E-02	-2.69E-03
10	3.46E-03	-3.71E-02	-4.42E-03
11	5.59E-03	-5.04E-02	-4.80E-03
12	5.65E-03	-5.48E-02	-3.59E-03
13	5.68E-03	-5.50E-02	-3.69E-03
14	4.17E-03	-5.02E-02	-2.62E-03
15	1.39E-03	-0.03651	-2.19E-04
16	2.23E-02	-8.27E-02	-5.82E-03
17	2.18E-02	-9.54E-02	-2.00E-03
18	2.08E-02	-9.82E-02	-2.72E-03

Block No	X	Y	w
19	1.83E-02	-9.57E-02	-8.37E-04
20	1.51E-02	-8.62E-02	-1.49E-03
21	3.52E-02	-0.1126	-1.34E-03
22	3.17E-02	-0.11783	-1.60E-03
23	2.87E-02	-0.11713	5.78E-04
24	2.57E-02	-0.11325	-2.26E-03

DECLARATION