To the Graduate Council:

I am submitting herewith a thesis written by Ming Zhao, entitled "Keyblock Stability Under Self-Weight and Surface Forces." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Civil Engineering.

Dr. Matthew Mauldon
Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of the Graduate School

KEYBLOCK STABILITY UNDER SELF-WEIGHT AND SURFACE FORCES

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Ming Zhao

December 1995

DEDICATION

To my wife, Jennie (Huaxin) Zhou, and my son, Paul (Yiping) Zhao, whose love, support, and sacrifice made my educational journey possible. Thank you.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to:

Dr. Matthew Mauldon, my major professor, for his arrangement of financial support through the whole process of my studies, his guidance in the research, and his encouragement and lots of time working with me.

Drs. Eric C. Drumm and Karen Chou for their being the members of my committee. Their advice is always helpful.

Dr. Elden L. DePorter, Professor of Industrial Engineering, for his instructions on linear programming coding techniques.

The faculty and staff in the Department of Civil and Environmental Engineering for their instructions and help.

ABSTRACT

This thesis presents a method to define the unstable regions in tunnels in hard, jointed rock. The method takes account of both the block self-weight and the forces on the block from in situ stresses. This problem is statically indeterminate in general, but bounds are found on the solution using the linear programming method. It is found that there is usually an upper limit to the size of unstable keyblocks, based on in-situ stresses, excavation geometry, fracture orientations and shear strength. In most cases, the unstable regions are much smaller than the maximum keyblock regions predicted by block theory. The results are used to determine the maximum size of unstable keyblocks in tunnels.

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

CHAPTER 1

INTRODUCTION

In tunnel design, most designers use rock mass classifications to determine the support forces. However, the validity of empirical design methods based on general rock classifications is questionable in highly discontinuous rock formations. This is because the conventional rock classification methods represent an overall rock quality criterion, which may not represent any specific segment of a tunnel. In this case, block theory is specially useful to determine a tunnel support force. Block theory was first presented systematically by Richard Goodman and Gen-hua Shi in 1985 to deal with highly jointed rock masses. Block theory defines a keyblock by the orientations of joint planes and the excavation geometry. A keyblock is a finite, removable, and unstable block of rock which is isolated from the rest of the rock mass by the intersection of rock joints, and excavation surfaces. Figure 1-1 illustrates a keyblock in a tunnel roof. The dashed lines show the maximum removable region for joint orientations as shown.

The support force is determined by block theory based on the volume and density of the maximum keyblock. However, the stability of the maximum keyblock depends not only on the block self-weight, but also on the surface forces acting on the block. In general the problem is statically indeterminate in two dimensions if both the block self-weight and the block surface forces are included. Based on block theory, Karzulovic (1988) and Brady and Brown (1993) analysed the stability of the maximum keyblock under the block self-weight and the block surface forces. Karzulovic (1988)

designed tunnel supports for symmetric blocks in which the problem is statically determinate. Brady and Brown (1993) applied a joint relaxation concept and solved the statically indeterminate problem by giving the block a vertical displacement.

Block theory works well for blocks of certain shapes. For example, it gives useful results for the left tunnel roof block in Figure 1-2, but not for the right tunnel roof block in Figure 1-2 because the support force as calculated is unrealistically large. In such cases we have to find other ways to analyse the block stability. The research concept discussed in this thesis is that the block surface forces from the in situ stresses may help support the maximum keyblock and prevent failure. The same principle holds for keyblocks smaller than the maximum keyblock (Figure 1-1).

1.1 The Aim of This Thesis

This thesis will examine the combined influence of the in situ stresses and block self-weight on the stability of the blocks in two dimensions. The approach of this thesis is to use linear programming to find limiting conditions for block stability. The linear programming method has been applied to geotechnical engineering for analyzing slope stability (Chuang, 1992) and block stability (Mauldon, 1995). Figure 1-3 is an example of results of our analysis, which shows that the unstable region (in which support is needed) is smaller than the maximum keyblock region. The thesis will show how the result in Figure 1-3 is obtained. The method is applied to a range of initial conditions, leading to the results presented in Chapter 6. Chapters 2 to 3 will discuss the stresses and forces on the block surfaces. Chapter 4 will analyse the block stability. The linear programming models will be formulated in Chapter 5.

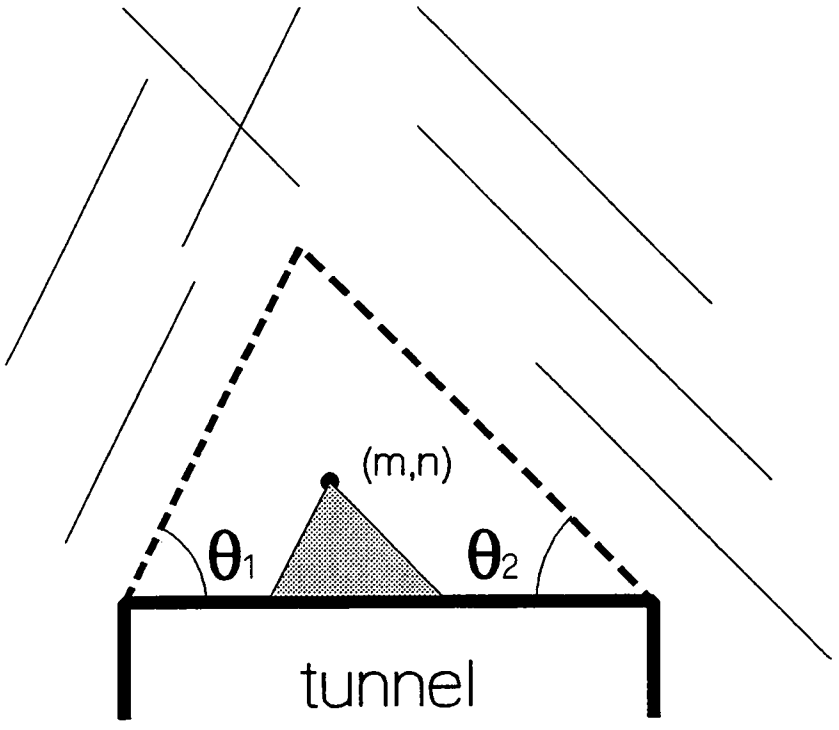

Figure 1-1 Keyblock (shaded) in tunnel roof.

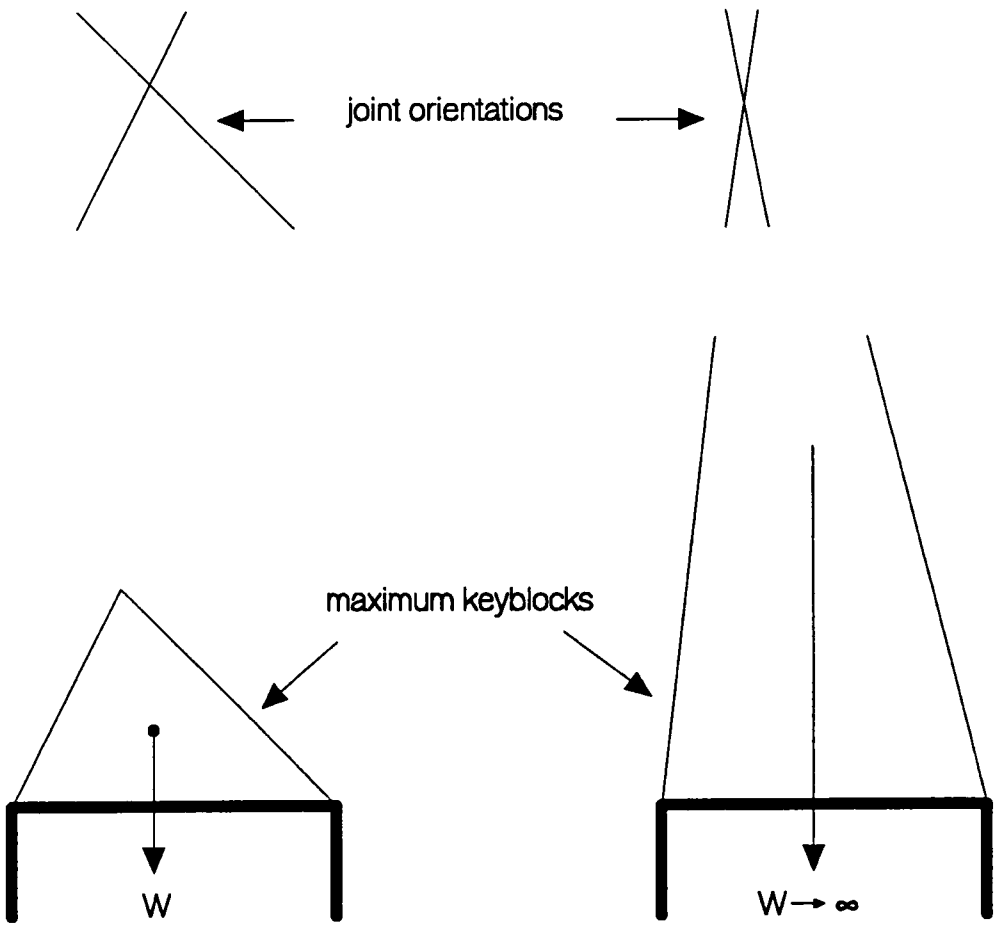

Figure 1-2 Maximum keyblocks based on different joint angles

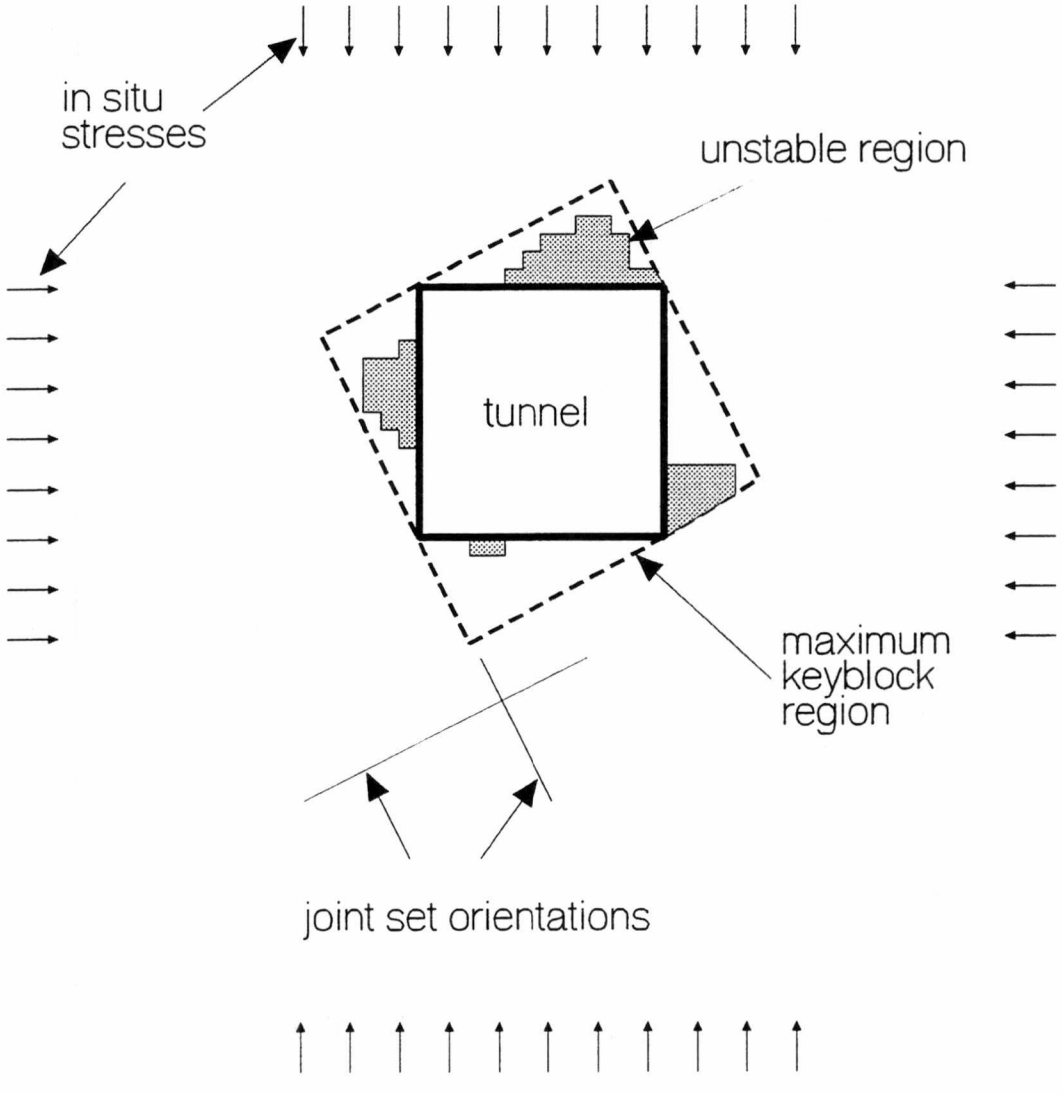

Figure 1-3 Sample result for $k_o = 0.5$, $\tan\phi_1 = \tan\phi_2 = 0.8$. The shaded region corresponds to vertices of unstable blocks.

1.2 Outline of Methodology

To solve the statically indeterminate problems of keyblock stability and find the lower and upper bounds to the tunnel loads, we have developed the following procedure to obtain the necessary results:

(1) Compute the in situ stress around the tunnel;

(2) Compute the initial block surface forces for all potential blocks;

(3) Introduce a variable force P in the same direction as gravity;

(4) Use linear programming to determine P_{max} and P_{min};

(5) Compare the weight W to the P_{max} and P_{min} to check the stability of all potential blocks.

1.3 Assumptions

For practical purposes, we assume that the 2-D fractures are infinite. We consider two joint sets, each with a fixed orientation. For a given tunnel, an intersection of two joints uniquely defines a block if the intersection point is inside the maximum keyblock region. Otherwise no block is formed. Also we assume that the rock mass around the tunnel is an elastic continuum for purposes of determining stresses and block surface forces. This implies that joint deformability has been disregarded in the stress analysis. This assumption is most appropriate for hard and relatively unweathered rock masses.

CHAPTER 2

STRESS ANALYSIS

The block surface properties include the joint orientations, surface geometry, and infill materials (Priest, 1993). The surface properties play important roles in the block stability. To compute the possible surface forces that may increase the stability of the maximum keyblock in the tunnel roof and reduce the support forces for the keyblock, initial stresses, rock deformability, joint relaxation, joint surface friction and joint dilatancy have been included in the keyblock stability analyses by many researchers, as summarized by Goodman (1992). The stresses acting on the block surfaces may be natural or induced (Herget, 1988). The natural stresses are the stresses found in rock mass before excavation, which comprise gravitational stresses, tectonic stresses, residual stresses, and thermal stresses. The induced stresses are the result of stress changes due to manmade excavations. We define the in situ stress state as the stress state around a tunnel which is a result of stress redistribution after the tunnel has been excavated. We should consider the in situ stress state around the tunnel, and design a tunnel support based on the in situ stresses.

The well known analytical solutions widely used in rock mechanics for the distribution of stresses around an opening are Kirsch's solution and Bray's solution. The Kirsch's solution assumes a continuous, linear elastic, and homogenous medium, whereas Bray's solution allows plastic behavior in the vicinity of the tunnel (Goodman, 1989). Figure 2-1 is an example showing how the stress magnitude varies around a

circular tunnel based on these two solutions. In Figure 2-1, σ_ρ and σ_θ are the radial and tangential stresses respectively, and r/a is the ratio of radial distance to the radius of the tunnel. Figure 2-2 shows that the stress direction changes around a tunnel. The major principal stress direction is shown by the long lines, and minor principal stress direction is shown by the short lines. Note that the principal stresses tend to arch over the tunnel opening.

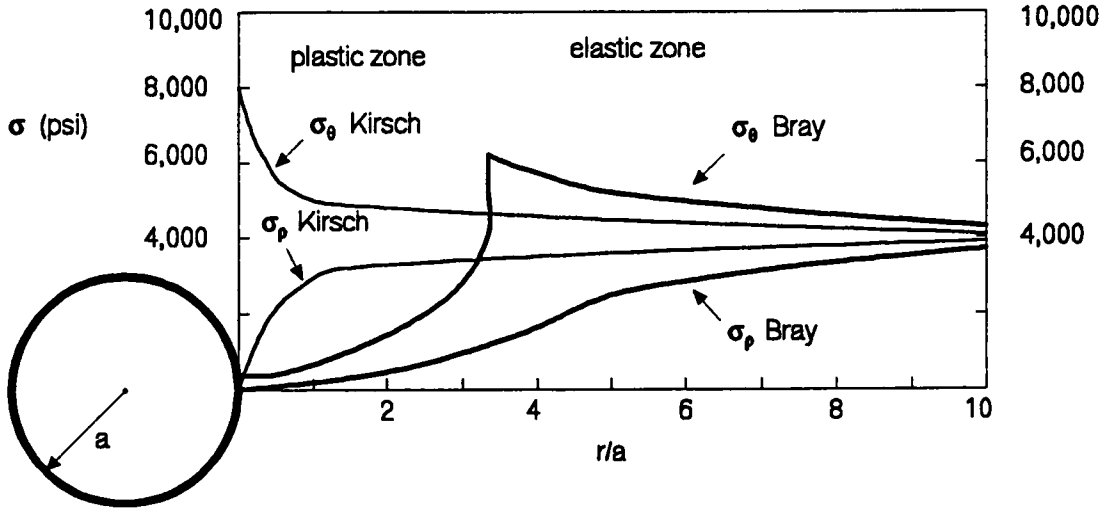

Figure 2-1 Stress magnitude variation around a tunnel (Goodman, 1989)

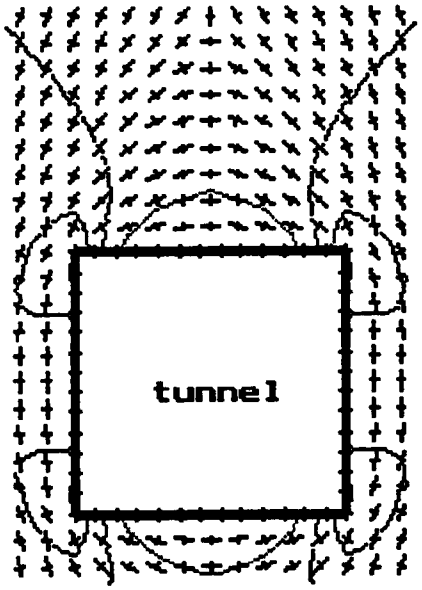

Figure 2-2 Stress direction variation around a tunnel.

It is difficult to find the exact surface forces of a block by integrating the Kirsch or Bray's equations over the block surface because too much computation is involved. To determine the in situ sresses, we use Examine 2-D (Curran and Corkum, 1989), a 2D boundary elementary program for calculating stresses around underground excavations in rock. The block surface forces can be found by computing the surface force of every element on the joint surface forming the block, and summing up all the element surface forces. This Chapter will discuss the determination of the block boundary stresses. The block surface forces will be discussed in Chapter 3. Following Brady & Brown (1993), we assume initially that the normal and shear stiffness of joints is sufficiently high that the joints do not affect the stress distribution.

2.1 Normal and shear stresses on a joint surface

Taking the orientations of the two joint surfaces to be fixed, the stresses obtained from Examine 2D at each node are transformed into normal and shear stresses on joint planes of these particular orientations. The following equations are used to calculate the stresses on joints of a given orientation (see Figure 2-3).

$$\begin{aligned} \sigma &= \sigma_x \cos^2(\alpha) + \sigma_y \sin^2(\alpha) + \tau_{xy} \sin(2\alpha) \\ \tau &= -0.5\, \sigma_x \sin(2\alpha) + 0.5\, \sigma_y \sin(2\alpha) + \tau_{xy} \cos(2\alpha) \end{aligned} \tag{2.1}$$

where σ = normal stress on the joint surface

τ = shear stress on the joint surface

σ_x = normal stress in horizontal direction

σ_y = normal stress in vertical direction

α = angle between the joint surface normal and positive x direction (see Figure 2-3)

τ_{xy} = shear stress in horizontal and vertical directions

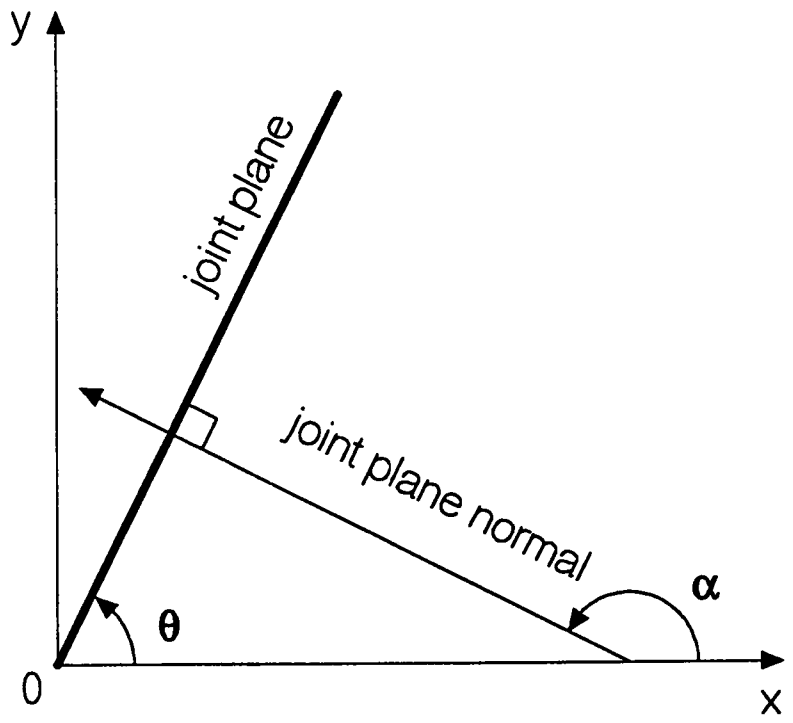

Figure 2-3 Definition of α and θ. Note that θ is the dip of the joint plane.

CHAPTER 3

NORMAL AND SHEAR FORCES ON BLOCK

3.1 Block Definition

Figure 3-1 shows the grid cells in the tunnel roof. Given the orientations of the two joint sets, each grid cell defines a potential block. In Figure 3-1, for example, in which both joint angles are equal to 45°. (a) block (2,4) is the block with apex at (2,4); (b) block(3,6) is the block with apex at (3,6); and (c) block (9,7) is the block with apex at (9,7). The region underneath the solid line corresponds to removable blocks; thus blocks (2,4) and (3,6) are removable, whereas block (9,7) is non-removable.

3.2 Joint Orientations

The angle θ_1 is the dip (angle below horizontal) of the left-dipping plane, and θ_2 is the dip of the right-dipping plane. θ_1 and θ_2 are chosen such that $\theta = \arctan(n)$, where n is a rational number. Then each block side can be represented as a path through cell midpoints. An example is shown in Figure 3-2, in which the block is defined by cell coordinates (m,n), with $\theta_1 = \arctan(2)$ and $\theta_2 = \arctan(1)$.

3.3 Shear and normal forces on block surfaces from in situ stresses

The computation of block surface forces resulting from in situ stresses is as follows.

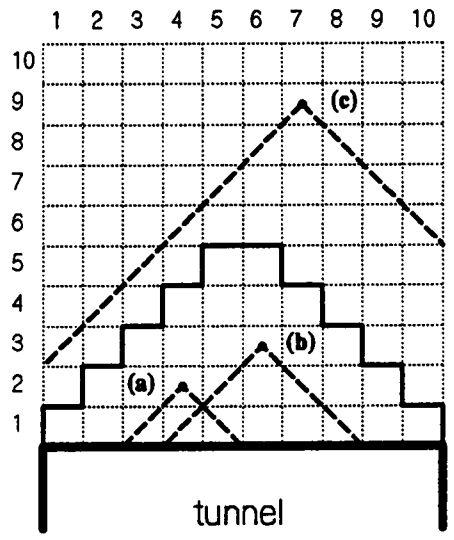

Figure 3-1 Block definition: (a) block (2,4); (b) block (3,6); (c) block (9,7).

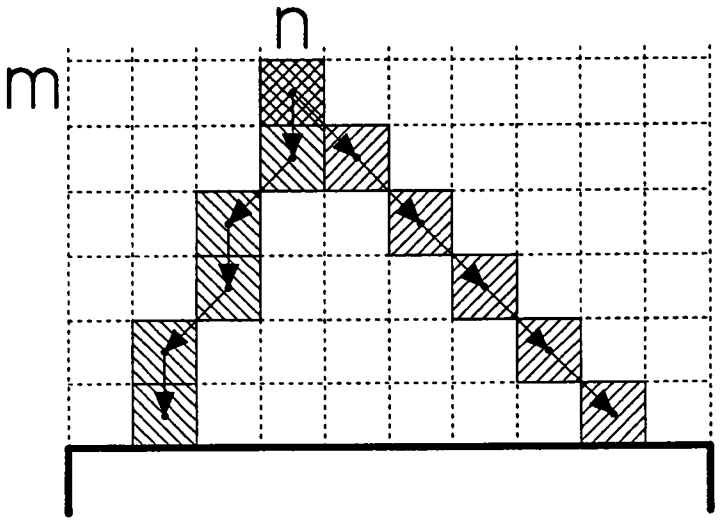

Figure 3-2 Shaded cells defining the surface of block (m,n) for θ_1 = arctan(2), θ_2 = arctan(1).

Each cell (Figure 3-2) has dimension 1×1. The normal and shear forces are found by forming the sum of stress components multiplied by the increments of length in each grid cell. For the block in Figure 3-2 the normal and shear forces N_1 and S_1 on plane 1, and N_2 and S_2 on plane 2, are given by

$$N_1 = \left(\sigma_{(1)}(m,n) + \sigma_{(1)}(m-1,n) + \sigma_{(1)}(m-2,n-1) + \ldots\right) / \sin(\theta_1) \tag{3.1}$$

$$S_1 = \left(\tau_{(1)}(m,n) + \tau_{(1)}(m-1,n) + \tau_{(1)}(m-2,n-1) + \ldots\right) / \sin(\theta_1) \tag{3.2}$$

$$N_2 = \left(\sigma_{(2)}(m,n) + \sigma_{(2)}(m-1,n-1) + \sigma_{(2)}(m-2,n-2) + \ldots\right) / \sin(\theta_2) \tag{3.3}$$

$$S_2 = \left(\tau_{(2)}(m,n) + \tau_{(2)}(m-1,n-1) + \tau_{(2)}(m-2,n-2) + \ldots\right) / \sin(\theta_2) \tag{3.4}$$

where $\sigma_{(i)}$ and $\tau_{(i)}$ denote the normal and shear stresses on joint plane i.

The directions of positive normal and shear forces N_1 , S_1 , N_2 and S_2 are shown in Figure 3-3, This sign convention, referred to the block geometry, is maintained for all blocks. The magnitudes of N_1 , S_1 , N_2 and S_2 are calculated by a QuickBasic program (refer to Appendix) based upon the definitions of stresses and forces discussed above.

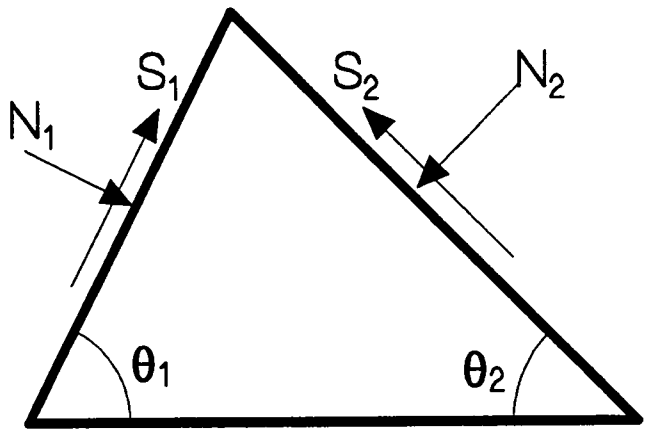

Figure 3-3 Positive directions of initial forces on a block

3.4 Shear and normal forces on block surfaces from block self-weight

There are three kinds of blocks: roof blocks, wall blocks, and floor blocks (see Figure 3-4). We have to analyse them separately because the block self-weight has different effects on the block surface forces.

3.4.1 *Roof Block*

For any roof block we now introduce a force ***P*** through the block centroid. See Figure 3-5. The additional forces from ***P*** on the block surfaces are M_1 , T_1 , M_2 , and T_2 . Figure 3-6 shows unit vectors $\hat{m}_1$ and $\hat{m}_2$ perpendicular to, and $\hat{t}_1$ and $\hat{t}_2$ parallel to the joint planes. We can express the force ***P*** in terms of components in these directions as follows,

$$\boldsymbol{P} = M_1 \hat{m}_1 + M_2 \hat{m}_2 + T_1 \hat{t}_1 + T_2 \hat{t}_2 \tag{3.5}$$

Since the vectors ($\hat{m}_1$, $\hat{m}_2$, $\hat{t}_1$, $\hat{t}_2$) do not form an orthonormal basis, the values of M_1 , M_2 , T_1 and T_2 are non-unique (Figure 3-5). The components of $\boldsymbol{P}$ in the directions $\hat{m}_i$ and $\hat{t}_i$ correspond to adjustments of the block contact forces (Figure 3-7). Figure 3-8 shows the net forces on the block surfaces. We will follow this analysis procedure for the analyses of wall and floor block forces.

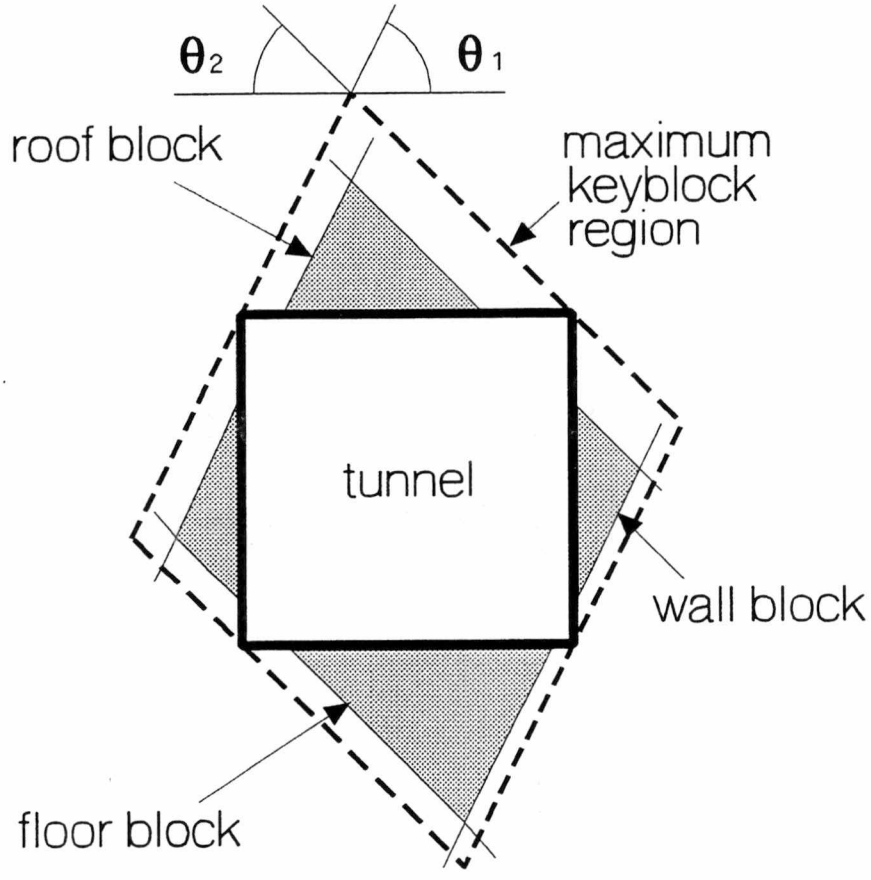

Figure 3-4 Roof, wall and floor blocks

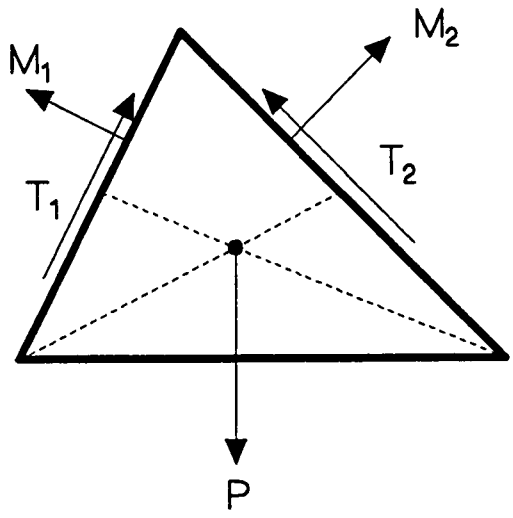

Figure 3-5 Additional surface forces from P on roof block

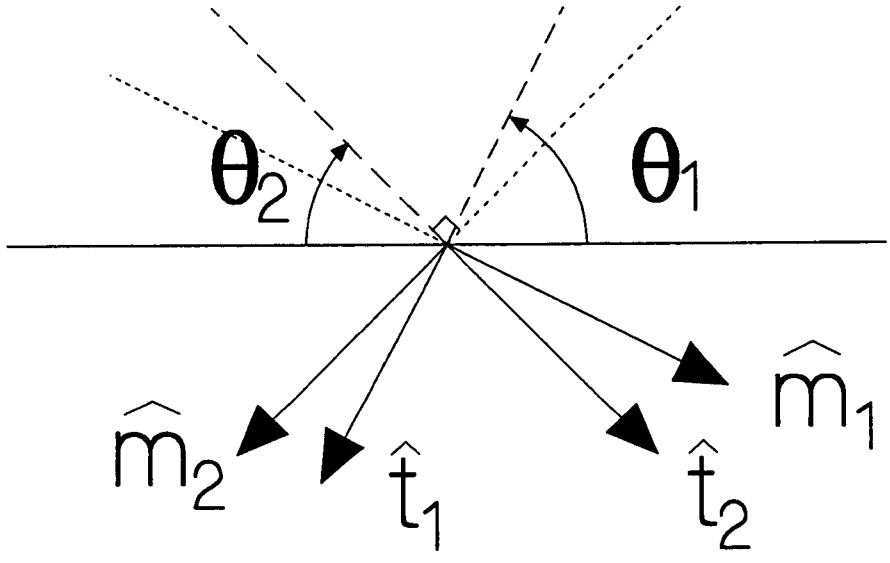

Figure 3-6 Unit vectors showing force directions

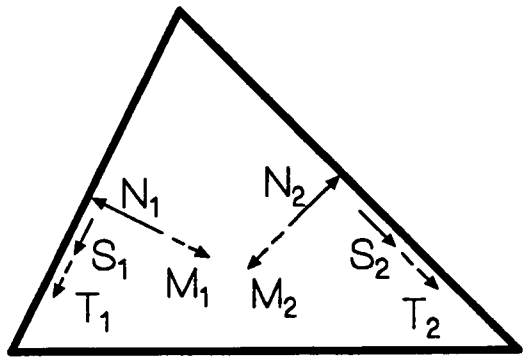

Figure 3-7 Adjustments of contact forces

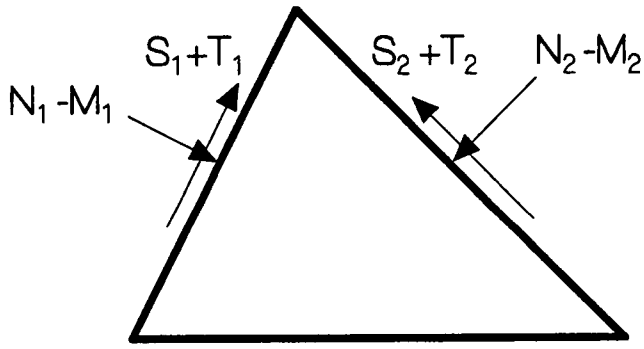

Figure 3-8 Net forces on roof block

3.4.2 *Wall Block*

Figure 3-9 shows a wall block with initial forces from the in situ stresses. Figure 3-10 shows the force components from ***P*** on the block surfaces. The net forces on the block surface are shown in Figure 3-11.

3.4.3 *Floor Block*

Figure 3-12 is a floor block with forces from in situ stresses. Additional forces from ***P*** on the floor block are shown in Figure 3-13. The net forces on the floor block are as shown in Figure 3-14.

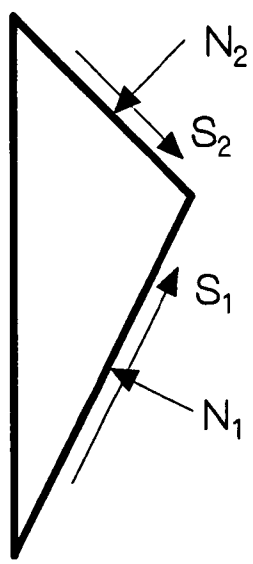

Figure 3-9 Wall block with initial forces from in situ stresses

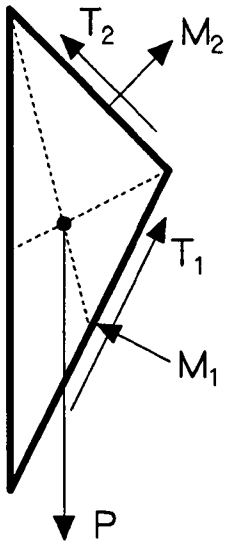

Figure 3-10 Additional forces on wall block from P

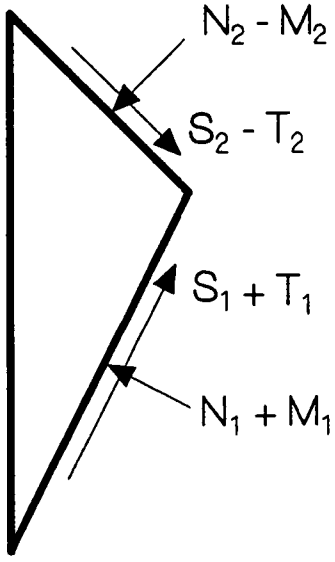

Figure 3-11 Net forces on wall block

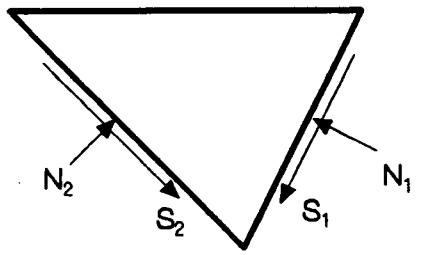

Figure 3-12 Floor block with initial forces from in situ stresses

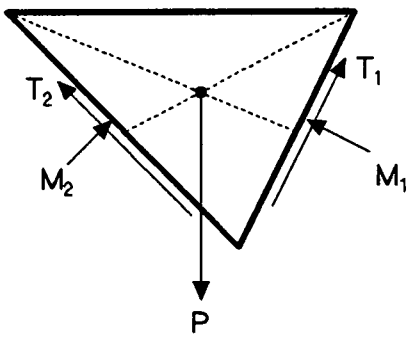

Figure 3-13 Additional forces from P on the floor block

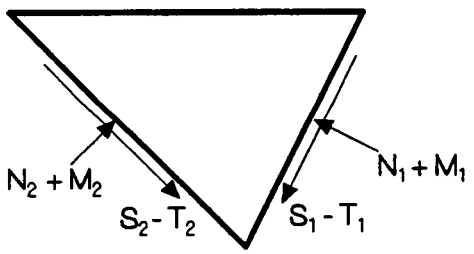

Figure 3-14 Net forces on floor block

CHAPTER 4

BLOCK STABILITY ANALYSIS

Block stability analysis depends on the block stability criteria. For a 2D problem, the block stability is determined by the stability of two surfaces that are in contact with the rock masses. A block surface may fail by tension or slip. If the block surface is moving away from its surrounding rock, the surface fails by tension. If the block surface is sliding along on the adjacent rock, the surface fails by slip. Failure is considered to be either tension or slip, whichever occurs first. Also, we consider both single modes and double modes to define the block stability. The single mode means that a block will fail if one block surface fails. The double mode requires that both block surfaces fail for the block to fail. In this chapter, we will establish the equilibrium conditions for roof, wall, and floor blocks, and then analyse the stability of the three kinds of blocks based on the stability definitions above.

4.1 Equilibrium Conditions

To find P we have to determine the effect of P on the block surfaces. To determine the limits on P for stability we have to use equilibrium equations. The equilibrium equations for roof, wall, and floor blocks are as follows. Note that we do not give the moment equation for each block because we will not use it in the future computation. The effect of the omission of the moment equation will be explained in the linear programming section.

4.1.1 *Roof Block*

The equilibrium equations for the roof block are obtained based on Figure 4-1. The directions of θ_1 and θ_2 are shown in Figure 4-1.

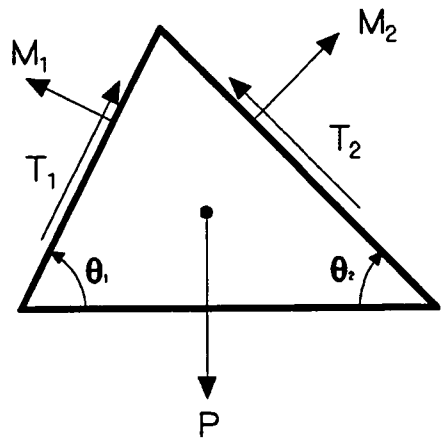

Figure 4-1 Force components from P on roof block

$$M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) - P = 0 \quad (4.1)$$

$$M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0 \quad (4.2)$$

4.1.2 *Wall Block*

The equilibrium equations of the wall block are obtained according to Figure 4-2. The directions of θ_1 and θ_2 are shown in Figure 4-2.

$$M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) - P = 0 \quad (4.3)$$

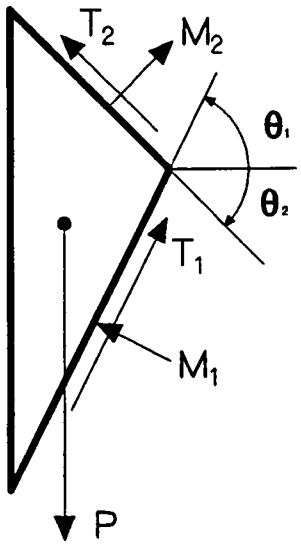

Figure 4-2 Force components from P on wall block

$$M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0 \tag{4.4}$$

4.1.3 *Floor Block*

The equilibrium equations of the floor block are as follows based on Figure 4-3. The directions of θ_1 and θ_2 are shown in Figure 4-3.

$$M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) - P = 0 \tag{4.5}$$

$$M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0 \tag{4.6}$$

4.2 Conditions for Stability

Our discussions for block stability are related to the single and double modes

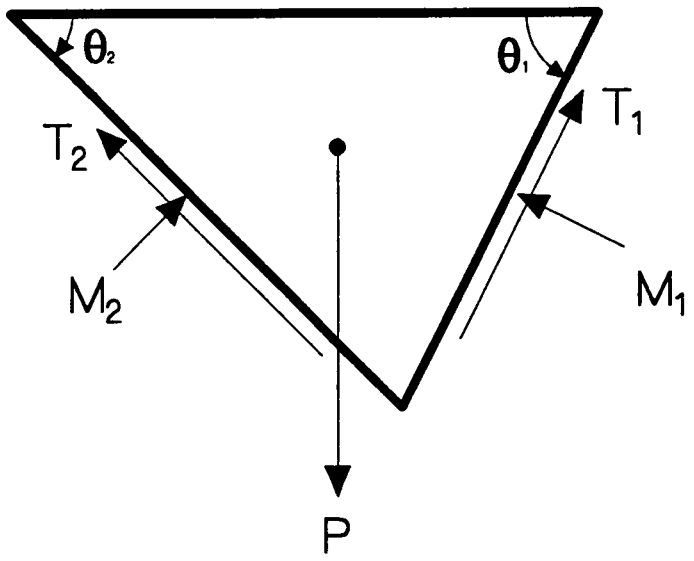

Figure 4-3 Force components from P on floor block

which have been defined at the beginning of this chapter.

4.2.1 *Single Mode*

The mechanical model for single mode can be represented as a series of four links in a chain, as shown in Figure 4-4. Each link in the chain represents a tensile or shear strength. In the single mode, the system will remain stable if all the links are stable. Therefore, the conditions for stability of the single mode are as follows:

no tension on plane 1

& no tension on plane 2

& no slip on plane 1

& no slip on plane 2

4.2.2 *Double Mode*

Similarly, the double mode is represented by two parallel chains, each with two links, as shown in Figure 4-5. The conditions for stability of double mode require either

Figure 4-4 Model of single mode

plane to be stable, as follows:

no tension on plane 1

& no slip on plane 1

or

no tension on plane 2

& no slip on plane 2

Note that there are two possibilities (cases).

4.3 Conditions for Instability

Again, we will discuss the conditions for instability based on single and double modes. Refer to Figure 4-4 and Figure 4-5 for single and double modes respectively.

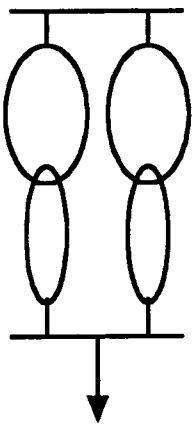

Figure 4-5 Model of double mode

4.3.1 *Single Mode*

For the single mode, the block will fail if any link fails (Figure 4-4). Therefore, the conditions for instability of single mode are:

tension on plane 1

or

tension on plane 2

or

slip on plane 1

or

slip on plane 2

4.3.2 *Double Mode*

There are four possible combinations for the conditions of instability of double

mode, which are:

tension on plane 1

& tension on plane 2

or

tension on plane 1

& slip on plane 2

or

slip on plane 1

& tension on plane 2

or

slip on plane 1

& slip on plane 2

4.4 Stability of Weightless Block

We have defined the conditions for both stability and instability of a block. We will restate the conditions in the form of equations in the linear programming section. The basic equations we will use in the analysis are as follows if we consider the block without weight to be applied by in situ stresses.

1. Block Surface Failure by Tension

The condition for tensile failure on plane i is given by

$$N_i \leq 0$$

where N_i is the magnitude of the normal force from the in situ stresses acting on plane

i. The block will fail whenever a tensile force occurs on the block surface. We assume that the joint has zero tensile strength.

2. Block Surface Failure by Shear

The condition for shear failure on plane i is given by

$$N_i \tan(\phi_i) \leq S_i$$

where N_i is the magnitude of the normal force from the in situ stresses acting on plane i, ϕ_i the friction angle of plane i, and S_i the shear force from the in situ stresses acting on plane i. The expression means that the block surface instability occurs when the shear force exceeds the friction force on the surface.

These two expressions are the basic equations for stability analysis of blocks without weight. Any analysis including additional forces will be based on these two equations. We will use these two equations in the model formulations in the linear programming section.

CHAPTER 5

LINEAR PROGRAMMING

As stated previously, if the block self-weight is included in the stability analysis, the problem is statically indeterminate in general. In a practical tunnel support design, the factor of safety is greater than 1.0, which means that we always provide a support force which is greater than required. In order to achieve this, it is necessary to know the required force. In the absence of additional loads, the required supporting force is proportional to the block weight. We have introduced a force ***P*** acting at the centroid of blocks in the same direction as gravity, as shown in Figures 3-5, 3-10, and 3-13. We determine the limiting (maximum and/or minimum) values of P based on certain conditions, and then compare them to the block weight (W). From the comparison, we will know if the block is stable. Since the block weight plays different roles in different type of block, we have to analyze the three kinds of blocks separately. All the possible limiting values of P for roof, wall, and floor blocks are shown in Table 5-1. Notes that the question marks in Table 5-1 mean that the limiting values depend on: (1) which plane is unstable; (2) single or double modes; (3) friction and force distribution. Using linear programming, we determine limiting values for P corresponding to the shaded cells in Table 5-1. Let's suppose that we have obtained P_{min} and P_{max} values for all blocks. For the roof and wall blocks, the interpretation of the results can be made by Equations 5.1 to 5.3 and Figure 5-1. For the floor block, the interpretation of the results is shown as Equations 5.4 to 5.6 and Figure 5-2. For limit analysis, the theory

of linear programming provides not only an efficient computational technique but also a convenient conceptual framework (Chuang, 1992). We have coded the simplex algorithmic procedure in QuickBasic (Appendix) and obtained P_{min} and P_{max} for the cases shown as shaded cells in Table 5-1.

Table 5-1 Limiting values of P for roof, wall, and floor blocks

roof block	initial state	plus weight	P_{max}	P_{min}
	stable	stable	> 0	- ∞
		unstable	∞	> 0
	unstable	stable	< 0	- ∞
		unstable	∞	< 0
wall block	initial state	plus weight	P_{max}	P_{min}
	stable	stable	> 0	< 0
		unstable		?
	unstable	stable	?	?
		unstable	> 0	< 0
floor block	initial state	plus weight	P_{max}	P_{min}
	stable	stable	∞	< 0
		unstable	< 0	- ∞
	unstable	stable	∞	> 0
		unstable	> 0	- ∞

Comparison of P_{min} and P_{max} to W: roof and wall blocks

Case(1) $W < P_{min} \Rightarrow$ The block is stable. (5.1)

Case(2) $W > P_{max} \Rightarrow$ The block is unstable. (5.2)

Case(3) $P_{min} \leq W \leq P_{max} \Rightarrow$ Potentially unstable. (5.3)

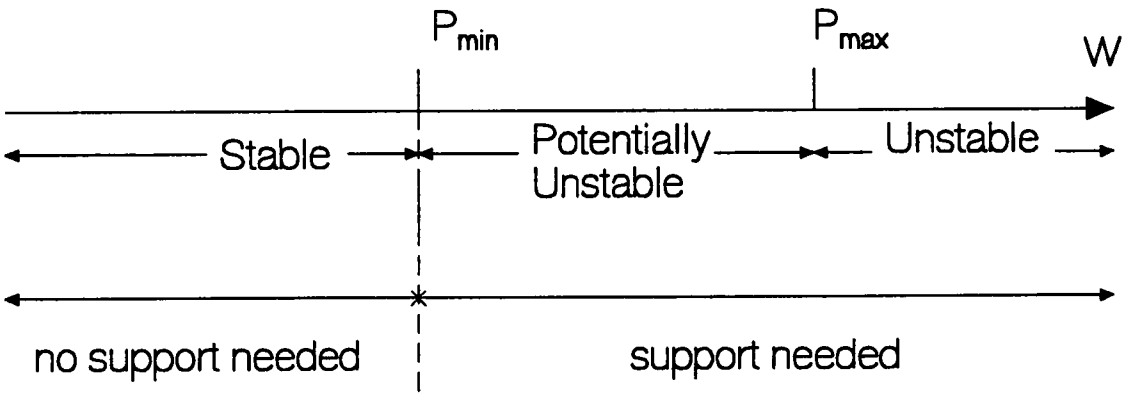

Figure 5-1 Interpretation of support requirements for roof and wall blocks

Note: The conservative value is taken because we consider the potentially unstable region needs supporting.

Comparison of P_{min} and P_{max} to W: floor blocks

Case(1) $W > P_{max} \Rightarrow$ The block is stable. (5.4)

Case(2) $W < P_{min} \Rightarrow$ The block is unstable. (5.5)

Case(3) $P_{min} \leq W \leq P_{max} \Rightarrow$ Potentially unstable. (5.6)

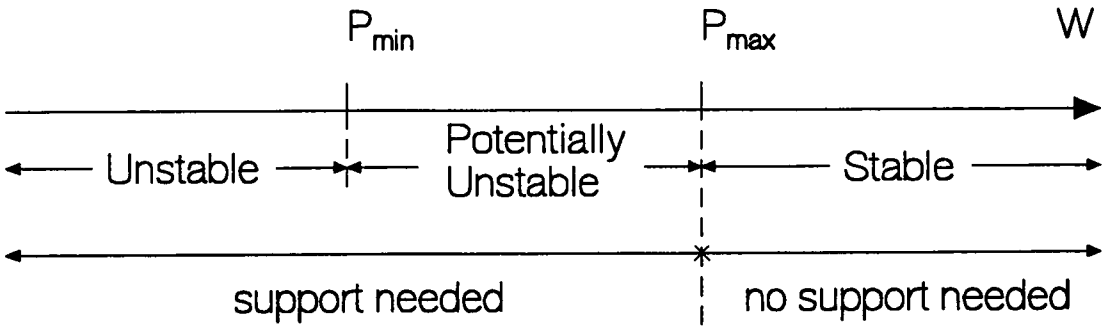

Figure 5-2 Interpretation of support requirements for floor block

Note: The conservative value is taken because we consider the potentially unstable region needs supporting.

5.1 Linear Programming

Linear programming is one of the most important and effective tools in use today in modern management analysis (Gass, 1984). It was first conceived by G.B. Dantzig around 1947. Linear programming is concerned with problems in which a linear objective function in terms of decision variables is to be optimized (i.e., either minimized or maximized) while a set of linear equations, inequalities, and sign restrictions are imposed on the decision variables as requirements. The following examples (Wu & Coppins, 1981) shows graphically how the linear programming works and how a constraint affects the optimal solution.

Linear Programming Model

model 1

Maximize $f = 12x_1 + 8x_2$ (5.7)

Subject to $5x_1 + 2x_2 \leq 150$ (5.8)

$x_1 , x_2 \geq 0$ (5.9)

We can portray the above model graphically by letting the horizontal axis represent x_1 and the vertical axis represent x_2. The nonnegativity constraints mean that only the first quadrant is meaningful. See Figure 5-3 for the graph of the model above. From Figure 5-3, we know that the maximum value of the objective function is 600 when the objective function intersects the corner point A(0,75). This is a simple example of linear programming. We found the maximum value by moving the objective function around the corner points within the feasible solution set.

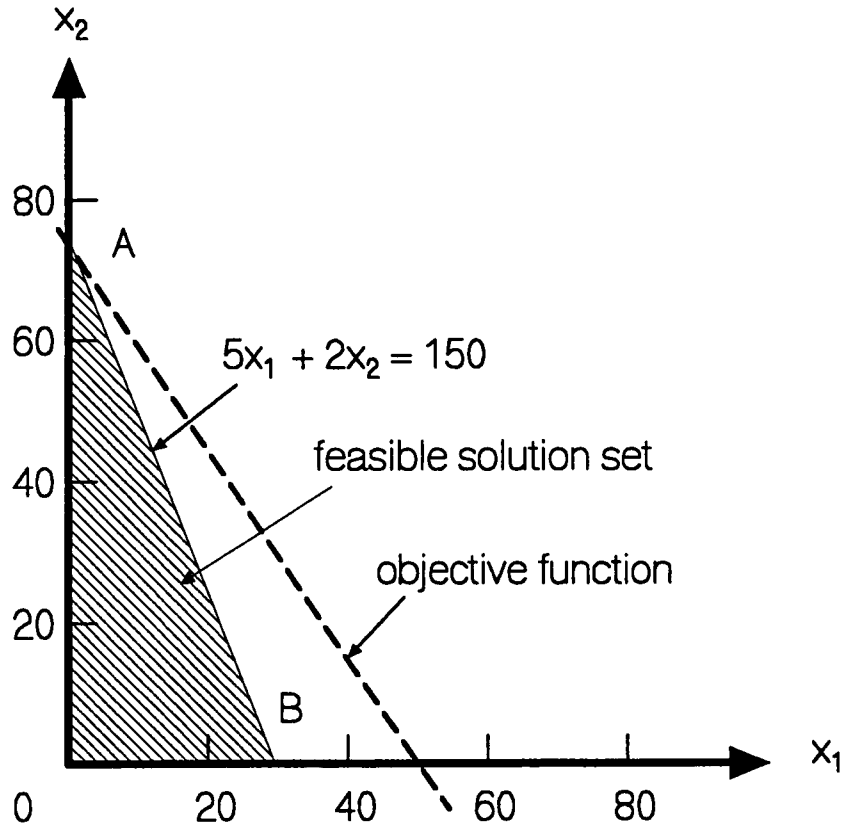

Figure 5-3 Graphical representation of model 1

The Influence of an Additional Constraint on Solution

Let's see how an additional constraint affects the solution. Figure 5-4 is the graphical representation of model 2.

model 2

Maximize	$f = 12x_1 + 8x_2$	(5.10)
Subject to	$5x_1 + 2x_2 \leq 150$	(5.11)
	$4x_1 + 2x_2 \leq 80$	(5.12)

$$x_1 , x_2 \geq 0 \qquad (5.13)$$

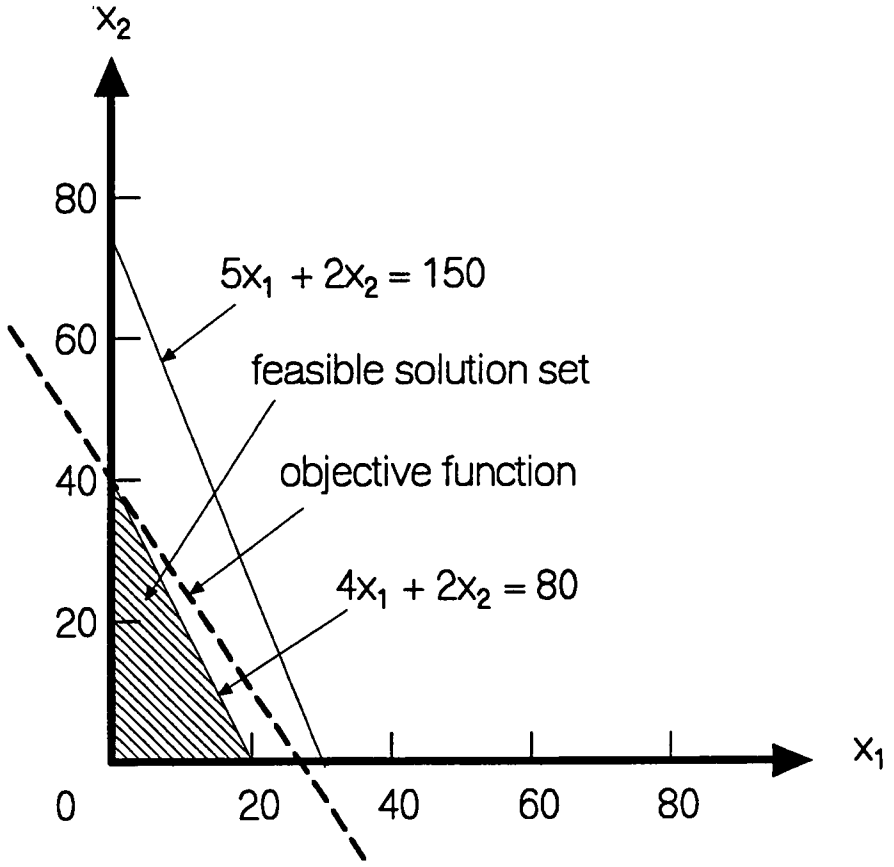

Figure 5-4 Graphical representation of model 2

From Figure 5-4, we know that the maximum value has been reduced from 600 to 320 after adding another constraint. The general rule in linear programming is that additional constraints cannot increase the maximum optimal value, and cannot reduce the minimum optimal value.

As discussed earlier, the moment equation has been omitted from the analysis

of keyblock stability. We now show that this omission has not compromised the results. An additional constraint (such as that from moment equilibrium) can only reduce the solution set; it cannot enlarge it. Thus if Figure 5-5(a) gives the results without considering moment equilibrium, the results with moments are represented schematically in Figure 5-5(b). The stable and unstable fields generally increase at the expense of the potentially unstable field. For design purposes, we recommend that both the unstable and potentially unstable fields from the moment-free analysis are treated as being unstable. As shown in Figure 5-5(c) this designation is conservative.

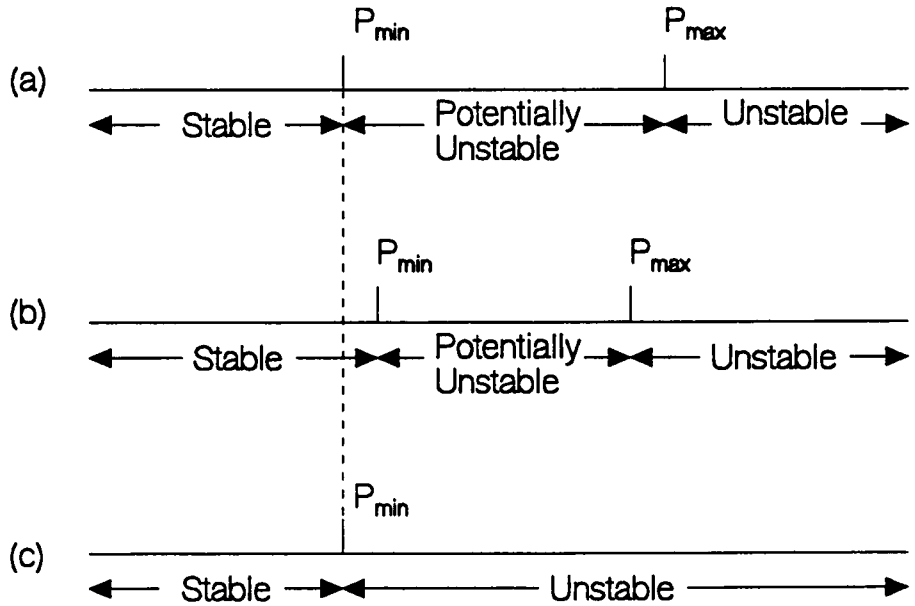

Figure 5-5 Effect of moments and design recommendations for roof and wall blocks.

5.2 Model Formulation

All the linear programming model formulations are based on (1) the net forces on block surfaces; and (2) the definitions of conditions for stability and instability which have been discussed in sections 4.2 to 4.3 and Table 5-1. The decision variables are M_1, M_2, T_1, and T_2 in our linear programming formulations and all of the decision

variables are nonnegative values based on the linear programming rules.

5.2.1 *Roof Block*

The net forces on the roof block are shown in Figure 5-6.

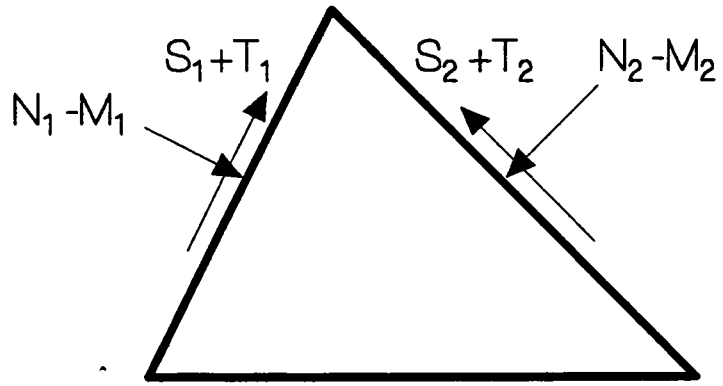

Figure 5-6 Net forces on roof block

The objective function is

$$P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) \tag{5.14}$$

The constraints for single mode are shown in Table 5-2, and for double mode in Table 5-3. We find P_{max} for stability and P_{min} for instability.

Table 5-2 The constraints of single mode for roof block

Roof Block - Single Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}		$M_1 < N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
P_{min}	P_{min-1}	$M_1 \geq N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-2}	$M_1 < N_1$ $M_2 \geq N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-3}	$M_1 < N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 \geq k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-4}	$M_1 < N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 \geq k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: (1) k_1 = coefficient of friction on joint plane 1; k_2 = coefficient of friction on joint plane 2.
(2) P_{max} = maximum P for stability.
(3) P_{min} = minimum of $\{P_{min-1}, P_{min-2}, P_{min-3}, P_{min-4}\}$ for instability.

Table 5-3 The constraints of double mode for roof block

Roof Block - Double Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}	$P_{max\text{-}1}$	$M_1 < N_1$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{max\text{-}2}$	$M_2 < N_2$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
P_{min}	$P_{min\text{-}1}$	$M_1 \geq N_1$ $M_2 \geq N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}2}$	$M_1 \geq N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 < k_1 N_1 - S_1$ $k_2 M_2 + T_2 \geq k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}3}$	$M_1 < N_1$ $M_2 \geq N_2$ $k_1 M_1 + T_1 \geq k_1 N_1 - S_1$ $k_2 M_2 + T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}4}$	$M_1 < N_1$ $M_2 < N_2$ $k_1 M_1 + T_1 \geq k_1 N_1 - S_1$ $k_2 M_2 + T_2 \geq k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: P_{max} = maximum of $\{P_{max\text{-}1}, P_{min\text{-}2}\}$ for stability.
P_{min} = minimum of $\{P_{min\text{-}1}, P_{min\text{-}2}, P_{min\text{-}3}, P_{min\text{-}4}\}$ for instability.

5.2.2 *Wall Block*

The net forces on the wall block surfaces are shown in Figure 5-7.

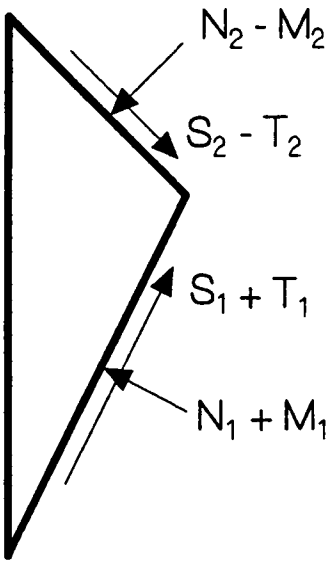

Figure 5-7 Net forces on wall block

The objective function is

$$P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) \tag{5.15}$$

The constraints for single mode are shown in Table 5-4, and for double mode in Table 5-5. We find P_{max} for stability and P_{min} for instability.

Table 5-4 The constraints of single mode for wall block

Wall Block - Single Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}		$M_1 + N_1 > 0$ $M_2 < N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
P_{min}	P_{min-1}	$M_1 + N_1 \le 0$ $M_2 < N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-2}	$M_1 + N_1 > 0$ $M_2 \ge N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-3}	$M_1 + N_1 > 0$ $M_2 < N_2$ $k_1 M_1 - T_1 \le S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-4}	$M_1 + N_1 > 0$ $M_2 < N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 \ge k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: (1) P_{max} = maximum P for stability.
(2) P_{min} = minimum of $\{P_{min-1}, P_{min-2}, P_{min-3}, P_{min-4}\}$ for instability.

Table 5-5 The constraints of double mode for wall block

Wall Block - Double Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}	$P_{max\text{-}1}$	$M_1 + N_1 > 0$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{max\text{-}2}$	$M_2 < N_2$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
P_{min}	$P_{min\text{-}1}$	$M_1 + N_1 \le 0$ $M_2 \ge N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}2}$	$M_1 + N_1 \le 0$ $M_2 < N_2$ $k_1 M_1 - T_1 > S_1 - k_1 N_1$ $k_2 M_2 - T_2 \ge k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}3}$	$M_1 + N_1 > 0$ $M_2 \ge N_2$ $k_1 M_1 - T_1 \le S_1 - k_1 N_1$ $k_2 M_2 - T_2 < k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}4}$	$M_1 + N_1 > 0$ $M_2 < N_2$ $k_1 M_1 - T_1 \le S_1 - k_1 N_1$ $k_2 M_2 - T_2 \ge k_2 N_2 - S_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: (1) P_{max} = maximum of $\{P_{max\text{-}1}, P_{min\text{-}2}\}$ for stability.
(2) P_{min} = minimum of $\{P_{min\text{-}1}, P_{min\text{-}2}, P_{min\text{-}3}, P_{min\text{-}4}\}$ for instability.

5.2.3 *Floor Block*

Figure 5-8 shows the net forces on a floor block.

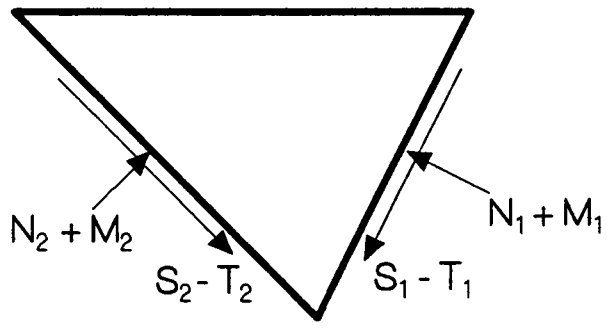

Figure 5-8 Net forces on floor block

The objective function is

$$P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2) \tag{5.16}$$

The constraints for single mode are shown in Table 5-6, and for double mode in Table 5-7. We find P_{max} for instability and P_{min} for stability.

Table 5-6 The constraints of single mode for floor block

Floor Block - Single Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}	$M_1 + N_1 > 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$	
P_{min}	P_{min-1}	$M_1 + N_1 \le 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-2}	$M_1 + N_1 > 0$ $M_2 + N_2 \le 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-3}	$M_1 + N_1 > 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 \le S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	P_{min-4}	$M_1 + N_1 > 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 \le S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: (1) P_{max} = maximum P for instability.
(2) P_{min} = minimum of $\{P_{min-1}, P_{min-2}, P_{min-3}, P_{min-4}\}$ for stability.

Table 5-7 The constraints of double mode for floor block

Floor Block - Double Mode		
objective function: $P = M_1 \cos(\theta_1) + M_2 \cos(\theta_2) + T_1 \sin(\theta_1) + T_2 \sin(\theta_2)$		
P_{max}	$P_{max\text{-}1}$	$M_1 + N_1 > 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{max\text{-}2}$	$M_2 + N_2 > 0$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
P_{min}	$P_{min\text{-}1}$	$M_1 + N_1 \le 0$ $M_2 + N_2 \le 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}2}$	$M_1 + N_1 \le 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 > S_1 - k_1 N_1$ $k_2 M_2 + T_2 \le S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}3}$	$M_1 + N_1 > 0$ $M_2 + N_2 \le 0$ $k_1 M_1 + T_1 \le S_1 - k_1 N_1$ $k_2 M_2 + T_2 > S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$
	$P_{min\text{-}4}$	$M_1 + N_1 > 0$ $M_2 + N_2 > 0$ $k_1 M_1 + T_1 \le S_1 - k_1 N_1$ $k_2 M_2 + T_2 \le S_2 - k_2 N_2$ $M_1 \sin(\theta_1) - M_2 \sin(\theta_2) - T_1 \cos(\theta_1) + T_2 \cos(\theta_2) = 0$

Note: (1) P_{max} = maximum of $\{P_{max\text{-}1}, P_{min\text{-}2}\}$ for instability.
(2) P_{min} = minimum of $\{P_{min\text{-}1}, P_{min\text{-}2}, P_{min\text{-}3}, P_{min\text{-}4}\}$ for stability.

CHAPTER 6

RESULTS AND DISCUSSIONS

Based on the linear programming formulations, the models were coded in QuickBasic for the roof and wall blocks to obtain unstable regions from the program. For the floor blocks, we used Quattro Pro to obtain the results. Figures 6-1 to 6-6 show the maximum keyblock regions based on block theory; and vertices of potentially unstable and unstable blocks based on our computation. In the cases we discussed, all the unstable areas are smaller than those from block theory. This is because the in situ stresses stabilize the blocks. This is what we want to find out in this research.

The cases we examined are for a constant field stress σ_v = 1.35 MPa , σ_h = 1.35 MPa and for a stress ratio, K_o = 1.0, where $K_o = \sigma_h/\sigma_v$. The value of σ_v = 1.35 MPa was chosen based on an average rock mass density of 0.027 MPa/m^3 at the depth of 50 meters. If the depth is less than 50 meters, our results are similar to those from block theory. Therefore, our results should be applied to where a tunnel is located at least 50 meters under the ground surface.

The analysis for the wall blocks and the floor blocks was based on stresses calculated for the rectangular grid above the roof. This stress grid was rotated and used for the wall and floor calculation, with joint angles as shown in Figure A-1 of the Appendix. This is valid only because the cases studied for the wall and floor were for K_0 = 1 and constant vertical stress, i.e., an isotropic stress state. For values of $K_0 \neq$ 1, the stresses for the wall and floor should be read directly from the boundary element

code. This capacity is not part of the QuickBasic programs as written.

From Figure 6-8, there is no significant difference between the unstable areas from single and double modes. Therefore, we used double mode results for Figures 6-1 to 6-6. In Figures 6-1 to 6-6, the unstable regions shown correspond to the vertices of blocks which are either unstable or potentially unstable according to the criteria given in Figures 5-1 and 5-2. The stress ratio has different effects on unstable regions with different joint dip (see Figure 2-3 for the definition of dip). The general picture can be seen from Figure 6-7. From Figure 6-7, the unstable area in the roof reduces with the increase of stress ratio if the joint dip is greater than 45°. On the other hand, the unstable area in the roof will increase with the stress ratio if the stress ratio is greater than 1.0 and joint dip is less than 45°. Based on block theory, the floor blocks are stable. However, in some cases, the floor blocks are unstable based on our research. (see Figures 6-1, 6-2, 6-4, 6-5).

A general tendency for the roof blocks is that the unstable area decreases with the increase of joint dip (Figure 6-8). If we vary the friction coefficient, the unstable area will reduce as friction increases (Figure 6-9).

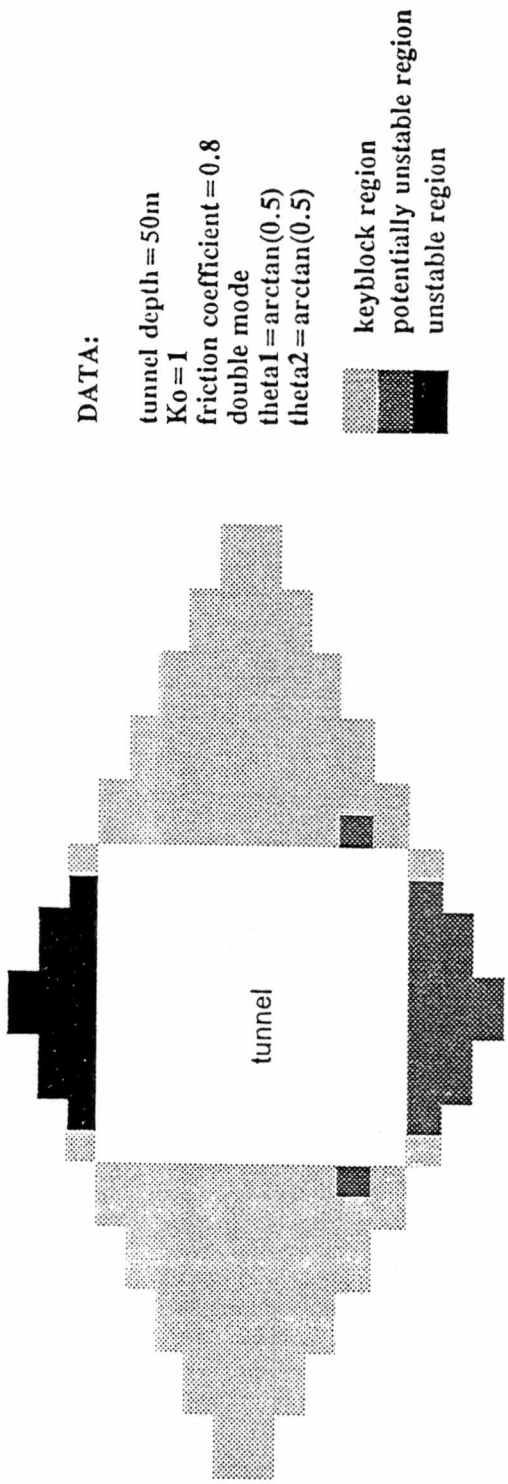

Figure 6-1 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa and $\theta_1 = \theta_2 = \arctan(0.5)$.

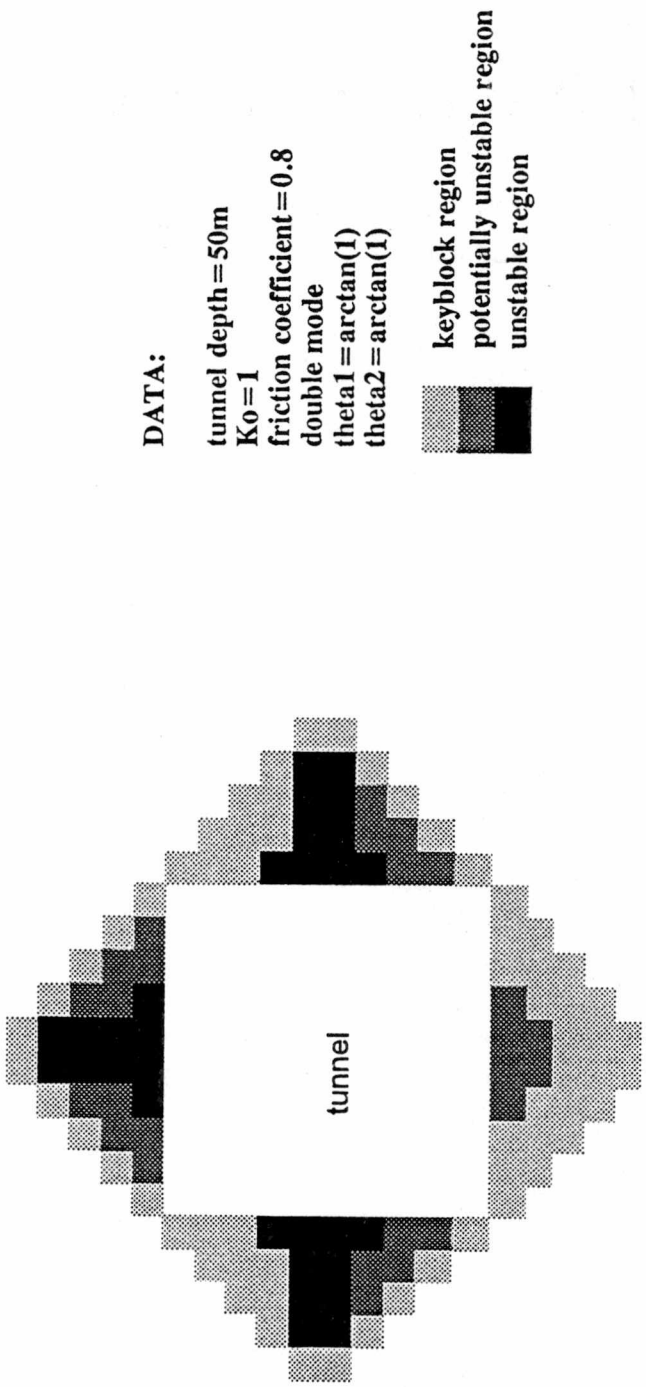

Figure 6-2 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa and $\theta_1 = \theta_2 = \arctan(1)$.

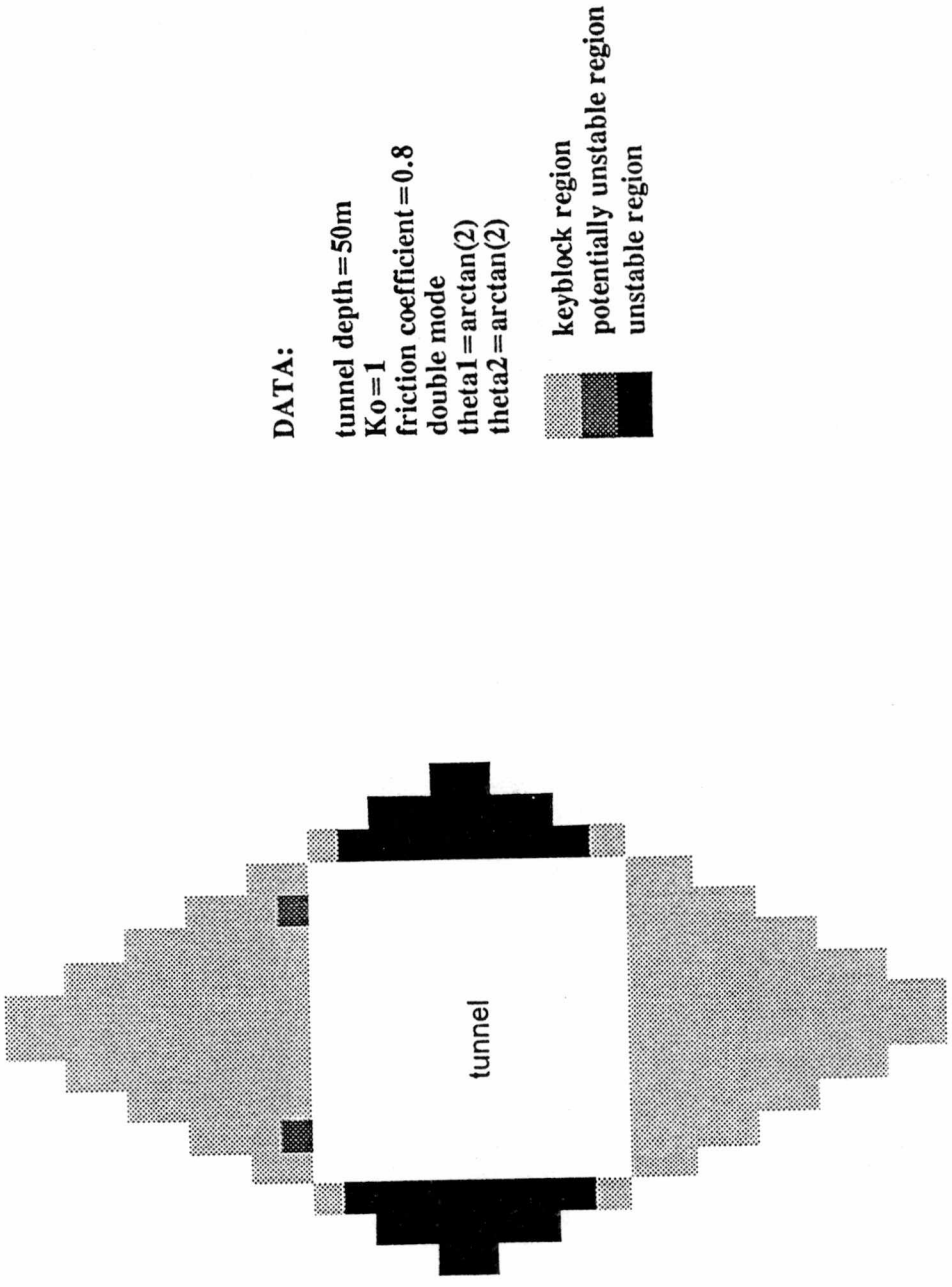

Figure 6-3 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa and $\theta_1 = \theta_2 = \arctan(2)$.

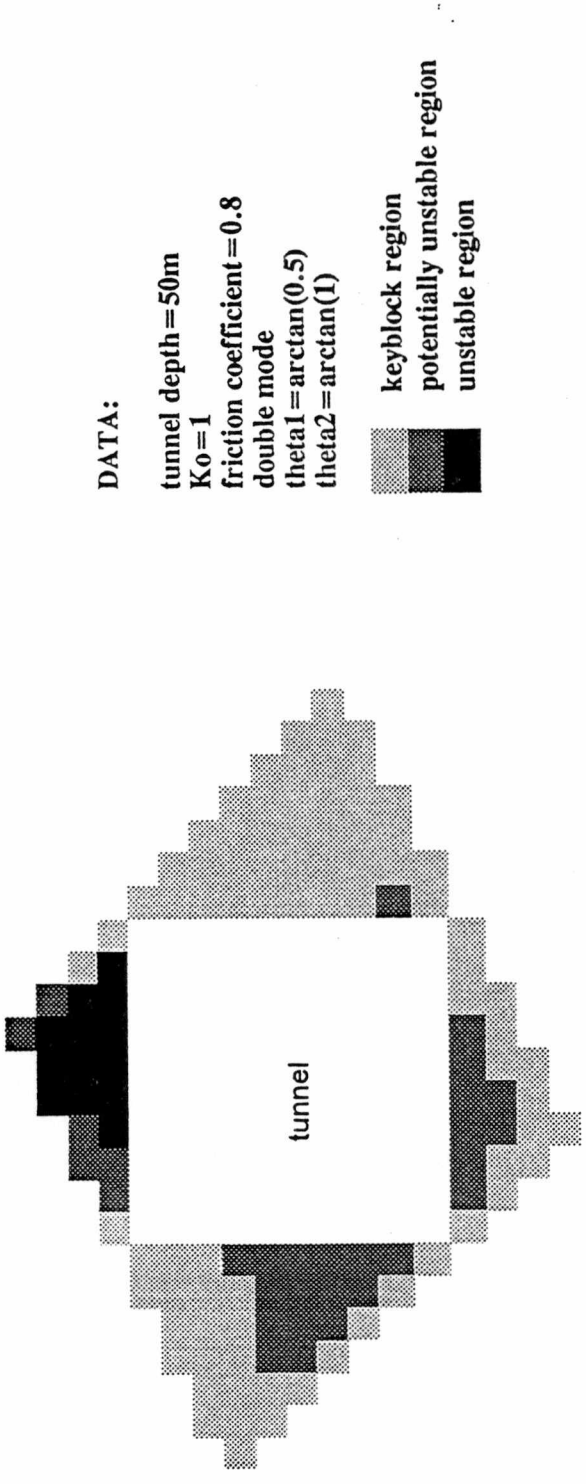

Figure 6-4 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa , $\theta_1 = \arctan(0.5)$ and $\theta_2 = \arctan(1)$.

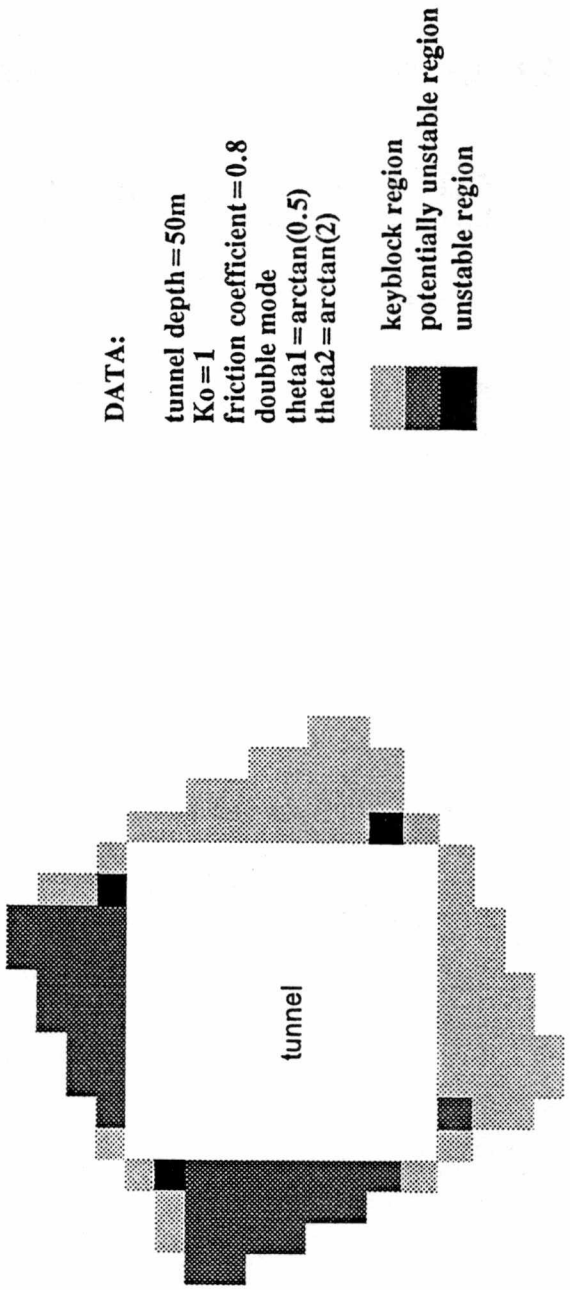

Figure 6-5 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa , $\theta_1 = \arctan(0.5)$ and $\theta_2 = \arctan(2)$.

Figure 6-6 Maximum block region and unstable block region with constant field stress $\sigma_v = \sigma_h = 1.35$ MPa , $\theta_1 = \arctan(1)$ and $\theta_2 = \arctan(2)$.

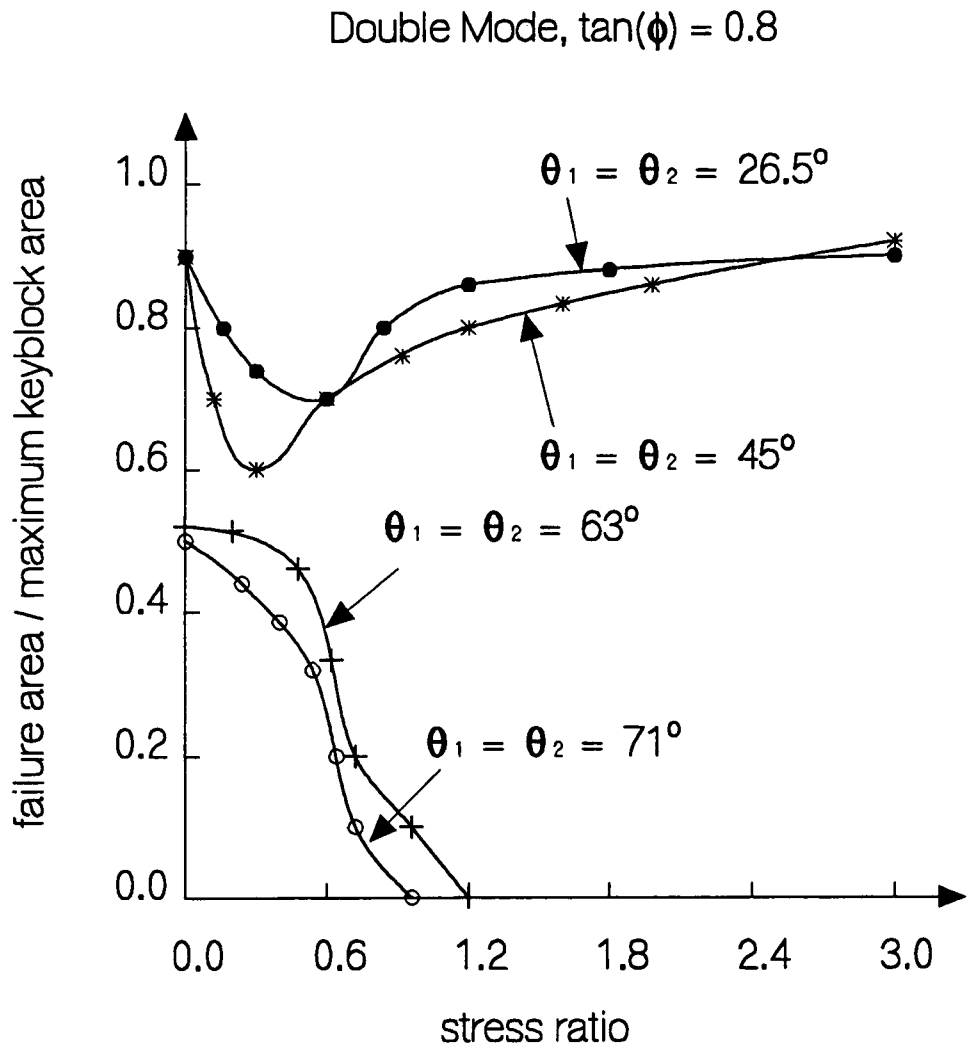

Figure 6-7 Roof unstable area variation as a function of stress ratio. Tanϕ = coefficient of friction, constant field stress $\sigma_v = \sigma_h = 1.35$ MPa.

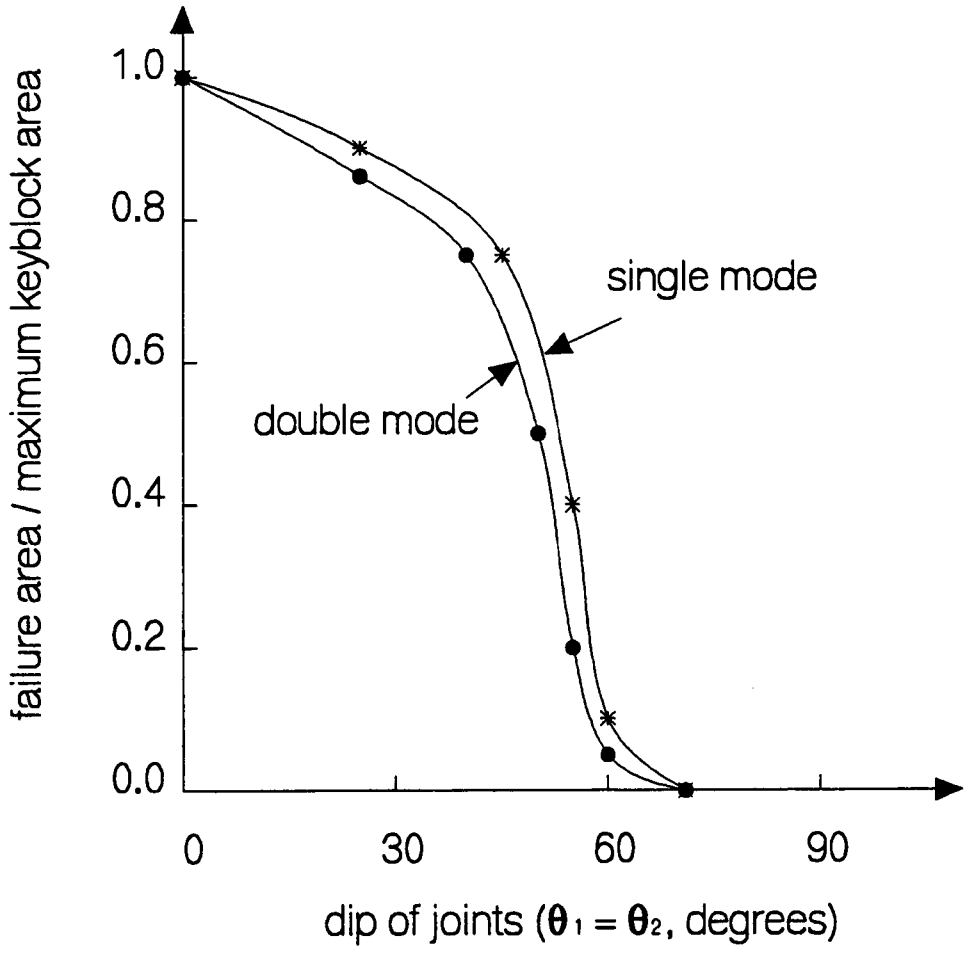

Figure 6-8 Roof unstable area variation as a function of joint dip. Coefficient of friction = 0.8, constant field stress $\sigma_v = \sigma_h = 1.35$ MPa.

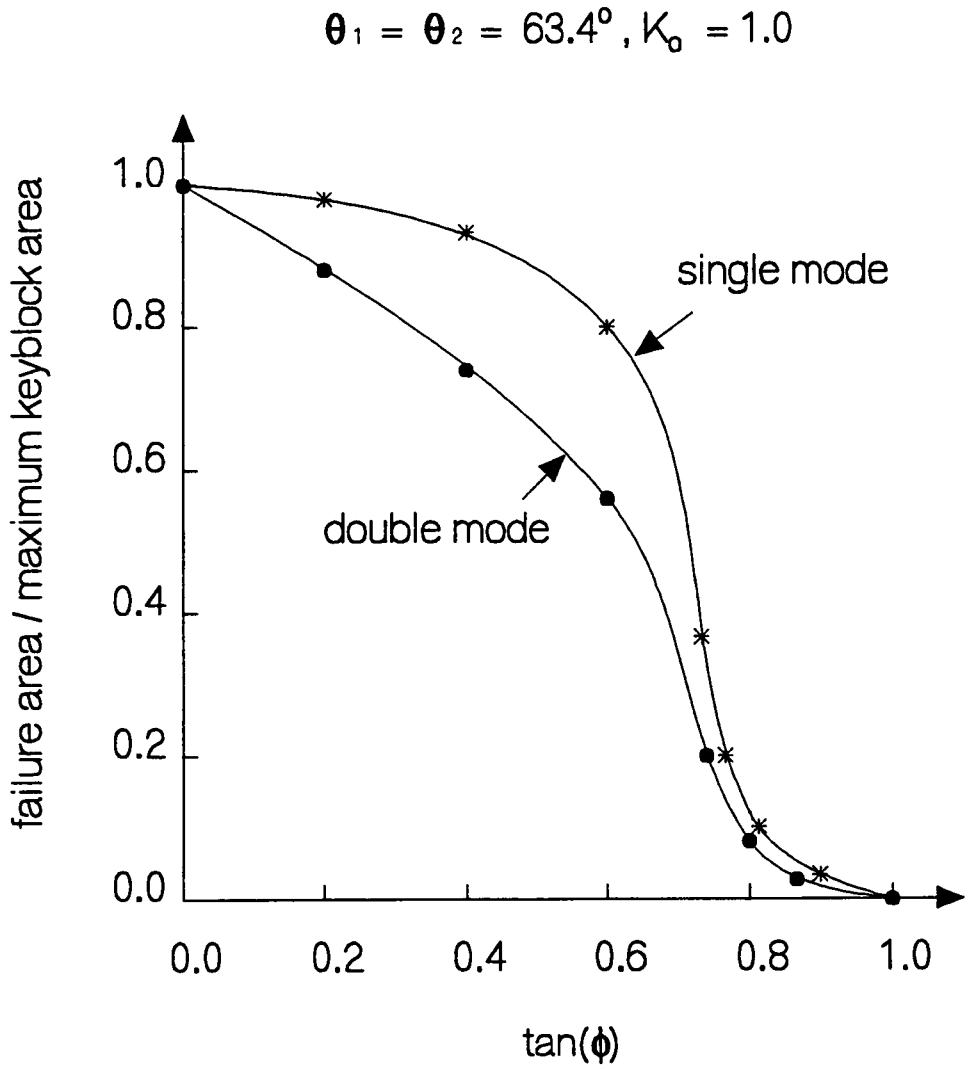

Figure 6-9 Roof unstable area variation as a function of friction coefficient. Constant field stress $\sigma_v = \sigma_h = 1.35$ MPa.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Conclusions

Based on the results, our conclusions are as follows:

(1) There tends to be an upper limit to the size of unstable keyblock due to the stabilizing effect of in situ stresses. The unstable region may be significantly less than the maximum keyblock region predicted by block theory.

(2) The unstable region in the tunnel roof will decrease with the increase of joint dip. The unstable region in the tunnel walls will increase with the increase of bottom plane dip. There are unstable regions in the floor in some cases based on our computation , while floor blocks are nonremovable based on block theory.

(3) Generally, the increase of stress ratio will only help support the blocks with the joint dip greater than 45°.

(4) Linear programming provides a powerful tool for determining the stability of keyblocks, when block self-weight and surface forces are included in the analysis.

(5) The results of our research can be used for the tunnel located in over 50 meters under the ground surface.

(6) For roof and floor blocks, there is no much difference of the results from single and double modes. For wall blocks, it is better to use double mode because the

block will slide along the bottom plane after the top plane fails.

(7) The advantage of this analysis approach over numerical models is that knowledge of joint location is not required, only representative orientations.

Future Research

(1) Block stability analysis is useful in hard, jointed rock masses. In rock engineering when blasting is used to excavate the tunnel, both the excavation and supporting costs are important. Future research should consider the total dollar effectiveness of these two aspects whatever theory will be used.

(2) It is necessary to extend the research to 3D analysis.

(3) Future research should concentrate on how to put the results into practice.

BIBLIOGRAPHY

BIBLIOGRAPHY

Brady, B.H.G. & E.T. Brown 1993. *Rock mechanics for underground mining.* London: Chapman & Hall.

Chuang, Poon-Hwei. 1992. Stability analysis in geomechanics by linear programming (I: Formulation). *Journal of geotechnical engineering*. Vol. 118, No. 11, Nov. 1992, ©ASCE.

Curran, J.H. & B.T. Corkum 1989. *Examine*^{2d} (version 3.0). Toronto: University of Toronto.

Gass, Saul I. 1985. *Linear programming methods and applications.* New York: McGraw-Hill Book Company.

Goodman, R.E. & Gen-hua Shi 1985. *Block theory and its application to rock engineering.* New Jersey: Prentice-Hall.

Goodman, R. E. 1993. *Engineering Geology.* New York: John Wiley & Sons.

Goodman, R. E. 1992. Block theory in rock engineering. *Engineering in rock masses.* Edited by Bell, F.G. Oxford: Butterworth-Heinemann Ltd.

Goodman, R. E. 1989. *Introduction to rock mechanics.* New York: John Wiley & Sons.

Herget, G. 1988. *Stresses in rock.* Brookfield: A. A. Balkema.

Karzulovic, A.L. 1988. The use of keyblock theory in the design of linings and supports for tunnel. *Ph.D. Thesis*. Berkeley: Department of Civil Engineering, University of California.

Mauldon, M. 1995. Keyblock sizes and tunnel rock loads. *Uncertainty evaluation in geo-engineering for the 10th ASCE Engineering Mechanics Div.* Specialty Conference, Boulder, May 1995.

Mauldon & Zhao. 1995. Stability of keyblocks under self-weight and surface forces. *Rock mechanics proceedings of the 35th U.S. symposium.* Rock mechanics, Daemen & Schultz (eds). Balkema, Rotterdam.

Priest, Stephen D. 1993. *Discontinuity analysis for rock engineering.* New York: Chapman & Hall.

Wu, Nesa & Coppins, Richard. 1981. *Linear programming and extensions.* New York: McGraw-Hill, Inc.

APPENDIX

APPENDIX

This appendix is the complete QuickBasic program for roof blocks we coded for computing the optimal values. The same results can be obtained from other programs, such as Maple, Quattro Pro, and Lindo. But we are able to obtain only one optimal value each time from these programs. So, we have to edit the input data and constraints (if necessary) each time to obtain different values, which is time and energy consuming. Therefore, we coded the simplex algorithm and obtained all required optimal values at the same time. For the floor block, we could not haved coded the model. Therefore, we used to Quattro Pro and obtained the floor results. To use these QuickBasic programs, please follow the following comprehensive instructions below.

The linear programming codes are for general use. To edit it for different problems, the only thing we have to do is the initial matrix setup. When setting up your new initial matrix, be sure to delete the old initial matrix in the program printed below. To facilitate the future use of the linear program, the following two examples will show how to set up the initial matrixes and edit the program. Refer to "Linear Programming and Extensions" by Nesa Wu and Richard Coppins (1981) for more examples and detailed explanations.

Example 1:

This example shows the initial matrix setup for P_{max} based on Equations 5.14 and 5.15. The actual program setup can be found from lines 100011 to 116300 in the appendix.

Let's rename the variables and coefficients in Equations 5.14 and 5.15 to use the conventional variables in the linear programming books.

$x_1 = M_1$
$x_2 = M_2$
$x_3 = T_1$
$x_4 = T_2$
y_1 , y_2 , y_3 , y_4 = slack variables
z_1 = artificial variable

Now the objective function change to

$$P = \cos\theta_1 x_1 + \cos\theta_2 x_2 + \sin\theta_1 x_3 + \sin\theta_2 x_4 + 0y_1 + 0y_2 + 0y_3 + 0y_4 - BMz_1 \qquad \text{(i)}$$

where BM is a very big value we assumed in the linear programming because we use Big-M simplex method (see Wu and Coppins, 1981, page 81) for the linear programming (BM=100 in our program, this value should be relatively much larger

than the coefficients in the objective function).

The constraints (Equation 5.15) change (after adding slack and artificial variables) to

$$
\begin{aligned}
& x_1 + y_1 = b_1 \\
& x_2 + y_2 = b_2 \\
& k_1 x_1 + x_3 + y_3 = b_3 \\
& k_2 x_2 + x_4 + y_4 = b_4 \\
& \sin\theta_1 x_1 - \sin\theta_2 x_2 - \cos\theta_1 x_3 + \cos\theta_2 x_4 + z_1 = b_5 \qquad \text{(ii)}
\end{aligned}
$$

where

$$
\begin{aligned}
b_1 &= N_1 \\
b_2 &= N_2 \\
b_3 &= k_1 N_1 - S_1 \\
b_4 &= k_2 N_2 - S_2 \\
b_5 &= 0
\end{aligned}
$$

When we use linear programming, we have to put all the coefficients of objective function and constraints in the same matrix. So we have the following matrix.

No	0	1	2	3	4	5	6	7	8	9	10	11
0	x_b	c_b	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	z_1	b_i
1	0	0	$\cos\theta_1$	$\cos\theta_2$	$\sin\theta_1$	$\sin\theta_2$	0	0	0	0	-BM	0
2	y_1	0	1	0	0	0	1	0	0	0	0	b_1
3	y_2	0	0	1	0	0	0	1	0	0	0	b_2
4	y_3	0	k_1	0	1	0	0	0	1	0	0	b_3
5	y_4	0	0	k_2	0	1	0	0	0	1	0	b_4
6	z_1	-BM	$\sin\theta_1$	$-\sin\theta_2$	$-\cos\theta_1$	$\cos\theta_2$	0	0	0	0	1	b_5

We translate the shaded part of the matrix above into the matrix elements row by row and put them into the "FOR - NEXT" loops. The initial matrix setup has been completed.

Example 2:

The following example shows the initial matrix setup for P_{min} based on Equations 5.16 and 5.17. The actual program setup can be found from lines 400011

to 500011 in the appendix. Refer to Example 1 for renaming the variables and BM value.

The objective function change to

$$P = \cos\theta_1 x_1 + \cos\theta_2 x_2 + \sin\theta_1 x_3 + \sin\theta_2 x_4 + 0y_1 + 0y_2 + 0y_3 + BMz_1 + BMz_2 \quad \text{(iii)}$$

The constraints (Equation 5.17) change (after adding slack and artificial variables) to

$$\begin{aligned}
&x_1 - z_1 = b_1 \\
&x_2 + y_1 = b_2 \\
&k_1 x_1 + x_3 + y_2 = b_3 \\
&k_2 x_2 + x_4 + y_3 = b_4 \\
&\sin\theta_1 x_1 - \sin\theta_2 x_2 - \cos\theta_1 x_3 + \cos\theta_2 x_4 + z_2 = b_5 \quad \text{(iv)}
\end{aligned}$$

where

$$\begin{aligned}
b_1 &= N_1 \\
b_2 &= N_2 \\
b_3 &= k_1 N_1 - S_1 \\
b_4 &= k_2 N_2 - S_2 \\
b_5 &= 0
\end{aligned}$$

The matrix form of Equations iii and iv is:

No	0	1	2	3	4	5	6	7	8	9	10	11
0	x_b	c_b	x_1	x_2	x_3	x_4	y_1	y_2	y_3	z_1	z_2	b_i
1	0	0	$\cos\theta_1$	$\cos\theta_2$	$\sin\theta_1$	$\sin\theta_2$	0	0	0	BM	BM	0
2	z_1	BM	1	0	0	0	0	0	0	-1	0	b_1
3	y_1	0	0	1	0	0	1	0	0	0	0	b_2
4	y_2	0	k_1	0	1	0	0	1	0	0	0	b_3
5	y_3	0	0	k_2	0	1	0	0	1	0	0	b_4
6	z_2	BM	$\sin\theta_1$	$-\sin\theta_2$	$-\cos\theta_1$	$\cos\theta_2$	0	0	0	0	1	b_5

We translate the shaded part of the matrix above into the matrix elements row by row and put them into the "FOR - NEXT" loops. The initial matrix setup has been completed.

To obtain the in situ stresses, we used EXAMINE^{2D} Version 3.0 © 1989 J.H.

Curran and B.T.Corkum. See the following details for how to run EXAMINE^{2D} and what input data were used.

1. MODEL
    SET LAYOUT
        Set limits
        Enter Minimum x,y coordinates:0,0
        Enter Maximum x,y coordinates:100,100
        Return
    BOUNDARIES
        ADD EXCA: We assume the vertical distance from ground surface to the tunnel roof surface is 50 meters. So, we key in the tunnel corner coordinates in this order: 20,20; 30,20; 30,30; 20,30.
        DISCRETIZE
            DEFAULT
            RETURN
        BNDRY COND
            DEFAULT
            RETURN
        RETURN
    STRESS GRD
        ADD GRID: The grid defines the area we will examine. Based on the tunnel coordinates, we key in the grid coordinates as follows: 10,10; 40,40.
            Enter number of intervals in x direction: 30
            Enter number of intervals in y direction: 30
        RETURN
    FLD STRESS
        CONSTANT
            Sigma 1: 1.35 MP_a
            Sigma 3: 1.35 MP_a
            Angle: 0
        RETURN
    Failr Crit
        Hoek-Brown
            m = 1.7
            s = 0.001
            uniaxial comp. strength of intact rock: 100 MP_a
        RETURN
    Elast Prop
        Young's Modulus = 50000 MP_a
        Poisson's Ratio = 0.2
    RETURN

2. COMPUTE

Enter y

Enter output file name - WE WILL EDIT THE FILE IN WORDPERFECT

3. EXIT

Part of the output file is print out as follows:

```
* JOB DESCRIPTION:        EXAMINE 2D  VERSION 3.0   (c)1990
EXAMINE - A BOUNDARY ELEMENT ANALYSIS PACKAGE
* CONTROL PARAMETERS
* Number of Lines    = number of line segments in excavation
*                      and surface geometry
* Stress field       = 1 for constant stress field
*                    = 2 for gravitational stress field
*                    = -n for user defined stress field
*                          where n=number of stress
*                          definition elevations
* Symmetry           = 1 for no symmetry
*                    = 2 for symmetry about line x=xsym
```

.
.
.

```
* |                        GRID POINT RESULTS                                    |
* |------|----------|---------|---------|---------|---------|---------|---------|---------|---------|
* |ELEMENT|__Coordinates__| SIGMA-XX | SIGMA-YY | SIGMA-XY | SIGMA- 1 | SIGMA- 3 | FACTOR | DISPLACE | DISPLACE |
* | No. | X | Y |      |      |      |      | SAFETY | X-DIR | Y-DIR |
* |------|------|------|---------|---------|---------|---------|---------|---------|---------|
```

1	10	10	1.622	1.6194	-0.098392	1.7191	1.5223	10.422	3.614e-05	-0.00010644
2	10	11	1.5917	1.5984	-0.10354	1.6986	1.4914	10.435	3.8244e-05	-0.00010648
3	10	12	1.5608	1.5779	-0.10916	1.6788	1.4598	10.44	4.0514e-05	-0.00010666
4	10	13	1.5289	1.558	-0.11549	1.6598	1.427	10.435	4.2983e-05	-0.00010699
5	10	14	1.4954	1.5389	-0.12272	1.6418	1.3926	10.415	4.5692e-05	-0.00010749
6	10	15	1.4594	1.5212	-0.13091	1.6248	1.3558	10.379	4.8683e-05	-0.00010819
7	10	16	1.4195	1.5056	-0.13981	1.6088	1.3163	10.322	5.1995e-05	-0.00010914
8	10	17	1.3745	1.4928	-0.14859	1.5936	1.2737	10.244	5.5645e-05	-0.0001104
9	10	18	1.3232	1.4837	-0.15568	1.5786	1.2283	10.149	5.961e-05	-0.00011208
10	10	19	1.2655	1.4784	-0.15883	1.5631	1.1808	10.042	6.38e-05	-0.00011427
11	10	20	1.2026	1.4763	-0.15542	1.5465	1.1323	9.9338	6.8043e-05	-0.00011706
12	10	21	1.1372	1.4756	-0.14318	1.528	1.0848	9.8348	7.2088e-05	-0.00012054

.
.
.

957	40	36	0.91305	0.96337	-0.09956	1.0409	0.83552	12.649	-4.0619e-05	-0.00017383
958	40	37	0.89221	0.93035	-0.092113	1.0053	0.81721	12.952	-3.7752e-05	-0.00017464
959	40	38	0.86961	0.89812	-0.085687	0.97073	0.797	13.247	-3.5172e-05	-0.00017531
960	40	39	0.84589	0.8664	-0.080091	0.93689	0.7754	13.539	-3.2839e-05	-0.00017588
961	40	40	0.82149	0.83508	-0.075116	0.9037	0.75286	13.832	-3.0719e-05	-0.00017634

We need only the GRID POINT RESULTS beginning from 1 to 961. So we have to delete other stuff. Now we use WordPerfect to edit the file above.

4. Open the file above and delete the unnecessary information. The data file should contain the following information only:

1	10	10	1.622	1.6194	-0.098392	1.7191	1.5223	10.422	3.614e-05	-0.00010644
2	10	11	1.5917	1.5984	-0.10354	1.6986	1.4914	10.435	3.8244e-05	-0.00010648
3	10	12	1.5608	1.5779	-0.10916	1.6788	1.4598	10.44	4.0514e-05	-0.00010666
4	10	13	1.5289	1.558	-0.11549	1.6598	1.427	10.435	4.2983e-05	-0.00010699
5	10	14	1.4954	1.5389	-0.12272	1.6418	1.3926	10.415	4.5692e-05	-0.00010749
6	10	15	1.4594	1.5212	-0.13091	1.6248	1.3558	10.379	4.8683e-05	-0.00010819
7	10	16	1.4195	1.5056	-0.13981	1.6088	1.3163	10.322	5.1995e-05	-0.00010914
8	10	17	1.3745	1.4928	-0.14859	1.5936	1.2737	10.244	5.5645e-05	-0.0001104
9	10	18	1.3232	1.4837	-0.15568	1.5786	1.2283	10.149	5.961e-05	-0.00011208

.
.
.

937	40	16	1.4195	1.5056	0.13981	1.6088	1.3163	10.322	-5.1995e-05	-0.00010914
938	40	17	1.3745	1.4928	0.14859	1.5936	1.2737	10.244	-5.5646e-05	-0.0001104
939	40	18	1.3232	1.4837	0.15568	1.5786	1.2283	10.149	-5.9612e-05	-0.00011208
940	40	19	1.2655	1.4784	0.15883	1.5631	1.1808	10.042	-6.38e-05	-0.00011427
941	40	20	1.2026	1.4763	0.15542	1.5465	1.1323	9.9338	-6.8042e-05	-0.00011706
942	40	21	1.1372	1.4756	0.14318	1.528	1.0848	9.8348	-7.2086e-05	-0.00012053
943	40	22	1.0737	1.4737	0.12094	1.5074	1.04	9.7566	-7.5626e-05	-0.00012469
944	40	23	1.0166	1.4675	0.089188	1.4845	0.99964	9.7096	-7.8336e-05	-0.00012944
945	40	24	0.97035	1.4543	0.050084	1.4594	0.96523	9.7023	-7.9947e-05	-0.00013466
946	40	25	0.93765	1.4322	0.0071368	1.4323	0.93755	9.7413	-8.0271e-05	-0.00014012
947	40	26	0.91943	1.4007	-0.035474	1.4033	0.91683	9.8307	-7.9257e-05	-0.00014557
948	40	27	0.91445	1.3606	-0.073658	1.3725	0.90261	9.9726	-7.6981e-05	-0.00015077
949	40	28	0.91965	1.3139	-0.10412	1.3397	0.89384	10.166	-7.3645e-05	-0.00015553
950	40	29	0.93072	1.2634	-0.12495	1.3051	0.88901	10.408	-6.9535e-05	-0.0001597
951	40	30	0.94302	1.212	-0.1359	1.2687	0.8863	10.69	-6.4976e-05	-0.00016321
952	40	31	0.95254	1.1622	-0.13827	1.2309	0.88385	11.003	-6.0273e-05	-0.00016608
953	40	32	0.95659	1.1157	-0.13432	1.1923	0.88004	11.334	-5.5678e-05	-0.00016838
954	40	33	0.95409	1.073	-0.12661	1.1535	0.87368	11.673	-5.1355e-05	-0.00017021
955	40	34	0.94525	1.034	-0.11735	1.1151	0.86415	12.009	-4.7394e-05	-0.00017167
956	40	35	0.93114	0.99767	-0.10807	1.0775	0.85133	12.335	-4.3821e-05	-0.00017286
957	40	36	0.91305	0.96337	-0.09956	1.0409	0.83552	12.649	-4.0619e-05	-0.00017383
958	40	37	0.89221	0.93035	-0.092113	1.0053	0.81721	12.952	-3.7752e-05	-0.00017464
959	40	38	0.86961	0.89812	-0.085687	0.97073	0.797	13.247	-3.5172e-05	-0.00017531
960	40	39	0.84589	0.8664	-0.080091	0.93689	0.7754	13.539	-3.2839e-05	-0.00017588
961	40	40	0.82149	0.83508	-0.075116	0.9037	0.75286	13.832	-3.0719e-05	-0.00017634

5. Save the file edited above. THIS FILE WILL BE READ BY THE QuickBasic program.

6. Open the QuickBasic programs called BSR.bas for roof blocks, BSLW.bas for left wall blocks, and BSRW.bas for right wall blocks.

7. Give the output file name for the final results in line 10 in the programs. THIS FILE CAN BE PUT INTO WORDPERFECT.

8. Give the data file name (see step 5) in line 20 for input file.

9. Run the QuickBasic programs. Follow the prompt to input the joint dips and stress ratio. The inputs of planes for roof, walls and floor are shown in Figure A-1. On the screen, Table 1 gives shear force/normal force on plane 1 for points above. Each point represents a block vertex. Table 2 gives shear force/normal force on plane 2. Table 3 gives the results.

10. Exit QuickBasic and go back to WordPerfect to print out the final results (open the file saved in step 7). The output formats for roof, walls and floor are shown in Figure A-2. Three example outputs for roof is shown in Figure A-3 to Figure A-5.

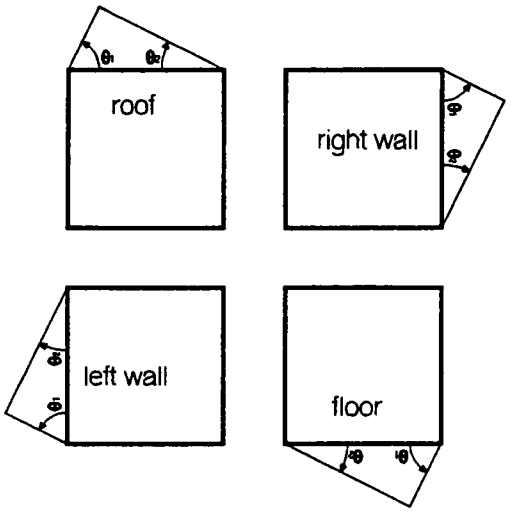

Figure A-1 Input of planes

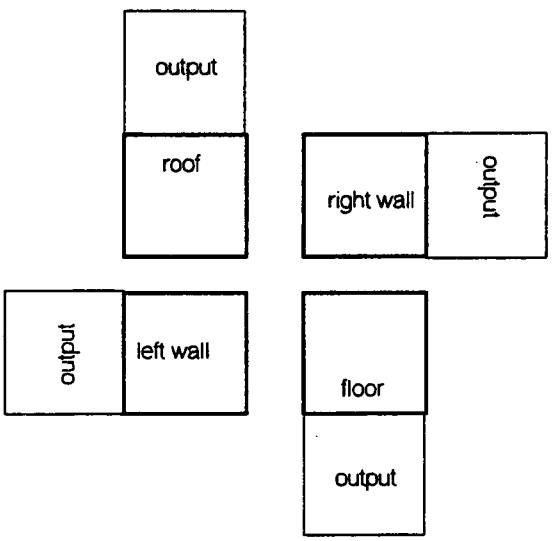

Figure A-2 Format of outputs

Table 3 Results

ROOF
double mode
theta1= 63.43495 theta2= 63.43495
Ko= 1
coefficient of friction on plane 1 = .8
coefficient of friction on plane 2 = .8

0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
1	2	1	1	1	1	1	1	2	1

0 = outside keyblock region
1 = table
2 = potentially unstable
3 = unstable

Figure A-3 Example output for roof block.

Table 3 Results

LEFT WALL
double mode
theta1 = 26.56505 theta2 = 26.56505
Ko = 1
coefficient of friction on plane 1 = .8
coefficient of friction on plane 2 = .8

```
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 3 3
0 0 0 0 0 0 0 0 3 3
0 0 0 0 0 0 0 3 3 3
0 0 0 0 0 0 0 3 3 3
0 0 0 0 0 0 0 0 3 3
0 0 0 0 0 0 0 0 3 3
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 1
```

0 = outside keyblock region
1 = table
2 = potentially unstable
3 = unstable

Figure A-4 Example output for left wall block.

Table 3 Results

RIGHT WALL
double mode
theta1= 26.56505 theta2= 26.56505
Ko= 1
coefficient of friction on plane 1 = .8
coefficient of friction on plane 2 = .8

1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0
3	3	3	0	0	0	0	0	0	0
3	3	3	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

0 = outside keyblock region
1 = table
2 = potentially unstable
3 = unstable

Figure A-5 Example output for right wall block.

The program BSR.bas is printed out below for general reference. Ask Dr. Mauldon for all the three programs.

The parameters in the following program are defined as following.

$k_1 = \tan\phi_1$, ϕ_1 = the friction angle of joint 1.
$k_2 = \tan\phi_2$, ϕ_2 = the friction angle of joint 2.
The definitions of θ_1 and θ_2 for roof blocks refer to Figure 4-1, for wall blocks refer to Figure 4-2, and for floor blocks refer to Figure 4-3.
LM = the low value of m.
We divided the 2D tunnel host rock into 900 cells (30 by 30) using m and n coordinates (see Figure A-6) . The original point is the bottom left corner. We defined LM, HM, LN, and HN as global variables to make the program editing easier.
HM = the high value of m.
LN = the low value of n.
HN = the high value of n.
BM = 100.
We used the big-M method to code the simplex algorithm. BM should be much greater than the largest coefficient of the decision variables in the objective function. Here we assumed BM=100, which was large enough because the largest coefficient of the decision variables in our objective function was less than 1. Smaller BM will lead to inexact solutions.
K_o = stress ratio = horizontal stress / vertical stress.

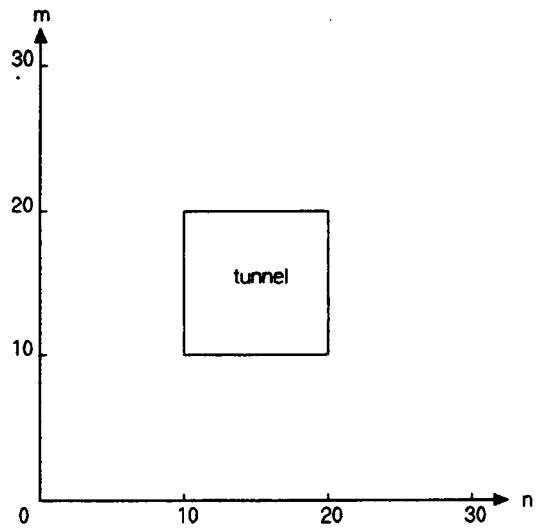

Figure A-6 Coordinates of m and n

```
REM     *** TITLE: BSR.bas ***
REM     *** QuickBasic code for roof block stability analysis ***
CLS
PRINT "QuickBasic code for block stability analysis"
PRINT "The Institute for Geotechnology"
PRINT "Department of Civil and Environmental Engineering"
PRINT "The University of Tennessee at Knoxville"
PRINT "May 1995"
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT "Press Enter to continue"
INPUT pause$
CLS

REM     *** main ***

CLS
REM     *** parameters ***
pi = 3.1415926#
cr = pi / 180
REM parameters for frictin angles
k1 = .8: k2 = .8
REM for loop range(m,n)
LM = 22: HM = 31: LN = 11: HN = 20
REM big-M value
BM = 100

REM  *** dimensions ***
REM  *** 1D ***
DIM x0(1000), y0(1000), sx0(1000), sy0(1000), t0(1000)
DIM s10(1000), s30(1000), FS0(1000), dx0(1000), dy0(1000)
REM  *** 2D ***
DIM sx(31, 31), sy(31, 31), t(31, 31)
DIM sigmaN1(32, 32), sigmaN2(32, 32), tauN1(32, 32), tauN2(32, 32)
DIM N1(32, 32), N2(32, 32), S1(32, 32), S2(32, 32)
DIM mat1(15, 15), mat2(15, 15), mat3(15, 15)
DIM block1(31, 31), block2(31, 31), W(31, 31)
DIM Pmax(31, 31), Pmin1(31, 31), Pmin2(31, 31), P1(31, 31)
DIM Pmin3(31, 31), Pmin4(31, 31), Pmin(31, 31), P2(31, 31)
DIM Pmin5(31, 31), Pmin6(31, 31), Pmin7(31, 31), Pmin8(31, 31)
DIM R1(32, 32), R2(32, 32)

REM     *** open output files for saving results ***
10 OPEN "d:\mz\out\mmtestr.out" FOR OUTPUT AS #2
20 OPEN "d:\mz\examine\mmtest.dat" FOR INPUT AS #1

REM compute N1,N2,S1,S2
110 GOSUB 500
```

```
REM print out R1(m,n)=S1(m,n)/N1(m,n)
CLS
GOSUB 170000000
INPUT pause&
CLS

REM print out R2(m,n)=S2(m,n)/N2(m,n)
CLS
GOSUB 171000000
INPUT pause&
CLS

REM compute Pmax
120 GOSUB 100011
130 GOSUB 120011

REM compute Pmin1
135 GOSUB 400011
140 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin1(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin2
145 GOSUB 500011
150 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin2(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin3
155 GOSUB 600011
160 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin3(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin4
165 GOSUB 700011
170 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin4(m, n) = Pmin(m, n)
```

```
NEXT n
NEXT m

REM compute Pmin5 failure by tension on both joints
175 GOSUB 800011
180 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin5(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin6 failure by shear on #1 and tension on #2
185 GOSUB 900011
190 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin6(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin7 failure by shear on #2 and tension on #1
195 GOSUB 1000011
200 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin7(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM compute Pmin8 failure by shear on both #2 and #1
205 GOSUB 1100011
210 GOSUB 200011
' copy values from LP
FOR m = LM TO HM
FOR n = LN TO HN
Pmin8(m, n) = Pmin(m, n)
NEXT n
NEXT m

REM sort for P1(m,n) -- single failure mode
220 GOSUB 100000000

REM sort for P2(m,n) -- double failure mode
230 GOSUB 101000000


REM compute block weight W(m,n)
320 GOSUB 130000000
```

```
REM indentify block stability(1) -- single mode
330 GOSUB 140000000

REM indentify block stability(2) -- double mode
335 GOSUB 141000000


REM print out P1(m,n)
CLS
'350 GOSUB 160000000
'INPUT pause&
'CLS

REM print out P2(m,n)
CLS
'355 GOSUB 161000000
'INPUT pause&
'CLS

REM print out Pmax(m,n)
CLS
'360 GOSUB 162000000
'INPUT pause&
'CLS

REM print out W(m,n)
CLS
'365 GOSUB 163000000
'INPUT pause&
'CLS

REM print out block(m,n) table(1) -- single mode
CLS
'340 GOSUB 150000000
'INPUT pause&
'CLS

REM print out block(m,n) table(2) -- double mode
CLS
345 GOSUB 151000000
INPUT pause&
CLS


400 END


REM     *** subroutines ***

500 REM *** compute N1,N2,S1,S2 ***
REM *** OPEN DATUM FILES FROM EXAMINE2D ***
REM     *** READ DATA FROM INPUT FILE ***
```

```
FOR j = 1 TO 961
INPUT #1, num0, x0(j), y0(j), sx0(j), sy0(j), t0(j), s10(j), s30(j), FS0, dx0, dy0
NEXT j


REM     *** input theta1 & theta2 ***
PRINT "Enter angle1 (0.5, 1, 2, 3 or 90 ONLY)=:"
INPUT A1
PRINT "Enter angle2 (0.5, 1, 2, 3 or 90 ONLY)=:"
INPUT A2
PRINT "Enter lateral stress rartio Ko"
INPUT Ko

590 GOSUB 9000  'change 1D into 2D & calculate the stresses on both joints


600 REM *** subroutine conditions ***
610 IF A1 = .5 THEN GOSUB 10000
620 IF A1 = 1 THEN GOSUB 20000
630 IF A1 = 2 THEN GOSUB 30000
640 IF A1 = 3 THEN GOSUB 40000
645 IF A1 = 90 THEN GOSUB 45000
650 IF A2 = .5 THEN GOSUB 50000
660 IF A2 = 1 THEN GOSUB 60000
670 IF A2 = 2 THEN GOSUB 70000
680 IF A2 = 3 THEN GOSUB 80000
685 IF A2 = 90 THEN GOSUB 85000

800  GOTO 100011

9000 REM *** change 1D into 2D and calculate stresses on both faces ***
9002 IF A1 = 90 THEN theta1 = pi / 2 ELSE theta1 = ATN(A1)
9004 IF A2 = 90 THEN theta2 = pi / 2 ELSE theta2 = ATN(A2)
9010 j1 = theta1 + pi / 2: j2 = -theta2 + pi / 2
9020 FOR m = 1 TO 31
9030   FOR n = 1 TO 31
9040          i = 31 * (n - 1) + m
9050          sx(m, n) = sx0(i) * 1000
9060          sy(m, n) = sy0(i) * 1000
9070          t(m, n) = t0(i) * 1000
9090          sigmaN1(m, n) = (sx(m, n) * (COS(j1)) ^ 2 + sy(m, n) * (SIN(j1)) ^ 2 + t(m, n) *
SIN(2 * j1))
9100          tauN1(m, n) = (-.5 * SIN(2 * j1) * sx(m, n) + .5 * SIN(2 * j1) * sy(m, n) +
COS(2 * j1) * t(m, n))
9110          sigmaN2(m, n) = (sx(m, n) * (COS(j2)) ^ 2 + sy(m, n) * (SIN(j2)) ^ 2 + t(m, n) *
SIN(2 * j2))
9120          tauN2(m, n) = -1 * (-.5 * SIN(2 * j2) * sx(m, n) + .5 * SIN(2 * j2) * sy(m, n) +
COS(2 * j2) * t(m, n))
9130   NEXT n
9140 NEXT m

9300 RETURN
```

```
9900  REM *** The angle subroutines go here ***

9910  REM *** joint #1 goes here ***

10000 REM *** theta1=arctan(1/2) ***
10010 FOR m = 22 TO 31
11027   j = n - 10
11030   FOR n = 1 TO 31
11040      N1(m, n) = (sigmaN1(m, n) + sigmaN1(m, n + 1)) / (2 * COS(theta1))
11050      S1(m, n) = (tauN1(m, n) + tauN1(m, n + 1)) / (2 * COS(theta1))
11062         IF j MOD 2 = 0 THEN N1(m, n) = N1(m, n) + N1(m - 1, n - 1)
11064         IF j MOD 2 <> 0 THEN N1(m, n) = N1(m, n) + N1(m, n - 1)
11072         IF j MOD 2 = 0 THEN S1(m, n) = S1(m, n) + S1(m - 1, n - 1)
11074         IF j MOD 2 <> 0 THEN S1(m, n) = S1(m, n) + S1(m, n - 1)
11080   REM PRINT #2, "N1(m,n), S1(m,n)"
11000   NEXT n
12000 NEXT m

13000 RETURN


20000 REM *** theta1=arctan(1) ***
21020 FOR m = 22 TO 31
21030   FOR n = 1 TO 31
21060         N1(m, n) = (sigmaN1(m, n) + sigmaN1(m, n + 1)) / (2 * SIN(theta1)) + N1(m -
1, n - 1)
21070         S1(m, n) = (tauN1(m, n) + tauN1(m, n + 1)) / (2 * SIN(theta1)) + S1(m - 1, n -
1)
21100   NEXT n
22000 NEXT m

23000 RETURN


30000 REM *** theta1=arctan(2) ***
31020 FOR m = 22 TO 31
31025   i = m - 21
31030   FOR n = 1 TO 31
31060         N1(m, n) = (sigmaN1(m, n) + sigmaN1(m, n + 1)) / (2 * SIN(theta1))
31062         IF i MOD 2 = 0 THEN N1(m, n) = N1(m, n) + N1(m - 1, n)
31064         IF i MOD 2 <> 0 THEN N1(m, n) = N1(m, n) + N1(m - 1, n - 1)
31070         S1(m, n) = (tauN1(m, n) + tauN1(m, n + 1)) / (2 * SIN(theta1))
31072         IF i MOD 2 = 0 THEN S1(m, n) = S1(m, n) + S1(m - 1, n)
31074         IF i MOD 2 <> 0 THEN S1(m, n) = S1(m, n) + S1(m - 1, n - 1)
31100   NEXT n
32000 NEXT m

33000 RETURN

40000 REM *** theta1=arctan(3) ***
41020 FOR m = 22 TO 31
41025   i = m - 21
41030   FOR n = 1 TO 31
```

```
41060         N1(m, n) = (sigmaN1(m, n) + sigmaN1(m, n + 1)) / (2 * SIN(theta1))
41062         IF i MOD 3 = 0 THEN N1(m, n) = N1(m, n) + N1(m - 1, n)
41064         IF i MOD 3 = 1 THEN N1(m, n) = N1(m, n) + N1(m - 1, n - 1)
41066         IF i MOD 3 = 2 THEN N1(m, n) = N1(m, n) + N1(m - 1, n)
41070         S1(m, n) = (tauN1(m, n) + tauN1(m, n + 1)) / (2 * SIN(theta1))
41072         IF i MOD 3 = 0 THEN S1(m, n) = S1(m, n) + S1(m - 1, n)
41074         IF i MOD 3 = 1 THEN S1(m, n) = S1(m, n) + S1(m - 1, n - 1)
41076         IF i MOD 3 = 2 THEN S1(m, n) = S1(m, n) + S1(m - 1, n)
41100   NEXT n
42000 NEXT m

43000 RETURN

45000 REM *** theta1 = vertical ***
46020 FOR m = 22 TO 31
46030   FOR n = 1 TO 31
46060         N1(m, n) = (sigmaN1(m, n) + sigmaN1(m, n + 1)) / (2 * SIN(theta1))
46066         N1(m, n) = N1(m, n) + N1(m - 1, n)
46070         S1(m, n) = (tauN1(m, n) + tauN1(m, n + 1)) / (2 * SIN(theta1))
46076         S1(m, n) = S1(m, n) + S1(m - 1, n)
46100   NEXT n
47000 NEXT m

48000 RETURN


49000 REM *** joint #2 goes here ***

50000 REM *** theta2=arctan(1/2) ***
50010 FOR m = 22 TO 31
51027   j = n - 10
51030   FOR n = 1 TO 31
51040      N2(m, n) = (sigmaN2(m, n) + sigmaN2(m, n + 1)) / (2 * COS(theta2))
51050      S2(m, n) = (tauN2(m, n) + tauN2(m, n + 1)) / (2 * COS(theta2))
51062         IF j MOD 2 = 0 THEN N2(m, n) = N2(m, n) + N2(m - 1, n + 1)
51064         IF j MOD 2 <> 0 THEN N2(m, n) = N2(m, n) + N2(m, n + 1)
51072         IF j MOD 2 = 0 THEN S2(m, n) = S2(m, n) + S2(m - 1, n + 1)
51074         IF j MOD 2 <> 0 THEN S2(m, n) = S2(m, n) + S2(m, n + 1)
51000   NEXT n
52000 NEXT m

53000 RETURN


60000 REM *** theta2=arctan(1) ***
61020 FOR m = 22 TO 31
61030   FOR n = 1 TO 31
61060         N2(m, n) = (sigmaN2(m, n) + sigmaN2(m, n + 1)) / (2 * SIN(theta2)) + N2(m -
1, n + 1)
61070         S2(m, n) = (tauN2(m, n) + tauN2(m, n + 1)) / (2 * SIN(theta2)) + S2(m - 1, n
+ 1)
61100   NEXT n
62000 NEXT m
```

```
63000 RETURN

70000 REM *** theta2=arctan(2) ***
71020 FOR m = 22 TO 31
71025  i = m - 21
71030  FOR n = 1 TO 31
71060        N2(m, n) = (sigmaN2(m, n) + sigmaN2(m, n + 1)) / (2 * SIN(theta2))
71062        IF i MOD 2 = 0 THEN N2(m, n) = N2(m, n) + N2(m - 1, n)
71064        IF i MOD 2 <> 0 THEN N2(m, n) = N2(m, n) + N2(m - 1, n + 1)
71070        S2(m, n) = (tauN2(m, n) + tauN2(m, n + 1)) / (2 * SIN(theta2))
71072        IF i MOD 2 = 0 THEN S2(m, n) = S2(m, n) + S2(m - 1, n)
71074        IF i MOD 2 <> 0 THEN S2(m, n) = S2(m, n) + S2(m - 1, n + 1)
71100  NEXT n
72000 NEXT m

73000 RETURN

80000 REM *** theta2=arctan(3) ***

81020 FOR m = 22 TO 31
81025  i = m - 21
81030  FOR n = 1 TO 31
81060        N2(m, n) = (sigmaN2(m, n) + sigmaN2(m, n + 1)) / (2 * SIN(theta2))
81062        IF i MOD 3 = 0 THEN N2(m, n) = N2(m, n) + N2(m - 1, n)
81064        IF i MOD 3 = 1 THEN N2(m, n) = N2(m, n) + N2(m - 1, n + 1)
81066        IF i MOD 3 = 2 THEN N2(m, n) = N2(m, n) + N2(m - 1, n)
81070        S2(m, n) = (tauN2(m, n) + tauN2(m, n + 1)) / (2 * SIN(theta2))
81072        IF i MOD 3 = 0 THEN S2(m, n) = S2(m, n) + S2(m - 1, n)
81074        IF i MOD 3 = 1 THEN S2(m, n) = S2(m, n) + S2(m - 1, n + 1)
81076        IF i MOD 3 = 2 THEN S2(m, n) = S2(m, n) + S2(m - 1, n)
81100  NEXT n
82000 NEXT m

83000 RETURN


85000 REM *** theta2 = vertical ***
85020 FOR m = 22 TO 31
85030  FOR n = 1 TO 31
85060        N2(m, n) = (sigmaN2(m, n) + sigmaN2(m, n + 1)) / (2 * SIN(theta2))
85066        N2(m, n) = N2(m, n) + N2(m - 1, n)
85070        S2(m, n) = (tauN2(m, n) + tauN2(m, n + 1)) / (2 * SIN(theta2))
85076        S2(m, n) = S2(m, n) + S2(m - 1, n)
85100  NEXT n
85110 NEXT m

86000 RETURN

100011 REM  *** Pmax section ***

110000 REM matrix(1) initialization
111100 c0 = 0
111110 c1 = COS(theta1)
```

111120 c2 = COS(theta2)
111130 c3 = SIN(theta1)
111140 c4 = SIN(theta2)
111141 c5 = 0
111142 c6 = 0
111143 c7 = 0
111144 c8 = 0
111145 c9 = -BM

111240 a10 = 0
112100 a11 = 1
112110 a12 = 0
112120 a13 = 0
112130 a14 = 0
112140 a15 = 1
112150 a16 = 0
112160 a17 = 0
112170 a18 = 0
112180 a19 = 0

113240 a20 = 0
113100 a21 = 0
113110 a22 = 1
113120 a23 = 0
113130 a24 = 0
113140 a25 = 0
113150 a26 = 1
113160 a27 = 0
113170 a28 = 0
113180 a29 = 0

115000 a30 = 0
115110 a31 = k1
115120 a32 = 0
115130 a33 = 1
115140 a34 = 0
115150 a35 = 0
115160 a36 = 0
115170 a37 = 1
115180 a38 = 0
115190 a39 = 0

114100 a40 = 0
114110 a41 = 0
114120 a42 = k2
114130 a43 = 0
114140 a44 = 1
114150 a45 = 0
114160 a46 = 0
114170 a47 = 0
114180 a48 = 1
114190 a49 = 0

```
116100 a50 = -BM
116110 a51 = SIN(theta1)
116120 a52 = -SIN(theta2)
116130 a53 = -COS(theta1)
116140 a54 = COS(theta2)
116150 a55 = 0
116160 a56 = 0
116170 a57 = 0
116180 a58 = 0
116190 a59 = 1

RETURN


REM  *** LP  section ***

120011 RN = 8: CN = 11

104010 FOR m = LM TO HM
104020 FOR n = LN TO HN

111148 mat1(1, 1) = c0
111149 mat1(1, 2) = c1
111150 mat1(1, 3) = c2
111160 mat1(1, 4) = c3
111170 mat1(1, 5) = c4
111180 mat1(1, 6) = c5
111190 mat1(1, 7) = c6
111200 mat1(1, 8) = c7
111210 mat1(1, 9) = c8
111220 mat1(1, 10) = c9
111230 mat1(1, 11) = 0

112200 mat1(2, 1) = a10
112210 mat1(2, 2) = a11
112220 mat1(2, 3) = a12
112330 mat1(2, 4) = a13
112440 mat1(2, 5) = a14
112450 mat1(2, 6) = a15
112460 mat1(2, 7) = a16
112470 mat1(2, 8) = a17
112480 mat1(2, 9) = a18
112490 mat1(2, 10) = a19
112500 mat1(2, 11) = N1(m, n)

113500 mat1(3, 1) = a20
113510 mat1(3, 2) = a21
113520 mat1(3, 3) = a22
113530 mat1(3, 4) = a23
113540 mat1(3, 5) = a24
113550 mat1(3, 6) = a25
113560 mat1(3, 7) = a26
113570 mat1(3, 8) = a27
```

```
113580 mat1(3, 9) = a28
113590 mat1(3, 10) = a29
113600 mat1(3, 11) = N2(m, n)

115210 mat1(4, 1) = a30
115215 mat1(4, 2) = a31
115220 mat1(4, 3) = a32
115230 mat1(4, 4) = a33
115240 mat1(4, 5) = a34
115250 mat1(4, 6) = a35
115260 mat1(4, 7) = a36
115270 mat1(4, 8) = a37
115280 mat1(4, 9) = a38
115290 mat1(4, 10) = a39
115300 mat1(4, 11) = k1 * N1(m, n) - S1(m, n)

114210 mat1(5, 1) = a40
114215 mat1(5, 2) = a41
114220 mat1(5, 3) = a42
114230 mat1(5, 4) = a43
114240 mat1(5, 5) = a44
114250 mat1(5, 6) = a45
114260 mat1(5, 7) = a46
114270 mat1(5, 8) = a47
114280 mat1(5, 9) = a48
114290 mat1(5, 10) = a49
114300 mat1(5, 11) = k2 * N2(m, n) - S2(m, n)

116210 mat1(6, 1) = a50
116215 mat1(6, 2) = a51
116220 mat1(6, 3) = a52
116230 mat1(6, 4) = a53
116240 mat1(6, 5) = a54
116250 mat1(6, 6) = a55
116260 mat1(6, 7) = a56
116270 mat1(6, 8) = a57
116280 mat1(6, 9) = a58
116290 mat1(6, 10) = a59
116300 mat1(6, 11) = 0

117000 REM *** Zj row ****
117010 mat1(RN - 1, 1) = 0
117020 FOR x = 2 TO CN
117030 mat1(RN - 1, x) = 0     'initialize mat1(RN-1,x)
117040 FOR y = 2 TO RN - 2
117050 mat1(RN - 1, x) = mat1(RN - 1, x) + mat1(y, 1) * mat1(y, x)
117060 NEXT y
117070 NEXT x

118100 REM Cj-Zj row
118110 mat1(RN, 1) = 0
118120 mat1(RN, CN) = 0
118130 FOR j = 2 TO CN - 1
```

```
118140 mat1(RN, j) = mat1(1, j) - mat1(RN - 1, j)
118150 NEXT j


119900 REM check B(i) see if any negative
119910 FOR i = 2 TO RN - 2
119920 IF mat1(i, CN) < 0 THEN GOTO 191000
119930 NEXT i


120000 REM *** check last row (RN) in mat1 see if any positive
120020 FOR j = 2 TO CN - 1
120030 IF mat1(RN, j) > 0 THEN GOTO 121000
120040 NEXT j
120050 GOTO 191000

121000 REM find the most positive element in last row
121010 largest1 = mat1(RN, 2): cindex1 = 2
121020 FOR j = 3 TO (CN - 1)
121030 IF mat1(RN, j) > largest1 THEN largest1 = mat1(RN, j): cindex1 = j
121040 NEXT j

122000 REM check mat1(i,cindex1) to find  pivot row
122010 FOR i = 2 TO RN - 2
122020 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 THEN ratio1(i) = mat1(i, CN) /
mat1(i, cindex1): smallest1 = ratio1(i): rindex1 = i: GOTO 123000
122030 NEXT i
122040 GOTO 191000


123000 FOR i = rindex1 + 1 TO RN - 2
123010 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 THEN ratio1(i) = mat1(i, CN) /
mat1(i, cindex1)
123020 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 AND ratio1(i) < smallest1 THEN
smallest1 = ratio1(i): rindex1 = i
123030 NEXT i

123040  pivotelement1 = mat1(rindex1, cindex1)

124000 REM *** compute all elements for mat2 ***
124010 REM elements NOT in pivotrow & pivotcolumn in mat2
124020 FOR i = 2 TO RN - 2
124030 FOR j = 2 TO CN
124040 IF i <> rindex1 OR j <> cindex1 THEN mat2(i, j) = mat1(i, j) - mat1(rindex1, j) *
mat1(i, cindex1) / pivotelement1
124050 NEXT j
124060 NEXT i

124100 REM elements in pivotrow for mat2
124110 FOR j = 2 TO CN
124120 IF j <> cindex1 THEN mat2(rindex1, j) = mat1(rindex1, j) / pivotelement1
124130 NEXT j
```

```
124200 REM elements in pivotcolumn for mat2
124210 FOR i = 2 TO RN - 2
124220 IF i < > rindex1 THEN mat2(i, cindex1) = 0
124230 NEXT i

124300 REM change the element value in mat2 at mat1 pivotelement position
124310 mat2(rindex1, cindex1) = 1

124400 REM the remaining elements for mat2

124410 'elements in column 1 of mat2
124420 FOR i = 1 TO RN
124430 IF i < > rindex1 THEN mat2(i, 1) = mat1(i, 1)
124440 IF i = rindex1 THEN mat2(rindex1, 1) = mat1(1, cindex1)
124450 NEXT i

124460 'elements in row 1 of mat2
124470 FOR j = 2 TO CN
124480 mat2(1, j) = mat1(1, j)
124490 NEXT j

127000 REM **** Zj row ****
127010 mat2(RN - 1, 1) = 0
127020 FOR x = 2 TO CN
127030 mat2(RN - 1, x) = 0    'initialize mat2(RN-1,x)
127040 FOR y = 2 TO RN - 2
127050 mat2(RN - 1, x) = mat2(RN - 1, x) + mat2(y, 1) * mat2(y, x)
127060 NEXT y
127070 NEXT x

128100 REM Cj-Zj row
128110 mat2(RN, 1) = 0
128120 mat2(RN, CN) = 0
128130 FOR j = 2 TO CN - 1
128140 mat2(RN, j) = mat2(1, j) - mat2(RN - 1, j)
128150 NEXT j


130000 REM **** check last row (RN) in mat2 see if any positive
130010 FOR j = 2 TO CN - 1
130020 IF mat2(RN, j) > 0 THEN GOTO 131000
130030 NEXT j
130040 GOTO 192000

131000 REM find the most positive element in last row
131010 largest2 = mat2(RN, 2): cindex2 = 2
131020 FOR j = 3 TO CN - 1
131030 IF mat2(RN, j) > largest2 THEN largest2 = mat2(RN, j): cindex2 = j
131040 NEXT j

132000 REM check mat2(i,cindex2) to find  pivot row
132010 FOR i = 2 TO RN - 2
132020 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 THEN ratio2(i) = mat2(i, CN) /
```

```
mat2(i, cindex2): smallest2 = ratio2(i): rindex2 = i: GOTO 133000
132030 NEXT i
132040 GOTO 191000


133000 FOR i = rindex2 + 1 TO RN - 2
133010 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 THEN ratio2(i) = mat2(i, CN) /
mat2(i, cindex2)
133020 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 AND ratio2(i) < smallest2 THEN
smallest2 = ratio2(i): rindex2 = i
133030 NEXT i

133040  pivotelement2 = mat2(rindex2, cindex2)

134000 REM *** compute all elements for mat3 ***
134010 REM elements NOT in pivotrow & pivotcolumn in mat3
134020 FOR i = 2 TO RN - 2
134030 FOR j = 2 TO CN
134040 IF i <> rindex2 OR j <> cindex2 THEN mat3(i, j) = mat2(i, j) - mat2(rindex2, j) *
mat2(i, cindex2) / pivotelement2
134050 NEXT j
134060 NEXT i

134100 REM elements in pivotrow for mat3
134110 FOR j = 2 TO CN
134120 IF j <> cindex1 THEN mat3(rindex2, j) = mat2(rindex2, j) / pivotelement2
134130 NEXT j

134200 REM elements in pivotcolumn for mat3
134210 FOR i = 2 TO RN - 2
134220 IF i <> rindex2 THEN mat3(i, cindex2) = 0
134230 NEXT i

134300 REM change the element value in mat3 at mat2 pivotelement position
134310 mat3(rindex2, cindex2) = 1

134400 REM the remaining elements for mat3

134410 'elements in column 1 of mat3
134420 FOR i = 1 TO RN
134430 IF i <> rindex2 THEN mat3(i, 1) = mat2(i, 1)
134440 IF i = rindex2 THEN mat3(rindex2, 1) = mat2(1, cindex2)
134450 NEXT i

134460 'elements in row 1 of mat3
134470 FOR j = 2 TO CN
134480 mat3(1, j) = mat1(1, j)
134490 NEXT j

137000 REM *** Zj row ****
137010 mat3(RN - 1, 1) = 0
137020 FOR x = 2 TO CN
137030 mat3(RN - 1, x) = 0    'initialize mat3(RN-1,x)
```

```
137040 FOR y = 2 TO RN - 2
137050 mat3(RN - 1, x) = mat3(RN - 1, x) + mat3(y, 1) * mat3(y, x)
137060 NEXT y
137070 NEXT x

138100 REM Cj-Zj row
138110 mat3(RN, 1) = 0
138120 mat3(RN, CN) = 0
138130 FOR j = 2 TO CN - 1
138140 mat3(RN, j) = mat3(1, j) - mat3(RN - 1, j)
138150 NEXT j


140000 REM check mat3 see if optimal

140010 REM *** check last row in mat3 see if any positive
140020 FOR j = 2 TO CN - 1
140030 IF mat3(RN, j) > 0 THEN GOTO 150000
140040 NEXT j
140050 GOTO 193000

150000 REM change mat3 to mat2 to employ mat2 computation procedures

150010 FOR i = 1 TO RN
150020 FOR j = 1 TO CN
150030 mat2(i, j) = mat3(i, j)
150040 NEXT j
150050 NEXT i
150060 GOTO 131000

190000 REM record Pmax(m,n)
191000 Pmax(m, n) = -1: GOTO 200000

192000 FOR i = 2 TO RN - 2
192010 IF mat2(i, 1) = -BM THEN GOTO 192110
192020 NEXT i
192100 Pmax(m, n) = mat2(RN - 1, CN): GOTO 200000
192110 Pmax(m, n) = -1: GOTO 200000

193000 FOR i = 2 TO RN - 2
193010 IF mat3(i, 1) = -BM THEN GOTO 193110
193020 NEXT i
193100 Pmax(m, n) = mat3(RN - 1, CN): GOTO 200000
193110 Pmax(m, n) = -1: GOTO 200000

200000 NEXT n
200010 NEXT m

RETURN


200011 REM  *** Pmin(m,n)  section ***
```

```
204010 FOR m = LM TO HM
204020 FOR n = LN TO HN

210000 REM matrix(1) initialization

211148 mat1(1, 1) = c0
211149 mat1(1, 2) = c1
211150 mat1(1, 3) = c2
211160 mat1(1, 4) = c3
211170 mat1(1, 5) = c4
211180 mat1(1, 6) = c5
211190 mat1(1, 7) = c6
211200 mat1(1, 8) = c7
211210 mat1(1, 9) = c8
211220 mat1(1, 10) = c9
211230 mat1(1, 11) = 0

213500 mat1(2, 1) = a10
213510 mat1(2, 2) = a11
213520 mat1(2, 3) = a12
213530 mat1(2, 4) = a13
213540 mat1(2, 5) = a14
213550 mat1(2, 6) = a15
213560 mat1(2, 7) = a16
213570 mat1(2, 8) = a17
213580 mat1(2, 9) = a18
213590 mat1(2, 10) = a19
213600 mat1(2, 11) = N1(m, n)

215210 mat1(3, 1) = a20
215215 mat1(3, 2) = a21
215220 mat1(3, 3) = a22
215230 mat1(3, 4) = a23
215240 mat1(3, 5) = a24
215250 mat1(3, 6) = a25
215260 mat1(3, 7) = a26
215270 mat1(3, 8) = a27
215280 mat1(3, 9) = a28
215290 mat1(3, 10) = a29
215300 mat1(3, 11) = N2(m, n)

214210 mat1(4, 1) = a30
214215 mat1(4, 2) = a31
214220 mat1(4, 3) = a32
214230 mat1(4, 4) = a33
214240 mat1(4, 5) = a34
214250 mat1(4, 6) = a35
214260 mat1(4, 7) = a36
214270 mat1(4, 8) = a37
214280 mat1(4, 9) = a38
214290 mat1(4, 10) = a39
214300 mat1(4, 11) = k1 * N1(m, n) - S1(m, n)
```

```
212200 mat1(5, 1) = a40
212210 mat1(5, 2) = a41
212220 mat1(5, 3) = a42
212330 mat1(5, 4) = a43
212440 mat1(5, 5) = a44
212450 mat1(5, 6) = a45
212460 mat1(5, 7) = a46
212470 mat1(5, 8) = a47
212480 mat1(5, 9) = a48
212490 mat1(5, 10) = a49
212500 mat1(5, 11) = k2 * N2(m, n) - S2(m, n)

216210 mat1(6, 1) = a50
216215 mat1(6, 2) = a51
216220 mat1(6, 3) = a52
216230 mat1(6, 4) = a53
216240 mat1(6, 5) = a54
216250 mat1(6, 6) = a55
216260 mat1(6, 7) = a56
216270 mat1(6, 8) = a57
216280 mat1(6, 9) = a58
216290 mat1(6, 10) = a59
216300 mat1(6, 11) = 0

217000 REM *** Zj row ****
217010 mat1(RN - 1, 1) = 0
217020 FOR x = 2 TO CN
217030 mat1(RN - 1, x) = 0     'initialize mat1(RN-1,x)
217040 FOR y = 2 TO RN - 2
217050 mat1(RN - 1, x) = mat1(RN - 1, x) + mat1(y, 1) * mat1(y, x)
217060 NEXT y
217070 NEXT x

218100 REM Cj-Zj row
218110 mat1(RN, 1) = 0
218120 mat1(RN, CN) = 0
218130 FOR j = 2 TO CN - 1
218140 mat1(RN, j) = mat1(1, j) - mat1(RN - 1, j)
218150 NEXT j

219900 REM check B(i) see if any negative
219910 FOR i = 2 TO RN - 2
219920 IF mat1(i, CN) < 0 THEN GOTO 291000
219930 NEXT i


220000 REM *** check last row (RN) in mat1 see if any negative
220020 FOR j = 2 TO CN - 1
220030 IF mat1(RN, j) < 0 THEN GOTO 221000
220040 NEXT j
220050 GOTO 291000

221000 REM find the most negative element in row 8
```

```
221010 smallest1 = mat1(RN, 2): cindex1 = 2
221020 FOR j = 3 TO (CN - 1)
221030 IF mat1(RN, j) < smallest1 THEN smallest1 = mat1(RN, j): cindex1 = j
221040 NEXT j

222000 REM check mat1(i,cindex1) to find  pivot row
222010 FOR i = 2 TO RN - 2
222020 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 THEN ratio1(i) = mat1(i, CN) /
mat1(i, cindex1): smallest1 = ratio1(i): rindex1 = i: GOTO 223000
222030 NEXT i
222040 GOTO 291000


223000 FOR i = rindex1 + 1 TO RN - 2
223010 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 THEN ratio1(i) = mat1(i, CN) /
mat1(i, cindex1)
223020 IF mat1(i, cindex1) > 0 AND mat1(i, CN) >= 0 AND ratio1(i) < smallest1 THEN
smallest1 = ratio1(i): rindex1 = i
223030 NEXT i

223040  pivotelement1 = mat1(rindex1, cindex1)

224000 REM *** compute all elements for mat2 ***
224010 REM elements NOT in pivotrow & pivotcolumn in mat2
224020 FOR i = 2 TO RN - 2
224030 FOR j = 2 TO CN
224040 IF i <> rindex1 OR j <> cindex1 THEN mat2(i, j) = mat1(i, j) - mat1(rindex1, j) *
mat1(i, cindex1) / pivotelement1
224050 NEXT j
224060 NEXT i

224100 REM elements in pivotrow for mat2
224110 FOR j = 2 TO CN
224120 IF j <> cindex1 THEN mat2(rindex1, j) = mat1(rindex1, j) / pivotelement1
224130 NEXT j

224200 REM elements in pivotcolumn for mat2
224210 FOR i = 2 TO RN - 2
224220 IF i <> rindex1 THEN mat2(i, cindex1) = 0
224230 NEXT i

224300 REM change the element value in mat2 at mat1 pivotelement position
224310 mat2(rindex1, cindex1) = 1

224400 REM the remaining elements for mat2

224410 'elements in column 1 of mat2
224420 FOR i = 1 TO RN
224430 IF i <> rindex1 THEN mat2(i, 1) = mat1(i, 1)
224440 IF i = rindex1 THEN mat2(rindex1, 1) = mat1(1, cindex1)
224450 NEXT i
```

```
224460 'elements in row 1 of mat2
224470 FOR j = 2 TO CN
224480 mat2(1, j) = mat1(1, j)
224490 NEXT j

227000 REM *** Zj row ****
227010 mat2(RN - 1, 1) = 0
227020 FOR x = 2 TO CN
227030 mat2(RN - 1, x) = 0    'initialize mat2(RN-1,x)
227040 FOR y = 2 TO RN - 2
227050 mat2(RN - 1, x) = mat2(RN - 1, x) + mat2(y, 1) * mat2(y, x)
227060 NEXT y
227070 NEXT x

228100 REM Cj-Zj row
228110 mat2(RN, 1) = 0
228120 mat2(RN, CN) = 0
228130 FOR j = 2 TO CN - 1
228140 mat2(RN, j) = mat2(1, j) - mat2(RN - 1, j)
228150 NEXT j

230000 REM *** check last row (RN) in mat2 see if any negative
230010 FOR j = 2 TO CN - 1
230020 IF mat2(RN, j) < 0 THEN GOTO 231000
230030 NEXT j
230040 GOTO 292000

231000 REM find the most negative element in row 8
231010 smallest2 = mat2(8, 2): cindex2 = 2
231020 FOR j = 3 TO CN - 1
231030 IF mat2(8, j) < smallest2 THEN smallest2 = mat2(8, j): cindex2 = j
231040 NEXT j

232000 REM check mat2(i,cindex2) to find  pivot row
232010 FOR i = 2 TO RN - 2
232020 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 THEN ratio2(i) = mat2(i, CN) /
mat2(i, cindex2): smallest2 = ratio2(i): rindex2 = i: GOTO 233000
232030 NEXT i
232040 GOTO 291000


233000 FOR i = rindex2 + 1 TO RN - 2
233010 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 THEN ratio2(i) = mat2(i, CN) /
mat2(i, cindex2)
233020 IF mat2(i, cindex2) > 0 AND mat2(i, CN) >= 0 AND ratio2(i) < smallest2 THEN
smallest2 = ratio2(i): rindex2 = i
233030 NEXT i

233040  pivotelement2 = mat2(rindex2, cindex2)

234000 REM *** compute all elements for mat3 ***
234010 REM elements NOT in pivotrow & pivotcolumn in mat3
234020 FOR i = 2 TO RN - 2
```

```
234030 FOR j = 2 TO CN
234040 IF i <> rindex2 OR j <> cindex2 THEN mat3(i, j) = mat2(i, j) - mat2(rindex2, j) *
mat2(i, cindex2) / pivotelement2
234050 NEXT j
234060 NEXT i

234100 REM elements in pivotrow for mat3
234110 FOR j = 2 TO CN
234120 IF j <> cindex1 THEN mat3(rindex2, j) = mat2(rindex2, j) / pivotelement2
234130 NEXT j

234200 REM elements in pivotcolumn for mat3
234210 FOR i = 2 TO RN - 2
234220 IF i <> rindex2 THEN mat3(i, cindex2) = 0
234230 NEXT i

234300 REM change the element value in mat3 at mat2 pivotelement position
234310 mat3(rindex2, cindex2) = 1

234400 REM the remaining elements for mat3

234410 'elements in column 1 of mat3
234420 FOR i = 1 TO RN
234430 IF i <> rindex2 THEN mat3(i, 1) = mat2(i, 1)
234440 IF i = rindex2 THEN mat3(rindex2, 1) = mat2(1, cindex2)
234450 NEXT i

234460 'elements in row 1 of mat3
234470 FOR j = 2 TO CN
234480 mat3(1, j) = mat1(1, j)
234490 NEXT j

237000 REM *** Zj row ****
237010 mat3(RN - 1, 1) = 0
237020 FOR x = 2 TO CN
237030 mat3(RN - 1, x) = 0    'initialize mat3(RN-1,x)
237040 FOR y = 2 TO RN - 2
237050 mat3(RN - 1, x) = mat3(RN - 1, x) + mat3(y, 1) * mat3(y, x)
237060 NEXT y
237070 NEXT x

238100 REM Cj-Zj row
238110 mat3(RN, 1) = 0
238120 mat3(RN, CN) = 0
238130 FOR j = 2 TO CN - 1
238140 mat3(RN, j) = mat3(1, j) - mat3(RN - 1, j)
238150 NEXT j

240000 REM check mat3 see if optimal

240010 REM *** check last row (8) in mat3 see if any negative
240020 FOR j = 2 TO CN - 1
240030 IF mat3(8, j) < 0 THEN GOTO 250000
```

```
240040 NEXT j
240050 GOTO 293000

250000 REM change mat3 to mat2 to employ mat2 computation procedures

250010 FOR i = 1 TO RN
250020 FOR j = 1 TO CN
250030 mat2(i, j) = mat3(i, j)
250040 NEXT j
250050 NEXT i
250060 GOTO 231000

290000 REM record Pmin(m,n)
291000 Pmin(m, n) = -1: GOTO 300000

292000 FOR i = 2 TO RN - 2
292010 IF mat2(i, 1) = BM THEN GOTO 292110
292020 NEXT i
292100 Pmin(m, n) = mat2(RN - 1, CN): GOTO 300000
292110 Pmin(m, n) = -1: GOTO 300000

293000 FOR i = 2 TO RN - 2
293010 IF mat3(i, 1) = BM THEN GOTO 293110
293020 NEXT i
293100 Pmin(m, n) = mat3(RN - 1, CN): GOTO 300000
293110 Pmin(m, n) = -1: GOTO 300000

300000 NEXT n
300010 NEXT m

RETURN


REM  *** matrix initialization ***

400011  REM compute Pmin1(m,n)

RN = 8: CN = 11
REM matrix(1) elements
c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = 0
c8 = BM
c9 = BM

a10 = BM
a11 = 1
a12 = 0
```

```
a13 = 0
a14 = 0
a15 = 0
a16 = 0
a17 = 0
a18 = 1
a19 = 0

a20 = 0
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 1
a26 = 0
a27 = 0
a28 = 0
a29 = 0

a30 = 0
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 1
a37 = 0
a38 = 0
a39 = 0

a40 = 0
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 0
a47 = 1
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN
```

```
500011  REM compute Pmin2(m,n)

RN = 8: CN = 11
REM matrix(1) elements
c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = 0
c8 = BM
c9 = BM

a10 = 0
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 1
a16 = 0
a17 = 0
a18 = 0
a19 = 0

a20 = BM
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 0
a26 = 0
a27 = 0
a28 = 1
a29 = 0

a30 = 0
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 1
a37 = 0
a38 = 0
a39 = 0

a40 = 0
a41 = 0
a42 = k2
a43 = 0
a44 = 1
```

```
a45 = 0
a46 = 0
a47 = 1
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN


600011 REM compute Pmin3(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = 0
c8 = BM
c9 = BM

a10 = 0
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 1
a16 = 0
a17 = 0
a18 = 0
a19 = 0

a20 = 0
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 0
a26 = 1
```

```
a27 = 0
a28 = 0
a29 = 0

a30 = BM
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 0
a37 = 0
a38 = 1
a39 = 0

a40 = 0
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 0
a47 = 1
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN

700011 REM compute Pmin4(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = 0
c8 = BM
c9 = BM
```

a10 = 0
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 1
a16 = 0
a17 = 0
a18 = 0
a19 = 0

a20 = 0
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 0
a26 = 1
a27 = 0
a28 = 0
a29 = 0

a30 = 0
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 0
a37 = 1
a38 = 0
a39 = 0

a40 = BM
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 0
a47 = 0
a48 = 1
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0

```
a59 = 1

RETURN

800011 REM compute Pmin5(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = BM
c8 = BM
c9 = BM

a10 = BM
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 0
a16 = 0
a17 = 1
a18 = 0
a19 = 0

a20 = BM
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 0
a26 = 0
a27 = 0
a28 = 1
a29 = 0

a30 = 0
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 1
a36 = 0
a37 = 0
a38 = 0
a39 = 0

a40 = 0
```

```
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 1
a47 = 0
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN

900011 REM compute Pmin6(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = BM
c8 = BM
c9 = BM

a10 = 0
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 1
a16 = 0
a17 = 0
a18 = 0
a19 = 0

a20 = BM
a21 = 0
a22 = 1
a23 = 0
```

```
a24 = 0
a25 = 0
a26 = 0
a27 = 1
a28 = 0
a29 = 0

a30 = BM
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 0
a37 = 0
a38 = 1
a39 = 0

a40 = 0
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 1
a47 = 0
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN

1000011 REM compute Pmin7(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
```

c7 = BM
c8 = BM
c9 = BM

a10 = BM
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 0
a16 = 0
a17 = 1
a18 = 0
a19 = 0

a20 = 0
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 1
a26 = 0
a27 = 0
a28 = 0
a29 = 0

a30 = 0
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 1
a37 = 0
a38 = 0
a39 = 0

a40 = BM
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 0
a47 = 1
a48 = 0
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)

```
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN

1100011 REM compute Pmin8(m,n)

RN = 8: CN = 11

c0 = 0
c1 = COS(theta1)
c2 = COS(theta2)
c3 = SIN(theta1)
c4 = SIN(theta2)
c5 = 0
c6 = 0
c7 = BM
c8 = BM
c9 = BM

a10 = 0
a11 = 1
a12 = 0
a13 = 0
a14 = 0
a15 = 1
a16 = 0
a17 = 0
a18 = 0
a19 = 0

a20 = 0
a21 = 0
a22 = 1
a23 = 0
a24 = 0
a25 = 0
a26 = 1
a27 = 0
a28 = 0
a29 = 0

a30 = BM
a31 = k1
a32 = 0
a33 = 1
a34 = 0
a35 = 0
a36 = 0
a37 = 1
```

```
a38 = 0
a39 = 0

a40 = BM
a41 = 0
a42 = k2
a43 = 0
a44 = 1
a45 = 0
a46 = 0
a47 = 0
a48 = 1
a49 = 0

a50 = BM
a51 = SIN(theta1)
a52 = -SIN(theta2)
a53 = -COS(theta1)
a54 = COS(theta2)
a55 = 0
a56 = 0
a57 = 0
a58 = 0
a59 = 1

RETURN



100000000 REM sort for minimum P1(m,n) -- single mode

REM change Pmin1 .. Pmin4 values when they are -1
FOR m = LM TO HM: FOR n = LN TO HN

IF Pmin1(m, n) = -1 THEN Pmin1(m, n) = 10000 * BM
IF Pmin2(m, n) = -1 THEN Pmin2(m, n) = 10000 * BM
IF Pmin3(m, n) = -1 THEN Pmin3(m, n) = 10000 * BM
IF Pmin4(m, n) = -1 THEN Pmin4(m, n) = 10000 * BM

NEXT n: NEXT m

REM find P1(m,n)=min{Pmin1...Pmin4} -- single mode

FOR m = LM TO HM: FOR n = LN TO HN
   P1(m, n) = Pmin1(m, n)
   IF Pmin2(m, n) < P1(m, n) THEN P1(m, n) = Pmin2(m, n)
   IF Pmin3(m, n) < P1(m, n) THEN P1(m, n) = Pmin3(m, n)
   IF Pmin4(m, n) < P1(m, n) THEN P1(m, n) = Pmin4(m, n)
   IF P1(m, n) = 10000 * BM THEN P1(m, n) = -1  'no Pmin in this case
NEXT n: NEXT m

RETURN
```

```
101000000 REM sort for minimum P2(m,n)
REM change Pmin5 .. Pmin8 values when they are -1
FOR m = LM TO HM: FOR n = LN TO HN

IF Pmin5(m, n) = -1 THEN Pmin5(m, n) = 10000 * BM
IF Pmin6(m, n) = -1 THEN Pmin6(m, n) = 10000 * BM
IF Pmin7(m, n) = -1 THEN Pmin7(m, n) = 10000 * BM
IF Pmin8(m, n) = -1 THEN Pmin8(m, n) = 10000 * BM

NEXT n: NEXT m

REM find P2(m,n)=min{Pmin5...Pmin8} -- double mode

FOR m = LM TO HM: FOR n = LN TO HN
   P2(m, n) = Pmin5(m, n)
   IF Pmin6(m, n) < P2(m, n) THEN P2(m, n) = Pmin6(m, n)
   IF Pmin7(m, n) < P2(m, n) THEN P2(m, n) = Pmin7(m, n)
   IF Pmin8(m, n) < P2(m, n) THEN P2(m, n) = Pmin8(m, n)

   IF P2(m, n) = 10000 * BM THEN P2(m, n) = -1  'no Pmin in this case

NEXT n: NEXT m

RETURN

130000000 REM compute block weight W(m,n)

FOR m = LM TO HM: FOR n = LN TO HN
W(m, n) = ((.5 * (m - 21) ^ 2) * (1 / TAN(theta1) + 1 / TAN(theta2))) * 27
NEXT n: NEXT m

RETURN

140000000 REM indentify block stability(1) -- single mode

FOR m = LM TO HM: FOR n = LN TO HN
   IF P1(m, n) = -1 THEN block1(m, n) = 3
   IF W(m, n) >= Pmax(m, n) THEN block1(m, n) = 3
   IF P1(m, n) < W(m, n) AND W(m, n) < Pmax(m, n) THEN block1(m, n) = 2
   IF W(m, n) < P1(m, n) THEN block1(m, n) = 1
   IF (m - 22) > A1 * (n - 11) THEN block1(m, n) = 0
   IF (m - 22) > A2 * (20 - n) THEN block1(m, n) = 0
NEXT n: NEXT m

RETURN

141000000 REM indentify block stability(2) -- double mode

FOR m = LM TO HM: FOR n = LN TO HN
   IF P2(m, n) = -1 THEN block2(m, n) = 3
   IF W(m, n) >= Pmax(m, n) THEN block2(m, n) = 3
   IF P2(m, n) < W(m, n) AND W(m, n) < Pmax(m, n) THEN block2(m, n) = 2
   IF W(m, n) < P2(m, n) THEN block2(m, n) = 1
```

```
   IF (m - 22) > A1 * (n - 11) THEN block2(m, n) = 0
   IF (m - 22) > A2 * (20 - n) THEN block2(m, n) = 0
NEXT n: NEXT m

RETURN

150000000 REM print out block(m,n) status(1) -- single mode
WIDTH 80
PRINT "single mode"
PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "Ko="; Ko
PRINT "coefficient of friction on plane 1 ="; k1
PRINT "coefficient of friction on plane 2 ="; k2
PRINT
FOR m = HM TO LM STEP -1
FOR n = LN TO HN
PRINT block1(m, n);
NEXT n
PRINT
NEXT m
PRINT
PRINT "0 = outside keyblock region"
PRINT "1 = table"
PRINT "2 = potentially unstable"
PRINT "3 = unstable"

REM print into a file #2

PRINT #2, "single mode"
PRINT #2, "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT #2, "Ko="; Ko
PRINT #2, "coefficient of friction on plane 1 ="; k1
PRINT #2, "coefficient of friction on plane 2 ="; k2
PRINT #2,
FOR m = HM TO LM STEP -1
FOR n = LN TO HN
PRINT #2, block1(m, n);
NEXT n
PRINT #2,
NEXT m
PRINT #2,
PRINT #2, "0 = outside keyblock region"
PRINT #2, "1 = table"
PRINT #2, "2 = potentially unstable"
PRINT #2, "3 = unstable"

RETURN

151000000 REM print out block(m,n) status(2) -- double mode
WIDTH 80
```

```
PRINT "Table 3  Results"

PRINT "ROOF"
PRINT "double mode"
PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "Ko="; Ko
PRINT "coefficient of friction on plane 1 ="; k1
PRINT "coefficient of friction on plane 2 ="; k2
PRINT
FOR m = HM TO LM STEP -1
FOR n = LN TO HN
PRINT block2(m, n);
NEXT n
PRINT
NEXT m
PRINT
PRINT "0 = outside keyblock region"
PRINT "1 = table"
PRINT "2 = potentially unstable"
PRINT "3 = unstable"

REM print into a file #2

PRINT #2, "Table 3  Results"
PRINT #2,
PRINT #2, "ROOF"
PRINT #2, "double mode"
PRINT #2, "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT #2, "Ko="; Ko
PRINT #2, "coefficient of friction on plane 1 ="; k1
PRINT #2, "coefficient of friction on plane 2 ="; k2
PRINT #2,
FOR m = HM TO LM STEP -1
FOR n = LN TO HN
PRINT #2, block2(m, n);
NEXT n
PRINT #2,
NEXT m
PRINT #2,
PRINT #2, "0 = outside keyblock region"
PRINT #2, "1 = table"
PRINT #2, "2 = potentially unstable"
PRINT #2, "3 = unstable"


RETURN


160000000 REM print out P1(m,n)
WIDTH 80

PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "minimum P"
```

```
PRINT "single mode"
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
IF P1(m, n) < > -1 THEN PRINT USING "###.## "; P1(m, n) / 100;
IF P1(m, n) = -1 THEN PRINT USING "###.## "; P1(m, n) = -1;
NEXT n
PRINT
NEXT m

RETURN

161000000 REM print out P2(m,n)
WIDTH 80

PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "minimum P"
PRINT "double mode"
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
IF P2(m, n) < > -1 THEN PRINT USING "###.## "; P2(m, n) / 100;
IF P2(m, n) = -1 THEN PRINT USING "###.## "; P2(m, n) = -1;
NEXT n
PRINT
NEXT m

RETURN

162000000 REM print out Pmax(m,n)
WIDTH 80

PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "maximum P"
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
IF Pmax(m, n) < > -1 THEN PRINT USING "###.## "; Pmax(m, n) / 100;
IF Pmax(m, n) = -1 THEN PRINT USING "###.## "; Pmax(m, n) = -1;
NEXT n
PRINT
NEXT m

RETURN

163000000 REM print out W(m,n)
WIDTH 80

PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "block weight"
```

```
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
PRINT USING "###.## "; W(m, n) / 100;
NEXT n
PRINT
NEXT m

RETURN

170000000 REM print out R1(m,n)=S1(m,n)/N1(m,n)
WIDTH 80

PRINT "Table 1  Ratio of shear force to normal force on plane 1"
PRINT
PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "R1(m,n)=S1(m,n)/N1(m,n) "
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
IF N1(m, n) <> 0 THEN R1(m, n) = S1(m, n) / N1(m, n) ELSE R1(m, n) = 10
PRINT USING "##.## "; R1(m, n);
NEXT n
PRINT
NEXT m

RETURN


171000000 REM print out R2(m,n)=S2(m,n)/N2(m,n)
WIDTH 80
PRINT "Table 2  Ratio of shear force to normal force on plane 2"
PRINT
PRINT "theta1="; theta1 / cr, "theta2="; theta2 / cr
PRINT "R2(m,n)=S2(m,n)/N2(m,n) "
PRINT

FOR m = HM TO LM STEP -1
FOR n = LN TO HN
IF N2(m, n) <> 0 THEN R2(m, n) = S2(m, n) / N2(m, n) ELSE R2(m, n) = 20
PRINT USING "##.## "; R2(m, n);
NEXT n
PRINT
NEXT m

RETURN
```

VITA

Ming Zhao was born in Shanxi, China on October 25, 1956. In February 1978, he entered China University of Mining and Technology where in January 1982, he received his Bachelor of Science degree in civil engineering. After working as a ground control engineer in Datong Coal for three years, he went to Shanxi Mining Institute and worked as a lecturer for five and half years. In July 1990, he went to University of British Columbia, Vancouver where he worked on cable bolt research and completed his Master of Engineering degree in Mining and Mineral Process Engineering in 1993. In August 1993, he came to the University of Tennessee at Knoxville and began a curriculum in civil engineering. He will receive a Master of Science in Civil and Environmental Engineering in December, 1995.