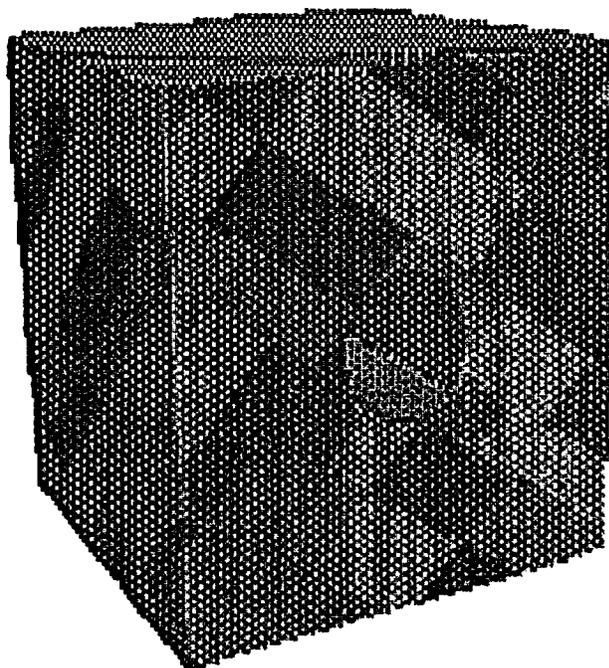


NST  
316189

# 3 D E C

3-D DISTINCT ELEMENT CODE

(Version 1.00)



(C) 1988  
ITASCA CONSULTING GROUP, INC.  
Suite 210  
1313 5th Street SE  
Minneapolis, Minnesota 55414  
September 1988

Sponsored by: FALCONBRIDGE LIMITED  
Toronto, Canada

## 3 D E C

### TERMS AND CONDITIONS FOR LICENSING 3DEC

YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE USING THE 3DEC PROGRAM. INSERTION OF THE 3DEC DISK INTO YOUR COMPUTER INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE AND YOUR MONEY WILL BE REFUNDED.

This program is provided by Itasca Consulting Group, Inc. Title to the media on which the program is recorded and to the documentation in support thereof is transferred to the customer, but title to the program is retained by Itasca. You assume responsibility for the selection of the program to achieve your intended results and for the installation, use and results obtained from the program.

#### LICENSE

You may use the program on only one machine at any one time.

You may copy the program for back-up only in support of such use.

You may not use, copy, modify, or transfer the program, or any copy, in whole or part, except as expressly provided in this document.

You may not sublicense, rent, or lease this program.

#### TERM

The license is effective until terminated. You may terminate it any time by destroying the program together with the back-up copy. It will also terminate if you fail to comply with any term or condition of this agreement. You agree upon such termination to destroy the program together with the back-up copy, modifications, and/or merged portions in any form.

## **3DEC TERMS AND CONDITIONS (continued)**

### **WARRANTY**

Itasca will correct any errors in the code at no charge for twelve (12) months after the purchase date of the code. Notification of a suspected error must be made in writing, with a complete listing of the input and output files and description of the error. If, in the judgment of Itasca, the code does contain an error, Itasca will (at its option) correct or replace the copy at no cost to the user or refund the initial purchase price of the code. Additionally, Itasca will provide, free of charge, all modifications made to the code within twelve (12) months after the purchase date.

### **LIMITATION OF LIABILITY**

Itasca assumes no liability whatsoever with respect to any use of 3DEC or any portion thereof or with respect to any damages or losses which may result from such use, including (without limitation) loss of time, money or goodwill which may arise from the use of 3DEC (including any modifications or updates that may follow). In no event shall Itasca be responsible for any indirect, special, incidental or consequential damages arising from use of 3DEC.

3 D E C  
(Version 1.00)

TABLE OF CONTENTS

	PAGE
TERMS AND CONDITIONS. . . . .	i
1.0 INTRODUCTION . . . . .	1-1
2.0 FORMULATION OF A THREE-DIMENSIONAL DISTINCT ELEMENT MODEL. . . . .	2-1
2.1 A Scheme to Detect and Represent Contacts in a System Composed of Many Polyhedral Blocks. . . . .	2-1
2.1.1 The Data Structure. . . . .	2-2
General Information . . . . .	2-2
Representation of Polyhedra . . . . .	2-3
Representation of Contacts . . . . .	2-5
2.1.2 Identification of Neighbors . . . . .	2-6
Cell Mapping and Searching. . . . .	2-6
Scheme for Triggering Neighborhood Searches. . . . .	2-8

TABLE OF CONTENTS  
(continued)

	PAGE
2.1.3 Contact Detection . . . . .	2-9
Requirements of the Scheme . . . . .	2-9
Direct Tests for Contact Between Two Blocks . . . . .	2-10
The Idea of a Common Plane . . . . .	2-11
Algorithm to Position the Common-Plane .	2-14
Translation of the Common-Plane. . . . .	2-16
Rotation of the Common-Plane . . . . .	2-17
Translation of the C-P During the Mechanical Cycle . . . . .	2-19
Overhead Associated with Common-Plane. .	2-21
Contact Types. . . . .	2-22
2.1.4 Deformable Blocks. . . . .	2-23
2.2 Mechanical Calculations for Motion and Inter- action of a System Composed of Many Polyhedral Blocks. . . . .	2-24
2.2.1 Model Generation . . . . .	2-26

TABLE OF CONTENTS  
(continued)

	PAGE
2.2.2 Calculation Cycle . . . . .	2-29
Contact Force Update . . . . .	2-30
Sub-Contact Force Update . . . . .	2-34
Block Motion Update . . . . .	2-39
Numerical Stability . . . . .	2-42
Damping . . . . .	2-43
2.3 References . . . . .	2-43
3.0 INPUT INSTRUCTIONS AND COMMAND DESCRIPTIONS . . . . .	3-1
3.1 Command-Mode Input . . . . .	3-1
3.1.1 General Command Sequence and Simple Command Description . . . . .	3-4
3.1.2 Full Description of Commands . . . . .	3-8
3.2 Screen-Mode Input . . . . .	3-63
4.0 PROBLEM SOLVING WITH 3DEC . . . . .	4-1
4.1 Introduction . . . . .	4-1
4.2 Suitability of 3DEC . . . . .	4-1
4.3 Model Generation . . . . .	4-3
4.4 Solution Procedure . . . . .	4-8

TABLE OF CONTENTS  
(continued)

	PAGE
5.0 PROGRAM GUIDE. . . . .	5-1
5.1 Offsets for Main Data Arrays. . . . .	5-2
Offsets for Block Data Array. . . . .	5-2
Offsets for Face Data Array . . . . .	5-4
Offsets for Face's Vertex-List Data Array . . . . .	5-5
Offsets for Block's Vertex Data Array . . . . .	5-5
Offsets for Contact Data Array. . . . .	5-6
Offsets for Contact Extension Array . . . . .	5-8
Offsets for Zone Data Array . . . . .	5-9
Offsets for Boundary Vertex Array . . . . .	5-9
Offsets for Joined Blocks Array . . . . .	5-11
Offsets for Joined Vertex-List Array. . . . .	5-11
5.2 Main Common Block (/B/) Variables . . . . .	5-12
/C/ Common Block Variables. . . . .	5-14
/H/ Common Block Variables. . . . .	5-14
Array Limits. . . . .	5-15
5.3 Graphics Transformation Common Block Variables. . . . .	5-15
TRACOM Common Block Variables . . . . .	5-15
BEDCOM Common Block Variables . . . . .	5-17
HPCOM1 Common Block Variables . . . . .	5-18
HPCOM2 Common Block Variables . . . . .	5-18
Array Limits. . . . .	5-19
5.4 History Offsets and Common Block (HISCOM, HCOM1 AND HCOM2) Variables. . . . .	5-19
Offsets for History Arrays. . . . .	5-19
HCOM1 Common Block Variables. . . . .	5-19
HCOM2 Common Block Variables. . . . .	5-20
Array Limit . . . . .	5-20

TABLE OF CONTENTS  
(continued)

	PAGE
5.5 Axial Reinforcement Offsets . . . . .	5-21
Offsets for Axial Reinforcement Array . . . . .	5-21
Offsets for Axial Reinforcement Property Array.	5-22
5.6 Cable Reinforcement Offsets . . . . .	5-23
Offsets for Cable Reinforcement Control Block .	5-23
Offsets for Cable Element Data Array. . . . .	5-23
Offsets for Cable Node Data Array . . . . .	5-24
5.7 Structural Liner Offsets. . . . .	5-26
Offsets for Structural Liner Control Block. . .	5-26
Offsets for Liner Node Array. . . . .	5-26
Offsets for Liner Element Array . . . . .	5-28
5.8 Cable Reinforcement and Liner Material Parameter Offsets . . . . .	5-30
Offsets for Property Array. . . . .	5-30
6.0 EXAMPLE PROBLEMS . . . . .	6-1
6.1 Example 1: Sliding Wedge . . . . .	6-1
6.2 Example 2: Falling Wedge . . . . .	6-2
6.3 Example 3: Tunnel in Faulted Rock. . . . .	6-7
6.4 Example 4: Rock Slope Stability. . . . .	6-10
6.5 References. . . . .	6-17

# **3DEC**

(Version 1.00)

## **1.0 INTRODUCTION**

**3DEC** is the latest version of the distinct element method for simulating the behavior of jointed rock masses. The basis for this code is the extensively-tested numerical formulation used by the two-dimensional version, **UDEC**. This numerical method is specifically designed for simulating both the quasi-static and dynamic response to loading of rock media containing multiple, intersecting joint structures. The distinguishing features of **3DEC** are described below.

- 1. The rock mass is modeled as a 3-D assemblage of rigid or deformable blocks.**
- 2. Discontinuities are regarded as distinct boundary interactions between these blocks; joint behavior is prescribed for these interactions.**
- 3. Continuous and discontinuous joint patterns are generated on a statistical basis. A joint structure can be built into the model directly from the geologic mapping.**
- 4. 3DEC employs an explicit-in-time solution algorithm which accommodates both large displacement and rotation and permits time domain calculations.**
- 5. The graphics facility permits interactive manipulation of 3-D objects. This greatly facilitates the generation of 3-D models and interpretation of results.**

The development of **3DEC** is based on state-of-the-art technology in micro-computers and three-dimensional graphics facilities. The model was developed on an IBM PC/AT, and the **3DEC** graphics has been designed to utilize the Enhanced Graphics Adapter device driver. The model takes full advantage of these graphics facilities by allowing the user to enter input interactively from both a printed command-mode and a graphics screen-mode. In the screen mode, the user can "move" into the model and make re-

gions invisible for better viewing purposes. This allows the user to build the model for a geotechnical analysis and instantly view the 3-D representation.

The program has restart capabilities which permit the saving and restoring of the model file at any stage of the development. This also permits the transfer of files to a mini-computer or larger system for running large models at greater computation speed. For example, a model can be developed quickly on the micro-computer, a file describing the model transferred to a larger computer (such as a VAX or CRAY) for computation, and a file containing the final state of the model transferred back to the micro-computer for post-processing.

3DEC has the facility to create two-dimensional "windows" through the 3-D model. On these windows, output can be presented in the form of principal stress plots, stress contour plots, relative shear plots, and vector plots. Three-dimensional output can also be presented in the form of wire-frame and vector plots. All these plots can be created in screen-mode by single keystrokes which move and rotate the 3-D model, orient the window, and produce the required output (vectors, contours, etc.). The output can then be directed to a hard-copy device for incorporation into reports.

The automatic zone generator in 3DEC allows the user to divide deformable blocks into finite difference tetrahedral zones. A single command allows the user to specify as fine a discretization as needed and to vary the discretization throughout the model. Thus, a fine tetrahedral mesh can be prescribed for blocks in the region of interest and a coarser mesh can be used for blocks further out.

Structural element logic is coupled to the distinct elements in 3DEC in order to simulate structural support interaction with the 3-D blocks. Two types of structural elements are provided:

- (1) cable or rockbolt elements (which can be used to represent fully-grouted or point-anchor rockbolts, cablebolts, or tie-back anchors);  
and
- (2) triangular plate elements (which are joined together to model concrete or shotcrete linings for a tunnel).

The present version of 3DEC is supplied on the Definicon System DSI-780 (32-bit) co-processor board with 4 mbytes RAM and 20 MHz cpu. This board will handle a 3DEC model with up to approximately 1,000 rigid blocks or 800 deformable blocks. The num-

ber of blocks required for an analysis will depend on the frequency of predominant discontinuities in the rock mass and the complexity of the geometry for the underground excavations.

This manual is organized in the following fashion. The formulation of the three-dimensional distinct element model is given in Section 2, which provides the theoretical background for the code. In Section 3, a detailed discussion of the input commands is given, including commands given in the graphics screen model as well as the typed command mode. Section 3 is the primary source for information on control of the 3DEC program. Section 4 describes techniques for the use of 3DEC in problem solving. The guidelines for setting up and executing engineering problems are discussed in this section. Section 5 presents a program guide of the data structure utilized in 3DEC. Section 6 presents example problems which test the various aspects of the code, along with comparisons to closed-form solution when applicable.

## 2.0 FORMULATION OF A THREE-DIMENSIONAL DISTINCT ELEMENT MODEL\*

### 2.1 A Scheme to Detect and Represent Contacts in a System Composed of Many Polyhedral Blocks

The distinct element method has advanced to a stage where the complex mechanical interactions of a discontinuous system can be modeled in three dimensions. An important component is the formulation of a robust and rapid technique to detect and categorize contacts between three-dimensional particles. The technique, described in this section, can detect the contact between blocks of any arbitrary shape (convex or concave) and represent the geometrical and physical characteristics prescribed for the contact (e.g., three-dimensional rock joint behavior). The method utilizes an efficient data structure which permits the rapid calculation on a personal computer of systems involving several hundred particles.

The distinct element method is a way to simulate the mechanical response of systems composed of discrete blocks or particles (Cundall and Strack, 1979). Particle shapes are arbitrary: any particle may interact with any other particle; and there are no limits placed on particle displacements or rotations. In view of this generality, a robust and rapid method must be found to identify pairs of particles that are touching and to represent their geometric and physical characteristics (e.g., whether faces, edges or vertices are involved and what the direction of potential sliding might be).

This section describes a way to perform this task rapidly for a three-dimensional system composed of many blocks. The scheme is embodied in a computer program called 3DEC, which is set up to run on a personal computer. In general, the blocks may be convex or concave, with faces that consist of arbitrary, plane polygons. This discussion addresses only the contact conditions and associated data structures utilized in this formulation; the mechanical calculations are described in Section 2.2.

---

\* This section is published as "Formulation of a Three-Dimensional Distinct Element Model — Part I: A Scheme to Detect and Represent Contacts in a System Composed of Many Polyhedral Blocks (by P. Cundall), *Int. J. Rock Mech., Min. Sci. & Geomech. Abstr.*, 25, 107-116 (1988); and "Formulation of a Three-Dimensional Distinct Element Model — Part II: Mechanical Calculations for Motion and Interaction of a System Composed of Many Polyhedral Blocks," *Int. J. Rock Mech., Min. Sci. & Geomech. Abstr.*, 25, 117-126 (1988) [by R. Hart, P. Cundall and J. Lemos].

Throughout this section, reference will be made to various elements of the data structure. It is important to have a data structure that allows relevant data to be retrieved rapidly when it is needed, particularly in view of the explicit nature of the mechanical calculations, which often entails many thousands of passes through the main cycle. Note that Key (1986) describes a data structure for 3-D sliding interfaces but, in his scheme, the potential interactions must be identified in advance by the user.

### 2.1.1 The Data Structure

General Information — All data are stored in a single main array that can hold real numbers or integers, mixed in whatever way is needed. This is achieved by an EQUIVALENCE statement in FORTRAN. In the micro-computer version of 3DEC, real numbers and integers are stored as 32-bit words. Each physical entity (such as a block, a face or a contact) is represented by a "data element", which is a contiguous group of two or more words. Data elements are allocated dynamically from the main array, as required, and linked to the data structure by pointers. The various types of linked lists and data elements are described in the following two sections. Data elements which are no longer needed (e.g., data from a contact that has broken) are collected together in a heap. If new data elements are needed, this heap is checked before allocating fresh memory. In a program such as 3DEC, elements come in only a small range of possible sizes. Thus, a general "garbage collector" is not required, because new data elements can frequently be allocated from discarded elements of the same length. It should be noted that linked list schemes take very little computer time to maintain: it only requires two or three integer assignments to delete or add an item to a linked list. No re-ordering is necessary.

Every element in the data structure has an address in the main array: an "address" is the index in the main array of the first word of the data element. For instance, rock blocks are not referred to by sequential numbers, or by user-given numbers, but by addresses in the main array. Normally, users do not need to know about addresses, because blocks, contacts, etc. are identified by their coordinates.

A guiding principle in designing the data structure has been to reduce computer time at the expense of using more memory to store data and pointers to data. Memory is rapidly becoming plentiful and cheap, while the processing speed of computers is not increasing at the same rate. Linked lists are used extensively in 3DEC. However, a linked-list data structure is not well-suited to supercomputers that derive their speed from vector processing or from pipelining, because data are not organized sequentially in memory. In contrast, the program is placed to take advantage of a machine consist-

ing of parallel processors connected in a 3-D cubic structure. The program's data space is partitioned naturally by the cell logic described below. Each processor could take charge of one or more cells and the particles contained in them. As particles move, they are re-mapped to adjacent cells—and to adjacent processors, if appropriate. Interactions (contacts) that span processor boundaries are handled by the fast data busses that interconnect processors. Each processor is assumed to have enough local memory to contain all the data associated with the 3DEC cells for which it is responsible. The explicit calculations of 3DEC are well-suited for parallel processing because, conceptually, they are already done in parallel within each step. This is not true for implicit methods, in which all elements interact numerically at each step.

Representation of Polyhedra — There are two types of polyhedral block that can be modeled by 3DEC: rigid blocks, which have six degrees of freedom (three translational and three rotational); and deformable blocks, which are subdivided internally into tetrahedra that have three translational degrees of freedom at each vertex (or node). Rigid blocks have plane faces that are polygons of any number of sides. Figure 2-1 illustrates the data structure for a rigid block. Note that each element in the data structure, although drawn separately, is embedded in the main data array, and connected by pointers as shown. Blocks are accessed via a global pointer that gives entry to a list of all blocks in arbitrary order. The data element for each block contains a pointer that gives access to lists of vertices and faces. Each face element points to a circular list that contains addresses of the vertices that make up the face, arranged in order. It is possible, then, to access vertex data in two ways: first, the vertices can be scanned directly (for example, to update their velocities and coordinates during block motion); and second, the vertices can be accessed via each face (which is useful during contact detection).

The data structure for fully-deformable blocks is similar, but each original polygonal face is discretized into triangular sub-faces, in accordance with the internal discretization into tetrahedra. Each sub-face, and its associated data structure, is exactly like a regular face. However, one word in the block element points to a list of all internal tetrahedral zones. A following section provides more details on deformable blocks.

As far as the user is concerned, 3DEC accepts blocks that are concave, or even hollow or multiply-connected. However, there are so many advantages to convex blocks that, within the program, concave blocks are decomposed into two or more convex blocks: one is termed a "master block"; the others are "slave blocks". In all the logic described here, slave blocks are treated in exactly the same way as master blocks so as to take advantage of convexity. However, in the mechanical calculations, the whole block (master and slaves) is treated as one. Thus, a common center of gravity is determined,

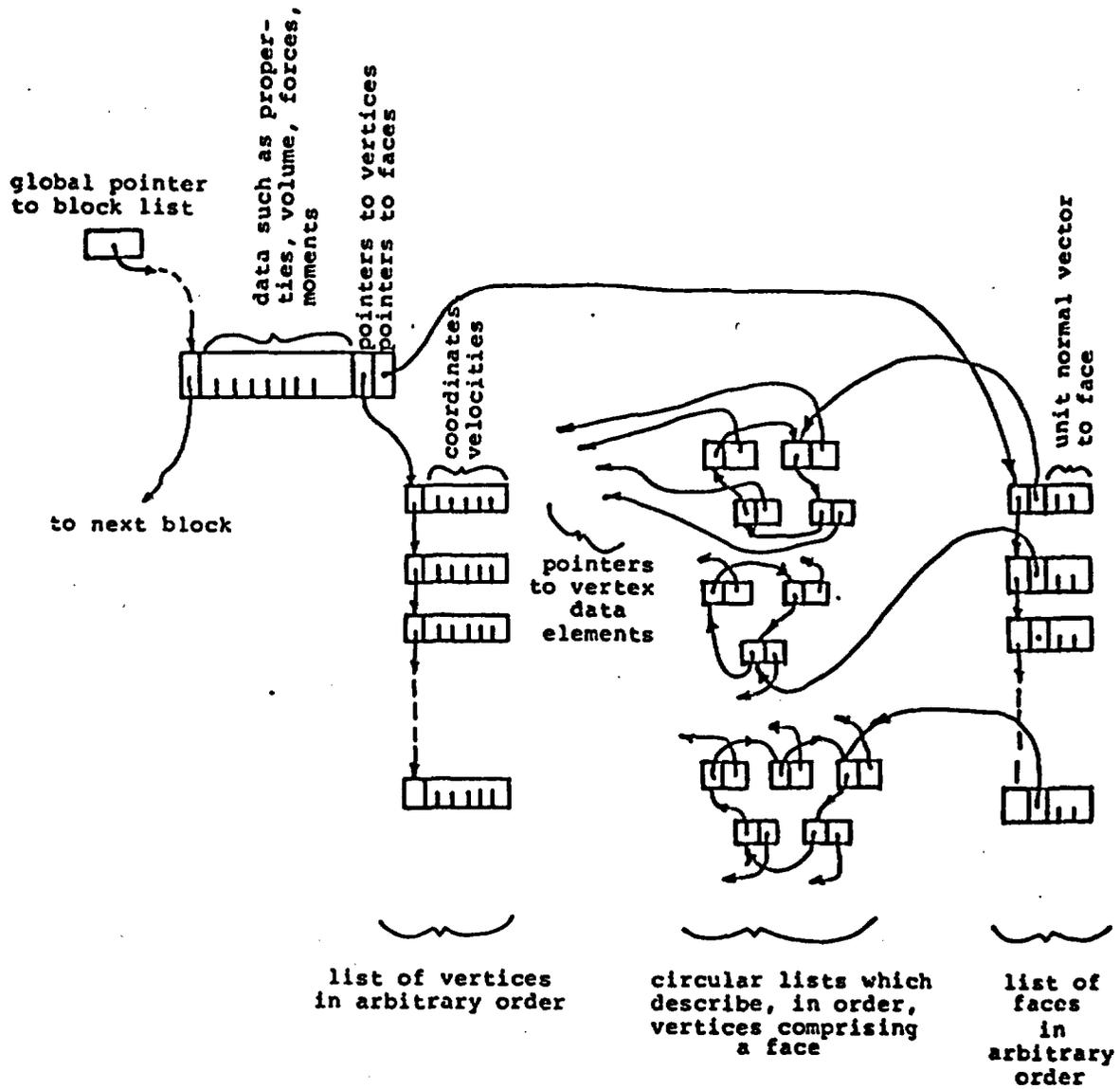


Fig. 2-1 Simplified Data Structure Associated With a Polyhedral Rigid Block

a common mass, and so on. In what follows, convexity will be invoked to justify several procedures, but nothing limits the program's ability to treat blocks of arbitrary shape.

**Representation of Contacts** — For each pair of convex, rigid blocks that touches (or is separated by a small enough gap), a single data element is assigned. This element corresponds to the physical contact between the two blocks and contains relevant data such as friction, shear force, and so on. If either of the blocks is deformable (internally discretized), the single contact element contains a pointer to a series of sub-contacts for each of the contacting nodes on the deformable block(s). The logic for deformable blocks is described in a later section.

The major items in each contact element are shown in Figure 2-2(a). Each contact is linked globally to all other contacts, as well as locally to the pair of blocks that make up the contact. The form of the data structure that embodies this linkage is shown in Figure 2-2(b) for an example system consisting of four blocks. Each contact can be accessed in several ways, depending on the need. During the main calculation cycle, when all forces are updated, it is convenient to scan through all contacts one at a time. This is done by using the linked list that is attached to the global pointer. Once a contact has been accessed, its constituent blocks are identified from the contact's block pointers. During contact detection and re-assignment, it is convenient to know what contacts already exist with a given block. Local lists, which originate with each block, thread through the block's contacts. For example, if, in Figure 2-2(b), the pointer from block C is followed, the three contacts of block C are discovered.

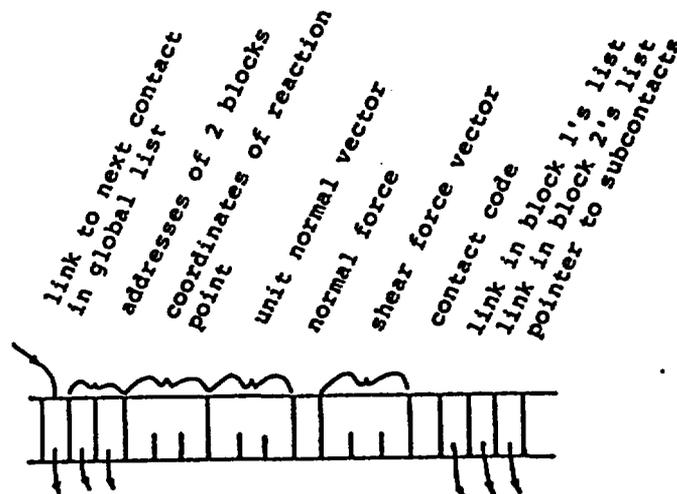


Fig. 2-2(a) Main Items in Contact Data Element

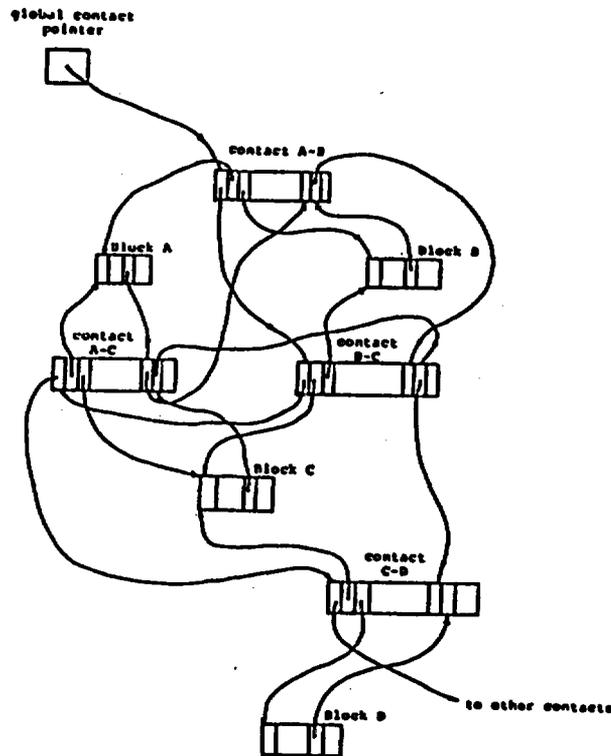


Fig. 2-2(b) Local and Global Contact Links for a Four-Block Example

### 2.1.2 Identification of Neighbors

Before the relative geometry of a pair of blocks can be investigated by the computer program, candidate pairs must be identified. It is prohibitive, in computer time, to check all possible pairs, as the search time increases quadratically with the number of blocks. In two dimensions, it is possible to set up a "canonic" data structure that represents the voids between blocks automatically (Cundall, 1980). It is then a simple matter to scan the local voids surrounding a block in order to obtain a list of all possible contacting blocks. This scheme has a search time that increases linearly with the number of blocks in a system but, unfortunately, the data structure does not translate into three dimensions. A less elegant, but perhaps more robust, scheme was used in programs BALL and TRUBAL, which model disks and spheres, respectively (Cundall & Strack, 1979). This is the method adopted here for the identification of neighbors.

Cell Mapping and Searching — The space containing the system of blocks is divided into rectangular 3-D cells. Each block is mapped into the cell or cells that its "envelope space" occupies. A block's envelope space is defined as the smallest three-dimensional box with sides parallel to the coordinate axes that can contain the block. Each cell stores, in linked-list form, the addresses of all blocks that map into it. Figure 2-3 il-

illustrates the mapping logic for a two-dimensional space (as it is difficult to illustrate the concept in three dimensions). Once all blocks have been mapped into the cell space, it is an easy matter to identify the neighbors to a given block: the cells that correspond to its envelope space contain entries for all blocks that are near. Normally, this "search space" is increased in all directions by a tolerance, so that all blocks within the given tolerance are found. Note that the computer time necessary to perform the map and search functions for each block depends on the size and shape of the block, but not on the number of blocks in the system. The overall computer time for neighbor detection is consequently directly proportional to the number of blocks, provided that cell volume is proportional to average block volume.

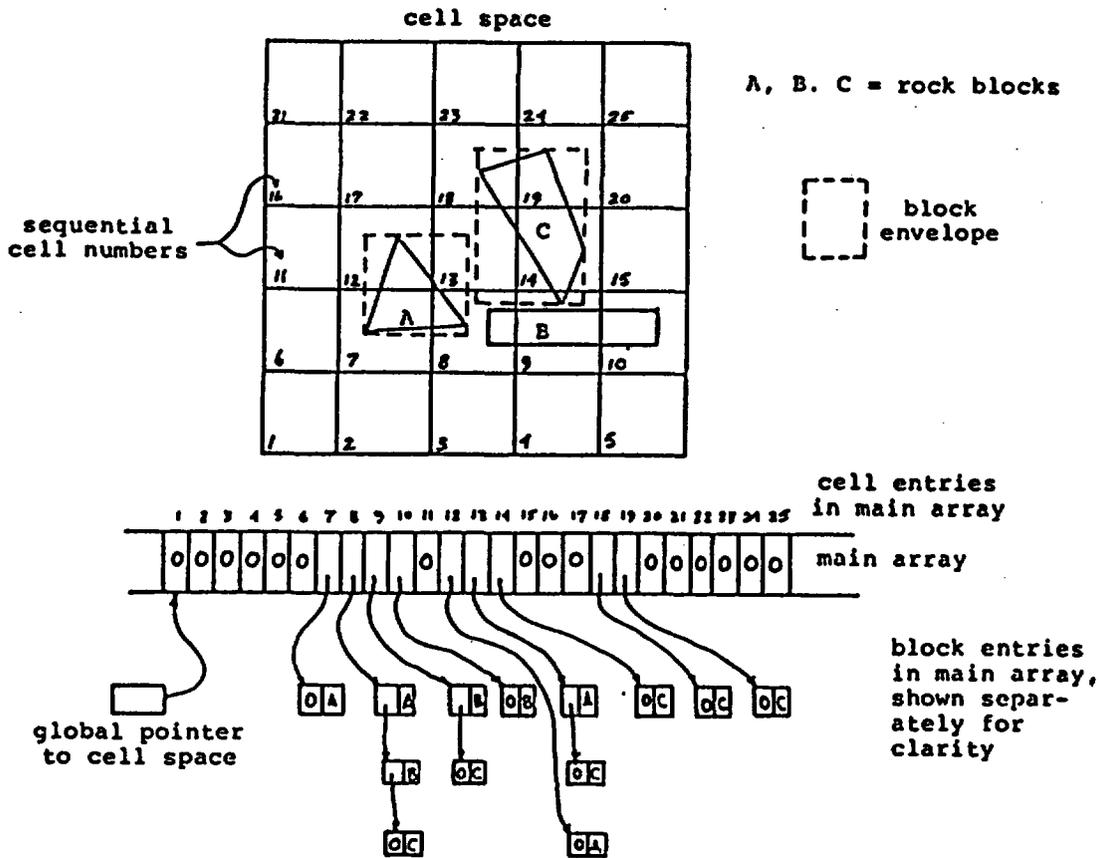


Fig. 2-3 Examples of Block Mapping to Cell Space, In Two Dimensions

It is difficult to provide a formula for optimum cell size because of the variety of block shapes that may be encountered. In the limit, if only one cell is used, all blocks will map into it, and the search time will be quadratic. As the density of cells increases, the number of non-neighboring blocks retrieved for a given block will decrease. At a certain point, there is no advantage in increasing the density of cells, because all the blocks retrieved will be neighbors. However, by further increasing the cell density, the time associated with mapping and searching increases. The optimum cell density must therefore be in the order of one cell per block, in order to reduce both sources of wasted time.

Scheme for Triggering Neighborhood Searches — As a block moves during the course of the simulation, it is re-mapped and tested for contact with new neighbors. This process is triggered by the accumulated movement of the block: a variable  $u_{acc}$ , set to zero after each re-map, is updated at every timestep, as follows:

$$u_{acc} := u_{acc} + \max\{\text{abs}(du)\} \quad (2-1)$$

where  $du$  is the incremental displacement of a vertex, and the  $\max\{\}$  function is taken over all vertices of the block. When  $u_{acc}$  exceeds CTOL (a pre-set tolerance), re-mapping and contact testing is activated. The contact testing is done for a search volume that is  $2 \cdot \text{CTOL}$  larger in all dimensions than the block envelope. In this way, maximum movement of the block, and any potential neighbor, is allowed. If any block attempts to move outside the cell space (i.e., the total volume covered by cells), the cell space is re-defined to be 10% larger in the effected dimension. In this case, a complete re-map of all blocks occurs.

The value of CTOL is also used to determine whether a contact is created or deleted. If two blocks are found to be separated by a gap that is equal to or less than CTOL, a contact is created. Conversely, if an existing contact acquires a separation that is greater than CTOL, the contact is deleted.

The logic described above ensures that the data structure for all potential contacts is in place before physical contact takes place. It also ensures that contact searching is only done for moving blocks; there is no time wasted on relatively inactive blocks.

### 2.1.3 Contact Detection

**Requirements of the Scheme** — After two blocks have been recognized as neighbors, they are then tested for contact: if they are not in contact, the maximum gap between them must be determined so that block-pairs separated by more than a certain tolerance may be ignored. For block-pairs separated by less than this tolerance, but which are not touching, a "contact" is still formed which, although it carries no load, is tracked at every step in the mechanical calculation. In this way, interaction forces start to act as soon as the blocks touch. (Note that contact detection, a lengthy process, is not done at every mechanical step.)

The contact-detection logic must also supply a unit normal vector, which defines the plane along which sliding can occur. This unit normal should be well-behaved (i.e., it should change direction in a continuous fashion) as the two blocks move relative to one another. The logic should be able to handle, in a reasonable way, certain extreme cases, such as that illustrated in Figure 2-4.

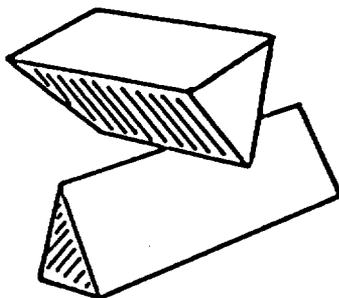


Fig. 2-4 Extreme Case In Which The Determination of Contact Normal is Difficult

Finally, the contact-detection logic must classify the type of the contact rapidly—e.g., face-to-edge, vertex-to-face, etc. This information is needed in order to select the most appropriate physical law to apply at each contact.

In summary, the contact-detection logic must supply, with as little delay as possible, the contact type (if touching), the maximum gap (if not touching), and the unit normal vector. A direct approach is described first. The difficulties with this approach are pointed out, and a better scheme is described.

**Direct Tests for Contact Between Two Blocks** — The simplest approach is to test all possibilities for interaction. In three dimensions, there are many ways for blocks to touch one another—e.g., each vertex of the first block may be tested for contact with each vertex, edge and face of the second block, and so on. If the first block (block A) has  $v_A$  vertices,  $e_A$  edges and  $f_A$  faces, and the second block (block B) has  $v_B$  vertices,  $e_B$  edges and  $f_B$  faces, the number of distinct contact combinations is

$$n = (v_A + e_A + f_A) (v_B + e_B + f_B) \quad (2-2)$$

As an example, two cubes give rise to 676 contact possibilities. In practice, not so many tests are needed, because some types of contact are encompassed by other types, as limiting cases. It appears that only vertex-to-face and edge-to-edge contacts need to be checked, as the other types of contact may be recovered in the following ways:

- (1) vertex-to-vertex is detected when three or more vertex-to-face contacts exist at the same location;
- (2) vertex-to-edge is recognized when two vertex-to-face contacts coincide;
- (3) edge-to-face occurs when two edge-to-edge contacts exist between two blocks; and
- (4) face-to-face is recognized by three or more edge-to-edge contacts or three or more vertex-to-face contacts.

Even when this reduced set of tests is done, the number of combinations is

$$n = v_A \cdot f_B + v_B \cdot f_A + e_A \cdot f_B + e_B \cdot f_A \quad (2-3)$$

For the two-cube example, this number is 240.

Two observations are relevant to what will follow. First, the number of tests depends quadratically on the average number of block edges (or vertices or faces). Second, the tests are not always simple—e.g., in a vertex-to-face test, it is not sufficient just to

check if the vertex lies above or below the face. It is also necessary to show that the vertex lies within the projected perimeter of the polygon that makes up the face; this is not easy to do quickly.

Considering the requirements noted in above, the contact type is determined in the course of the detection calculation, as each type is checked in turn. The determination of the unit normal is easy in some cases (e.g., in all cases involving a face) but difficult in others (in particular, for edge-to-edge, edge-to-vertex and vertex-to-vertex contacts, when it is undefined). Furthermore, there is no guarantee that the contact normal will evolve in a smooth way when there is a jump from one contact type to another. Using this scheme of direct testing, the determination of maximum gap between two arbitrary, non-touching blocks is not a trivial task.

In response to these difficulties, the following new scheme was conceived.

The Idea of a Common Plane — The difficulties noted in the previous section arise from the need to test one arbitrary polyhedron directly with another. Many of the difficulties would vanish if the problem could be split into the following two parts:

- (1) determine a "common-plane" that, in some sense, bisects the space between the two blocks; and
- (2) test each block separately for contact with the common-plane.

The "common-plane" is analogous to a metal plate that is held loosely between the two blocks (see Figure 2-5). If the blocks are held tightly and brought together slowly, the plate will be deflected by the blocks and will become trapped at some particular angle when the blocks finally come into contact.

Whatever the shape and orientation of the blocks (provided they are convex), the plate will take up a position that defines the sliding plane for the two blocks. To carry the analogy a bit further, imagine that the plate is now repelled by the blocks even when they do not touch. As the blocks are brought together, the plate will take up a position midway between them, at a maximum distance from both. Then we can easily find the gap between the blocks, simply by adding the block-to-plate distances. In fact, there are many things that would become easier if we could somehow contrive a numerical equivalent for the metal plate (the common-plane referred to above). Supposing for the moment that we do have a way to do this, the task of testing for contact is simplified and

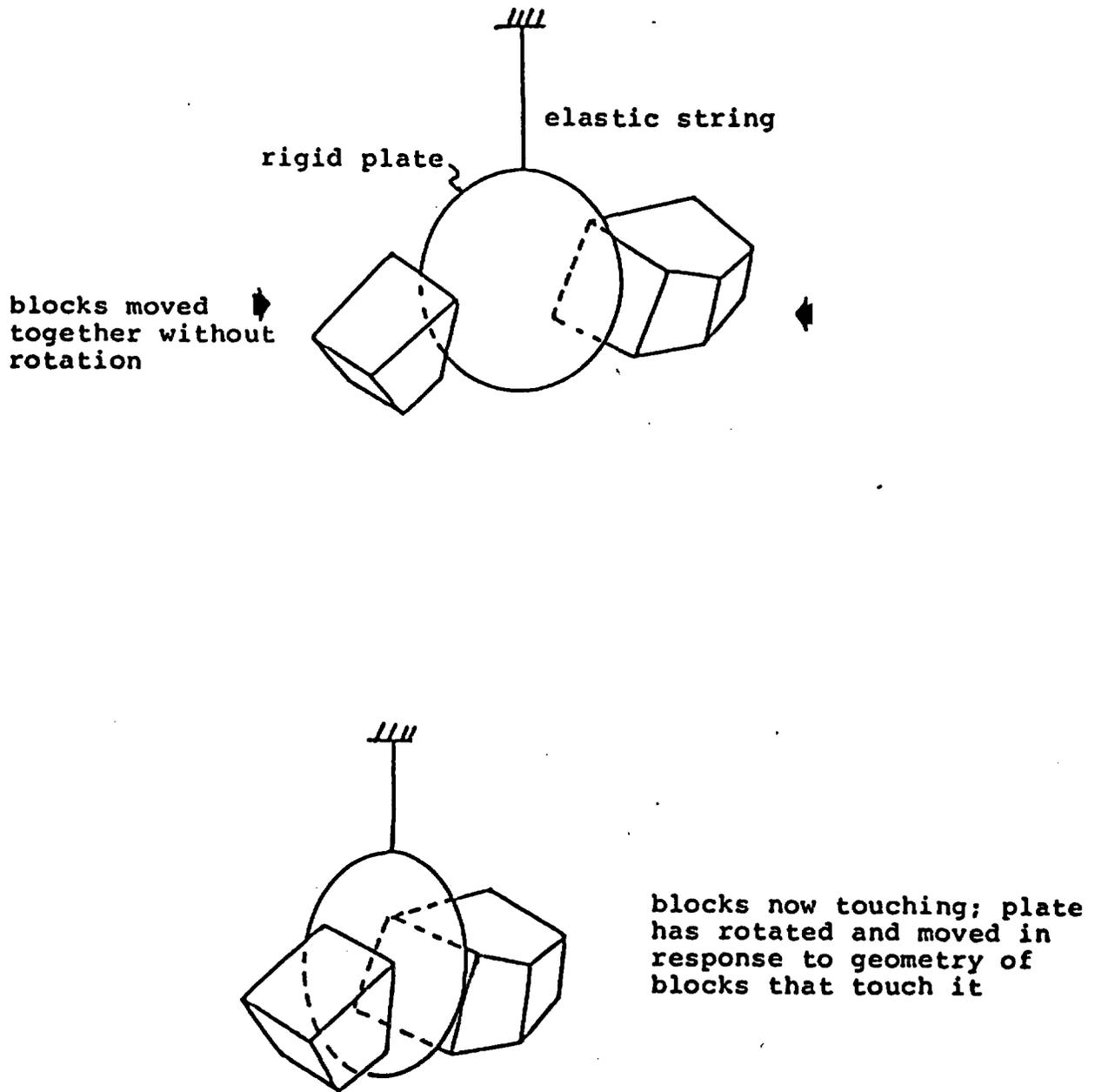


Fig. 2-5 Visualization for Positioning of Common-Plane in Response to Block Geometry

speeded up in the following ways. (The term "common-plane" is abbreviated below as "c-p".)

1. Only simple vertex-to-plane tests need to be carried out (using dot products). Because the blocks (or sub-blocks) are convex (see above), face and edge contacts are recognized simply by counting the number of vertex-to-plane overlaps for both blocks.
2. The number of tests depends linearly on the number of vertices (compared to the quadratic dependence noted above). Because we test the vertices of block A with the c-p, and separately test the vertices of block B with the c-p, the number of tests is

$$n = v_A + v_B \quad (2-4)$$

For the two-cube example, the number of tests is 16, as compared with 240 noted above.

3. There is no need to test if a potential contact lies within the perimeter of a face. If both blocks touch the c-p, then they must touch each other. (If they did not touch, then the c-p would touch neither, as the c-p is defined as bisecting the space between blocks.)
4. The unit contact normal is equal to the c-p unit normal—no additional calculations are necessary.
5. Since the c-p normal is uniquely defined, the problem of discontinuous evolution of the contact normal is eliminated. The c-p normal may change rapidly (as in vertex-to-vertex contact), but it will not jump due to a change in contact type.
6. The determination of minimum gap between two non-touching blocks is trivial: it is simply the sum of the distances of the two blocks from the c-p.

**Algorithm to Position the Common-Plane** — Having established that the common-plane simplifies and speeds up contact testing, we need to provide a means to position the common-plane and to show that the overhead associated with it does not outweigh the benefits that it brings to contact testing. The algorithm for locating and moving the c-p is based on geometry alone and is applied at every time step, in parallel with the mechanical calculations. The algorithm is stated in words as:

**"Maximize the gap between the c-p and the closest vertex."**

Figure 2-6 shows several examples, in two dimensions, of c-p which are consistent with the algorithm given above. Any rotation or shift of the c-p would reduce the gap between c-p and the closest vertex or leave it unchanged. For overlapping blocks, the same algorithm applies, but the words "gap" and "closest" must be used in their mathematical sense for the case of negative signs—i.e., "gap" means "negative overlap" and "closest" means most "deeply buried." To improve readability, the algorithm may be restated for the case of overlapping blocks:

**"Minimize the overlap between the c-p and the vertex with the greatest overlap."**

Starting conditions must be provided to the algorithm for two blocks which are not already recognized as being in contact. An initial guess is made: the c-p is placed midway between the centroids of the two blocks, with a unit normal vector pointing from one centroid to the other:

$$C_i = \frac{A_i + B_i}{2} \tag{2-5}$$

$$n_i = \frac{Z_i}{z}$$

where  $Z_i = B_i - A_i$ ,

$$z^2 = Z_i Z_i,$$

$n_i$  = unit c-p normal,

$C_i$  = reference point of c-p,

$A_i$  = position vector of block A's centroid,

$B_i$  = position vector of block B's centroid, and

indices  $i, j$  and  $k$  take the values 1 to 3 and denote components of a vector or tensor. (The summation convention applies for repeated indices.)

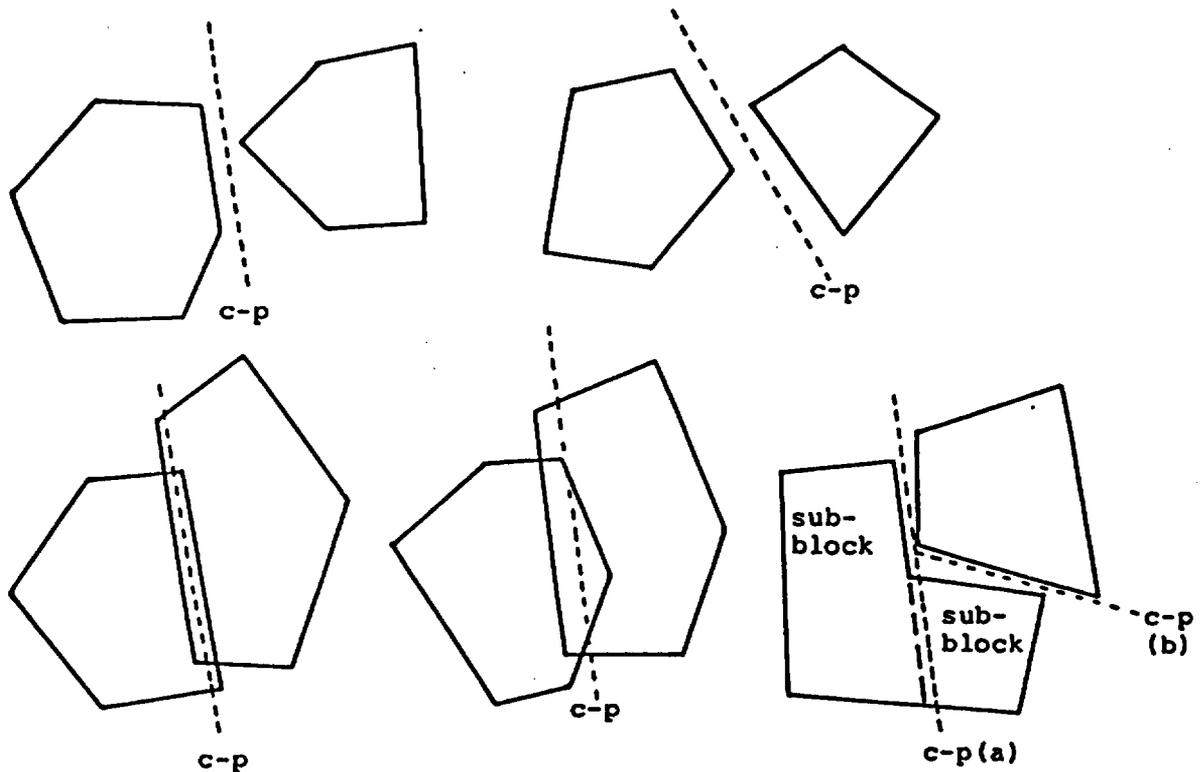


Fig. 2-6 Examples of the Common-Plane Between Two Blocks

The algorithm then applies a translation and a rotation to the c-p so as to maximize the gap (or minimize the overlap). The function of the reference point,  $C_i$ , is two-fold: it is (a) the point about which c-p rotations are applied, and (b) the reaction point for normal and shear forces when the two blocks are touching.

Translation of the Common-Plane — The translation is divided into parts that are normal and tangential to the c-p. The normal translation is found by scanning each block for its nearest vertex to the c-p:

$$d_B = \min\{ n_i V_i(B) \} \quad (2-6)$$

$$d_A = \max\{ n_i V_i(A) \} \quad (2-7)$$

where  $d_A$  is the distance to the nearest vertex on A (negative for a gap),  
 $d_B$  is the distance to the nearest vertex on B (positive for a gap),  
 $V_i(A)$  is the position vector of a vertex on A,  
 $V_i(B)$  is the position vector of a vertex on B,  
 $\min\{ \}$  is the minimum, taken over all vertices of B, and  
 $\max\{ \}$  is the maximum, taken over all vertices of A.

The shift in the c-p's reference point is then

$$C_i := \frac{C_i + (d_A + d_B)n_i}{2} \quad (2-8)$$

The total gap is  $d_B - d_A$ . If the blocks are touching (i.e., the total gap is negative), the reference point becomes the reaction point for contact forces and is determined in the part of the program that deals with the mechanical calculation (see below). For non-touching blocks, the reference point is moved mid-way between the nearest vertices:

$$C_i = \frac{V_i(A_{\max}) + V_i(B_{\min})}{2} \quad (2-9)$$

where  $V_i(A_{\max})$  is the vertex on A nearest to the c-p, and

$V_i(B_{\min})$  is the vertex on B nearest to the c-p.

Rotation of the Common-Plane — Whereas translation of the c-p can be done in one step, rotation must be done iteratively, because the nearest vertex on a block can change as the c-p is rotated. Since the unit normal can rotate about two independent axes, the maximization of gap is equivalent to a hill-climbing process. Two orthogonal axes are chosen arbitrarily, both orthogonal to the c-p unit normal vector. The unit normal is perturbed in each of these directions, both in a positive and negative sense, making four perturbations in all. If  $p_i$  and  $q_i$  are the orthogonal unit normal vectors, the four perturbations to  $n_i$  are:

$$n_i := \frac{n_i + kp_i}{z}$$

$$n_i := \frac{n_i - kp_i}{z}$$

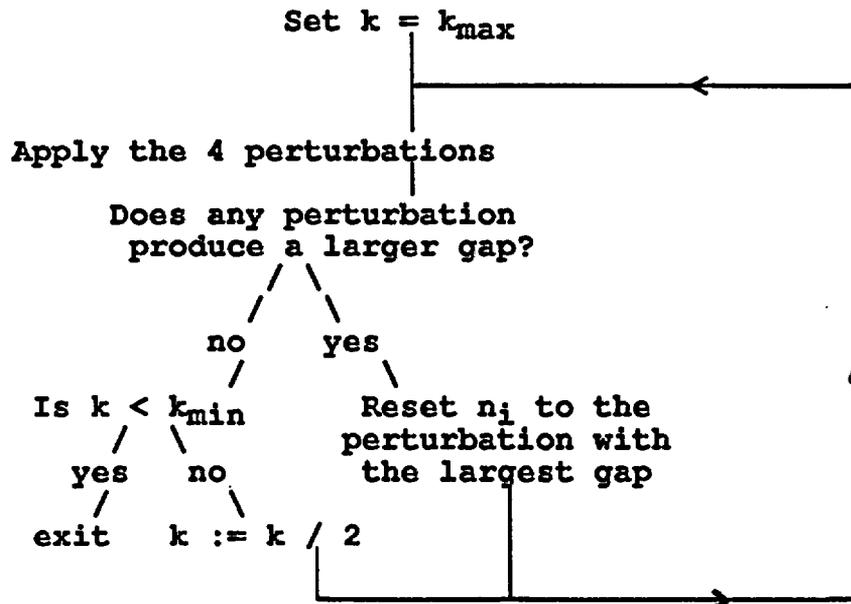
$$n_i := \frac{n_i + kq_i}{z}$$

$$n_i := \frac{n_i - kq_i}{z}$$

(2-10)

where  $z^2 = 1 + k^2$ , and  $k$  is the parameter that determines the size of the perturbation.

When a contact is being created for the first time, the parameter  $k$  is set initially to  $k_{\max}$ , a value that corresponds to a five-degree perturbation in angle. The following iteration procedure is then used to search for a maximum value of gap. Note that maximum gap is defined as  $\max\{d_B, d_A\}$ , with  $d_A, d_B$  given by Eqs. (2-6) and (2-7).



The iteration stops when the gap decreases for all four of the smallest perturbations; it follows that this final unit normal is the one that gives rise to the largest gap. The smallest perturbation,  $k_{\min}$ , has a value that corresponds to an angle change of 0.01 degree. In order to prevent the iteration terminating prematurely on a saddle-point, the perturbation axes are rotated by 45 degrees on alternate cycles of the iteration. If the gap, at any stage in the iteration, exceeds the tolerance set for contact formation (CTOL), the iteration process halts and the contact is deleted.

When a contact already exists, the four perturbations are initially tried with  $k = k_{\min}$ . If there is no increase in maximum gap, nothing further is done. Otherwise, the iteration given above is performed.

Translation of the C-P During the Mechanical Cycle — When forces exist on a contact, the normal translation of the c-p is still done according to Eq. (2-8). However, the following two further incremental translations are applied to the c-p:

(1) rigid body translation

The first part of the additional translation is related to the average motion of the two blocks, at the contact point:

$$C_i := \frac{C_i + [du_i(A) + du_i(B)]}{2} \quad (2-11)$$

where  $du_i(A)$  and  $du_i(B)$  are the incremental displacements of blocks A and B, respectively, at the contact point (reference-point).

For example, if both blocks are moving through space at the same speed, the reference-point on the c-p will also be moved at this speed. In such a case, the correction supplied by Eq. (2-8) will be unnecessary. Note that the incremental displacements in Eq. (2-11) include the effects of block rotation.

(2) relative rotation

As mentioned before, the reference-point of the c-p is assumed to be the point at which the resultant contact force acts. As the upper surface of a contact plane rotates relative to the lower surface, the resultant contact force will move, since the plane will become unequally loaded. Figure 2-7 illustrates this effect. The relation between the movement of the reference-point and the rotation of the surfaces must depend on the nature of the interface, and on the current normal stress if the normal stiffness is stress-dependent. It appears that the relation described above is a material property and cannot be derived from geometrical factors alone. In 3DEC therefore, the translation of the reference-point is taken as the relative angle change of the two blocks multiplied by a user-specified constant,  $K_T$ :

$$C_i := C_i + K_T e_{ijk} n_j [dT_k(B) - dT_k(A)] \quad (2-12)$$

where  $dT_k(A)$  is the incremental rotation vector of block A,

$dT_k(B)$  is the incremental rotation vector of block B, and

$e_{ijk}$  is the permutation tensor.

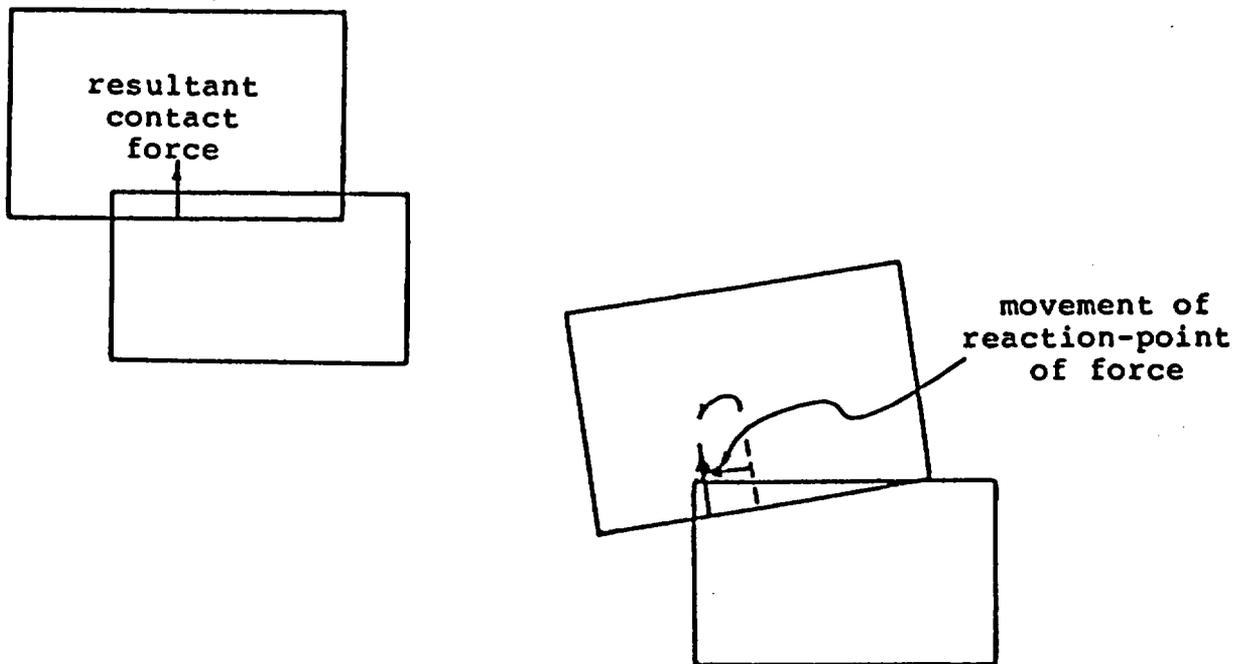


Fig. 2-7 Illustration of the Movement of the Resultant Contact Force Due to Rotation of the Surfaces in Contact (overlaps are exaggerated)

$K_T$ , as defined above, has the dimension of length, but it should probably be normalized by the length of the contact plane in the direction of movement of  $C_i$ . However, more data is needed from laboratory tests before  $K_T$  can be defined properly.

(3) limit to movement of reference-point

Because the reference-point is the point at which contact forces act, it must lie on the surface of both blocks. After applying Eqs. (2-11) and (2-12), the reference-point is tested against each face of the two blocks comprising the contact. If it is found to lie outside any one, it is brought back toward the face as follows:

$$C_i := C_i + [n_i n_k n_k(f) - n_i(f)] \cdot d \quad (2-13)$$

where  $n_i(f)$  is the outward unit normal of the face, and  $d$  is the normal distance from the face to  $C_i$ .

Formula 2-13 brings  $C_i$  completely back to the face only when  $n_i$  is orthogonal to  $n_i(f)$  but, in other cases, the repeated use of the formula (which happens automatically, as it is invoked in every calculation cycle) achieves convergence. Figure 2-8 illustrates this. The same effect will occur if  $C_i$  should fall outside two or more faces simultaneously, which may happen at a corner or edge.

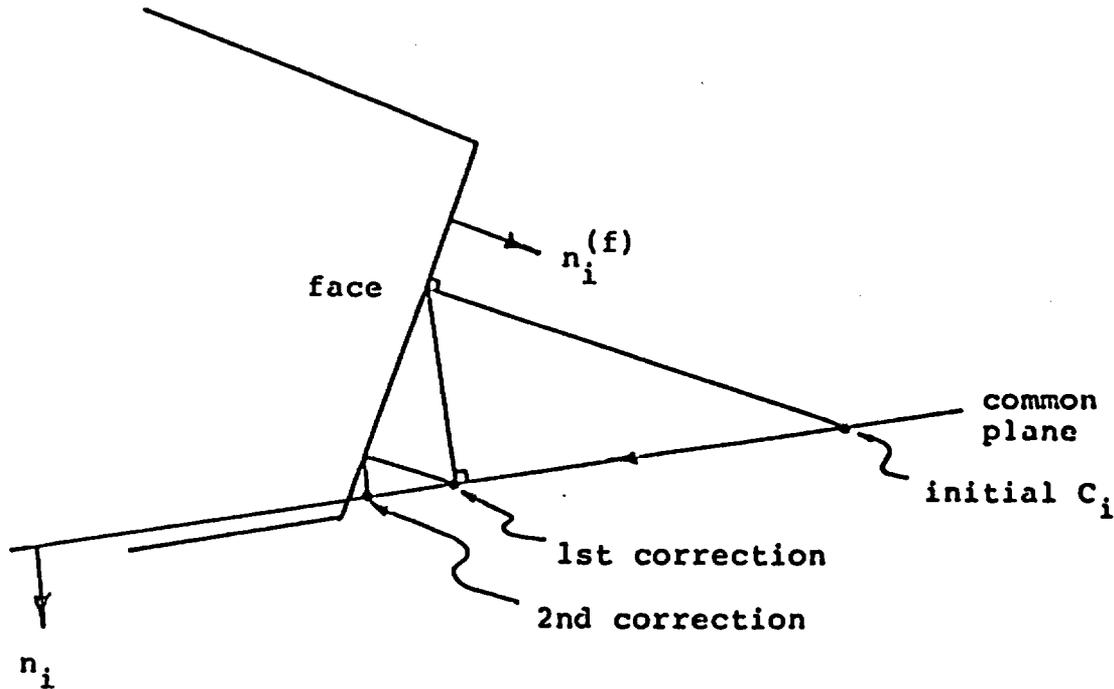


Fig. 2-8 Procedure to Bring Reference Point  $C_1$  Within Block Boundary

Overhead Associated with Common-Plane — The number of operations necessary to establish the c-p for a pair of blocks depends linearly on the number of vertices. For the translation correction of Eq. (2-8), the number of tests is  $v_A + v_B$ . For the rotation iteration, the number of tests is  $4N(v_A + v_B)$ , where  $N$  is the number of iterations. The total number of tests is therefore

$$n = (4N + 1) (v_A + v_B) \quad (2-14)$$

It is difficult to compare this formula directly with Eqs. (2-1) and (2-2), which correspond to the direct-testing scheme to find contacts. For contacts already established, the c-p scheme is superior to the direct-testing scheme, because only one rotation iteration is usually needed to confirm that the c-p position is optimal. However, on initial contact formation, the number of iterations may be in the range 9 to 30 (found by experiment). For blocks with few vertices, the c-p scheme takes more tests to establish contact conditions than the direct-testing scheme. However, when the six advantages noted previously are considered, the c-p scheme is preferred overall.

**Contact Types** — Contact type is important because it determines the mechanical response of the contact. For example, a vertex-to-face contact will behave differently from a face-to-face contact. In rock mechanics, face-to-face contacts are thought of as "joints" in which stresses are the important variables, rather than forces. Contacts may be classified into types by noting how many vertices of each block touch the c-p. The following table relates the contact type to the number of touching vertices from each block.

For a face-to-face contact, it is necessary to define an area of contact in order to use a stress-displacement law for the mechanical behavior of the interface. Because both faces comprising the contact are convex polygons, the common area of contact is also a convex, simply-connected polygon. The calculation of common area is then a straightforward, but lengthy, procedure.

Table 2-1

## CONTACT TYPES

Number of Vertices Touching

<u>Block A</u>	<u>Block B</u>	<u>Contact Type</u>
0	0	null
1	1	vertex-vertex
1	2	vertex-edge
1	>2	vertex-face
2	1	edge-vertex
2	2	edge-edge
2	>2	edge-face
>2	1	face-vertex
>2	2	face-edge
>2	>2	face-face

**2.1.4 Deformable Blocks**

The scheme described above is also used for deformable blocks—only one common-plane is found for each block pair that is in contact and only one regular data element is allocated for the contact. If one block of the pair is deformable, then the face that is in contact with the c-p may contain a number of internal surface nodes, each of which has three independent degrees of freedom. In this case, a "sub-contact" is created for each node on the face. The sub-contact keeps track of the interface forces of its associated node and other conditions such as sliding and separation. In all respects, the sub-contact acts as though it is a normal vertex-to-face contact—the interface displacement at each node is taken as the node displacement minus the displacement of the coincident point on the opposing face. Forces are calculated at each sub-contact from the appropriate physical law (e.g., elastic plus Coulomb friction in the shear direction).

We have discussed only the nodes on one side of an interface. If the other side of the interface is also a face belonging to a deformable block, then identical conditions apply: sub-contacts are created; and relative displacements and hence forces are calculated.

When two deformable blocks come together, the contact logic described above is equivalent to two sets of contact springs in parallel—in this case, the forces from both sets are divided by two, so that the overall interface behavior is the average of that of both sets. Although, numerically, the contacts with deformable blocks look like vertex-to-face contacts, they can be made to behave like face-to-face contacts by associating an area with each node, and a stress-displacement contact law can then be used. The area "owned" by each surface node is, in general, equal to one third of the area of the surrounding triangles, but this calculation must be adjusted when the node is close to one or more edges on the opposing block.

The c-p logic described previously is strictly applicable only to convex blocks with planar faces, but these conditions may be violated if large strains occur with deformable blocks. In practice, the program is used to model a rock mass, where displacements may be large but strains usually quite small. In these circumstances, the logic will still work but, in situations where block strains become large (say, >1%), the scheme may need to be modified.

At present, 3DEC does not allow the use of rigid and deformable blocks in the same problem. Also, the interaction between deformable blocks is restricted to the case in which the blocks contact along faces that are approximately parallel to the c-p. In addition, only small relative displacements between deformable blocks can be handled presently. In order to allow large displacements, the logic must be extended by incorporating a procedure to relocate automatically each sub-contact, as the associated vertex crosses a face boundary in the other block. The logic should also avoid abrupt deletion of a sub-contact whose associated vertex slides out of the other blocks' faces. The existing sub-contact forces should be reallocated to ensure a smooth transition between neighboring states, as in the two-dimensional code UDEC.

## 2.2 Mechanical Calculations for Motion and Interaction of a System Composed of Many Polyhedral Blocks

It is often difficult to define a particular numerical method and distinguish it from other methods which may be similar. For example, discontinuous bodies have been modeled numerically by Burman (1971), Rodriguez-Ortiz (1974), Chappell (1979), Goodman and Shi (1985) and Key (1986), among others. Numerical techniques in geomechanics tend to overlap, especially when modifications are made that borrow features from other methods that are supposedly different. The "distinct element method" is not very different from other techniques, particularly when all existing variations are considered, but three distinct features are usually associated with it. First, bodies can undergo large ro-

tation and large displacements relative to one another; second, interaction forces between bodies arise from changes in their relative geometrical configuration; and, third, the solution scheme is explicit in time.

Because of these features, the distinct element method is particularly well suited to investigate problems in rock mechanics which address the question of stability of discontinuous rock masses. The important points of the distinct element formulation as it relates to rock mass stability are as follows.

1. Both stability and instability are modeled. When a net force exists on the block, it accelerates and moves to a new position. If, the forces balance, then either the system remains at rest, or it moves with constant velocity.
2. Forces arise between two blocks when the blocks intersect. Normally, the overlap is small in relation to block dimensions.
3. The calculation marches from one state to another in small increments of time. The "final solution" may be equilibrium or it may be a state of continuing motion.

A principal criticism of the method in the rock mechanics community is that the formulation has only been developed for two-dimensional systems. In general, a two-dimensional model cannot fully represent the behavior of a discontinuous rock.

This section describes the formulation of the mechanical calculations of the distinct element method in three dimensions. This discussion is based upon work presented previously by Cundall and Strack (1979) and Cundall and Hart (1985). The description of the formulation is given with specific application to stability analysis in rock mechanics and only addresses the mechanical calculation for rigid body motion and block interaction. This description of motion is a sufficient representation for stability studies in which the applied stress state is low compared to the intact rock strength and in which motion is concentrated along structural features.

The three-dimensional program also accounts for block deformability and failure of intact material. In this formulation, each polyhedral block is subdivided into an internal finite difference mesh consisting of constant strain tetrahedral elements. The solution is an explicit, large-strain one, with elastic and elasto-plastic models for the block material. A more complete description of the deformable block logic will be presented in a subsequent paper, but an analogous formulation for two-dimensional blocks is given by

Lemos (1987). The contact detection scheme for three-dimensional rigid and deformable blocks is discussed in Section 2.1.

The three-dimensional formulation described here is embodied in a computer program called 3DEC, which is written specifically to run on a personal computer. 3DEC makes liberal use of interactive microcomputer graphics in three dimensions to assist in generation of the model and presentation of results. A description of model generation is provided to demonstrate the application of interactive graphics for rock mechanics analysis.

### 2.2.1 Model Generation

A prerequisite of a three-dimensional model of discontinuous rock is the ability to generate joint patterns that resemble those in the natural rock. Because of the variability of rock structure, joint patterns must be described in statistical terms. Lemos (1987) describes a procedure proposed by Cundall for generation of joint sets in a two-dimensional distinct element model. The extension to three dimensions is described herein.

The 3DEC code uses a joint set generator which permits the creation and superposition of several joint sets in order to create a blocky structure. Sets ranging from single faults to a system of many joints can be created. In this model, a joint set is characterized by six parameters:

- (1) dip direction;
- (2) dip angle;
- (3) spacing between joints in the set;
- (4) number of joints in the set;
- (5) location point for one joint in the joint set; and
- (6) persistence (i.e., probability that any given block lying in the path of a joint will be split).

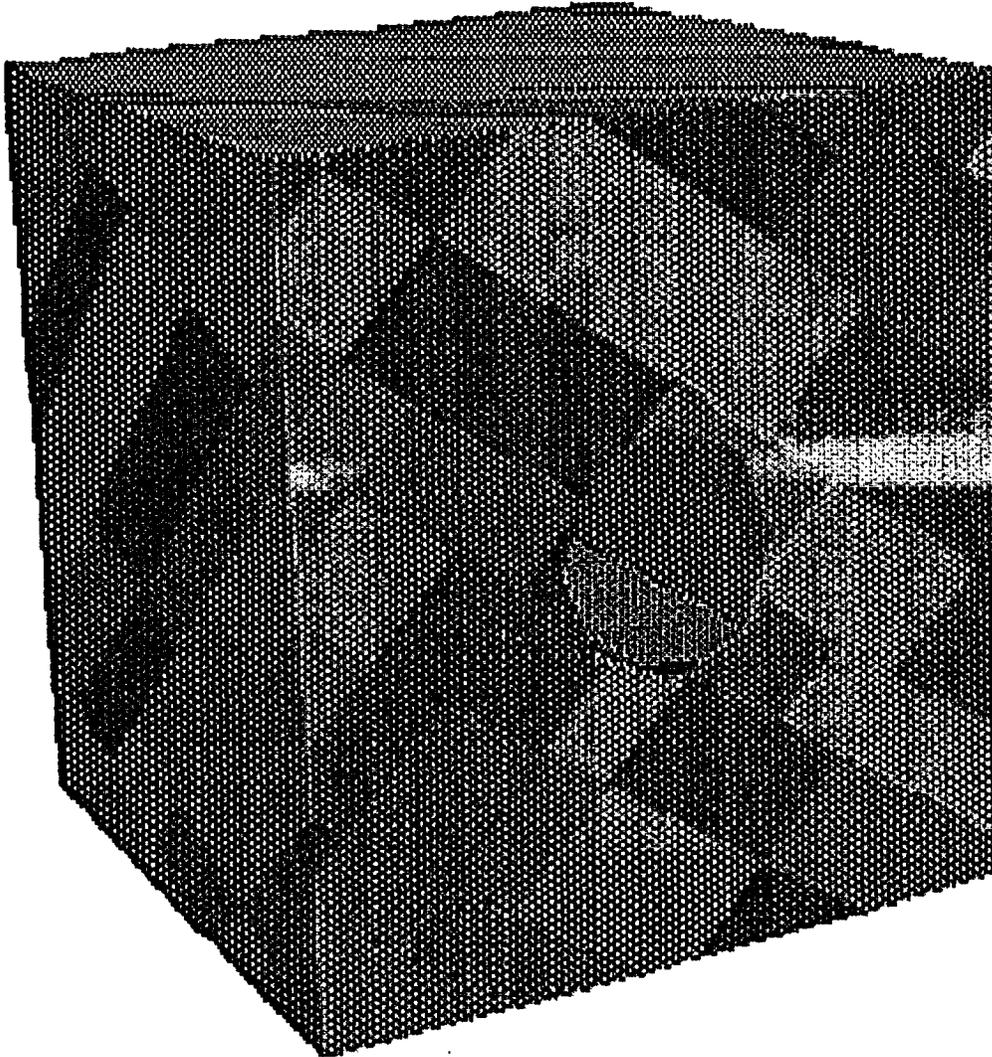
The dip direction, dip angle and spacing are each defined by a mean value and a random deviation from the mean, according to a given probability distribution. At present, a uniform probability density function is used in 3DEC. The persistence parameter creates discontinuous joints in the model and describes the areal extent of the discontinuity

within a plane. For a persistence equal to 1.0, all blocks along the joint plane are split; for a persistence equal to 0.5, half of the blocks are split, on average.

The joint generation process is conducted in 3DEC by issuing the description for the joint set in the command mode and viewing the results in the graphics screen mode. The user can switch back and forth between command and screen modes to view the result of the joint set command directly after the command is given. Alternately, joints can be generated individually in the screen mode by single strokes made directly from the keyboard. Discontinuous joints can be created manually by first "hiding" blocks, in either the command mode or screen mode, and then generating joints. Hidden blocks are not split by the joint generator command and can be made visible again after generation is completed.

A simple example of the application of the joint generator in command mode is demonstrated in Figure 2-9. A blocky system is created from the following sets of joint commands.

<u>SET</u>	<u>1</u>	<u>2</u>	<u>3</u>
dip angle			
mean	45°	45°	45°
deviation	0°	0°	0°
dip direction			
mean	220°	70°	70°
deviation	0°	0°	0°
spacing			
mean	0.25	0.5	0.5
deviation	0.1	0	0
persistence	1.0	0.5	0.5
number	15	15	15
origin	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)



**Fig. 2-9** Blocky System Created With Three Joint Set Commands  
(Then, circular tunnel is "excavated" with the **TUNNEL** command.)

Discontinuous joints can also be made by "joining" blocks. This technique, described in Section 2.1, can be used to join blocks in such a fashion as to create discontinuous joints and internal cracks within the model.

The joint generator is quite general and allows the user to "cut up" the model to represent the defined structural features. Also, joints are identified individually so that different constitutive behavior and material properties can be specified for each joint or joint set.

In addition to splitting blocks with the joint generator, blocks can be created separately. For example, a block representing a foundation footing can be created and located on the top of the model shown in Figure 2-9.

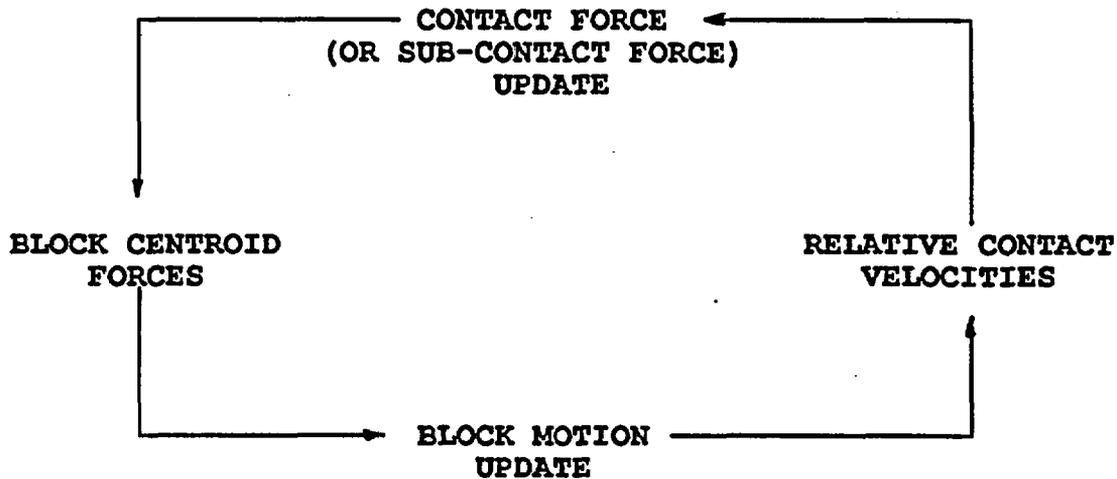
Finally, holes can be created anywhere in the model. An automatic tunnel generator is used to create single or intersecting tunnels, drifts, etc. With this generator, the user defines faces at the ends of the tunnel by a sequence of vertices. Straight lines connecting the two faces define the tunnel boundary. Any convex shape of the tunnel face can be prescribed by an arbitrary number of vertices. The faces may be positioned either outside the model boundary or inside the model. Figure 2-9 illustrates a circular tunnel in the block system. Blocks created to facilitate tunnel generation, involving fictitious joints, are joined automatically by the tunnel generator. Joined blocks are shown in the same color on Figure 2-9.

This example contains 159 blocks and requires 800 kbytes memory on a personal computer. If a processor board is used in the computer (e.g., the DSI-780 -processor board with 4 mb memory), then up to approximately 1,000 blocks can be modeled. The run time for a model consisting of 1,000 blocks, using a DSI-780 board, is roughly 48 hours.

### 2.2.2 Calculation Cycle

3DEC is based upon a dynamic (time domain) algorithm which solves the equations of motion of the block system by an explicit finite difference method. A solution scheme based upon the equations of motion is demonstrated by Cundall [9] to be better suited to indicate potential failure modes of discontinuum systems than schemes which disregard velocities and inertial forces (e.g., successive over-relaxation). At each timestep, the law of motion and the constitutive equations are applied. In the case of rigid blocks, the latter consist only of the contact force-displacement relations. For deformable blocks, sub-contact force-displacement relations are prescribed. The integration of the law of motion provides the new block positions and, therefore, the contact-displacement increments (or velocities). The contact (or sub-contact) force-displacement law is then used to obtain the new contact (or sub-contact) forces, which are to be applied to the

blocks in the next timestep. The cycle of mechanical calculations is illustrated below; the components of the cycle are described in the following sections.



**Contact Force Update** — Section 2.1 described the procedure for updating the geometric parameters associated with contact between blocks. The unit normal to the common-plane is taken as the contact normal,  $n_i$  (pointing from block A to block B.) The reference point on the common-plane,  $C_i$ , defines the line of action of the contact force and is taken as the contact location.

For rigid blocks, the contact velocity (defined as the velocity of block B relative to block A at the contact location) is calculated as:

$$v_i = \dot{x}_i^B + e_{ijk} \omega_j^B (C_k - B_k) - \dot{x}_i^A - e_{ijk} \omega_j^A (C_k - A_k) \quad (2-15)$$

where  $A_i$  and  $B_i$  are the position vectors of the centroids of blocks A and B,

$\dot{x}_i^A$  and  $\dot{x}_i^B$  are translation velocity vectors of centroids of blocks A and B,

$\omega_i^A$  and  $\omega_i^B$  are the corresponding angular velocity vectors,

$e_{ijk}$  is the permutation tensor, and

indices  $i,j,k$  take the values 1 to 3, and denote components of a vector or tensor in the global coordinate system (The summation convention applies for repeated indices.).

The contact displacement increment vector is calculated as:

$$\Delta U_i = v_i \Delta t \quad (2-16)$$

which can be resolved into normal and shear components along the common-plane. The normal displacement increment is then given by

$$\Delta U^n = \Delta U_i n_i \quad (2-17)$$

and the shear displacement increment vector by

$$\Delta U^s_i = \Delta U_i - \Delta U_j n_i n_j \quad (2-18)$$

The unit normal to the common-plane,  $n_i$ , is updated every timestep. In order to account for the incremental rotation of the common-plane, the vector representing the existing shear force  $F^s_i$  (in global coordinates) must be corrected as:

$$F^s_i := F^s_i - e_{ijk} e_{kmn} F^s_j n^{old}_m n_n \quad (2-19)$$

where  $n^{old}_m$  is the old unit normal to the common-plane.

The contact displacement increments are used to calculate the elastic force increments. The normal force increment, taking compressive force as positive, is

$$\Delta F^n = -K_n \Delta U^n A_c \quad (2-20)$$

and the shear force vector increment is

$$\Delta F^s_i = -K_s \Delta U^s_i A_c \quad (2-21)$$

where  $K_n$  and  $K_s$  are the joint stiffnesses [stress/displacement], and  $A_c$  is the contact area.

Contact stresses are then calculated directly from the contact forces and area.

The calculation of the contact area for face-to-face contacts is straightforward, as noted in Section 2.1, and involves only the calculation of the common area of contact which is a convex, simply-connected polygon. For contacts other than face-to-face the area of contact is more difficult to define. At present a minimum value of the contact area is assumed in Eq. (2-20) and (2-21) to ensure a lower bound value for contact stiffness. The minimum area is calculated as a small fraction (e.g. 5%) of the average face area, but can be redefined by the user.

The total normal force and shear force vectors are updated

$$F^n := F^n + \Delta F^n \quad (2-22)$$

$$F^s_i := F^s_i + \Delta F^s_i \quad (2-23)$$

and adjusted according to the contact constitutive relations. At present, 3DEC employs a Coulomb friction law with a limited tensile strength. If the joint tensile strength,  $T$  is exceeded (i.e.,  $F^n < -TA_c$ ), then both shear and normal forces are set to zero. Otherwise, the maximum shear force is calculated as

$$F_{\max}^S = cA_c + F^n \tan\phi \quad (2-24)$$

where  $c$  is the joint cohesion [stress], and  $\phi$  is the friction angle.

If the absolute value of the shear force,  $F^S$ , given by

$$F^S = (F_i^S F_i^S)^{1/2} \quad (2-25)$$

is greater than  $F_{\max}^S$  (i.e., if  $F^S > F_{\max}^S$ ), then the shear force is reduced to the limiting value—i.e.,

$$F_i^S := F_i^S \frac{F_{\max}^S}{F^S} \quad (2-26)$$

The contact forces are then added to the forces and moments acting on the centroids of both blocks. The contact force vector, which represents the action of block A on block B, is given by

$$F_i = - (F^n n_i + F_i^S) \quad (2-27)$$

The force and moment sums of block A are therefore updated as

$$F_i^A := F_i^A - F_i \quad (2-28)$$

$$M_i^A := M_i^A - e_{ijk}(C_j - A_j)F_k \quad (2-29)$$

Similarly, for block B,

$$F_i^B := F_i^B + F_i \quad (2-30)$$

$$M_i^B := M_i^B + e_{ijk}(C_j - B_j)F_k \quad (2-31)$$

**Sub-Contact Force Update** — The location and orientation of the c-p are updated every cycle. The unit normal of the c-p,  $n_i$ , defines the contact normal, and is assumed to be the same for all sub-contacts.

As deformable blocks are discretized into tetrahedral zones, their boundary is composed of triangular faces. Gridpoints placed inside the original rigid block faces are thus treated as regular vertices. A sub-contact is established between one vertex of one block and a triangular face of the other block.

The relative velocity across a sub-contact is obtained from the velocity of the vertex,  $V_i^V$ , and the velocity of the corresponding point on the opposing face,  $V_i^F$ .

The velocity  $V_i^F$  can be calculated by linear interpolation of the velocities of the three vertices of the face—i.e.,

$$V_i^F = W_A V_i^{A_1} + W_B V_i^{B_1} + W_C V_i^{C_1} \quad (2-32)$$

The weighting factors can be calculated by transforming the coordinates of the vertices A, B and C into a local reference system, with one axis normal to the face plane. Denoting by  $X^A$  and  $Y^A$  the local in-plane coordinates of vertex A, the weighting factor  $W_A$  is given by:

$$W_A = \frac{y^C x^B - y^B x^C}{(x^A - x^C)(y^B - y^C) - (y^A - y^C)(x^B - x^C)} \quad (2-33)$$

The other two factors can be obtained by circular permutation of the superscripts A, B and C.

The unit normal  $n_i$  points from block 1 to block 2. Therefore, the relative velocity is calculated as:

$$V_i = V_i^V - V_i^F \quad (2-34)$$

when vertex V belongs to block 1 and, otherwise, as

$$V_i = V_i^F - V_i^V \quad (2-35)$$

The increment in relative displacement at the sub-contact is given by:

$$\Delta U_{\mathbf{i}} = \mathbf{v}_{\mathbf{i}} \Delta t \quad (2-36)$$

The normal displacement increment is:

$$\Delta U^n = \Delta U_{\mathbf{i}} n_{\mathbf{i}} \quad (2-37)$$

and the shear displacement increment vector (expressed in global coordinates) is:

$$\Delta U^{\mathbf{s}_{\mathbf{i}}} = \Delta U_{\mathbf{i}} - \Delta U_{\mathbf{j}} n_{\mathbf{i}} n_{\mathbf{j}} \quad (2-38)$$

The elastic force increments can then be calculated by using the contact (areal) normal and shear stiffnesses  $K_n$  and  $K_s$  and the sub-contact area  $A_c$ .

The normal force increment, taking compressive force as positive, is:

$$\Delta F^n = -K_n \Delta U^n A_c \quad (2-39)$$

and the shear force increment vector is:

$$\Delta F^{\mathbf{s}_{\mathbf{i}}} = -K_s \Delta U^{\mathbf{s}_{\mathbf{i}}} A_c \quad (2-40)$$

The sub-contact area is obtained by assigning to the vertex a region formed by 1/3 of the areas of the triangular faces containing the vertex and lying on the c-p. The area of the intersection of this region with the other block's faces lying on the c-p is then calculated.

$A_c$  is taken as one-half of this area, in order to account for the fact that the sub-contacts are established for vertices of both blocks—therefore resulting in two sets of parallel "springs".

Total subcontact forces are updated as:

$$\mathbf{F}^n := \mathbf{F}^n + \Delta \mathbf{F}^n \quad (2-41)$$

$$\mathbf{F}^{s_i} := \mathbf{F}^{s_i} + \Delta \mathbf{F}^{s_i} \quad (2-42)$$

and then corrected by application of the assumed constitutive relation, such as the Coulomb slip criterion described below.

The sub-contact force vector, which represents the action of block 1 on block 2, is given by:

$$\mathbf{F}_i = -(\mathbf{F}^n n_i + \mathbf{F}^{s_i}) \quad (2-43)$$

This vector and its negative are applied to blocks 2 and 1, respectively.

On the vertex side of the sub-contact, this force is added to the other gridpoint forces. On the face side, the force is distributed among the three vertices (A, B, C), using the interpolation factors defined above—i.e.,

$$\mathbf{F}^A_i := \mathbf{F}^A_i \pm \mathbf{F}_i W_A$$

$$\mathbf{F}^B_i := \mathbf{F}^B_i \pm \mathbf{F}_i W_B \quad (2-44)$$

$$\mathbf{F}^C_i := \mathbf{F}^C_i \pm \mathbf{F}_i W_C$$

The joint constitutive model incorporated in 3DEC for deformable blocks is the generalization of the Coulomb friction law. In a similar fashion as for rigid blocks, both shear and tensile failure are considered, and joint dilation is included.

In the elastic range, the behavior is governed by the joint normal and shear stiffnesses  $K_n$  and  $K_s$ , as described above by Eqs. (2-39) and (2-40).

For an intact joint (i.e., without previous slip or separation), the tensile normal force is limited to:

$$T_{\max} = - TA_C \quad (2-45)$$

where  $T$  is the joint tensile strength.

The maximum shear force allowed is given by:

$$F^s_{\max} = cA_C + F^n \tan\phi \quad (2-46)$$

where  $c$  and  $\phi$  are the joint cohesion and friction angle.

Once the onset of failure is identified at the sub-contact, in either tension or shear, the tensile strength and cohesion are taken as zero—i.e.,

$$T_{\max} = 0 \quad (2-47)$$

$$F^s_{\max} = F^n \tan\phi \quad (2-48)$$

This instantaneous loss in strength approximates the "displacement-weakening" behavior of a joint. The new contact forces are corrected in the following manner (note that normal compressive force is positive):

for tensile failure:

$$\text{If } F^n < T_{\max}, \text{ then } F^n = 0 \text{ and } F^s_i = 0 \quad (2-49)$$

for shear failure:

$$\text{If } F^s > F^s_{\max}, \text{ then } F^s_i := \frac{F^s_{\max}}{F^s} \quad (2-50)$$

where the shear force magnitude,  $F^S$ , is given by

$$F^S = (F_i^S F_i^S)^{1/2} \quad (2-51)$$

Dilation takes place only when the joint is at slip. The shear increment magnitude,  $\Delta U^S$ , is given by:

$$\Delta U^S = (\Delta U_i^S \Delta U_i^S)^{1/2} \quad (2-52)$$

This displacement leads to a dilation of:

$$\Delta U^n(\text{dil}) = \Delta U^S \tan \psi \quad (2-53)$$

where  $\psi$  is the dilation angle.

The normal force must be corrected to account for the effect of dilation—i.e.,

$$F^n := F^n + K_n A_C \Delta U^S \tan \psi \quad (2-54)$$

Real joints display a reduction in the dilation angle as the residual friction state is approached. In 3DEC, the joints can be prevented from dilating indefinitely by prescribing a limit shear displacement  $U_{lim}^S$ . When the magnitude of the shear displacement exceeds  $U_{lim}^S$ , the dilation angle is set to zero.

Block Motion Update — The equations of translational motion for a single block can be expressed as

$$\ddot{x}_i + \alpha \dot{x}_i = \frac{F_i}{m} + g_i \quad (2-55)$$

where  $\dot{x}_i$  = the velocity of the block centroid,

$\alpha$  = the viscous (mass-proportional) damping constant,

$F_i$  = sum of forces acting on block (from block contacts and applied external forces),

$m$  is the block mass, and

$g_i$  is the gravity acceleration vector.

The rotational motion of an undamped rigid body is described by Euler's equations, in which the motion is referred to the principal axes of inertia of the body:

$$\begin{aligned} I_1 \dot{\omega}_1 + (I_3 - I_2) \omega_3 \omega_2 &= M_1 \\ I_2 \dot{\omega}_2 + (I_1 - I_3) \omega_1 \omega_3 &= M_2 \\ I_3 \dot{\omega}_3 + (I_2 - I_1) \omega_2 \omega_1 &= M_3 \end{aligned} \quad (2-56)$$

where  $I_1, I_2, I_3$  are principal moments of inertia of the block,

$\omega_1, \omega_2, \omega_3$  are angular velocities about the principal axes, and

$M_1, M_2, M_3$  are components of torque applied to the block referred to principal axes.

An accurate dynamic analysis of a blocky system involves the solution of the above equations. However, dynamic analysis of rock mechanics systems requires the use of deformable blocks (Lemos, 1987). As mentioned previously, blocks can be discretized into a tetrahedral zone mesh, and the inertial terms are then consistently represented.

Rigid block models are more appropriate for quasi-static analyses and, in these cases, the rotational equations of motion can be simplified. Because velocities are small, the non-linear term in the above equations can be dropped, uncoupling the equations. Also, because the inertial forces are small compared with the total forces applied to the blocks, an accurate representation of the inertia tensor is not essential. In 3DEC, therefore, only an approximate moment of inertia,  $I$ , is calculated based upon the average distance from the centroid to vertices of the block. This allows the above equations to be referred to the global axes. Inserting a viscous damping term, the equations become:

$$\dot{\omega}_i + \alpha \omega_i = \frac{M_i}{I} \quad (2-57)$$

where the velocities  $\omega_i$  and the total torque  $M_i$  are now referred to the global axis.

A centered finite-difference procedure is used to integrate the equations of motion. The following expressions describe the translational and rotational velocities at time  $t$  in terms of the values at mid-intervals.

$$\dot{x}_i(t) = \frac{1}{2} \left[ \dot{x}_i\left(t - \frac{\Delta t}{2}\right) + \dot{x}_i\left(t + \frac{\Delta t}{2}\right) \right] \quad (2-58)$$

$$\omega_i(t) = \frac{1}{2} \left[ \omega_i\left(t - \frac{\Delta t}{2}\right) + \omega_i\left(t + \frac{\Delta t}{2}\right) \right]$$

The accelerations are calculated as

$$\ddot{x}_i(t) = \frac{1}{\Delta t} \left[ \dot{x}_i\left(t + \frac{\Delta t}{2}\right) - \dot{x}_i\left(t - \frac{\Delta t}{2}\right) \right] \quad (2-59)$$

$$\ddot{\omega}_i(t) = \frac{1}{\Delta t} \left[ \dot{\omega}_i\left(t + \frac{\Delta t}{2}\right) - \dot{\omega}_i\left(t - \frac{\Delta t}{2}\right) \right]$$

Inserting these expressions in the equations of translational and rotational motion, Eqs. (2-55) and (2-57), respectively, and solving for the velocities at time  $[t + (\Delta t/2)]$  results in

$$\dot{x}_i\left(t + \frac{\Delta t}{2}\right) = \left[ D_1 \dot{x}_i\left(t - \frac{\Delta t}{2}\right) + \left[ \frac{F_i(t)}{m} + g_i \right] \Delta t \right] D_2 \quad (2-60)$$

$$\dot{\omega}_i\left(t + \frac{\Delta t}{2}\right) = \left[ D_1 \dot{\omega}_i\left(t - \frac{\Delta t}{2}\right) + \left[ \frac{M_i(t)}{m} \Delta t \right] \right] D_2$$

where  $D_1 = 1 - (\alpha \Delta t/2)$ , and

$D_2 = 1/[1 + (\alpha \Delta t/2)]$ .

The increments of translation and rotation are given by

$$\Delta x_i = \dot{x}_i [t + (\Delta t/2)] \Delta t \quad (2-61)$$

$$\Delta \theta_i = \omega_i [t + (\Delta t/2)] \Delta t$$

The position of the block centroid is updated as

$$x_i(t + \Delta t) = x_i(t) + \Delta x_i \quad (2-62)$$

The new locations of the block vertices are given by

$$x^v_i(t + \Delta t) = x^v_i(t) + \Delta x_i + e_{ijk} \Delta \theta_j [x^v_k(t) - x_k(t)] \quad (2-63)$$

For groups of joined blocks, the motion calculations are only performed for the master block, whose mass, moment of inertia and centroid position are modified to represent the group of blocks. Once the motion of the master block is determined, the new position of the centroid and vertices of the slave blocks are calculated by expressions similar to Eq. (2-63).

The force and moment sums,  $F_i$  and  $M_i$ , for all blocks are reset to zero every cycle after the block motion update is completed.

**Numerical Stability** — The central difference algorithm is only numerically stable if the timestep  $\Delta t$  is less than the critical timestep. An estimate of the critical timestep is calculated in the program by analogy to a single degree-of-freedom system. The smallest block mass in the problem,  $M_{min}$ , and the largest normal or shear contact stiffness,  $K_{max}$ , are used to calculate the timestep as

$$\Delta t = \text{FRAC} \cdot 2 \left[ \frac{M_{min}}{2K_{max}} \right]^{1/2} \quad (2-64)$$

FRAC is a user-defined factor which accounts for the fact that a block may be in contact with several blocks. A value of FRAC equal to 0.1 is usually sufficient to ensure numerical stability.

For quasi-static processes, inertial forces usually have no effect on the solution as long as they remain small compared with the other forces in the system. Therefore, inertial masses can be scaled to improve the convergence time (gravitational masses are not affected). A procedure that is effective in many cases is to assign the same mass to all blocks.

**Damping** — For quasi-static analysis viscous damping is commonly used in the equations of motion to achieve convergence to a steady-state solution—i.e., either equilibrium or steady-state failure (collapse). Viscous damping introduces body forces that retard steady-state collapse and, in extreme cases may influence the mode of failure. Cundall (1982) and (1987) describe the use of adaptive damping as an effective method to overcome this difficulty. This approach is used in 3DEC. Adaptive damping continuously adjusts the viscosity such that the power absorbed by damping is a constant proportion of the rate of change of kinetic energy in the system. Therefore, as the kinetic energy approaches zero, the damping power also tends to zero.

### 2.3 References

Burman, B. C. "A Numerical Approach to the Mechanics of Discontinua," Ph.D Thesis, James Cook University of N. Queensland, Australia, 1971.

Chappell, B. A. "Load Distribution and Redistribution in Discontinua," *Int. J. Rock. Mech. Min. Sci. & Geomech. Abstr.*, 16, 391-399 (1979).

Cundall, P. A. "Adaptive Density-Scaling for Time-Explicit Calculations," Proceedings of the 4th International Conference on Numerical Methods in Geomechanics (Edmonton, 1982), pp. 23-26.

Cundall P. A. "UDEC - A Generalized Distinct Element Program for Modelling Jointed Rock," Report PCAR-1-80, Peter Cundall Associates Report, European Research Office, U.S. Army, Contract DAJA37-79-C-0548, 1980.

Cundall P.A., and O.D.L. Strack. "The Distinct Element Method As a Tool for Research in Granular Media, Part I," Department of Civil and Mineral Engineering, University of Minnesota Report to National Science Foundation, Grant ENG 76-20711, 1979.

Cundall P. A., and O.D.L. Strack. "A Discrete Numerical Model for Granular Assemblies," *Geotechnique*, 29, 47-65 (1979).

Cundall, Peter A., and Roger D. Hart. "Development of Generalized 2-D and 3-D Distinct Element Programs for Modeling Jointed Rock," Itasca Consulting Group; Misc. Paper SL-85-1, U.S. Army Corps of Engineers, 1985.

Goodman R. E., and G. Shi. Block Theory and Its Application to Rock Engineering. New Jersey: Prentice-Hall, 1985.

Key, S. W. "A Data Structure for Three-Dimensional Sliding Interfaces," International Conference on Computational Mechanics, Tokyo (1986).

Lemos, José. "A Distinct Element Model for Dynamic Analysis of Jointed Rock with Application to Dam Foundations and Fault Motion," Ph.D. Thesis, University of Minnesota, June 1987.

Rodriguez-Ortiz, J. M. "Estudia del Comportamiento de Medios Granulares Heteroteneos Mediante Modelos Discontinuos Analogicos y Matematicos," Ph.D. Thesis, Universidad Politecnica de Madrid, 1974.

### 3.0 INPUT INSTRUCTIONS AND COMMAND DESCRIPTIONS

Program 3DEC can receive input in three different ways: first, commands can be typed in and acted on immediately; second, a file of commands can be prepared and processed by the program without user intervention; and, third, the graphics objects on the screen can be manipulated directly by single keystrokes. The following section describes the commands that can be given interactively or from file--the format is the same in either case. The screen-oriented keystrokes are described in Section 3.2.

#### 3.1 Command-Mode Input

All input commands are word-oriented and consist of a primary command word followed by keywords and numerical input, as required. The commands, given on the following pages, are typed literally on the input line. You will note that only the first few letters are capitalized. The program requires only these letters to be typed for the command to be "recognized". Commands may be typed in upper case or lower case. Many of the keywords are followed by a series of numbers which provide the numeric input required by the keyword. Words that begin with a lower-case letter stand for numbers. Integers are expected when the word begins with i,j,k,l,m or n; otherwise, a real (or decimal) number is expected. The decimal point can be omitted from a real number but must not appear in an integer. The keywords and numbers may be separated by any number of spaces or by any of the following delimiters:

( ) , / =

< > denotes optional parameter(s)—the brackets are NOT to be typed.

... indicates that an arbitrary number of such parameters may be given.

Anything that follows an "\*" is taken to be a comment and is ignored. It is useful to make such comments in the input file since the comments are reproduced on the output.

The sign "&" at the end of a line denotes that the next line is a continuation of keywords or numeric input.

Geometric locations are prescribed for some commands by coordinate points (e.g., x1,y1,z1 x2,y2,z2) or by polyhedral regions or ranges (i.e., xl,xu yl,yu zl,zu). When ranges are specified [see, for example, CHANGE, the x-range delimiters (xl and xu), y-

range delimiters (yl and yu), or the z-range delimiters (zl and zu] should not be equal. For example, the command

**> CHANGE 10,10 1,10 1,10 MAT=2**

may not work because the x-range is not correctly specified. The range must encompass the points to be changed.

The following sign conventions are used in 3DEC and must be kept in mind when entering input. The sign is positive for:

- tensile stress
- extensional strain
- joint normal force (stress) in compression
- joint normal opening

The sign for a force or velocity boundary condition is determined by the direction of the force or velocity vector (i.e., + when pointing in the positive axis direction).

SI units are recommended for the numeric input; however, any consistent set of engineering units may be used. No conversions are performed in the program. Table 3-1 illustrates examples of consistent sets of units.

The commands to 3DEC are listed in two ways to assist the user in preparing code input. First, a summary of commands is given according to the modeling function performed and the recommended order for entering commands. This will help the user identify model conditions required for a specific analysis. Following this, full descriptions of all commands are given in alphabetical order.

Table 3-1  
SYSTEMS OF UNITS

	METRIC				BRITISH	
Length	m	m	m	cm	ft	in
Density	kg/m <sup>3</sup>	10 <sup>3</sup> kg/m <sup>3</sup>	10 <sup>6</sup> kg/m <sup>3</sup>	10 <sup>6</sup> g/cm <sup>3</sup>	slugs/ft <sup>3</sup>	snails/in <sup>3</sup>
Force	N	kN	MN	Mdynes	lb <sub>f</sub>	lb <sub>f</sub>
Stress	Pa	kPa	MPa	bar	lb <sub>f</sub> /ft <sup>2</sup>	psi
Gravity	m/sec <sup>2</sup>	m/sec <sup>2</sup>	m/sec <sup>2</sup>	cm/sec <sup>2</sup>	ft/sec <sup>2</sup>	in/sec <sup>2</sup>
Stiffness	Pa/m	KPa/m	MPa/m	bar/cm	lb <sub>f</sub> /ft <sup>3</sup>	lb/in <sup>3</sup>

where 1 bar = 10<sup>6</sup> dynes/cm<sup>2</sup> = 10<sup>5</sup> N/m<sup>2</sup> = 10<sup>5</sup> Pa

1 atm = 1.013 bars = 14.7 psi = 2116 lb<sub>f</sub>/ft<sup>2</sup> = 9.807x10<sup>4</sup> Pa

1 slug = 1 lb<sub>f</sub>-s<sup>2</sup>/ft = 14.59 kg

1 snail = 1 lb<sub>f</sub>-s<sup>2</sup>/in

1 gravity = 9.81 m/s<sup>2</sup> = 981 cm/s<sup>2</sup>

**3.1.1 General Command Sequence and Simple Command Description** — Commands may, in general, be given in any logical order. The following command sequence is recommended. More detailed descriptions of commands are given in the next subsection.

#### **I. Specify Program Control**

The user has options to:

- (1) **Begin a NEW problem without leaving 3DEC;**
- (2) **CALL a previously-prepared remote input command file to read into 3DEC;**
- (3) **RESTORE an existing saved state from a previously-executed problem;**
- (4) **RETURN program control from a remote input file to local mode;**
- (5) **QUIT program execution.**
- (6) **Interrupt program execution at any time during cycling. By pressing the <esc> key, the execution will stop so that the user can inspect the calculations. The user may save the state of the model at this point or continue program operation.**
- (7) **Run 3DEC on the DSI background loader. Before running 3DEC in the background, a data file called "3DEC.DAT" must be created in the current directory. Then, type RLOAD to install the loader, followed by RUN -B 3DEC to execute the run. A file called "OUT" contains the result of the run. The status of the run can be monitored by typing to the screen the file called "\$CONSOLE.OUT".**

## II. Input Problem Geometry

The general procedure to establish the problem geometry is to:

- (1) create single or multiple rigid block(s) defining the original boundary of the modeled region using one or several **POLYGON** commands;
- (2) create single or multiple joints in a block using the **JSET** command;
- (3) create boundaries of man-made structures (e.g., slopes, tunnels, stopes, etc.), either manually with the **JSET** command or automatically with the **TUNNEL** command; and
- (4) delineate regions in the model with the **MARK** command for future alteration (see Subsection VIII).

Additionally, split blocks can be rejoined using the **JOIN** command. All possible future blocks must be created during this step as, once execution begins, blocks can no longer be created. Once rigid blocks are zoned as fully-deformable blocks, they can no longer be split or joined. Joined, fully-deformable blocks can be un-joined.

## III. Specify Block Deformability

All blocks are rigid by default. The **GENERATE** command discretizes specified blocks into constant strain finite difference tetrahedral zones. Rigid and deformable blocks cannot be used in one model.

## IV. Assign Constitutive Relations and Properties

Block and joint constitutive relations are specified using the **CHANGE** command. All properties are designated using the **PROPERTY** command.

## V. Apply Boundary and Initial Conditions

General boundary conditions are prescribed using the **BOUNDARY** command. The following commands are used to specify initial conditions.

<b>FIX</b>	current velocities for specified rigid blocks are fixed
<b>FREE</b>	revokes the action of the <b>FIX</b> command
<b>GRAVITY</b>	sets gravitational acceleration in x-, y- and z-directions
<b>INSITU</b>	sets initial zone stresses in all fully-deformable blocks
<b>RESET</b>	resets certain variables to zero

## VI. Specify Conditions During Solution

The following commands are used to specify conditions during solution.

<b>DAMPING</b>	material damping (used to absorb kinetic energy)
<b>FRACTION</b>	fraction of critical timestep used in solution
<b>MSCALE</b>	sets zone and block masses for mass-scaling

In addition, it is often useful to monitor certain variables through time by using the **HISTORY** command.

## VII. Cycle the Problem

The **CYCLE** command is used to execute a given number of computation cycles.

## VIII. Perform Alterations

### A. Excavate/Fill Openings

Openings in the model can be excavated with the **EXCAVATE** command and backfilled with the **FILL** command. Individual blocks or block regions can be deleted with the **DELETE** command.

### B. Change Material Models, Properties, or Boundary Conditions

Block or joint material models can be changed at any time with the **CHANGE** command. Properties can be changed with the **PROPERTY** command, and boundary conditions can be changed with the **BOUNDARY** command.

### C. Specify Artificial Passive Support

Artificial passive support may be specified by the **STRUCT AXIAL** command, which generates local reinforcement elements across joints for modeling fully-bonded passive reinforcement in hard rocks.

Alternatively, **STRUCT CABLE** generates reinforcing elements across joints and within fully-deformable blocks for modeling fully-bonded passive reinforcement and explicitly including the shear behavior of the grout annulus.

The **STRUCT LINER** command generates a liner in a cylindrical opening in the model.

## IX. Output

The **PRINT** and **PLOT** commands may be utilized to examine the current problem state. The **PLOT** command allows the user to enter the screen-input mode, in which blocks appear in a perspective view and various quantities may be displaced. Other output commands are listed below.

<b>HEADING</b>	prescribes heading for problem
<b>SAVE</b>	saves the current problem state in a remote file

### 3.1.2 Full Description of Commands

**BOundary** <xl xu yl yu zl zu> keyword value . . .

or

**BOundary** <Vertex iv> keyword value . . .

Boundary conditions are specified over a range xl to xu, yl to yu and zl to zu or at a single vertex with number iv. If the range is omitted, the boundary condition applies to the entire boundary. Boundary conditions are defined by a given boundary condition keyword. The keywords and associated parameters are:

#### Force Boundary

<b>XLoad</b>	<b>fx</b>	<b>x-direction force (see Note 1)</b>
<b>YLoad</b>	<b>fy</b>	<b>y-direction force (see Note 1)</b>
<b>ZLoad</b>	<b>fz</b>	<b>z-direction fore (see Note 1)</b>

#### Stress Boundary

**STress** **sxxo syyo szzo sxyo sxzo syzo**

boundary stress parameters: **xx stress, yy stress, zz stress**  
**xy stress, xz stress, yz stress (see Note 1)**

An x,y,z coordinate region must be specified for the **STRESS** boundary condition (see Note 2).

**BOundary (continued)**

XGrad	sxxx syyx szzx sxyx sxzx syzx
YGrad	sxxy syyy szzy sxyy sxzy syzy
ZGrad	sxxz syyz szzz sxyz sxzz syzz

linearly-varying boundary stress where  $s_{xx0}$ ,  $s_{yy0}$ ,  $s_{zz0}$ ,  $s_{xy0}$ ,  $s_{xz0}$  and  $s_{yz0}$  are stresses at origin (0,0,0) defined by the STRESS keyword and where

$$s_{xx} = s_{xx0} + (s_{xxx} * x) + (s_{xxy} * y) + (s_{xxz} * z)$$

$$s_{yy} = s_{yy0} + (s_{yyx} * x) + (s_{yyy} * y) + (s_{yyz} * z)$$

$$s_{zz} = s_{zz0} + (s_{zzx} * x) + (s_{zzy} * y) + (s_{zzz} * z)$$

$$s_{xy} = s_{xy0} + (s_{xyx} * x) + (s_{xyy} * y) + (s_{xyz} * z)$$

$$s_{xz} = s_{xz0} + (s_{xzx} * x) + (s_{xzy} * y) + (s_{xzz} * z)$$

$$s_{yz} = s_{yz0} + (s_{yzx} * x) + (s_{yzy} * y) + (s_{yzz} * z)$$

(see Note 1)

XGRAD, YGRAD and ZGRAD keywords must follow the STRESS keyword on the same input line. (Use "&" at the end of line if a continuation line is required.)

An x,y,z coordinate region must be specified for the XGRAD, YGRAD and ZGRAD boundary conditions (see Note 2).

Example

```
BOUND -1,.1 -1,1 -1,1 STRESS -5,0,0,0,0 &
      YGRAD 5,0,0,0,0
```

This command applies a gradient to the xx-stress varying from -10 at  $y = -1$  to zero at  $y = 1$ .

**Boundary (continued)**

**Note 1:** All loads and stresses specified after the end of a **CYCLE** command are added into an incremental force to be applied during the next **CYCLE** command according to the type of history specified. At the end of this **CYCLE** command, the incremental vectors are added to the total vector and the incremental vector is set to zero. Therefore, for example, a change in a boundary stress condition must be input as a stress incremental change.

**Note 2:** Care must be taken when using the **STRESS**, **XGRAD**, **YGRAD** and **ZGRAD** commands to apply stresses only to the outer boundary of the model. If no coordinate region is given, stresses will be applied to boundaries of internal blocks as well as outer blocks.

**CAUTION:** The stress boundary condition affects all degrees of freedom. For example, if **BOUND STRESS** follows a **BOUND XVEL**, **YVEL** or **ZVEL** command affecting the same vertices, then the effect of the previously-described **XVEL**, **YVEL** or **ZVEL** will be lost.

**Velocity (Displacement) Boundary**

<b>XVEI</b>	<b>vx</b>	x-direction velocity (for FDEF blocks only)
<b>YVEI</b>	<b>vy</b>	y-direction velocity (for FDEF blocks only)
<b>ZVEI</b>	<b>vz</b>	z-direction velocity (for FDEF blocks only)

**Non-Reflecting (Viscous) Boundary**

Non-reflecting boundaries are available for dynamic analyses:

<b>Mat</b>	<b>n</b>	material number n assigned to far-field properties (required for non-reflecting boundaries)
<b>XVisc</b>		non-reflecting boundary in x-direction
<b>YVisc</b>		non-reflecting boundary in y-direction
<b>ZVisc</b>		non-reflecting boundary in z-direction

**Boundary (continued)****Free Boundary**

<b>XFree</b>	removes boundary condition applied in x-direction
<b>YFree</b>	removes boundary condition applied in y-direction
<b>ZFree</b>	removes boundary condition applied in z-direction

**Boundary Multiplier Histories**

User-supplied, multiplier histories may be stored by 3DEC for use later with history applicators (see below). The user-supplied history must contain three logical records of the following form:

- Line 1. heading of up to 80 characters
- Line 2. np, tdel
- np is the number of points (an integer value) and tdel is the timestep (a real value)
- Line 3. through Line np+2
- np real values of the history variable—i.e., hist(i), i=1,np. These values are assumed to be equally spaced at intervals of tdel.

The first value of the input history is assumed to correspond to time = 0. If a history is supposed to start from its beginning, but the current problem time is not zero, a **RESET TIME** command should be given before cycling.

The keyword and associated parameters for storing the input history are:

**HRead** hisnum filename

read and store history hisnum from file filename (hisnum = 1 or 2 only)

**BOundary (continued)****Boundary History Applicators**

Boundary forces, stresses, or velocities can be applied as a constant, linear, sinusoidal, or user-supplied history which remains in effect for one **CYCLE** command. The keywords are:

<b>XHistory</b>	<b>&lt;type&gt;</b>	<b>x-direction boundary parameters</b>
<b>YHistory</b>	<b>&lt;type&gt;</b>	<b>y-direction boundary parameters</b>
<b>ZHistory</b>	<b>&lt;type&gt;</b>	<b>z-direction boundary parameters</b>
<b>History</b>	<b>&lt;type&gt;</b>	<b>x-, y- and z-direction boundary parameters</b>

where types are:

<b>Constant</b>		<b>constant value (default)</b>
<b>Linear</b>		<b>load increment varies from 0 to 1 during the next cycle command</b>
<b>Sine</b>	<b>freq tload</b>	<b>sine wave load with frequency freq and applied for a period of tload (CAUTION: Period of tload must be shorter than time period of subsequent <b>CYCLE</b> command.)</b>
<b>Cosine</b>	<b>freq tload</b>	<b>cosine wave load with frequency freq and applied for a period of tload (CAUTION: Period of tload must be shorter than time period of subsequent <b>CYCLE</b> command.)</b>
<b>hisnum</b>		<b>multiplier history number hisnum (previously read into 3DEC with the <b>BOUNDARY HREAD</b> command) is applied (hisnum = 1 or 2 only)</b>

**CALL** <filename>

A remote input file, <filename> can be run with the **CALL** command. Any series of input instructions can be placed in this file to run in a remote or batch mode. The command **RETURN** must be inserted in the remote file to return input to the local mode. At present, the file must not contain a **CALL** command itself.

**CHange** <range> keyword value <keyword value> . . .

Block and joint material characteristics are prescribed and changed with the **CHANGE** command. All blocks with centroids lying within the optional coordinate range <range> have block material characteristics changed according to the keyword. The range, given in the order  $x_l$   $x_u$   $y_l$   $y_u$   $z_l$   $z_u$ , must be first on the input line following the command. If the range is omitted, all blocks have characteristics changed.

Block material characteristics in tunnel regions or **MARKED** regions can also be changed with the following keyword:

Region	n	Block characteristics are changed for blocks within region number n, where n is specified previously by the <b>TUNNEL</b> or <b>MARK</b> command.
--------	---	---

Characteristics of joints (i.e., contacts between blocks) are also prescribed and changed with the **CHANGE** command. Selected joints with contact coordinates lying within the optional coordinate range and/or the optional orientation range are changed. The orientation range is defined by the following keywords and parameters:

DD = a <tol>	dip-direction, in degrees, of joint(s) changed (A tolerance in dip-direction can be specified by the optional value <tol>; default = 2°.)
--------------	---

DIP = a <tol>	dip, in degrees, of joint(s) changed (A tolerance in dip angle can be specified by the optional value <tol>; default = 2°.)
---------------	---

ORG = x y z <tol>	origin (i.e., one location along joint) of joint changed (A tolerance in distance from the location can be specified by the optional value <tol>; default = 0.1.)
-------------------	---

**CHange (continued)**

Defaults for the above keywords, if not specified, are DD = 0°, DIP = 0°, ORG = 0,0,0. However, if none of the keywords are given, then all joints in the model will have characteristics changed. Any joint falling within the orientation and location specified by the above keywords will have contact material characteristics changed.

The following keywords are used to change characteristics:

**Block Characteristics**

- Cons = n                      Constitutive number n is assigned to designated blocks (default is n = 1; see note below)
- Mat = n                        Material property number n is assigned to designated blocks (default n=1).

**NOTE:** The present version of 3DEC has the following constitutive models for block deformability:

<u>CONS</u>	<u>Model Description</u>
1	linearly-elastic, isotropic (default)
2	elastic/plastic, Mohr-Coulomb failure*

---

\***WARNING:** This model should be used with caution. Accurate solutions to plasticity problems can only be achieved if all tetrahedral zones have a gridpoint at the centroid of the block and diagonally-opposed tetrahedra in the block.



**CYcle**      n

Execute n timesteps (cycle 0 is permitted as a check on data). Cycling can be interrupted at any time by pressing the <esc> key. Control will be returned to the user after the current cycle is completed.

**Damping**    Auto <fac mult1 mult2>  
              fcrit freq <Stiffness> <Mass>

Viscous damping is specified for the blocks. For static or steady-state problems, the objective is to absorb vibrational energy as rapidly as possible. In this case, the first form of the command should be used (AUTO keyword); this causes energy to be absorbed in proportion to the rate of change of kinetic energy. The optional parameter, fac, is the ratio of damping dissipation to kinetic energy change. If fac is not given, the default value of 0.5 is taken and gives a fast convergence in most cases. The multipliers, mult1 and mult2, adjust the damping coefficient by a fractional amount as the ratio changes. The default value of 1.05 for mult1 and 0.99 for mult2 are optimum in most cases.

The second form of the command is normally used for dynamic calculations when a certain fraction of critical damping is required over a given frequency range. This type of damping is known as Rayleigh damping, where fcrit = the fraction of critical damping operating at the center frequency of freq. (Note: Input frequencies for the program are in cycles/sec—not radians/sec.) The optional modifiers STIFFNESS and MASS denote that the damping is to be restricted to stiffness- or mass-proportional, respectively. If they are omitted, normal Rayleigh damping is used.

**NOTE:** Care is needed with stiffness-proportional damping because it can cause numerical instability if it is too high at the high eigenfrequencies. If 3DEC crashes with a floating point overflow, you should suspect this first and try the same run with only mass-proportional damping.

**DElete** <xl xu yl yu zl zu> <Vol vmax> <Region n>

or

**DElete** Block n

All blocks are deleted in the range  $x_l < x < x_u$ ,  $y_l < y < y_u$  and  $z_l < z < z_u$  with volumes smaller than  $v_{max}$ . If  $v_{max}$  is not specified all blocks are deleted in the range. Blocks in region  $n$  (defined by the **MARK** or **TUNNEL** commands) can also be deleted. Blocks can also be deleted individually by block number  $n$ . Type **PRINT BLOCK** for list of block numbers.

**Excavate**    **Region n**

All blocks within region n (specified by the **MARK** or **TUNNEL** commands) are excavated (i.e., removed for calculation purposes). The blocks can be replaced later with the **FILL** command. This command can only be used with **FDEF** blocks.

**FILL**      Region n <keyword...>

All excavated blocks within region n (specified by the **MARK** or **TUNNEL** commands) are filled (i.e, restored for calculation purposes). The blocks and joints within region n can be assigned material characteristics with the following optional keywords.

**Cons = n**              Constitutive number n is assigned to designated blocks.

**Mat = n**                Material property number n is assigned to designated blocks.

**JCons = n**             Constitutive number n is assigned to contacts between blocks in the filled region.

**JMat = n**              Material number n is assigned to contacts between blocks in the filled region.

The defaults for the material and constitutive number are 1 if the keyword is not specified.

**Fix** $x_l$   $x_u$   $y_l$   $y_u$   $z_l$   $z_u$ 

All blocks with centroids in the range  $x_l < x < x_u$ ,  $y_l < y < y_u$ , and  $z_l < z < z_u$  have current velocities fixed. Joined blocks must be unjoined (using **JOIN OFF n**) before fixing.

**FRAction** fb <fz>

fb is taken as the fraction of critical timestep to be used for block timestep (default fb=0.1). fz is the fraction of critical timestep to be used for zone timestep (default fz = 1.0).

**FREE****xl xu yl yu zl zu**

All blocks with centroids in the range  $x_l < x < x_u$ ,  $y_l < y < y_u$ , and  $z_l < z < z_u$  are set free. By default, all blocks are free initially.

**GEnerate** <x1 xu y1 yu z1 zu> <Fix x1 y1 z1 Fix x2 y2 z2 ...> Edge edavg

All blocks with centroids in the range  $x_1 < x < x_u$ ,  $y_1 < y < y_u$  and  $z_1 < z < z_u$  are discretized as fully-deformable. The parameter edavg must be given to define the average edge length of the tetrahedral zones.

The optional keyword FIX allows the user to define the location of a single vertex at position  $x_1, y_1, z_1$ . This command is used when a boundary condition is to be specified at a specific location (either interval or external).

**GRavity** gx gy gz

Gravitational accelerations are set for the x-, y- and z-directions.

**HEading** The next input line is taken as a heading printed on subsequent restart files and plot copies.

**Hide**

<xl xu yl yu zl zu> <keyword>

All blocks with centroids in the range  $x_l < x < x_u$ ,  $y_l < y < y_u$ , and  $z_l < z < z_u$  are made invisible; they are put on a stack and can be recalled later with the **SEEK** command or, in the screen-input mode, with the "f" key (see Section 3.2). When blocks are invisible, they are not split by the "k" key or the **JSET** command; in this way, discontinuous joints may be made. However, the invisible blocks interact normally with other blocks and are remembered on restart. Only visible blocks may be deleted, by the "d" key or by command **DELETE**. Only visible blocks may have their material or constitutive numbers changed by the **CHANGE** command and region numbers assigned or changed by the **MARK** command.

The following optional keywords can also be used.

<b>Mat</b>	n	hides only blocks of a specified material type number n.
<b>Region</b>	n	hides blocks of a specified region number n. Region numbers are specified with the <b>TUNNEL</b> or <b>MARK</b> command.

**NOTE:** the command **HIDE REGION = 0** will hide all blocks except those in a tunnel or marked region.

**CAUTION!** All blocks cannot be made invisible; the last block remains.

**HIStory** keyword <keyword>

A time history is kept of selected variables. Variables are accessed by their "history" number, which corresponds to the numerical position of the variable in the list of variables following the **HISTORY** command. The variable value is stored every **Ncyc** cycles. A time history of as many as 40 variables can be made in one run. The available keywords are:

<b>Address</b>	<b>i j</b>	any real variable with address <b>i</b> and offset <b>j</b> (see Section 5.0 for offsets)
<b>Ncyc</b>	<b>n</b>	histories are sampled every <b>n</b> cycles (default is <b>n = 10</b> )
<b>XDis</b>	<b>x y z</b>	x-displacement at location nearest to <b>x,y,z</b>
<b>XXStr</b>	<b>x y z</b>	xx-stress at location nearest to <b>x,y,z</b>
<b>XVel</b>	<b>x y z</b>	x-velocity at location nearest to <b>x,y,z</b>
<b>YDis</b>	<b>x y z</b>	y-displacement at location nearest to <b>x,y,z</b>
<b>YYStr</b>	<b>x y z</b>	yy-stress at location nearest to <b>x,y,z</b>
<b>YVel</b>	<b>x y z</b>	y-velocity at location nearest to <b>x,y,z</b>
<b>ZDis</b>	<b>x y z</b>	z-displacement at location nearest to <b>x,y,z</b>
<b>ZZStr</b>	<b>x y z</b>	zz-stress at location nearest to <b>x,y,z</b>
<b>ZVel</b>	<b>x y z</b>	z-velocity at location nearest to <b>x,y,z</b>

**HIS**tory      keyword

The following forms of the **HISTORY** command allow the user to write history number *nhis* (*nhis* = 1 to total number of histories) to the screen or to a disk file on the hard disk.

<b>Type</b>	<i>nhis</i>	displays the value of the variable on the console screen during cycling. (Only one variable can be selected at a time)
<b>Write</b>	<i>nhis fn</i>	The history (timestep number, history value) of history number <i>nhis</i> is written to a file " <i>fn</i> " on the hard disk. (" <i>fn</i> " is the file name defined by the user.) This file may be printed or manipulated after stopping 3DEC. Successive <b>HIS WRITE</b> commands will sequentially add to the history file. However, the first file written will overwrite any existing history file.

**Insitu**      **STress** **sxxo** **syyo** **szzo** **sxyo** **sxzo** **syzo**

Initial zone stresses in all FDEF blocks are set to **sxxo**, **syyo**, **szzo**, **sxyo**, **sxzo**, **syzo**.

An optional linearly-varying initial stress initialized at the origin (0,0,0) can be set by the following keywords and variables.

**XGrad**      **sxxx** **syyx** **szzx**   **sxyx** **sxzx** **syzx**  
**YGrad**      **sxyy** **syyy** **szyy**   **sxyy** **sxzy** **syzy**  
**ZGrad**      **sxxz** **syyz** **szzz**   **sxyz** **sxzz** **syzz**

where  $s_{xx} = s_{xxo} + (s_{xxx} * x) + (s_{xxy} * y) + (s_{xxz} * z)$   
 $s_{yy} = s_{yyo} + (s_{yyx} * x) + (s_{yyy} * y) + (s_{yyz} * z)$   
 $s_{zz} = s_{zxo} + (s_{zzx} * x) + (s_{zzy} * y) + (s_{zzz} * z)$   
 $s_{xy} = s_{xyo} + (s_{xyx} * x) + (s_{xyy} * y) + (s_{xyz} * z)$   
 $s_{xz} = s_{xzo} + (s_{xzx} * x) + (s_{xzy} * y) + (s_{xzz} * z)$   
 $s_{yz} = s_{yzo} + (s_{yzx} * x) + (s_{yzy} * y) + (s_{yzz} * z)$

**XGRAD**, **YGRAD** and **ZGRAD** keywords must follow the **STRESS** keyword on the same input line. (Use "&" at the end of the line if a continuation line is required.)

Example:

**INSITU**      **STRESS**      **-5,0,0,0,0,0 &**  
                 **YGRAD**      **5,0,0,0,0,0**

This command applies a gradient to the xx-stress varying from -5 at  $y=0$  to -55 at  $y = -10$ .

**ISet**

ival ia ioff

The integer value ival is inserted in the main array at the address ia, with offset ioff. See Section 5,0, Program Guide, for offsets.

**Join**

**ON** n1 n2  
**OFF** n

Adjacent blocks n1 and n2 are joined into one block. The block numbers n1 and n2 can be found using the **PRINT BLOCK** command. A block can be "unjoined" from surrounding blocks with the command **JOIN OFF n**, where n is the block number.

**Jset** <keyword = value> < ..... >

Generates one set of discontinuities, according to the parameters specified by the following keywords. Blocks are split completely (i.e., no partial cracks are allowed).

**DD = a <adev>** dip-direction, in degrees, of joints comprising the set given by angle a, where  $0 < a < 360^\circ$ , measured clockwise from the positive z axis, which is taken as North (see note below).

A random deviation, different for each block cut, is specified by adev.

**DIP = a <adev>** dip, in degrees, of the joints comprising the set (i.e., the angle of the joints below the  $y=0$  plane; see note below).

An optional random deviation, adev, specifies the limit of deviation; a different deviation is applied to each block split.

**Num = n** number of joints in the set. Joints are produced symmetrically about the joint-set origin, given by ORG (x,y,z).

**Org = x y z** origin, or starting-point, of the joint set. The first joint generated will pass through this point.

**Persistence = p** probability that any given block lying in the path of a joint will be split—i.e., if  $p=0.5$ , then 50% of the blocks will be split on average.

**Jset (continued)**

**Spacing = s <sdev>**            spacing between joints of the set. A random deviation (different for each block cut) is specified by sdev.

Defaults for the above parameters, if not given, are zero, except for PERS and NUM, which both default to 1.

**Join**                            joins blocks across the joints created by the JSET command.

**NOTE:** The reference axes for 3DEC are a left-handed set (x,y,z) which are oriented, for joint generation, x (east), y (vertically up) and z (north). If the problem orientation is different from these reference axes, the joint or joint set will have to be transformed from the problem reference frame to the model reference frame. See Section 4.3, Model Generation, for recommended transformation procedures.

**MArk** <range> <Block n> Region i

All blocks with centroids lying within the optional coordinate range <range> are marked. The range, given in the order  $x_l$   $x_u$   $y_l$   $y_u$   $z_l$   $z_u$ , must be first on the input line following the command. If the range is omitted, all blocks are marked. Individual blocks can be marked with the optional **BLOCK** keyword, where n is the block number. Type **PRINT BLOCKS** for a list of current block numbers.

Blocks are marked by a region number i. A marked block does not influence the calculations in any way, but serves to delimit a region for recognition by the **CHANGE**, **DELETE**, **EXCAVATE** and **FILL** commands. Note that blocks are also assigned region numbers by the **TUNNEL** command. Type **PRINT BLOCKS** for a list of current block region numbers.

**Mscale**      ON zcmass bcmass  
                 OFF

Automatic mass scaling for FDEF blocks is turned on to speed up convergence of static problems involving very non-uniform grids or very non-uniform elastic properties. The user may elect to set all zone masses to zcmass and rigid block masses to bcmass by specifying those parameters. Experience has shown that average zone and block masses generally lead to satisfactory solutions.

Values for minimum, average and maximum zone and block masses can be found using the **PRINT MAX** command. Mass scaling can be turned off at the user's discretion (and must be turned off for dynamic calculation).

**New**

**This command allows the user to begin a new problem without leaving 3DEC.**

**PLot**      <keyword> . . .

If no keyword is given, the program switches to screen-input mode, in which the blocks appear in a perspective view. Several quantities may also be displayed in this mode. See Section 3.2 for details.

By specifying the keyword, a specific plot can be displayed. The following keywords are presently available.

**Center =  $x_S y_S z_S$**

Coordinates associated with screen reference axes which can be used to shift the viewing plane of the model in the plane of the screen. For the screen reference frame, the  $x_S$ -axis lies in the plane of the screen and points to the right; the  $y_S$  axis lies in the plane of the screen and points upward; and the  $z_S$ -axis lies normal to the screen and points outward.

**Dip = a**      dip, in degrees, of a viewing plane through the model (i.e., the angle of the plane below the  $y=0$  plane,  $0 \leq a \leq 90^\circ$ ). The viewing plane is always oriented parallel with the terminal screen. The model is rotated by the dip angle DIP and dip direction DD in order to align the viewing plane with the screen (default is  $a = 0^\circ$ ).

**DD = a**      dip direction, in degrees, of a viewing plane through the model, given by the angle  $a$ , where  $0 \leq a \leq 360^\circ$  measured clockwise from the positive  $z$ -axis. The viewing plane is always oriented parallel with the terminal screen. The model is rotated by the dip angle DIP and dip direction DD in order to align the viewing plane with the screen (default is  $a = 180^\circ$ ).

**DScale**      v

sets the value of the maximum length of arrow to v (in problem units) for displacement vector plots.

## PLOT (continued)

**History**      n1 <n2...>

The time history of variables n1,n2... are plotted (where the variable numbers are set by the **HISTORY** command).

The following supplemental keywords may be used to modify history plots to achieve more meaningful presentation. The following supplements may be used after the **HISTORY** keyword.

**AB**    n      plots history n on abscissa axis (n assigned by **HISTORY** command) [Default for abscissa is problem time.].

**XRev**            reverses sign of history plotted to abscissa axis.

**YRev**            reverses sign of history plotted to ordinate axis.

**HP**              makes a hardcopy plot of the screen plot on a Hewlett-Packard (or compatible) pen plotter. This keyword only works for wireframe cross-section plots, at present.

**Joint**           plots a joint plane on the viewing plane. A joint plane must exist at the location defined by **DIP**, **DD** and **LOCATION** to plot a joint plane. This keyword must be last on the input line.

**PLot (continued)****Location =** x y z

location of one point in the model which is positioned to lie on the viewing plane. The model is translated by the magnitudes x, y and z (in problem units) from the model axes origin to position this point in the center of the viewing plane on the screen. If the **LOCATION** keyword is not specified, the model centroid is positioned at the center of the viewing plane.

**Magnify** n

magnify or diminish the size of the plot by factor n

**SScale** v

sets the value of the maximum length of stress magnitude to v (in problem units) for principal stress and planar traction plots.

**VScale** v

sets the value of the maximum length of arrow to v (in problem units) for velocity vector plots.

**Xsec** creates a two-dimensional cross-section through the model in the viewing plane defined by the keywords **DIP**, **DD** and **LOCATION**.

**Poly** <Material n> <Constitutive m> keyword ...

Create a single rigid polyhedron defining the original boundary of the modeled region. The polyhedron has a material number  $n$  and a constitutive number  $m$ ; defaults are  $n = 1$  and  $m = 1$  if these keywords are omitted. The polyhedron may be created in two ways, either by specifying the faces of the polyhedron with the **FACE** keyword or by defining a bounding region with the **REGION** keyword. More than one **POLY** command may be created per run, but all polyhedra must be convex.

The keyword and parameters for defining the polyhedron by faces are:

```
Face x1 y1 z1      x2 y2 z2      x3 y3 z3 ... &
Face x1 y1 z1      x2 y2 z2      x3 y3 z3 ... &
Face x1 y1 z1      x2 y2 z2      x3 y3 z3 ... & ...
```

The polyhedron is formed by planar faces with the **FACE** keyword where the coordinates  $(x_1, y_1, z_1)$ ,  $(x_2, y_2, z_2)$ , etc. define the boundary of the face. The vertices must be entered in counterclockwise order looking along the outward normal to the face.

Alternatively, a regular six-sided region can be created with the **REGION** keyword. The keyword and parameters to define a bounding region are:

```
Region xl xu yl yu zl zu
```

The region extends from coordinates  $x_l$  to  $x_u$ ,  $y_l$  to  $y_u$  and  $z_l$  to  $z_u$ .

**Print**      <range> keyword

Printed output is produced for all data prescribed by keywords. Printing of certain variables (designated by \*) can be limited by an optional coordinate range identified by <range>. The range, given in the order xl xu yl yu zl zu, must be first on the input line following the command. If the range is omitted, all data are printed.

Printed output for joint data keywords (designated by +) can be limited for selected joints within an optional joint orientation range <range>. The orientation range is defined by the following keywords and parameters:

**DD = a**      <tol>      dip-direction, in degrees, of joint data printed (A tolerance in dip-direction can be specified by the optional value <tol>; default = 2°.)

**DIP = a**      <tol>      dip, in degrees, of joint data printed (A tolerance in dip angle can be specified by the optional value <tol>; default = 2°.)

**ORG = x y z**  
                 <tol>      origin (i.e., one location along joint) of joint data printed (A tolerance in distance from the location can be specified by the optional value <tol>; default = 0.1.)

Defaults for the above keywords, if not specified, are DD = 0°, DIP = 0°, ORG = 0,0,0. If none of the keywords are given, then all joint data in the model are printed.

Printing can be interrupted at any time by pressing the <esc> key.

Data are printed for the following keywords:

<b>Axial</b>	axial reinforcement data
<b>BList</b>	block contact list data
<b>Blocks*</b>	block data (Hidden blocks are excluded.)

## PPrint (continued)

## BOundary\*

Forces  
Disp

force or displacement boundary conditions

Boundary code for x-, y- and z-constraints  
(ix, iy, iz) at vertices is:

0 free  
1 force (or stress)  
2 viscous  
3 velocity

## Cable

cable reinforcement data

## Contacts\*+

Forces  
Disp  
State

force, displacement, or state data for contacts

The state indicator code for contacts is as follows:

0 elastic  
1 at slip  
2 slipped in the past  
3 tensile failure

## CEll

cell data

## CP\*+

common-plane data (unit normal vector and  
reference point)

## Face\*

block face data

## PPrint (continued)

History	<n1 n2...>	time history of variables n1,n2.... If no number is specified, a list is given of history numbers and corresponding variable and coordinate location.
Lg		list of joined gridpoints
LIner		structural liner data
Maxima*		maxima and minima of block and zone data (Hidden blocks are excluded.)
Property		material properties
State		present state of global parameters
Velocity*		block velocity data (Hidden blocks are excluded.)
VERtex*		block vertex data (Hidden blocks are excluded.)
Zone*		block zone data (Hidden blocks are excluded.)

The state indicator code for zones is as follows.

0	elastic
1	at failure
2	failed in the past
3	tensile failure

**PROperty** Material n keyword v <keyword v . . .>

The first parameter, n, must be the specification of the material number. Block and joint properties are assigned to a material number by the **PROPERTY** command. Material numbers are assigned to blocks and joints in the model by the **CHANGE** command. Note that both block properties and joint properties can be assigned to the same material number. Material properties, v, are defined for material number n. All properties must be specified as a positive value. Property keywords for the available constitutive (see **CHANGE** command) models are:

JCONS = 1

Units

KN	joint normal stiffness	[stress/displacement]
KS	joint shear stiffness	[stress/displacement]
Cohesion	joint cohesion	[stress]
Dilation	joint dilation angle	[tangent of dilation angle]
Friction	joint friction (total friction, including dilatancy component)	[tangent of friction coefficient angle]
Tensile	joint tensile strength	[stress]
Zdilation	shear displacement for zero dilation	[displacement]

CONS = 1, 2

Units

Bulk (or K)	block bulk modulus	[stress]
G	block shear modulus	[stress]
Density	block density	[mass/volume]

**PROperty (continued)**

<u>CONS = 2</u>		<u>Units</u>
BCohesion	block cohesion	[stress]
BTension	block tensile strength	[stress]
Phi	block friction angle	[degrees]
PSi	block dilation angle	[degrees]

Quit

The run stops.

**RESet** keyword <keyword ...>

Certain variables are reset to zero according to the keyword:

- Disp** All displacements of gridpoints of FDEF blocks are set to zero, but gridpoint coordinates are not affected. This command does not alter the physics of the problem being modeled because displacements are not used in any calculations.
- Hist** All histories are cleared.
- JDis** All relative displacements (shear and normal) of contacts are set to zero. This has no effect on the model results and is useful for identifying effects of incremental changes.
- Time** Problem time is set to zero. This has no effect on the problem being modeled but is useful for input histories and for convenience in presenting results.
- JSTR** All joint stresses (shear and normal) are set to zero.
- MAT = n** Only blocks or contacts with material number n have displacements or stresses reset (must precede DISP, JDIS and JSTR keywords).

**Restore** <filename>

The saved state is restored using data from the named file (or else from the default file, if no file is given).

**REturn**

returns program control to the local mode.

**RSet**      **v ia ioff**

The real value **v** is inserted in the main array at address **ia** with offset **ioff**.  
See Section 5.0, Program Guide, for offsets.

**SAve**      <filename>

The current problem in memory is saved at its present state on the named file, <filename>. If the file already exists, it will be overwritten. If no filename is specified, the named file is "D3.SAV".

**SEek**      <n>

All invisible blocks (saved by the **HIDE** command or the "h" key) are restored (i.e., made visible). If the optional block number n is specified, only that block is made visible.

**SET**      <keyword>

This command sets global condition variables which remain constant during a run. The keywords for this command are:

**Log**            **ON**

opens a file named "3DEC.LOG" on the default disk drive. If a file "3DEC.LOG" already exists, it is overwritten. Any text which is printed to the screen from this point on is also written to the log file. This is particularly useful for keeping a record of interactive sessions. The file may be edited to create batch data files.

**OFF**

turns off the logging function. It does not close the log file "3DEC.LOG". If **SET LOG ON** is given at some later stage in the session, subsequent screen output will be appended to the file.

**Output**        **p**

p may be any logical device or filename. COM1, COM2 or LPT1 will send plot data out to external devices. (CON will send plot data to the screen.) p may be a disk filename. [NOTE: If no filename is specified, plot data will be sent to device COM1 (interactive mode) or file PLT (batch mode).]

**Xform**           **ON**

permits writing a formatted save file for transfer between versions of 3DEC installed on the DSI board and a mainframe computer.

**OFF**

writes an unformatted binary save file which cannot be transferred between computers but is smaller in size. (Default is XFORM=OFF.)

**STRUCT** keyword

The **STRUCT** command is used to define the geometry and properties for axial and cable reinforcing elements and structural lining elements. The keywords used to define the elements are as follows.

**Axial** Axial reinforcement with local normal restraint across discontinuities is provided. The geometry and property number for this reinforcement are defined by the values following the **AXIAL** keyword. The required input is in the form

**STRUCT AXIAL x1 y1 z1 x2 y2 z2 Prop n**

where x1,y1, etc. define the axial reinforcement endpoints.

When axial reinforcement is used with fully-deformable blocks, the **FDEF** blocks must be discretized before reinforcement locations are specified. Prop n defines the property number assigned to the reinforcement.

**Struct (continued)**

**Cable**            The cable element geometry, discretization, property number and pre-tensioning are defined by the values which follow the "CABLE" keyword. The required input is in the form

**STRUCT CABLE x1 y1 z1 x2 y2 z2 Seg=ns Prop=n <Tens=t>**

where x1,y1, etc. define the cable endpoints, Seg=ns is the number of segments into which the cable is divided, Prop=n is the property number assigned to the cable, and Tens=t is the optional cable pretension value.

**Liner**            The structural element liner, discretization, and property number are defined by the values which follow the LINER keyword. The required input is in the form

**STRUCT LINER x1 y1 z1 x2 y2 z2 Seg=nr,na Prop=n <Range=r>**

where x1, y1, etc. define the endpoints of the liner axis, nr is the number of radial segments into which the liner is divided, na is the number of annular segments into which the liner is divided, Prop=n is the property number assigned to the liner, and Range=r is an optional parameter to limit the search for possible liner/host medium contacts to a cylinder with radius r.

**STRUCT (continued)**

**Prop**            The axial, cable and liner element material properties and area are assigned using this keyword in the form

**STRUCT Prop=n keyword = value . . .**

This command associates property values to a particular property number n set by the AXIAL, CABLE or LINER keyword.

Axial reinforcement properties are assigned using the following keywords:

<b>RKax = value</b>	axial stiffness [force/displacement]
<b>RLen = value</b>	1/2 "active length" [length]
<b>RUlt = value</b>	ultimate axial capacity (must be greater than zero) [stress]

Cable properties are assigned using the following keywords. Figures 3-1 and 3-2 illustrate the relation of grout and cable material properties, respectively.

<b>E = value</b>	Young's modulus of cable
<b>Area = value</b>	cross-sectional area of cable
<b>Kbond = value</b>	bond stiffness of grout [force/unit cable length/displacement]
<b>SBond = value</b>	bond strength of grout [force/unit cable length]
<b>Yield = value</b>	yield strength (force) of cable

## SStruct (continued)

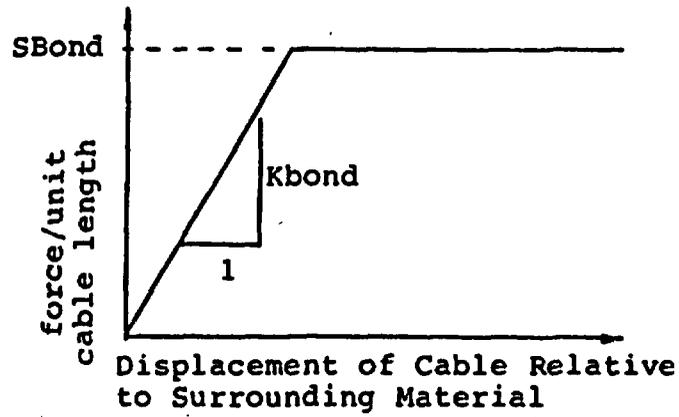


Fig 3-1 Grout Material Properties

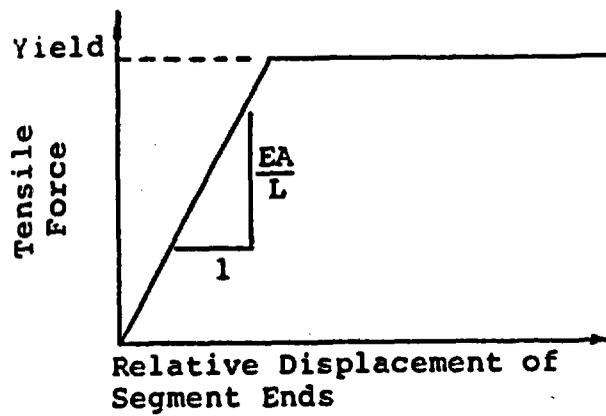


Fig. 3-2 Cable Material Properties  
 NOTE: Cable elements support tensile forces only.

**S**Struct (continued)

Liner properties are assigned using the following keywords:

<b>E = value</b>	Young's modulus of liner material
<b>NU = value</b>	Poisson's ratio of liner material
<b>Thick = value</b>	liner thickness
<b>KN = value</b>	normal stiffness [force/displacement] for contact between liner and host medium
<b>KS = value</b>	shear stiffness [force/displacement] for contact between liner and host medium
<b>TENS = value</b>	tension [force] limit for contact between liner and host medium
<b>COH = value</b>	cohesion [force] limit for contact between liner and host medium
<b>FRIC = value</b>	friction coefficient for contact between liner and host medium



**<esc>**

By pressing the **<esc>** key (without the ENTER key), the user can interrupt a run which is cycling. The run will stop and return to interactive mode for the user to enter new commands from the keyboard.

The **<esc>** key can be used to interrupt zone generation; the block currently being zoned will be made rigid again.

The **<esc>** key can also be used to interrupt the **PRINT** command.

### 3.2 Screen-Mode Input

When in screen mode, the following keystrokes may be made; it is not necessary to follow the keystroke with the ENTER key.

- a "axes" — x-,y-,z-coordinate axes are drawn to help the user orient the model. The axes can be removed by typing key "a" again.
  
- c "color" — The color mode for blocks can be changed. When the color key is pressed, the color mode menu appears and the following keys (and only these) operate:
  - b "block" — The blocks are plotted in different colors. Colors will change as the model is altered.
  
  - f "freeze" — The present block color state is frozen and will not change.
  
  - m "material" — The block colors are assigned according to material type specified for the block.
  
  - r "region" — The block colors are specified according to region number specified for the block.
  
- f "find" — The last (invisible) block saved is restored (i.e., made visible). If the "f" key is struck repeatedly all saved blocks will be recalled in the opposite sequence from which they were hidden.
  
- j "joint structure" — The joint structure in the model is highlighted (not operating at present).
  
- k "knife" — All visible blocks are split into two at the current location of the cut plane. A block is not split if the cut plane passes too close to a corner.

- l "liner/cable" — structural liner or cable information is displayed on wireframe plots. (The "l" key only operates after the "w" (wireframe) key is pressed.) When the liner key is pressed, the liner/cable menu appears and the following keys (and only these) operate:
  - a "axial" — Axial cable forces are plotted at center of each element.
  - c "cable" — Cable geometry is plotted.
  - l "liner" — Liner plate elements are plotted.
  
- m "magnify" — The size of the plot is magnified. Repeated typing of the "m" key increases the magnification.
  
- o "options" — Plotting options are available by typing the "o" key. When the option key is pressed, the option menu appears and the following keys (and only these) operate:
  - b "background color" — The background color may be switched from black to gray.
  - e "buffers equal" — The plots in the foreground buffer and background buffer are made equal.
  - f "freeze scale" — The present scale for vector plots will remain constant for all vector plots. The scale can be "unfrozen" by returning to the option menu and pressing the "f" key again.
  - n "N-E-U axis" — The axis labels are changed to North-East-Up.
  - p "plot border" — A title border replaces the menu for the purpose of making hard-copy screen dumps. When the border is visible, all screen-mode keys still operate. To return to the menu, type the "o" key followed by the "p" key.
  
- q "quit" — The program switches to command-mode input (see Section 1.1).

- r "run" — The problem will begin cycling from the screen mode. Press any key to stop the run.
- s "stresses" — Stresses are displayed on two-dimensional cross-sections through the model (the "s" key only operates after the "w" (wireframe) key and the "x" (cross-section) key are pressed.) When the "s" key is pressed, the stress menu appears and the following keys (and only these) operate:
- f "fill" — plot contours in filled color mode
  - p "principal stress" — plot of principal stress magnitudes and directions
  - t "planar traction" — plot of tractions acting on cross-section. Circles indicate relative magnitude of tractions, and arrow indicates direction and relative magnitude of shear component of traction. If only a dot shows in the center of the circle, the traction is entirely a normal stress; if the arrow extends to the full radius of the circle, the traction is entirely a shear stress.
- 1 minimum principal stress contours
  - 2 intermediate principal stress contours
  - 3 maximum principal stress contours
  - 4 xx-stress contours
  - 5 xy-stress contours
  - 6 xz-stress contours
  - 7 yy-stress contours
  - 8 yz-stress contours
  - 9 zz-stress contours

- t **"target"** — A target or cursor is displayed. It is removed when any other key is typed. When the target is visible, the following keys (and only these) operate:
  - c **"color"** — changes the color of the block selected by the cursor.
  - d **"delete"** — The block selected by the cursor is deleted.
  - f **"fix"** — The targeted block is fixed.
  - h **"hide"** — The block selected by the cursor is made invisible; it is put on a stack, and can be recalled later with the "f" key or **SEEK** command. When blocks are invisible, they are not split by the "k" key or the **JSET** command; in this way, discontinuous joints can be made. However, the invisible blocks interact normally with other blocks, and are remembered on re-start. Only visible blocks may be deleted, by the "d" key or by command **DELETE**. Only visible blocks are affected by the **CHANGE** command, and region numbers are assigned or changed by the **MARK** command. **CAUTION:** All blocks cannot be made invisible—the last block remains.
  - j **"join"** — Two adjacent blocks can be manually jointed into one block. The cursor must be positioned over the first block to be joined and the "j" key must be struck. The cursor will re-appear and must then be positioned over the second block and the "j" key hit again. Several blocks can be joined by repeating this feature.
- u **"un-magnify"** — The size of the plot is diminished (un-magnified). Repeated typing of the "u" key decreases the magnification.

- v** "vectors" — Vectors are displayed on wire-frame and cross-section plots. When the vector key is pressed, the vector menu appears and the following keys (and only these) operate:
- d** "displacement" — Displacement vectors are plotted at vertices.
  - n** "joint normal displacement" — Relative normal displacement vectors along joints are plotted at contacts.
  - s** "joint shear displacement" — Relative shear displacement vectors along joints are plotted at contacts.
  - v** "velocity" — Velocity vectors are plotted at vertices.
- w** "wireframe" — A wireframe plot of the model is displayed. It is removed by hitting the "w" key again.
- x** "x-section" — A two-dimensional cross-section is created through the model in the viewing plane.
- arrow keys** — The four arrow keys on the numeric keypad are used to move objects, change the view, move the cursor, and so on. The particular action of the keys depends on the mode that is currently set. There are 5 modes, set by typing the appropriate number keys (on the top row of the keyboard).
- 1** In mode 1, the up and down arrows move the eye position (i.e., the perspective view is changed).
  - 2** In mode 2, the four arrow keys cause the displayed block system to move to left or right or up and down.
  - 3** In mode 3, the up/down keys cause the block system to rotate about the model x-axis; the left/right keys cause a rotation about an axis pointing upward in the plane of the CRT screen.

arrow keys (continued)

- 4 In mode 4, the up/down keys cause the cut-plane to move nearer to or further from the screen.
  - 5 In mode 5, the up/down keys cause the block system to move nearer to or further from the screen; the left/right keys cause a rotation about an axis normal to the plane of the screen.
- + increases (by a factor of 5) the movement caused by the arrow keys (see below).
- decreases (by a factor of 5) the movement caused by the arrow keys.

## **4.0 PROBLEM SOLVING WITH 3DEC**

### **4.1 Introduction**

When using 3DEC for problem solving, the modeler must have a clear understanding of the capabilities and limitations of the code. This section provides information to help the modeler evaluate the suitability of 3DEC for a particular analysis. Following this, suggestions are given on a recommended approach for model generation, solution procedure and interpretation of results.

The following problem solving methodology for 3DEC is similar to that recommended for the Itasca codes FLAC and UDEC, and it is recommended that the modeler become proficient with these codes before using 3DEC.

### **4.2 Suitability of 3DEC**

3DEC is a complex code with the capability to model a three-dimensional physical system in great detail. However, there are limitations to the effectiveness of the code, and the modeler must be careful not to force the code beyond its capabilities.

3DEC is similar to other continuum stress analysis codes in many respects. A single 3DEC block can be treated as a 3-D continuum model with specified initial and boundary conditions, material constitutive relations, and properties. The internal discretization of the block into constant strain, finite difference, tetrahedral zones corresponds to zoning in a 3-D continuum code. 3DEC, in fact, can be used to perform a continuum analysis if so desired, although the code is not as efficient for continuum analysis as is a conventional continuum code.

3DEC is best suited for analysis of three-dimensional discontinuum systems with well-defined discontinuities and where the response of the system is predominantly controlled by the behavior of these discontinuities. The modeler should have a sufficient understanding of the anticipated system response in order to assess whether 3DEC is applicable. In many cases, this level of understanding may not be immediately available and preliminary idealized calculations may be warranted to improve this understanding.

It is recommended that some form of two-dimensional analysis (e.g., with FLAC and/or UDEC) always be performed before using 3DEC. A two-dimensional analysis can prove quite useful in defining the predominant mechanisms in the behavior of a system and also in evaluating the effectiveness of 3DEC to represent these mechanisms. Further-

more, 2-D models are more efficient to run than 3-D models and can thus investigate parameter sensitivity more readily. Consequently, model conditions for 3DEC, such as the size of the model and the level of detail required, can be developed on the basis of preliminary two-dimensional calculations.

For example, a fundamental problem in the analysis of a jointed rock mass is the difficulty with modeling every joint in the rock. Preliminary calculations with FLAC (treating the rock as a continuum) and UDEC (treating the rock as a discontinuum) can provide insight to the level of jointing detail required for the 3DEC analysis.

An important consideration when using 3DEC is the calculational run time required for problem solving. Studies with 3DEC are more restrictive than comparable studies with 2-D codes. For example, UDEC is approximately ten times faster than 3DEC for comparable analysis. The following procedure is recommended to estimate the required run time for 3DEC using the DSI-780 coprocessor board.

For static analysis, a loading alteration in 3DEC (e.g., due to change in the stress state or to an excavation step) typically requires approximately 1500-2000 calculation cycles to diminish inertial effects and bring the model to an equilibrium state. The computer run-time for an alteration step of 2000 cycles is approximately 160 secs per block for a rigid block model or 40 secs per gridpoint for a deformable block model. Thus, a model consisting of 100 rigid blocks requires approximately 4-1/2 hours run-time for one alteration step of 2000 cycles, and a model consisting of 1000 gridpoints requires approximately 11 hours for one step. The run time varies linearly with the cycling number. An effective way to judge when equilibrium or a steady-state condition has been achieved in a static analysis is to monitor changes in problem variables (i.e., stress, velocity, displacement) using the HISTORY command.

### 4.3 Model Generation

The reference axes for 3DEC are a left-handed set (x,y,z) oriented, by default, as x (east), y (vertically up), and z (north). In generating a 3DEC model of a given problem, the following problem conditions may require re-orientation for input into the model:

- (1) geometry of the problem structures (i.e., mine layout, tunnel locations, etc.)
- (2) geometry of the geologic features (e.g., faults and joint sets); and
- (3) orientation of the in-situ stress field.

In the ideal situation, the problem geometries would align with the 3DEC reference axes. In general, though, this is not the case, and one or two of the problem geometries will require transformation to the 3DEC model reference frame.

Typically, it is best to orient the 3DEC model axes to align with the geometry of the problem structures—e.g., the mine grid or centerline of a tunnel. For graphics presentation, it is best to position the origin of the model axes at the center of the structure for which the analysis is intended. In this case, the field principal stresses may require transformation for application to the model boundary, and the geologic features may require re-orientation from global to local problem axes.

A short program is provided to transform field stresses into a set of stress components referenced to the local problem axes defined for the 3DEC model. The orientation of the local (model) axes is defined by the dip and dip direction of the local z-axis. The local y-axis lies in the vertical plane containing the z-axis dip vector and the x-axis lies in the horizontal plane. The program, TRANS, calculates local stress components on the basis of the following input data.

field principal stress 1 ( $\sigma_1$ ): magnitude, dip and dip direction

field principal stress 2 ( $\sigma_2$ ): magnitude, dip and dip direction

field principal stress 3 ( $\sigma_3$ ): magnitude, dip and dip direction

local z-axis: dip and bearing

TRANS computes, first, a set of stress components referenced to a left-handed set X,Y,Z of global axes which are oriented X (north), Y (east) and Z (vertically up). Then, the set of stress components referenced to the model axes are calculated. The output stress components are recorded on a file named "TRANS.REC".

The following example illustrates the transformation of field stresses to boundary stresses for the 3DEC model. A tunnel ventilation raise is oriented with an axis dip of  $84^\circ$  and dip direction of  $125^\circ$ . The 3DEC model axes are oriented as shown in Fig. 4-1. The z-axis is directed down the raise, the y-axis lies in the vertical plane containing the raise dip vector, and the x-axis lies in the horizontal plane, directed N $35^\circ$ E. The origin of the model axes is located at the center of the raise.

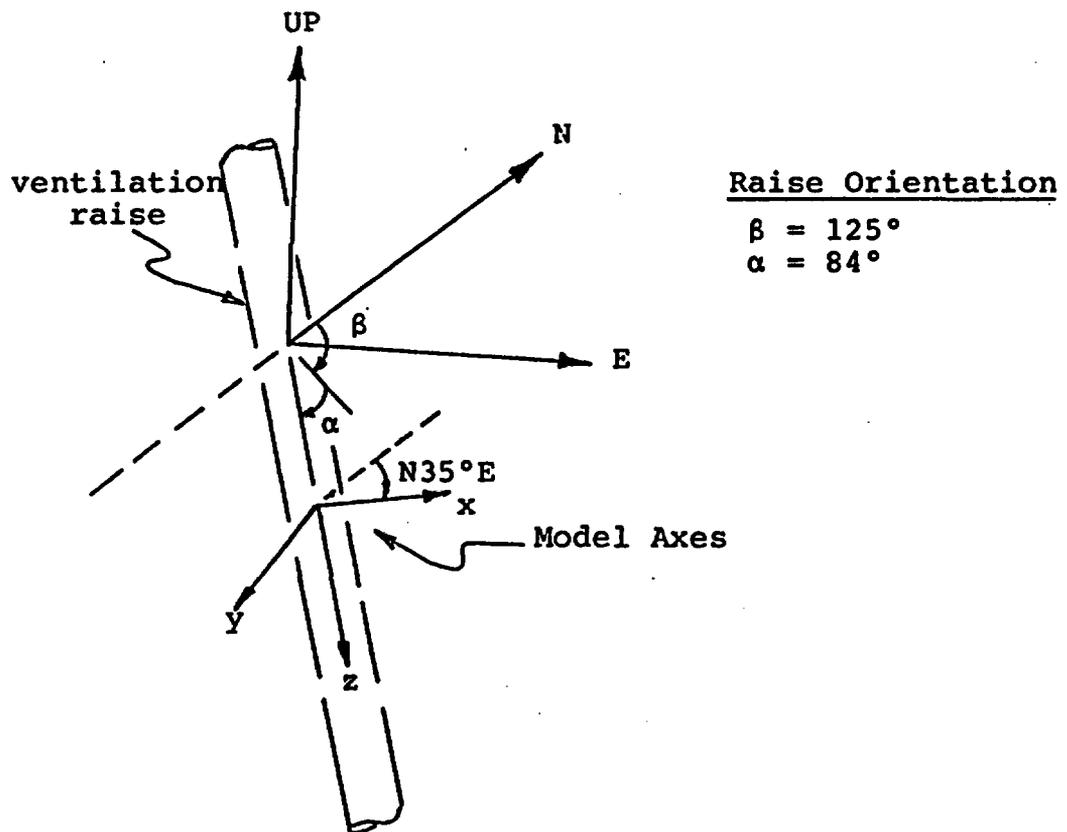


Fig. 4.1 Orientation of 3DEC Model Axes (x,y,z) Relative to North-East-Up Reference Axes

The field stresses for this problem are listed below.

$\sigma_1 = 30$  MPa directed  $24^\circ/231^\circ$  (dip/dip direction)

$\sigma_2 = 15$  MPa directed  $5^\circ/138^\circ$  (dip/dip direction)

$\sigma_3 = 12$  MPa directed  $66^\circ/36^\circ$  (dip/dip direction)

For a z-axis orientation of dip =  $84^\circ$  and dip direction =  $125^\circ$ , program TRANS computes the following stress components relative to the local axis (in 3DEC, tensile stresses are considered positive).

Stress Data for 3DEC (left-handed axes)

Field Stress:

Principal Stress 1    Magnitude -30.0    Dip 24.0    Bearing 231.0

Principal Stress 2    Magnitude -15.0    Dip 5.0    Bearing 138.0

Principal Stress 3    Magnitude -12.0    Dip 66.0    Bearing 36.0

Stresses Relative to Global Axes [X (north), Y (east), Z (vertically up)]:

FXX        -19.44            FYY        -22.47            FZZ        -15.09

FXY        -5.79            FYZ        -5.17            FZX        -4.38

Model z-axis Orientation:

Dip        84.0            Bearing     125.0

Stresses Relative to Model Axes:

SXX        -25.87            SYX        -16.38            SZZ        -14.74

SXY        4.07            SYZ        1.59            SXZ        -6.16

This information is contained in "TRANS.REC". The boundary stresses applied to the model are then

$$\begin{array}{lll} \sigma_{xx} = -25.87 \text{ MPa} & \sigma_{yy} = -16.38 \text{ MPa} & \sigma_{zz} = -14.74 \text{ MPa} \\ \sigma_{xy} = 4.07 \text{ MPa} & \sigma_{yz} = 1.59 \text{ MPa} & \sigma_{zx} = -6.16 \text{ MPa} \end{array}$$

Orientation of structural features in 3DEC must be specified with respect to the x,y,z reference axes of the model. Orientation is input as a dip direction and dip, with the x-z plane defining the reference horizon. The dip direction is measured clockwise from the positive z-axis in this plane, and dip is measured downward from this surface in a plane defined by the dip vector and the y-axis.

If the problem axes do not align with the model axes [i.e., positive z-axis (north), x-axis (east) and y-axis (up)], then the orientation of the structural features will have to be transformed from the problem axes to the model axes. This can be accomplished by making use of a stereonet.

The reorientation of structural features to the model reference axes is demonstrated for the raise example described previously. A fault which crosses the raise is oriented  $11^\circ/148^\circ$ . The dip and dip direction must be redefined relative to the x,y,z model axes as oriented in Fig. 4-1.

The fault pole is plotted on the lower hemisphere stereonet shown in Fig. 4-2. The relationship between the fault and the model axes is found by plotting the positive x-, y- and z-axes of the model on the stereonet. The dihedral angle between the fault pole and each axis is then read from the stereonet. In this example, the dihedral angles are  $86^\circ$  for x-axis to fault pole,  $73^\circ$  for y-axis to fault pole, and  $17^\circ$  for z-axis to fault pole.

On a second stereonet, Fig. 4-3, the model axes are oriented to align with the axes of the stereonet: z (north), x (east) and y (up). The intersection of the three dihedral angles on this plot gives the pole of the fault (plotted on the upper hemisphere). The orientation of the fault relative to the model axes is thus  $74^\circ/2^\circ$ .

4-7

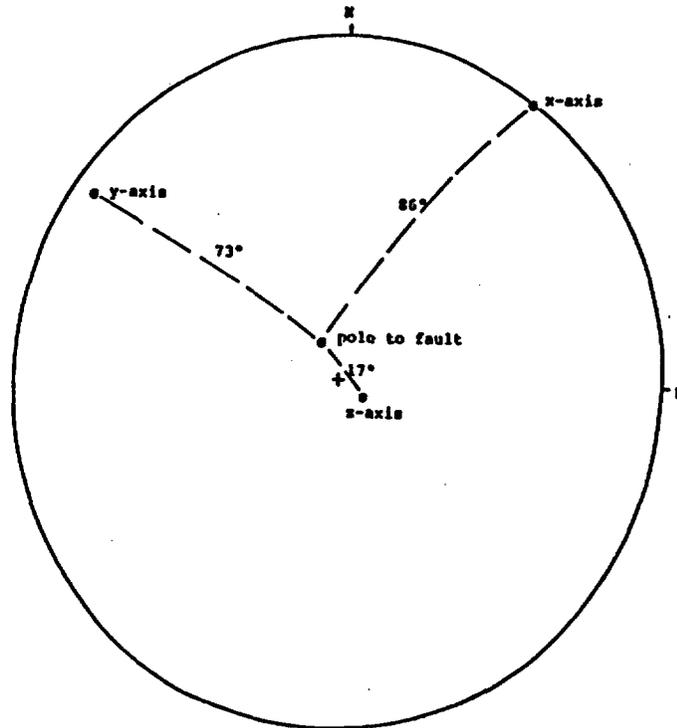


Fig. 4-2 Stereonet Plot of Pole to Fault and Model Reference Axes Relative to Problem North-East Axes

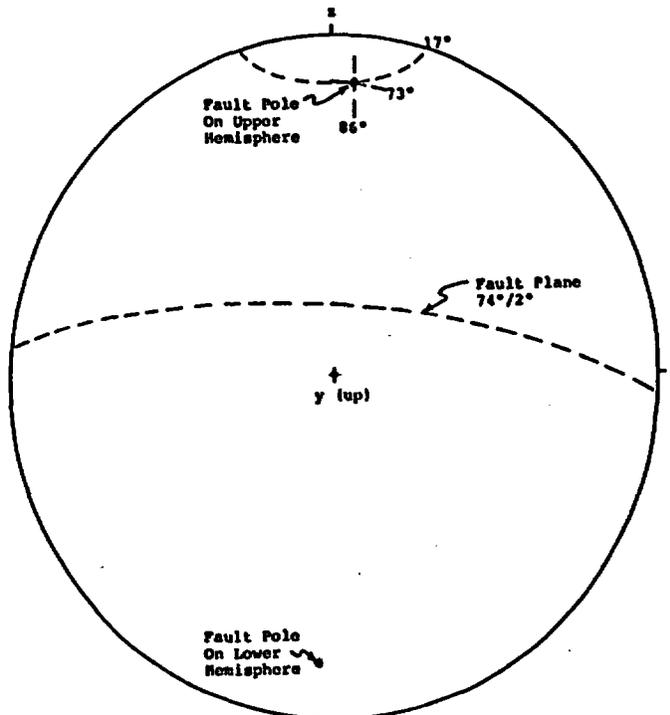


Fig. 4-3 Stereonet Plot of Fault Relative to Model Axes

#### 4.4 Solution Procedure

To run a simulation with 3DEC, the problem geometry, boundary conditions, and material constitutive relations and properties must first be specified. These procedures are similar to those of nearly all stress analysis codes. Additionally though, the region to be modeled must be subdivided into blocks defined by discontinuities. Blocks may be further discretized into constant strain finite difference tetrahedrons to model deformation of individual blocks. Each zone is defined by its four corner vertices, or gridpoints. Generation of zones is done automatically, with the size of zones specified by the user. Blocks which have been internally discretized are called fully-deformable blocks.

Once the problem geometry has been defined, boundary and initial conditions must be provided. The conditions can consist of:

- (1) fixing block centroids, gridpoint displacements, or velocities in the x-, y- and/or z-directions;
- (2) applying pressure on any outer boundary;
- (3) applying x-, y- and/or z-direction forces at any block centroid or boundary gridpoint;
- (4) initial stresses in the body; and
- (5) gravity.

The actual solution of a problem using an explicit code such as 3DEC differs from many of the commonplace implicit codes. Explicit codes are often termed "time-marching" in that the basic equations of motion are solved at successive timesteps until equilibrium or a steady-state condition is achieved. In practical terms, this means that, for the user, the problem is solved in a physically-meaningful manner.

The general solution procedure involves problem set-up (defining geometry, material constitutive relations, parameter properties, boundary conditions and in-situ stresses) and then the successive changing of conditions of the problem, such as excavating material, changing boundary conditions, etc. Following each problem step (changes to one or more conditions), the problem is cycled to equilibrium or a steady-state condition. This procedure is convenient because it physically represents the processes which occur in the natural environment. Modeling of a few simple problems, such as those contained in the manual, is suggested to familiarize the user with the solution process.

## 5.0 PROGRAM GUIDE

The program guide contains the complete contents of all the groups of variables in the data structure. The linkages of the various groups are described and shown schematically in Section 2.1. Variables in 3DEC are stored in "Linked Lists", which consist of addresses (the first memory location for a particular item—i.e., block or contact), and offsets, which prescribe the memory location relative to the address. Addresses can be found using the **PRINT** command. In most cases, the first integer given is the address. Commands in 3DEC allow commonly used real variables to be tracked (using the **HISTORY** command) or changed (using the **CHANGE** or **RESET** commands). However, it is sometimes desirable to track or change less commonly used variables. The commands **HISTORY ADDRESS**, **RSET** and **ISSET** are general purpose commands that allow any problem variable to be monitored or changed if its address and offset are known. The following sections list the offsets for various data.

## 5.1 Offsets for Main Data Arrays

### Offsets for Block Data Array

Each block data array consists of 42 words (NVBL=42). This array is accessed via the IBPNT pointer.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	--	block type number MRIG=1 rigid block MFDEF=2 fully-deformable block
1	KB	pointer to next block in block list (0 for end)
2	KF	pointer to one face in block's face list
3	KV	pointer to one vertex in block's vertex list
4	KZ	pointer to one zone in FDEF block's zone list
5	KC	pointer to one contact on block
6	KMAT	material number
7	KCONS	constitutive number
8	KBCOD	fixity condition: 0 free block 1 fixed block 2 slave block
9	KCEN	start of x,y,z coordinates of block centroid
12	KXD	start of x,y,z components of velocity

Offsets for Block Data Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
15	KTD	start of x,y,z components of angular velocity - (counterclockwise positive)
18	KVOL	block volume
19	KBM	block mass
20	KBI	start of x,y,z components of moment of inertia
23	KBFX	start of x,y,z components of block centroid force sum
26	KBFT	start of x,y,z components of block centroid mo- ment sum
29	KXL	start of x,y,z components of load applied to block centroid
32	KBDSF	block density-scaling factor
33	KCOLOR	color index used in display
34	KLINK1	spare offset
35	KLINK2	spare offset
36	KMOVE	accumulated displacement, used to trigger up- dates (contact detection and re-mapping into cells)
37	KBSEQ	spare offset
38	KBMSF	master/slave condition: MASTER=1 master block MSLAVE=2 slave block 0 normal block

Offsets for Block Data Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
39	KBMSLK	pointer to next block in master/slave list
40	KF2	pointer to joint face list for fdef blocks
41	KBREG	region number

Offsets for Face Data Array

Each face data array consists of 7 words (NVPO=7). This array is accessed from the block pointer KF.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	MFPOLY	=3, denotes face
1	KNFP	pointer to next face on host block (0 for end)
2	KNBP	pointer to host block
3	KFPVL	pointer to one vertex-list item on this face
4	KFPUN	start of x,y,z components of unit normal vector to this face

Offsets for Face's Vertex-List Data Array

Each face vertex-list data array consists of 3 words (NVPV=3). This array is accessed from the face pointer KFPVL.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	--	pointer to the next vertex on host face
1	KPVV	pointer to associated vertex in block's vertex list
2	KPVF	temporary link used in cut-plane routine

Offsets for Block's Vertex Data Array

Each vertex data array consists of 18 words (NVVR=18). This array is accessed from the block pointer KV and from the vertex-list pointer KPVV.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	--	vertex type number MVFAC=4 face vertex MVINT=5 interior vertex MVCOR=7 corner vertex
1	KNV	pointer to next vertex on host block (0 for end)
2	KHB	pointer to host block
3	KVX	start of x,y,z coordinates of vertex
6	KGXD	start of x,y,z components of velocity of vertex
9	KGFX	start of x,y,z components of force sum at vertex
12	KGM	vertex mass

Offsets for Block's Vertex Data Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
13	KGUD	start of x,y,z components of gridpoint displacement
16	KGDSF	density scaling factor for gridpoint
17	KBDEX	extension pointer to boundary vertex array

Offsets for Contact Data Array

Each contact data array consists of 28 words (NVCN=28). This array is accessed via the ICPNT pointer.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	MCON	=6, denotes contact
1	KNC	pointer to next contact in contact list
2	KCB1	address of first block involved in contact
3	KCB2	address of second block involved in contact
4	KCX	start of x,y,z coordinates of contact-plane reference point
7	KCNORM	unit normal of contact-plane (triple)
10	KCFN	normal force (a scalar)
11	KCFS	global components of shear force (triple)
14	KCAR	contact area

Offsets for Contact Data Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
15	KCCOD	code number
		<u>Contact Types</u>
	0	null contact
	MCFF=1	face-face
	MCFE=2	face-edge
	MCFV=3	face-vertex
	MCEE=4	edge-edge
	MCEV=5	edge-vertex
	MCVV=6	vertex-vertex
	MCMS=7	contact between joined blocks (master/slave logic)
16	KCN1	link field for block KCB1's list of contacts
17	KCN2	ditto for block KCB2
18	KCEX	extension pointer to FDEF contact array list
19	KCJUMP	last jump value of unit normal (used in UCP)
20	KCDN	relative normal displacement (a scalar)
21	KCDS	relative shear displacement (triple)
24	KCM	material type number
25	KCC	constitutive number
26	KCMOVE	spare offset
27	KCIND	contact failure indicator
	0	elastic
	1	at slip
	2	elastic but previously at slip
	3	tensile failure

Offsets for Contact Extension Array

Each contact extension array consists of 17 words (NVCX=17). This array is accessed from the contact pointer KCEX.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>								
0	KNCX	pointer to next array in list (0 for end)								
1	KCXV	vertex of vertex/face pair for the sub-contact								
2	KCXB	block, corresponding to vertex								
3	KCXF	face of vertex/face pair for the sub-contact								
4	KCXW1	weighting fraction of first vertex of face								
5	KCXW2	ditto, second vertex								
6	KCXW3	ditto, third vertex								
7	KCXFN	normal sub-contact force								
8	KCXFS	shear sub-contact force (three component vector)								
11	KCXDN	relative normal displacement of sub-contact (a scalar)								
12	KCXDS	relative shear displacement of sub-contact (triple)								
15	KCXAR	half area associated with sub-contact								
16	KCXIND	sub-contact failure indicator <table border="0" style="margin-left: 2em;"> <tr> <td>0</td> <td>elastic</td> </tr> <tr> <td>1</td> <td>at slip</td> </tr> <tr> <td>2</td> <td>elastic but previously at slip</td> </tr> <tr> <td>3</td> <td>tensile failure</td> </tr> </table>	0	elastic	1	at slip	2	elastic but previously at slip	3	tensile failure
0	elastic									
1	at slip									
2	elastic but previously at slip									
3	tensile failure									

Offsets for Zone Data Array

Each zone data array consists of 13 words (NVZO=13). This array is accessed from the block pointer KZ.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KNZ	pointer to next zone in host block (0 for end)
1	KZG	start of quadruple pointer to four vertices of tetrahedral zone
5	KZS	start of six components of stress tensor for zone
11	KZM	zone mass
12	KPLAS	plasticity indicator 0     elastic 1     at yield 2     elastic but previously at yield 3     surpassed tension cutoff

Offsets for Boundary Vertex Array

Each boundary vertex array consists of 14 words (NVBD=14). This array is accessed via the IBDPNT pointer.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KBDV	pointer to next boundary vertex
1	KBDVER	pointer to block vertex
2	KBDX	type of boundary condition in the x-direction
3	KBDY	type of boundary condition in the y-direction

Offsets for Boundary Vertex Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
4	KBDZ	type of boundary condition in the z-direction
		<u>Boundary Condition Types</u>
	0	free boundary
	MLOAD=1	load boundary
	MVISC=2	viscous boundary
	MVEL=3	velocity boundary
5	KBDFX	total x-force
6	KBDFY	total y-force
7	KBDFZ	total z-force
8	KBDFXI	applied x-force increment
9	KBDFYI	applied y-force increment
10	KBDFZI	applied z-force increment
8	KBDVX	applied x-velocity
9	KBDVY	applied y-velocity
10	KBDVZ	applied z-velocity
11	KBDCX	x-direction viscosity coefficient
12	KBDCY	y-direction viscosity coefficient
13	KBDCZ	z-direction viscosity coefficient

Offsets for Joined Blocks Array

Each joined block array consists of 2 words (NVLG=2) for each group of joined vertices. This array is accessed via the ILGPNT pointer.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KLGN	pointer to next joined vertex group
1	KLGVL	pointer to one vertex list-item in joined vertex group

Offsets for Joined Vertex-List Array

Each joined vertex-list array consists of 2 words (NVLGV=2). This array is accessed from the joined blocks pointer KLGVL.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KLGVN	pointer to next vertex-list item in group of joined vertices
1	KLGVV	pointer to associated vertex in block's vertex list

## 5.2 Main Common Block (/B/) Variables

LUNIF	unit number for local input file (=1)
LUNOF	unit number for output file (=6)
LUNG	unit number for save/restore file (=3)
LUNP	unit number for remote data file (=4)
LUNH	unit number for formatted history input file (=7)
LUND	spare unit number
LOCAL	.TRUE. for interactive mode
PCFLAG	.TRUE. if code is being run on IBM PC; else false
VNULL(3)	null array (0,0,0)
NERR	error number
ERFLAG	.TRUE. if an error has occurred
STFLAG	.TRUE. if the first input line has been processed
PLTFLG	.TRUE. if screen plotting has been done
ADFLAG	.TRUE. if auto damping is requested
DSFLAG	.TRUE. if density scaling is requested
NCELL	number of cells on one side of the cell space
IPCELL	pointer to the cell-space array
CELL(3)	dimension of one cell in each of 3 orthogonal directions
CELOGR(3)	global origin of the cell space
CELDUM(2)	spare
JMPSAV	index of last computed GOTO in MON
JUNK	pointer to list of spare memory groups
MFREE	first unused memory address
IBLOCK	current block number
ISTACK	stack pointer
NCINC	number of cycles since last CYCLE command
TTOT	total time
NCYC	currently requested number of cycles
NCTOT	total number of cycles
TDEL	timestep
FRAC	requested fraction of critical timestep for blocks
FRACZ	requested fraction of critical timestep for zones
JMPGEN	routing number for continuation line in GEN
GAM	mass damping factor $(\text{ALPHA} \cdot \text{TDEL})/2.0$
ALPHA	mass damping coefficient
BETA	stiffness damping coefficient
CON1	mass damping factor $(1.0 - \text{GAM})$
CON2	mass damping factor $(1.0 / (1.0 + \text{GAM}))$

BDT	stiffness damping factor (BETA/TDEL)
ADFAC	auto damping ratio
ADMUL	auto damping downward multiplier
ADMUP	auto damping upward multiplier
EKE	total kinetic energy in model
ATOL	split is prevented if cut-plane is within ATOL of a vertex
BTOL	tolerance to determine if old vertex is near enough (in CUTPRT)
CTOL	distance at which a contact is made
DTOL	spare tolerance
ETOL	spare tolerance
DEGRAD	PI/180.0
PI	3.14159
BCMASS	block mass for density scaling
ZCMASS	zone mass for density scaling
IBPNT	pointer to list of blocks
IBDEAD	pointer to last visible block in block list
ICPNT	pointer to list of contacts
IBDPNT	pointer to list of boundary vertices
IBDLST	pointer to last boundary vertex in list
GRAV(3)	x,y,z components of gravitational acceleration
FXHIS	current value of x-load history
FYHIS	current value of y-load history
FZHIS	current value of z-load history
IHPNT(2)	pointer to boundary input history array
IHNP(2)	number of points in boundary input history
SHT1(2)	time for first point of boundary input history
SHDT(2)	time increment for boundary input history
PTLOAD	frequency for boundary sine and cosine history
PFLOAD	period for boundary sine and cosine history
IXHIS	type of boundary x-load history
IYHIS	type of boundary y-load history
IZHIS	type of boundary z-load history
MATBD	boundary material number for viscous boundaries
AKN(NMAT)	normal joint stiffness
AKS(NMAT)	shear joint stiffness
AMU(NMAT)	joint friction coefficient
COH(NMAT)	joint cohesion
DENS(NMAT)	density
BULK(NMAT)	block bulk modulus
SHEAR(NMAT)	block shear modulus

ALAM1(NMAT)	Lame constant
ALAM2(NMAT)	Lame constant
BCOH(NMAT)	block cohesion
PHIG(NMAT)	block friction angle
PSIG(NMAT)	block dilation angle
CN2(NMAT)	plasticity constant
PHI(NMAT)	plasticity constant
PSI(NMAT)	plasticity constant
PLS1(NMAT)	plasticity constant
PLS2(NMAT)	plasticity constant
PLS3(NMAT)	plasticity constant
TENS(NMAT)	joint tensile strength
DIL(NMAT)	joint dilation coefficient
BTEN(NMAT)	block tensile strength
ZERDIL (NMAT)	joint shear displacement at which dilatancy is set to zero
ILGPNT	pointer to list of joined grid-points
IREPNT	pointer to list of axial reinforcement data
IRPPNT	pointer to list of axial reinforcement properties
ICBPNT	pointer to list of cable reinforcement data
VERS	version number
ARFTYP	average face area
ARCMIN	minimum contact area
ILNPNT	pointer to list of liner data
IPRPNT	pointer to list of properties for cable reinforcement and liner
ASPARE	spare variable locations
A(MTOP)	main array

#### /C/ Comon Block Variables

BAFLAG	.TRUE. if input/output in batch mode
LOG	.TRUE. if log file is activated
LUNL	unit number for log file

#### /H/ Common Block Variables

HED(80)	heading
---------	---------

Array Limits

<u>Size</u>	<u>Parameter</u>	<u>Description</u>
10	NMAT	maximum number of materials
5	NCONS	maximum constitutive numbers
2	NTYP	number of block types (rigid and FDEF)
700,000	MTOP	size of main array (a) for 4-mbyte DSI board
130+22*NMAT	MCOM	size of common block /b/

## 5.3 Graphics Transformation Common Block Variables

TRACOM Common Block Variables

ANGX	rotation angle about x axis
ANGY	rotation angle about y axis
ANGZ	rotation angle about z axis
CAX	cosine of ANGX
SAX	sine of ANGX
CAY	cosine of ANGY
SAY	sine of ANGY
CAZ	cosine of ANGZ
SAZ	sine of ANGZ
DISX	x-shift of system from origin
DISY	y-shift of system from origin
DISZ	z-shift of system from origin
ZVP	
ZEYE	distance of viewpoint (eye) from origin
ZEP	distance of projection plane from origin
XSCRL	
ZSCRU	
YSCRL	
YSCRU	
XSCRD	
YSCRD	

TRACOM Common Block Variables (continued)

WDXD  
WDYD  
CUT                    minus the distance of the cut plane from the origin  
ZWIND                z distance from the origin at which cut-off of blocks occurs  
IXWIND(2)  
IYWIND(2)  
DISSCL  
VELSCL  
STRSCL  
IMAGNF  
RMAGNF  
IKCON  
IZCTPT  
IGCTPT  
DIP  
DD  
CFSCAL  
CUTFLG  
AXIFLG  
PLFLAG  
OUTFLG  
COLFLG  
BOLFLG  
CREGFLG  
VVFLAG  
DVFLAG  
XXCFLG  
YYCFLG  
ZZCFLG  
XYCFLG  
YZCFLG  
ZXCFLG  
XVCFLG  
YVCFLG  
ZVCFLG  
XDCFLG  
YDCFLG  
ZDCFLG

TRACOM Common Block Variables (continued)

SVFLAG  
ZONFLG  
MAGFLG  
NVFLAG  
CZFLAG  
CHGFLG  
JXFLG  
BUFMOD  
DEBUG  
STRFLG  
PRNFLG  
CONFLG  
JNTFLG  
SCLFLG  
BGFLAG  
MENFLG  
CABFLG  
CFFLG  
LINFLG  
FILFLG  
FCOLFLG  
NEFLG

BEDCOM Common Block Variables

ISFREE  
IHEAP  
ISEG(ISMAX)

HPCOM1 Common Block Variables

HPFLG  
IHPCOL  
HPRX1  
HPRX2  
HPRY1  
HPRY2  
HPRX3

HPCOM2 Common Block Variables

HPFILE

Array Limits

<u>Size</u>	<u>Parameter</u>	<u>Description</u>
50,000	ISMAX	

#### 5.4 History Offsets and Common Block (HISCOM, HCOM1 AND HCOM2) Variables

##### Offsets for History Arrays

Each history array consists of 8 words (NVOH=8). This array is accessed via the IOHPNT pointer in the HCOM1 common block.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KOHN	pointer to next history array
1	KOHNH	number of histories in array
2	KOHNP	number of points in each history
3	KOHT1	time of first history point
4	KOHDT	time increment of history
5	KOHNC	cycle increment for history
6	KOHC1	cycle number for first history point
7	KOHP	

##### HCOM1 Common Block Variables

NOHIS	number of histories
NOHCYC	cycle increment
NOHTYP	history output to screen
NPOH	
NOHREC	
IOHPNT	pointer to stored history data
IOHBL	address of latest history array
IOHADD(NOHPMAX)	history address

HCOM2 Common Block Variables

HISLAB(NOHMAX)

Array Limit

<u>Size</u>	<u>Parameter</u>	<u>Description</u>
40	NOHMAX	maximum number of history variables

## 5.5 Axial Reinforcement Offsets

### Offsets for Axial Reinforcement Array

Each axial reinforcement array consists of 13 words (NVRE=13). This array is accessed via the IREPNT pointer in the B Common Block.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KREN	pointer to next reinforcement array
1	KREB1	address of first block connected with reinforcement element
2	KREB2	address of second block connected with reinforcement element
3	KREF1	address of face on first block connected with element
4	KREF2	address of face on second block connected with element
5	KREX	start of x,y,z coordinates of reinforcement location between block 1 and block 2
8	KRED	direction cosine of reinforcement between block 1 and block 2
11	KREF	axial force in reinforcement
12	KREMAT	reinforcement material type number

**Offsets for Axial Reinforcement Property Array**

Each axial reinforcement property array consists of five words (NVRP=5). This array is accessed from the control block pointers.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next array (0 for end)
1	KRPKAX	elastic axial stiffness
2	KRPULT	ultimate axial capacity
3	KRPLEN	1/2 "active" length
4	KRPID	property number

## 5.6 Cable Reinforcement Offsets

### Offsets for Cable Reinforcement Control Block

All cable reinforcement data are accessed through the main control block which consists of two words (NVCB=2). The address for the first word in the control block is given by ICBPNT.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KCNODS	pointer to list of cable nodes
1	KCELS	pointer to list of cable (axial) elements

### Offsets for Cable Element Data Array

Each cable element array consists of ten words (NVCELEM=10). This array is accessed from the control block pointer KCELS.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next element (0 for end)
1	KELNA	address of node A
2	KELNB	address of node B
3	KELTAD	address of properties for this element
4	KELL	element length
5	KELFAX	element axial force (tension negative)
6	KELTYP	element property number
7	KELUT1	element direction cosine (relative to x-axis)
8	KELUT2	element direction cosine (relative to y-axis)
9	KELUT3	element direction cosine (relative to z-axis)

Offsets for Cable Node Data Array

Each cable element node consists of 30 words (NVCNOD=30). This array is accessed from the control block pointer KCNODS.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next node (0 for end)
1	KNDTAD	address of properties for this node
2	KNDHZ	host zone for this node
3	KNDX	x-coordinate of node
4	KNDY	y-coordinate of node
5	KNDZ	z-coordinate of node
6	KNDXD	x-velocity of node
7	KNDYD	y-velocity of node
8	KNDZD	z-velocity of node
9	KNDFX	x-force sum
10	KNDFY	y-force sum
11	KNDFZ	z-force sum
12	KNDM	node mass
13	KNDUX	x-displacement
14	KNDUY	y-displacement
15	KNDUZ	z-displacement
16	KNDW1	weight for host node 1

Offsets for Cable Node Data Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
17	KNDW2	weight for host node 2
18	KNDW3	weight for host node 3
19	KNDW4	weight for host node 4
20	KNDFS	shear force on node
21	KNDBFL	bond flag (1 = intact, 2 = broken, 3 = none)
22	KNDT1	tangent direction cosine (relative to x-axis)
23	KNDT2	tangent direction cosine (relative to y-axis)
24	KNDT3	tangent direction cosine (relative to z-axis)
25	KNDEFL	effective length for shear force calculation
26	KNDTYP	node property number
27	KNDAPX	applied force in x-direction
28	KNDAPY	applied force in y-direction
29	KNDAPZ	applied force in z-direction

## 5.7 Structural Liner Offsets

### Offsets for Structural Liner Control Block

All structural liner data are accessed through the main control block, which consists of 2 words (NVLN=2). The address for the first word in the control block is given by ILNPNT.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0	KLNODS	pointer to list of liner nodes
1	KTRELS	pointer to list of flat triangular elements

### Offsets for Liner Node Array

Each liner node consists of 41 words (NVLNOD=41). This array is accessed from the control block point KLNODS.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next array (0 for end)
1	KLNTAD	address of properties for this node
2	KLNHF	address of host (contact) face
3	KLNX	x-coordinate of node
4	KLNY	y-coordinate of node
5	KLNZ	z-coordinate of node
6	KLNXD	x-velocity of node
7	KLNYD	y-velocity of node
8	KLNZD	z-velocity of node
9	KLNFY	x-force sum
10	KLNFY	y-force sum
11	LKNFZ	z-force sum
12	LKNM	node mass
13	KLNUX	x-total displacement of node
14	KLNUY	y-total displacement of node
15	KLNUZ	z-total displacement of node
16	KLNFN	normal force between liner and host medium

Offsets for Liner Node Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
17	KLNFSX	x-component of shear force between liner and host medium
18	KLNFSY	y-component of shear force between liner and host medium
19	KLNFSZ	z-component of shear force between liner and host medium
20	KLNFS	resultant shear force between liner and host medium
21	KLNBFL	not used
22	KLNT1	not used
23	KLNT2	not used
24	KLNT3	not used
25	KLNEFL	not used
26	KLNTYP	property number
27	KLNAPX	applied force in x-direction
28	KLNAPY	applied force in y-direction
29	KLNAPZ	applied force in z-direction
30	KLNMOI	moment of inertia
31	KLNMX	moment sum of x-axis
32	KLNMY	moment sum of y-axis
33	KLNMZ	moment sum of z-axis
34	KLNTD1	angular velocity about x-axis
35	KLNTD2	angular velocity about y-axis
36	KLNTD3	angular velocity about z-axis
37	KLNTH1	total angular rotation about x-axis
38	KLNTH2	total angular rotation about y-axis
39	KLNTH3	total angular rotation about z-axis
40	KLNIND	bond flag (0 = intact, 1 = broken)

Offsets for Liner Element Array

Each liner element consists of 83 words (NVTELEM=83). (Most words are used to store components of the element stiffness array.) The liner element array is accessed from the control block pointer KTRELS.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next array (0 for end)
1	KTRNA	address for node a
2	KTRNB	address for node b
3	KTRNC	address for node c
4	KTRAD	address of properties for this element
5	KTRTYP	property number
6	KTRUN1	direction cosine (relative to x-axis) of normal to element
7	KTRUN2	direction cosine (relative to y-axis) of normal to element
8	KTRUN3	direction cosine (relative to z-axis) of normal to element
9	KTRNAX	local x-coordinate (relative to element centroid) for node a
10	KTRNBX	local x-coordinate (relative to element centroid) for node b
11	KTRNCX	local x-coordinate (relative to element centroid) for node c
12	KTRNAY	local y-coordinate (relative to element centroid) for node a
13	KTRNBY	local y-coordinate (relative to element centroid) for node b

Offsets for Liner Element Array (continued)

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
14	KTRNCY	local y-coordinate (relative to element centroid) for node c
15-35		components of stiffness matrix for plane stress triangular element. Components of upper triangular region of matrix stored sequentially by rows.
36-80		component of stiffness matrix for plate-bending triangular element. Components of upper triangular region of matrix stored sequentially by rows.
81	KMZDIA	main diagonal term for fictitious rotational stiffness
82	KMZOFF	off diagonal term for fictitious rotational stiffness

## 5.8 Cable Reinforcement and Liner Material Parameter Offsets

### Offsets for Property Array

Each property array consists of 14 words (NVTYPE=14). This array is accessed from the main common block variable IPRPNT.

<u>Offset</u>	<u>Parameter</u>	<u>Description</u>
0		pointer to next array (0 for end)
1	KTYPE	elastic modulus
2	KTYPAR	cross-sectional area of cable element
3	KTYPKB	shear stiffness of cable grout annulus (units=force/disp/unit length)
4	KTYP SB	shear strength of cable grout annulus (units=force/unit length)
5	KTYPYI	yield capacity of cable element (units=force)
6	KTYPID	property number
7	KTYPT	thickness of liner
8	KTYPNU	Poisson's ratio
10	KTYPKN	normal stiffness (force/disp) of liner/rock interface
11	KTYPKS	shear stiffness (force/disp) of liner/rock interface
12	KTYPTN	tensile capacity (force) of liner/rock interface
13	KTYPCH	cohesive capacity (force) of liner/rock interface
14	KTYPMU	friction coefficient for liner/rock interface

## 6.0 EXAMPLE PROBLEMS

The following section provides examples which were chosen to illustrate the various features of 3DEC. Where possible, the results are compared to closed-form solutions for verification of the code operation.

### 6.1 Example 1: Sliding Wedge

The first test for the code 3DEC is the analysis of wedge failure resulting from sliding along two intersecting planar surfaces. A single wedge is formed by two structural features defined by:

A:	dip = 40°	dip direction = 130°
B:	dip = 60°	dip direction = 220°

The same mechanical properties are assumed for both planes. Classical wedge stability analysis presented in Hoek and Bray (1977) gives a friction coefficient required for stability equal to 0.66 for this wedge geometry.

The 3DEC model shown in Fig. 6-1 consists of four rigid blocks. Three of the blocks are fixed, and only the isolated wedge is allowed to move under gravity. The mechanical properties selected for this model are:

density:  $\rho = 2000 \text{ kg/m}^3$

joint shear and normal stiffness:  $K_n = K_s = 10 \text{ MPa/m}$

gravity acceleration:  $10 \text{ m/sec}^2$

The wedge is first allowed to consolidate under its weight by setting the friction on both planes to a high value and applying adaptive damping. Then, friction is reduced until failure occurs. Figure 6-1 shows the failure mechanism in progress.

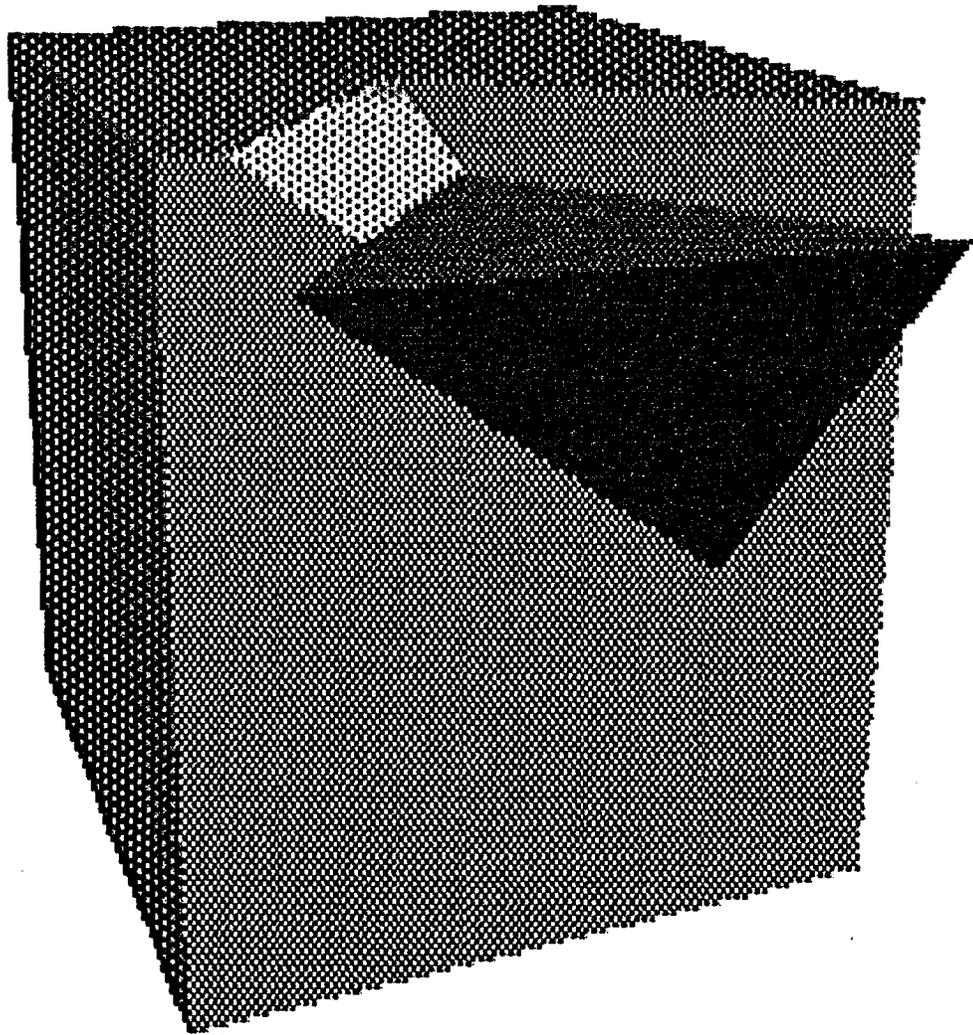


Fig 6-1 Sliding Wedge Model

Two of the wedge faces contact with faces of adjacent blocks. The edge at the back of the wedge contacts with an edge of the rear block. Joint shear and normal stiffness of 10 MPa/m are assigned to all three contacts. A friction coefficient of 0.646 is calculated for stability (within 2% of the analytical solution).

### Example 1 Data File

```

* Example 1 - Sliding Wedge
* sliding wedge --- verification test
*
poly region -1 1 -1 1 -1 1
* create two intersecting joints
jset dd=130 dip=40 org=0 1 0
jset dd=220 dip=60 org=0 1 -0.25
* joint properties
prop mat=1 d=2e-3 kn=10 ks=10 fric=100
* fix other 3 blocks and apply gravity
fix -1 1 -1 0.8 -1 1
grav 0 -10 0
damp auto
hide -1 1 -1 0.8 -1 1
hist yvel .25 .2935 -1 type 1
seek
* consolidation under gravity with high friction
cycle 200
save ex1a.sav
* reduce friction to 0.7 : stable
prop mat=1 fric=0.7
cycle 200
* reduce friction to 0.66 : stable
prop mat=1 fric=0.66
cycle 400
* reduce friction to 0.646 : stable
prop mat=1 fric=0.646
cycle 400
* reduce friction to 0.645 : failure
prop mat=1 fric=0.645
cycle 6800
save ex1b.sav
return

```

## 6.2 Example 2: Falling Wedge

The second problem for 3DEC concerns the stability of a rock wedge in the roof of an excavation. This problem takes into account the stabilizing effect of the in-situ stresses and also shows the influence of joint stiffness. A closed-form solution for the friction coefficient required for stability of the wedge is given by Goodman et al. (1982).

The wedge shown in Fig. 6-2 is defined by three planes dipping at  $60^\circ$  and with dip directions of  $30^\circ$ ,  $150^\circ$  and  $270^\circ$ . The height of the wedge is 1 meter. The mechanical properties are the same as those in Example 1. An in-situ state of stress with the two principal stresses parallel to the excavation roof equal to  $-0.05$  MPa and a zero vertical principal stress is applied. For these property conditions, the closed-form solution yields a friction coefficient required for wedge stability of 0.685.

The numerical simulation is performed by fixing all the blocks surrounding the wedge and setting the contact forces between the three wedge faces and the adjacent blocks in accordance with the in-situ stress state. Displacement of the wedge under gravity leads to the relaxation of these contact stresses. The friction coefficient is reduced in the model until failure occurs. The 3DEC simulation calculates a coefficient for stability of 0.684 (within 0.2%). The failure of the wedge is indicated in Figure 6-2.

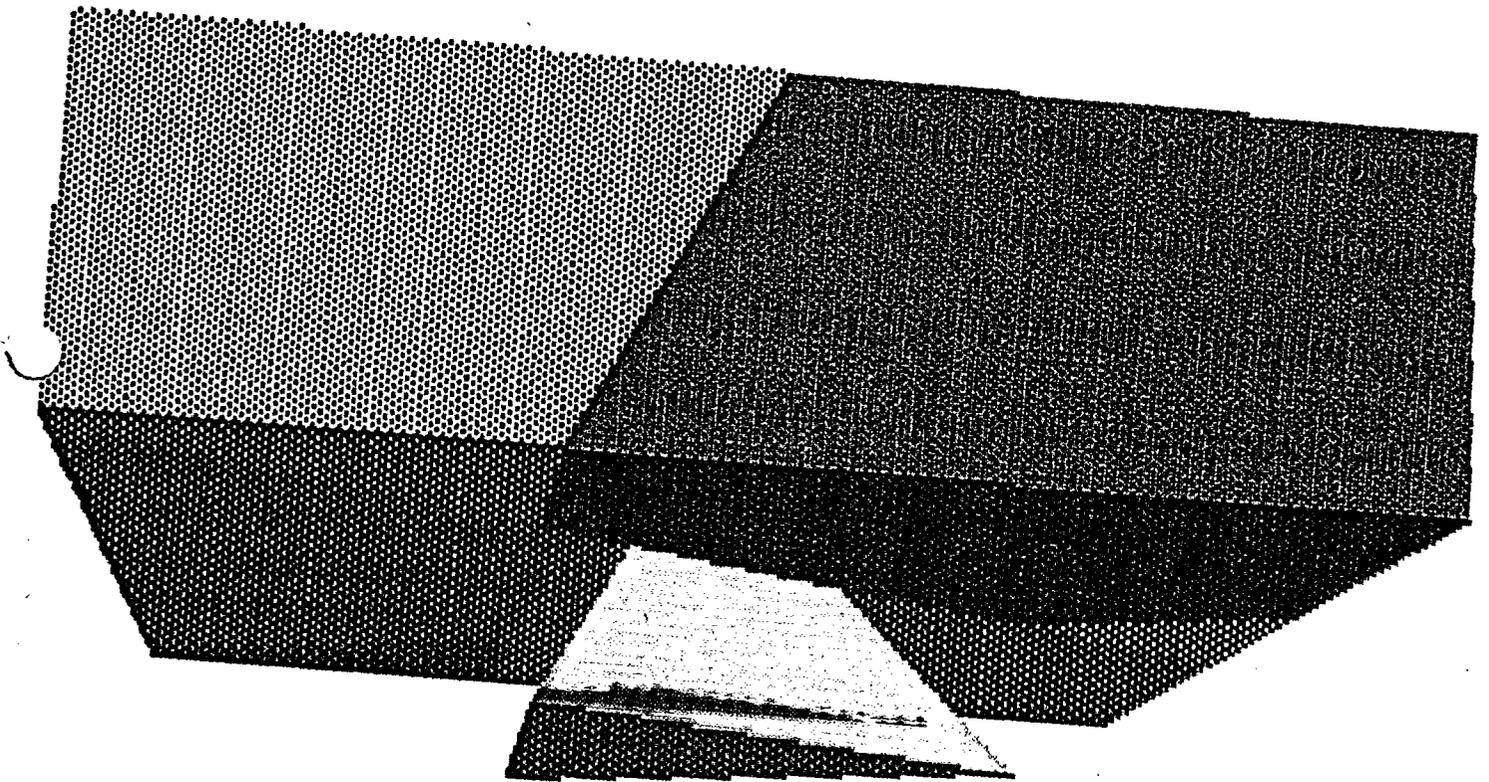


Fig. 6-2 Falling Wedge Model

Example 2 Data File

\* Example 2 - Falling Wedge  
\* roof wedge --- verification test  
\*

poly reg -2,2 -1,1 -2,2  
\*

jset  
\*

jset dip 60 dd 30 org 0 1 0  
jset dip 60 dd 150 org 0 1 0  
jset dip 60 dd 270 org 0 1 0  
delete -2,2 -1,0 -2,2  
\*

prop mat=1 d=2e-3 kn=10 ks=10 fric=100

grav 0 -10 0

damp auto

insitu stress -0.05 0 -0.05 0 0 0

fix -2,2 .3,1 -2,2

hist yvel -0.5769,0,-1 ty 1

cyc 400

\* set friction to 0.685

prop mat=1 fric=0.685

cyc 800

\* set friction to 0.684

prop mat=1 fric=0.684

cyc 1200

save ex2.sav

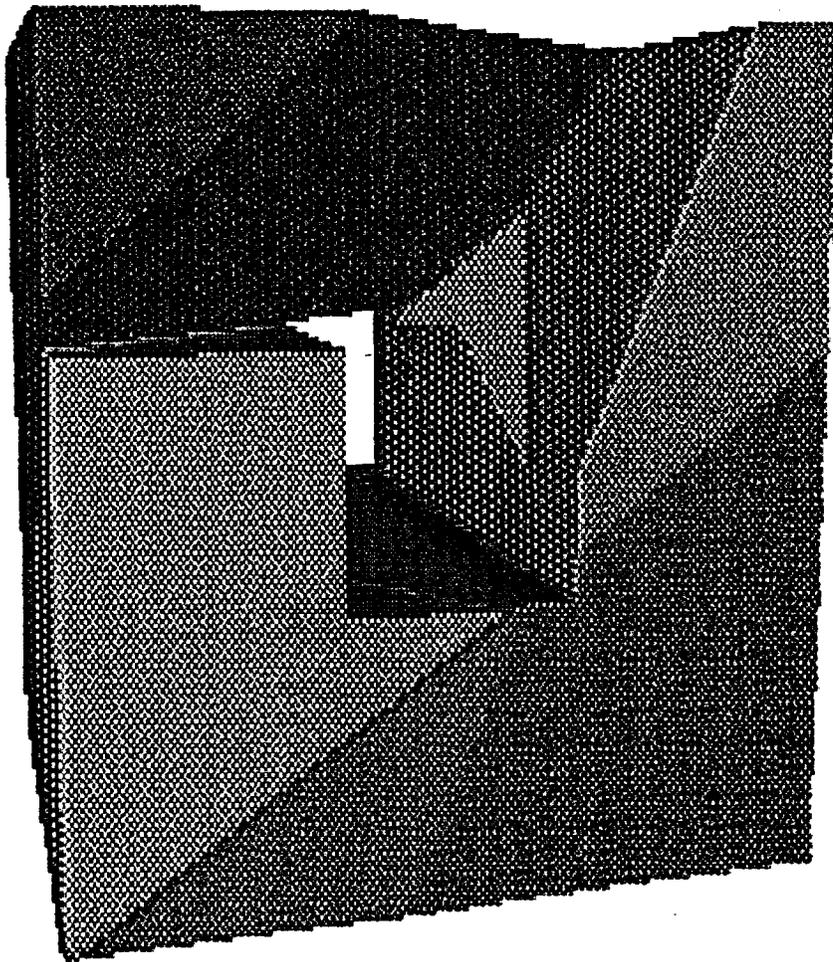
return

### 6.3 Example 3: Tunnel In Faulted Rock

This example extends the wedge models described in the first two examples to a more realistic problem of tunnel stability. The problem is defined as a single excavation in a faulted rock. The rock contains three major faults: one dipping at  $50^\circ$  with a dip direction of  $320^\circ$ ; the second dipping at  $40^\circ$  with a dip direction of  $230^\circ$ ; and the third dipping at  $65^\circ$  with a dip direction of  $270^\circ$ . A square opening is made in the model using the tunnel generator. The intersection of the faults with the tunnel is shown in Figure 6-3.

The three faults and the excavation form an isolated wedge in the roof of the excavation, which is potentially unstable and can slide along the fault plane dipping at  $65^\circ$ . Blocks in the front of the wedge are hidden from view in Figure 6-3 for better viewing of the wedge. (Hidden blocks are still present for mechanical calculations.) The model is subjected to gravity loading, and the progressive failure of the roof wedge is illustrated in Figure 6-3.

This model consists of 21 blocks and requires 480 kbytes of memory on the personal computer. A run time of approximately one-half hour is sufficient to identify failure in this case.



**Fig. 6-3** Tunnel in Faulted Rock (Front Blocks are hidden for better viewing of sliding wedge.)

Example 3 Data File

```

* Example 3 - Tunnel in Faulted Rock
* create a wedge that can slide into a square tunnel
*
prop m=1 d 2000 ks 1e9 kn 1e9
pol reg -1 1 -1 1 -1 1
jset dd=270 dip=65 org .3,0,0
jset dd=230 dip=40 org 0,0,-.3
jset dd=320 dip=50 org 0,0,.3
tunnel Reg=1      a -.3 -.3 -1.5 -.3 .3 -1.5 .3 .3 -1.5 .3 -.3 -1.5 &
                  b -.3 -.3 1.5 -.3 .3 1.5 .3 .3 1.5 .3 -.3 1.5
* (delete interior block)
delete -0.3,0.3 -0.3,0.3 -1.5,1.5
*
* apply gravity load
grav 0 -10 0
prop m=1 fric=100
damp auto
fix -1,1 -1,-.3 -1,1
*
hist yvel 0.3 0.3 0
hist ty=1
*
cyc 200
sav tuna.sav
* reduce friction along wedge
prop m=1 fric=0.5
reset time disp hist
hist ydis 0.3 0.3 0
hist ty=1
cycle 5000
save tunb.sav
return

```

#### 6.4 Example 4: Rock Slope Stability

This example is a three-dimensional representation of a rock slope in sedimentary rock. The example is an extension of the problem presented by Starfield and Cundall (1988) and involves a cut slope in rock with steeply dipping foliation planes. A rotational failure was found to occur with simultaneous sliding along both the foliation planes and shallow-dipping fracture planes. The rotation failure mode was identified by two-dimensional distinct element analysis as the principal mechanism for the slope collapse.

This problem was reproduced using 3DEC, with similar results to those of the two-dimensional analysis. A three-dimensional variation was then performed by introducing two intersecting discontinuities in the slope, forming a wedge similar to that in Example 1. The problem geometry is illustrated in Figure 6-4.

The failure mode for this problem combines the rotational failure with sliding of the wedge. The development of collapse is depicted in Fig. 6-5. The sliding wedge dominates the failure for this problem setting, but the rotational mechanism contributes to the collapse, particularly on the left portion of the slope face, as seen in Figure 6-5. Vertical sections taken through the wedge (Fig. 6-6) and through the right side of the face (Fig. 6-7) show the distinguishing response of the two types of failure. Displacement vectors at block vertices are plotted on these two figures, indicating the relative block movement. Cross-sectional plots can be oriented at any angle through the model, and various parameters can be presented on these sections.

This analysis demonstrates the application of the code to study conditions under which failure can occur. The present model investigated the effect of two joint sets and two intersecting discontinuities. The model contained 39 blocks and the run time, to the state of Fig. 6-5, was 3 1/2 hours, using a DSI-780 processor board. This state is further than required to identify that the slope has failed. A run time of approximately two hours is sufficient to identify failure in this problem. By making several runs with different properties, we can define the conditions leading to collapse of the slope. We can also introduce more discontinuities, if required, and still perform analyses in a reasonable time.

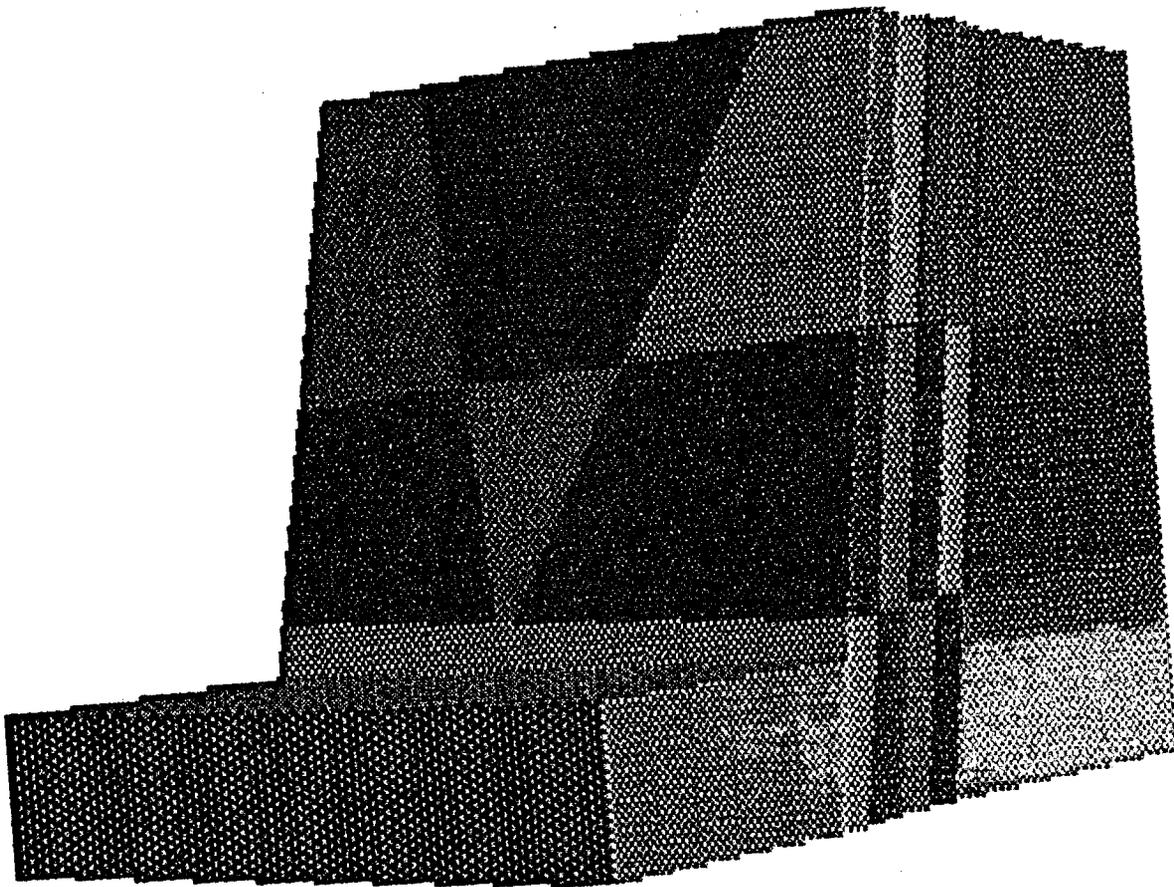


Fig. 6-4 Rock Slope Model

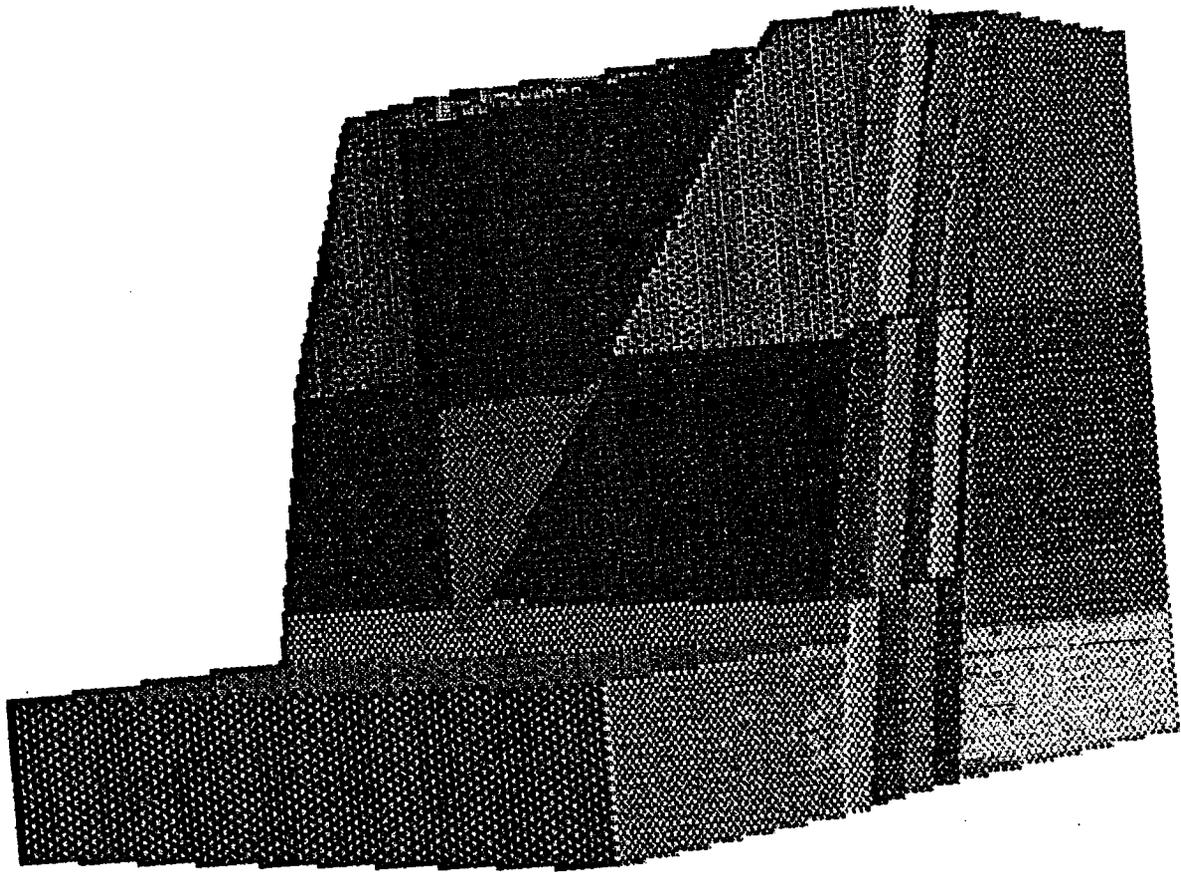


Fig. 6-5 Rock Slope Failure in Progress (Cycle 1500)

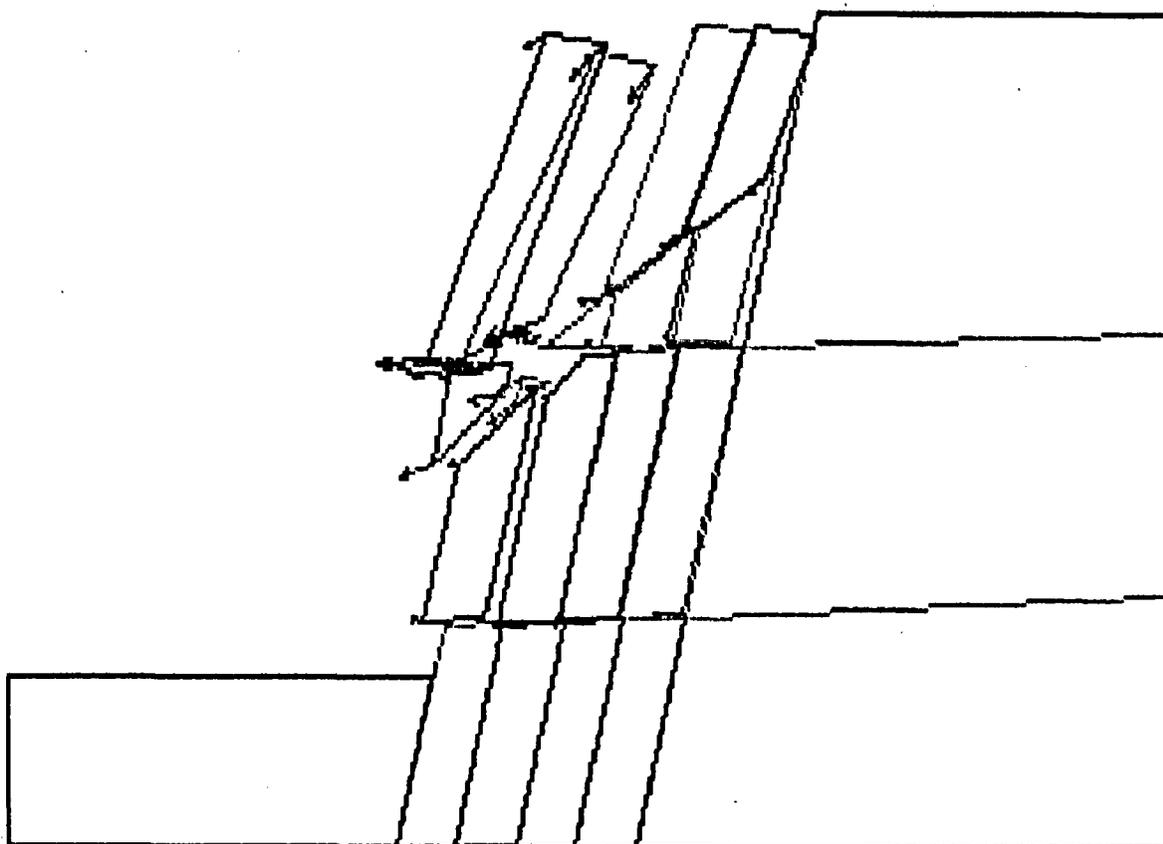


Fig. 6-6 Vertical Section Through Wedge in Slope (displacement vectors shown)

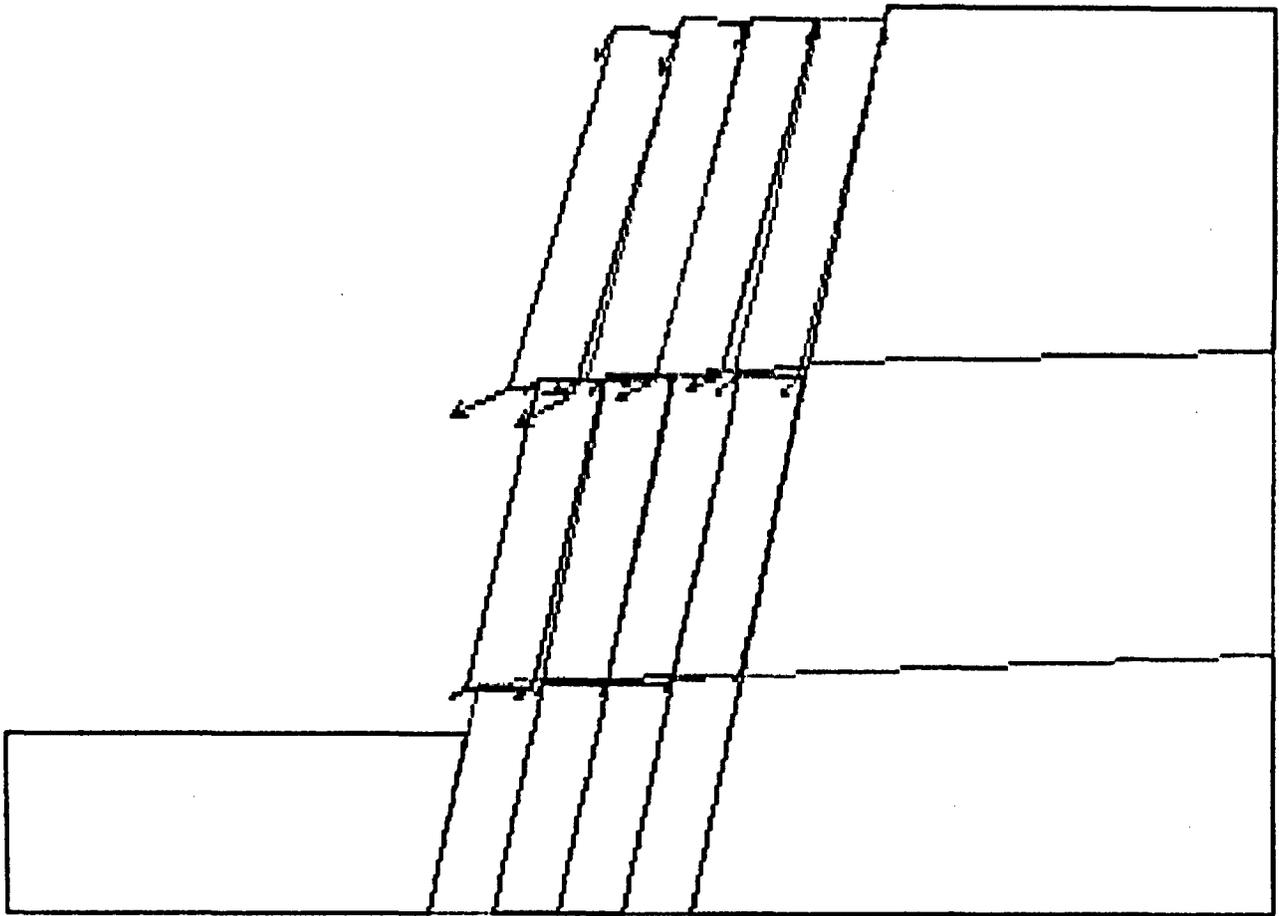


Fig. 6-7 Vertical Section Through Right Side of Slope (displacement vectors shown)

Example 4 Data File

```

* Example 4 - Rock Slope Stability
* rotational and sliding failure of slope
*
poly reg 0,80 0,50 -30,80
* boundary blocks on sides of slope
jset dip 90 dd 180 org 0,0,0
jset dip 90 dd 180 org 0,0,50
hide 0,80 0,50 -30,0
hide 0,80 0,50 50,80
* shallow-dipping fracture planes
jset dip 2.45 dd 235 org 30,12.5,0
jset dip 2.45 dd 315 org 35,30,0
* high angle foliation planes
jset dip 76 dd 270 spac 4 num 5 org 38,12.5,0
hide 30,80 0,50 0,50
jset dip 0 dd 0 org 0 10 0
seek
hide 0,80 0,50 -30,0
hide 0,80 0,50 50,80
hide 0 80 0 10 0 50
hide 55,80 0,50 0,50
hide 0,30 0,50 0,50
* intersecting discontinuities
jset dip 70 dd 200 org 0 0 35
jset dip 60 dd 330 org 50 50 15
seek
* fix boundary blocks and initialize gravity
fix 0 80 0 10 0 50
fix 55 80 0 50 0 50
fix 0,80 0,50 -30,0
fix 0,80 0,50 50,80
hide 0,80 0,50 -30,0
hide 0,80 0,50 50,80
delete 0,30 10,50 0,50
gravity 0 -10 0
* assign properties
prop m=1 d=2000 kn=1e9 ks=1e9 f=100.
prop m=2 kn=1e9 ks=1e9 f=0.0
chan dip 90 dd 180 jmat=2

```

Example 4 Data File (continued)

```
* prepare to cycle
damp auto
mscale on 1.0 7.122e6
hist yvel 30,30,30 ty=1
cyc 500
save ex4a.sav
prop m=1 f=0.05
cyc 12000
save ex4b.sav
return
```

## 6.5 References

Goodman, R. E., Gen-hua Shi and William Boyle. "Calculations of Support for Hard, Jointed Rock Using the Keyblock Principal," in Issues in Rock Mechanics (Proceedings of the 23rd Symposium on Rock Mechanics, University of California, Berkeley, August 1982), pp. 883-898. New York: Society of Mining Engineers, 1982.

Hoek, E., and J. W. Bray. Rock Slope Engineering (2nd Ed.). London: Inst. of Mining & Metallurgy, 1977.

Starfield, A. M., and P. A. Cundall. "Towards a Methodology for Rock Mechanics Modelling," accepted for publication in the Int. J. Rock Mech. Min. Sci. & Geomech. Abstr., 1988.