ED 115 205 IR 002 490

AUTHOR	Hamilton, Claude Hayden, III
TITLE	An Interactive Preprocessor Program with Graphics for a Three-Dimensional Finite Element Code.
INSTITUTION	Texas Univ., Austin. Project C-BE.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	EP-34-2-27-75
PUB DATE	27 Feb 75
NOTE	95p.; For related documents see IR 002 463 and 464
EDRS PRICE	MF-$0.76 HC-$4.43 Plus Postage
DESCRIPTORS	*Computer Graphics; Computer Oriented Programs; *Computer Programs; Computers; *Display Systems; Geometric Concepts; Higher Education; Input Output Devices; Programing
IDENTIFIERS	*Interactive Preprocessor Programs; Preprocessor Programs; Project C BE; Structural Mechanics

ABSTRACT

The development and capabilities of an interactive preprocessor program with graphics for an existing three-dimensional finite element code is presented. This preprocessor program, EDGAP3D, is designed to be used in conjunction with the Texas Three Dimensional Grain Analysis Program (TXCAP3D). The code presented in this research is capable of the verification and modification of data generated by TXGAP3D. The particular areas which can be handled by the code are those of grid point location, element connectivity, and boundary condition information as well as the addition and deletion of elements from the generated data. The interactive graphics provide a simple flexible visual aid and can be used with any graphics machine capable of interpreting standard Calcomp instructions. A user's input guide to EDGAP3D is appended along with an example session illustrating the use of the preprocessor program. (Author/CH)

AN INTERACTIVE PREPROCESSOR PROGRAM WITH GRAPHICS FOR A THREE-DIMENSIONAL FINITE ELEMENT CODE

EP-34/2/27/75

Claude Hayden Hamilton, III

Department of Civil Engineering
The University of Texas at Austin

(To fulfill written requirement for Master of Science Degree, January 1975)

*The materials contained herein were supported in part by PROJECT C-BE under Grant GY-9340, "The Use of Computer-Based Teaching Techniques in Undergraduate Science and Engineering Education," from the National Science Foundation to The University of Texas at Austin, Drs. John J. Allan and J. J. Lagowski, Co-Directors.

DIRECTORS:

Dr. John J. Allan III
Dr. J. J. Lagowski

ADDRESS:

413 Engineering Lab Building
The University of Texas at Austin
Austin, Texas 78712

(512) 471-4191

~~4~~5

ABSTRACT

The development and capabilities of an interactive preprocessor program with graphics is presented. This preprocessor program is designed to be used in conjunction with the Texas Three Dimensional Grain Analysis Program (TCGAP3D). The code presented in this research is capable of the verificiation and modification of data generated by TXGAP3D. The particular areas which can be handled by the code are those of grid point location, element connectivity, and boundary condition information as well as the addition and deletion of elements from the generated data. The interactive graphics provides a simple flexible visual aid and can be used with any graphics machine capable of interpreting standard Calcomp instructions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

The development of computer codes using the finite element method for performing structural analysis has been proceeding rapidly during the past decade. A majority of the effort expended has been made in the area of the central analysis program while little effort has been made in the areas of data generation or visual aids.[1] As a result, the analyst is often burdened with the task of preparing vast amounts of complicated input data to specify a problem. In cases where the analyst has available some form of automated data generation, he is often hampered by the inability to evaluate and correct parts of these data without having to completely regenerate the whole. These constraints become especially undesireable in the use of three-dimensional finite element codes, in which the amount of input data is voluminous and difficult to check. The development of preprocessor programs which are separately running computer programs may ease and/or improve the use of finite element programs.[2] The purpose of this research is the development of an efficient, interactive, preprocessor program for an existing three-dimensional finite element code. The particular preprocessor developed is concerned mainly with the editing of existing data and includes interactive graphics for visually checking this data.

There are certain data files associated with the use of any finite element code. These files contain information relating to the geometry of the problem to be analyzed, the connectivity of the elements, material properties, and the boundary conditions applied to the elements. These are the data which a preprocessor program handles. An additional difficulty arises when the problem to be analyzed is a three-dimensional one. This difficulty is that while it is easy to visualize most arbitrary two dimensional problems, it is more difficult to work in three dimensional space without some form of visual aid. The inclusion of flexible, interactive graphics routines in the development of the preprocessor was thus indicated. The particular preprocessor code, EDGAP3D, developed in conjunction with this research consists of basically two parts. The first, which allows the user to check and/or modify the previously generated input data for the proper geometry, connectivity and boundary conditions, and the second part which provides a visual means for checking both geometry and connectivity. The first part is referred to as an editor while the second forms the graphics package. These two parts or modules are linked together by a driver routine which directs the program execution to the appropriate module according to instructions issued by the user. There is no particular required sequence of instructions and the program execution can be arbitrarily directed back and forth between the two modules to suit the purpose of the user.

The EDGAP3D program is designed to be used in conjunction with an existing three dimensional finite element code, the Texas Three Dimensional Grain Analysis Program (TXGAP3D). The TXGAP3D code is constructed as a series of sequentially running subprograms. The sequence of execution of these subprograms is user controlled and to an extent may be arbitrarily directed, stopped, or restarted, depending on the directions established by the user. These subprograms are identified as SETUP, SOLVE, STRESS, REZONE, and PLOT. The general paths of execution for these subprograms is indicated in Figure 1a. SETUP is an automated data generation program which performs grid point generation, mesh generation and boundary condition application. The SOLVE program generates the element stiffnesses, forms the total stiffness matrix and solves for the nodal point displacements using a frontal solution technique. The STRESS program calculates the element stresses. The REZONE program consists of routines for refining the element mesh in specified regions. The

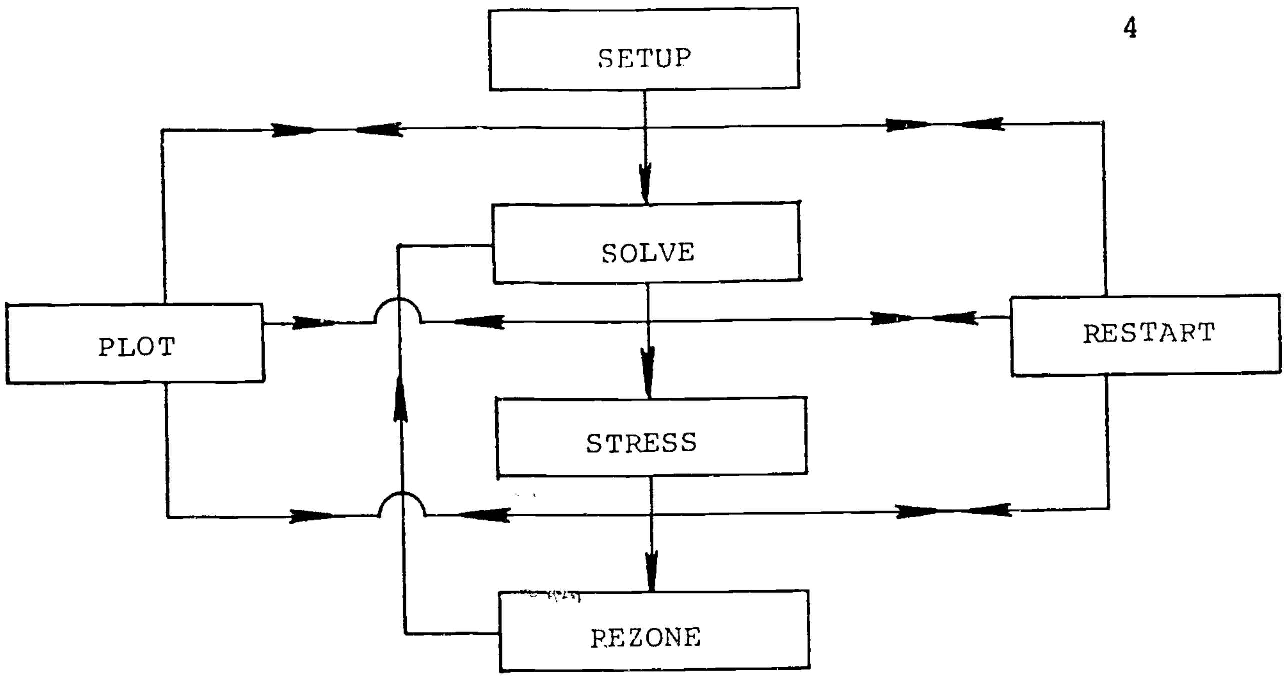


Figure 1a. TXGAP3D Program Execution Paths

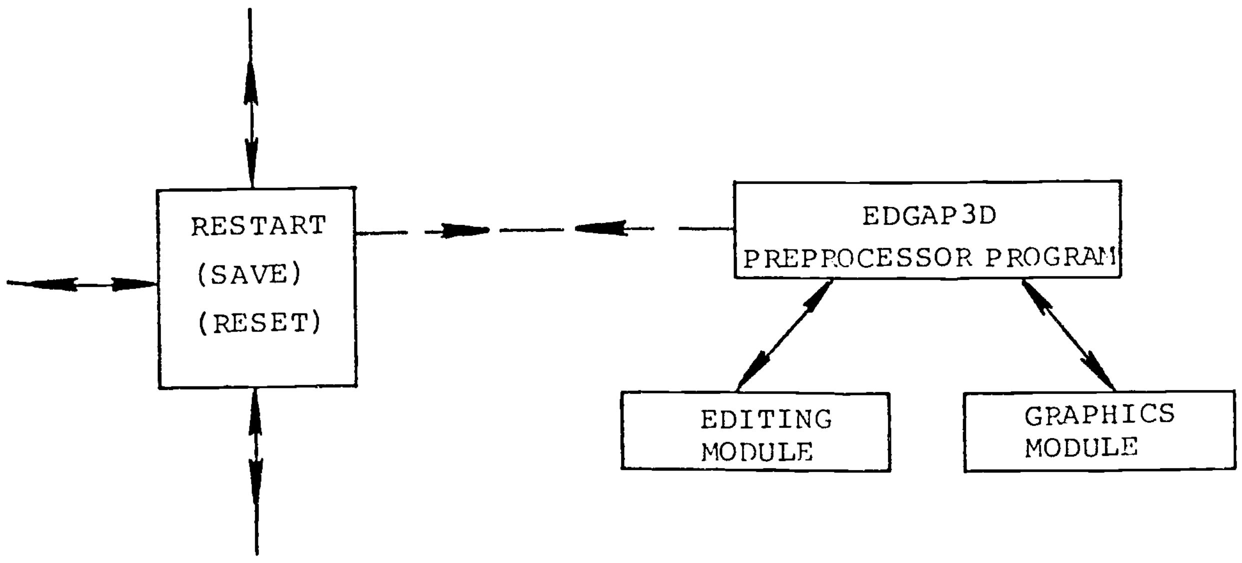


Figure 1b. EDGAP3D Preprocessor Interfacing with TXGAP3D

PLOT program provides a rudimentary package for plotting the element mesh.[3]

The linkage of the EDGAP3D program is accomplished through the RESTART feature available in TXGAP3D. Figure 1a and Figure 1b illustrate this linkage. The RESTART feature allows user specified stopping and restarting which saves and resets the data required to proceed with the program execution. The ability to stop and restart the program between the various subprograms arises from the fact that all communication between subprograms is via low speed disk files.[4] EDGAP3D is designed to access, examine, and modify these disk files. After these files have been examined and modified, the execution of TXGAP3D may be resumed with the corrected data. Although the preprocessing package may be used during any RESTART sequence, it is most useful immediately after the initial SETUP of the problem. The REZONE program causes extensive modification to the file containing input data and the EDGAP3D program is not capable of modifying this data, even though this capability would be a desireable feature. The data may, of course, be examined and displayed at any time.

The purpose of a preprocessor program is not only to ease the handling of data, but also to decrease the amount of time the analyst must spend preparing data. EDGAP3D has been designed to operate in such a way as to minimize the interaction time. On the operating system for which this program was developed, the interaction time is a direct function of the length of a program, significant coding effort has been to minimize the operating size of the EDGAP3D program. This has been accomplished largely through the use of modular programming similar to that of TXGAP3D.[4] Another area in which emphasis is placed in order to simplify EDGAP3D use has been to make the command directives as similar as possible to the TXGAP3D command directives.

CHAPTER II

EDITING CAPABILITIES ASSOCIATED WITH EDGAP3D PREPROCESSOR PROGRAM

The editor section of the EDGAP3D preprocessor program allows the user to examine and modify the data he has generated. This area of the preprocessor program specifically allows the user to work with three of the data files common to finite element problems. These files being those of the specified geometry, the element connectivity, and the boundary conditions. In addition to these features, the capability for the addition (and/or deletion) of elements, within certain restrictions, to (or from) the generated data set is included in the editor package.

Data manipulation is an important feature of the EDGAP3D program. The acquisition of information and internal handling of this information are among the primary considerations in the program structure. The Texas Three Dimensional Grain Analysis Program (TXGAP3D) generates the data for a problem and stores this information on two low speed disk files. One of these files contains the geometry in the form of all the grid point coordinates that have been generated. The second file has the information defining the element connectivity and the boundary conditions applied to the elements. It is through these two files that the EDGAP3D program is interfaced with the TXGAP3D program. The information on these data files is that which is examined and modified. Since these files are very large for large problems, the routines in the editor package which access these disk files store much of the information on relocatable high speed disk files. At the completion of the editing session, the two low speed disk files are regenerated containing the corrected information.

The EDGAP3D program does not include the capability for the editing of material property data.

There are some limitations associated with the usage of the EDGAP3D preprocessor program which are particularly related to the editor package. The principle limitation is that the user is restricted to previously defined data in both the addition of elements and the manipulation of element connectivity. This means that in order for an element to be added, all the points defining the connectivity and geometry must have been defined in TXGAP3D. In other words, an element can only be defined by already existing node points and the editor package does not allow for the addition of node points. The restriction that node points (or grid points) may not be added is offset by the ability to move grid points. If extra points are generated, they may be repositioned as needed during editing to define the proper geometry for the new element(s).

Geometry

The geometry for a problem is defined by the location of the grid points. The coordinates for each point are generated by an automated grid generator in TXGAP 3D and are saved on a low speed disk file. Each point in the grid is identified by a unique, I. J, K numbering system which defines its location in the grid. The TXGAP 3D program has two identifying schemes associated with it. One is the "external" identifier which the user refers to. In this scheme, every other node is sequentially numbered, and as far as the user is concerned, the other nodes have no labels. The "internal" labels are those actually used by the program for identifying nodes and in this scheme every node is sequentially numbered. Figure 2 shows a two dimensional example of these grid point labeling schemes. Because the user sees only the external numbering scheme which labels only every other node, a new scheme for identifying the unnamed nodes has been devised. This scheme is also illustrated in Figure 2 and a detailed discussion of this scheme is given in Appendix A.

The routines in the editor module of the EDGAP 3D program provide the ability to check and change the coordinate locations of every grid point established during the data generation step in TXGAP 3D, whether the point has or has not been used to define an element. The issuing of a command

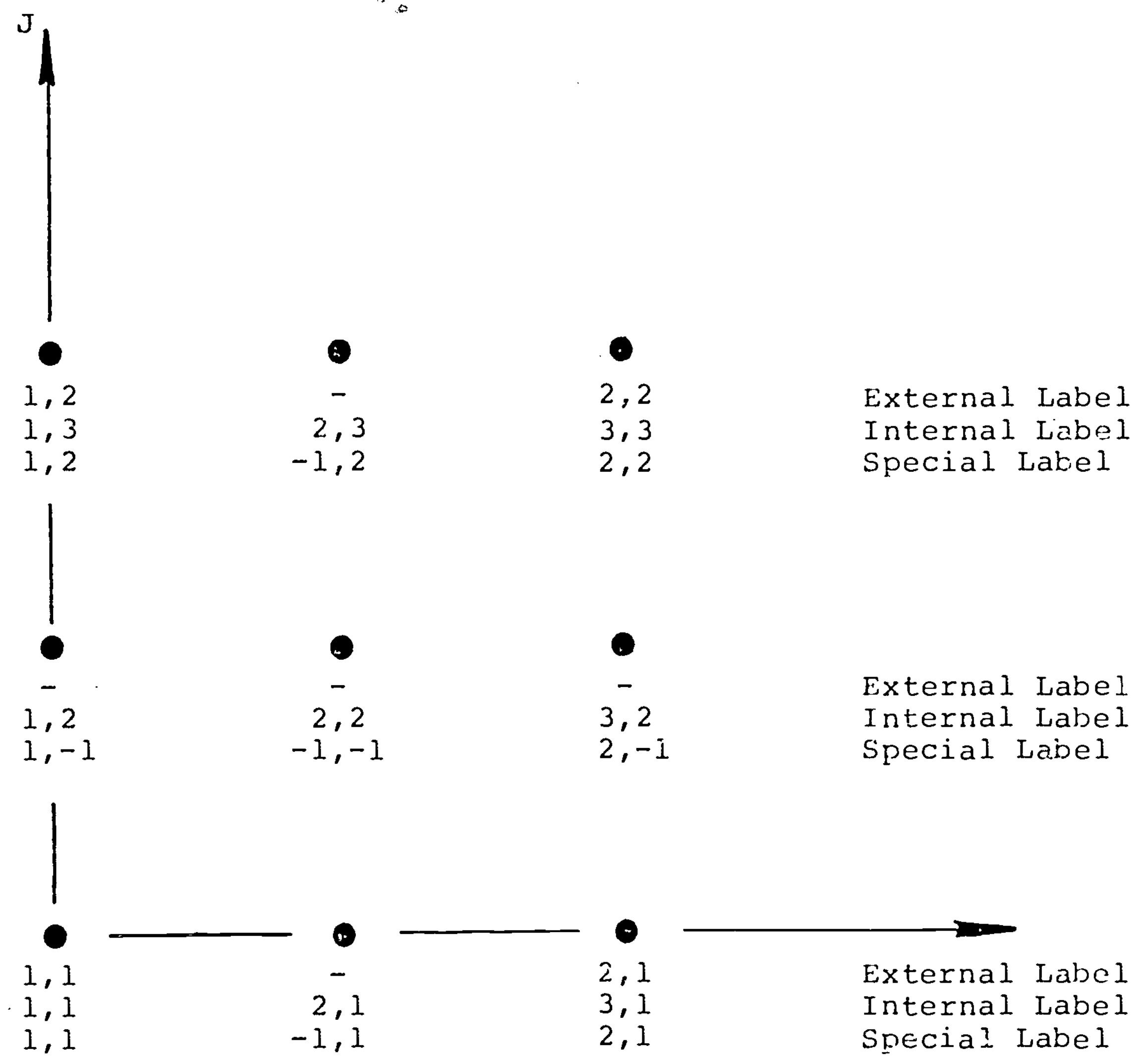


Figure 2. Grid Point Identification Schemes

directive identifying the grid point by the special, I, J, K labeling scheme described in Appendix A causes the program to print both the internal label and the X, Y, Z coordinates of the point. This information is printed to the user in the form of alphanumeric output. To change the location of a grid point, the user issues the same command directive identifying the grid point by its special label and includes the additional parameters of a flag key and the new X, Y, Z location of the point. Again, the internal label and the old coordinate positions are output as this step is executed by the routines. The issuing of a command to move a grid point causes not only a modification of the list of data containing the grid points, but also initiates a search of the element data, replacing all the points located at the old position by ones at the new one. Thus, if two nodal points identified by different labels have the same coordinates and only one is moved, the element data list will be modified as though both had been moved.

To illustrate the features described above, the following examples are presented. Suppose it is desired to first examine both a grid point with an external label and one without one in a list of generated data. The following commands and output illustrates both types of uses of the command directive for these cases.

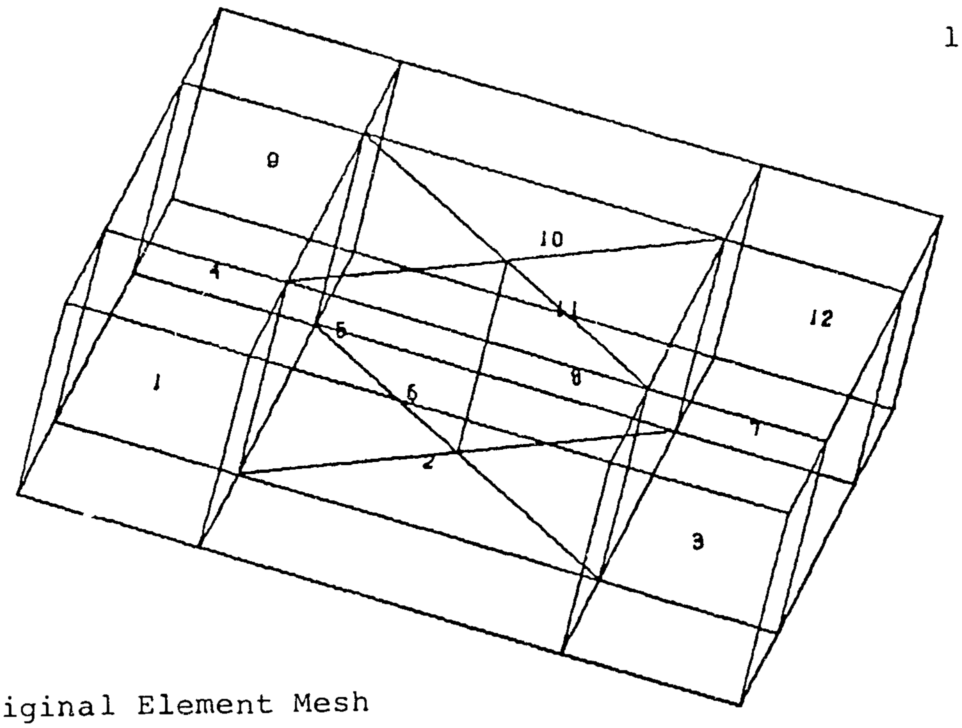
```
* GRID,2,1,2+
  __INTERNAL_I,J,K                    X,Y,Z_COORD.
  ____3_1_3                       1.200E+00 0.  1.333E+00
* GRID,-4,1,2
  __INTERNAL_I,J,K                    X,Y,Z_COORD.
      8,1,3                       4.200E+00 0.  1.333E+00
```

Now to illustrate the capability to change a nodal point location, the same points used above will be shifted to new locations. Note that the locations returned are the original positions. This is done so that if an error is made (i.e., one moves the wrong point) the original position is readily available for correcting the error. The resulting changes may be seen in Figure 3a and Figure 3b which show the nodes before and after modification

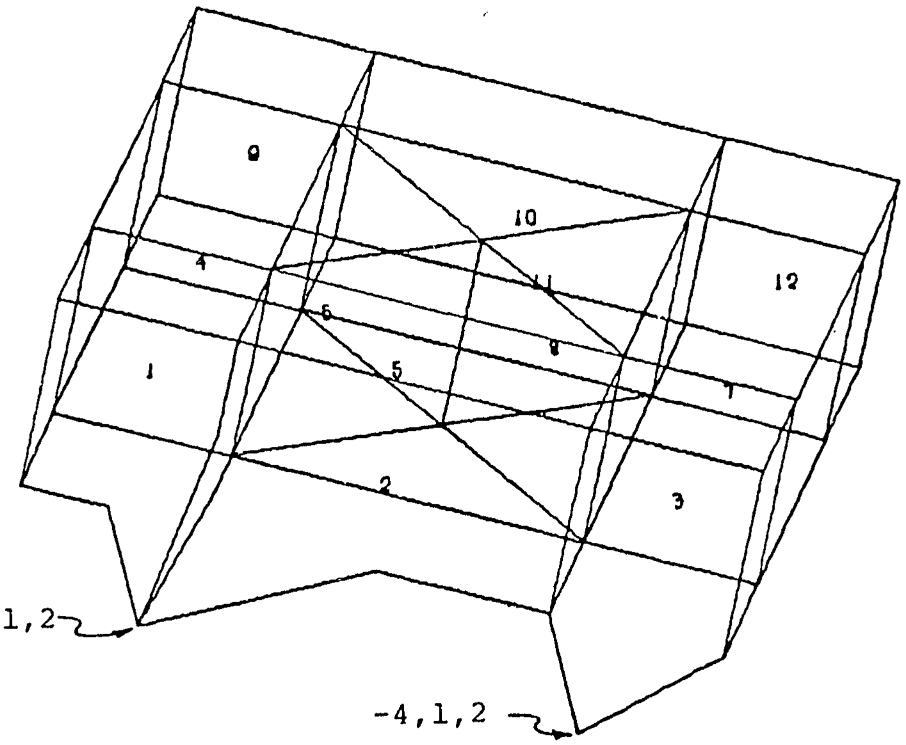
```
* GRID,2,1,2,MOVE,1,-.5,.75
  __INTERNAL_I,J,K                    X,Y,Z,_COORD.
  ____1_1_3                0.               0.    1.333E+00
* GRID,-4,1,2,MOVE,4,-.5,.75
  __INTERNAL_I,J,K                    X,Y,Z_COORD.
  ____8_1_3              4.200E+00     0.    1.333E+00
```

The ability to add or generate new grid points is not included in the preprocessor developed in this study.

+ Lines preceeded by an * will designate input by the user.



(a) Original Element Mesh

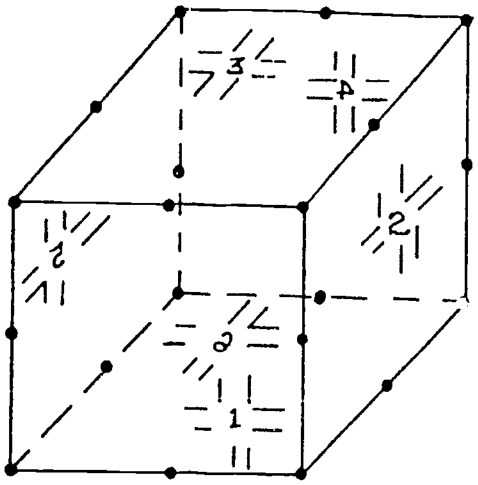


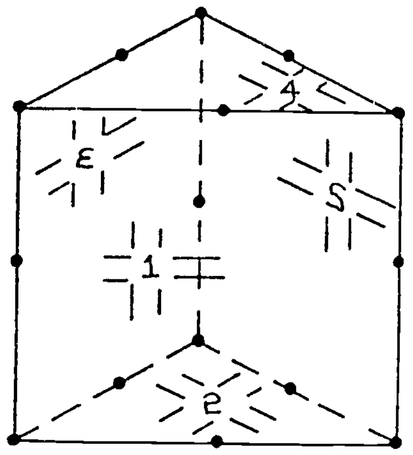
(b) Element Mesh with Shifted Grid Points

Figure 3. Grid Point Modification

Connectivity

The definition of elements is accomplished through the specification of the geometry of each particular element. The most common method used to accomplish this is by specifying the grid points which define the extremities of an element. An element is then considered to be "connected" to these nodal points. The number of points required to specify the connectivity of an element is dependent on the particular type of element. The element library in TXGAP3D consists of elements belonging to the isoparametric family having quadratic interpolations. These elements require the specification of the node points at each corner of the element and a midside node between each corner in order to define their connectivity. The particular elements available have the general geometric shapes of bricks, prisms, and tetrahedrons (see Figure 4). The specification of the connectivity of these elements assumes a preferrential order and requires only the specification of the corner nodes in order to define the elements (each midside node is determined by the program). The preferrential ordering of the nodal points is indicated by the element nodal point numbers in Figure 4. Each corner node of the element is identified by the proper, external, I, J, K label associated with the grid point to which it is connected. For the purpose of element identification, an





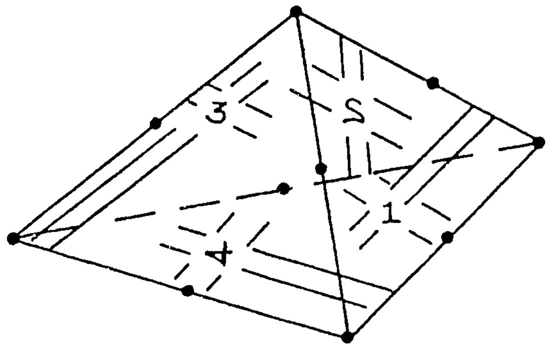


Figure 4. Element Nodes and Face Numbers

element name is given each element. This name is the external I, J, K label of the first node defining the elements connectivity.

The routines developed in the editor module of EDGAP3D have the capability of examining the connectivity associated with either elements or nodal points. For elements, the user specifies either the element name (I, J, K) or the element number. The element number represents the number which corresponds to the position in which an element appears on the element data file. This element number is most easily obtained from the graphical displays which will be discussed in the next chapter. For nodal points, the user would identify that grid point by its external name. When the connectivity of an element has been specified, there is for each element (corner) node number, an external I, J, K grid point label associated with it. If the element name is used as the identifier, every element by that name is located and the connectivity listed.* In cases in which the connec-

* While it is not often desireable to do so, it is possible to generate in the TXGAP3D program several elements which have the same name. This situation has been considered in the development of the preprocessor so that when this does occur, elements by the same name are processed sequentially. This allows the user to correct with a minimum of effort any errors that may have been caused by identical names.

tivity of a grid point is requested, the output lists each element name and the associated element node number connected to that grid point.

The capability to modify an element's connectivity available in this module of the EDGAP3D program. In the use of this feature, the element name or element number whose connectivity is to be modified is identified. This causes the existing connectivity to be listed. Then the individual element node's connectivity is changed in a random sequence. As noted previously, the connectivity can only be changed to previously existing, externally labeled grid points. If an attempt is made to change the element's connectivity to an undefined grid point, a warning message is issued and the input command for the change is ignored. When the connectivity of an element is modified, the geometry is also changed. Routines are included to perform internally the required modification of the element's geometry list whenever the connectivity list is changed.

To illustrate the various features associated with the verification and modification of element connectivity, the following examples are provided. Examples a) and b) illustrate element connectivity by specification of element name and element number respectively. Example c) shows the ability to determine which elements are connected to a specified grid point. Example d) illustrates the modification of an element's connectivity.

a)

```
* CONNECT,ELEMENT,2,1,2
 _CONNECTIVITY FOR ELEMENT____2,_1,_2
 __NODE_______I_____J_____K
 _   1        2     1     2
     2        4     1     2
     3        4     2     2
     4        2     2     2
     5        2     1     3
     6        4     1     3
     7        4     2     3
     8        2     2     3
```

b)

```
* CONNECT,ELEMENT,6
 _CONNECTIVITY FOE ELEMENT NO. 6
   NODE       I     J     K
     1        2     2     2
     2        3     3     2
     3        2     4     2
     4        2     2     3
     5        3     3     3
     6        2     4     3
```

c)

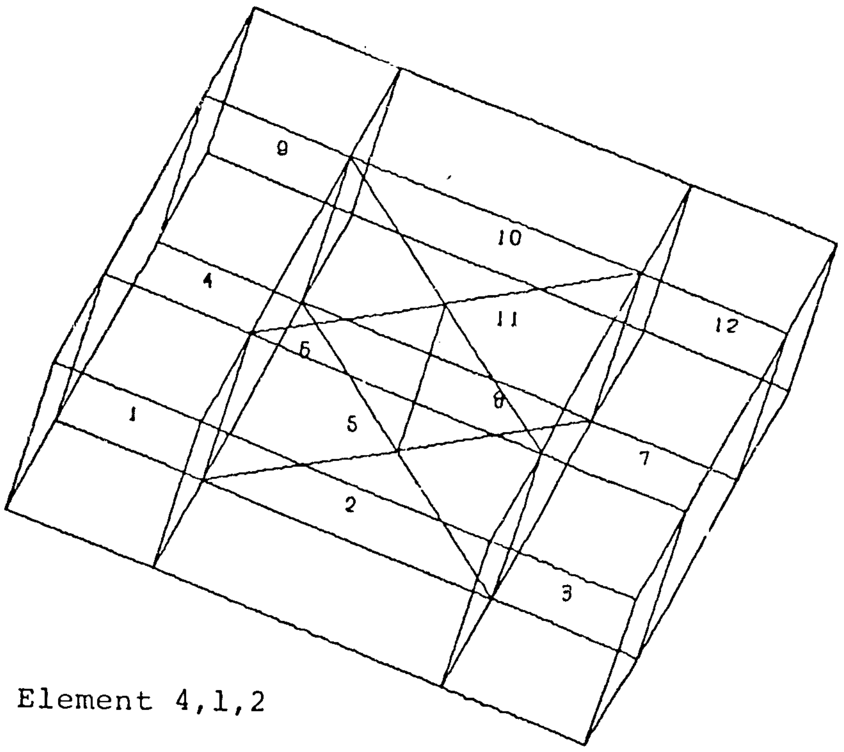
```
* CONNECT,NODE,3,3,3
 _CONNECTIVITY FOR NODE          - 3,_3,_3
 _____LOCATED AT COORD.            2.400E+00,3.000E+00,2.667E+0
 _ELEMENT          ELEMENT            LOCAL
 _NUMBER          I   J   K           NODE
     5            2   2   2            6
     6            2   2   2            5
     8            3   3   2            4
    11            4   4   2            6
```

```
d)         * CONNECT, MODIFY, 4,1,2

           _ CONNECTIVITY FOR ELEMENT         4,_1,_2

          __ NODE       I       J       K

             1          4       1       2
             2          5       1       2
             3          5       1       2
             4          4       2       2
             5          4       1       3
             6          5       1       3
             7          5       2       3
             8          4       2       3

           *6,6,1,3
           *2,6,1,2
           *2,6,1,1
           *FINISH
```

The modification of element 4,1,2 results in the element nodes 6 and 2 being changed to grid points (6,1,3) and (6,1,1) respectively. The results of this modification may be seen in Figure 5b, which indicates the changes made from Figure 5a.

Boundary Conditions

In the TXGAP 3D program, the application of boundary conditions is performed on an element basis. Depending on the type of boundary condition, this application is considered to be acting on either a nodal point or on a face of the element. As with the numbering of the element's nodal points, there is a specific pattern in the numbering of the element faces. This pattern is indicated by the numbering of the element faces as indicated in Figure 4. The specification of



(a) Original Element Mesh

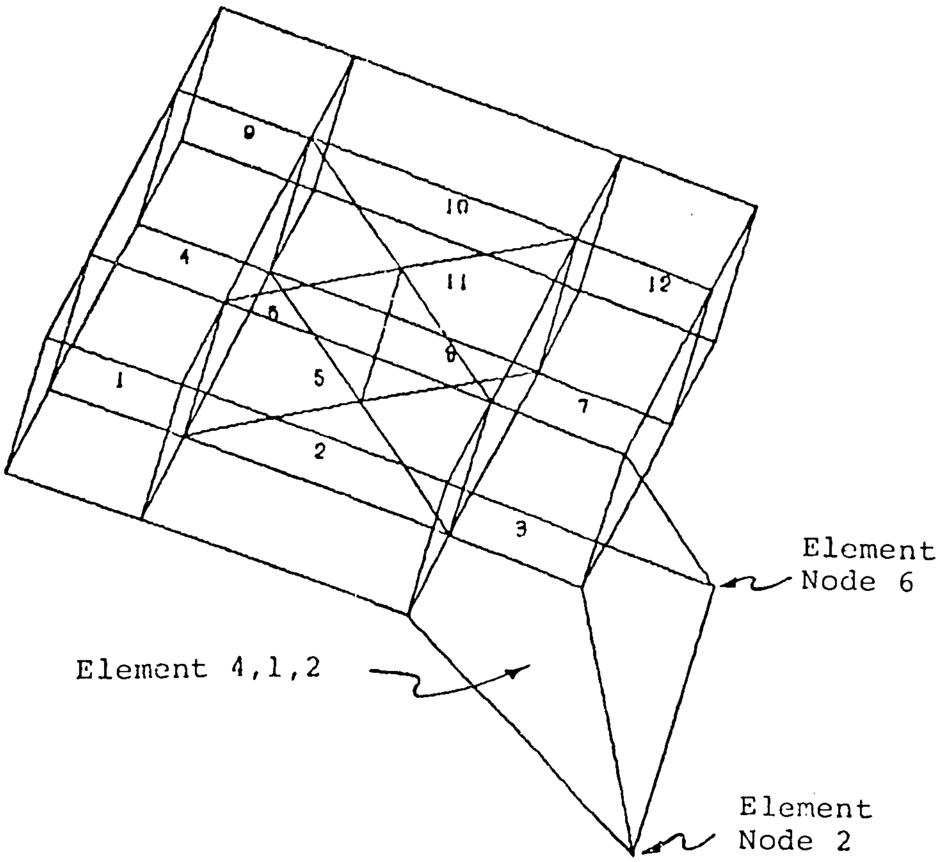


Figure 5. Element Connectivity Modification

boundary conditions in TXGAP3D requires the identification of the boundary condition type, the element by its element name, the node or face on which the boundary condition acts, and the value of the boundary condition. The routines in EDGAP3D associated with boundary condition information are capable of finding, adding, and deleting boundary conditions from the element data. All the boundary condition types in the TXGAP3D program are applicable in EDGAP3D with the exception of the CLAMPed type boundary condition. The lack of this boundary condition type arises from the manner in which this type of boundary condition is treated in the TXGAP3D program. A complete list of the boundary condition types applicable in EDGAP3D is to be found in Appendix A.

The preprocessor program allows the user two methods for verifying boundary conditions. These methods are by boundary condition type and by element. In the first method, it is possible to obtain, for a specified boundary condition type, all the elements, the location on the elements, and the value at that location to which that type of boundary condition has been applied. The second method outputs all the boundary conditions, their positions, and their values that have been specified on the requested element. In this method, either the element name or element number may be used to request this information.

The modification of boundary conditions is carried out on an element by element basis by adding and deleting the appropriate boundary conditions. If only the value of a boundary condition needs to be changed, it is unnecessary to delete the originally specified condition first. However, if the type or location is to be changed, the incorrect boundary condition must be deleted. When the addition or deletion mode is entered, the program is designed to first print all the existing boundary conditions on the specified element. The input specifying the boundary condition type, location, and value to be added (or deleted) is then required by the program. It should be noted that the program allows the addition (or deletion) of only one boundary condition at a time. There is no looping feature for multiple additions or deletions. If the element name is used to specify the element, the program will search for any other elements by that name before terminating the addition or deletion sequence. In the deletion mode, it is unnecessary to specify the value of the boundary condition being deleted.

The following examples illustrate the verification and modification of boundary condition information in the EDGAP3D program. Example a) illustrates the use of the feature which locates all the elements on which the specified boundary condition has been applied. Example b) is one in which all the boundary conditions applied on the specified element are located and listed. Examples c) and d) show the use of the addition and deletion modes.

a)
```
* BC,UX
___X-DISPLACEMENT BOUNDARY CONDITION
___ELEMENT              NODE             VALUE
   1,_4.,_2              3                 0.
   1,_4,_2               4                 0.
   1,4,2                 7                 0.
   1,4,2                 8                 0.
   2,4,2                 3                 0.
   2,4,2                 4                 0.
   2,4,2                 7                 0.
   2,4,2                 8                 0.
   5,4,2                 1                 0.
   5,4,2                 2                 0.
   5,4,2                 5                 0.
   5,4,2                 6                 0.
```

b)
```
* BC,ELEMENT,1,4,2
___BC FOR ELEMENT(S)____1,_4,_2
__BC_TYPE_______NODE/FACE____VALUE
    UX               2          0.
    UX               3          0.
    UX               6          0.
    UX               7          0.
    UY               2          0.
    UY               3          0.
    UY               6          0.
    UY               7          0.
    UZ               2          0.
    UZ               3          0.
    UZ               6          0.
    UZ               7          0.0
    PRESSURE         4         -7.500E+00
    PRESSURE         4         -7.500E+00
    PRESSURE         4         -7.500E+00
    PRESSURE         4         -7.500E+00
```

c)

```
* BC,ADD,4,1,2
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
  __ENTER BC INFO. TO BE ADDED OR DELETED
  PRESSURE,3,100.
```

d)

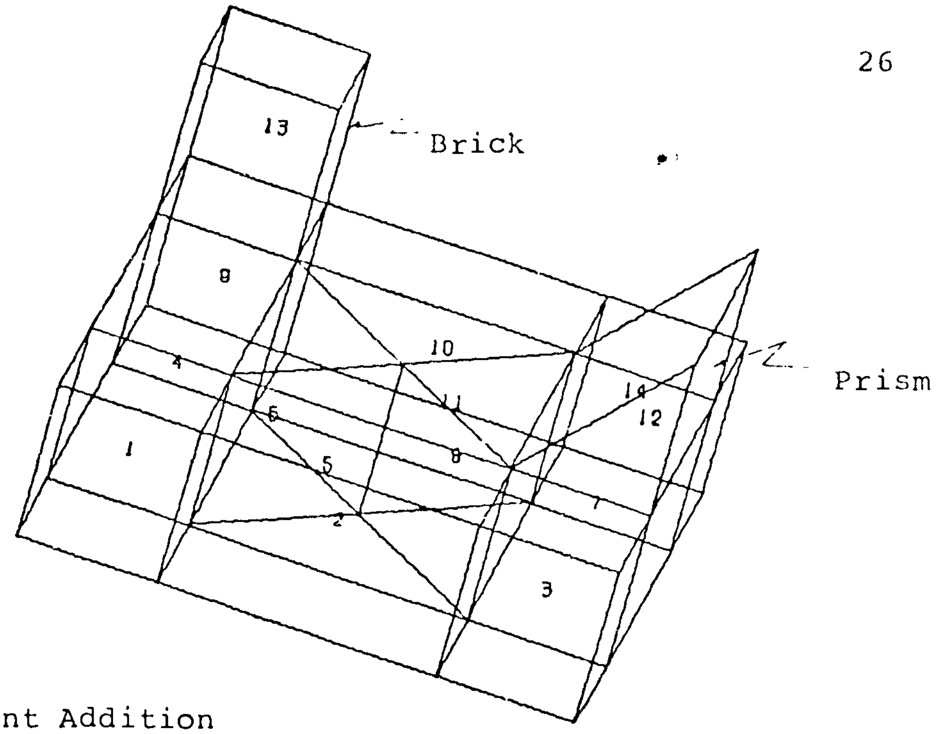
```
* BC,DELETE,4,1,2
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
       PRESSURE        5          5.500E+00
       PRESSURE        3          1.000E+02
       PRESSURE        3          1.000E+02
       PRESSURE        3          1.000E+02
       PRESSURE        3          1.000E+02
  __ENTER BC INFO. TO BE ADDED OR DELETED
* PRESSURE,5
```

Element Addition and Deletion

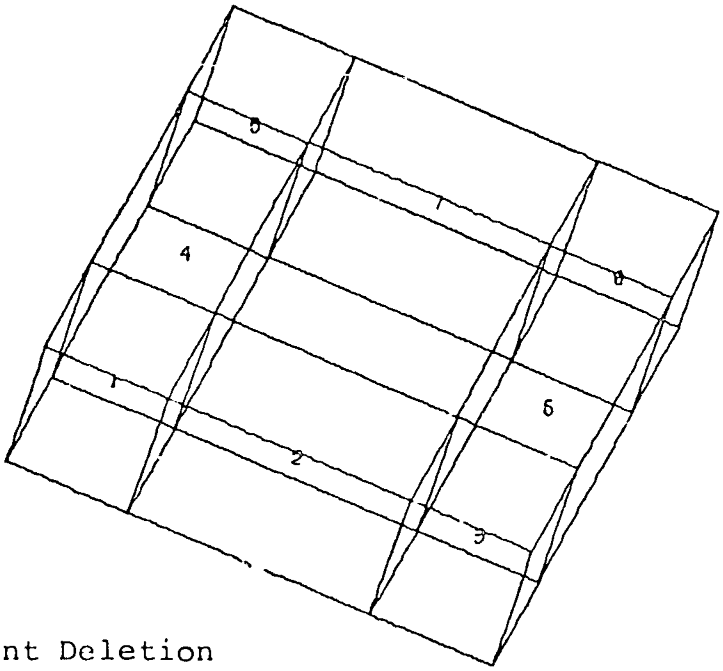
The addition of elements in the EDGAP3D code is accomplished in the same manner as in the TXGAP3D program. The command directives for this feature are identical in both programs. The directives consist of the specification of the element type, its material property type, and the connectivity of the corner nodes defining the element. The only difference between the two codes in these commands directives is that the optional features available in TXGAP3D are not applicable in the preprocessor program. This means that every corner node must be specified in the proper sequence when defining any of the element types. Since an element is defined by specifying the I,J,K (external) label of the grid points it is connected to, it is a necessary requirement that all of these grid points exist. Issuing of the command directive to generate a new element in the data list also causes the routines to build a geometry list of the nodal point coordinates for the element. If one of the specified grid points does not exist, a warning to the user is issued and the command directive to add the element is ignored.

Element deletion is accomplished by simply flagging those elements to be deleted with a key so that when the element data is regenerated on the disk file at the end of the editing session, the flagged elements are not included on the new file. For element deletion, either the element name, or element number may be used to specify the element. In cases where the element name is used, all elements with that name are flagged to be deleted.

The following examples are typical of the use of the addition and deletion features in the EDGAP 3D program. The results of the issuing of these command directives are illustrated in Figures 6a, and 6b, which show the addition and deletion of elements from the element mesh of Figure 5a.



(a) Element Addition



(b) Element Deletion

Figure 6. Element Addition and Deletion using EDGAP3D

(a) Addition of elements

```
* BRICK,1,1,4,3,2,4,3,2,5,3,1,5,3,1,4,4,2,4,4,2,5,4,1,5,4
* PRISM,1,4,2,3,5,2,3,5,2,4,4,4,3,5,4,3,5,4,4
```

(b) Deletion of elements

```
+ X DELETE,2,2,2
      ___DELETED ELEMENT      2      2      2
      ___DELETED ELEMENT      2      2      2+
    * DELETE,3,3,2
      ___DELETED ELEMENT      3      3      3
    * DELETE,11
      ___DELETED ELEMENT        11
```

+ This message was printed twice because there were two elements having the same name. Both elements were deleted from the data.

CHAPTER III

GRAPHICS DEVELOPMENT AND CAPABILITIES

While it is fairly easy to sketch or mentally visualize one and two dimensional figures, the capability to do this for arbitrarily shaped three dimensional figures is extremely difficult without some sort of aid. It becomes evident that the inclusion of a graphics section is a necessary feature when attempting to do any work with three dimensional finite elements. The purpose of the graphics section in the EDGAP 3D preprocessor program is to provide a visual aid in the form of a simple, flexible system that allows for both interactive display and hard copy output. The actual graphics features which have been included have the ability to rotate the mesh about any or all of the figures principle axes, the ability to "scale" the figure to increase the resolution of complicated areas, and the choice of several types of graphical output with respect to both the display device and the type of display desired. The flexibility in display device is limited to either interactives graphics on an IMLAC PDS-1 graphics terminal or a Calcomp ballpoint pen plotter, while the types of displays available are for the plotting of a region of the mesh that is of interest, of single elements, or of single elements highlighted in the region of interest.

The general structure of the graphics section consists of a large data block in which the information to be displayed is stored and routines containing the algorithms required to perform the various functions and operations. When the graphics section is accessed, the limits defining the region to be displayed are established. The list of element data is then examined and the information required to display the elements within the defined region is stored in the display data block (or list). The examination of the element data file is done on an element by element basis. A protection feature has been included to prevent exceeding the size of the display data block. The specifications of one of the available algorithms (rotation, scale, etc.) causes the information in the display data block to be operated on and in some cases to be modified.

In the development of any graphical display system, it is soon discovered that the available equipment will impose major restrictions on the display capabilities. The graphics section associated with EDGAP 3D has certain machine limitations related to it. The particular graphics terminals involved in the interactive display of the input data are IMLAC PDS-1 refresher scopes. A refresher scope is a cathode ray tube (CRT) device coupled with a memory bank in which the information related the display is stored. The term "refresher" scope comes from the fact that the memory bank must be cycled through several times a second (40 times) to keep the display from fading from the screen.[5] The IMLAC terminal is basically a mini-computer with an 8000 word memory. When coupled with the CDC 6000 series computer, however, the first 4000 words of storage are reserved for the IMLAC's operating system and teletypewriter functions. This leaves only the last 4000 words for graphics display. With long vector hardware, this allows for the display of slightly more than 1300 vectors (both visible and invisible) at any one time, since three words of memory are required for every vector. A vector is defined as a line between two points.

Considering the machine limitations, the size of the display data block in the graphics section has been limited to 1000 vectors. This display data block size was selected because it provides an adequate buffer for problems which might produce an IMLAC display memory overflow. By setting the display data block size at a 1000 vectors, it is possible to retain this information in the CDC 6000 series computer's memory without sacrificing interactive response time or having to resort to some form of INPUT/OUTPUT operation when manipulating the display data. With the vector display capability set at this limit, it is possible to display up to approximately 35 brick elements (which have the largest number of vectors per element), or a greater number of prismatic and tetrahedronal elements, which have decreasingly fewer vectors.

A particular features used in the development of the graphics section code is that all the actual drawing instructions are issued by standard Calcomp plotting calls. The interfacing of the Calcomp calls with the IMLAC graphics terminals is accomplished by several undocumented routines developed on the University of Texas at Austin computer system. These routines translate the Calcomp instructions into IMLAC machine language which is then sent to the graphics terminal by the use of a high speed binary Input/Output file. The specification of the type of output device determines the calling of these routines.

Rotation Algorithm Development

The algorithm developed to perform the rotation of the mesh about its' axes represents the transformation of a right-handed Cartesian coordinate system about its axes. First considering a rotation about the x-coordinate axis, it can be shown that the transformation to a new, primed reference frame is expressed in matrix form as

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

where the angle of rotation, θ, is measured as being positive in a clockwise direction about the x-axis when viewing the origin from a point located on the plus x-axis (reference Figure 7). Similarly, a rotation about only the y coordinate axis is expressed as

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

where the rotation angle ϕ is about the y-axis (see Figure 7). The rotation transformation for the z-coordinate axis is

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

where the rotation angle ψ is measured about the z-axis (see Figure 7).[5]

The approach taken in problems of rotations about multiple axes is to consider this situation to be of a sequential type in which the rotation about first one axis is

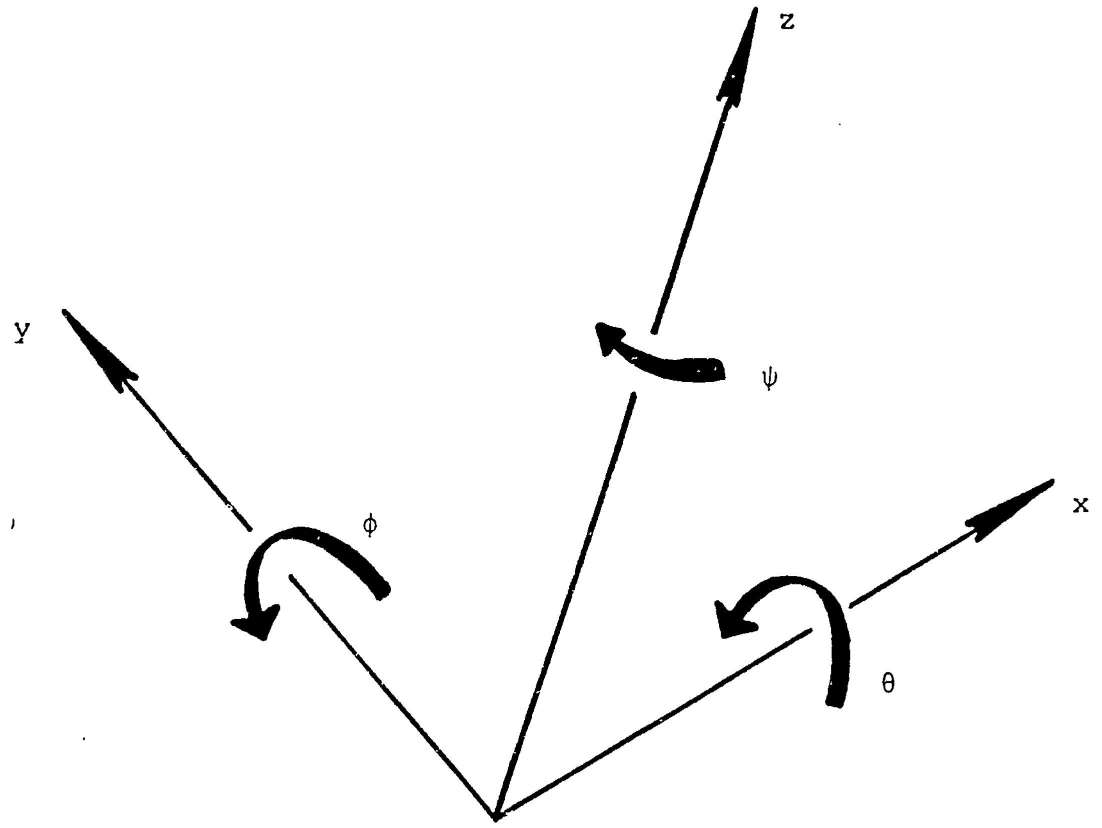


Figure 7. Definition of Rotation Angles

performed, transforming the system to a new orientation about which the next rotation is carried out. The problem that exists with this technique is that the order in which the sequential rotations are carried out becomes important. That is, for example, that the coordinate transformation about the x and then the y-coordinate axis does not produce the same results as that about the y and then the x-coordinate axis. The algorithm developed in conjunction with the graphics section in EDGAP3D assumes that for multiple transformations, the sequential ordering is an x, then y, then z-coordinate axis transformation. This sequential transformation is expressed mathematically in matrix form as

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

Noting that if no rotation is desired or specified about any of the coordinate axes, the coordinate transformation matrix associated with that axis reduces to the identity matrix.

Therefore, this same system of equations may be used to produce any of the rotation combinations tabulated in Table 1.

TABLE 1

SEQUENTIAL ROTATION COMBINATIONS
x
y
z
x, then y
x, then z
y, then z
x, then y, then z

If any other ordering of the coordinate transformations is desired, the same algorithm is used, but the coordinate transformation order is changed by specifying separately, one angle at a time, the coordinate axis rotation in the order desired.

Rotation of the coordinates defining the mesh can produce a shifting of data points to locations outside of the originally defined limits of the display region. In order to display the total region after the rotation of the mesh, it is necessary to establish new limits defining the display region. This is done by examining the transformed display data list and locating the maximum and minimum values on each coordinate axis. The various combinations of these maximum and minimum values are then used to define the new limits of the display region.

Scaling Algorithm Development

The graphical presentation of data often results in areas which are extremely detailed and from which it is difficult to discern meaningful information. A scaling feature helps eliminate this problem by providing the capability of enlarging, or increasing the scale of these detailed regions with respect to the display area (i.e., by blowing up the region of interest). The approach taken in the development of the scaling algorithms, which is facilitated by working with a data block of display information, is to scale all the display information. The effect of scaling is to increase (or decrease) the size of the mesh. The limits originally established to define the region of interest are used to calculate the factors for fitting the region on the screen. By retaining these limits when the display data is scaled and displaying only the data still lying within these limits, the resolution of the figure is increased (or decreased) according to the manitude of the scale factor. The scaled data falling outside the screen limits are not displayed.

It is possible after having scaled the display information that the area in which the greater resolution is desired now falls outside the display limits. A feature which allows the translation of the mesh's origin along lines parallel to the coordinate axes makes it-possible to shift these detailed regions back into the display area. When the mesh has been rotated from its original reference frame, the shift variables are automatically transformed to the new reference frame before translating the origin. The shift variable transformation is based on the same alogrithms described in the rotation feature development. This means that the sequential rotation ordering is assumed, and in this particular instance it is not possible to alter the order of the sequence from that described in Table 1. A two dimensional representation of the scale-shifting technique is illustrated in Figures 8a through 8d.

The particular algorithms used to perform the scaling and origin translation are

$$x_{st} = (x - x_t)\ F$$
$$y_{st} = (y - y_t)\ F$$
$$z_{st} = (z - z_t)\ F$$

where F is the scale factor and is always greater than zero. The variables subscripted by t are the translation magnitudes

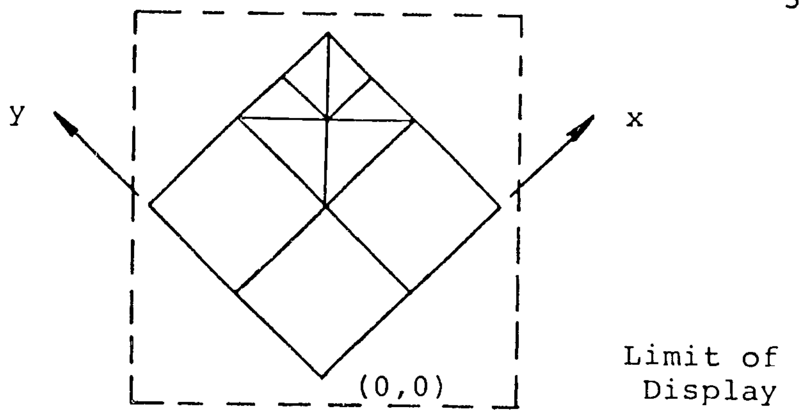


Figure 8a. Original Unscaled and Untranslated Region of Element Mesh

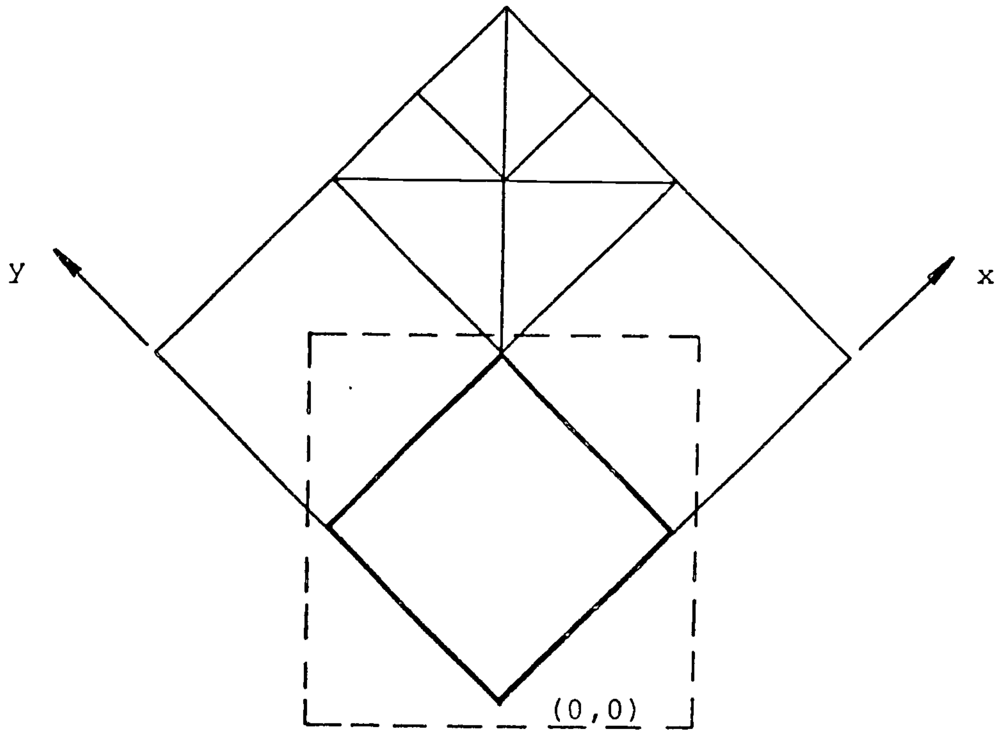


Figure 8b. Untranslated Region Scaled by a Factor of Two

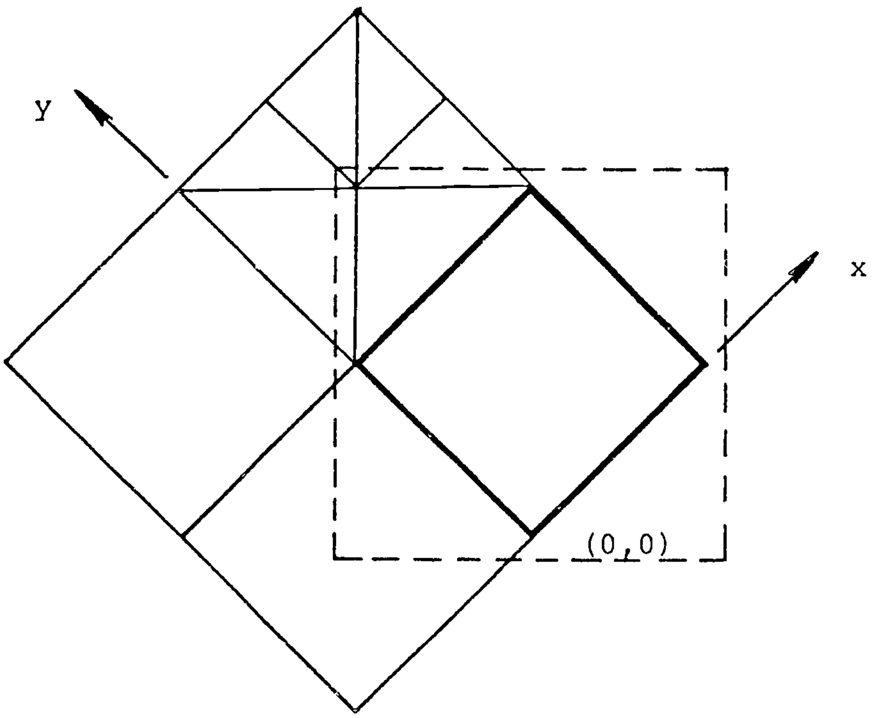


Figure 8c. Scaled Region Translated Along x-Axis

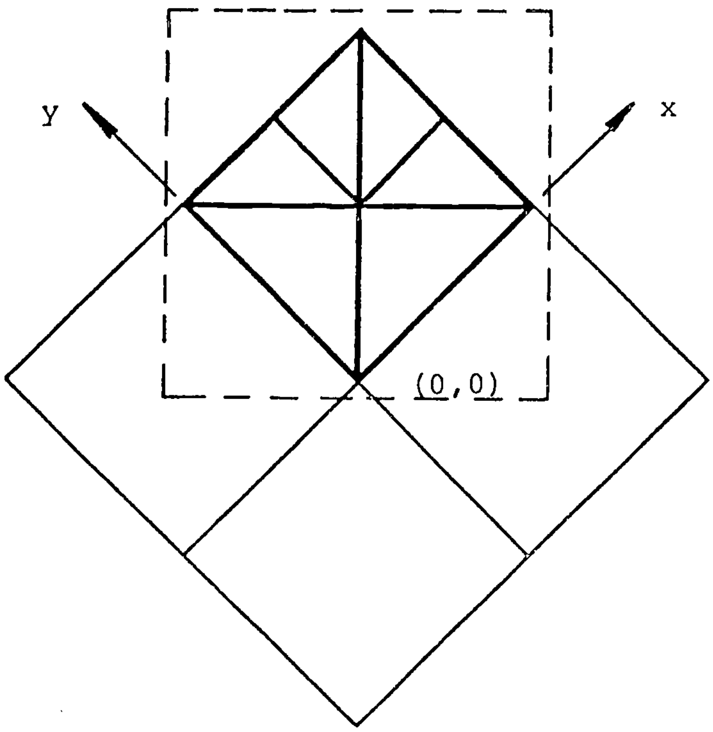


Figure 8d. Scaled Region Translated Along Both x and y-Axes

along the appropriate axes, and the st subscripted variables are the new, scaled and shifted coordinates stored in the display memory.

Display Feature Development

The representation of a three dimensional figure on a two dimensional surface requires the development of several algorithms. One of these algorithms involves the actual mapping from the three dimensional system to the two dimensional system and the other involves the problem of devising a system by which the figure is scaled and shifted with respect to the screen in order to maximize the amount of screen used. Another aspect of plotting figures is the issuing of pen instructions. The graphical display of three dimensional meshes developed here represents an isometric projection of the mesh.

The transformation mapping from a three dimensional coordinate system into a two dimensional screen coordinate system can be deduced from Figure 9 in which the screen coordinates are determined to be the combination of the geometric projections onto the screen coordinate axes. The actual algorithms are expressed by

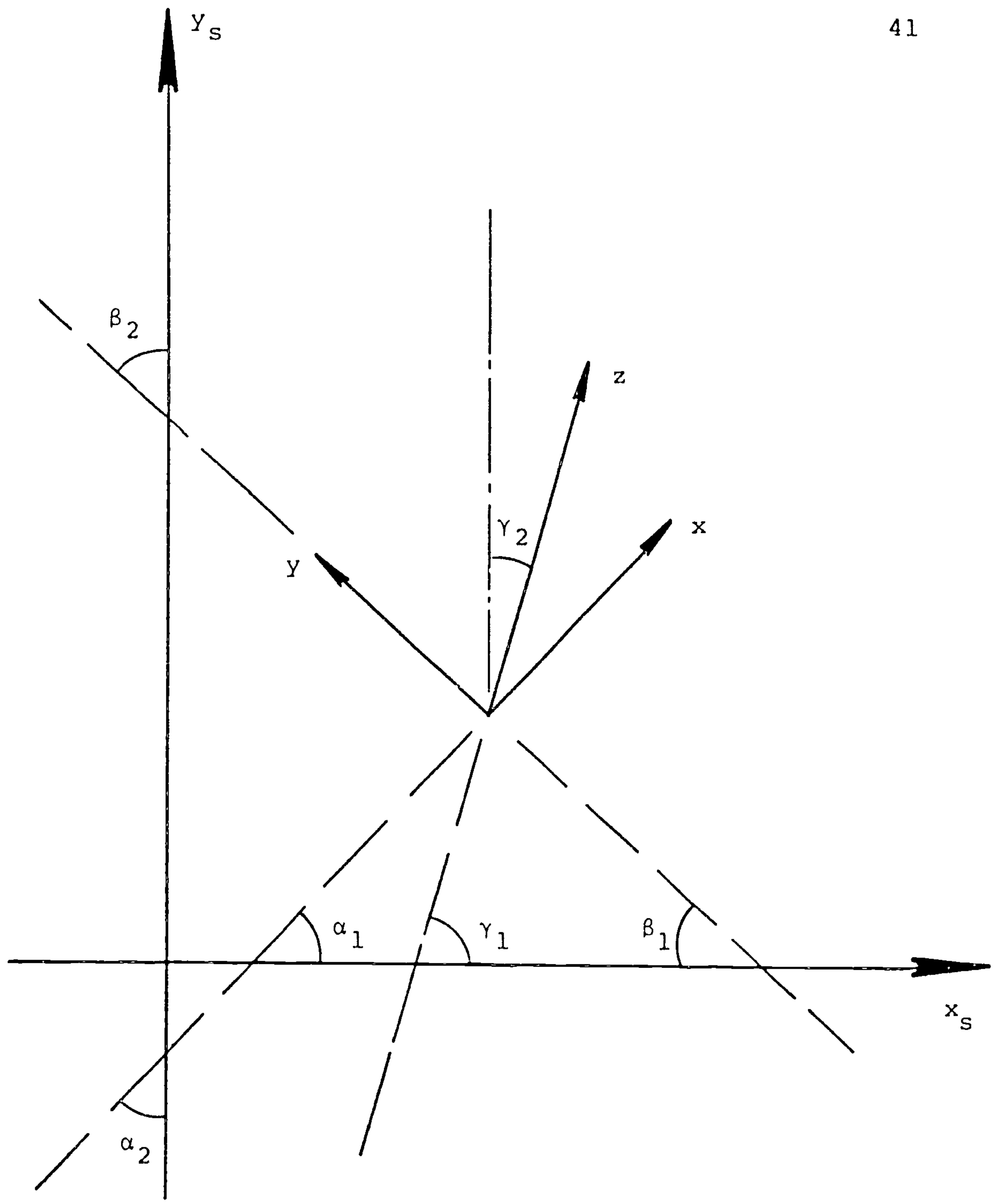


Figure 9. Definition of Projection Angles

$$x_s = x \cos\alpha_1 + y \cos\beta_1 + z \cos\gamma_1 - c_1$$

1)

$$y_s = x \cos\alpha_2 + y \cos\beta_2 + z \cos\gamma_2 - c_2$$

where x_s and y_s are the screen coordinates measured in inches from the lower lefthand corner of the screen with the x_s axis being horizontal and the y_s axis being vertical. The subscripts on the angles (1 & 2) reference the angles between the figures coordinate axes and the screen's x_s and y_s coordinate axes respectively. The variables c_1 and c_2 represent the magnitude in inches which the mesh must be shifted along the x_s and y_s axis respectively in order to maximize the usage of the display area. These shift variables are defined in Figure 10. The computations to be made are the determination of the cosines of the angles α_i, β_i, and γ_i. Consider a three dimensional reference vector set composed of {(1,0,0), (0,1,0), (0,0,1)} where the components are (x, y, z). These vectors can be shown to form an orthogonal basis for a three dimensional space, and will be referred to as $\{\overline{V}_{R1}, \overline{V}_{R2}, \overline{V}_{R3}\}$. Performing all the rotations on this basis set that are carried out on the display data then results in a new basis set $\{\overline{V}_{R1}', \overline{V}_{R2}', \overline{V}_{R3}'\}$ from which the cosine of the angles required for the transformation from the 3-D system to the 2-D surface can be determined. This is done by introducing

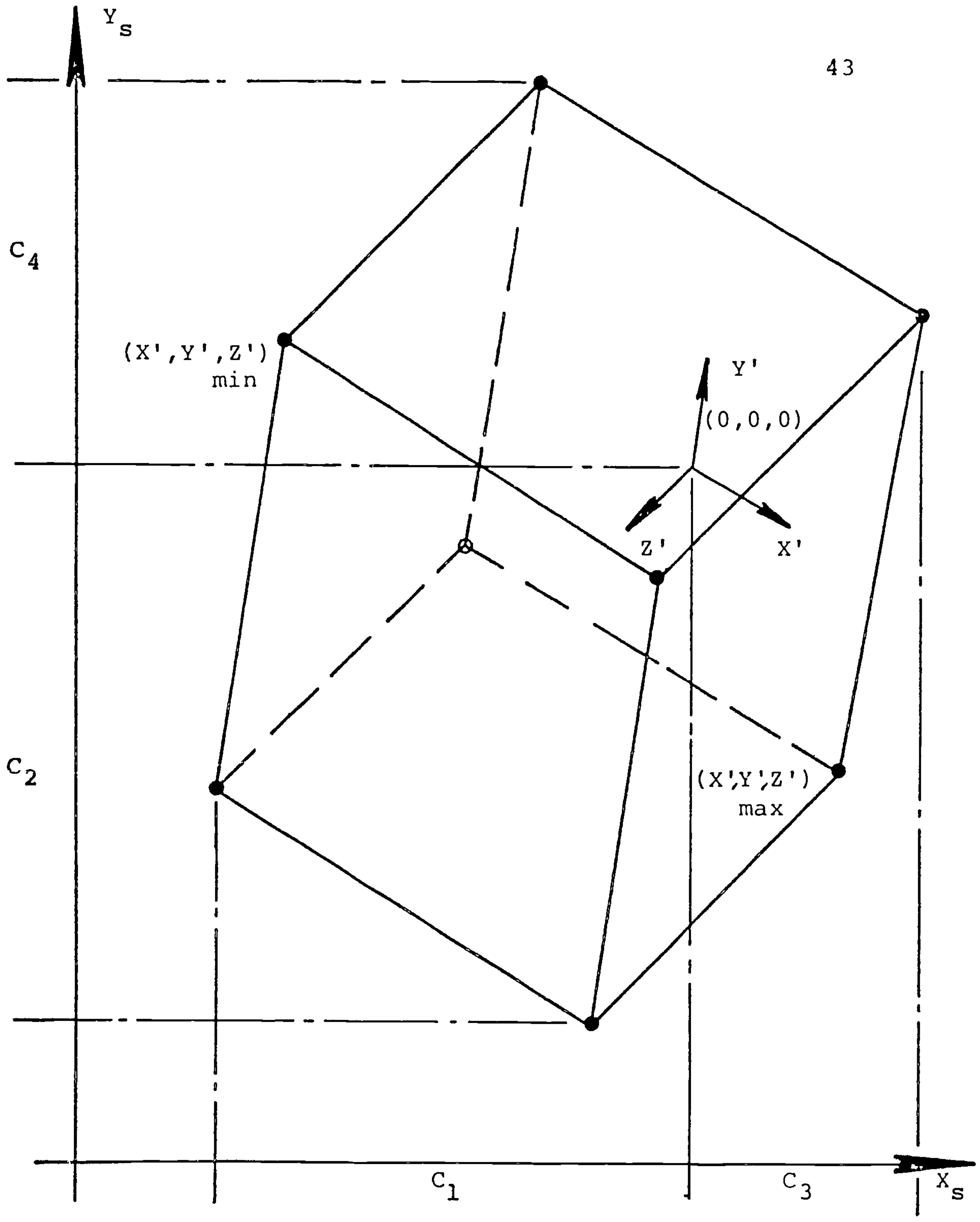


Figure 10. Display Screen Scaling Box Definition

a display vector basis set $\{\overline{V}_{D1}, \overline{V}_{D2}\}$ which consists of $\{(1,0,0), (0,1,0)\}$, and recalling the relationships for the inner products of vectors.

$$\overline{A}\cdot\overline{B} = ab\cos\theta = a_1b_1 + a_2b_2 + a_3b_3 \qquad 2)$$

where a_i and b_i are the vectors components and the scalar magnitudes are

$$a = (a_1{}^2 + a_3{}^3)^{1/2}\ ,\ b = (b_1{}^2 + b_2{}^2 + b_3{}^3)^{1/2}$$

and solving equation 2 for the cosine of the angle yields

$$\cos\theta = (a_1b_1 + a_2b_2 + a_3b_3)/ab \qquad 3)$$

An examination of the basis sets for both the reference vectors, V_{Ri}, and the display vectors, V_{Di}, quickly reveals that their scalar magnitudes are always equal to unity. Therefore, relation 3 reduces to the form, when expressed in terms of these vectors, of

$$\cos\theta = \overline{V}'_{Ri}\cdot\overline{V}_{Dj} = v'_{Ri1}v_{Dj1} + v'_{Ri2}v_{Dj2} \quad \begin{matrix} i=1,2,3 \\ j-1,2 \end{matrix} \qquad 4)$$

Note that v_{Dj3} is always equal to zero and this term drops out. By using relation 4) the cosines of the α_j, β_j, γ_j, angles are determined as

$$\cos\alpha_1 = \bar{V}'_{R1}\cdot\bar{V}_{D1} \qquad \cos\alpha_2 = \bar{V}'_{R1}\cdot\bar{V}_{D2}$$

$$\cos\beta_1 = \bar{V}'_{R2}\cdot\bar{V}_{D1} \qquad \cos\beta_2 = \bar{V}'_{R2}\cdot\bar{V}_{D2}$$

$$\cos\gamma_1 = \bar{V}'_{R3}\cdot\bar{V}_{D1} \qquad \cos\gamma_2 = \bar{V}'_{R3}\cdot\bar{V}_{D2}$$

In order to maximize the display of a mesh within the limits of the screen while insuring that all of the mesh be displayed, the following scheme was devised. All the display data is examined after each transformation to determine the maximum and minimum values in each of the principal directions, x', y', and z', where the prime denotes the transformed coordinate system. The various combinations of these values define a box which contains the mesh region to be displayed (see Figure D.3). By transforming these combinations into the screen coordinate system, the maximum and minimum projections on both the screen coordinate axes are determined. The minimum values, c_1 and c_2, define the translations required to shift the mesh enough to assure the total image lies within the screen limits. The maximum difference between the maximum and minimum values on either of the screen axes (i.e., the maximum of $c_4 - c_2$ or $c_3 - c_1$) is used to establish the screen scale factor by which the display data is multiplied before plotting. The screen scale factor is a function of the screen width and this maximum difference. In Figure 10, the origin of the transformed coordinate system is shown to lie within the maximum-minimum box. This is not a requirement of the algorithms and poses no problems if it does not occur.

The actual plotting of a mesh presents two problems which must be solved. One is to minimize the number of vectors required to draw an element without creating extraneous lines or pen moments in the display, and the other is to design the plotting package so that only one vector is drawn between two nodes no matter how many elements may join these nodes. The last problem arises from the fact that multiply drawn lines are brighter on the graphics screen and darker on the Calcomp plots, tending to accentuate these lines and making it difficult to distinguish such things as the image's depth. Both of the problems described above are associated with the pen position during the plotting of the figure. In the pen up position, an invisible vector (no line) is drawn and with the pen down a visible vector (line) is drawn. By using combinations of pen positions, the number of lines required to draw each type of element can be determined and stored in the program. As the element data is examined at the beginning of a graphics session, the pen instructions associated with the elements which are stored in the display data list are also saved. This provides the drawing instructions needed to display individual elements. When multiple elements are to be displayed, a looping feature is entered which causes a comparison to be made between the vector about to be drawn and all previously drawn vectors in the display list. If the vector has already been drawn, the pen instruction for the present vector is set at up and the pen is more to the next point without drawing the line. Invisible vectors are used to reduce the amount of lines actually drawn, but they do not decrease the amount of information required by the graphics terminals to display the figure. This is because the same number of words of memory are required to draw both types of vectors.

Illustrative Example of Graphics

The following example illustrates the use of the various features available in the graphics section of EDGAP3D. The parameters associated with the DISPLAY command indicates that the output file is the IMLAC graphics terminals. Of course the figures which appear were actually generated on the Calcomp plotter.

* PLOTS,_1,_1,_1,50,50,50
  defines the plotting region as being from (_1,_1,_1) to (50,50,50) inches

* ROTATE,
  rotate the element information about the x-axis and then about the z-axis.

* DISPLAY
  display the region specified by PLOTS; see Figure 11

* DISPLAY,2, ,4
  display element number four, see Figure 12

* DISPLAY,3,1,20
  display element number four highlighted in the region defined by PLOTS, see Figure 13

* SCALE,1.7
  scale the display information by a factor of 1.75

* DISPLAY
  display the scaled display information, see Figure 14

* SCALE,,17,17,
  translate the origin 17 inches along the x-axis, 17 inches along the y-axis, and inches along the z-axis

* DISPLAY
  Figure 15 shows the results of the origin translation with the previously scaled display information

* END,PLOTS
  terminates the graphics module.

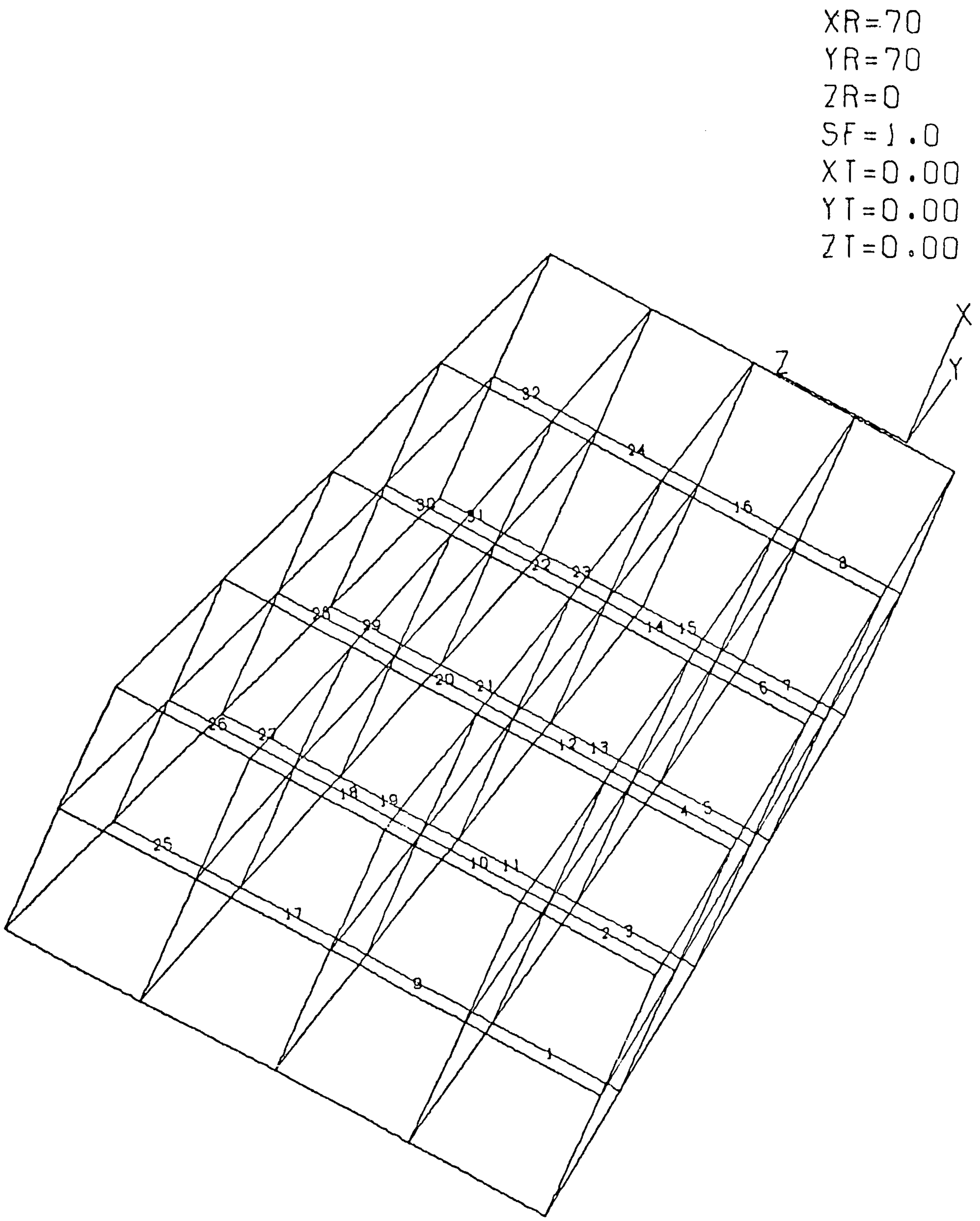


Figure 11. Element Mesh in Region Specified

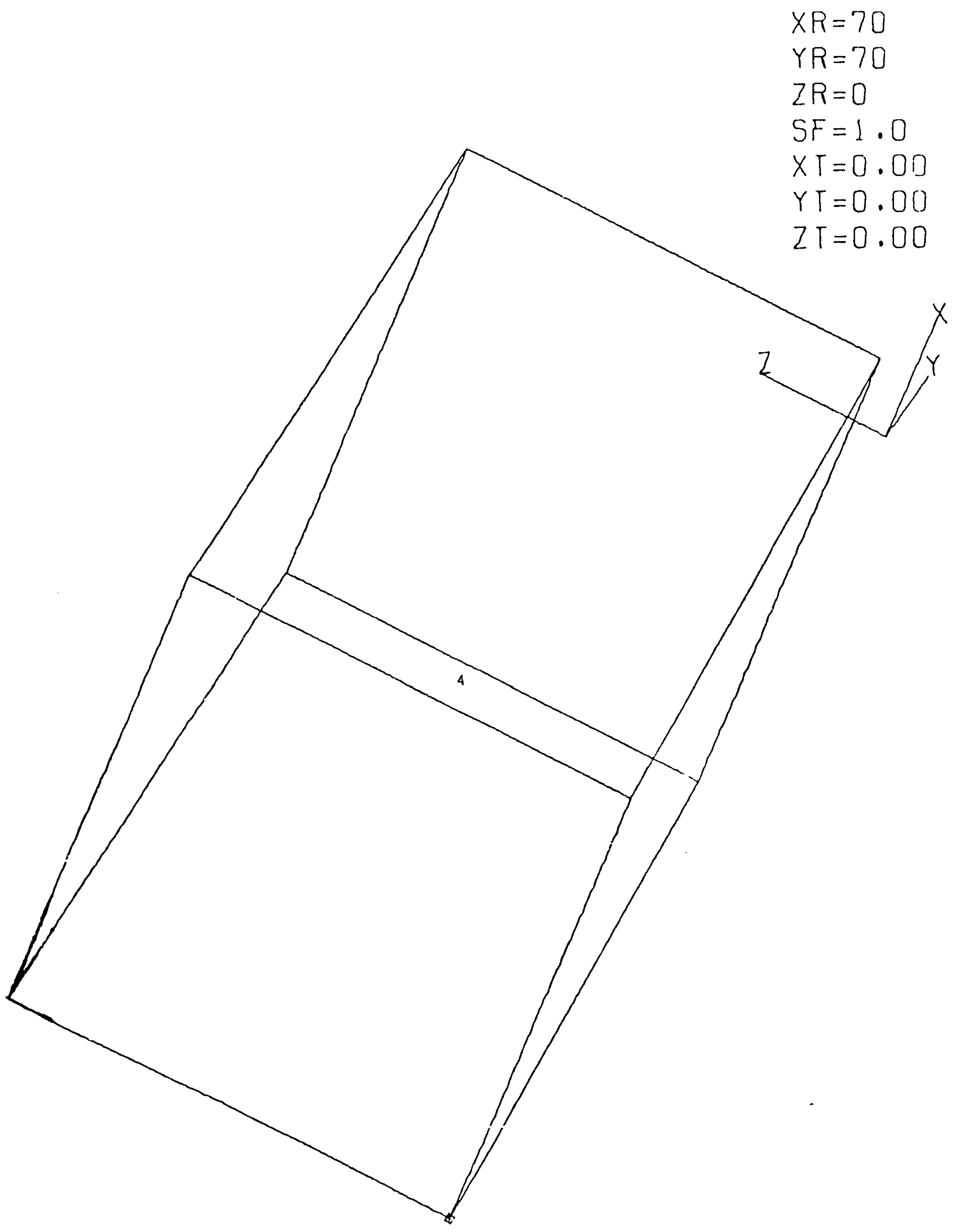


Figure 12. Element Plot

XR=70
YR=70
ZR=0
SF=1.0
XT=0.00
YT=0.00
ZT=0.00

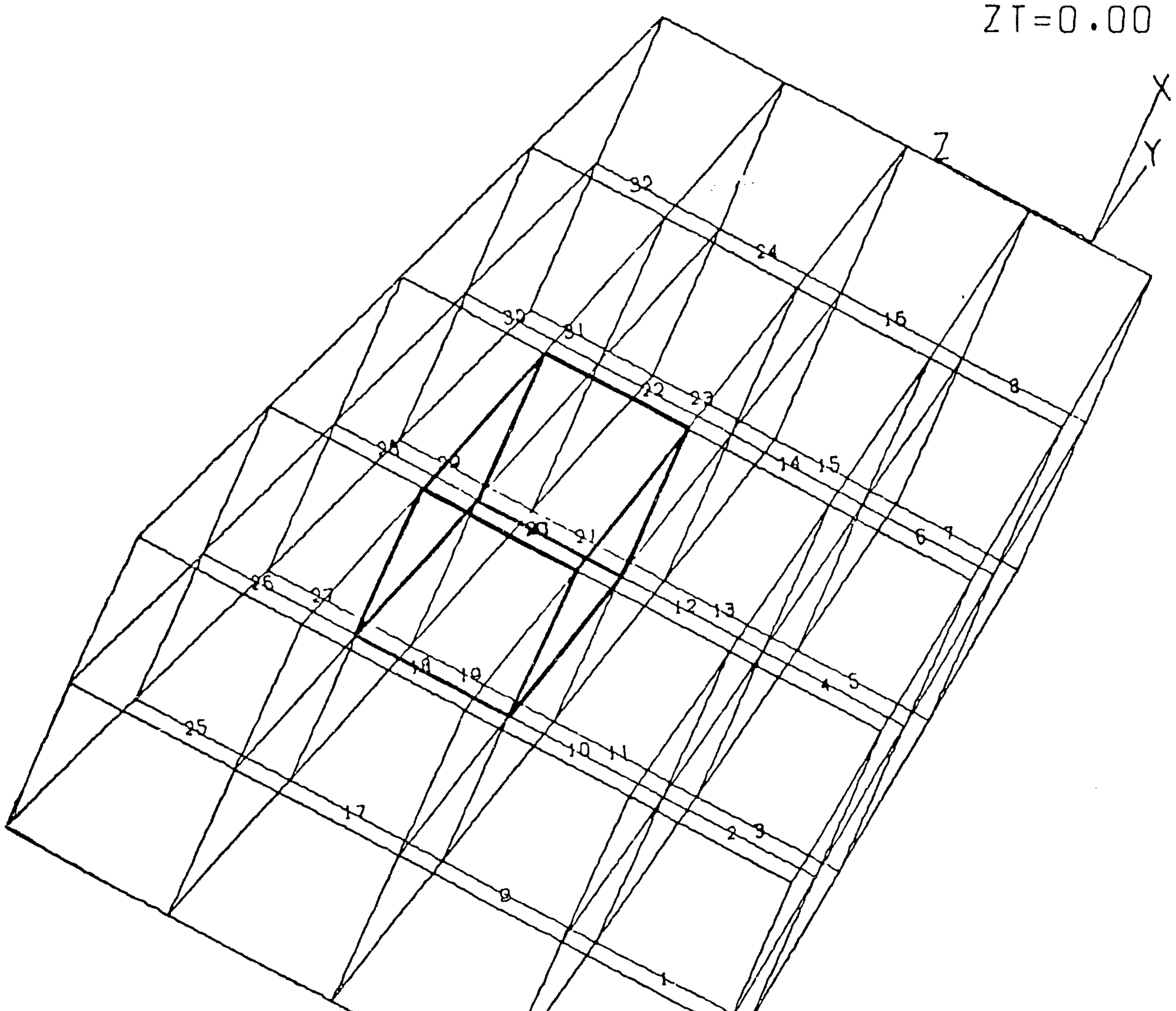


Figure 13. Element Highlighted in Specified Region

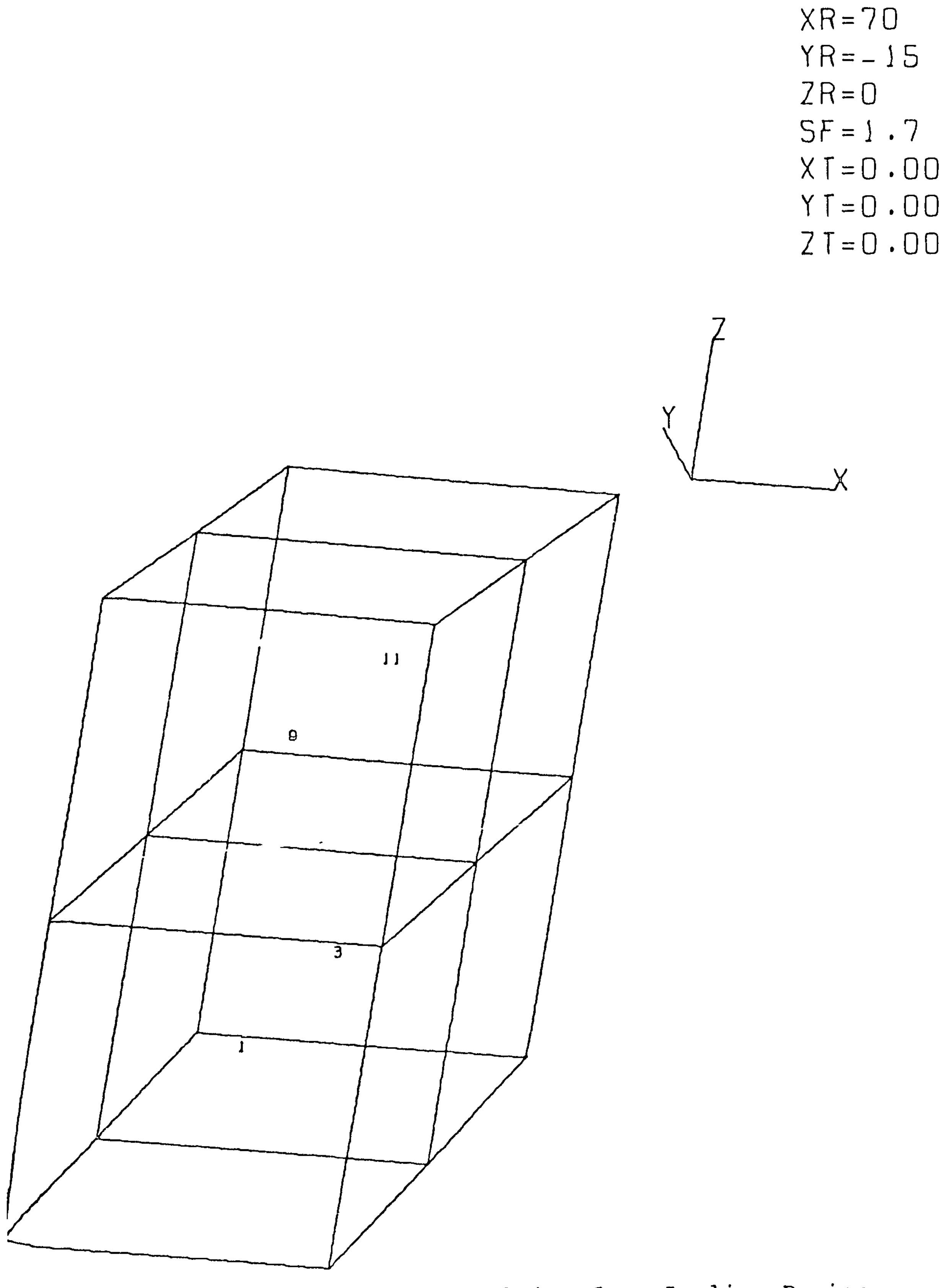


Figure 14. Element Mesh Resulting from Scaling Region

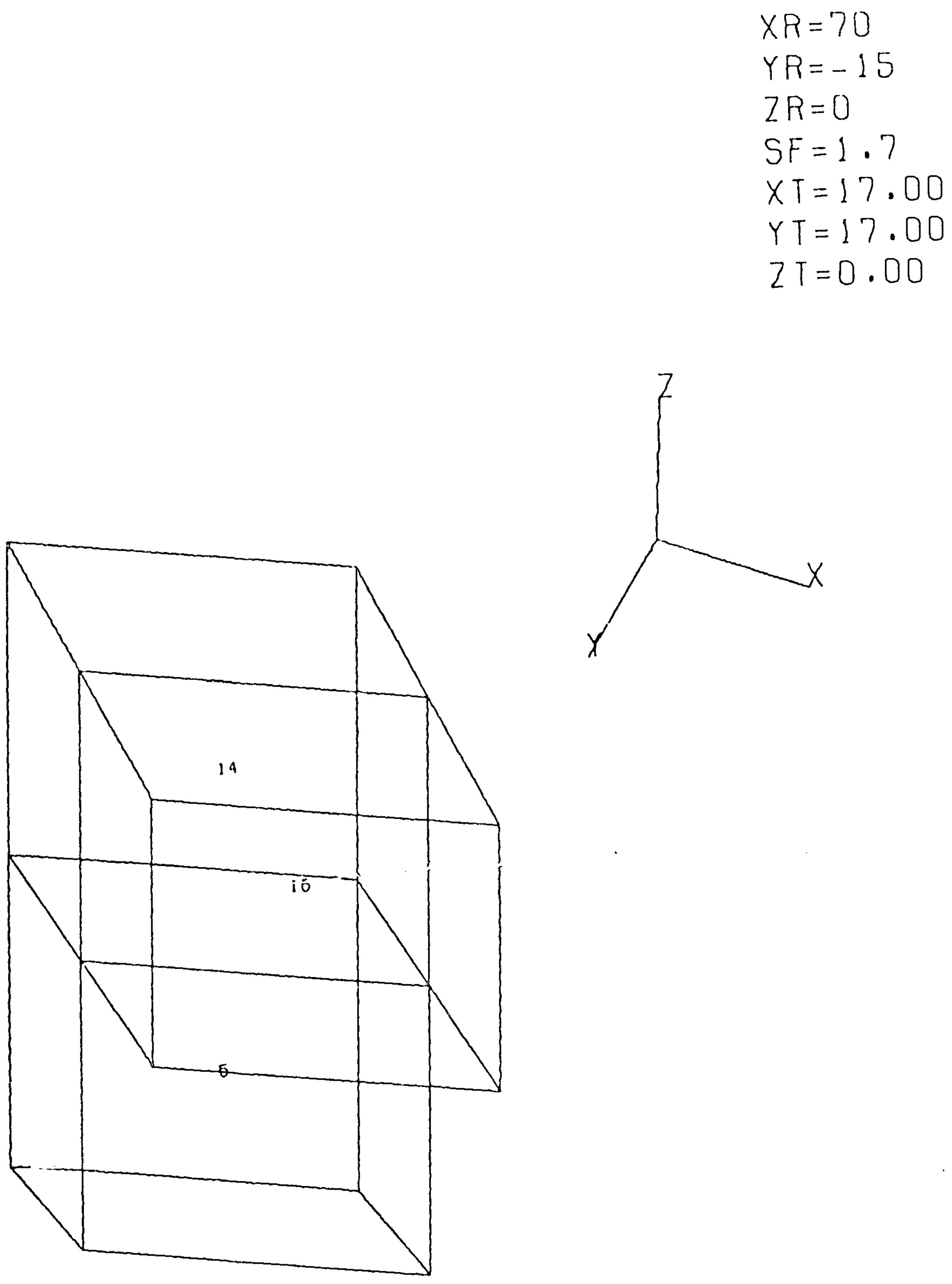


Figure 15. Element Mesh Resulting from Scaling and Translating Region

CHAPTER IV

CONCLUSIONS

The EDGAP3D preprocessor program developed in this research provides a means for interactively verifying and correcting data generated by the three dimensional finite element code, TXGAP3D. The principle limitation of the code is that it is restricted to handling of existing data since it has no data generation capabilities. It is quite likely that situations can arise where verification and correction of data is insufficient and some form of data generation would be desirable. This is particularly true in the areas of geometry and element addition. Another capability which should be included is that of handling material properties, thus eliminating the need for complete regeneration of the data when only the material properties of the problem have been changed. Another area in which this particular preprocessor could be improved is to include the capacity to examine and modify data generated by the REZONE program of TXGAP3D. The capability to handle this information would be very valuable. While the area of graphics could be improved with such features as hidden line capability, it is felt that the trade off in computer core requirements does not warrant the effort. One of the best possible ways to improve the EDGAP3D program would be to more closely couple the data generator in TXGAP3D with it. For example, to run the data generator (SETUP) and EDGAP3D linked directly together as a single program.

APPENDIX A

USER'S INPUT GUIDE TO EDGAP3D

EDGAP3D operates in two basic command modes, depending on the function to be performed. These are the editing mode which features two levels of command entries, and the graphics mode which has only one command level. An attempt has been made to have the available input commands follow the format of the commands of TXGAP3D. The following convention and terminology is adhered to in the description of the input entries.

i.) UPPER CASE words are actual alphanumeric input as they appear in the entries, e.g., EDIT.

ii.) Lower case words are variable names whose values appear in the entries, e.g., imax.

iii.) All entries are in the free field format, i.e., individual words are separated by commas. A maximum of ten, nonblank, characters is allowed in any one data field (blanks are ignored). The content of each kind of entry is shown underlined.

iv.) Optional parameters will be designated by [ ]. The omission of an optional parameter within a sequence of parameters is indicated by successive commas. If the omission is not followed by any parameters to be specified, the commas are not required.

v.) Elements are normally identified by the i, j, k number of the first node that the element is connected to, here after referred to as the element name. In EDGAP3D, the additional capability of identifying an element by the order in which it appears on the element data tape, here after referred to as the element number.

vi.) In the generation of the grids in the present version of TXGAP3D, the user specifies the i, j, k names (numbers) of every other grid point. This will be referred to as the external node name. It is these nodes which are used to define the elements and correspond to corner nodes of each element. The grid points which lie between the externally named points have no external node name. All grid points have an internal node name which is computed by I*2-1, etc., for the named points and I*2, etc., for the unnamed points.

The use of EDGAP3D is dependent on data generated by the SETUP portion of TXGAP3D with a SAVE command being issued during some portion of TXGAP3D input. The SETUP-SAVE features causes the generation of three files containing pertinent data. These files are TAPE13 (number of elements), TAPE12 (element data), and TAPE15 (grid point coordinates). All of these files are assumed to be in the local file area of the user's job at the time EDGAP3D is executed.

Command Mode Entries

Command mode entries direct the calling sequence to the specific overlay to perform the prescribed job steps in the editing or graphics modes. The following commands are possible: EDIT, PLOTS, END, and STOP. The issuance of the command EDIT, PLOTS, and END in general cause some preliminary preparatory operations to be carried out in conjunction with the particular command. Since there is no unique sequence of steps for the program, each command mode with its associated features will be described separately.

Editing

EDIT

This overlay allows for the verification and modification of the element data (TAPE12) and grid point data (TAPE15) generated by the TXGAP3D program. The features available include the ability to check and change grid point locations and element connectivity, as well as the addition and deletion of elements. Boundary condition data may also be verified, modified, added to, or deleted. Note that the ability to edit material properties has not been included, although an element material type specification may be changed by first deleting the element and then adding it back with the new material type being specified.

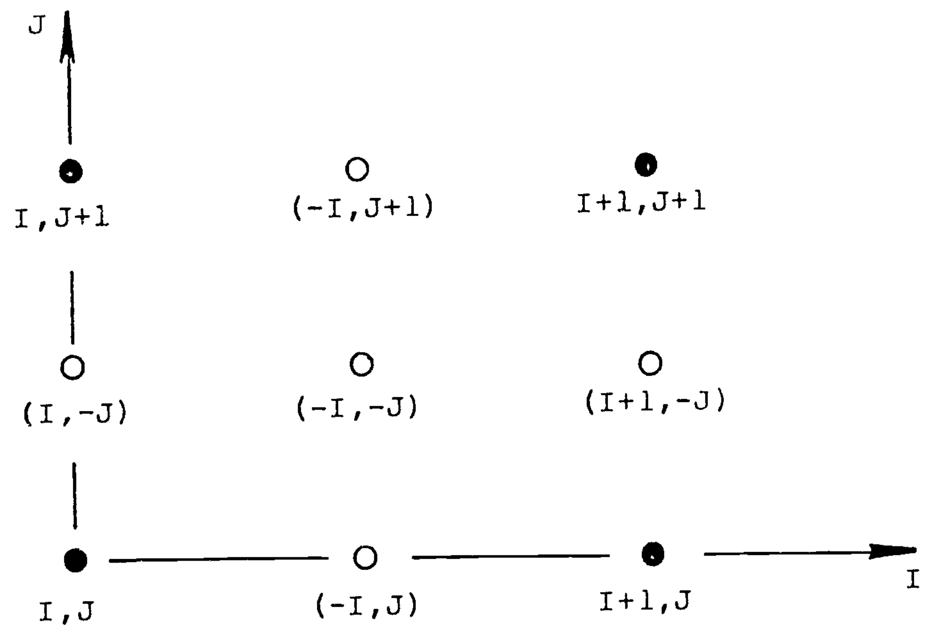
The issuance of the EDIT command causes the reading of the element data file and the storage of pertinent data. An EDIT session must be terminated by an END command which causes the regeneration of the element data file with the corrected data.

As element names are identified by the i, j, k number of the first node that the element is connected to, it is possible to have several elements with the <u>same</u> name. It is desirable to avoid a situation like this because boundary conditions (in TXGAP3D) are applied to elements according to their element name and it is therefore possible to apply boundary conditions on elements other than the one intended. If this situation does arise, however, there is no particular problem with EDGAP3D, since it is structured to locate and allow the modification of elements in a sequential fashion, even when multiple element names occur. In the case of multiple element names, the first element is found and when modifications (if any) to it are completed, the routines automatically proceeds to the next element with that name. In cases where only verification information is requested, the program is designed to provide the data by element for <u>all</u> the elements with the same name.

Grid Point Features

```
GRID,i,j,k,[MOVE,newx,newy,newz]
```

i,j,k = external i,j,k name of the grid point. For grid points which have no external name, the following convention is used. Specify the negative of the number of the next, named, grid point in the direction of the origin (moving along i,j, or k equal a constant lines) from the point of interest. Refer to the 2-D figure below as an illustrative example.



The solid points ( ) indicate grid points with external names. The circle points (O) indicate grid points without external names. The names in parenthesis indicates the names to be specified to check and move unnamed points.

MOVE causes the modification of the location of the grid point named.

newx,newy,newz = the x,y,z location to which the named grid point is to be moved.

When the GRID command is used with only the grid point identifiers, the present location and the internal name of the grid point is returned. If the MOVE option is included, the internal grid point name and its original location is returned, then the points location is modified.

Element Connectivity Features

CONNECT,type,i,[j,k]

type = NODE,ELEMENT, or MODIFY

i,[j,k] = external node number (NODE), element name or element number (ELEMENT or MODIFY). Note that when type is specified as NODE, i,j,k values must be specified and they refer to the external node name of the element corner nodes only.

For "type" specified as NODE, the program is designed to output all the elements connected to the specified node in terms of element names and the associated element nodal point number. If "type" is specified as ELEMENT, the program outputs the element nodal point numbers and the external node names associated with these for all elements with the same name. When modifying an elements connectivity, the present connectivity of the specified element is first returned and then a looping call is made to the data entry, which has the following form.

<u>node,newi,newj,newk</u>

node = element corner node number whose connectivity is to be modified

newi,newj,newk = the new external node name of the corner node

The modification to the element's connectivity is continued until the command, FINISH, is entered in place of the data entry. The routine will then perform a search for the next element with the same element name, if an element name was originally with MODIFY, otherwise the next editing command may be entered. If no modification is desired after having specified "type" as MODIFY, then enter FINISH and no changes will be made.

Element Deletion & Generation Features

<u>DELETE,i,[j,k]</u>

i,[j,k] = element number or element name

The issuance of this command cases the element specified to be flagged for deletion from the element data tape at the end of the editing session. Note that if the element <u>name</u> feature is used, that <u>all</u> elements with this name will be flagged for deletion.

<u>eltype,mat,il,jl,kl,i2,......,iN,jN,kN</u>

eltype = BRICK,BRICKH,PRISM,PRISMH,TETRA, or TETRAH

mat = material type number

il,....,kN = the i,j,k's of the corner nodes defining the element. Note that all this information must be specified, and N=8 for brick shaped elements, N=6 for prismatic elements, and N=4 for tetrahedron shape elements.

The format for the addition of elements is exactly the same as in TXGAP3D with the exception that all the data must be specified. This command causes the generation of a new element without boundary conditions applied to it which is added to the end of the element data list.

Boundary Condition Features

BC,oper or type,[i],[j,k]

oper = ELEMENT,ADD, or DELETE

type = UX,UY,UZ,FX,FY,FZ,PRESSURE, or SLOPE

[i],[j,k] = element number or element name and is functional only in conjunction with an operation.

The specification of a boundary condition type causes the output of all boundary conditions of that type. This is done in terms of the element name, the node or face on which the boundary condition is applied and the boundary condition value. When the operation, ELEMENT, is specified, all the boundary conditions applied to the element(s) designated by the element name or element number are printed. The output is in the form of the boundary condition type, the node or face on which it is applied, and its value.

The addition, deletion, and modification of boundary conditions is carried out through the operation of ADD or DELETE. The use of the ADD and DELETE features first causes the return of the existing boundary conditions for the element specified, followed by an extra data entry of the form specified below.

type, node or face, value, [value,value,value]

node or face = node or face number of the element [see figures 7 through 9 in Ref [1]

value = the value of the boundary condition

The PRESSURE boundary condition is specified by four values (three for a triangular face) or pressure at the corner nodes of the face. The ordering is the same as specified in TXGAP3D USER'S INPUT GUIDE, and for constant pressures only the first value need be specified. To change the value of a boundary condition, the ADD command is used in conjunction with the data entry specifying the new value of the boundary condition. For the deletion of boundary conditions, the extra data entry need only specify the boundary condition and the node or face.

If after having entered the boundary condition modification mode (ADD or DELETE) it is desired not to carry out the modification, or in the case of multiple elements with the same element name where selective modification may be desired, the command FINISH may be substituted for the data entry in order to proceed without making changes.

Graphics

PLOTS,[xmin,ymin,zmin,xmax,ymax,zmax],[imin,jmin,kmin,imax,jmax,kmax]

xmin,....,zmax =	the minimum and maximum x,y,z coordinates of the region to be displayed
imin,....,kmax =	the minimum and maximum i,j,k numbers of the region to be displayed

The overlay called by the PLOTS command allows for graphical output in the form of IMLAC displays or CALCOMP ball point pen plots for the region specified in the call. The information required to display element mesh is read element by element from the element data file with only the elements in the specified region being retained in the display memory. Since the amount of display memory allocated for the storage of this information is limited, a protect feature is built in which prevents overflowing of the memory area. The information contained in the display memory can be manipulated and displayed, even if an overflow is encountered. To view information which is not contained in the display memory because of overflow or otherwise, a new call to PLOTS may be issued during a graphics session, specifying a new region of interest. Note that this can be done without having terminated the graphics session.

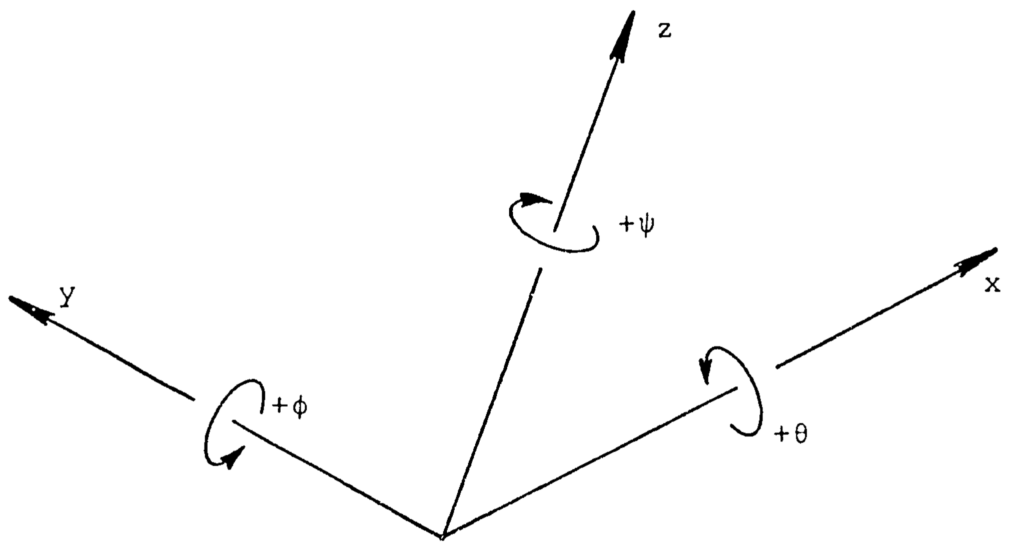
The graphics features in this overlay are limited to three basic commands, some of which have variable parameters. These commands produce the rotation, scaling, and displaying of the data in the display memory. Graphics sessions must be terminated by an END command entry.

The IMLAC PDS-1 graphics terminals are basically mini computers which require their own operating system to interpret instructions issued to them. The features in EDGAP3D have been designed for use with the "EXEC" operating system (Version 3.8) developed by the Computer Based Education (CB-E) research group at the University of Texas at Austin. It is assumed that this operating system has been loaded into the IMLAC's memory prior to executing EDGAP3D.

Rotation Feature

```
ROTATE,[xr],[yr],[zr]

          xr,yr,zr = the rotation in degrees about the x,y,
                     or z axes [the default values are xr=
                     45°, yr=-45°, zr=0]
```

The use of this command causes the data in the display memory to be operated on to perform the specified coordinate rotations. The sign convention used in specifying the degree of rotation about an axis is for positive angle to represent a clockwise rotation about the axis where viewing the origin from a point on the plus (+) side of the axis (reference the figure below). The rotations about the axes may be specified individually or in any combination. The default values are used only when no rotation angles are specified. Note that when multiple angles are specified, the algorithm assumes a sequential transformation of the order of xr, then yr, and then zr, or any combination of these as long as it is left to right (e.g., xr then zr, or yr then zr, etc.). After a rotation has been performed, the display data is automatically rescaled to insure that the region of interest will fall within the plotting limits.



This command allows the user to obtain arbitrary views of the mesh. Initially the mesh coordinate axes are oriented so that it's x and y-axes correspond to the screen x and y-axes, with the mest z-coordinate axis being out of the plane of the screen.

Scaling Features

SCALE,[sfac],[xt],[yt],[zt]

sfac = the scale factor (must be greater than zero). Default value is one (1.0).

xt,yt,zt = x,y, or z translation along the axis specified in either the plus or minus direction. Default values are zero (0.0).

The SCALE routine allows the user to enlarge or reduce the region to be displayed with respect to the screen. This option provides the user with a method for obtaining a higher resolution of various areas of the display data. When the SCALE feature is exercised, the display data is modified without changing the plotting limits. Since only information totally within the plotting limits is displayed, it may become necessary to translate the scaled data to shift particular regions where higher resolution is desired back into the viewing limits in order to display them. For example, consider the steps required to increase the resolution of the detailed region in Figure 16a. Then scaling by a factor of two and translating along the x and y-axes one unit results in Figure 16b, where only the detailed region lies totally within the plotting limits and is therefore the only portion of the figure displayed. Note that an element must lie completely within the plotting limits to be drawn.

Display Features

DISPLAY,[icode],[idev],[elnum]

```
        icode = 1:  region defined by PLOTS (default value)
                2:  single element plots
                3:  element highlighted in the defined region

        idev  = 1:  IMLAC terminal (CRT graphics) (default
                    value
                2:  Calcomp ballpoint pen plots

        elnum = the element number to be plotted when icode
                is either 2 or 3.
```

The DISPLAY routine is the routine which does the actual issuing of drawing instructions to the specified output device. This output device can be any type of machine that will accept standard Calcomp plotting calls. When a call is made to DISPLAY, the system checks the display data element by element, checking to verify that the element lies

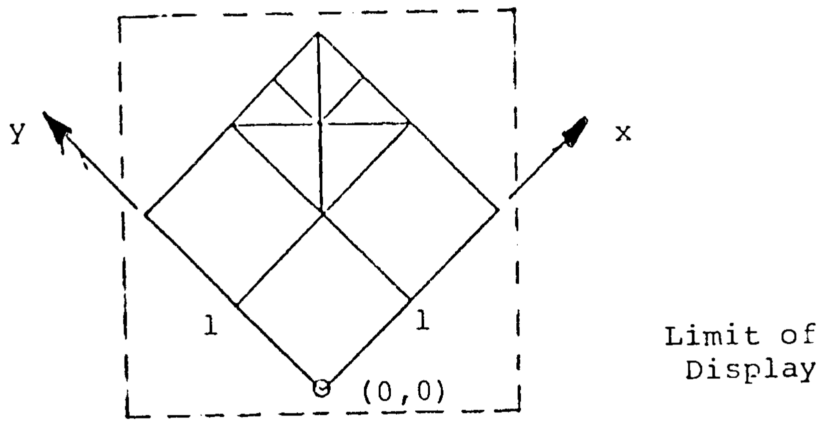


Figure 16a. Element Mesh in Region of Interest

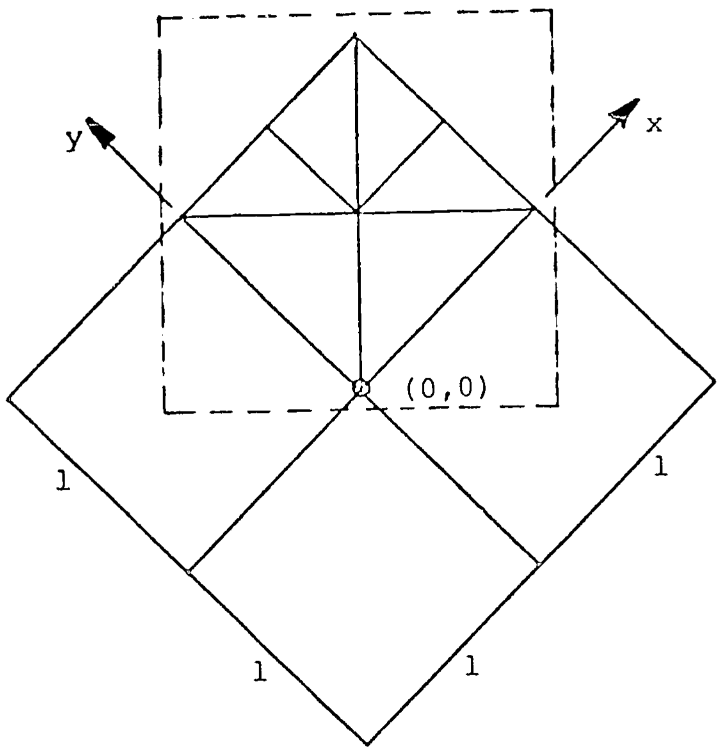


Figure 16b. Element Mesh Scaled by a Factor of Two and Translated One Unit All x and y-Axes

within the plotting (or screen) limits. Plotting instructions are issued element by element when the element lies totally within the plotting limits. The element number is written in the center of the element. Also included with each type of plot is a right-handed coordinate reference frame showing the present orientation of the mesh xyz-coordinate axes to the user and a data block showing the values of the current status of the orientation from its initial orientation. The mesh is initially oriented with its x and y coordinate axes corresponding to the screens x and y axes and the z axis of the mesh being out of the plane of the screen.

Terminator Commands

END,section

section = PLOTS,EDIT

This command is required to terminate both the graphics and the editing sections of the program. In the case of terminating the editing section, this command causes the regeneration of the element data tape used by TXGAP3D.

STOP

This command is used to terminate the program.

APPENDIX B

Example Session

The following is an example session illustrating the use of the EDGAP3D Preprocessor Program. An attempt has been made to exercise all the features available in the code. It is assumed that the data files generated by TXGAP3D are local to the user's files. The particular data files are TAPE12 (element data), TAPE13 (total number of elements generated), and TAPE15 (grid point data). It is assumed that the files are to be located by the file names indicated above. The commands for displaying information indicate the Imlac graphics terminal as the output device, while the illustrations are actually Calcomp plots.

```
cc:
* RFL,45000./
* EDGAP3D
GO:
* PLOTS,-1,-1,-1,6,7,7       (region of interest is to be
                             from (-1,-1,-1) to (6,7,7)
* ROTATE,40,30               (rotate the region of interest
                             40° about the x and then 30°
                             about the y-axes.)
* SCALE,.65                  (scale the display information
                             by a factor of .65)
* DISPLAY                    (display the all the elements
                             in the region of interest)
                             see Figure 17
* END,PLOTS                  (end the Graphics Module)
* EDIT                       (enter the Editing mode)
* GRID,2,1,2                 (output the location of grid
                             point (2,1,2)
```

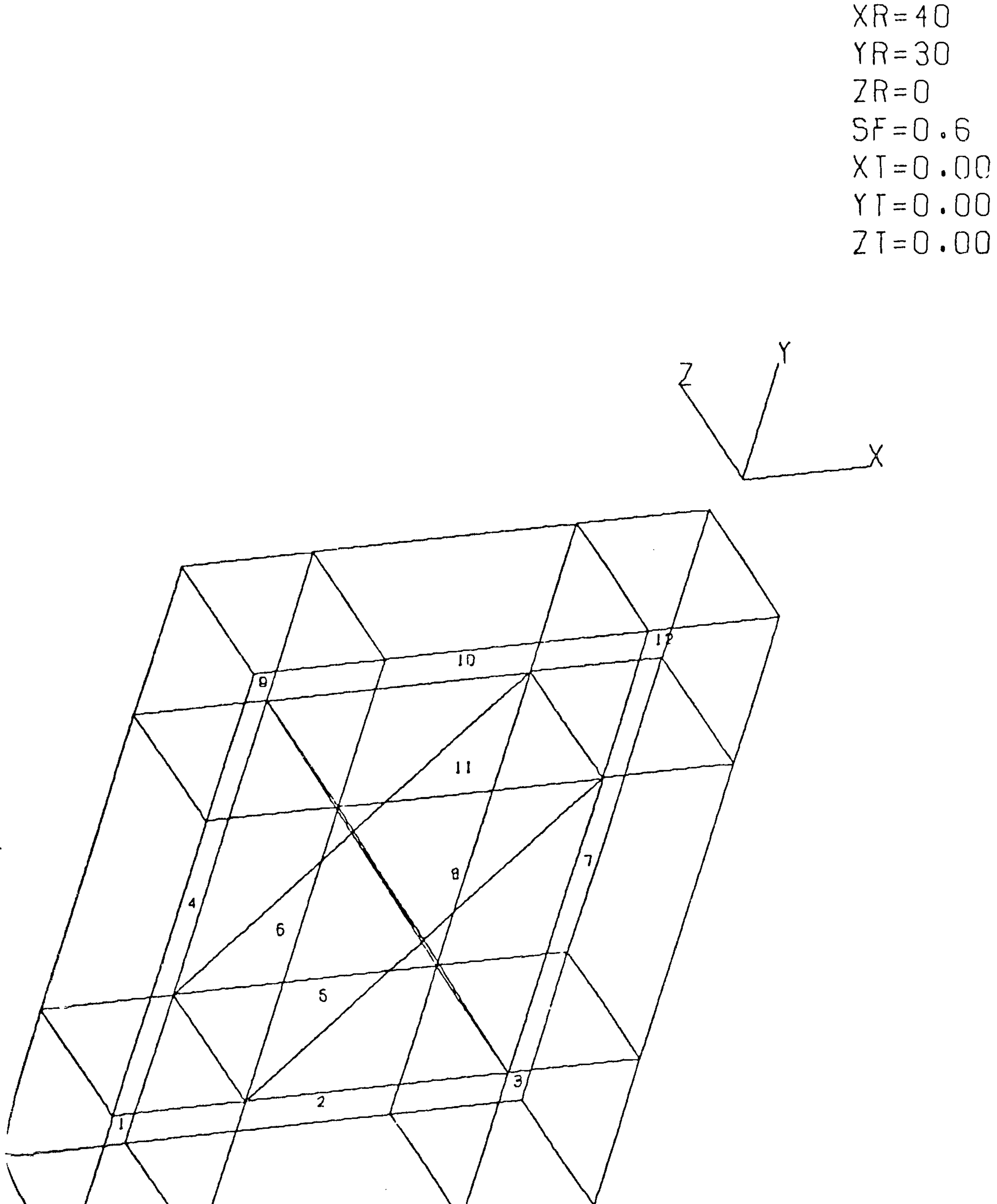


Figure 17. Element Mesh

```
                                             x,y,z_COORD.
___INTERNAL_I,J,K                    1.200E+00      0.        1.333E+00
______3_1_3                          (locate grid point (_4,1,2)
* GRID,-4,1,2                                x,y,z_COORD.
___INTERNAL_I,J,K                    4.200E+00      0.        1.333E+00
______8_1_3                                  (move grid point (2,
* GRID,2,1,2,MOVE,1,-.5,.75                  1,2) to (1.0,-0.5,
                                             0.75))
                                             x,y,z COORD.
___INTERNAL I,J,K                    1.200E+00      0.0       1.333E+00
______3_1_3                          (find connectivity for element
* CONNECT,ELEMENT,2,1,2              2,1,2)
                                     2,_1,_2
__CONNECTIVITY FOR ELEMENT
___NODE            I              J          K
   1               2              1          2
   2               4              1          2
   3               4              2          2
   4               2              2          2
   5               2              1          3
   6               4              1          3
   7               4              2          3
   8               2              2          3
* CONNECT,ELEMENT,6                  (find the connectivity for the
                                     sixth element on the data file)
___CONNECTIVITY FOR ELEMENT NO. 6
          NODE            I            J          K
           1              2            2          2
           2              3            3          2
           3              2            4          2
           4              2            2          3
           5              3            3          3
           6              2            4          3
* CONNECT,NODE,4,1,2                 (find the elements connected
                                     to grid point 4,1,2)
___CONNECTIVITY FOR NODE 3,_3,_3
          LOCATED AT COORD. 2.400E+00, 3.000E+00, 2.667E+00
          ELEMENT              ELEMENT           LOCAL
          NUMBER            I     J      K       NODE
            5               2     2      2        6
            6               2     2      2        5
            8               3     3      2        4
           11               4     4      2        6
* CONNECT,MODIFY,4,1,2               (changed the connectivity of
                                     element 4,1,2)
__CONNECTIVITY FOR ELEMENT                     4,_1,_2
```

```
         NODE                     I          J          K
          1                       4          1          2
          2                       5          1          2
          3                       5          2          2
          4                       4          2          2
          5                       4          1          3
          6                       5          1          3
          7                       5          2          3
          8                       4          2          3
* 6,12,1,5                        (change connectivity of node
                                  6 to 12,1,5
____WARNING:                      COMMAND          IGNORED
-----GRID POINT 12,1,5 IS NONEXISTANT
* 6,6,1,3                         (change connectivity of node
                                  2 to 6,1,3)
* 2,6,1,1                         (change connectivity of node
                                  2 to 6,1,1)
* FINISH                          (end connectivity modifications)
* END,EDIT                        (end editing module)
* PLOTS,-1,-1,-1,7,7,7
* ROTATE,70
* ROTATE,,,15
* SCALE,.7
* DISPLAY,1,1                     (display the elements from the
                                  modified element data, see
                                  Figure 18)

* END,PLOTS
* EDIT
* BC,UX                           (find all the x-DISPLACEMENT
                                  BOUNDARY CONDITION)
___x-DISPLACEMENT BOUNDARY CONDITION
----ELEMENT           NODE          VALUE
    1, 4, 2            3             0.
    1, 4, 2            4             0.
    1, 4, 2            7             0.
    1, 4, 2            8             0.
    2, 4, 2            3             0.
    2, 4, 2            4             0.
    2, 4, 2            7             0.
    2, 4, 2            8             0.
    5, 4, 2            1             0.
    5, 4, 2            2             0.
    5, 4, 2            5             0.
    5, 4, 2            6             0.
* BC,ELEMENT,1,4,2                (find all the boundary conditions
                                  on element(s) 1,4,2)
```

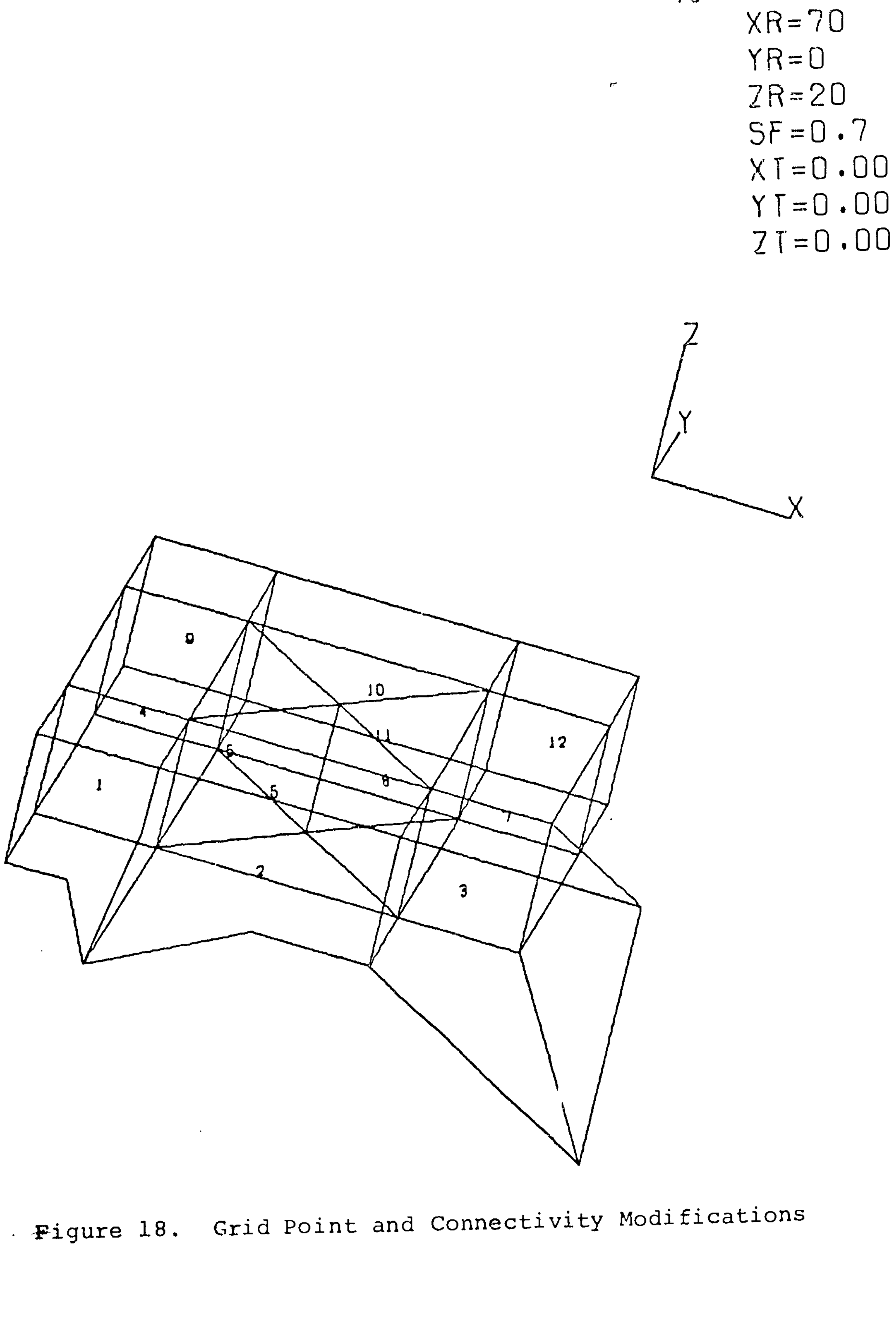


Figure 18. Grid Point and Connectivity Modifications

```
---_BC or ELEMENT(s)                                 1,_4, 2
--BC TYPE                NODE/FACE              VALUE
--  UX                      2                    0.
    UX                      3                    0.
    UX                      6                    0.
    UX                      7                    0.
    UY                      2                    0.
    UY                      3                    0.
    UY                      6                    0.
    UY                      7                    0.
    UZ                      2                    0.
    UZ                      3                    0.
    UZ                      6                    0.
    UZ                      7                    0.
  PRESSURE                  4                -7.500E+00
  PRESSURE                  4                -7.500E+00
  PRESSURE                  4                -7.500E+00
  PRESSURE                  4                -7.500E+00
* BC,ADD,4,1,2                           (addition of boundary condition
                                         to element 4,1,2)
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
--_ENTER BC INFO. TO BE ADDED OR DELETED
*-PRESSURE,3,100.                        (add pressure BC on face 3 of
                                         element 4,1,2)
* BC,DELETE,4,1,2                        (deletion of boundary condition
                                         from element 4,1,2)
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
  PRESSURE                  5                 5.500E+00
  PRESSURE                  3                 1.000E+02
  PRESSURE                  3                 1.000E+02
  PRESSURE                  3                 1.000E+02
  PRESSURE                  3                 1.000E+02
--_ENTER BC INFO. TO BE ADDED OR DELETED
*-PRESSURE,5                             (delete pressure BC from face
                                         5 of element 4,1,2)
* BRICK,1,1,4,3,2,4,3,2,5,3,1,5,3,1,4,4,2,4,4,2,5,4,1,5,4
                                         (add brick element 1,4,3 to
                                         element data)
* PRISM,1,4,2,3,5,2,3,5,2,4,4,4,3,3,5,4,3,5,4,4
                                         (add prismatic element 4,2,3
                                         to element data)
* DELETE,2,2,2                           (delete element(s) 2,2,2 from
                                         element data)
```

```
---DELETED ELEMENT      2      2     2
---DELETED ELEMENT      2      2     2
* DELETE,3,3,2                (indicates multiple elements
                              by same name (all deleted))
---DELETED ELEMENT      3      3     2
* DELETE,11                   (delete element no. 11 in the
                              element data list)
---DELETED ELEMENT     11
* END,EDIT
* PLOTS,_1,_1,_1,7,7,7
* ROTATE,70
* ROTATE,,,15
* SCALE,.7
* DISPLAY                     (display the newly modified
                              element data in the specified
                              region.  see Figure 19)

* DISPLAY,3,,4                (display element number four
                              in the highlighted region, see
                              Figure 20)
* SCALE,1.2                   (scale the display region by
                              a factor of 1.2)

* SCALE,,,2,1                 (shift the origin two inches
                              along the y-axis and one inch
                              along the z axis)
* DISPLAY                     (show the results, see Figure 21)
* END,PLOTS
* STOP                        terminate execution of the pro-
                              gram
```

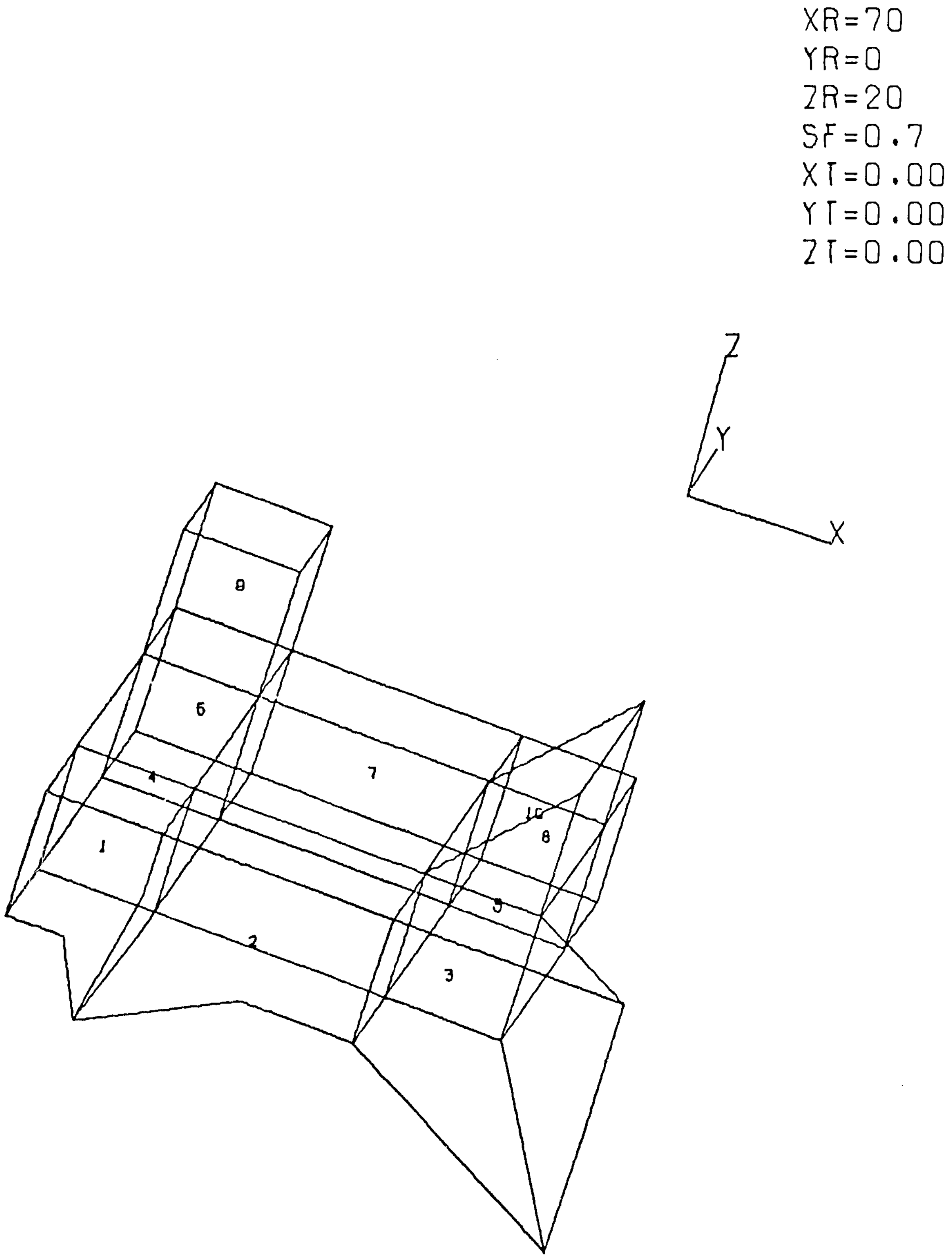


Figure 19. Element Addition and Deletion of Previously Modified Mesh

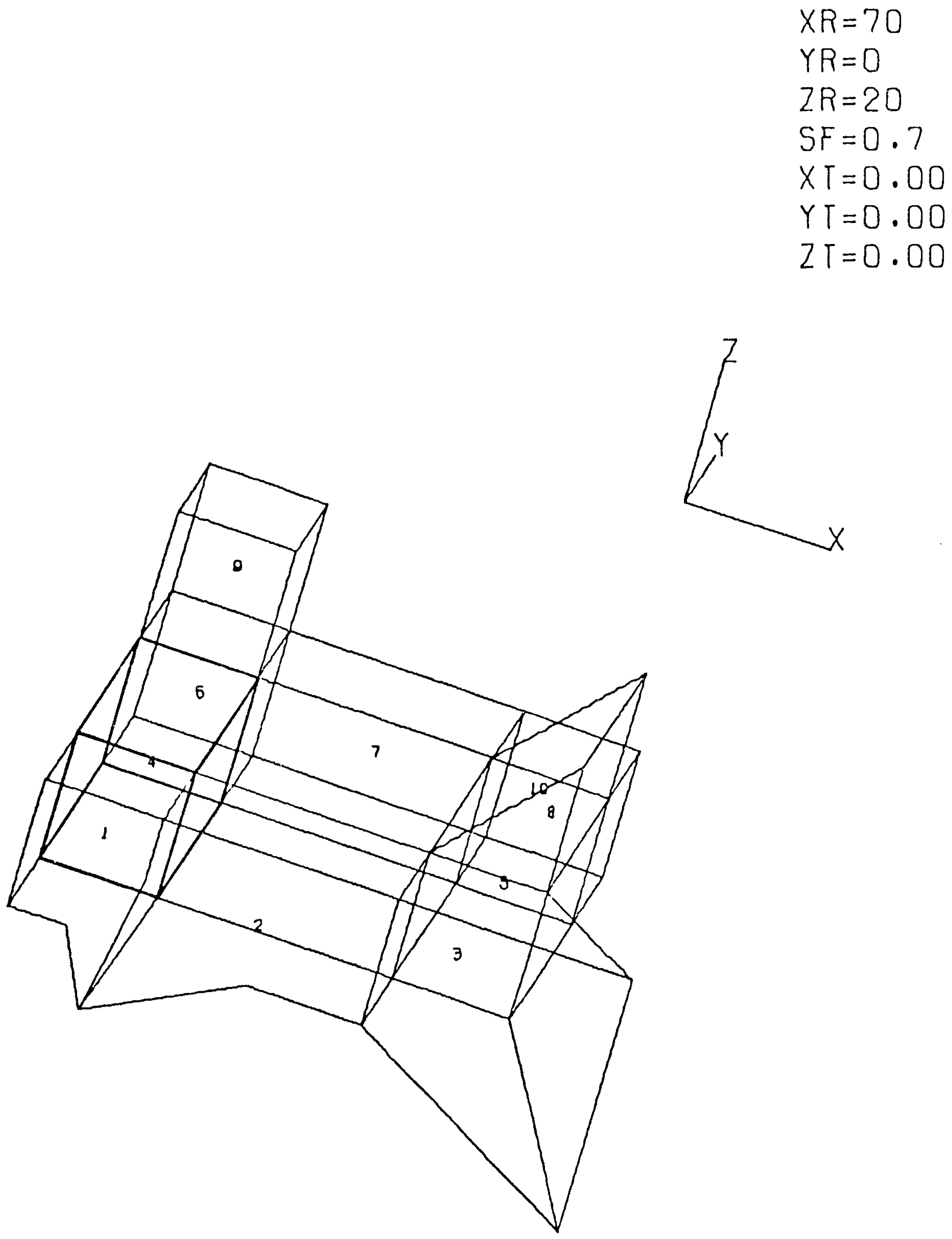


Figure 20. Element Highlighted in Modified Region

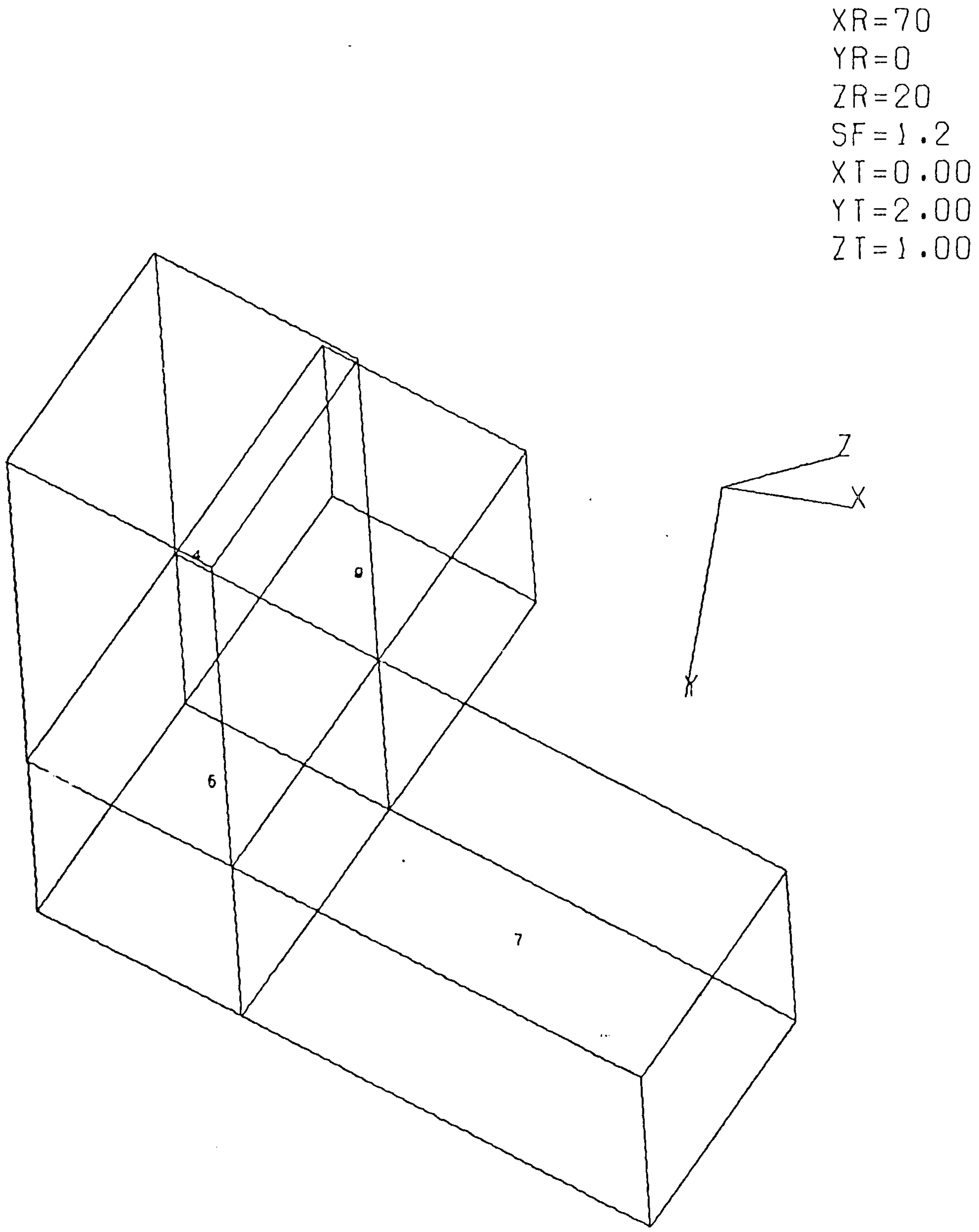


Figure 21. Element Mesh Display Resulting from Scaling and Translating Modified Region

BIBLIOGRAPHY

1. Herness, E. D. and J. L. Tocher, "Design of Pre- and Postprocessors," Structural Mechanics Computer Programs, Ed. W. Pilkey, K. Saczalski, H. Schaeffer; University Press of Virginia, Charlottesville, Virginia (1974), pp. 887-898.

2. Napolitano, L. G., R. Monti, and P. Murino, "Preprocessors for General Purpose Finite Element Programs," Structural Mechanics Computer Programs, Ed. W. Pilkey, K. Saczalski, H. Schaeffer; University Press of Virginia, Charlottesville, Virginia (1974), pp. 807-823.

3. Nicolau del Roure, R. G., E. B. Becker, and R. S. Dunham, "The Texas Three Dimensional Grain Analysis Program," University of Texas TICOM Report 74-2, February, 1974.

4. Dunham, R. S., E. B. Becker, and F. M. Guerra, "Organization and Functional Purpose of Finite Element Computer Programs," ASME Conference on Pressure Vessels and Piping, Miami, June, 1974.

5. Newman, W. M. and R. F. Sproull, Principles of Interactive Computer Graphics. McGraw-Hill Book Company: New York, 1973.

6. Sciarra, J. J., "Vibration Analysis in 3D with Computer Graphics," Sound and Vibration. January, 1970, pp. 10-21.

7. User Reference Manual; Inlac PSD-1 Programmable Display System. Imlac Corporation, 1971; Revision C.

COMPUTER-BASED EDUCATION COURSES

AEROSPACE ENGINEERING

Aircraft Design—Drs. W. T. Fowler and D. G. Hull
Structural Analysis—Dr. Eric Becker

ARCHITECTURE

Survey of Environmental Control Systems—Dr. F. N. Arumi

CHEMICAL ENGINEERING

Process Analysis and Simulation—Dr. D. M. Himmelblau
Optimal Control—Drs. T. F. Edgar, E. H. Wissler and J. O. Hougen

CHEMISTRY

Vector Space Theory of Matter—Dr. F. A. Matsen
Physical Chemistry Laboratory—Dr. John M. White
Organic Chemistry—Drs. J. C. Gilbert and G. H. Culp
Introductory Chemistry—Dr. J. J. Lagowski
Principles of Chemistry—Dr. J. J. Lagowski
Introduction to Chemical Practice—Dr. J. J. Lagowski

CIVIL ENGINEERING

Computer Methods for Civil Engineering Laboratory—Dr. C. Philip Johnson et. al.

ECONOMICS

Theory of Income and Employment—Dr. James L. Weatherby

ENGLISH

English Composition—Dr. Susan Wittig

HOME ECONOMICS

Child Development—Dr. Mary Ellen Durrett

LINGUISTICS

Language and Society—Dr. W. P. Lehmann

MATHEMATICS

Calculus I, II—Dr. John P. Alexander

MECHANICAL ENGINEERING

Dynamic Systems-Synthesis—Dr. L. L. Hoberock
Probability and Statistics for Engineers—Dr. G. R. Wagner
Energy Systems Laboratory—Dr. G. C. Vliet
Element Design—Dr. John J. Allan III
Nuclear Reactor Engineering—Dr. B. V. Koen
Kinematics and Dynamic Mechanical Systems—Dr. W. S. Reed

PSYCHOLOGY

Introduction to Psychology—Self Paced—Dr. Jan H. Bruell
Statistical Methods in Psychology—Dr. James M. Swanson

PHYSICS

Computer Introduction to Physics—Dr. J. D. Gavenda

ZOOLOGY

Genetics—Dr. Richard Richardson
Experimental Genetics—Dr. Richard Richardson
Biophysical Analysis—Dr. J. L. Fox