MESH GENERATION FOR TWO-DIMENSIONAL REGIONS USING THE TEKTRONIX DVST (DIRECT VIEW STORAGE TUBE) GRAPHICS TERMINAL

Verlan K. Gabrielson

Sandia Laboratories, Livermore

ABSTRACT

This paper describes the code DVMESH and the use of the Tektronix DVST graphics terminal for applications of preparing mesh data for use in various two-dimensional axisymmetric finite element stress analysis and heat transfer codes.

INTRODUCTION

The use of direct view storage tube (DVST) terminals, such as the Tektronix 4002A and 4014-1, for various graphics and interactive applications has expanded greatly during the past few years. This is due to their relatively low cost and their adaptability to the terminal networks used on most computer systems.

At Sandia Laboratories, Livermore (SLL), the DVST terminals are being distributed throughout the laboratory. These terminals are interfaced to a Control Data Corporation 6600 computer and are accessed via CDC's Intercom software system. The accessibility, ease of use, and adequate display area of the terminals make this graphic system effective for interactive applications which do not require large data bases or high rates of data transfer. (This does not imply that such applications cannot be implemented, but they increase the time between interactive responses. The time the user waits for a response is a primary consideration in any interactive application and becomes more sensitive in a system like SLL's since the data and programs are processed and stored on a host computer which is servicing a large number of users.)

To support this expansion of graphics facilities, a number of programs have been coded to evaluate applications for which interactive graphics can be used most effectively. The DVMESH code uses the DVST terminal to prepare mesh data for various finite element stress analysis and heat transfer codes. The project was designed to evaluate techniques, attributes, and limitations of the DVST terminal for this type of application.

The need is quite real; designing finite element models is normally one of the more time-consuming tasks of the structural engineer. The task of designing the proper mesh for an analysis is quite dependent on having an adequate pictorial display of the mesh. Thus design time depends on the speed with which graphic displays can be created, changed, and verified. For this reason, a mesh code designed around the capabilities of the terminal for problems requiring a lot of graphics verification should become a very useful tool for the structural engineer.

The design parameters and goals of the DVMESH code have been to:

(a) Make it applicable to two-dimensional regions

(b) Allow its input data to come from various sources

(c) Implement available interactive features of the DVST, with special emphasis on data sets entered at the terminal keyboard

(d) Design a standard output data set which will allow the mesh to be designed in modules

(e) Design for approximately 25,000 words of memory

(f) Provide necessary intermediate files in order to recover after abnormal or normal interruptions

The following sections describe the Tektronix terminals, details of DVMESH and its implementation on a sample problem, and some general comments concerning the DVST terminal for this type of application.

THE TEKTRONIX TERMINAL

The DVST terminals use the CDC Intercom software system which resides on the CDC 6600. Applications programs are written in FORTRAN, using system subroutines for displaying text, constructing line vectors, and initiating the cross-hair cursor. The display screen is a write-only display which can be refreshed only by erasing the entire screen.

The 4002A terminal being used has a display screen of 1024 (X) by 780 (Y) viewable points and space for thirty-nine lines of alphanumeric text. The terminal has a standard keyboard and an interactively driven cross-hair cursor. These features can be seen in Figure 1.

Features of the system which are attributes in graphics applications are:

(a) An easy-to-use keyboard in which all entries are displayed in a scratch area on the screen for verification of the line of data as it is being keyed.

(b) Hard copy prints of any display are easily obtainable.

(c) Using the Intercom system as the interface between the terminal and the computer allows the user to communicate directly with the SCOPE operating system on the 6600. This provides access to the mass storage devices on the 6600 and permits the use of the UPDATE and file processing programs, plus the Intercom text editor program.

Features which are limitations in graphics applications are:

(a) A minimum of interactive hardware is provided. All displays have to be designed by the program residing on the host computer. Such features as windowing, rotation, etc., are done by the user's program.

(b) The system cannot permit a selective erase. Thus more effort is required in programming the displays for prompting messages, data verification, and modifications to subregions of the display.

(c) In effective interactive processing the wait time between operations should be minimized. This time depends on the rate of data transfer and the size of the data buffer. With the Intercom system, it also becomes a function of the number of users on the network and the workload on the 6600.

The DVMESH code was designed and implemented on the above system, but recent deliveries of 4014-1 displays with their 4906 (X) by 3120 (Y) viewable points and up to sixty-four alphanumeric line capability will enhance the capabilities of the program. In addition, the PLOT-10 software package available from Tektronix will be implemented on the system.

THE DVMESH CODE

The DVMESH code applies to two-dimensional regions. These regions may have irregular boundaries, material interfaces, voids, etc. The ability to create an adequate mesh over the regions depends on the user's skill in using the functions available in DVMESH, and the restrictions imposed by the stress analysis or heat transfer code for which the data is being prepared. In all of these codes the meshing problem is similar to those described in the survey paper on mesh generation (reference 1).

A space is subdivided into quadrilateral elements consistent with the needs of the problem and the type of analysis. The mesh data required are the coordinates of the nodes defining the corners of the quadrilateral element. The DVMESH code is designed to produce mesh data for finite element codes which require that all the nodes be mapped onto a unit grid, such that each node can be identified by an integer pair (i, j) denoting specific rows and columns in the grid. Figures 2

and 3 illustrate this mapping. This procedure, which is common in many finite element codes and creates difficult problems in meshing irregular regions, defines the connectivity of the mesh and simplifies the problem of matrix storage and boundary definitions. For this type of application, in which data preparation and graphics verification are very time-consuming, the interactive capabilities of the DVMESH code have been quite useful.

DVMESH interactive capability is implemented in three phases: input boundary definition, mesh development, and output generation. The following sections give a brief description of each phase. The code uses only the hardware features of the keyboard and cross-hair cursors. "Functions" in the following descriptions refer to programmed routines in the DVMESH code which can be used by entering special keywords at the terminal.

INPUT BOUNDARY DEFINITIONS

The input data set consists of coordinates of points, line segments, and circular arcs which define the basic boundaries of the regions to be meshed. An illustration of a sample data set is shown with the sample problem. Four input options have been implemented, providing considerable flexibility to the user for data preparation. The options include:

(1) Data sets entered at the terminal

(2) Punched card records stored on disk file

(3) Data sets generated by digitizers

(4) Data sets extracted from large data bases generated by the APT (automatic programming tool) processor used for automatic drafting purposes

The input data records are identified by line numbers and can be displayed and edited by entering functions at the terminal with the line identifiers. These functions, which are used in an interactive mode, include inserting, changing, and deleting specified lines in the input data set and displaying selected data sets.

A graphics display of the boundary data set can be obtained at any time during input; this provides a means to verify, correct errors, and make adjustments to the data at the terminal. The user iterates on these operations until the boundary data set is properly defined. Functions used in the interactive mode include plotting the data within given min-max values and "windowing" by contracting or expanding the plot about a point which is identified with the cross-hair cursors.

MESH DEVELOPMENT

The meshing algorithm is quite simple. The mesh is designed as a rectangular array of quadrilateral elements having (m) rows and (n) columns. Each node of an element is uniquely identified by an integer pair (i, j) representing its row-column location in the grid, and the coordinates of the nodes are stored at equivalent locations in their respective arrays. The user defines the nodes at selected locations on the boundary data shown on the display by use of a special function and assigns to each its (i, j) location. The entire mesh can be defined by the user, but normally only a minimum set of nodes need be identified. All nodes not defined by the user are computed by linear interpolation along each row or column. This is illustrated in the sample problem.

INTERACTIVE PROCEDURE

When the meshing code is initiated, the boundary data set which exists within the current min-max values is displayed and the maximum size of the mesh arrays for the given problem is entered at the terminal by the user. The user proceeds by defining nodes on the perimeter of the boundary data. When a sufficient number have been defined, a mesh is constructed within the defined nodes. By observing the mesh the user proceeds by selecting the necessary functions to accomplish various tasks. The result of each function is displayed and a mesh can be reconstructed at any step in the process. This interactive procedure is continued until a satisfactory mesh is developed over the given boundary data set.

THE PICK FUNCTION

The basic meshing function is PICK, which is used to define node locations by use of the cross-hair cursor. This function, entered at the keyboard by the user, displays the cross-hair which can be interactively moved to any location on the display. The cursor is turned off by a keyboard interaction. When the cursor is turned off, the user records the node's location in the mesh by entering at the terminal its (i, j) coordinates. Following this entry, an X is displayed at the point selected by the program for this node and the cross-hair is reinitiated for further processing.

Interaction involving the cross-hair cursor requires the use of the system subroutine LOC. When the cursor is turned off, the function returns the coordinates of the cross-hair intersections and the keyboard entry used in the interrupt. This provides a valuable programming tool, since each key can represent a functional response. In this application

the keys N, P, and L provide alternatives in the location of the node at the cross-hair intersection. The N option locates the point at the cursor and the X option transfers control out of this procedure. The P and L options provide a means to locate the node on the boundary data set. P implies that the node is located at the point in the boundary data set nearest the cross-hair, and L implies locating the node on the nearest line segment.

SPECIAL FUNCTIONS

After an initial mesh has been constructed from the nodes defined by the PICK function, further development of the mesh can be done by use of the following functions. These functions are illustrated in the sample problem.

CIRP — is a function for locating nodes on a circular arc by identifying three nodes on the arc which have been previously defined.

BFIT — is a function similar to PICK which moves each node along a given row or column to the nearest point in the boundary data set.

VOID — is a function which permits regions in the mesh to be defined as voids on subsequent displays.

INIT — is a function providing a means to reinitialize the entire grid or parts of it when errors have to be corrected or adjustments made.

SHFT — is a function which allows nodes of a given row or column to be located as normal translations from a defined row or column. This is very useful in obtaining accurate definitions in meshing structures composed of layered materials, interface gaps, etc.

Nodes not defined by the above functions can be generated by one of the grid functions:

GRDI — displays the mesh by interpolating between defined nodes on each row of the grid, then interpolating between defined nodes on each column.

GRDJ — similar to GRDI, except nodes are defined on each column, then on each row.

GRIJ — used for defining nodes within a section of the grid having convex or concave boundaries which may not mesh properly using GRDI or GRDJ. It requires that all nodes on the boundary of the section be specified prior to use.

ERAS — In practice, several iterations will be required to obtain the desired mesh. This may require many erasures of the display. The ERAS function provides this capability. The display is erased and the boundary data set plus the status of the defined nodes are reset to the definitions prior to the erase. The defined nodes include all that have been defined by the above functions except those computed by interpolation in the GRDI and GRDJ functions.

OUTPUT PREPARATION

The output data file prepared by the DVMESH code consists of problem identifiers and the coordinates of the nodes identified by a specific row and column. This array, stored on disk file, is the standard output file which can be implemented by the various analytical codes. It also provides a restart capability for DVMESH.

Interactive features at this level permit the mesh to be viewed in detail and allow location of a given node by (x, y) and (i, j) coordinates; this may be useful for boundary and material identifications in the finite element analysis.

With a memory allocation of 25,000 words, meshes of over 2,000 nodes can be developed by DVMESH. For the case of large problems, the code output file is designed so that the mesh can be developed in sections.

SAMPLE PROBLEMS

The following sequence of figures illustrates how a mesh is developed using DVMESH at an interactive terminal. The procedure illustrates only one of many possible procedures to obtain a mesh.

Input data options:

```
 TYPE INPUT FILE AND PROBLEM HEADER
60--TYPED INPUT
40--PUNCHED CARD INPUT
20--DIGITIZED DATA FILE
 42-- APT DATA FILE
13--RESTART DATA FILE
```

All displays shown in this section were prepared from hard copy prints of displays on the terminal. For example, input data entries from terminal proceed as follows:

```
60      SAMPLE PROBLEM
 TYPE DATA CARD
DATA 1 1 1
 TYPE DATA CARD
SIDE  1 13  0.  0.  2.  4.2   .2
 TYPE DATA CARD
SIDE  2 1 3  1.  .5  2.5  .5   2.5  .7
 TYPE DATA CARD
SIDE   2 1 2  1.0  .7   1.0   .5
 TYPE DATA CARD
ARC  4 1 2  0.  4.  2.6  270.   340.
 TYPE DATA CARD
ARCP  6 1 2  1.9  1.6   2.4   2.15  2.8  3.1
 TYPE DATA CARD
ARCP   7 1 2  1.9  1.6  1.8   1.4  2.2  1.1
 TYPE DATA CARD
ARCP   7 1 1   2.2  1.1  3.0   1.15  4.2  1.3
 TYPE DATA CARD
```

Procedure is completed when END is typed.

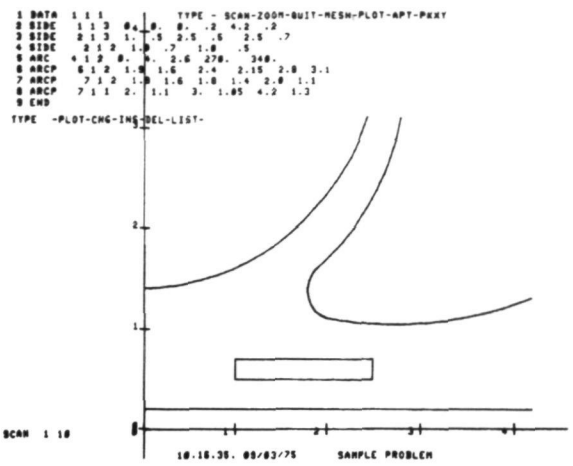
Editor Phase options are PLOT-CHG-INS-DEL-LIST.
Display of data entered at terminal using LIST function:

```
1 DATA  1 1 1
2 SIDE   1 13  0.  0.  2.  4.2   .2
3 SIDE   2 1 3  1.  .5  2.5  .5   2.5  .7
4 SIDE    2 1 2  1.0  .7   1.0   .5
5 ARC   4 1 2  0.  4.  2.6  270.   340.
6 ARCP   6 1 2  1.9  1.6   2.4   2.15  2.8  3.1
7 ARCP    7 1 2  1.9  1.6  1.8   1.4  2.2  1.1
8 ARCP    7 1 1   2.2  1.1  3.0   1.15  4.2  1.3
9 END
TYPE  -PLOT-CHG-INS-DEL-LIST-
```

Example of changing data entry 2:

```
CHG  2
          TYPE DATA  -- FOR INS  USE LAST TO END
SIDE  1 1 3  0.  0.  0.  .2  4.2  .2
 TYPE  -PLOT-CHG-INS-DEL-LIST-
```

The entry PLOT displays the data graphically:



Plotting phase functions available at this phase of problem:

TYPE - SCAN-ZOOM-QUIT-MESH-PLOT-APT-PKXY

Transfer is made to the meshing program when the MESH function is entered.

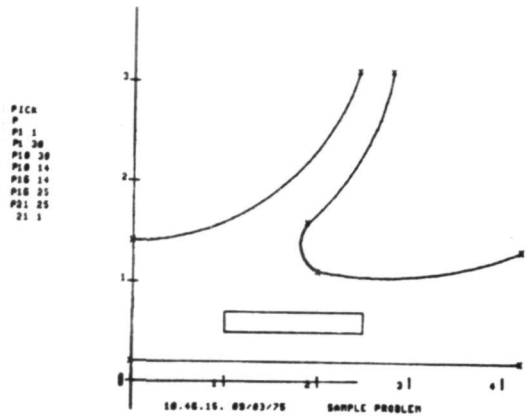
Before meshing proceeds, the user must have some insight into the size of grid desired and relative locations of the maximum and minimum (i, j) node identifiers. This can be done by drawing a rough sketch of the desired mesh as a guide to the interactive development.

The maximum row and column is entered when the MESH function is typed. All references to node identifiers in the mesh code must be within these bounds.

The following illustrates the available functions for mesh development:

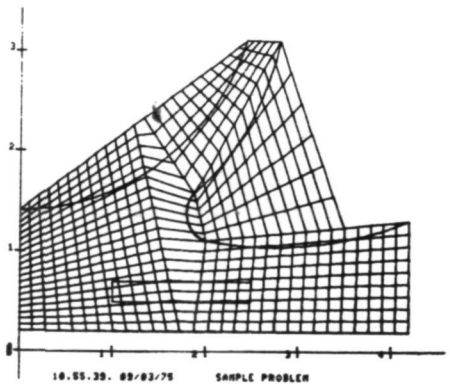
This example illustrates the use of some of these functions for developing a mesh over the above boundary data set. The first function used is PICK in which the cross-hair cursor is moved to the lower-left corner of the boundary data. When P is typed the node is located at the end point of the line segment and identified by entering the (i, j) values of (1, 1).

The interaction continues by use of the cross-hair and the PICK function until a set of nodes is defined on the perimeter of the boundary data set. For this case the PICK entries are defined in a counterclockwise direction. The following displays these nodes:



The nodes defined above are a sufficient set for constructing a mesh over the region.

The following display results when GRDI is entered.



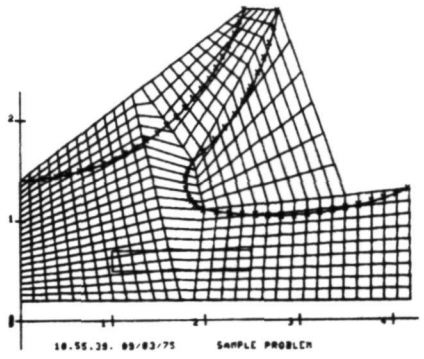
At this point the functions BFIT and CIRP are used to locate given rows and columns on the boundary data set. In this example row 21 is located on the circular arc passing through nodes (21, 1), (21, 10), and (21, 25). The node (21, 10) is located on the circular arc by the BFIT or PICK function. From this data the nodes (21, 2), ..., (21, 24) are defined on the circular arc in equal angular segments by the function entries:

```
BFIT   21 10
CIRP  21 1  21 10  21  25  1
```

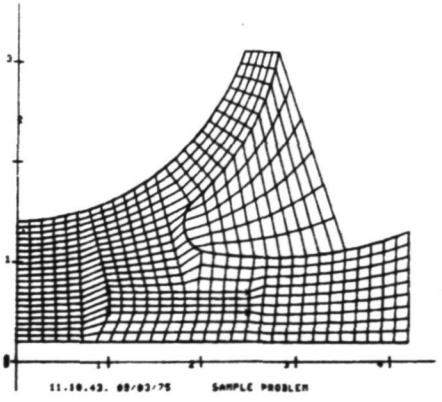
In like manner node points between (16, 14) - (16, 25), (10, 14) - (10, 30), and (10, 14) - (16, 14) are located on their respective circular arcs:

```
BFIT    16  20
CIRP  16 14  16 20 16 25  1
BFIT   10 20
CIRP  10 14  10 20  10 30  1
BFIT   12 14
CIRP  10 14  12 14  16 14  -1
```

The X's on the boundary sets in the figure illustrate how the functions are verified on the display.



To redisplay the mesh, the ERAS function is used followed by GRDI, which interpolates new values for all nodes not defined by the special functions. The following display results:

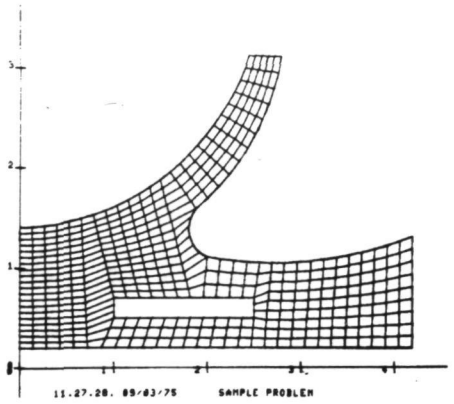


This display also shows the result of defining four nodes about the rectangular region using the PICK function.

To remove nodes which lie in regions defined as voids the following entries are made:

```
VOID  11 15  15 25
VOID   6  8   7 17
```

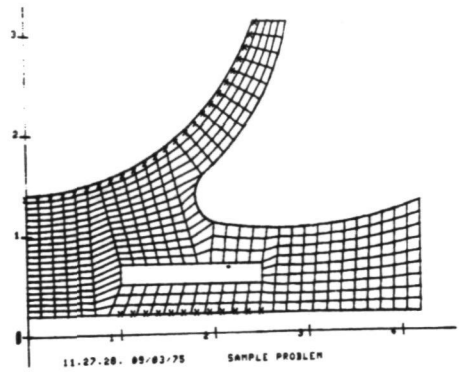
These nodes are verified on the display, then ERAS and GRDI functions are entered, resulting in the following mesh·



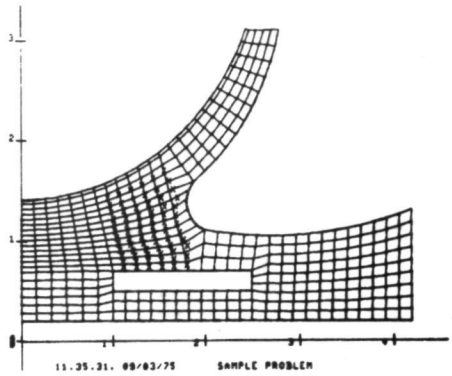
To illustrate the use of the SHFT function the figure shows where nodes are located when the following entries are made:

```
SHFT  21 1  21 25  .84  -1  0
SHFT  4 7   4 18   0.   -.3  -3  0
```

The first relocates row 20 a .04 normal distance from row 21. The second entry relocates selected columns of row 1 a fixed distance from row 4.

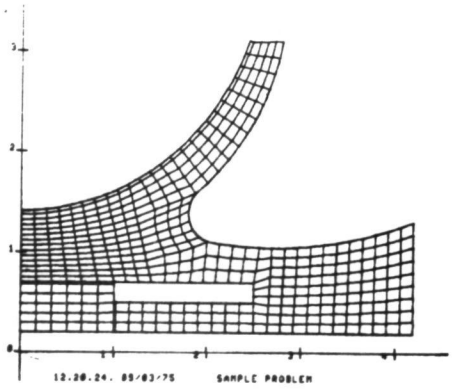


To reset the nodes in the convex region of column 14 the GDIJ function is used in the region bounded by the nodes (7, 7) and (20, 14) as shown in the display:



The final mesh in the following display shows the result of GDIJ function plus the use of the SHFT function for defining the interfaces along sections of row 7 and column 7. The SHFT function entries were:

```
SHFT  1 1  1 7  10  148  5  0
SHFT  1 7  4 7  102  .0  0  1
```



The data sets are now processed onto an output file which can be used in various analytical codes. As can be observed, the procedure to obtain a mesh is not fixed but is dependent on the user, his needs, and his experience. The goal is to initiate a crude mesh over the region and then refine the mesh by visual observation and the use of the various special functions.

GENERAL OBSERVATIONS

The development of DVMESH provided considerable experience for interactive meshing applications. Using a system with a minimum of interactive features and relatively slow response time required numerous tradeoffs between the frequency of refreshing the display and the complexity of the task for each function.

Implementing DVMESH as an interactive process implies that the user will develop the grid visually, since it is quite inefficient to consult notes, procedures, etc. while at the terminal. This requires instructions and prompts to be displayed, as well as verification of functions implemented by the user. These procedures are normally used in interactive programs but are essential in this application, since the response of the system is quite variable. These additional displays have the effect of filling the screen quickly, since no selective erase (a feature of larger graphic systems) is available.

The design of interactive programs for a DVST device requires some special considerations associated with the erase function. Since an erase results in a total erasure of the display, special efforts must be implemented to reproduce meaningful data with minimum effect on time and prior efforts. As shown in the sample problem, a number of erasures were required to complete a problem. To minimize the effect of the erase, a file is created which contains node definitions generated by the functions PICK-CIRP-BFIT-VOID-INIT-SHFT, etc., prior to the erase. Processing this file after the erase resets the meshing procedure to a point prior to the last grid instruction from which processing can continue. This requires that after each erase the graphics display is rebuilt, which can be time-consuming. To reduce this time requires special programming efforts such as minimizing axis annotation, design of mesh, etc.

Another important item of any interactive program is recovery when erroneous data is entered at the terminal. In DVST applications the programs must be coded to check on data consistency, proper data values, number of entries, illegal characters, etc. The programming time spent on this feature is worth the effort, since transmitting erroneous data is easily done at the terminal. In DVMESH a very useful subroutine called MASK has been used for this purpose. It provides a way to test for data consistency and the proper number of data entries for each type of function. It allows all data entries to be entered in a free field format, which is an essential feature for DVST terminal inputs.

An aspect of the DVMESH code which has proven to be quite valuable is the ease of making modifications to existing meshes. Since most of the meshes apply to design problems, the need to make parameter studies by making minor adjustments to an existing mesh occurs quite often. With DVMESH, the existing data can be retrieved from the computer data files, adapted to the new design, and verified graphically in much less time than by previous procedures.

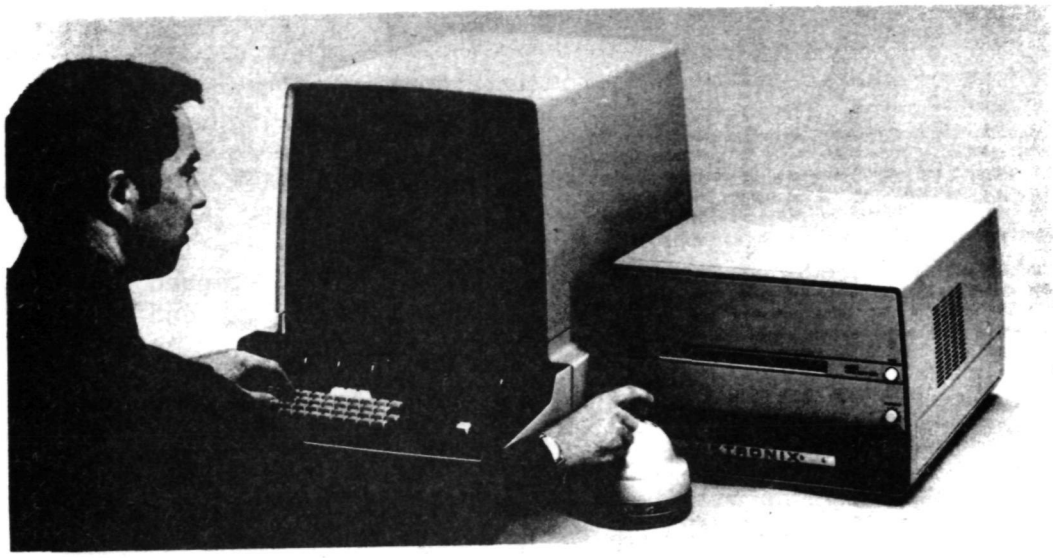
CONCLUSION

Most of the goals noted in the introduction have been achieved in the DVMESH code. The code is in operation and is being used as a practical alternative at this time for constructing meshes over two-dimensional regions having irregular boundaries. The code provides a way for a user to design a mesh from a low-cost terminal by a visual, interactive procedure. The time spent preparing the mesh may be similar to conventional batch procedures for a given problem, but the elapsed time from start to a verified mesh may be much less since the mesh can be verified graphically at each step of the process.

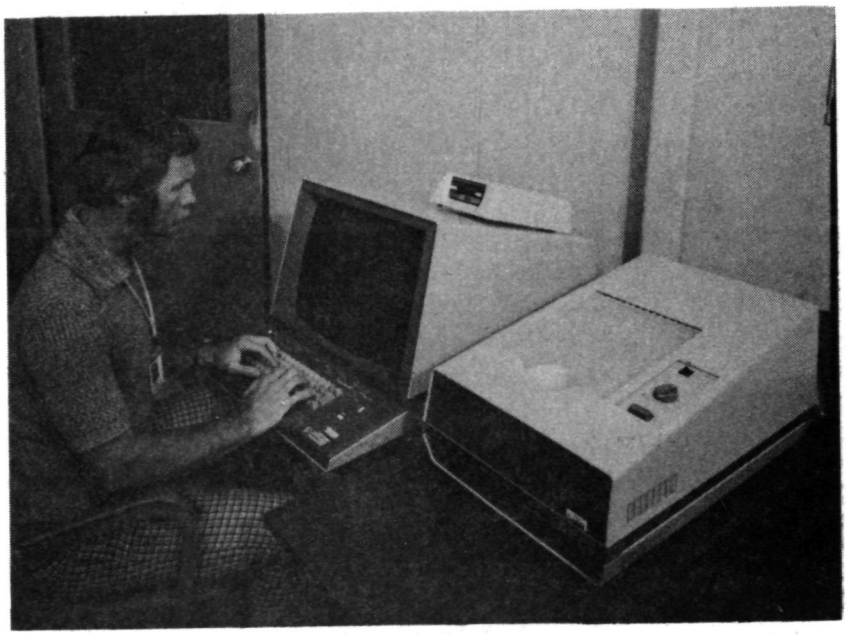
The code is written in FORTRAN but requires several system routines dependent on the graphics hardware used. It is available from the author on request. We anticipate that the design of the code will not be changed, but additions will be made with experience and new applications.

REFERENCE

1. Buell, W. and Bush, B.: Mesh Generation - A Survey. Transactions of the ACME, Journal of Engineering for Industry, Feb. 1973.



(a) 4002A.



(b) 4014-1.

Figure 1.- Tektronix terminal facilities.

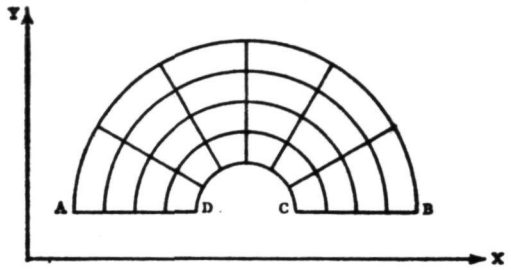


Figure 2.- Grid with equally spaced boundary points.

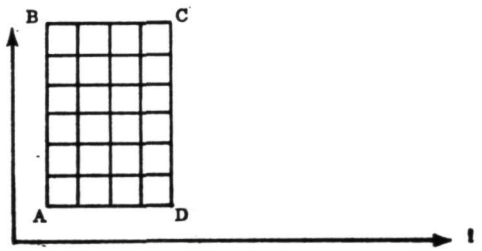


Figure 3.- I-J grid mapping from grid in figure 2.