N85-25869

NPLOT: AN INTERACTIVE PLOTTING PROGRAM FOR NASTRAN FINITE ELEMENT MODELS

Gary K. Jones and Kelly J. McEntire
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

SUMMARY

NPLOT (NASTRAN Plot) is an interactive computer graphics program for plotting undeformed and deformed NASTRAN finite element models. It has been developed at NASA's Goddard Space Flight Center. It provides flexible element selection and grid point, ASET and SPC degree of freedom labelling. It is easy to use and provides a combination menu and command driven user interface. NPLOT also provides very fast hidden line and haloed line algorithms. The hidden line algorithm in NPLOT has proved to be both very accurate and several times faster than other existing hidden line algorithms. It uses a fast spatial bucket sort and horizon edge computation to achieve this high level of performance. The hidden line and the haloed line algorithms are the primary features that make NPLOT unique from other plotting programs.

INTRODUCTION

Structural analysts at the Goddard Space Flight Center, GSFC, have always had the need to be able to graphically display finite element models quickly and accurately. Plots with depth cues give a much better visual representation and aid the analyst in interpretation and error checking. On a vector type of graphics device (such as Tektronix 4014's or pen plotters) the two best ways to show depth is via hidden line plotting or by haloed line plotting. The problem with the available hidden line algorithms is that they are normally time consuming and interfere with the quick response time desired in interactive graphics. One of the authors, Gary Jones, has developed a hidden line algorithm that satisfies these needs. This algorithm provides fast and accurate hidden line plotting of finite element models. The response time to plot a hidden line view of a model is near that for a normal all lines visible plot and provides linear time performance. A variation of this algorithm was used to produce a fast haloed line plot routine. A haloed line plot shows all aft lines broken to show depth. It is particularly well suited for plotting models composed of many line elements and few surface elements. For this class of models, hidden line plotting is not an effective tool.

This paper describes the current version of NPLOT. First, the development of NPLOT is discussed. Second, a description of NPLOT is

given, describing the many useful features found in NPLOT. Third, a detailed discussion of the hidden/halo line algorithm is presented along with benchmark performance data of NPLOT compared with other hidden line algorithms. Finally, concluding remarks are presented followed by references and figures.

NPLOT DEVELOPMENT / GOALS

The development of NPLOT was informally initiated in 1981 as simply a base for testing haloed and hidden line algorithms. Once these algorithms were developed and had proved to be very effective tools for model display, the main goal became to develop NPLOT into an effective tool for the structural analyst. NPLOT was and is currently being developed to meet specific goals and targets. The prime development target for NPLOT is that it must be an effective state-of-the-art graphics tool for GSFC structural analysts. Some specific requirements are:

1. NPLOT must effectively support the NASTRAN structural analysis code being used at GSFC.

2. NPLOT must run on the Digital Equipment Corporation VAX computers used by the Engineering Directorate at GSFC and support the available graphic hardware; i.e., Tektronix and Raster Technology terminals and Hewlett Packard pen plotters.

3. NPLOT must provide fast interactive performance together with a easy to use human interface.

4. NPLOT must provide effective graphic tools such as haloed and hidden line plotting.

NPLOT used as its starting point the PLOT code developed at GSFC by M. Weiss and M. Johns for the plotting of NASTRAN finite element models; however, at its current state of development, NPLOT contains almost none of that original code. Original algorithms and routines were developed for haloed line, hidden line, and horizon edge computation. A new executive was developed together with a better and more complete NASTRAN interface. It is expected that NPLOT will continue to evolve to meet new requirements; some of the near term activity will focus on:

1. Develop and add a fast shaded color hidden surface algorithm to NPLOT. The preliminary concept for the algorithm has been developed but it remains to be implemented and debugged. This algorithm would enable NPLOT to provide effective support for displaying model stresses, energy levels, and temperatures.

2. Add mode shape animation capability to NPLOT. This feature would make use of the multiple bit planes on the Raster Technology terminals to provide film strip animation.

3. Interface NPLOT with the Integrated Analysis Capability, IAC, program [1,2] developed by Boeing Aerospace Company for GSFC. This would provide NPLOT with good access to a wide spectrum of useful NASTRAN output data.

4. Investigate the feasibility and effectiveness of running NPLOT on an IBM PC-AT desktop computer system.

DESCRIPTION OF NPLOT

NPLOT has proved to be a very useful and versatile computer graphics program. It meets most of the plotting requirements for those who use NASTRAN at GSFC. It is also in use at NASA/LaRC, NASA/JSC and several GSFC contractors. In this section the NPLOT implementation will be discussed first, followed by a description of NPLOT's features and then a few words on NPLOT's user interface.

Implementation:

NPLOT was developed on a DEC VAX computer running the VMS operating system. The graphics was developed using a Tektronix 40XX storage tube terminal and a Tektronix 4105 raster terminal. NPLOT makes graphics calls to Precision Visual's DI3000 graphics subroutines. This subroutine package follows the Core standard. Making calls to DI3000 allows NPLOT to be device independent and therefore can be run on any terminal that has a DI3000 device driver. For those sites that do not have a license for DI3000, a set of interface routines have been developed that translate the DI3000 calls used in NPLOT into Tektronix PLOT10 calls.

Extensive use of the structured programming constructs and character manipulation functions of FORTRAN 77 are incorporated into the computer code. The character manipulation functions allow NPLOT to efficiently process NASTRAN free field bulk data decks. Non-standard FORTRAN 77 statements were avoided to allow the code to be transportable. The only problem that may occur when compiling NPLOT with a non VAX/VMS FORTRAN 77 compiler may be with a few open statements. Note however, NPLOT does make use of the virtual memory feature of VAX/VMS to speed operation and simplify implementation; this fact could make transfer of NPLOT to a non-virtual memory computer difficult.

Geometry may be entered in rectangular, cylindrical, and spherical coordinate systems using the CORD2R, CORD2C and CORD2S NASTRAN cards. The CORD1R, CORD1C and CORD1S cards are not supported. The coordinate systems may reference other coordinate systems since it is not a requirement that each reference the basic system. This combination allows a tree structured geometric system to be processed. The NPLOT user may, at his command, output a table to disk containing the grid point ID's and the XYZ coordinates in the basic system.

Features:

Clearly the most important feature of NPLOT is its ability to create hidden line and haloed line views of mathematical models both quickly and accurately. The hidden line algorithm generates views of models with all hidden lines removed, figure 1. The haloed line algoritnm displays views with aft lines broken in an effort to show depth while keeping the entir model visible, figure 2 . A discussion of these algorithms follows in the next section. NPLOT, of course, also can plot a normal all lines visible view of a model usually referred to as a wire frame view, figure 3.

Another important feature of NPLOT that allows it to perform post processing is its ability to plot deformed shapes, figure 4. NPLOT reads the displacements from a NASTRAN F06 file. It can read either static displacements or eigenvectors.

All subcases or mode shapes can be read in at once. The displacements are written into a unformatted scratch file where they are available for rapid access when the user wishes to display a deformed shape. It is then a simple matter to enable the deformed shape, change subcases or mode shapes and change the scale factor for subsequent plots.

NPLOT allows the user to specify element filters which select specific elements for plotting. Elements can be selected based upon their type, property, and ID. Elements can also be selected by a model segment. This is accomplished by inserting special segment delimiters in the bulk data deck and then specifying the delimiter label during the interactive session. Any or all of these filters can be activated at the same time to allow a great deal of selectivity. The elements can then be labelled with their respective ID's with a simple command once the plot is displayed on the screen, figure 5.

NPLOT also allows the user considerable flexibility in specifying which grid points are to be labeled with their grid point ID's. Specific grid point ID's can be selected and an eight character name tag can be associated with each grid point. The name tag can be useful for distinct identification. The user can also specify SPC sets, ASET and OMIT grid points to be labelled. These grid points will then be labelled with name tags indicating the degrees of freedom involved, figure 6.

NPLOT allows the standard display operations such as rotation and perspective. It also allows different view planes to be selected. These are X-Y, Y-Z and X-Z viewing planes. A zoom function is also allowed on terminals with a locator such as a graphics cursor, tablet, light pen or joy stick. The center of the area desired for zooming is selected with the locator and then a numeric key is pressed to indicate the zooming scale factor. Another display feature available is the Z-axis cut option which allows the user to cut away a percentage of the fore part of the model, figure 7. This is useful because it can reveal detail on the inside of a model.

The calculation of the model's horizon edges (edges where visibilty can change) is normally used just to speed up the hidden line computation. However, missing elements can be clearly located for most models by just plotting the horizon edges. NPLOT lets the user toggle the display set from all edges to just the horizon edges. Illustrated in figure 8 is a wire frame plot of the horizon edges for a model with a missing element.

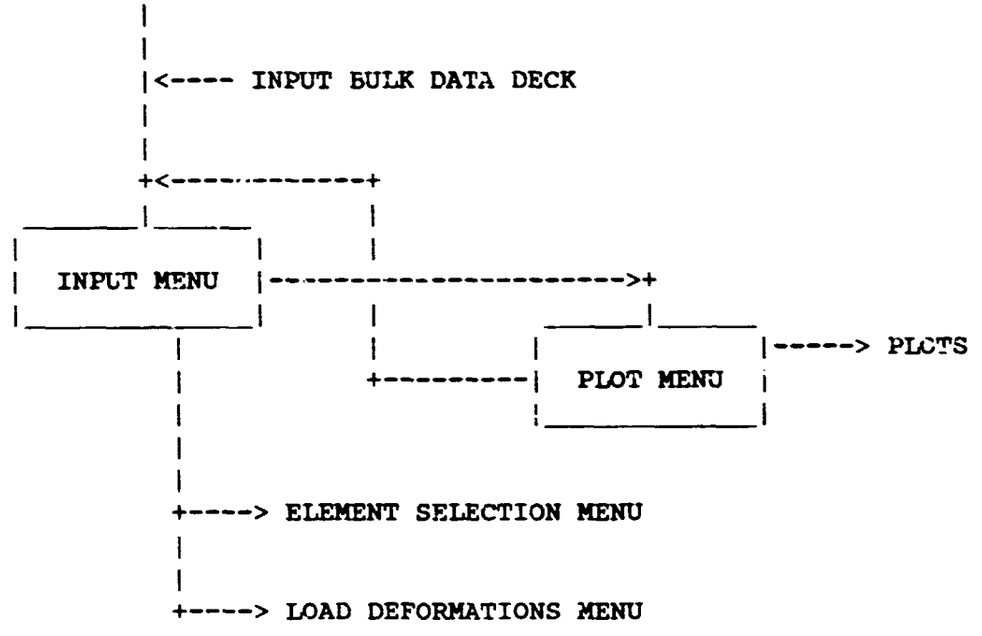
Another feature that aids the user is the plot file generator. Before beginning a plot the user can toggle on the plot file generator which will write all subsequent plot labels and screen vectors into a plot file until the toggle is turned off again. This plot file can then be read by other programs such a pen plotter program. The HP7580A pen plotter is used by GSFC's Mechanical Engineering Branch when larger and more precise plots are desired.

User Interface:

NPLOT'S user interface is intended to make NPLOT easy and quick to use. It is also intended to allow frequent users to become efficient at using the program. Frequent use will enable users to take short cuts once they are familiar with the instruction set. This aim is accomplished by using a combination of menu and command driven user interface with available help menus for detailed information

concerning each command. NPLOT is controlled via commands from two main menus. The basic command/menu structure is:

Within each of these menus are commands that invoke sub-menus. The input menu is used for selecting element sets, element labels, grid labels and deformation sets. The plot menu is used to select the display operations and then to execute the plot option desired. Once the plot is on the screen the interface becomes command driven. The same commands that were available from the preceding menus are now available for immediate execution. This enables the frequent user to avoid returning to the menu every time he wishes to manipulate the display or execute a command.



COMMAND MENU STRUCTURE

HIDDEN LINE / HALOED LINE ALGORITHMS

Hidden Line:

The development of a new hidden line algorithm was not taken lightly. Techniques to perform hidden line plotting have been much discussed beginning with the advent of computer graphics in the early 1960's and continuing into the present era. Given the bulk of this prior work [3-11], why develop a new method? The answer is that these prior methods, as of 1980, appeared to lack the speed necessary for effective interactive use, lack features necessary to plot NASTRAN models, or the referenced papers provided insufficient implementation details. Except for the Watkins technique [11], coded algorithms were not available. Experience in using the Watkins technique had shown it to be not acceptable for hidden line plotting of NASTRAN models. References 12 through 16 were published after our algorithm had been substantially completed.

Several different variations of the same basic hidden line method have been sequentially developed by G. Jones in the course of this effort. To keep track of the different versions, they were assigned names JONES/A through JONES/E. JONES/D was used in the first production version of NPLOT and was described in reference 17. The fastest and most recent version of the hidden line algorithm, JONES/E, is incorporated into the current version of NPLOT. The basic flow for JONES/E is as follows:

1. INPUT: The main inputs to JONES/E from NPLOT are the global edge list, global surface list, edge/surface adjacency table and grid point table. It should be noted that NPLOT operates to produce nonredundant global edge and surface lists. The global surface list uses a four node flat surface representation; NPLOT processes triangles through 20 node solid elements to this surface data format.

2. PREPARATION: The edge and surface lists are processed to produce arrays for edge and surface data. For example, the minimum/maximum X, Y, and Z values for each edge and surface are computed. The horizon edges of the object for the viewing transformation are computed. Spatial sorting of the edge and surface data is performed. Illustrated in figure 9 is a simplistic view of the spatial sort cells used by JONES/E. Based on the complexity of the model, N x N mesh X-Y sort cells are imposed on the model and lists of pointers to horizon edge data and surface data are generated for each cell via bucket sorting. Different mesh densities are used for edge and surface sorting. The mesh density used for the horizon edge sort is based on the total number of edges in the model. The mesh density for surface sorting is based on the number of surfaces in the model. The functional relationship between these measures of model complexity and mesh densities are set heuristically by varying the mesh density and observing the resulting performance for a number of models. JONES/E currently uses mesh definitions derived for JONES/D and so may not be optimum. After the cell lists are created, they are depth sorted based on the depth of the horizon edge or surface.

3. EDGE VISIBILITY: The global edge list is processed in two passes. On the first pass only the horizon edges are processed and the remaining edges are processed on the second pass. In either case the subsequent loop operations are the same. The edge cell coordinates for the edge are determined via a look-up table. The edge cells associated with the edge are binary searched to find the depth to limit the search for horizon edges that intersect with the edge. Its intersections with all horizon edges, that have not been found to be invisible by a prior calculation, are determined. The edge is broken into segments using its end points and the points of intersection. Each segment is either all visible or all invisible. The mid-point of each segment is computed and checked against the appropriate cell surface list to ascertain visibility. This requires computing the surface cell coordinates for the mid-point and then performing a binary search to find the depth in the surface cell list to limit the search for obscuring surfaces. Containment and depth computations are then performed to ascertain mid-point visibility, and hence, segment visibility.

This algorithm was tailored to support the plotting of NASTRAN models, therefore in the current implementation:

1. A line penetrating a surface usually results in a visible plot error. This is desirable for NASTRAN plotting since this usually indicates a modelling error.

2. Grid points are required where elements meet. This is normally the case in NASTRAN models.

3. Surfaces must be planar for accurate plotting. This is true for commonly used NASTRAN elements.

In operation, the algorithm is remarkably fast for plotting NASTRAN models. There are two chief reasons for this speed. The first being the efficiency of the spatial sort and the horizon edge technique in reducing the number of edge to edge compares in computing the required line intersections. The second being the effectiveness of the spatial sorts in reducing mid-point to surface compares in computing edge segment visibility.

The spatial sort functions as a divide and conquer technique; this is facilitated by the fact that in general NASTRAN models have fairly uniform topological granularity and the relative granule size decreases as model size increases. Thus, the spatial sort serves to linearize the operation of the algorithm. It is worth noting that Writtram [14] in a article published concurrently with the development of NPLOT used a horizontal strip form of spatial sort to achieve a high speed hidden line algorithm. The form of the spatial sort in JONES/E, and in the prior JONES/D, algorithm is somewhat different from Whittram's in that the spatial cells are boxes not strips and the fact that in JONES/E (and JONES/D) the hidden line determination does not make use of the concept of an active edge list or an active polygon list to reduce the computations.

The main difference between the current JONES/E algorithm and the prior JONES/D is the addition of the horizon edge method to the algorithm. A horizon edge is any model edge across which visibility can change. The concept of using horizon edges in hidden line computation was noted by Appel [18] and more recently used by

Hornung [16] to generate a very high speed hidden line algorithm for closed single surface objects. In JONES/E, horizon edges are computed and used to drastically reduce the number of edge compare operations. On a per cell basis, the reduction in operations is from the order of total edges squared to horizon edges times total edges. The net effect of using horizon edges was to speed up the algorithm by about 50 percent.

The authors make no claim to have "solved the hidden line problem". Contrary to what some have claimed, it appears that no existing algorithm is effective for the full spectrum of common topologies, for example curved surfaces. Inadequate research and a lack of understanding of the problem are usually evident when such claims are put forth. JONES/E was simply designed to process NASTRAN models or other similar topologies in an efficient manner.

Haloed Line:

The use of haloed line plotting was first discussed by Appel [5]. In haloed line plotting, the aft edges are broken where they intersect with more forward edges; this produces a well defined depth effect for the viewer, figure 2. The initial haloed line code for NPLOT was written by T. Carnahan, GSFC, based on the techniques defined by Appel [5]. The haloed line algorithm in the current version of NPLOT has been recoded by one of the authors (G. Jones) to incorporate the same spatial sort techniques as in the JONES/E hidden line algorithm and thereby increase its speed of operation.

Haloed line computation requires much more line intersection calculation than hidden line plotting thereby increasing the cpu time. Whereas hidden line plotting effectively truncates the total edge list, haloed line plotting operates to increase the total edge list by splitting up edges into several segments Thus, haloed line plotting generates more terminal I/O than wire frame or hidden line plotting. For these reasons, haloed line plotting should be slower than the other plot types.

In haloed line plotting the NPLOT user can specify the size of the gap so as to produce different effects. Haloed line plotting can be very effective in certain situations:

1. For models with few surface elements but many line elements, CBAR's and CROD's, haloed line plotting is very effective at showing depth information. Hidden line plotting is ineffective for this type of model.

2. When the user wants to peer inside a model but retain depth cues. haloed line plotting is an effective technique. This is similar to allowing transparency in hidden surface plotting on raster devices.

Performance:

The basic performance of the three plots type in NPLOT were assessed by measuring their performance with a collection of 20 NASTRAN models. The model sizes ranged from 55 grids/126 edges/67 surfaces up to 3730 grids/7547 edges /3626 surfaces. Wire frame and hidden line plots of the largest model are shown in figures 10 and 11. The performance of the algorithms were measured in terms of a processing rate expressed in terms of edges per cpu second. The cpu times were measured on a normally loaded VAX 11/780

computer and included the time to perform any preparatory work, execute the plot function module (wire frame, haloed, hidden), run the PLOT10 module, and perform the I/O to paint the object on the screen.

Shown in figure 12 is a graph of the measured performance of the three algorithms. All three algorithms show fairly linear performance for the range of models used in the tests. Wire frame plotting yielded an average rate of about 300 edges per cpu second, hidden line plotting was somewhat slower at about 150 edges per cpu second and haloed line plotting was the slowest at around 100 edges per cpu second. Wall clock response time for hidden line plotting was about the same as that for wire frame plotting. This was due to the fact that hidden line plotting involves less terminal I/O than wire frame plotting. Haloed line plotting was the slowest but this was expected. Even so, haloed line plotting was sufficiently fast to meet the demands of the interactive user. Haloed line plotting is the preferred plot type for models with few surfaces and many line elements.

The net effect of the various optimizing techniques employed in the hidden line algorithms can be seen from our experience in plotting one of the test models. The first - cut hidden line algorithm was a basic brute force line intersection technique with little code optimization; its processing rate for the test model was about 2.5 edges per cpu second. A subsequent version with more code optimization and a few simple short cuts worked at a rate of about 18 edges per cpu second. The JONES/D algorithm, which used the X-Y spatial sort, performed at a rate of about 100 edges per cpu second. The JONES/E algorithm in the current version of NPLOT uses the X-Y spatial sort together with the horizon edge technique, and achieves a processing rate of about 175 edges per cpu second for this particular model. Thus in this instance JONES/E performs about 70 times quicker than a brute force method.

How fast can an optimum hidden line algorithm run? A reasonable bounding upper limit might be the speed for wire frame plotting. For the particular hardware / software used at our computer facility (VAX 11/780, FORTRAN 77, Tektronix terminals using 9600 baud) the wire frame process rate was about 300 edges per second, thereby implying that no hidden line algorithm could run more than twice as fast as JONES/E for this particular computing environment.

The Watkins hidden line/surface method [11] and Hedgley's algorithm were compared to the JONES/E algorithm via comparative testing. The MOVIE program uses the Watkins method for hidden line and hidden surface computation. A VAX implementation of MOVIE was used for this study. The MOVIE implementation of Watkins does not support line element types so an all surface model was used to make the comparison. The test model consisted of 857 surfaces and 1242 edges. The cpu time for just the hidden line generation in MOVIE was 41.5 seconds; the corresponding time for NPLOT was 6.9 cpu seconds. The SKETCH hidden line routine developed by Hedgley [13] was obtained and converted to the VAX 11/780 computer. The routine as delivered was limited to about 250 polygons; therefore, a relatively small model was used for testing, 183 surfaces/324 edges. The cpu time for SKETCH was 19.3 cpu seconds and the time for the JONES/E algorithm in NPLOT was 1.9 cpu seconds. The level of performance for SKETCH, about 9 polygons per cpu second, seems consistent with the data

presented by Hedgley [13]. In reference 13 the processing rate for SKETCH on a CDC 6500 computer, which is about the same speed as a VAX 11/780, was given as about 10 polygons per cpu second.

CONCLUDING REMARKS

The NPLOT computer graphics program has been shown to be an effective tool for the interactive display of NASTRAN finite element models. It offers a variety of state of the art tools to aid the analyst. It is easy to use and provides an on line help facility for the inexperienced user. NPLOT's very fast hidden line and haloed line algorithms are unique and effective graphics tools for the analyst. Analysts using NPLOT usually prefer hidden line or haloed line plots in place of wire frame plots due to the more realistic model display. Current activity is focused on increasing the post-processing functionality of NPLOT.

REFERENCES

1. Walker, W., and Vos, R., *IAC Executive Summary*, NASA CR-175196, May 1984.

2. Vos, R.G., Beste, D.L., and Gregg, J., *IAC User Manual*, NASA CR-175300, July 1984.

3. Newman, W.M., and Sproull, R.F., *Principles of Interactive Computer Graphics*, McGraw-Hill Co., 1979.

4. Bareau, H., "Convenient Representation Method For Spatial Finite Element Structures", *Computers & Structures*, 10(5), October 1979, pp. 815-819.

5. Appel, A., Rohlf, F.J., and Stein, A.J., "The Haloed Line Effect for Hidden Line Elimination", *Computer Graphics*, 13(3), August 1979, pp. 151-157.

6. Franklin, W.R., "A Linear Time Exact Hidden Surface Algorithm", *Computer Graphics*, 14(3), August 1980, pp. 117-123.

7. Giloi, W.K., *Interactive Computer Graphics. Data Structures. Algorithms. Languages*, Prentice-Hall Inc., 1978.

8. Griffiths, J.G., "A Surface Display Algorithm", *Computer Aided Design*, 10(1), January 1978, pp. 65-73

9. Griffiths, J.G., "A Bibliography of Hidden-Line and Hidden-Surface Algorithms",*Computer Aided Design*, 10(3), May 1978, pp. 203-206.

10. Griffiths, J.G., "Tape-Oriented Hidden Line Algorithm", *Computer Aided Design*, 13(1), January 1981, pp. 19-26.

11. Watkins, G.S., *A Real Time Visible Surface Algorithm*, University of Utah, UTEC-CSc-70-101, June 1970.

12. Emery, A.F., *VIEW*, University Of Washington, Department of Mechanical Engineering, 1982.

13. Hedgley, D.R., Jr., *A General Solution to the Hidden Line Problem*, NASA RP-1085, March 1982.

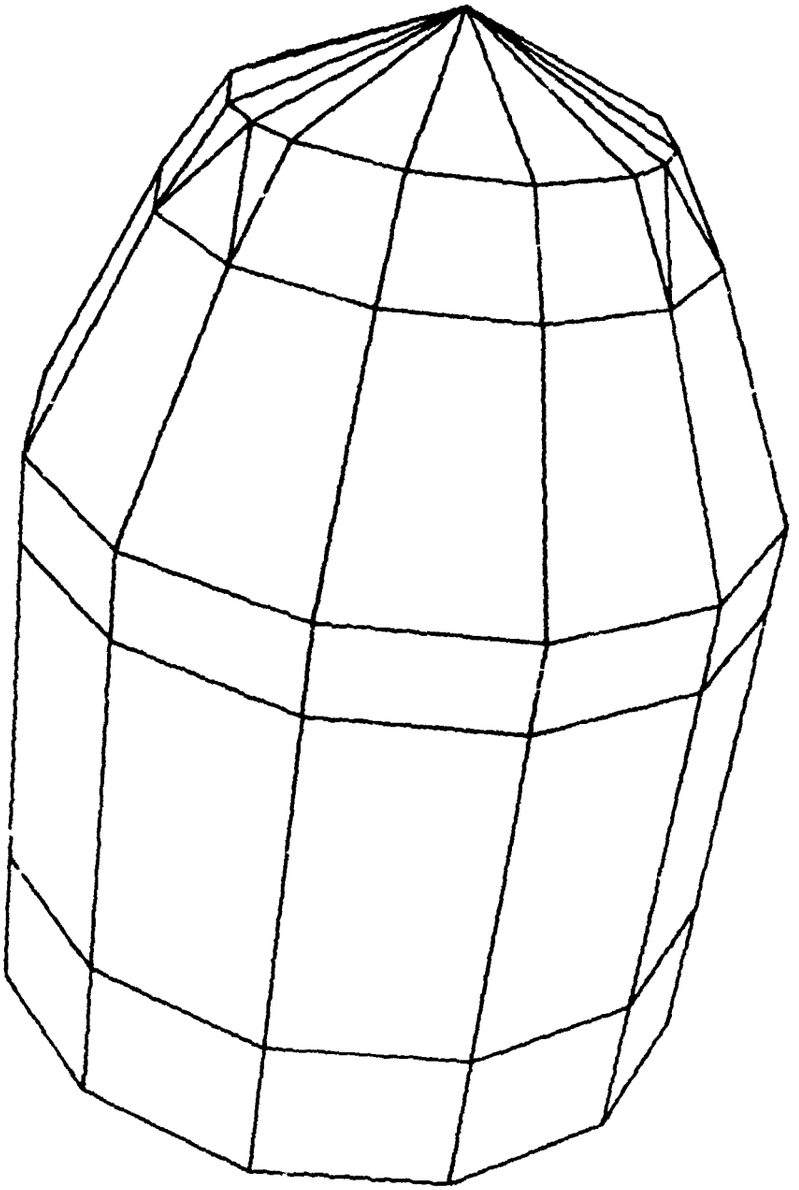
14. Writtram, M., "Hidden-Line Algorithm for Scenes of High Complexity", *Computer Aided Design*, 13(4), July 1981, pp. 187-192.

15. Janssen, T.L., "A Simple Efficient Hidden Line Algorithm", *Computers & Structures*, 17(4), 1983, pp. 563-571.

16. Hornung, C., "An Approach To A Calculation-Minimized Hidden Line Algorithm", *Computers & Graphics*, 6(3), 1982, pp. 121-126.

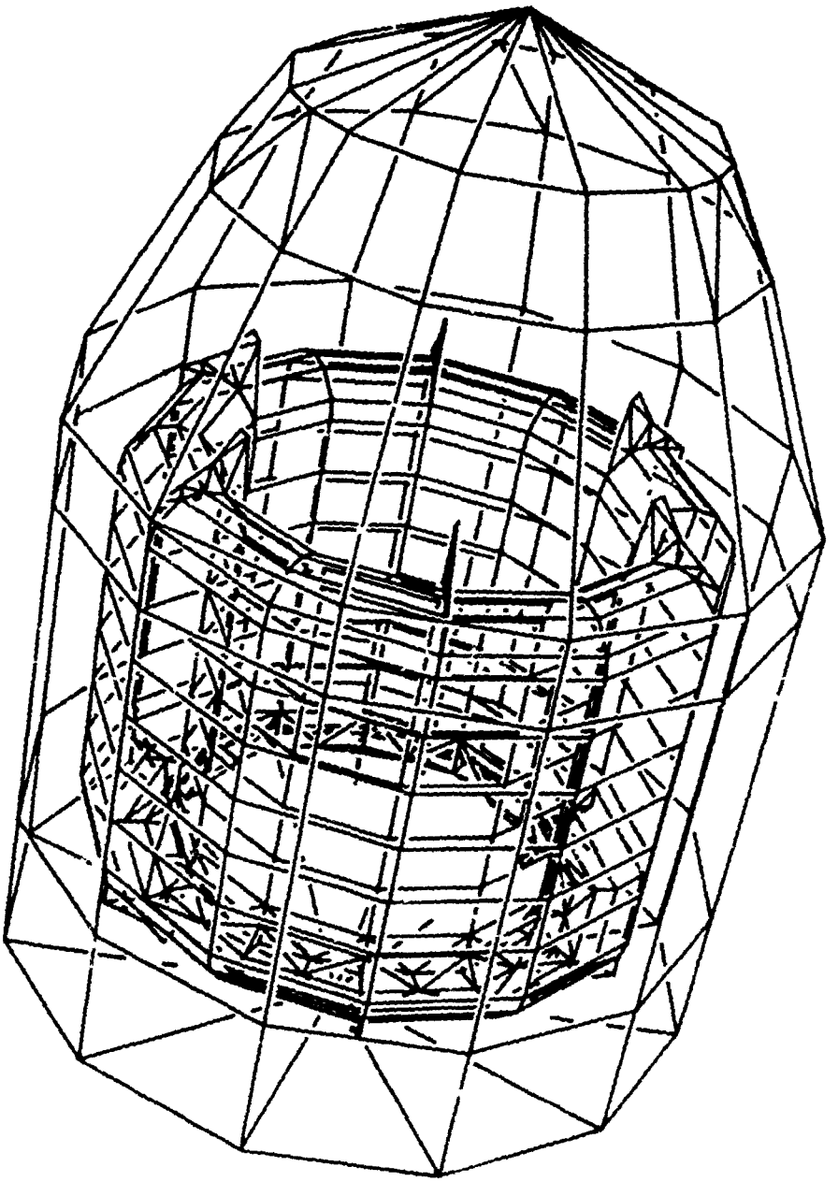
17. Jones, G.K., *A Fast Hidden Line Algorithm For Plotting Finite Element Models*, NASA TM-83981, August 1982.

18. Appel, A., "The Notion of Quantitative Invisibility and the Machine Rendering of Solids", *Proc. ACM National Conference*, 1967, pp. 387-393.



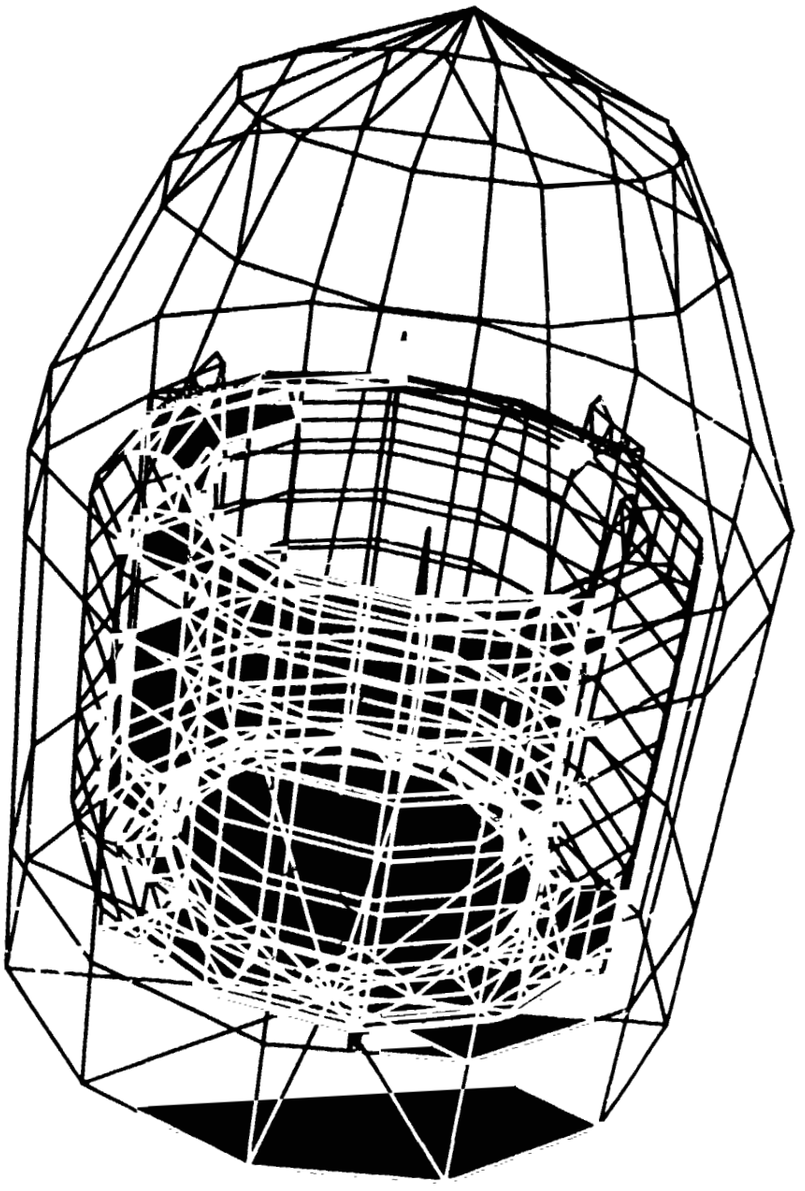
HIDDEN LINE PLOT

FIGURE 1



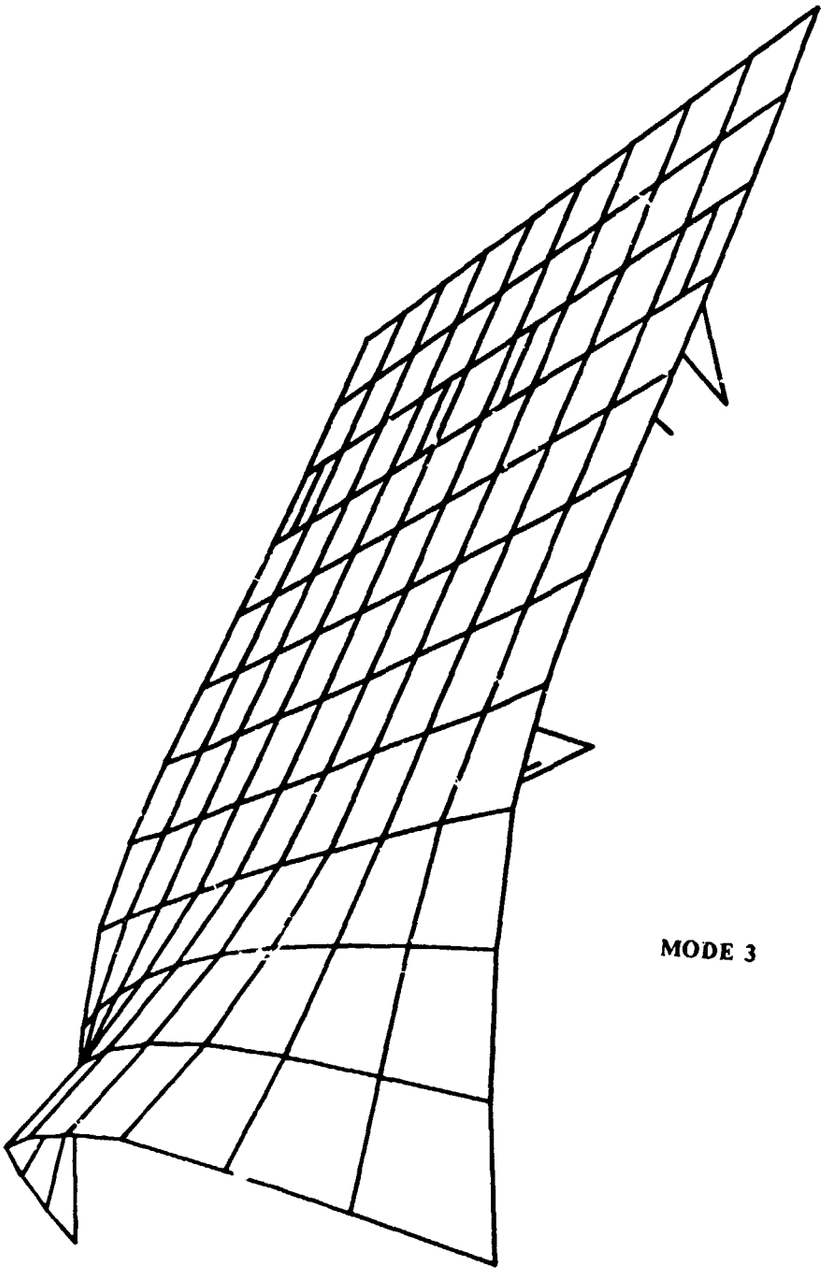
HALOED LINE PLOT

FIGURE 2



WIRE FRAME PLOT

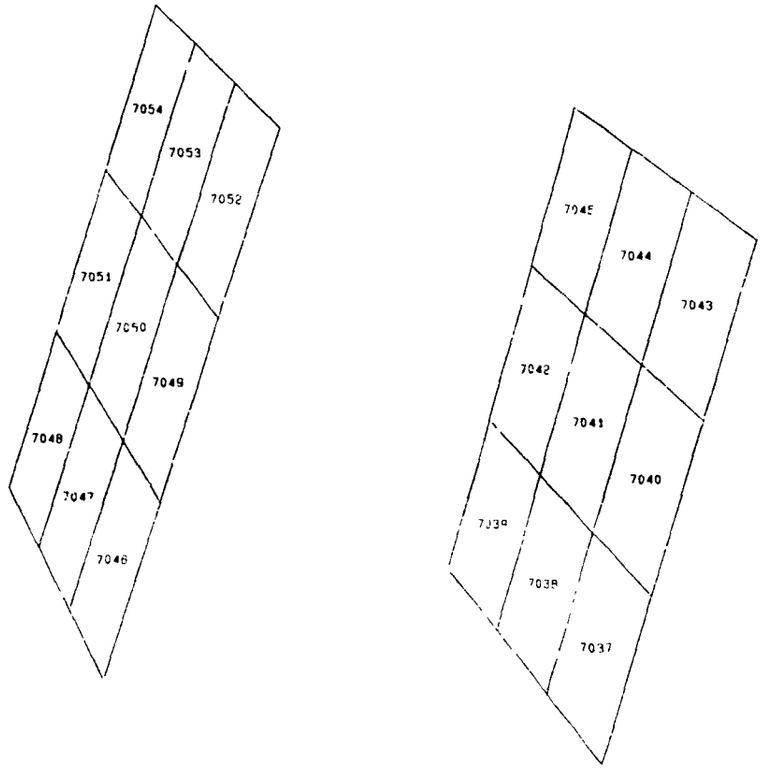
FIGURE 3



MODE 3

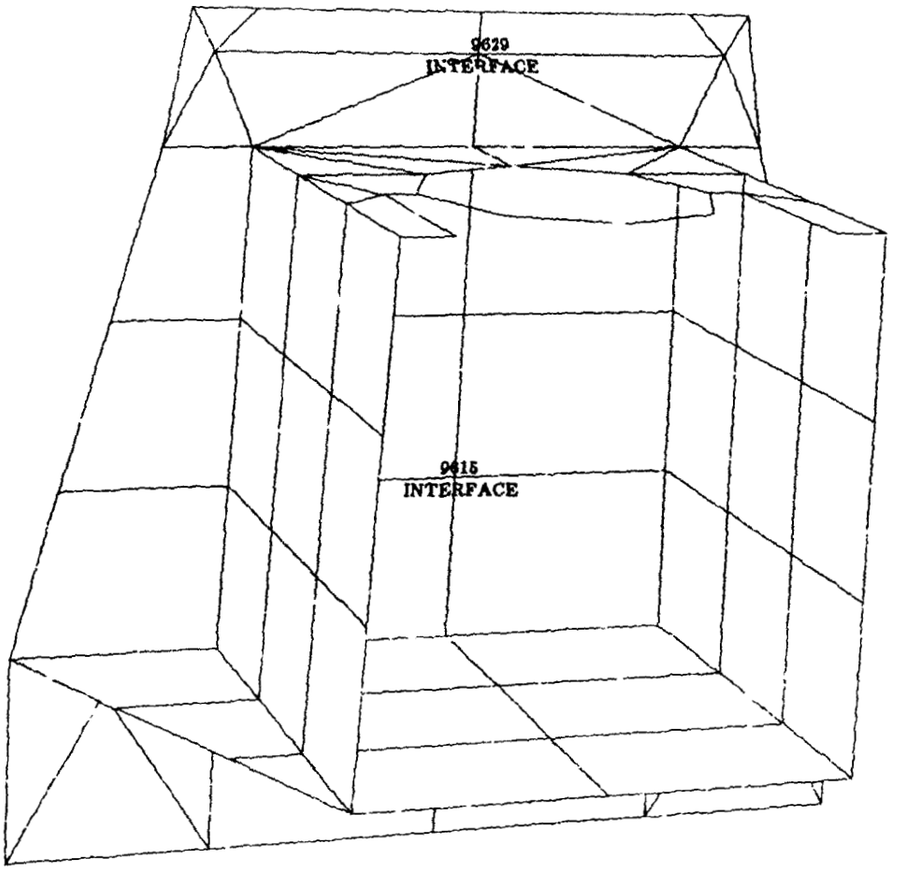
PLOT OF DEFORMED SHAPE

FIGURE 4



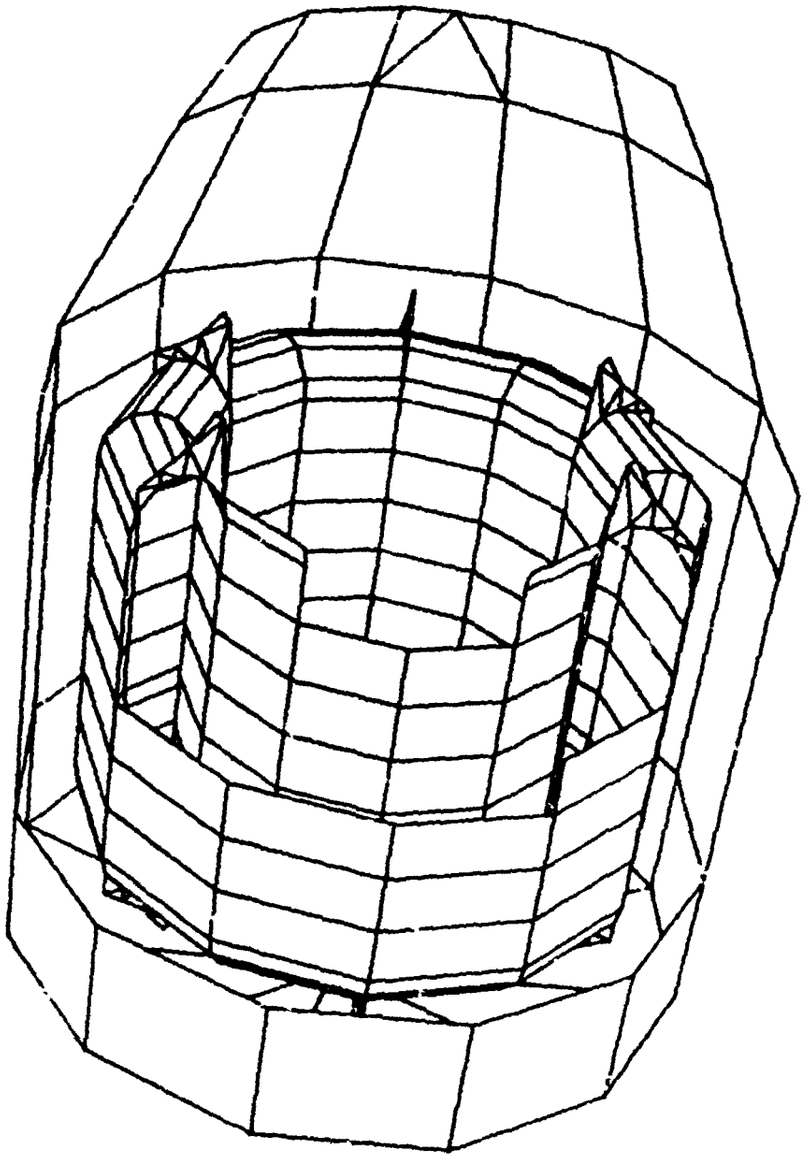
PLOT WITH ELEMENTS LABELLED

FIGURE 5



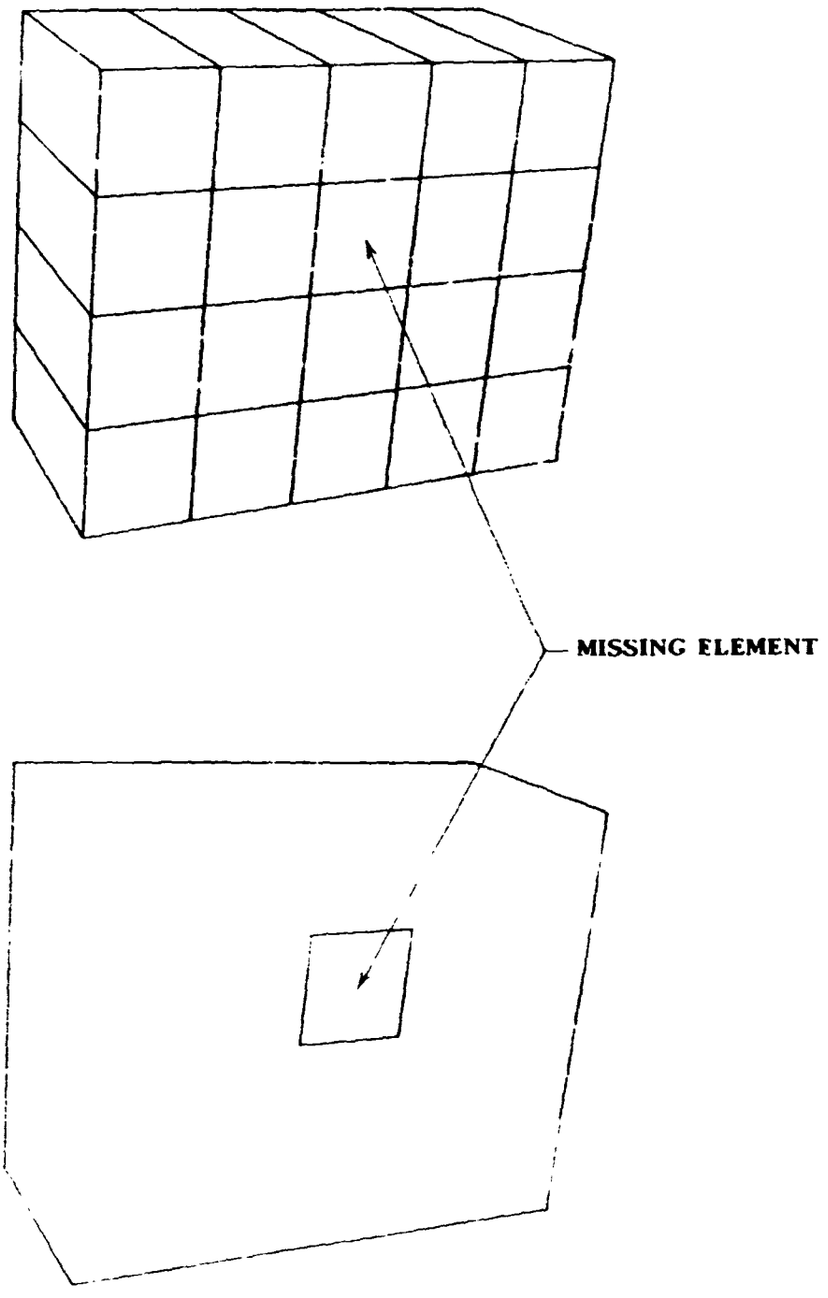
PLOT WITH GRIDS LABELLED AND TAGGED

FIGURE 6



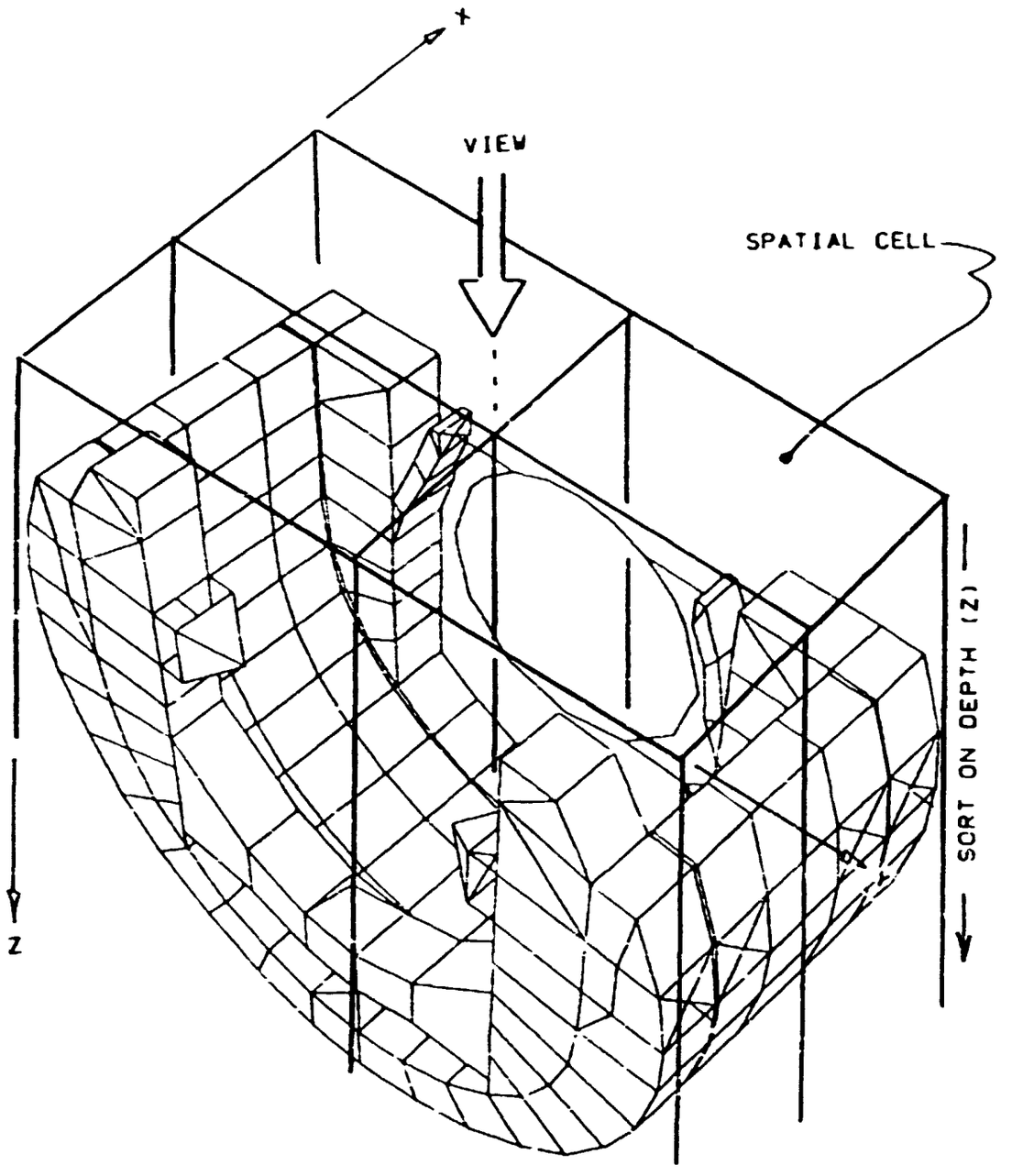
PLOT WITH Z-CUT

FIGURE 7



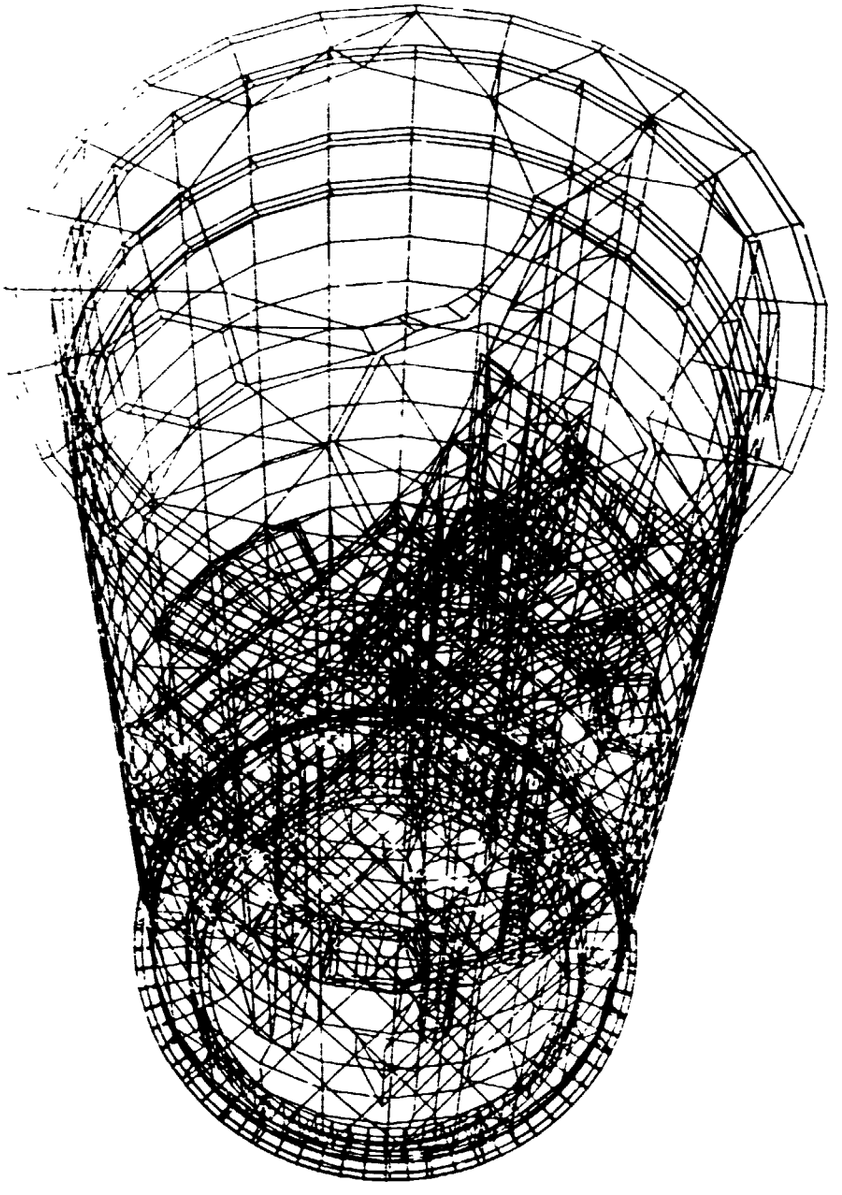
HORIZON EDGE PLOT

FIGURE 8



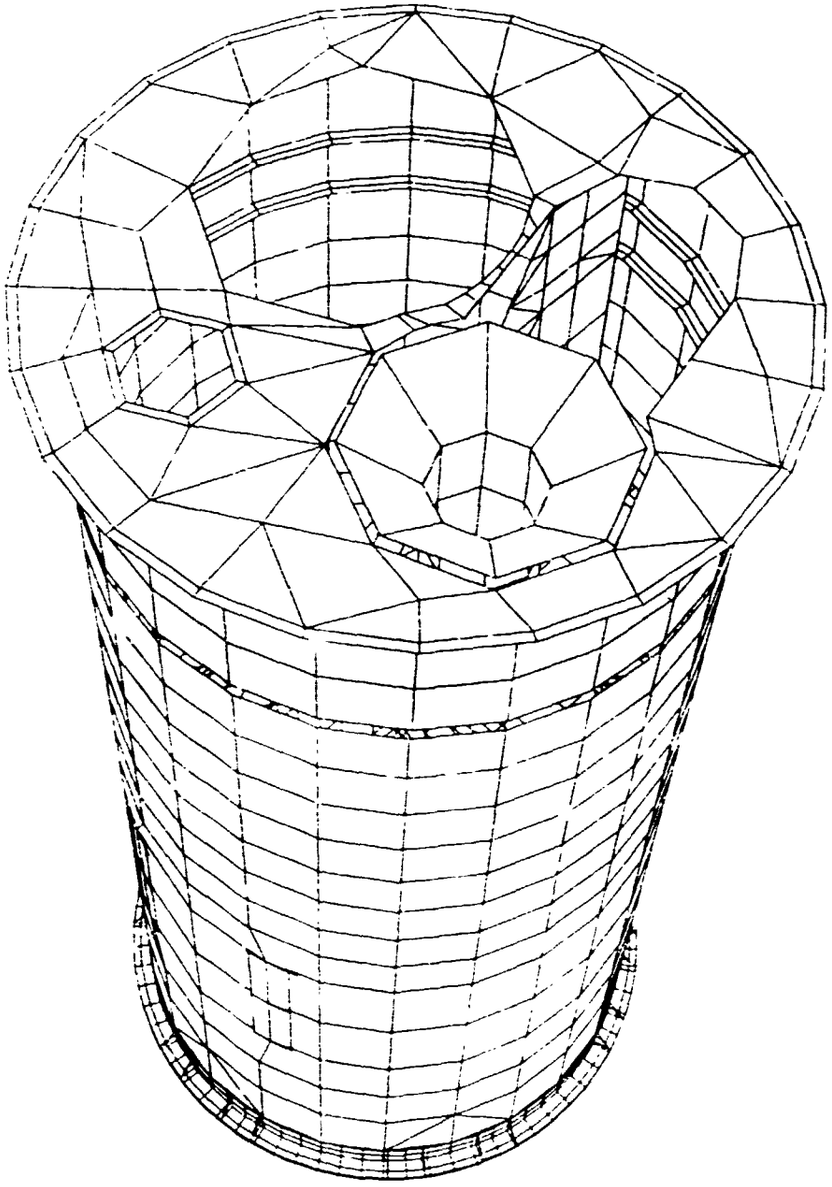
SPATIAL CELLS USED IN X-Y BUCKET SORTING

FIGURE 9



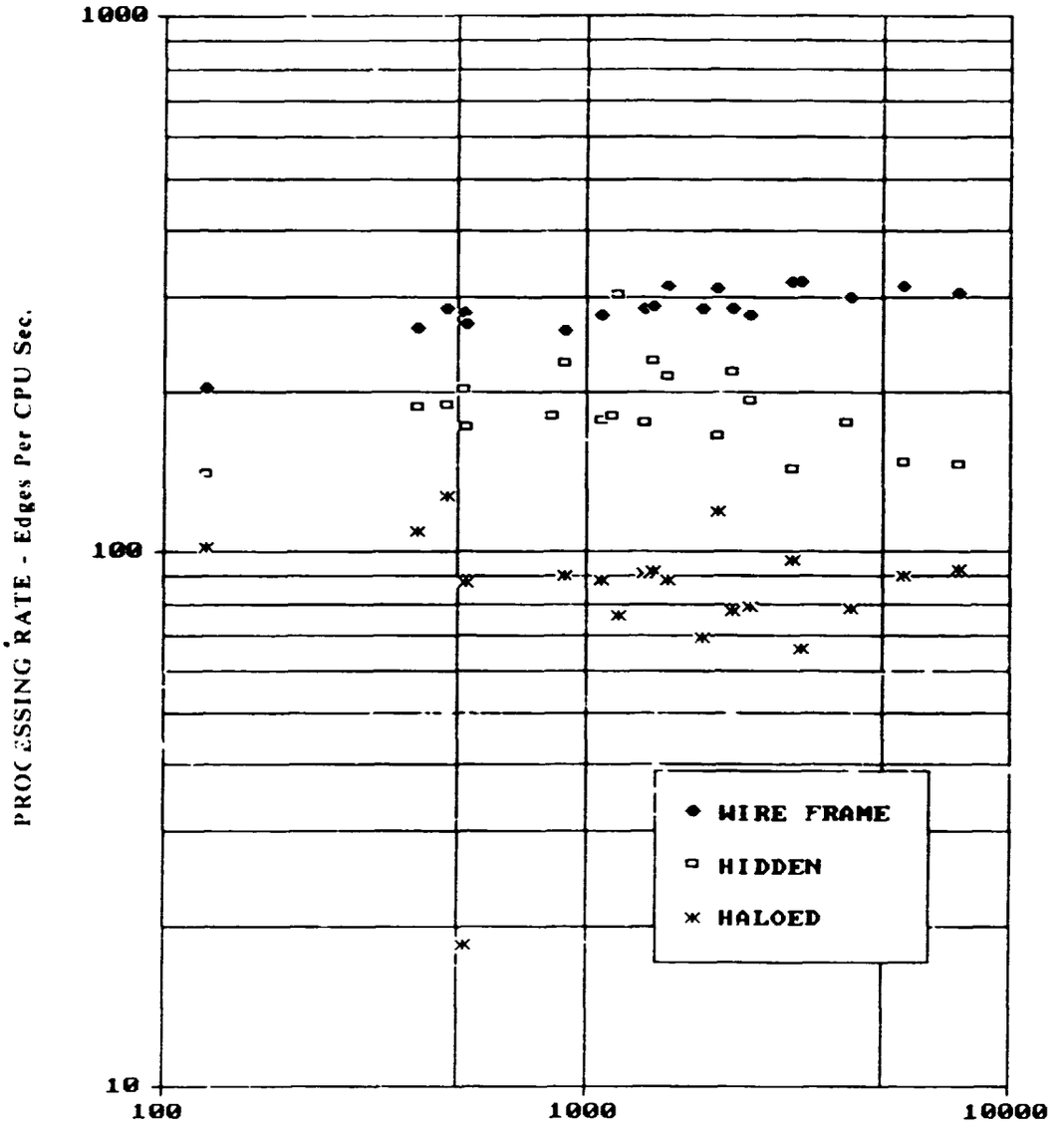
LARGE TEST MODEL - WIRE FRAME PLOT

FIGURE 10



LARGE TEST MODEL - HIDDEN LINE PLOT

FIGURE 11



PERFORMANCE OF NPLOT ALGORITHMS

FIGURE 12