NASTRAN PRE- AND POSTPROCESSORS USING LOW-COST INTERACTIVE GRAPHICS

E. D. Herness and H. Z. Kriloff

Boeing Computer Services, Inc.

ABSTRACT

Low-cost graphics are now available to the engineer. The low-cost storage tube terminal, the time-sharing computers, and the communication through the ordinary telephone are the hardware advances that now make low-cost graphics a reality. Time-sharing system software and an easy to use graphics library are the software that make developing pre- and post-processors easy and inexpensive. A design for a NASTRAN preprocessor is given to illustrate a typical preprocessor. Several displays of NASTRAN models illustrate the preprocessor's capabilities. A design of a NASTRAN postprocessor is presented along with an example of displays generated by that NASTRAN processor.

INTRODUCTION

A large part of the cost for structural analysis is in the preparation of correct input data (Reference 1). Pre- and postprocessors are effective in reducing the cost of input data preparation and output analysis (Reference 2).

The data used to define a NASTRAN model and the results generated by a NASTRAN analysis are graphical. This is the reason the plotting capability has been so widely used. The major problem with the plotting package in NASTRAN is one of computer operations. The time it takes to get a NASTRAN job scheduled, run on the computer and plots returned is often several days. This is too long a time for an analyst to effectively check his input in this manner. As a result many expensive NASTRAN runs are made that solve the wrong model. This suggests a NASTRAN preprocessor using interactive graphics.

A NASTRAN preprocessor using low-cost interactive graphics can reduce the time and cost of making a NASTRAN analysis. Input errors can be located and corrected in minutes at very low cost. The NASTRAN input deck is more correct before the first NASTRAN run is made and the analyst has the confidence of seeing and verifying the input model.

A NASTRAN postprocessor using low-cost interactive graphics can replace pages of output with a few graphical displays. A displacement vector or mode shape can be displayed and visualized in minutes that may take several hours to understand from the listing. The speed with which one can see and interpret the results and the ability to interactively choose the data and view it are the major advantages of a low-cost interactive postprocessor over the NASTRAN plotting package.

The purpose of this paper is to discuss the advances in hardware and software that make low-cost interactive graphics easy to use. The design and capability of a NASTRAN preprocessor is discussed along with several displays of input models. The design and capabilities of a NASTRAN postprocessor that displays the NASTRAN results are also discussed.

LOW-COST GRAPHICS HARDWARE

The major reasons for the increased utilization of computer-based graphics have been the steady decrease in the cost of the hardware required and the improved performance of these systems. The decrease in cost is due to improved technology and the economics of increased utilization. These factors carry through to each of the components that make up the graphics system: the host computer, the communications lines and the graphic terminal (see Figure 1). New trends may change the present techniques for implementing graphics, but the direction of increased interaction with graphics images will continue.

The advent of time-sharing computing systems has decreased the cost of using a computer because now the user pays for only the resources that he uses, not those he has access to. This improved utilization allows the user to consider applications where little computation is performed and data are merely manipulated and reformatted by the computer. These applications instead of wasting resources allow more users to use a given computer system, decreasing the cost to each user. The user also has available the improved and specialized software development aids that are only economical when utilized by many users and he can share data storage costs among a number of application systems. The user is now not restricted to a specific package or hardware system but can now choose that system that best suits the needs of his application. This then decreases the costs to modify the software so that it will fit a given hardware configuration and system.

The user does not need to be geographically near the host computer. Due to advances in the field of telecommunications, standard telephone lines can be used to carry the signals from the host computer to a remote terminal. Since the computer signals are digital and the telephone lines transmit analog signals only, a device is used to modulate the analog carrier signal with the digital signal, and to demodulate (separate) the signals at the other end. This device is called a modem (modulator-demodulator) and has decreased in price while increasing in data transmission rate.

The largest cost decrease has occurred in the development of graphic terminals. Increased usage, simplified technology and improved design have made graphic terminals easier to use and less expensive. The three different types of terminals that will most affect graphics development in the next few years are the direct-view storage tube, the plasma panel and the intelligent terminal.

The direct-view storage tube (DVST) is a modified form of cathode ray tube (CRT) where an electron beam strikes the phosphor-coated face of the tube. This excites the phosphor which then emits light when it returns to the normal unexcited state. In the DVST a negatively charged grid is placed between the phosphor and a low energy unfocused electron beam. The negative charge prevents the electrons in the electron beam from reaching the phosphor and exciting it. However, a second, high energy focused electron beam is used to remove electrons from the grid, allowing the phosphor to be excited at that location. Therefore, the grid serves as a memory for the image being drawn. The use of the electrostatic grid memory allows the system to have both a low speed connection between the terminal and the computer and to provide storage for very complex graphical images. The DVST is widely marketed as a commercial terminal, thereby providing decreased cost due to mass production economics.

The plasma panel is a newer, less complex technology. The unit is totally digital, eliminating the need for circuitry such as digital-to-analog converters and can serve as both an input and a display device. While presently not in wide usage, these displays and other matrix displays similar to these should gradually replace the CRT displays in the next decade. The advantages of compactness, simplicity, and the optical mixing of photographic and computer-generated images will gradually overcome the advantages of the presently more widely used displays.

The third influence in the low-cost terminal market is the potential effect of the intelligent terminal. This is a display device combined with a micro- or miniprocessor. These processors are already in the under $100 price range and can replace much of the specialized digital circuitry required by the present generation of terminals. Combined with the graphic display these processors provide a terminal that removes part of the processing load from the host computer, improves system response characteristics, and standardizes the user interfaces for many very different host computer systems. The advantages of personalized computing quickly outweighs any small additional cost (if any) required by adding the processor. The main deterrent to the widespread usage of such systems will be the high cost of developing software to support the terminal processing systems.

While the cost of the hardware used by graphic systems has steadily decreased, software prices have often increased the cost of using such systems. Since the software is as important as the hardware (sometimes more so since software can be used to modify the performance of the hardware while the hardware seldom changes the type of software required) the cost and usage of software is an important systems consideration.

The software required by a graphics system consists of a time-shared operating system, a higher level graphics language, and a graphics library. The time-shared operating system has two components that are very important for graphics systems. These components are the command interpreter and the text editor. The command interpreter provides the ability to select among a series of options, those programs, defaults and data files that will be used in a given task. The text editor is used to prepare programs and data for manipulation by the system.

A number of considerations must be addressed in the implementation of a higher level graphics language. These considerations are: the choice of an implementation language, the use of interpreters, the restrictions of device independent software and the use of general data structures.

The choice of a graphics language that would remove many of the difficulties of presently available languages is always a very tempting alternative. However, it takes from 7 to 15 years for a new programming language to be widely accepted no matter how many useful features are part of the language. Since graphics is only one part of the requirements for a programming language this long lead time prevents the utilization of a new language. However, there is a way to modify the syntax of a programming language without creating a new language. This is through the use of a precompiler that converts the modified syntax into some more widely used language. This gives the programmer the option to use the new syntax where it benefits the program development and the accepted language where the advantages of portability and greater utilization are more important.

Another choice in the implementation of a graphics system is whether the language should be provided as a compiler or as an interpreter. In a compiler the program code is translated into machine instructions as a complete entity. More than one language statement may be used to provide information about data structures, execution control or type of algorithm. Most of the execution decisions are made at this point allowing optimization of the final code. If an interpreter is used each statement is translated and immediately executed in the order determined by the program execution. This allows less optimization but greater flexibility in the execution of the program. As the graphic commands get higher level (i.e., each statement defines a more complex operation) the distinction between a compiler and an interpreter becomes less significant. The ideal mixture is a very high-level compiler allowing complex user-defined data structures and code optimization with statements that are mini-interpreters allowing greater freedom in the execution phase. Features such as data editing and execution control can thereby be utilized without recompiling the program.

The use of a compiler solves one other problem. Since there are a large number of graphic display devices, each with both desirable and undesirable features, the ability to use more than one device with the same program code (device independence) is an important feature of any graphics system. In an interpreter-based system the machine code for each display device would have to be part of the system and each time the program provided a display command it would have to check which device it was using. This would increase the size of the user program and decrease the execution speed. In a compiler this decision would be made only once and the system would load only those routines required by the devices being used. To change devices would require the recompiling or reloading of the user's program.

The data used to generate graphic images can exist in a number of different formats within the computer. These forms are dictated by the display hardware, the application program or the operations that may be performed on the data. When these forms are simple (array type sequences) a wide variety of graphic software can be developed to edit or enhance the image to be displayed. As the data structures become more complex, less graphic software is available as a standard part of the system and it becomes the user's responsibility to provide the necessary routines. Therefore, the more complex data structures are used only where they add to the utility of the information representation.

Thus, a system implemented as a high-level precompiler with a simple data structure provides the best mixture of efficiency and flexibility in a graphics system. Where a large enough library of primitive operations are available to the system user, graphics can be simply and usefully added to application programs.

THE GRAPHICS DISPLAY LIBRARY

Each of the hardware, software or user-directed operations are implemented by one or more calls to library subprograms. These subprograms provide capabilities for accurate, fast and easy generation of the display portion of the application program. In order to provide for the device independence described above, the system should consist of two types of subprograms: those that communicate with the terminal (called the low-level routines) and those that manipulate the user's data (called high-level routines). A simple, flexible interface between the two levels allows the implementor to exchange low-level routines when he changes terminal systems. The low-level routines are provided by the terminal vendor (Reference 3) and simply draw lines, erase lines, and change the terminal state.

The higher level routines are the routines called by either the user or the precompiler in order to express the sequence of operations required to generate an image display or a user interaction. These allow the user to select and define defaults for a graphics terminal (initialization) to define the logical program control and the display format. The program control can be expressed through menus, function keys, input options and other types of processing of input from devices such as keyboards and crosshairs. The displays are generated by defining coordinate systems (rectangular or polar), scales (linear, logarithmic, etc.), labeling and text generation. The scales are determined either automatically in order to fit the data or a section of the data (blowup) or explicitly by the user. Where data fall outside the screen scale, a scissoring routine is used to exclude all elements outside the region. Where the data are three-dimensional, routines are required to rotate the objects and to project the data into a two-dimensional system. Since all terminals do not have a device to generate a permanent copy of displays generated, a technique to preserve a representation of the object on the screen for later hardcopy plotting is also necessary. An example of a system that provides all these capabilities is the BCS (Boeing Computer Services) Interactive Graphics (BIG) System (Reference 4).

Once a system provides all the above facilities it can be used to generate programs that are easy to use, simple to implement and solve many different types of problems. Examples of the application of this system to NASTRAN are given in the following sections.

ADVANTAGES OF MULTIPLE PROCESSORS

In the implementation of a graphics display facility into an already existing computer program the designer has two options. He can insert the code into the program at the appropriate places or he can construct separate program modules that utilize the same data bases as the existing program. This latter choice often requires the reprogramming of certain operations that already exist in the functioning system. It also imposes additional overhead on the system because of increased disk input/output and data reformatting. However, in certain cases, the advantages of this second choice far outweigh the disadvantages.

A good reason for selecting the multiple processor form or design over the code insertion design is where the functioning program is already computationally bound. In that case the response time for the display part of the system will be increased due to the overhead calculations of the main program. Either complex logic must be incorporated into the program in order to improve system response or the display sections must be separated from the program logic sequence. Once this has been done the necessary additional programming for a pre- or postprocessor has been performed.

Another reason for using a separate processor is that as a program gets very large the slightest change tends to require a major effort. By building a separate processor the original program code is left undisturbed and the effort is minimized.

The last reason is that where the modules are small it is possible to create multiple versions of a subsystem for different classes of users. This can increase the utility of the program with very little additional effort since many sections of the programs will be identical.

Therefore the advantages for computer-bound NASTRAN tasks of improved response, minimal code disturbance and multiple versions led to the insertion of graphics displays using pre- and postprocessors.

NASTRAN PREPROCESSOR

An interactive graphics preprocessor for NASTRAN is a valuable addition to the NASTRAN capability. Errors in geometry and connectivity input can be identified and corrected in minutes from a terminal. These same errors may take days and several NASTRAN runs to find using a batch process with off-line plots. The relationship of the NASTRAN input display program to the rest of the system is shown in Figure 2. The time-share system command mode and editor are used to generate and update the NASTRAN deck and pass it to the NASTRAN Input Display program.

The program logic for the NASTRAN preprocessor is as follows: The program reads the NASTRAN deck. The grid cards are converted to the base XYZ system and stored in data arrays. The element connectivity cards are read and an element connectivity table is built. At the time the connectivity cards are read, a check is made to see that each grid reference on the connectivity card has been defined by a grid card. An error message is printed out that identifies the missing grid points. After the NASTRAN data has been read, the remainder of the program is executed interactively from the terminal. This provides the user with the opportunity to select those operations that contribute most to the data verification process. Since many types of errors might occur and only the user can detect most of them (that's why we use graphics) this type of operation is both cost-effective (less spoiled runs) and productive (the user does not have to observe useless data displays). The user then chooses the next step from the following menu:

DEFINE DISPLAY PARAMETERS
DEFINE DISPLAY SET
DISPLAY STRUCTURE

DEFINE DISPLAY PARAMETER allows the user to choose the view angle. By defining different view angles the structure may be rotated in any direction. Detail which is hidden because of a given orientation can be interactively changed to show clearly the structural idealization. Often several viewing angles are necessary in order to examine all of the geometry and connectivity of the model.

DEFINE DISPLAY SET allows the user to select any part of the model to be displayed. The linear elements, triangular elements and/or quadilateral elements within one or more ranges of element ID's may be interactively selected as the display set. The linear elements, triangular elements and/or quadrilateral elements may be selected to be displayed individually or in combination. In addition specific element ID's or a range of ID's may be selected. The set selection option allows the model to be displayed section by section. Duplicate lines can be displayed in separate views.

DISPLAY STRUCTURE causes the program to create the previously defined display with user-chosen viewing angles. The user has the option to display gridpoint ID's and/or element ID's. Another important capability of this option is the BLOWUP feature. The corners of a new display window are defined as two points on the display. The window defined by these points are the new scale limits. The structure is displayed so that the window is expanded to the boundaries of the display screen. This is an important capability in viewing complex structural details.

The NASTRAN preprocessor described above has been developed and extensively used. Figure 3 shows the NASTRAN demonstration problem 1-1 in a 3-dimensional view. The model includes several linear, triangular and quadrilateral elements. Figure 4 shows a spherical cap problem that is defined in the NASTRAN demo 1-2. The spherical cap model has been rotated counterclockwise so that the gridpoint ID's are more visible. Figure 5 shows a blown up portion of the same spherical cap. Notice the lines on the right and at the bottom are clipped at the display boundary.

The initial capability of the preprocessor includes the display of the gridpoints and element connectivity. Planned extensions include the display of single point constraints, multipoint constraints, forces and concentrated masses.

NASTRAN POSTPROCESSOR

A NASTRAN postprocessor that interactively displays deflections and mode shapes has been developed from the NASTRAN preprocessor capability. The source of the input data is now the checkpoint tape rather than the NASTRAN input deck. Usually it is advisable to extract from the checkpoint tape the data blocks BULKDATA, EQEXIN and UGY and save them on a separate file. The NASTRAN postprocessor then reads that file and builds the geometry array, the element connectivity and the displacement vectors. The highest level menu of the NASTRAN postprocessor is:

DEFINE DISPLAY PARAMETERS

DEFINE DISPLAY SET

DISPLAY UNDEFORMED STRUCTURE

DISPLAY DEFORMED STRUCTURE

DISPLAY UNDEFORMED AND DEFORMED STRUCTURE

The DEFINE DISPLAY PARAMETERS option allows the user to define the viewing angles, choose the subcase displacement vector and define the magnification factor on the displacement vector.

The DEFINE DISPLAY SET option is identical to the option in the NASTRAN preprocessor.

The three DISPLAY options display the undeformed structure, the deformed structure or both the undeformed and deformed structure. The user may choose to display the gridpoint ID's, the element ID's or both. On any display the user may choose to blow up a portion of the structure by selecting a viewing window defined by two points. That rectangle is then expanded to the full screen area.

The above program has been developed for the BCS MAINSTREAM-EKS System that runs on the CDC 6600 using the BCS Interactive Graphic System. Figure 6 shows the ninth vibration mode of the helicopter structure used in Reference 5.

CONCLUDING REMARKS

Advances in computer hardware, the low-cost graphics terminal and data communications make low-cost graphics feasible. Advances in graphics software allow the user to easily develop interactive pre- and post-processors for NASTRAN. The preprocessor provides a very effective way to interactively check the input data. The postprocessor provides immediate display of the results on low-cost interactive terminals.

REFERENCES

1. Tocher, J. L. and Felippa, C. A.: Computer Graphics Applied to Production Structural Analysis. Proceedings of the Symposium of the International Union of Theoretical and Applied Mechanics, Liege, Belgium, August 1970.

2. Herness, E. D. and Tocher, J. L.: Design of Pre- and Post-Processors. Proceedings of the International Symposium on Structural Mechanics Software, University of Maryland, June 1974.

3. Information Display Products: Tektronix Plot-10 Terminal Control System. Document No. 062-1474-00, Beaverton, Oregon (1972).

4. Quenneville, C. E.: BCS Interactive Graphic System. Boeing Computer Service Document No. 10201-044, June 1975.

5. Tocher, J. L. and Herness, E. D.: A Critical View of NASTRAN. Numerical and Computer Methods in Structural Mechanics, Academic Press, Inc., N.Y., 1973, pp. 151-173.

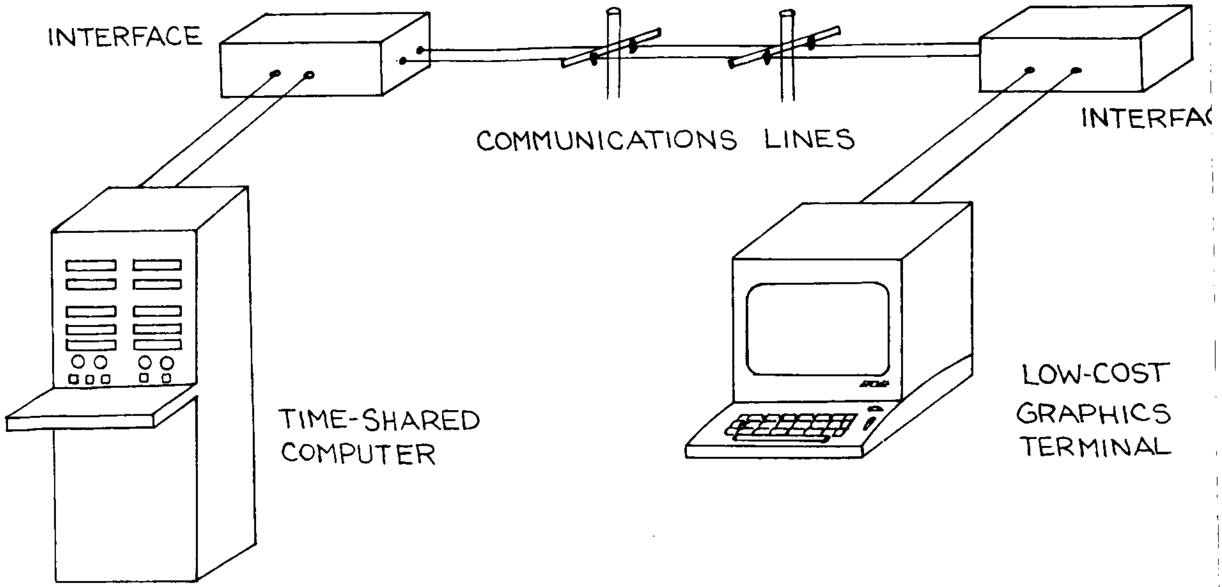

Figure 1.- Hardware Elements of a Low-Cost Graphics System.

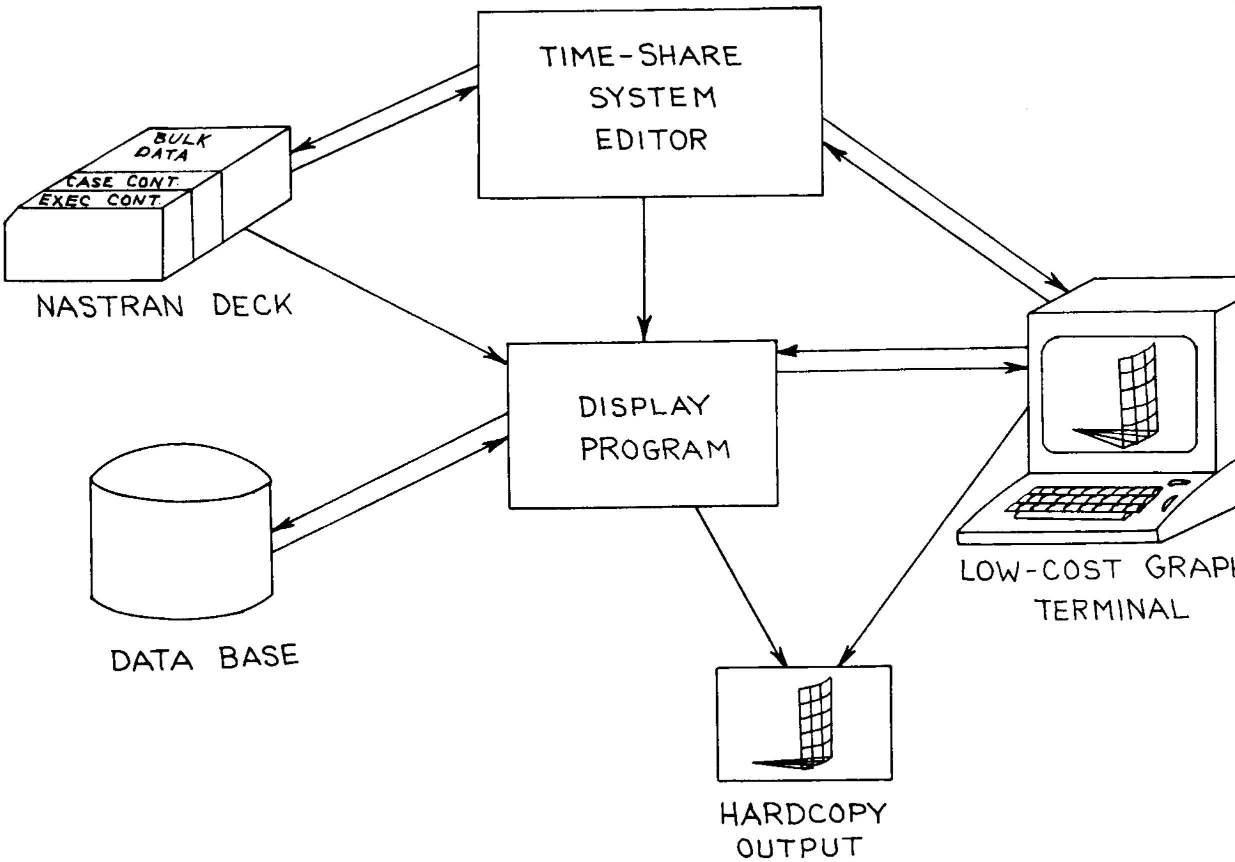

Figure 2.- Input Display Program Design.

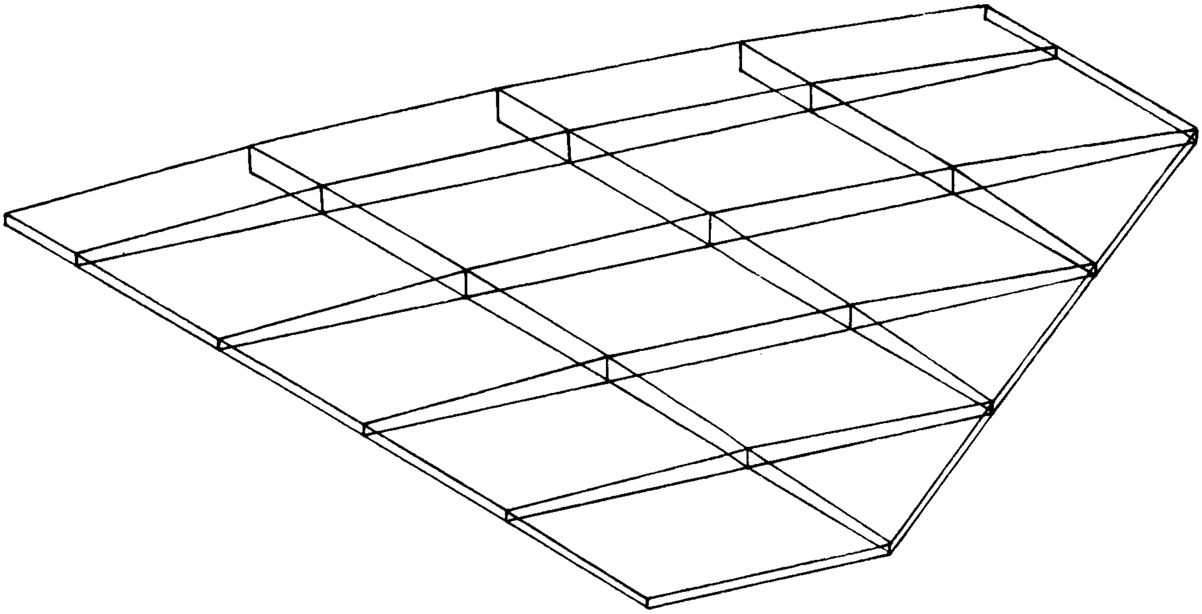

Figure 3.- Demo 1-1 Delta Wing.

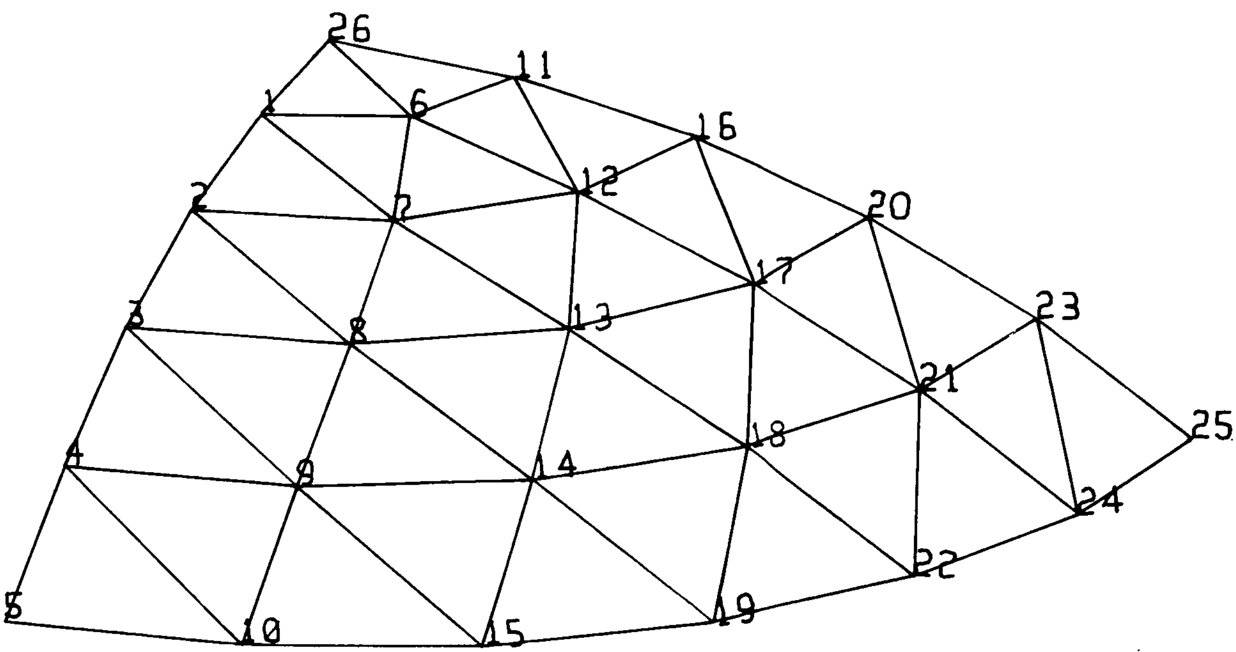

Figure 4.- Demo 1-2 with Grid ID's.

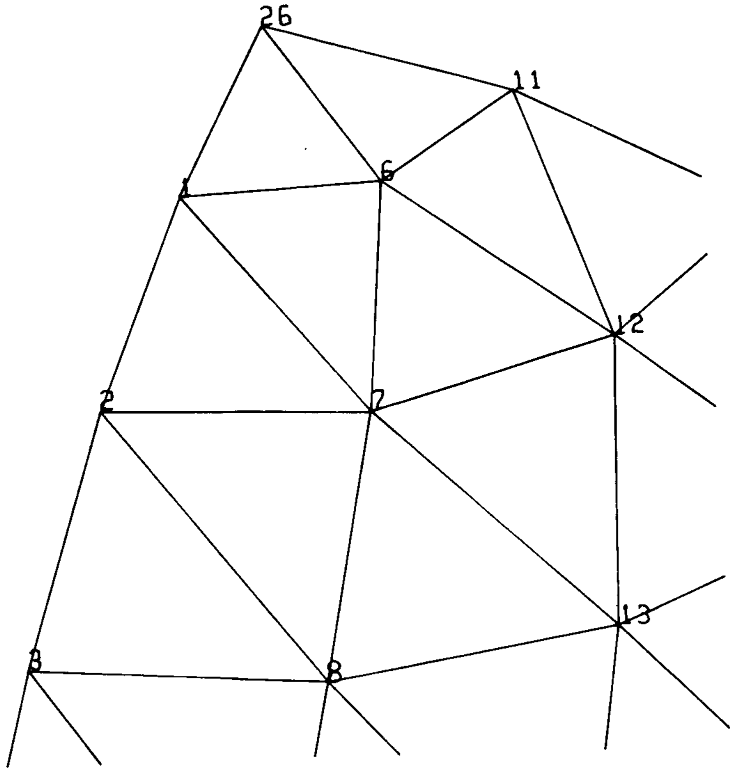

Figure 5.- Demo 1-2 Blowup.

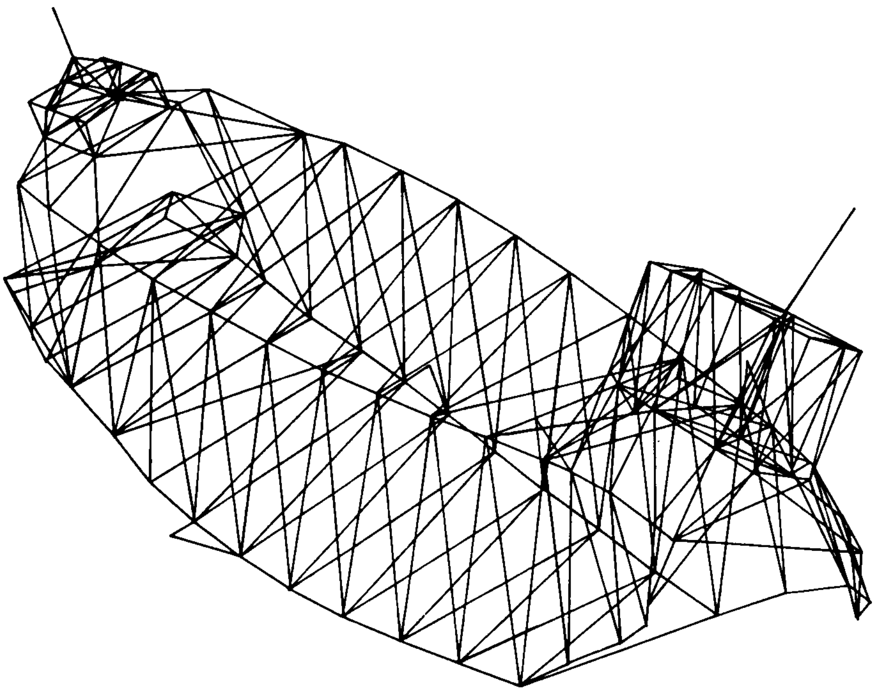

Figure 6.- Ninth Vibration Mode of a Helicopter Fuselage.