Lewis/Grant

IN-61

60408

P-188

COLOR POSTPROCESSING FOR THREE-DIMENSIONAL FINITE ELEMENT MESH QUALITY EVALUATION AND EVOLVING GRAPHICAL WORKSTATIONS

by

Malcolm J. Panthaki

ORIGINAL CONTAINS COLOR ILLUSTRATIONS

Report to NASA Lewis Research Center

Grant Number NAG 3-395

(NASA-CR-180215) COLOR POSTPROCESSING FOR 3-DIMENSIONAL FINITE ELEMENT MESH QUALITY EVALUATION AND EVOLVING GRAPHICAL WORKSTATION (Cornell Univ.) 188 p CSCL 09B N87-18997 Unclas G3/61 43650

Program of Computer Graphics

Report Number 87-1

Cornell University

Ithaca, New York 14853

Research supervised by

John F. Abel, Donald P. Greenberg,

Anthony R. Ingraffea, and S. Mukherjee

March 1987

COLOR POSTPROCESSING FOR THREE-DIMENSIONAL FINITE ELEMENT MESH QUALITY EVALUATION AND EVOLVING GRAPHICAL WORKSTATIONS

by

Malcolm J. Panthaki

Report to NASA Lewis Research Center

Grant Number NAG 3-395

Program of Computer Graphics

Report Number 87-1

Cornell University

Ithaca, New York 14853

Research supervised by

John F. Abel, Donald P. Greenberg,

Anthony R. Ingraffea, and S. Mukherjee

March 1987

ABSTRACT

This research represents the third and final phase of a project on general-purpose, interactive color graphics postprocessing for three-dimensional computational mechanics. Three general tasks are accomplished. First, the existing program (POSTPRO3D) is ported to a high-resolution device. In the course of this transfer, numerous enhancements are implemented in the program. The performance of the new hardware is evaluated from the point of view of engineering postprocessing, and the characteristics of future hardware are discussed.

Second, interactive graphical tools are implemented to facilitate qualitative mesh evaluation from a single analysis. The literature is surveyed and a bibliography compiled in this area of research. Qualitative mesh sensors are examined, and the use of two-dimensional plots of unaveraged responses on the surface of three-dimensional continua is emphasized in an interactive color raster graphics environment.

Finally, a new postprocessing environment is designed for state-of-the-art workstation technology. Modularity, personalization of the environment, integration of the engineering design processes, and the development and use of high-level graphics tools are some of the features of the intended environment.

ACKNOWLEDGEMENTS

This report is essentially a reproduction of the thesis submitted by the author in partial fulfillment for the Degree of Master of Science. This research on postprocessing has been partially sponsored by the National Aeronautical and Space Administration Lewis Research Center under Grant NAG 3-395 entitled "Interactive Color Display of Three-Dimensional Engineering Analysis Results." Additional support has been obtained from the National Center for Earthquake Research (SUNY Buffalo) through Project No. 86-4011, and by Cornell's Program of Computer Graphics. The faculty investigators for the project are John F. Abel, Donald P. Greenberg, Anthony R. Ingraffea, and S. Mukherjee. The assistance of Jerome F. Hajjar, Bruce C. Bailey and Paul A. Wawrzynek are gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES*

---

* A portion of the figures in this thesis are reductions of computer generated images and color photographs of images from a raster screen. Some of these figures have been reduced to the degree that some of the lettering is not legible. In each of these cases, the small lettering contains no pertinent information.

CHAPTER 1

INTRODUCTION

The advance of powerful digital computers has drastically increased the complexity of problems that the engineer can analyse. With the increased complexity of analyses, postprocessing has become a necessity. In recent years, fast color graphics that is not prohibitively expensive is beginning to make interactive color three-dimensional postprocessing feasible. Graphical postprocessing facilitates a better understanding of the behavior of the structure besides allowing the analyst easy access to relevant numerical information.

The effort to close the preprocessing-analysis-postprocessing design loop has inspired much research in the area of automatic mesh refinement. This has led to the introduction of a new postprocessing function - mesh quality evaluation. Numerical analyses are approximate solutions to a real problem. Properly formulated numerical solutions have been shown to approach asymptotically the "exact" ones as discretization is indefinitely increased. There is a tradeoff between cost-effectiveness and the degree of error. Thus, an important postprocessing function is to indicate to the analyst the quality of the analysis being examined.

At the start of this research, the capacity to view multiple-material, stepwise nonlinear or multiple loadcase analyses and to create subobjects existed on a Lexidata System 3400 frame buffer. The resolution of this device is 512 x 640. The frame buffer supported a 12-bit in, 8-bit out, pseudo-color, one-channel color look-up table with a 16-bit depth buffer and 4 white, hardwired overlays. Rendered polygons are automatically depth-buffered at scan conversion and the capability to depth buffer three-dimensional vectors drawn in an overlay plane allowed interesting use of grids of lines in the subobject extraction process. The analyst was able to choose one of six color maps and interactively change the scale of the map from linear to nonlinear or to specify upper and lower limits. A single light source existed and could be interactively "moved" to illuminate the object as the analyst required. The lack of sufficient speed in the rendering of boundary polygons necessitated the use of outlines during the rotation of the structure [1].

This thesis describes the third year of development of three-dimensional color postprocessing techniques. The crux of this research lies in the enhancement of the existing environment. The interactive postprocessing implemented to date has been used for the evaluation of finite difference, finite element and boundary integral equation results. Emphasis during the third year of research has been on the implementation of finite element mesh quality evaluation and the enhancement of the existing environment, partially through the exploitation of high-resolution, fast raster graphics displays. The availability of more cost effective and faster computing facilities has made possible the numerical solution of increasingly larger problems. This has led to the need to modify and improve the postprocessing capabilities to permit the evaluation of the results of these large analyses. Problems incurred during the development of these new capabilities and during the port of the computer program, as well as new directions, in both sofware and hardware, are discussed. Based on the experience gained from over three years of research in this area, a new CAD environment has been designed. This environment exploits the advanced features of state-of-the-art, high-level raster graphics workstation technology.

1.1 Background

The main role of any postprocessing system is to provide the user with an interactive graphical environment that facilitates the evaluation of behavioral responses. The interpretation of these responses graphically will be referred to in this thesis as "response viewing." This is an essential tool in the design process and allows the engineer to gauge the behavior of the design quickly and interactively. User interactivity and flexibility are the features stressed during the design of the program. Tablet-driven on-screen menus are used as the user-interface. No feature that would take more than a few seconds to be implemented is provided to the user. This, in certain cases, limits the funtionality available to the user and necessitates the use of state-of-the-art raster graphics workstations. These are high resolution devices with high-powered geometry engines for shading, depth-buffering, transforming and, finally, rendering polygonal information in real time.

A relatively new field of research is that of finite element mesh quality evaluation and "error" computation. This research is inspired by the drive to create self-adaptive design programs with automatic mesh refinement capabilities. The need to refine the mesh is expressed in the form of qualitative or quantitative error or mesh quality sensors. Qualitative error sensors provide the user with initial insight into the accuracy of the solution in localised regions of the mesh and, hence, the need to remesh locally.

One of the important objectives of postprocessors is to provide the user with the ability to obtain an initial "feel" for the quality of the analysis under review. When an engineer has performed the approximate numerical analysis of a complex system, the first questions that he should consider are, "How accurate is the approximation?" and "Is there a need to re-analyze the problem with a refined model?" Ideally, one would prefer to utilize a computer program which automatically begins to answer these questions by, say, self-adaptive refinement, [2-4]. In addition, if a sequence of analyses with successfully refined meshes has been performed, it is possible to estimate errors quantitatively and to apply extrapolation techniques to obtain improved approximations, [5-7]. However, most analysis systems now used in practice do not include capabilities for such advanced concepts as self-adaptation, error-estimation, or extrapolation. Instead, the analyst attempts to develop an adequate analytical model and performs an initial analysis. Answering the above questions for the initial and subsequent analyses then becomes a question of judgment tempered by tradeoffs between feasible analytical effort and the accuracy desired. The assessment of accuracy is often subjective. In the worst extreme, those who misuse numerical methods neglect to perform even subjective assessments. This could be attributed primarily to naivety but also to the lack of effective tools to facilitate critical evaluation.

To facilitate answering the result-interpretive questions posed by the analyst -- "What is the behavior of the analysed system?" and "What are the optimal results to be drawn from the numerical analysis?" -- smoothing of response parameters is necessary. If the questions regarding accuracy posed above are as important as the result-interpretive questions of postprocessing, the engineer who uses typical postprocessors may face a dilemma as the tools available may not adequately permit him to answer both sets of questions. In particular, the smoothing of computed responses may mask the information needed to assess the quality of the analysis. On the other hand, standard graphical interpretation tools can be used to convey some types of analysis accuracy information if available; e.g., if continuous quantitative error estimates are available, the usual contouring methods can be used to display them.

1.2 Objectives

Numerical analyses typically create volumes of numerical results that have to be sifted by the analyst. This data contains, embedded within it, the answers to both sets of questions posed above. Typically, smoothing techniques would result in the loss of the analysis-accuracy information. Thus, it is necessary to store the unsmoothed results as well.

The main objective of this research is to create an environment within the existing postprocessor that would enable the analyst to examine the results of a single analysis from the perspective of mesh quality evaluation and error sensing. The tools developed would enable reaching an initial decision on the "quality" of the analysis. They would also indicate the need to perform a more refined discretization of local subdomains of the structure. The thrust is toward evaluation of a single analysis and is qualitative in nature. Research is on-going in the area of quantitative error measures, and developments to date will be briefly discussed in Chapter 4 of this thesis.

Also included in the current research is the enhancement of the existing graphical, color postprocessing environment by improving the performance of existing functionality, adding new features, and by porting the program to a high-resolution device.

Finally, the experience gained in the process of the port is consolidated in the form of suggestions and the design of an integrated CAD environment for workstations.

There are a number of specific objectives which this research aims to address:

(1) Mesh quality evaluation.

(2) Enhancement of the existing postprocessing functionality.

(3) Port of the software to a high-resolution device.

(4) Design of a highly modular, new CAD environment for workstations.

(5) Increasing the modular nature of the code and using, as much as possible, a device-independent graphics package that would facilitate portability and future modification of the code during research in this area.

(6) The expansion of the operating database to include unaveraged responses at each of the nodes on the boundary of the structure.

(7) Enhancing the translation facilities for the postprocessor.

1.3 Scope

A large number of minor refinements have been implemented in the existing system, and these are too numerous to list here. The major enhancements are listed and described in this thesis.

A new feature of the program allows the user to plot responses along a line on the surface of the structure. Unaveraged responses at the boundary nodes are used to extract response values along the plot line. The presence or absence of inter-element discontinuities is used as a qualitative mesh quality sensing device. The user can define the plot line interactively and can also create a file containing the plot information. Smoothed responses, too, can be graphed on the screen. The high resolution obviates the need for anti-aliasing vectors in most cases.

The capacity of the postprocessor to handle large problems has been increased. It has been set up to handle problems of up to 2,500 elements, 15,000 nodes and 4,000 boundary faces. Creation of subobjects by material type has been made visual with the use of color maps to render the structure. Unaveraged boundary response information is now stored in the database to facilitate mesh quality evaluation. The ability to generate this information for subobjects has been implemented. The program now has the capability to add annotation to the main display window on the raster scope before storing the image for hardcopy creation.

The translator programs have been improved. The extrapolation technique used to obtain nodal response values for 15-noded isoparametric (3-D wedge) finite elements was inadequate in regions of high stress gradients. A standard least-squares technique using the shape functions of the linear wedge element as an interpolant is now employed to extrapolate gauss point responses to the nodes. A new translator program to salvage the linear elastic information from the first linear step of an erroneous nonlinear analysis has also been written.

The code has been made more modular. Low-level graphics calls have been isolated. In approximately half the cases, a device-independent package has been used. In those cases where the package does not support the required functionality, calls have been made to the graphics library supported by the Rastertech frame buffer -- Onelib. These calls have been isolated in a layer of code that can easily be modified in the future, if necessary. The use of software overlays and software color maps, although slowing down some of the functionality, are features that improve portability of the program. The move from the one-channel Lexidata system to the three-channel Rastertech system necessitated many of these changes in the program design.

The accessing and allocation of virtual memory during run-time becomes a natural problem area as the size of the program and as the size of problems evaluated by the program increase. The manner in which memory is acquired has been streamlined as far as possible in the postprocessor, as well as in the boundary and outline extractor programs. As the program grows larger, this is a problem that will need to be tackled from a different perspective, most appropriately through the re-design of the database.

A new CAD environment has been designed. This design is being implemented at the Program of Computer Graphics, Cornell University. High-level graphics tools are used in the design. A highly modular approach simplifies maintainance and the addition of graphical functionality. Device and database independence, and the creation of a personalized postprocessing environment are some of the main goals of the design.

1.4 Organization of the Thesis

Chapter 2 discusses the new hardware and software environments created for the postprocessor during the third year of research. The necessity for a hardware change is stressed, and the performance of the new frame buffer is evaluated. The algorithms that have been re-designed due to the hardware change are described. Future hardware trends are proposed. The drive to create a modular, portable and flexible program is discussed.

Chapter 3 enumerates the enhancements made to the existing postprocessing environment. Material extraction (the process of interactively creating a "subobject" based on material types) has been made more visual. The existing database has been expanded to include software color maps, and unaveraged boundary response information. Additional response parameters can now be viewed. The response smoothing technique for 15-noded 3-D wedge isoparametric finite elements has been improved. A new Annotation menu page has been added -- this allows the analyst to add annotation like text, circles, arrows, etc., to an image.

Chapter 4 discusses the main drive of this research - mesh quality evaluation. A literature search is documented and the use of inter-element response discontinuities as a qualitative mesh quality sensor is demonstrated. 2-D plotting of unaveraged response parameters is used as a qualitati\e mesh sensor. Algorithms used to develop interactive plot loop and plot line definition on the surface are described. Example problems are used to demonstrate the efficacy of the suggested measures.

Chapter 5 details the principles of the design of a new postprocessing environment. The advantages of the modular design are discused in detail. Suggestions are made for the integration of the design process, and the improvement of the database structures used.

Chapter 6 summarizes this research to date and suggests future research areas and trends in the field of color graphical postprocessing.

Appendix A outlines the file management scheme used by the preprocessing, analysis, and postprocessing programs in use at the Program of Computer Graphics in Cornell University. Finite element and boundary element applications are treated separately. The contents of relevant data files generated by these programs are briefly listed.

Appendix B deals with the memory problems associated with the size of the analyses evaluated by the program. The mainframe-host environment is compared to the high-powered graphics workstation environment. Suggestions for efficient memory management are made.

CHAPTER 2

NEW COMPUTING ENVIRONMENTS

An inevitable feature of any research environment is constant evolution. This feature is especially apparent in areas that involve the use of computers and computer graphics. There is an almost continuous improvement of the cost/performance ratio due to rapidly improving technology. Concomittantly, software must evolve with the hardware. The new features of the graphics hardware -- improved resolution, faster polygon rendering speeds, hardware transformations and lighting calculations etc. -- need to be exploited by corresponding changes in the software. Fortunately, as more graphics functionality is absorbed into hardware, the software which uses these functions becomes simpler.

This chapter deals with the evolving computing environment, both hardware and software, and the accompanying changes to the existing postprocessing environment. The performance of the new hardware is analysed and compared to that of the existing hardware.

2.1 Hardware and Firmware

The Lexidata System 3400 represents early 1980's technology while the Rastertech Model One/380 represents the technology of 1984. This sections compares and contrasts two technologies in a "changing-technology" situation with respect to engineering postprocessing as viewed by the author and his colleagues.

The initial hardware consists of a System 3400 color raster display monitor with a 12-bit frame buffer, a 12-bit depth buffer, and 4 hardwired overlay planes, and is produced by Lexidata Corporation. The screen resolution is 640 x 512. The frame buffer is driven by a host VAX 11/780 computer on a time-sharing basis. There are a number of features of this hardware that make it suitable for engineering postprocessing. The single-channel, pseudo-color look-up table is ideal for the simple 3-D shaded images that are required. Three-dimensionality of viewing is maintained by simple cosine shading that is done in software. Hardwired overlays with a higher color intensity prove to be visible over any contoured images and are fast. Depth buffering of 3-D vectors drawn in any of the overlays is a useful feature in the subobject extraction process and when displaying the displaced shape [1].

Figure 2.1 shows the hardware configuration and Local Area Networks in the Program of Computer Graphics in Cornell University. The Rastertech model One/380 frame buffers and the Hewlett-Packard Series 9000/320 workstations are the available raster graphics devices.

2.1.1 Necessity for a Hardware Change

Although the Lexidata monitor and frame buffer were suitable in many ways, the 640 x 512 resolution was not sufficient for a number of features that were to be added to the program. The quality of the image, especially for lines and curves, leaves much to be desired. To facilitate rendering speed, a simple cosine diffuse shading law is incorporated. This requires higher resolution if clear and sharp images are desired. The low resolution of the Lexidata make the implementation of multiple viewports infeasible. 2-D plots of unaveraged reponse parameters is used as a qualitative measure of the performance of the mesh. This feature, too, requires higher screen resolution.

A present goal of computational analysis software is the real time simulation of complex analyses -- to be able to use an intelligent workstation as a viewing window into a transient or nonlinear solution of a problem. The attainment of this goal must be viewed from many angles. From the perspective of the viewing mechanism, the bottlenecks usually lie with rendering capabilities of the frame buffer. Hardware transformations, shading, hidden surface removal and scan-line conversion have improved the rendering capabilities of modern raster graphics workstations. The speed of the Lexidata permits the rotation of only the wire-frame representation. Faster polygon rendering speeds would allow real time rotation of the polygonal representation of the structural model. As a step in this direction, it was decided to transfer the postprocessing software to a newer, high-resolution device with faster rendering capabilities.

The hardware change would enhance the quality of the image and improve rendering capabilities. This change would also serve as a test-bed for the next phase -- the complete redesign of postprocessing software on the H.P. Series 9000/320 workstations.

2.1.2 The Rastertech Model One/380

The Rastertech Model One/380 is based on the latest technology of the time of its release. This system consists of a high resolution 1280 x 1024 color monitor and a 40-bit frame buffer. The frame buffer is configured as a 3-channel, real-color system with 8 bits of information for each of the three red, green, and blue channels. It also contains a 16-bit depth buffer used for hidden surface removal during scan-line conversion. Frame buffer commands are buffered and executed when either the buffer is filled or a "buffer-dump" is generated.

The display list can be segmented and has the capacity to store one megabyte of information. This feature of the new hardware could not be taken advantage of as the display list of the larger problems needed far more than the available one megabyte of storage space. Thus the hardware shading and transformations, which could only be performed on the display list, were not used and had to be done in software.

2.1.3 Performance of the New Hardware

In general, the performance of the new hardware with respect to the particular engineering postprocessing features that have been found useful in this and earlier research is disappointing. Besides performance inadequacies, frequent hardware problems occurred. The following section compares the new Rastertech frame buffer (RT) with the Lexidata (LD) frame buffer from the point of view of engineering postprocessing.

1) 1-Channel vs 3-Channel Color Graphics Systems: The LD has a 1-channel, pseudo-color look-up table whereas the RT has a 3-channel,

real color system. If only 6 color maps, each of 10 base colors, are desired, and if the maximum of 256 shades of each of the base colors of the currently used map are to be stored, then it can be seen that the 4096 colors that could be simultaneously displayed on the LD monitor are more than adequate. The 1-channel system allowed for rapid execution of such useful features as "changing color scales" or "contrasting" [BCB]. An informal survey of program users showed that these features are constantly used. On the LD, these were performed almost instantaneously by reloading the color look-up tables. The 3-channel system on the RT does not allow the rapid execution of these features. Both features are now much slower and the algorithms involved were much more complex (Section 2.2.3 describes the new algorithms). Large numbers of polygons are now redrawn, whereas in the 1-Channel system, the color look-up tables were merely reloaded to accomplish the same result. Another, less important feature that has been affected is the ability to change the orientation of the "light source". The user could interactively "move" the single light source around to illuminate the structure suitably. This feature has now been removed from the RT version as it is no longer interactive.

2) Hard-wired overlays: Most interactive graphics, engineering programs make use of overlays. More specifically, the existing postprocessor makes use of overlays for many essential functions. Often, use is made of two and sometimes three overlays at the same time. The LD has four hard-wired overlays. The RT system has none. Software overlays are used in the RT version of the program.

The important features that make use of overlays are [1,8]:

a) Display of tables.

b) Movement of "grids" used in the element extraction process for the creation of subobjects.

c) Display of displaced shapes in the form of deformed meshes and undeformed meshes.

d) Display of element and node numbers in the Attribute Viewing Page.

e) The new Line-Plotting feature -- to draw plot loops and lines on the surface of the structure, display meshes and the actual plot itself.

Often, many of the above features are needed simultaneously; hence the need for multiple overlays that can be individually written into or cleared.

The advantages of hard-wired overlays over "software overlays" are many and it is the belief of the author that hard-wired overlays are a necessity for an efficient engineering postprocessing environment for the following reasons:

a) Hard-wired overlays operate faster.

b) The overlays in the LD have 125.0% intensity levels and are, hence, bright. This is necessary as a contoured image is often 'overlayed' by a mesh, which has to be brighter than the background to be visible. The RT version uses software overlays and these are dim in most cases (Section 2.2.1.2). Often it is found necessary to have to "thicken" vectors that are drawn in a soft overlay to make them more visible. This further slows down the process. Also, from an applications programmer's point of view, it is a lot easier to develop computer code for a frame buffer that has hard-wired overlays built into the system. The software overlays of the RT system necessitated the use of the device independent RAG package developed at the Program of Computer Graphics in Cornell University. This forms an additional "layer" of routines and was found to reduce the speed of execution of the program.

3) Phantom-Write Feature of the Lexidata: This is a useful feature that is built into the LD graphics library. When switched on, it allows the automatic depth-buffering of vectors drawn into an overlay. This ensures that those parts of a vector that lie "behind" existing polygons, (in the image plane), are not drawn. It is found to be a useful feature when grids drawn in an overlay are "passed through" the structure interactively. The intersection of the grid plane with the structure then becomes obvious to the user who can position the it appropriately with respect to the structure. This feature is used extensively in the process of subobject creation, an important function of the existing postprocessor [1]. The RT does not allow vectors to be depth-buffered. Thus, the process of visually positioning a grid of lines with respect to the structure is now difficult.

4) Speed of Rendering Polygons: The RT has 4 times the number of pixels when compared to the LD, but its published speed of rendering led the author to believe that a better performance would be obtained. This was not found to be the case. The RT frame buffer is faster than the LD frame buffer by approximately 4 times; but as there are 4 times as many pixels, there is no gain in rendering speed.

5) Reading Depth Values out of the Depth-buffer: The LD allowed the program to read screen depth values out of the depth buffer for a given pixel location. This useful feature made the important "hit-testing" function relatively fast and simple [8]. The fact that this is not

possible using the RT makes hit-testing more complex, algorithmically. Most frame buffers manufactured today do not allow the access of depth information partly because "scratch" space is used to store this information and its retrieval is slow. Thus, the new algorithm conforms to present workstation and frame buffer standards.

6) Screen Resolution: The resolution of the LD system is 640 x 512 and that of the RT system is 1280 X 1024. This added resolution makes the images on the RT look significantly better and sharper. Also, the 2-D line-plotting feature that has been added to the new RT version, (described in detail in Chapter 4 of this thesis), could not have been effectively implemented using the lower resolution. The vectors used to plot the responses would have been far too jagged, and this defeats the purpose of the line plots which is to study inter-element response discontinuities. The future implementation of "multiple viewports" requires the higher resolution of the RT monitor.

Summarizing, the overriding disadvantage of the new RT system seems to be the fact that it is unreliable and is prone to breakdown without warning. The higher resolution is the sole feature that redeems this system. The loss of some of the postprocessing functionality mentioned above and the lower rate of performance of others are direct consequences of the characteristics of the system. The RT frame buffer seems to have been designed for static, realistic-image synthesis and not for the more dynamic, and interactive, engineering functions of the postprocessor. The move to this system is to be viewed as a valuable exercise in preparation for the more important complete redesign of the postprocessor on the H.P. Series 9000/320 workstations.

2.1.4 Future Considerations

Present trends indicate that the rendering speed of raster devices will keep increasing for a while. In the forseeable future, certain postprocessing functionality that now hinders interactivity and is nevertheless essential will be performed almost instantaneously. The next step is real time rotation of the boundary polygonal representation of the geometric model. Contouring is a function that is not only computationally intensive, but requires rapid rendering capabilities. A networked environment of workstations and high-speed super-computers may be the solution to this problem.

Real time simulation of transient-dynamic, and nonlinear problems require supercomputing power and super-workstation rendering capabilities for the front-end user interface. This would necessitate changes in hardware design, networking, and the use of asynchronous program environments.

2.1.4.1 Latest Technology

The new H.P. Series 9000/320 systems represent the state-of-the-art in raster graphics workstation technology. These new systems improve rendering capability by orders of magnitude. The display list is stored in virtual memory and hence its size is governed by the virtual memory capacity of the workstation. Segmentation of the display list is a feature that allows editing and manipulation of specific parts of the image display list. 3-D transformations, and simple diffuse and specular shading are done in hardware. The resolution of the monitors is 1280 x 1024. The frame buffer is a 3-channel, real-color system which can be configured as a 24-bit single buffer or 2 12-bit image planes for double buffering. A 16-bit depth buffer is used for hidden surface removal.

One of the important features of this machine is the existence of a hardware "transform engine" to drive the 3-D transformations of the display list. This speeds up polygonal rendering. Rotation of complex polygonal environments is now possible using double buffering to smooth out the process. Some of the disadvantages of a 1-channel system are negated with higher rendering speeds. Contrasting of base colors would now be fast even if a number of the polygons have to be redrawn.

As shading calculations are done in hardware, it will be possible to allow the user to interactively re-orient one or more light sources to illuminate the object as required. The "worst-case-scenario" of the process of changing the color map is redrawing all the boundary polygons. As this is now an interactive process for complex environments, changing the light source, too, could conceivably be as fast as it was on the 1-channel Lexidata frame buffer.

Dithering is a relatively new concept that is used to artificially improve the quality of an image with a small number of allowable pixel values. It deceives the eye into perceiving a much larger number of shades than actually exist by a process of statistical averaging over an area. This would allow the programmer to configure the 24 bits of image memory as a dithered, 12-bit double buffer with 4-bits of Red, Green and Blue image information per pixel per buffer, giving all the advantages of a 1-channel system with minimal sacrifice of image quality.

As the problems being solved increase in size and complexity, the rendering capabilities of the workstation, too, must improve. It is the belief of the author that the basic configuration of the Lexidata frame buffer, with a few modifications, is ideal for engineering postprocessing -- a 1-channel, pseudo-color look-up table with hardwired overlays. This can be substituted by the use of dithering in a 3-channel system.

2.1.4.1 Expected Evolution

Future trends will use the intelligent workstation as a window into a real time simulation of a transient problem with, possibly, changing topology -- for example, crack propogation through a continuum. Changing topologies introduce other possible bottlenecks like boundary extraction. An efficient database and database management system ought to minimise this problem by recognizing the geometrically localized nature of the changing topology.

Analysis would be a parallel process run either on a supercomputer or on a network of main frames [9]. Memory for the storage of data would become a concern as the size of problems increases exponentially. Common shared databases would minimise storage requirements and maximise the efficiency of accessing the information. Display would be handled by the workstation using Local Area Networks (LANs) to access a common database. All transformations, hidden surface removal, clipping, shading calculations and scan-line conversion will be done in hardware.

Many hard lessons have been learned in the last ten years of computer graphics software development. An obvious problem is the lack of high level "tools" that almost every graphics applications programmer needs. Every programmer spends a disproportionate amount of time reinventing countless (software) wheels -- windowing, menu managing, perspective and other transformations etc. These functions ought to be offered to the programmer as system services to be used when required. This will ease the load of much redundant programming and allow the structural engineer to concentrate more on engineering research than on graphics applications programming [10].

2.2 Software Design

Evolving hardware necessitates changes in software design philosophy. Basic algorithms change and the increased rendering speeds improve postprocessing functionality offered to the analyst. The range of this functionality varies not only as the capabilities of the device, but also inversely as the size of the problem being displayed. Certain features that are interactive for small problems may not be for larger problems. The postprocessing environment must be intelligent enough to make this decision. For example, the contrasting feature loses much of its importance for very large problems. As postprocessing of these problems becomes unwieldy, the aim is primarily to get a first, quick "feel" for the behavior of the structure and to identify critical design regions. A more thorough examination of the structure is done by extracting a subobject containing these critical regions.

2.2.1 Flexibility and Portability

There are numerous problems associated with creating large graphics programs that are portable. In a research environment, hardware and software requirements change continuously. It is almost impossible to fully anticipate the software requirements of such an evolving environment. Thus, it is important to design flexible and modular code that can be easily modified when necessary. Postprocessing all types of engineering analyses using a single program is a difficult task. Changes in hardware and its capabilities make this task more difficult. It requires proper design, careful planning at the initial stages and a code-development philosophy that stresses modularity.

2.2.1.1 Modularity of Graphics Code

Flexibility is a feature of graphics code desired by both users and future researchers who plan to modify it for their own specific uses. A modular design allows the "hooks" to be installed for the addition of functionality. This design philosophy also allows more than one researcher to work on the overall task of creating such a postprocessing environment. As a postprocessor ought to serve as a viewing front-end for all kinds of research-type engineering analyses, it is important to build into it the capability to process any database. The researcher would design a database to suit his own purposes and use the existing framework to view the reults of analyses -- communication is through a set of database modifying and querying routines. Making this task as easy as possible will save research time in the future.

An attempt has been made to further modularize the postprocessor. The changes made are too numerous to list here. Low level graphics calls have been seperated from the main body of the applications code. Redundant segments of code have been converted into subroutine calls. An important feature of modular programming is the proper use of program parameter values. Overlays are referenced symbolically according to the task they perform -- for example, the use of variables like MESH_OVERLAY allow the programmer to easily experiment with the visual impact of changing the overlay in which the mesh is displayed. This also creates the programming "hooks" for software that allows the analyst to select overlays for the display of specific high-level graphics objects like the mesh, outline, displaced mesh, 2-D plots, etc.

2.2.1.2 Software Overlays

One of the changes necessitated by the move to the Rastertech system was the use of "software overlays." The RT does not contain any hardwired overlays. A software overlay is code that controls the I/O into the desired bit plane of the image memory that searves as an overlay. Low level READ_ENABLE, WRITE_ENABLE, and DISABLE calls are used.

The Rastertech image memory consists of 3 8-bit planes for Red, Green and Blue video gun intensities. For the implementation of software overlays on this device, the 8th bit in each of the 3 channels is used, enabling 3 overlays -- a red, a green and a blue. As these are not hardwired, speed is reduced. Also, the intensity of the colors is the same as that of the image plane. Thus if the red overlay is used to display an object over a red image, it is not visible. The intensity of the hardwired overlays on the Lexidata and the new H.P. Series 9000/320 systems is raised by 10-25% over that of the image plane and ensures, for example, that a white overlay is visible over a white object in the image plane.

As the code controlling the overlays has been modularized, a change to a system with hardwired overlays is simple. Only the device dependent low-level calls will need to be modified.

2.2.1.3 Use of a Device Independent Graphics Package

It is essential to install a layer of device-independent graphics code between the applications code and the device-dependent code. This improves the portability of the program.

The program has been modified to make use of an in-house, device-independent graphics package called RAG, (**RA**ster Graphics). This package allows device-independent programming of frame buffers and includes a subset of operations common to most of them. Soft (virtual) frame buffers are supported as data holding devices. Display-type functions (cursor, zoom, overlay_on) can be called, but will have no effect on the soft frame buffers. High level graphic data routines provide for the writing of certain graphics primitives including pixels, vectors, rectangles, and polygons. Polygons and vectors may be antialiased. All pertinent configuration information is kept in a descriptor whose address is used as a handle. First the descriptor is allocated, then filled with pertinent information depending on the specific frame buffer and the configuration desired. Then the frame buffer is initialized. Once the frame buffer has been allocated, all attribute setting operations are invalid. It is necessary to deallocate it first, then change the attributes before re-allocation. This allows the applications programmer to configure the frame buffer as required using high level graphics calls. Cursor, zoom, and overlay routines are provided by RAG.

Use is made of RAG's capabilities to set up software overlays. This marginally reduces the portability of the program. If moved to another environment, there will need to be a device with hardwired overlays, or a similar package supporting high level calls to control soft overlays.

2.2.2 Translator Programs for the Postprocessor

Changes are made to the existing finite element translator program. The technique used for response smoothing in 15-noded wedge isoparametric elements is changed (Section 3.4 discusses the new algorithm). The existing database stored only the smoothed (averaged) response values at the nodes in the structure. Now, unaveraged response information at the nodes on the boundary of the structure are stored in the database. A file containing the Gauss point responses for all the elements is also created in the translator and used if a subobject is created. This is detailed in chapter 4.

2.2.2.1 Suggestions for Future Translator Programs

The existing finite element translator program is an example of "patchwork" code. As the database of the postprocessor grew in size and complexity, the translator program began to reflect these changes. This kind of interface design, (with the developer of the postprocessor also writing the translator programs), has a number of problems associated with it. The most crippling is that it restricts the analysis researcher to the format that the "current version" of the postprocessor requires. Smoothing of discontinuous responses is performed during the translation phase and the format of the data used is hardcoded. Various researchers might require one or all of a number of smoothing schemes to be implemented and compared. It has also been found that the introduction of new response types is a tedious process.

Thus, any changes made to the databases of either the analysis programs or the postprocessor itself requires changes to be made in the translator programs. This reduces flexibility and forces a certain format on both the postprocessor and the analysis code. It is necessary to allow researchers to design their own databases and, hence, develop their own translator and database querying subroutines. This procedure allows the researcher to build a postprocessing environment based on specific requirements -- one of the ultimate goals of a flexible, research-oriented postprocessing environment (chapter 5.)

2.2.3 Algorithms Redesigned Due to the Hardware Change

The port of the program from the Lexidata to the Rastertech system with its different hardware characteristics necessitated changes in a number of basic algorithms. A number of these are associated with color maps.

2.2.3.1 The Hit-testing Algorithm

This is a database querying algorithm whose purpose is to return to the program characteristics of the "hit" pixel like its color, the boundary polygon it belongs to, the "hit" element number, the closest node, a response value at a node, etc. The Lexidata frame buffer allows read and write access of the depth buffer. The previous algorithm made use of the depth information of the "hit" pixel. The Rastertech frame buffer does not allow read access to depth information. Thus, the algorithm had to be redesigned.

```
begin algorithm
        1) The X_SCREEN and Y_SCREEN "hit" pixel screen coordinates are
           computed from the returned tablet coordinates.
        2) The coordinates of each of the vertices of each boundary face
```

are transformed to screen coordinates using the total transformation matrix (TOT_TRANS_MAT).

3) An X and Y "Box Test" is done on all the boundary faces to check if the hit pixel lies within the box created using the maximum and minimum X and Y screen coordinates of each boundary face. A list of all such polygons is obtained. If only 1 polygon passes the Box Test go to step 6.
4) Cull this list to remove all back-facing polygons. If only 1 polygon passes this test go to step 6.
5) Pass the remaining polygons through the Surrounder Test to check to see if the pixel lies inside the polygon. (Note here that a polygon could pass the Box Test and fail this.)
6) Solve for the plane equations of the remaining polygons (note that these equations are in screen space).
7) Solve for the Z_SCREEN value of the hit pixel by substituting the X_SCREEN and Y_SCREEN values in each of these equations. If only 1 polygon has passed all the tests go to step 9.
8) Now select the minimum of the Z_SCREEN values computed in step 6 (i.e. the one closest to the viewer). Store the number of the "hit" boundary polygon.
9) X_SCREEN, Y_SCREEN and Z_SCREEN are transformed to X, Y and Z coordinates in untransformed object space by using the inverse of TOT_TRANS_MAT.

end algorithm

Now that the position of the "hit" pixel, and the boundary polygon it belongs to are known, database dependent calls are used to extract additional information as required. Although the new algorithm has more computations, there is no loss of interactivity. It is also more general to assume that the depth buffer cannot be "read-enabled", enhancing the portability of the program.

2.2.3.2 Contrasting Specific Base Colors

The function of this algorithm is to allow the user to change a specific base color to a dark grey, so that all the polygons drawn using this color can be differentiated on the contoured image of the structure. This is a useful feature, allowing the user to quickly gauge those parts of a structure that lie in a response range of interest. In a 1-channel system, this is easily and rapidly accomplished. Only those Color Look-Up Table (CLUT) values that correspond to this base color and its shades need be changed to grey and its corresponding shades. All polygons drawn using these as indices instantaneously change color. It must be noted that no polygons are redrawn. This simple procedure is not possible on a 3-channel system. Changing a section of the CLUT values would affect a random number of colors.

begin algorithm

1) While the response is being contoured, the boundary polygon information is stored. These are arranged according to the base color used for each. There are a maximum of 10 such sets (1 for each of the 10 base colors in the contour color map). Along with the screen coordinates of the vertices of these polygons, the number of vertex points, and the angle between the normal to each polygon and the unit vector representing the light source are stored.

2) When the user selects a base color to be contrasted, all the polygons stored under this base color, if any, are redrawn in grey (or in the specific base color if "uncontrasting" is desired.)

end algorithm

This algorithm is much slower than the one on the Lexidata. This is one of the problems of a 3-channel system and cannot be circumvented. Besides being slower, a large amount of memory is required to store the boundary polygons and their associated attributes. This procedure becomes impractical for very large problems. For example consider a problem with 4,000 boundary faces. Assume that 3,000 are front facing polygons and that the problem is analysed using 20-noded brick elements. Each face is first broken down into 8 triangles. Each of these triangles are broken down into smaller polygons, each of a single base color. Assume that on an average, each triangle gets broken down into 3 smaller polygons. This gives a total of 72,000 boundary polygons, each of a specific base color. The memory needed to store this information is approximately 5 megabytes.

The same procedure would be possible using segments for the display list. The reason this is not done on the Rastertech frame buffer is that there is a limit of one megabyte of memory for the display list. Larger problems needed far more than this. On the H.P 9000/320 systems, it is possible to use segments as the display list can be as large as the amount of virtual memory available on the machine, which is atleast 32 megabytes. In this case, the transformations and lighting computations are handled by the hardware and the process is speeded up tremendously.

2.2.3.3 Changing the Color Maps Used for Contouring

This feature is necessary as the visual perception of different users with respect to color differentiation varies. The existing postprocessor allowed the user to choose one of six available color maps.

If a different color map is chosen in the Lexidata version, the change is seen almost instantaneously as the only process is the reloading of 2560 values in the CLUT. No polygons are redrawn. For the reasons mentioned above, this is not possible on the 3-channel, Rastertech system. The complete structure needs to be redrawn. The algorithm is thé same as the one used to contrast specific base colors. In this case, the polygons of all base colors are redrawn using the corresponding new base color from the new color map. To reduce redundant rendering, a record is kept of those colors that have been contrasted. The polygons corresponding to these are not redrawn (as they would be the same color, grey, even for the new color map).

This process will benefit by the use of internal segments of the frame buffer. The H.P 9000/320 system will permit the use of this feature even for very large problems.

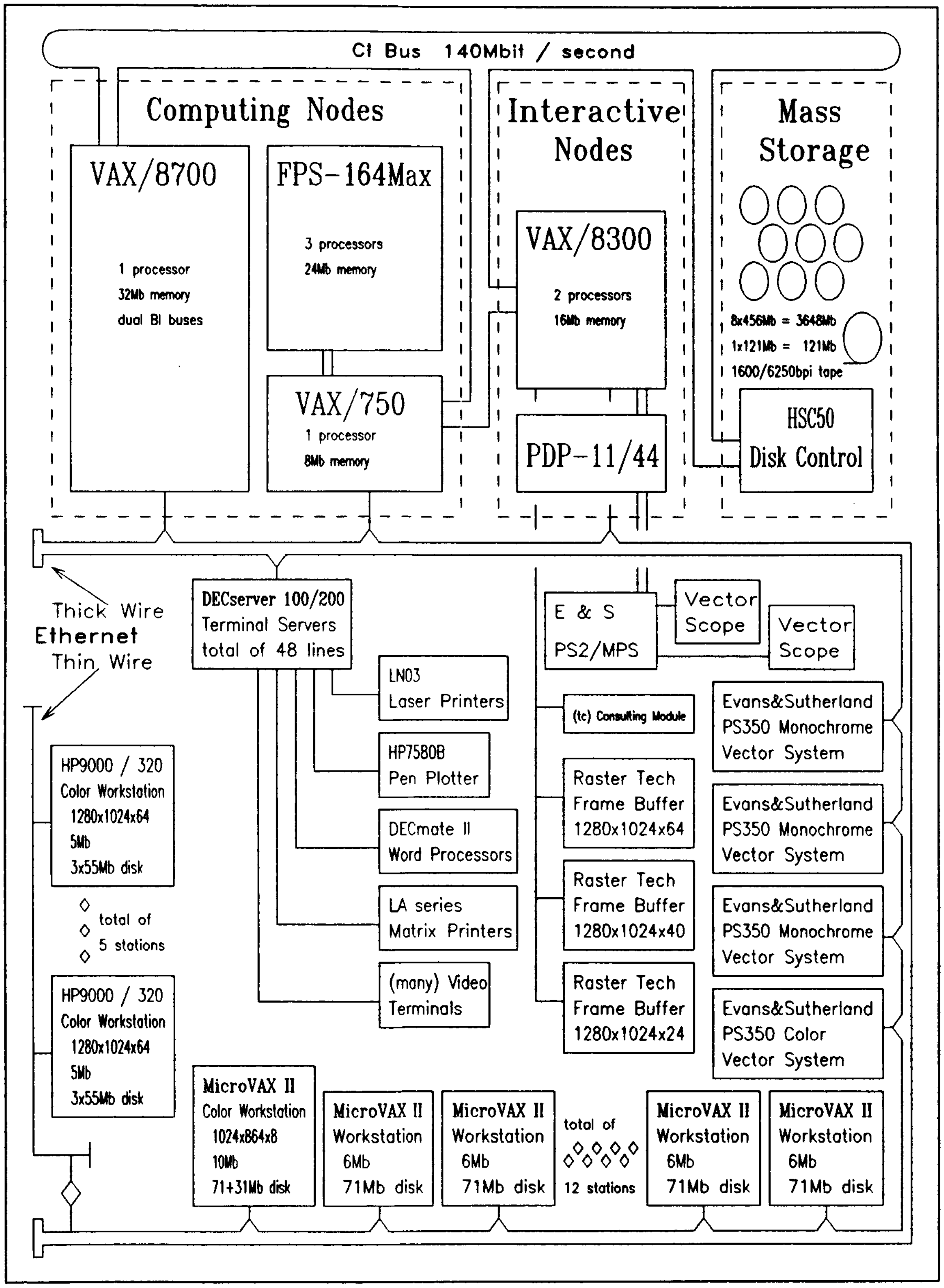


Figure 2.1 Hardware configuration - PCG, Cornell University

CHAPTER 3

ENHANCEMENTS TO THE EXISTING ENVIRONMENT

This chapter discusses various enhancements made to the existing postprocessing environment. Some of these are necessitated by the hardware change. The process of subobject extraction by specific material types has been made more visually interactive. The existing database has been expanded to include information useful for the qualitative evaluation of finite element meshes. It is also possible, to contour additional response parameters such as pore water pressure. The response smoothing technique used for 15-noded 3-D wedge isoparametric elements has been improved. Finally, a new section of the program allows the user to add annotation to any image; this feature permits the postprocessor to serve needs of both demonstrations and design reports.

3.1 Material Extraction

Material extraction is a process by which the user can create a subobject of a multiple-material parent structure based on material types. All elements of one or more user-specified material types are extracted.

In the existing postprocessor, the user was required to choose material types from a table consisting of the numbers corresponding to the materials in the structure. No material information -- parameters like moduli or spatial location within the structure -- was provided.

The process is now not only visually interactive, but information about the materials in the structure can be obtained. On entering the Material Extraction menu page, the structure is redrawn using a different color for each material. A color map associates material numbers with the colors used to differentiate them. This gives the user visual feedback of the spatial location of the different material types in the structure. Material information can be obtained by "pointing" to either the image in the main view or to the appropriate color on the color map. The required material types may also be selected in the same manner. Figure 3.1 shows the image of a dam and its foundation. The various materials are shown, each in a different color. The material corresponding to the "arch" portion of the dam is interactively selected and the elements belonging to this material type are redrawn using a dark grey to contrast them from the rest of the structure. Flexibility is maintained by allowing the user to "unselect" any selected material types before asking the program to extract the subobject. The user hits the "EXTRACT" button to initiate the extraction process and the program creates a set of data files for the subobject. Figure 3.2 is an image of the extracted subobject. Strain energy density is contoured on its surface.

<u>Implementation :</u> When the structure is initially redrawn according to material type, a software display list of all the boundary polygons, segmented by material type, is maintained. This speeds up the contrasting procedure as only the appropriate boundary polygons are redrawn using a different color. The internal display list of the Rastertech frame buffer is not used due to the fact that memory space is limited to only one megabyte.

The same procedure can be made faster by using the internal display list and segmenting capabilities of the HP Renaissance systems. A second display list of boundary polygons sorted according to material type can be generated the first time the user enters the Material Extraction menu page. This list can be stored in virtual memory for the complete duration of the interactive session. The fact that the display list of the Renaissance resides in virtual memory allows the second list to be stored without creating problems with regard to memory space. The expected performance reduction due to memory "page faulting" will need to be studied.

3.2 Changes to the Existing Database

The features of greatest importance to a user of postprocessing programs is flexibility, a wide range of functionality and interactive response time. This poses a dilemma to the designer of a database for engineering postprocessing. It can be stated as the tradeoff between the use of additional memory, and the loss of interactivity due to page faulting or computations done on the fly. On the one hand, a large database becomes difficult to handle and slow to manipulate, and on the other, too many computations done during the interactive session reduces interactivity. Thus the designer of the database must make crucial decisions that determine what is to be computed before the interactive session and stored in the database, and what is to be computed during the session.

A decision has been made to store more information in the existing database. This information relates to color maps, gauss point response values for all the elements, and unaveraged response values at nodes on the boundary of the structure. One of the consequences of this change is a larger database with the accompanying problems associated with size. Response to database querying commands has not become noticeably slower, but it now seems appropriate to redesign the database. This question is addressed in greater detail in Chapter 5.

3.2.1 Software Color Maps

The Lexidata frame buffer is configured as a one-channel system with pixel values that are indices into a 12-bit color look-up table (CLUT). As the Rastertech frame buffer cannot be configured in this manner, the 256 shades of each of the 10 base colors of the current color map are now stored in the database. This acts as the software equivalent of the CLUT in the Lexidata frame buffer. The three channels of the Rastertech frame buffer are each loaded linearly with values between 0 and 255 and are left unchanged during the interactive session.

Consider the case of a polygon that has to be rendered. The base color of the polygon has been computed depending on the response values in that region of the structure relative to the global maximum and minimum values. The angle that the normal to the polygon makes with the unit vector representing the direction of the light source is computed in software. The required shade of the base color, as an RGB triplet, is extracted from the CLUT stored in the database. This is now used to index into the three Red, Green and Blue channels before sending the polygon down to the frame buffer for rendering. This software mapping simulates a 1-channel system while utilizing the larger color pallette of the 3-channel system.

The use of software color maps has a number of advantages. Coupled with dithering, it could artificially increase the number of "allowable" shades of colors that may be represented on the screen. This would improve the quality of the image without necessitating a larger pallette of colors or higher resolution. For example, the programmer may require a pixel to be colored with an RGB triplet (200.5, 50.75, 146.3). The Digital to Analog Converters (DACs), used to convert color index values to voltages for the guns, allow only integer values between 0 and 255. The dithering software would first detect that this color lies between (200, 50, 146) and (201, 51, 147). It would then use a statistical distribution of allowable pixel values in a certain region of pixels around this one. The effect is to fool the observer into perceiving the required shade at that location. The HP Renaissance systems do the dithering computations in hardware during scan conversion. This speeds up the process considerably.

Another advantage of using software color maps is an increase in the portability of the graphics code. The postprocessor can now be used to interface with both one and three-channel systems easily. Hence, if the program needs to be ported between two such systems, only the device-dependent code requires modification.

3.2.2 Unsmoothed Response Information at the Boundary Nodes

The existing database stores averaged responses at each node in the structure. Element by element extrapolation of gauss point responses results in different values of the same response at nodes which have more than one element framing into them. These values are averaged, resulting in the forced creation of single-valued responses at each node. The ranges of each of the response values at a node serve as an indication of the quality of the analysis in that region. Unaveraged response information is used to create 2-D plots along any line on the surface of the structure (refer to Chapter 4 for details.) If only the averaged values are stored, information regarding the quality of the analysis is lost.

The new version of the database stores unaveraged responses at the nodes of each boundary face. This information is not stored at the internal nodes as the memory requirements would be large, and the use of this information is low. These values would be used only when the structure is sectioned or when a subobject is created.

All computations and data management for the unaveraged responses of the parent structure is done in the translation phase. Unaveraged values of the strain energy density, pore water pressure (if it exists in the analysis), six cartesian stress components and six cartesian strain components are stored for the nodes of each boundary face. Principal stresses and strains, the tensor invariant quantities, and effective stress and strain are computed, if required, during the interactive session.

The alternative to storing the unaveraged values is to store the gauss point responses for each element and to do the extrapolation on the fly. This would require less memory space (though not significantly less), but would not be interactive. To be able to extract the information to create 2-D plots of unaveraged responses interactively, it is necessary to store the unaveraged values in the database. The price is paid in the form of memory space. Typically, it has been found that storing unaveraged response information increases memory requirements up to 50% in problems with large surface area to volume ratios. It must be noted that if a boundary node has 'n' boundary faces framing into it, 'n' values of each response - extrapolated from the gauss points of each of the corresponding 'n' elements - must be stored for the node. This accounts for the dependancy of the memory requirement on the surface area to volume ratio.

3.2.3 Subobject Creation

This is an interactive process by which the user can extract a part of the parent structure and create a new object, called a subobject, that can be individually viewed. The criterion for the extraction may be a cutting plane or a spatial enclosure, or may depend on material attributes.

To be able to render 2-D plots of unaveraged responses on the surface of subobjects, either unaveraged responses at all the internal nodes or the gauss point response values of the elements should exist in the database of the parent during subobject creation. For the reasons discussed in section 3.2.2, the unaveraged values at the internal nodes are not stored. Hence it is necessary to maintain a copy of the responses at the gauss points of all the elements of the parent. This is done during the translation phase. A file containing the gauss point values of the elements of the parent is created. This is the "problem_name".GPV file.

The existing element extraction algorithm did not "compute" any response values for the subobject database. Each internal node was associated with single-valued (averaged) responses. The subobject response database was a subset of the parent response database. In the new version, unaveraged response information at the nodes on the newly formed boundary surfaces does not exist in the parent database. These values have to be computed using the gauss point values of the elements that form the outer layer of the subobject, obtained from the .GPV file of the parent. For each of the boundary faces, responses are extrapolated from the gauss points of the associated element to the nodes on the face. This information forms part of the subobject response database and may be used to generate 2-D plots of unaveraged responses on its surface. A "subobject_name".GPV file containing the gauss point responses of the elements of the subobject is created. This file, a subset of the corresponding file of the parent, is used if a subobject of this subobject is extracted.

Although the additional computations necessary during subobject extraction make it slower, the process is interactive for the size of problems being postprocessed. At present, these computations are done on the fly and the user cannot proceed with the session until the extraction is complete. As the size of problems being analyzed increases, that of subobjects becomes correspondingly larger. This would mean that subobject extraction would no longer be an interactive process. This process would then have to be submitted to the host in a batch environment, thereby allowing the user to continue with the session without interruption. On completion of the batch process, the subobject database could be drawn into working memory to be viewed individually.

3.2.4 New Response Parameters for Display

The addition of new parameters into the postprocessor is not as simple as it could have been with a different approach to programming practice. In large graphics programs, it is necessary to build flexibility and the "hooks" for future functional expansion into the code. There are a number of sections of the postprocessor that need to be coded dynamically; especially in a research environment, the requirements of users change often. Sufficient foresight is necessary to ensure that atleast these sections of the code can be easily modified or extended. The manipulation of lists of character strings is a typical postprocessing function that demonstrates the need for a flexible programming style. For example, the "names" associated with response parameters, and the types of parameters available for display can change. The existing postprocessor has the names of responses hard-coded into the program. The new design of the postprocessor (Chapter 5) solves this problem by making use of a generalized list processor to process any list of strings. Response names, in the form of strings, are stored in a data file created by the analyst. Changing a response name or adding a new response becomes the simple task of editing this file. Similarly, messages to the analyst, connected with specific response names, are now hard-coded into the program. A dynamic list of the existing response names, (character strings), would solve this problem elegantly.

The existing database has been well designed to accomodate additional response parameters for display. The numbers of scalar, vector, and tensor parameters that the database contains form part of the input data. This makes the addition of new responses into the database a simple task.

New Parameters : On-going research in the analysis of drilled concrete shafts in soil necessitated the inclusion of pore water pressure as a scalar response parameter that can be handled by the postprocessor. The finite element translator program is modified to read the pore water pressure response file created by the analysis. Also, the postprocessor now offers this scalar quantity as a response that can be contoured, when applicable.

The existing boundary element analysis program generated only displacements and tractions as output and the postprocessor displayed these response parameters. It is often easier to understand the behavior of a structure when the tensor stress components can be contoured on its surface. Thus, this analysis program has been modified to compute cartesian tensor components of stress from the tractions. To be able to display these values, the boundary element translator program and the postproprocessor have also been modified . Averaged and unaveraged stress information is stored for each of the nodes of the boundary elements in the modified database. The unaveraged response values facilitate the use of the plotting feature (refer to Chapter 4 for details).

3.3 Annotation

As the postprocessor is used as visual and graphical display of computational mechanics research, there exists a need to add simple forms of annotation to an image. This is useful for reproduction of images during demonstrations, and in technical papers or reports.

An Annotation menu page has been added to the postprocessor. In this page, a user may add annotation to the image in the main view window. Text, arrows and circles can be added in overlay. Annotation may be erased from the entire screen, or from a part of it, defined by "rubber-banding" a rectangle interactively. These are basic features, but serve the purpose temporarily. Figure 3.3 shows an example of annotation in the green overlay, added to a contoured image.

An existing feature of the postprocessor that complements the addition of annotation to an image is the Snap feature. The Lexidata version of the postprocessor allows the user to store an image by creating a file of the pixel information. This image can later be reproduced on the screen. The low resolution and fast I/O to and from that frame buffer makes both these processes interactive. On the Rastertech, higher resolution and slower frame buffer I/O makes this process non-interactive; it takes as long as 10-15 minutes to store an image and up to 5 minutes to reproduce it. It must be noted that the information necessary to store a 1280x1024 24-bit image is 8 times more than that for a 640x512 12-bit image. The Snap feature has been temporarily deactivated and will be revived in the HP version of the program.

3.4 Response Smoothing - 3-D Isoparametric Wedge Elements

The existing technique for the extrapolation of responses from the 14 gauss points to the nodes of 15-noded, 3-D isoparametric wedge elements is inaccurate in regions of high response gradients. Figure 3.4 shows the spatial location of the 14 gauss points. The technique assumed that these planes may be approximated to spatially coincide with the "end" planes of the wedge element. This implicitly assumes that the variation of responses along the longitudinal axis of the element is negligible. The extrapolation is done from the 7 gauss points, "on each end plane," to the nodes on that plane. The responses at gauss points a, b, c, d, e, f, and g are extrapolated to the 6 nodes on the face directly, using the shape functions of a 6-noded planar isoparametric triangular element. The same applies to gauss points h, i, j, k, l, m, and n. The responses on each of the end planes is uncoupled during the extrapolation process. The inaccuracy of this assumption increases as the variation of responses in the longitudinal direction of the element increases.

The extrapolation procedure implemented in the new version of the postprocessor makes more general assumptions about the variation of responses within the element. A linear least squares extrapolation technique is used to solve the overdetermined system -- 14 known gauss point response values and 6 unknown vertex nodal values. The derivation of this linear transformation is detailed below [11]. The shape functions of the 6-noded, linear wedge element are the interpolating polynomials for the least squares fit. The final values obtained are at the 6 vertex nodes of the element. Midside nodal values are obtained by simple averaging of the computed values at adjacent vertex nodes.

The linear least squares transformation may be described as,

$$\{ X \} = [T] \{ G \} \tag{3.1}$$

in which

$\{ X \}$ = (6x1) vector of unknown vertex response values

$[T]$ = (6x14) concatenated form of the transformation matrix

$\{ G \}$ = (14x1) vector of known gauss point values

For the given gauss point configuration, $[T]$ is a constant matrix for any 15-noded isoparametric wedge element. The components of $[T]$ have been precomputed and hard-coded into the program for speed of execution. The elements of $[T]$ are computed by solving the overdetermined Linear Least Squares problem defined as the minimization of the Euclidian norm,

$$\| [N] \{ X \} - \{ G \} \|_2 \tag{3.2}$$

in which

$[N]$ = (14x6) matrix of shape functions of the 3-D, linear, 6-noded wedge element evaluated at the 14 gauss points of the element

$[N]$ is not the interpolating shape function matrix used in the finite element solution of the 15-noded wedge element. In this case, the interpolant acts between response quantities which are functions of the first derivatives of the basic displacement degrees of freedom.

Hence, linear shape functions are used as the interpolants.

The problem given by equation 3.2 can be restated as computing { X } such that the following expression is minimized,

$$[\ [\ N\ ]\ \{\ X\ \} - \{\ G\ \}\ ]^2 \tag{3.3}$$

Expanding (3.3), one obtains

$$= [\ [\ N\ ]\ \{\ X\ \} - \{\ G\ \}\ ]\ [\ [\ N\ ]\ \{\ X\ \} - \{\ G\ \}\ ] \tag{3.4}$$

$$= [\ N^T\ ][\ N\ ]\{\ X^T\ \}\{\ X\ \} - 2\ [\ N^T\ ]\{\ G\ \}\{\ X\ \} + \{\ G^T\ \}\{\ G\ \} \tag{3.5}$$

Differentiating (3.5) with respect to { X } and equating the result to a null vector, one obtains

$$2\ [\ N^T\ ][\ N\ ]\{\ X\ \} - 2\ [\ N^T\ ]\{\ G\ \} = \{\ 0\ \} \tag{3.6}$$

or

$$[\ N^T\ ][\ N\ ]\{\ X\ \} = [\ N^T\ ]\{\ G\ \} \tag{3.7}$$

If,

$$[\ A\ ] = [\ [N^T]\ [N]\ ]^{-1} \tag{3.8}$$

is substituted into (3.7), one obtains

$$[\ A\ ]^{-1}\ \{\ X\ \} = [\ N^T\ ]\ \{\ G\ \} \tag{3.9}$$

Premultiplication of both sides of the equation by [**A**] yields,

$$\{ \mathbf{X} \} = [\mathbf{A}] [\mathbf{N}^T] \{ \mathbf{G} \} \qquad (3.10)$$

or

$$\{ \mathbf{X} \} = [\mathbf{T}] \{ \mathbf{G} \} \qquad (3.11)$$

in which,

$$[\mathbf{T}] = [\mathbf{A}] [\mathbf{N}^T] \qquad (3.12)$$

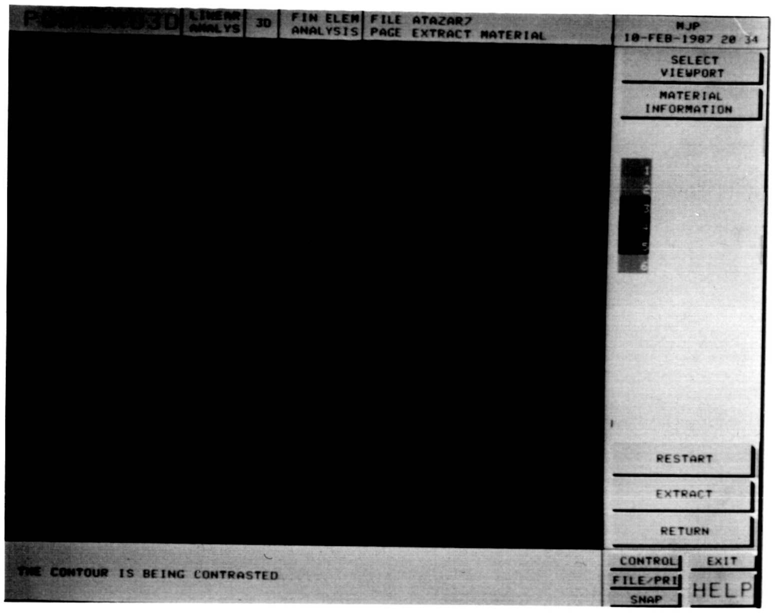
Note that [**A**] can be explicitly computed because [[$\mathbf{N}^T$][**N**]] is a 6x6, square matrix. [**T**] is the required 6x14 transformation matrix. Its components depend on the shape functions of the 6-noded linear wedge element, evaluated at the 14 gauss points of the 15-noded element. The explicit form of [**T**] is documented in Appendix C.

As [**T**] has been explicitly coded into the program, the extrapolation process is fast. Use of this technique improves the accuracy of the extrapolation in regions of high response gradients as the responses are linearly extrapolated along the longitudinal axis of the element rather than just within the triangular end planes of the element.

3.5 Default View Specification

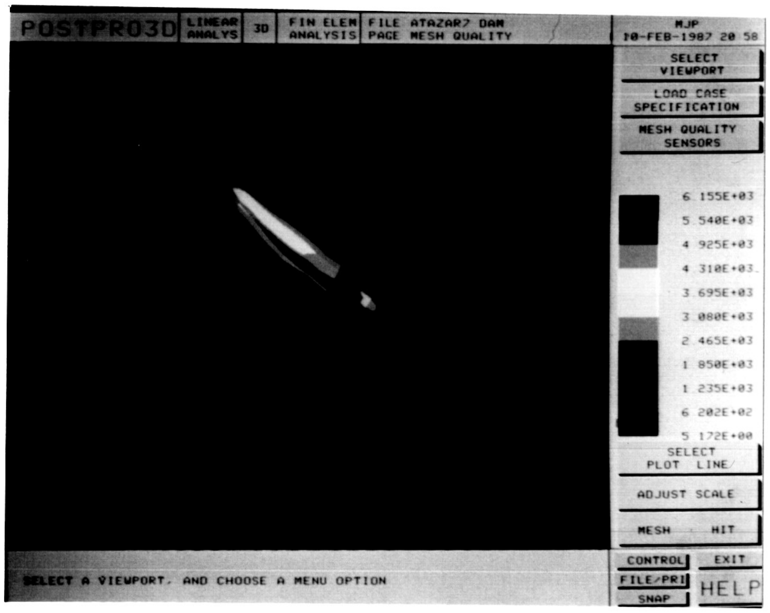
The primary feature of the View Specification menu page is the interactive control of the processes that allow the analyst to rotate and spin the structure to the most convinient position for viewing. It has been found that the analyst periodically positions a given structure in the same location. A new feature now allows the analyst to store and modify a desired default position.

If an analyst desires to store a default position, a button - "STORE DEFAULT POSITION" - is hit. This creates a "file_name".POS file that contains the 4x4 transformation matrices that form the total concatenated transformation matrix. When a structure is selected for postprocessing, the program checks the directory for the existence of this file. If it exists, the default transformations stored in it are read, and used to position the structure. A new .POS file can be created by repeating the operation.



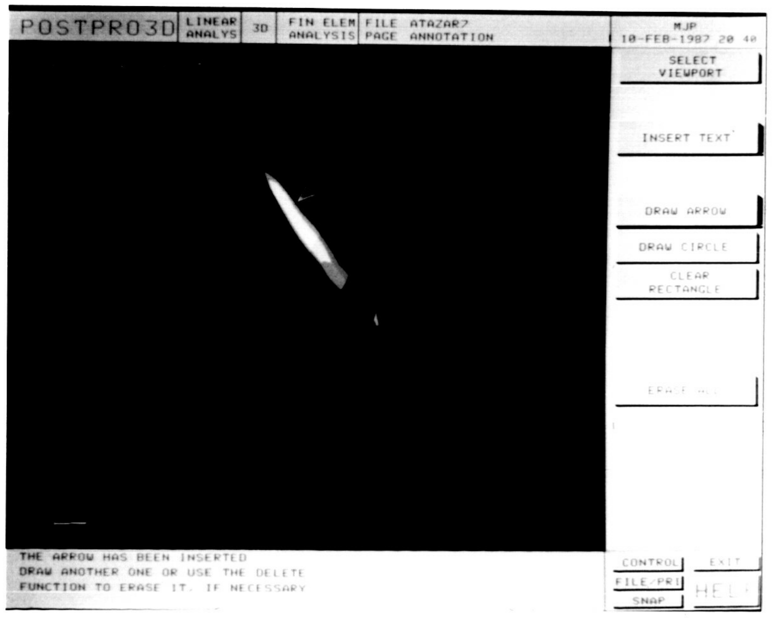
THE CONTOUR IS BEING CONTRASTED

Figure 3.1 Material Extraction - material 1 is selected



SELECT A VIEWPORT, AND CHOOSE A MENU OPTION

Figure 3.2 Material 1 extracted - contours of strain energy density



THE ARROW HAS BEEN INSERTED
DRAW ANOTHER ONE OR USE THE DELETE
FUNCTION TO ERASE IT. IF NECESSARY

Figure 3.3 Annotation menu page - image with annotation

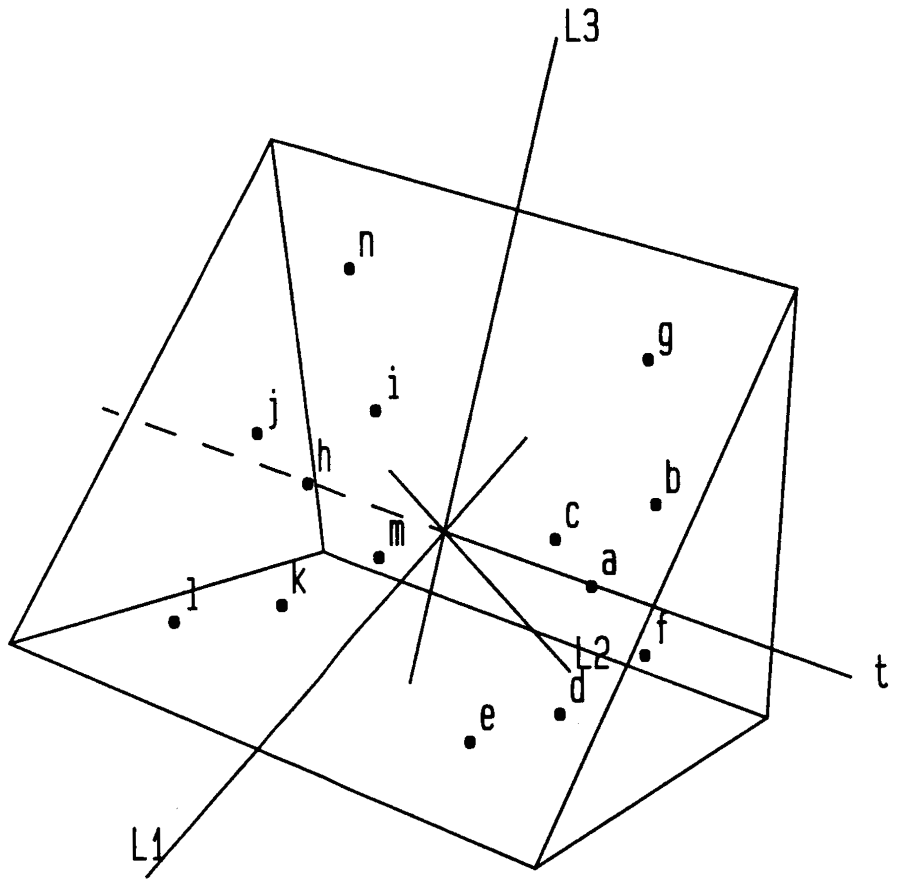


Figure 3.4 Gauss point locations for a 15-noded wedge finite element

CHAPTER 4

MESH QUALITY EVALUATION

Research in the area of mesh quality evaluation has shown that the most useful evaluators are those computed **a posteriori**. This chapter outlines current research in the area of **a posteriori** finite element error estimation from a single analysis, and describes the graphical implementation of some qualitative error sensors in POSTPRO3D. The objective of this section of the research is to provide an analyst with interactive graphics tools to facilitate the qualitative evaluation of a finite element analysis. The algorithms used are described and their efficacy is demonstrated with the use of example problems.

4.1 Definitions

In the literature on **a posteriori** error analysis, there is a profusion of terminology used to label similar error quantities. Although the terminology differs, the basis for classification is generally uniform and is the "cause" or source of the error.

The development of the finite element method so far shows that it is especially suited for the solution of positive definite, self-adjoint, elliptic systems of partial differential equations. The method leads to discretization equations that are solved using a digital computer with finite wordlength. The numerical solution over the domain is never exact -- except in the case of trivial problems like constant stress states of simple configurations like patch tests. An infinite degree-of-freedom system is reduced to a finite degree-of-freedom system so that it can be processed by the digital computer. Mathematical idealizations, necessary for further simplification, introduce errors. The finite wordlength of digital computers result in the round-off of decimal values and the truncation of values when numbers of dissimilar magnitude are added or subtracted; therefore, the solution of the linear or nonlinear simultaneous equations is inexact.

4.1.1 Error Quantities

Before error sensors are described, it is essential to define some of the terms encountered in the literature that characterize various types of error quantities.

4.1.1.1 Local Error Quantities

These are error quantities that are computed at any nodal location in the domain of the solution.

<u>Total Error</u>: Assume that careful measurements of the actual response of a structure to a set of external forcing functions are made. Let these "exact" measured responses be denoted by $u_E(x,t)$ and the computer solution by $u_C(x,t)$.

Then, the Total Error, $e_T(x,t)$, is given by:

$$e_T(x,t) = u_E(x,t) - u_C(x,t) \tag{4.1}$$

This error is the sum of the various errors defined below.

Mathematical Modelling Error: Real structures always contain features that are unique to each and probabilistic in occurrence -- material properties, flaws, geometrical irregularities, and external loading. These are not only difficult to model mathematically, but, if modelled accurately, would drastically increase computational expense. So a number of simplifying idealizations are normally made. These result in that part of the Total Error called the Mathematical Modelling Error. Let $u_M(x,t)$ be the "exact" response of the idealized continuum at the nodal locations. Then the Mathematical Modelling Error, $e_M(x,t)$, is given by:

$$e_M(x,t) = u_E(x,t) - u_M(x,t) \tag{4.2}$$

Discretization Error: This error quantity, though not necessarily the largest portion of the Total Error, is of most interest to structural analysts. It is a natural consequence of the reduction of an infinite degree-of-freedom system (the real structure) to a finite degree-of-freedom system (the discretized model). Unless the order of the interpolating functions used to describe the behavior of the finite elements can exactly represent the structural response, this reduction introduces a Discretization Error into the computed response parameters at every point in the structure. Let $u_D(x,t)$ be the "exact" response of the discretized mathematical model. Then, the Discretization Error, $e_D(x,t)$, is given by:

$$e_D(x,t) = u_M(x,t) - u_D(x,t) \tag{4.3}$$

Roundoff and Truncation Error: The finite wordlength of digital computers compels the truncation of values after a certain number of decimal places. The solution process involves a large number of computations; the cumulative truncations and roundoff result in a Roundoff Error being introduced into the solution. The Roundoff and Truncation Error, $e_R(x,t)$, is given by:

$$e_R(x,t) = u_D(x,t) - u_C(x,t) \tag{4.4}$$

Solution Error: The Solution Error is the sum of the Discretization and Roundoff Error quantities. It can also be expressed as the Total Error less the Mathematical Modelling Error. This is the error in the solution assuming that the "exact" structural response is that of the continuum described by the mathematical model (with its simplifying assumptions). The Solution Error, $e_S(x,t)$, is given by:

$$e_S(x,t) = u_M(x,t) - u_C(x,t) \tag{4.5a}$$

$$= e_D + e_R \tag{4.5b}$$

4.1.1.2 Global Error Quantities

Global error quantities attempt to characterize the overall quality of the solution as a single scalar value. Rather than compute an error quantity such as e(x,t), some norm of these values, $||e||$, is computed. Error norms used follow the usual vector norm rules:

Positivity Rule: $$||e|| \geq 0.0 \qquad (4.6a)$$

$$(\; ||e|| = 0.0 \text{ iff } e(x,t) \equiv (0) \;)$$

Homogeniety Rule: For any arbitrary scalar λ,

$$||\lambda e|| = |\lambda| \; ||e|| \qquad (4.6b)$$

Triangle Inequality: If $e_1 + e_2 = e$, then

$$||e_1|| + ||e_2|| \geq ||e|| \qquad (4.6c)$$

Utku and Melosh [12] give a brief description of error norms.

4.1.2 Hierarchical Finite Elements

In the finite element method, convergence is obtained by using either the h- or the p-convergent methods. Both involve an increasing number of degrees-of-freedom, but use different techniques to introduce them into the discretized model. Hierarchically defined finite elements are used in p-convergent methods.

These are finite elements over whose domain interpolating polynomials of any arbitrary order may be defined. They are constructed such that inter-element continuity requirements -- even between elements that have interpolating polynomials of differing order -- are exactly satisfied. This process is accomplished by condensing or constraining the stiffness matrix of the element with the higher interpolating order. An important feature of the element formulations is that the stiffness matrix of an order **n** element is a subset of the stiffness matrix of an order (n+1) element. This implies that the addition of a single degree-of-freedom in an element leads to the introduction of a row and a column to the global stiffness matrix. Making use of the previously triangularized matrix, this fact is exploited to make the re-solution process inexpensive.

The characteristics of these finite elements make them ideal for a p-convergent method of adaptive refinement. A solution with a user-prescribed tolerance may be obtained when coupled with a global error estimator. Gago, Kelly, and Zienkiewicz [13,14] give a detailed description of the use of hierarchical finite elements and global error norms for error analysis and adaptive mesh refinement for 2-dimensional elasticity.

4.2 Mesh Quality Sensors

Mesh quality sensors are scalar quantities computed for each finite element subdomain. They attempt to characterize the quality of the solution locally and are functions of the computed response parameters and their gradients. Knowledge of the way some of the responses, such as strain energy density, behave as the solution tends to the exact one is also used.

Primarily, mesh quality sensors must lead to absolute or relative estimates of the discretization error and, when accuracy is inadequate, indicate regions of the mesh that require refinement. They must be locally stable, i.e. they must converge as the local solution converges to the exact one. It is also desirable that they are computationally inexpensive; otherwise, it might be cheaper and simpler to re-analyse the model with a higher number of degrees-of-freedom. Melosh and Utku [15] list various sensors proposed in the literature and briefly compare their performances.

To increase the efficiency and reduce the cost of the design process, postprocessing software tools must display the results of analyses in a user-friendly, flexible, and detailed manner. An important function is to indicate to the analyst those regions of the mesh that require refinement, if any. Mesh quality sensors perform this task. They may also be used as the criteria for automatic mesh redesign, that is, a process which aims at generating a solution of desired accuracy while minimizing computer time in the search for the optimal mesh.

4.3 Literature Survey

This section briefly discusses the progress of research in the field of **a posteriori** error analysis for finite element solutions. Key papers are referenced; the bibliographies of these papers are extensive. Energy-based methods are emphasized.

4.3.1 General Research-Review Papers

There are some papers in the literature that have surveyed this field in detail. Shephard [2] presents a review of the progress made in the area of finite element grid optimization in the 1970's.

Melosh and Utku [12] examine finite element technology and the processes which bear on designing efficient meshes. Discretization error sensors are discussed and their characteristics compared. The conclusions contain guidelines for choosing appropriate element types and grid arrangements that minimize the error in the solution.

Utku and Melosh [15] survey the state-of-the-art of various aspects of finite element error phenomena such as the quantification of its basic components, the understanding of their propagation, and the estimation of the errors present in the computed response.

4.3.2 Energy Basis for the Evaluation of Errors

The most widely used mesh quality sensors are energy-based. The reasons for this become clear on examination of the functions that a sensor must preform. A useful sensor must be a locally stable scalar quantity that characterizes the complex behavior of a structure. The strain energy density, at various points in the structure, satisfy these criteria exactly. The real structure is idealized as a finite degree-of-freedom system. This "stiffens" the structure and reduces its strain energy absorption capacity. As the number of degrees-of-freedom increase, the model becomes more flexible and is allowed to absorb more strain energy. A convergence of the total absorbed strain energy in the model indicates overall convergence of response values like displacement. Thus, strain energy and its gradient are efficient and intuitive mesh quality sensors.

The search for an efficient sensor has resulted in a bifurcation of research efforts:

1) Existing mathematical theory of error analysis for the numerical solution of partial differential equations is used to compute error

norms that attempt to bracket the global discretization error in as narrow a range as possible.

2) Strain energy density and its associated gradients and differentials are used to qualitatively and quantitatively characterize the error locally, over the finite element sub-domains of the structure.

4.3.2.1 Energy Norms

Energy norms constitute a mathematical approach to error estimation. They are used to characterize the discretization error globally.

Consider the following partial differential equations:

$$\mathbf{L}\,\{\mathbf{u}\} + \mathbf{p}' = \{0\} \quad \text{in } \Omega \tag{4.7a}$$

$$\mathbf{M}\,\{\mathbf{u}\} + \mathbf{r} = \{0\} \quad \text{in } \Gamma \tag{4.7b}$$

in which,

Ω	= the domain of the differential equations
Γ	= the boundary of the domain
$\{\mathbf{u}\}$	= the desired set of responses at the nodal points
$\mathbf{L}, \mathbf{M}$	= linear differential operators
$\mathbf{p}$	= a forcing function (say loads on a structure)
$\mathbf{r}$	= any arbitrarily defined set of boundary conditions

The approximate solution is obtained by assuming a trial function expansion of the kind

$$\mathbf{u}^{*} = \sum_{m=1}^{M} a_m N_m \tag{4.8}$$

If u is the exact solution, then,

$$\text{error, } E = (u - u^{*}) \tag{4.9}$$

The energy norm is defined as,

$$||E|| = [\int_{\Omega} E \, L \, E \, d\Omega]^{1/2} \tag{4.10}$$

This norm represents the "correctness" with which the function is modelled by the approximation (4.8).

Let R be the domain residual given by

$$R = L \{u^{*}\} + p \neq \{0\} \tag{4.11}$$

Then it can be shown by the Galerkin method that

$$||E||^{2} = - \int_{\Omega} E \, R \, d\Omega \tag{4.12}$$

This residual, R, is computed by substituting the solution into the differential equation. The energy norm, (4.12), is then computed. This method has some obvious disadvantages. First, the explicit form of the differential equations being solved must be known and often it is not. Secondly, (4.12) is a volume integral over the entire domain of the equations and is expensive to compute.

The mathematical theorems associated with research in this area and their proofs are proposed and derived by Babuska and Rheinboldt [4, 16-19]. Proofs for the one-dimensional case are detailed and the extension to two and three dimensions is outlined. The approach is based on the use of energy norms in the estimation of the discretization error. Babuska applies the error norm to the finite element solution of the Laplace Equation over an L-shaped domain. This has since become a benchmark problem in discretization error analysis research. The error involved in one- and two-dimensional elasticity problems has been closely bracketed.

Zienkiewicz and Morgan [20] define the discretization error and describe the use of an energy norm to estimate it in a p-convergent technique. Kelly, Gago, Zienkiewicz, et al. [13,14] give a detailed description of the practical application of this method in a computer program, and solve example problems in one and two dimensions to demonstrate the efficiency of the method for error sensing and adaptive mesh refinement. Unfortunately, this energy norm when coupled with hierarchically defined finite elements is not always reliable. Zienkiewicz [20], referring to the energy norm computed by using the domain residual and a hierarchic description of elements, says, "It has to be noted that this estimate is only valid in some asymptotic sense as 'h' (the characteristic mesh size) tends to zero, and is not reliable on finite meshes..."

The limitations of the energy norm approach are twofold. First, research in this area has produced limited results in actual applications due to a number of reasons. The approach is highly mathematical and computationally expensive. This scheme can be efficiently implemented only with the use of hierarchical finite elements, but this tends to make the error quantity unstable both locally and globally. Second, all the research and applications so far are restricted to 1- and 2-dimensional elasticity. The extension of the theory to 3-dimensional structures has not been achieved.

4.3.2.2 Strain Energy Density

Section 4.3.2.1 discusses the theoretical basis for the use of strain energy density as a mesh quality sensor. Research in the area of automatic redesign of finite element meshes began with the use of strain energy and its gradients.

Among the first to study the gradient of strain energy density as an error sensor were Melosh, Killian, and Marcal [21,22]. This sensor is called the Backward Energy Difference.

4.3.2.2.1 Theoretical Basis for Energy Difference Methods

In the limit, as a characteristic mesh size tends to zero, each element could be imagined to reduce to a single point with a single value of stress, strain, and hence, strain energy density. From this can be deduced that the signs of a converged or converging solution include: (a) The change in the total strain energy absorbed by the model after the introduction of an additional degree-of-freedom is negligible. (b) The change in the strain energy absorbed by each finite element after the introduction of an additional degree-of-freedom is negligible, i.e., the strain energy absorbed by each of the elements tends towards a constant value (a different constant for each element) as the solution approaches $u_D(x,t)$ -- the "exact" solution of the mathematically idealized, discretized structure. Criterion (b) can be used to ascertain whether an element has reached convergence or not.

Another qualitative measure of convergence, also based on strain energy, is a study of the rate of change of this value within an element. If the variation is rapid, then the region specified by the element has not attained its final strain energy absorption capacity. This is a useful visual aid in the process of evaluating the quality of the solution and is discussed further in section 4.4.1.

4.3.2.2.2 Research in Energy Differential Techniques

Melosh and Marcal [22] proposed the use of the Backward Energy Difference as a mesh quality sensor. It was suggested that this value could be approximated by taking the difference between the strain energy density at the "center" of the element and at any other point (say a Gauss point). This approximate value is called the Specific Energy Difference and its use is demonstrated with some simple examples.

Peano and Riccioni [3] demonstrated the limitations of the above method and, along similar theoretical lines, proposed the Forward Energy Difference sensor. Hierarchical finite elements are used to introduce degrees-of-freedom in locally defined areas of the mesh. This method, in conjunction with fracture mechanics techniques, is applied to the solution of real problems [23-25] and has been found to be successful.

Shephard, et al. [26-27] used linearly spaced, strain energy density contours as mesh quality indicators for the automatic generation of near-optimal meshes.

4.3.3 Residuals of Governing Differential Equations

Carey and Humphrey [28-30] suggested the use of equation or domain residuals as error estimators. The applications and derivations are made for one-dimensional cases and, "...the extension to higher dimensions, though posing more technical programming difficulties, has been indicated." The authors mention that for higher dimensions the problems of inter-element continuity and the use of constraints or Lagrange multipliers to force inter-element compatibility into the governing functionals complicate the issue. The explicit differential equations need to be known and the method is computationally expensive.

4.3.4 Errors for Specific Differential Equations

Some researchers have mathematically derived discretization error bounds for specific types of partial differential equations solved numerically using the finite element technique. The most common of these is the Poisson's Equation which governs a variety of so-called "field" problems, including torsion and steady-state heat and fluid flow. Error bounds based on the solution of this equation using the finite element method can be found in the literature [31-34].

Reference [34] is the solution of an error bound for the Poisson's Equation using a least squares finite element solution criterion. Element residuals are used as a measure of error. The author states, "...while the relative changes of the above residuals for increasing numbers of elements or degrees-of-freedom will indicate convergence, they will give no absolute measure of the error in the solution. Thus the idea of error measures is introduced."

4.4 Qualitative Evaluation of Meshes Using Graphics

Some of the main techniques available in the literature for **a posteriori** error estimates of discretization error have been presented in the previous section. This section discusses some qualitative sensors that may be used in interactive graphics postprocessors and the implementation in POSTPRO3D. Emphasis is given to the use and implementation of inter-element response discontinuities as mesh quality sensors.

A qualitative sensor indicates to the analyst, in an overall behavioral sense, the accuracy of the solution. A study is made of unsmoothed response values computed from the solved displacements at the nodal locations. These sensors give the analyst an initial "feel" for the quality of the solution and indicate the need for further accuracy checks. To some degree, the experience of the analyst is called upon to compare the variation of response parameters over the domain with previous cases and to make a judgment about the quality of the solution.

4.4.1 Strain Energy Density Contours

The theoretical basis for the use of strain energy density contours has been discussed in section 4.3.2.2. The effectiveness of this measure as an error sensor is demonstrated by Shephard [26].

This qualitative measure has been found to be useful. Consider a case where the discrete color map used for the color contours contains ten discrete ranges between the global maximum and minimum response values. Any element that attempts to simulate a variation of strain energy density that is a significant portion of the global range (say more than 3 of the discrete colors of the map) would need to be subdivided if a more accurate representation of responses in the region is required. The example problems in section 4.4.3.2.3 discuss the use of strain energy density contours in conjuction with 2-D plots of unaveraged responses as mesh quality sensors.

4.4.2 Known Surface Tractions

A qualitative mesh sensor used by analysts to get an initial "feel" for the quality of the analysis is the comparison of the tractions extracted from the finite element solution with the values of the tractions on those surfaces of the structure that have specified traction boundary conditions. For example, a structure may have a "free" surface along which normal and shear stresses are zero. The deviation of the numerical solution from the known values is an indication of the quality of the solution in the vicinity of the boundary. Note that the absolute deviation cannot be used as a measure. It must be compared with a characteristic value of that response elsewhere within or on the boundary of the domain.

The effectiveness of this sensor is limited. If the structure has a large volume to surface area ratio, large parts of the mesh lie away from the boundary. The error in the solution near the boundaries, indicated by this sensor, does not allow the analyst to draw any conclusions about the performance of the mesh in the interior of the structure. Also, this technique relies strongly on the use of an accurate method to extract tractions on the boundary. It is well known that the values of responses obtained at points away from the sampling points in an element depend largely on the extrapolation technique used. Thus, this sensor may be used to get an initial feel for the analysis in conjunction with other reliable ones.

Often, the errors found in surface tractions are due to the inadequacies of the element displacement shape functions to extrapolate Gauss point responses to the boundary. This is reinforced by the fact that the response gradients close to boundary surfaces are usually high.

Haber [35] suggests a new technique to extract accurate surface tractions from a finite element displacement solution. The method uses the solved stresses at the Gauss points of the elements along the boundary and the Principle of Virtual Displacements. It involves the re-solution of a set of simultaneous equilibrium equations involving the elements along the boundary of interest. The tightly banded nature of this new coefficient matrix reduces computational expense. The tractions at the surface nodes are the unknowns in this set of equations. The method is found to be accurate and reliable even in the case of abrupt discontinuities in surface tractions or singularities. The implementation of this method ensures that the error in the tractions is due to the discretization process in the vicinity of the boundary and not due to the use of inaccurate response-extrapolation techniques.

4.4.3 Inter-element Response Discontinuities

The solution of any structural problem by the stiffness method (the most common FE approach) is achieved by solving a set of simultaneous equations formed in an attempt to satisfy the conditions imposed by equilibrium and compatibilityand by the traction and displacement boundary conditions. The "exactness" of the solution depends on the degree to which some or all of these conditions are satisfied over the entire domain of the continuum. The displacement boundary conditions are imposed on the solution as data input. The enforcement of single-valued displacements at the nodal locations ensures the satisfaction of compatibility at these discrete locations, and for conforming elements, compatibility is satisfied everywhere. But intra-

and inter-element equilibrium is not necessarily satisfied, nor is boundary equilibrium. One signal of a converged solution, however, is that equilibrium is approached everywhere.

During the translation phase of postprocessing with POSTPRO3D, stress and strain responses sampled at interior points of elements are extrapolated to the nodes. A single node usually has a number of elements connected to it. As equilibrium is not locally satisfied, the values of a particular response at one such node -- obtained by the extrapolation process applied to each of the elements attached to it -- are not identical. In the real structure, each point in the continuum must have single-valued responses. This deviation from single-valued, pointwise, reponses is an indication of a lack of equilibrium locally, and hence, of the relative magnitude of the local discretization error.

As an alternative to extrapolation, nodal values in each element can be obtained by direct application of the response-nodal displacement equations at each node [36]. However, this calculation is significantly more time consuming in 3-D and would reduce interactivity. Therefore, it is not implemented in POSTPRO3D.

The ratio of the difference between the maximum and minimum response values at a node and the "exact" response value at that node is an indication of the relative magnitude of the response discontinuity in that region of the mesh. If these response "jumps" are large in a particular region of the mesh, then the solution can be estimated to be far from converged in that region. This qualitative measure of solution error is exploited graphically in POSTPRO3D. The following sections discuss its implementation -- interactive graphical algorithms involved and example problems to demonstrate its use in mesh quality sensing.

4.4.3.1 The Energy Error Quantity

An attempt was made to quantify a measure of "response jumps". The "jumps" for each of the six Cartesian stress and strain components at every node are combined into a single scalar quantity called the Energy Error Quantity, E, given by,

$$E = \frac{1}{2} \left[\sum_{\substack{i=j \\ j=1}}^{3} \delta(\sigma_{ij}) \cdot \delta(\varepsilon_{ij}) \right] \tag{4.13}$$

in which, at a node,

$$\delta(\sigma_{ij}) = (\sigma_{ij,max} - \sigma_{ij,min}) \quad \text{and}$$

$$\delta(\varepsilon_{ij}) = (\varepsilon_{ij,max} - \varepsilon_{ij,min})$$

represent the magnitude of the "jumps" of each of the 6 cartesian stress and strain components. E is analagous to the strain energy density of the "jumps" in the extrapolated values of the responses at a node into which more than one element frames. This measure of response discontinuities has four major problems: (a) It is no longer a strain energy measure in a material nonlinear analysis in which plastic yielding occurs. In this case it can be considered a measure of the "plastic work" of the response discontinuities -- if the 1/2 multiplier is removed. Another problem is that if part of the structure remains elastic, the two measures can not be compared. (b) At any node which belongs to a single element, the measure breaks down as,

$$\delta(\sigma_{ij}) \equiv \delta(\varepsilon_{ij}) \equiv 0.0 ,$$

and this implies that $E \equiv 0.0$, regardless of the accuracy of the mesh in this region. (c) Interfaces between different material types are regions of real response discontinuities. At these locations, no response averaging can be done and the measure breaks down as an estimator of discretization error. (d) Finally, there is a problem of scale -- relative magnitude of the "jumps" with respect to the actual values of the responses at the node.

Consider a node at which the actual strain energy density is close to zero. The exact stresses and strains are also close to zero. An inexact finite element solution gives responses that oscillate about zero, as it attempts to model an average value close to zero. This gives relatively large values of $\delta(\sigma_{ij})$ and $\delta(\varepsilon_{ij})$, and hence, E. If the Energy Error Quantity is normalized -- dividing it by the averaged strain energy density at the node -- meaningless ratios above 1.0 are obtained. Use of the maximum strain energy density in the computed results (for normalization) proved equally fruitless.

For the reasons outlined above, the Energy Error Quantity is unacceptable as a measure of mesh quality at all locations in a finite element mesh.

4.4.3.2 2-D Line Plots of Unsmoothed Responses

Abel, Panthaki, and Wawrzynek [36] discuss the use of 2-D plots as mesh quality sensors, and demonstrate this technique using a 2-D example problem. Extensions to 3-D problems and the use of interactive, color computer graphics are discussed in this and the following sections. Section 4.4.3.2.4 summarizes the results, and discusses the limitations of this approach.

In this section, it is shown heuristically, by means of example problems, that a 2-D plot of an unsmoothed response is a locally stable mesh quality sensor that approaches a "steady state" after the region has achieved convergence. Also described are the computer algorithms used to implement this measure in POSTPRO3D.

This qualitative sensor is found to be an accurate indication of the regions of the mesh that require additional degrees-of-freedom and those that have achieved convergence -- convergence can be reached locally, in certain parts of the domain. The local stability of this sensor is demonstrated by the fact that the infusion of additional degrees-of-freedom in a region that has converged does not affect its behavior. This is a necessary characteristic of a mesh quality sensor.

4.4.3.2.1 Extraction of Responses Along a Plot Line

Smoothed or averaged response data are needed for the purposes of discrete color contouring on the surface of a structure. But this data masks the information that is a necessary part of visual error sensing. Unsmoothed response data must be used to generate 2-D response plots if these are to be used as mesh quality sensors. This necessitates the storage of both smoothed and unsmoothed information for rapid access.

The existing database contained only averaged response information. To render the 2-dimensional plots interactively, it is necessary to also store unsmoothed response information at the nodes on the surface of the structure. The database has been modified to contain the unaveraged strain energy density, pore water pressure (if it exists in the analysis), and six Cartesian stress and strain components at the nodal locations of each of the boundary polygonal faces. The unaveraged responses are stored only at the surface nodes to reduce the memory requirement. The details of this addition to the existing database are discussed in section 3.2.2.

Assume that the analyst has defined a plot line on the surface of the structure along which it is required to plot various response parameters. The extraction algorithms create a list of line segments that describe the position of the plot line on boundary faces -- each boundary face that the plot line lies on has a 3-D line segment associated with it and the plot line is composed of these line segments connected in space. It is required to interpolate the known, unaveraged response values at the nodes of each of these boundary faces to the line segments for generating the response plot. The efficiency of the 2-D plot as a mesh quality sensor depends on the accuracy of the magnitudes of the inter-element response discontinuitues. It is thus important to use an accurate and consistent interpolation technique to extract the response values along each of the 3-D line segments that compose the plot line.

There is a particular problem associated with curved isoparametric finite elements. Section 4.4.3.2.2 discusses the algorithms used to extract the 3-D line segments. The method used is exact for boundary faces that are planar, but approximate for curved boundary faces. A curved face of an isoparametric finite element is defined using the shape functions in natural coordinates and is given by

$$\{ C \} = [N] \{ Cn \} \qquad (4.14)$$

$$\Rightarrow \quad X = \sum_{i=1}^{num\_nodes} (N_i \cdot X_i) \qquad (4.14a)$$

$$\Rightarrow \quad Y = \sum_{i=1}^{num\_nodes} (N_i \cdot Y_i) \qquad (4.14b)$$

$$\Rightarrow \quad Z = \sum_{i=1}^{num\_nodes} (N_i \cdot Z_i) \qquad (4.14c)$$

in which,

$\{ C \}$ = Cartesian coordinates of the point = $\lfloor X \; Y \; Z \rfloor^T$

$\{ C_n \}$ = Cartesian coordinates of the nodes of the element

$\{ N_i \}$ = Shape functions evaluated at the point

The definition of the face in Cartesian coordinates is nonlinear in $\lfloor X \; Y \; Z \rfloor$. The required line segment (a curve in 3-D space) is the intersection of the user-defined cutting plane and this boundary face. The solution of this nonlinear problem is iterative and expensive. Thus, an approximate scheme is chosen for curved elements.

The algorithms used for the extraction of line segments assume that the boundary face is planar. When the face is curved, the extracted line segment will either lie inside (for concave outward curvature) or outside (for convex outward curvature) the element. The algorithm computes the natural coordinates of points along the segment in

3-dimensional natural space, $\lfloor R \quad S \quad T \rfloor$. But it is known that the "real" line segment lies on a plane that has a constant value of one of the three natural coordinates, i.e., R or S or T = +1 or -1. The algorithm finds the natural coordinate that has the constant value on the boundary face and assigns the appropriate value to it, Step 3 of the following algorithm. The point is "projected" onto the boundary face orthogonally, in the direction of the natural coordinate that has a constant value on the face. The coordinates of the projected point in 2-D natural coordinates (this point is on the boundary face) are now known, and the unaveraged response values at the nodes are interpolated to it using appropriate shape functions, Step 4 of the following algorithm.

Algorithm for Extracting the Response Value at a Point:

Consider the case of a single boundary face and the line segment associated with it. It is required to extract the response values at points along this segment of the plot line.

The algorithm used to extract the response value at any point along the line segment is outlined below. It is repeated at each of the points on the segment at which the response value is desired.

The input data to the algorithm includes: (a) The type of finite element the face belongs to -- 20-noded brick, 15-noded wedge, etc. (b) Coordinates of the nodes on the face in world space. (c) Unaveraged response values at the nodes. (d) Coordinates of the end points of the line segment in world space.

begin algorithm

Step 1: The world coordinates of the point are computed

$$\lfloor \textbf{P\_WORLD} \rfloor = \lfloor \text{X\_POINT} \quad \text{Y\_POINT} \quad \text{Z\_POINT} \rfloor$$

Step 2: An iterative technique is used to compute the natural coordinates of this point,

$$\lfloor \textbf{P\_NAT} \rfloor = \lfloor \text{R} \quad \text{S} \quad \text{T} \rfloor$$

a) Start with an arbitrary guess of the natural coordinates

$$\lfloor \textbf{NAT\_GUESS} \rfloor = \lfloor \text{R\_GUESS} \quad \text{S\_GUESS} \quad \text{T\_GUESS} \rfloor$$

b) Compute the Shape Function matrix, $[N]$, using these natural coordinates.

c) Use $[N]$ and the world coordinates of the nodes to compute the world coordinates corresponding to these natural coordinates -

$$\lfloor \textbf{P\_GUESS} \rfloor = \lfloor \text{X\_GUESS} \quad \text{Y\_GUESS} \quad \text{Z\_GUESS} \rfloor$$

d) Compute the derivatives of the Shape Functions at this point.

e) Compute the 3x3 Jacobian matrix $[J]$, its determinant $|J|$ and inverse $[J]^{-1}$.

f) Compute a new guess of the natural coordinates,

$$\lfloor \textbf{NAT\_GUESS} \rfloor = \lfloor \textbf{DIFF} \rfloor [\ J \ ]^{-1} \qquad (4.15)$$

in which,

$$\lfloor \textbf{NAT\_GUESS} \rfloor = \lfloor \text{R\_GUESS} \quad \text{S\_GUESS} \quad \text{T\_GUESS} \rfloor$$

$$\begin{aligned}\lfloor \textbf{DIFF} \rfloor &= \lfloor \textbf{P\_GUESS} \rfloor - \lfloor \textbf{P\_WORLD} \rfloor \\ &= \lfloor (\text{X\_GUESS}-\text{X\_POINT}) \ (\text{Y\_GUESS}-\text{Y\_POINT}) \ (\text{Z\_GUESS}-\text{Z\_POINT}) \rfloor\end{aligned}$$

g) Check a tolerance value to see if convergence has been attained.

h) If the tolerance is not satisfied, repeat Steps b to g using the new guess for natural coordinates. Else, continue to Step i.

i) $\lfloor \textbf{P\_NAT} \rfloor = \lfloor \textbf{NAT\_GUESS} \rfloor$

Step 3: As described above, it is known that the line segment lies on one of the boundary faces of the element. Use the connectivity of the element and that of the boundary face, find the natural coordinate that has a constant value on this face. Assume that on this boundary face, T = -1. Then

$$\lfloor \textbf{P\_NAT} \rfloor = \lfloor R \quad S \quad -1 \rfloor$$

Thus, for the purposes of the least squares technique, it is assumed that the point lies on the face, T = -1, and the natural coordinates of the point in 2-D natural space are given by,

$$\lfloor \text{P\_NAT} \rfloor = \lfloor R \quad S \rfloor$$

This is equivalent to projecting the point (that lies inside or outside the element) onto the boundary surface.

Step 4: The information that now exists includes the natural coordinates of the point on the boundary face and the unaveraged responses at the nodes of that face. Shape functions are used to interpolate the nodal values to this point.

* If the face belongs to a 20-noded brick, the shape functions of a 4-noded quadrilateral element (Q4) are used.

* If the face belongs to a triangular facet of a 15-noded wedge finite element, the shape functions of a 3-noded triangular element (T3) are used.

Step 5: Check to see if the response value is required at other points along this line segment.

```
If <MORE_VALUES_REQUIRED> then
    go to Step 1
else
    exit from the algorithm
end if block
```

end algorithm

The response values extracted along the plot line are obtained in a manner consistent with the formulation of the finite elements along which it lies. As the stresses and strains are functions of the derivatives of the displacements, the interpolants used are the shape functions of elements one order lower. No averaging is done at inter-element boundaries. Figure 4.1 shows the boundary surface of a finite element mesh used to model an arch dam. A1 and B1 are two points on the curved surface of the dam. The line joining these points (along the surface) is the user-defined plot line along which responses are to be plotted. Figure 4.2 is the plot of Epsilon Z (the strain in the global Z direction, the "gravity direction" for the dam) from A1 to B1. The vertical "jumps" seen in the plot are the inter-element discontinuities. The analyst can visually compare the magnitude of these "jumps" with the "average" value of the response in the region. It is this comparison, and not the absolute values of the "jumps" that is useful as a sensor of local mesh quality.

4.4.3.2.2 Interactive Tools Implemented in POSTPRO3D

Three new menu pages have been designed and implemented in POSTPRO3D to facilitate interactive mesh quality sensing. The analyst can move rapidly between these menu pages and extract 2-D plots of unaveraged responses along any line on the surface of a structure. Strain energy density can be contoured on the surface.

Figure 4.3 is the layout of the MESH QUALITY PAGE (MQP). On this page, the analyst interactively selects a scalar mesh quality sensor -- strain energy density or energy error quantity -- and displays its distribution using smoothed, discrete contours on the surface of the structure. The functions on the MQP are almost exactly the same as those on the RESPONSE VIEWING PAGE (RVP). The analyst has easy access to the Load Specification Page and may increment or decrement load steps in a nonlinear analysis.

Figure 4.4 is the layout of the SELECT PLOT LINE PAGE (SPLP). On this page the analyst defines a cutting plane by selecting three non-collinear points in space. (These points do not need to lie on the surface of the structure.) The cutting plane is used to find the plot loops on the surface of the structure (these represent the intersection of the cutting plane and the surface.) Flexibility and speed of execution are emphased. Both tablet/pen-controlled and terminal data input modes are allowed. The analyst can also access the Coordinates Page (to get the world coordinates of nodes), or the Rotate Page (to change the view specification of the structure, background color, etc.).

On extracting the plot loops the analyst must enter the SELECT LINE END POINTS PAGE (SLEP) -- Figure 4.5 shows a schematic layout. On this page, the analyst interactively defines the plot line on the surface of the structure along which 2-D plots of unsmoothed responses are to be generated. The plot line can be on any of the plot loops.

Figure 4.6 is the layout of the PLOT RESPONSES PAGE (PRP). The analyst moves down to this page to select and plot responses along the plot line defined in the SPLP. The analyst can select strain energy density, or any of the Cartesian or principal stress and strain quantities to be plotted. The increment (or decrement) feature allows "stepping through" a nonlinear analysis with automatic re-plotting. Color contours of smoothed responses can also be displayed. The analyst must return to the SPLP to be able to change the end points of the plot line or to define a new plane for plot loops extraction.

Extraction of Plot Loops

The definition of a plot line on the surface of an arbitrary 3-dimensional solid appears deceptively simple. Knowledge of the end points of the plot line, and a constraint that restricts its length to the shortest possible, are sufficient data to extract the line in space. But this is an expensive, nonlinear optimization problem which is not interactive for arbitrary geometries. To make the extraction process simpler, the analyst is required to input additional data.

The analyst defines a "cutting plane" using three points that are not collinear on the surface. These points are interactively input by either using the hit-testing process (tablet/pen driven) or by typing in the world coordinates at the terminal. The hit-testing procedure finds the closest node on the surface. Flexibility is emphasied -- the analyst can change one or all the selected points before initiating the extraction of the plot loops. To facilitate terminal input of coordinates data, the Coordinates Page can be accessed from the Select Plot Line Page (SPLP). The analyst can obtain coordinates information in this page.

A plot loop is a closed, 3-D loop that lies both on the defined cutting plane and on the surface of the structure. More than one such loop can exist for a particular orientation of a cutting plane through a structure with "holes." Each loop is a collection of connected 3-D line segments in space. Each line segment is the intersection between a boundary face and the cutting plane. Algorithms have been designed to extract these plot loops rapidly for arbitrary orientations of cutting planes and arbitrary geometries. Figure 4.7 shows the simulation of a wrench. The analyst has selected three points that define the cutting plane. The program extracts and displays the two plot loops shown in the figure. The analyst can interactively select the end points of a plot line on any of the extracted loops. Figure 4.8 is the plot of unaveraged strain energy density along a user-defined plot line on plot loop 1. Figure 4.9 is the plot of unaveraged strain energy density along a user-defined plot line on plot loop 2. The vertical "jumps" represent inter-element discontinuities of the plotted response.

Algorithm For the Extraction of Line Segments:

The algorithm used to extract line segments is described below. (Parts of the algorithm are explained using Fortran-like pseudo-code.)

The input data to the algorithm includes: (a) Connectivities of the boundary faces -- the node numbers are used to get the coordinates in untransformed world space. (b) Screen coordinates of the 3 user-defined points on the surface of the structure.

```
begin algorithm

    Step 1: Check to see if the 3 points are collinear.

        If <NOT_COLLINEAR> then
            go to Step 2
        else if <COLLINEAR> then
            inform user
            abort extraction process
            wait for next user command
        end if block
```

Step 2: Use the hit-testing algorithms (Section 2.2.3.1) to compute the untransformed world coordinates of the 3 points.

Step 3: Compute the coefficients, $\lfloor A\ B\ C\ D \rfloor$, of the plane equation on which these points lie,

$$A_1x + B_1y + C_1z + D_1 = 0.0 \tag{4.16}$$

Step 4: Step 4 is performed for each boundary face.

```
do BOUND_FACE_COUNT ← 1, NUM_BOUND_FACES
```

a) Obtain the untransformed world coordinates of the nodes on the boundary face.

b) Check to see if the cutting plane cuts the face. (This condition is passed even if the plane lies along an edge of the face.)

```
If <FACE_IS_CUT> then
   proceed with the following steps
else
   go to the end of this processing loop
end if block
```

c) Compute the plane equation that defines the boundary face,

$$A_2x + B_2y + C_2z + D_2 = 0.0 \tag{4.17}$$

```
        d) Compute the coordinates of the end points of the line
           segment -- these are computed as the intersection of the 2
           planes given by equations (4.16) and (4.17).

    end do block

end algorithm
```

The information that has now been extracted from the raw input data is the spatial locations of a set of 3-D line segments. At this point in the extraction process, no information about the connectivity of these segments exists. The next task is to sort these line segments into coherant, connected, 3-D loops (or a single loop).

Algorithm For Sorting the Line Segments into Coherent Plot Loops:

The algorithm used to sort the line segments is described below. (Parts of the algorithm are explained using Fortran-like pseudo-code.)

The "matching" of these line segments in space to form closed plot loops can be achieved in a number of ways. Simple sorting procedures can be used to reduce the CPU time taken to match the segments in space. One such sorting procedure is briefly outlined below.

```
begin algorithm

    Step 1: The "global spread" of the coordinates of the end points of
    the segments in each coordinate direction is computed,

      $\lfloor SPREAD \rfloor = \lfloor (MAX\_X - MIN\_X)\ (MAX\_Y - MIN\_Y)\ (MAX\_Z - MIN\_Z) \rfloor$   (4.18)

        where,
          MAX_M = max of (coordinate values of all end points in
                                global M direction )
          MIN_M = min of (coordinate values of all end points in
                                global M direction )

    Step 2: For highest efficiency, the direction of "maximum spread"
    is selected as the first sort direction.
       Assume that this is the global X direction.

    Step 3: Sort the line segments in ascending order of global X
    coordinate values.
      Care is taken to eliminate identical segments.  It is possible to
      have these segments if the cutting plane passes along edges shared
      by pairs of boundary faces.

    Step 4: Now use the X-sorted list to match Y and Z coordinate
    values.

end algorithm
```

For the sizes of problems that are currently analysed, the maximum number of line segments does not exceed a few hundred. It is found that for sorts of this size, the "brute force" method suffices to achieve interactive speeds. This is the technique that is implemented in POSTPRO3D.

<u>Algorithm to Match A List of Line Segments to Form Plot Loops</u>:

One of the problems encountered in this algorithm is "matching" two points in 3-D space. Round-off and truncation, and the methods used to compute the coordinates of the two points affect the accuracy of the "match." A suitable tolerance -- to account for these factors -- was arrived at by trial and error.

The input data to the algorithm includes: (a) The number of line segments that need to be sorted, NUM_SEGMENTS. (b) Untransformed world coordinates of the end points of the set of line segments obtained from the first part of the extraction process, SEG_COORDS.

```
begin algorithm

    Step 1: Start by using the first unprocessed segment.

        MATCH_SEGMENT ← <FIRST_UNPROCESSED_SEGMENT>

    Step 2: It is required to find the segment that matches the second
    end point of MATCH_SEGMENT.
```

```
do SEARCH_SEGMENT ← 1, NUM_SEGMENTS
    Step 3: Check to see if this segment has already been processed
    or if this is the segment whose match is required to be found.

        If <PASSES_ABOVE_TESTS> then
            go to the end of this processing loop
        else
            proceed with Step 4
        end if block

    Step 4: Check to see if SEARCH_SEGMENT matches an end point of
    MATCH_SEGMENT. At this point a check is also made to detect and
    eliminate identical segments.

        if <MATCHED> then
            if <NOT_IDENTICAL> then
                MATCH_SEGMENT ← SEARCH_SEGMENT
                begin from Step 2 using the new MATCH_SEGMENT
            else
                eliminate SEARCH_SEGMENT from database
                go to the end of this processing loop
            end if block
        else
            go to the end of this processing loop
        end if block
end do block
```

```
Step 5: Check to see if the last matched segment "closes" on the
first segment of this loop.
    If <LOOP_CLOSES> then
        proceed to Step 6
    else
        proceed to Step 2 and find the next matching segment
    end if block

Step 6: Check to see if there are any remaining unprocessed
segments -- the case of multiple plot loops.

    if <ALL_SEGMENTS_PROCESSED> then
        exit the algorithm
    else
        repeat Steps 1 to 6
    end if block

end algorithm
```

The algorithm extracts one or more plot loops from the input data consisting of randomly oriented line segments. For the size of extraction problems tested to date (upto a few hundred line segments) execution time is rapid. Figure 4.10 shows an interactive session in the SELECT PLOT LINE PAGE. The problem being viewed is an arch dam. The analyst has selected three points on the surface of the structure (these are highlighted in the red overlay.) The plot loop is extracted and is drawn in the green overlay.

C-2

Additional Interactive Features

There are a number of additional interactive features that have been implemented into POSTPRO3D. Flexibility and speed of execution are emphasized.

The interactive step following the extraction of the plot loops is the selection of the end points of the plot line. The user can enter the coordinates of the desired end points at the terminal, or use the pen to "point" to the screen. The program finds the line segment end-point (at an element edge) that is closest to the "hit" point. The algorithms used in this process are straightforward. A box test around all line segments generally reduces the number to be processed down to a few segments. The closest segment end point is then easily computed. As in the definition of the cutting plane, flexibility is emphasized. The analyst can change the selection as often as desired before pushing the EXTRACT PLOT LINE button. Figure 4.11 shows the selection of the end points of the plot line on the plot loop shown in figure 4.10 -- the selected plot line is drawn in the blue overlay and the rest of the plot loop is in the green overlay.

The toggle function allows the analyst to view plotted information along the entire plot loop -- from A to B or from B to A (proceeding in the same direction). This function can be activated in both the SELECT PLOT LINE PAGE and the PLOT RESPONSES PAGE (PRP). The active plot line is rendered in the blue overlay, while the rest of the plot loop is in the green one.

Finally, to plot a response along the selected plot line, the analyst must go to the PRP. Figure 4.12 shows the table on the PRP that allows the analyst to choose the response for plotting along the plot line (shown drawn in the green overlay.)

In the case of nonlinear, load-step or time-step analyses, the analyst can "step" through the various load-steps using the Increment feature. Assume that the major principal stress is plotted for load-step **n**. The Increment feature clears the old plot, swaps the new response set into the working database (set **(n - 1)** or **(n + 1)** depending on whether the - or + side of Increment is activated), and plots the major principal stress for the new set along the same plot line.

To obtain a hard-copy of a plot, the CREATE PLOT FILE option creates a file containing the relevant plot information, including the plot title, abcissa and ordinate values, and labels. This file is read into a graphing program run on a vector device (e.g. an Evans and Sutherland Picture System). The graphs shown in this thesis are created in this manner.

It is useful to be able to view color contours of smoothed responses and 2-D plots of unsmoothed responses simultaneously. The combination of the visual picture of the behavior of the structure and the extent of the response discontinuities in critical regions of the structure helps in the qualitative evaluation of the mesh. The analyst has direct access to the RESPONSE VIEWING PAGE for contouring.

4.4.3.2.3 Example Problems

Two 3-D problems are examined to demonstrate the use of 2-D plots of unaveraged responses as a mesh quality sensor. Sampling of response values is done at 5 points along each line segment (dividing each into four equal parts.) 20-noded isoparametric brick finite elements are used for each of the analyses discussed below.

1) ANGLE: Figure 4.13 shows a finite element mesh used in the analysis of an angle. Figure 4.14 shows the boundary conditions applied to the structure. Edge AB has displacements along the global X and Y directions fixed, and along the Z direction, free. A surface pressure is applied to the opposite face in the negative X direction. The section parallel to X-Y containing point B is a plane of symmetry (displacements are fixed in the Z direction, and free in the X and Y.) The analysis is performed using material nonlinear (elastic-perfectly plastic) constitutive equations for the finite elements. There are 4 load steps in the analysis. Step 0 is a linear elastic analysis -- the results are scaled to induce incipient yield at one (or more) Gauss points. Steps 1 to 3 are nonlinear load steps, each with increments of 20% of the initial yield load. The yeild criterion used is a Von Mises criterion with a yield stress of 60.0 units. However, values higher than 60.0 units may be obtained in contours or plots of the Von Mises stress because results are extrapolated from the Gauss points at which the yield criterion is applied.

Two analyses are performed -- the second has a more refined finite element mesh than the first. Figures 4.15 and 4.16 show plan and sectional views of the coarser of the two meshes containing 72 elements and 521 nodes. Figures 4.17 and 4.18 show plan and sectional views of the refined mesh containing 440 elements and 2477 nodes. A1B1, figure 4.17, is the first plot line used for plotting unaveraged responses while A2B2, figure 4.18, along the free edge is the second plot line. The coordinates of the end points of A1B1 and A2B2 are the same for each of the two meshes. Figure 4.19 shows the refined finite element mesh in the green overlay and the displaced mesh in the blue overlay.

Figures 4.20 to 4.23 are the plots of the unaveraged first stress invariant (INV-I -- hydrostatic pressure) along A1B1 for load steps 0 to 3. The observations drawn from these plots include: (a) ANGLE-fine shows lower INV-I values than ANGLE-coarse. As the mesh is refined, the structure becomes more flexible, reducing the stresses, while deformation increases towards the exact solution. (b) The region of the largest difference between the responses of the two meshes is also the region of the highest INV-I and greatest INV-I gradient. (c) In general, ANGLE-coarse has more prominent inter-element discontinuities and kinks (differences in slopes at inter-element boundaries). (d) As plasticity spreads through the structure -- highest for step 3 -- the kinks and discontinuities become more pronounced for ANGLE-coarse.

Figures 4.24 to 4.27 are the plots of unaveraged INV-I along A2B2 for load steps 0 to 3. The observations drawn from these plots include: (a) As the mesh is refined, INV-I reduces. (b) For the linear elastic analysis (figure 4.24) the central surface of the angle is approximately the neutral axis. ANGLE-coarse has only two elements through the thickness. Thus the inter-element boundary lies close to the neutral axis. At this boundary, the analysis can be considered to be "equally inaccurate" on either side; thus, no inter-element "jump" is seen, but there is a "kink." As the mesh is refined in this area (ANGLE-fine has 5 elements through the thickness) the kink vanishes. (c) The elements closer to A2 begin to plastify first and the plasticity spreads towards the neutrl axis. As this occurs, the difference between the results increases. In ANGLE-coarse, the single element in this region attempts to partially plastify, but the formulation of the element does not allow this -- instead, the complete element plastifies (figure 4.27). As the neutral axis moves towards the top surface, away from the mid-surface of the angle, inter-element "jumps" begin to appear (figures 4.26 and 4.27 -- load steps 2 and 3, respectively).

Plots of other responses, such as effective stress and major principal stress, have also been examined along the same plot lines. These yield essentially the same observations as above.

2) PLATE WITH A HOLE: The second example problem is a plate with a central hole. Figure 4.28 shows the geometry and loading. Face ABCD has a tensile pressure in the X direction, and an out-of-plane moment applied to it. The applied surface tractions on ABCD are shown in the figure. Face EFGH is a plane of symmetry - displacements in the X direction are fixed, and in the Y and Z directions, free. For stability, edge EH (not including the boundary of the hole) is fixed in the Z direction. The analyses are linear elastic.

Figure 4.29 is the boundary of the coarser of the two finite element meshes used to analyze the problem -- PLATE-coarse. The finite element mesh consists of 72 elements, 447 nodes, and 3 elements through the thickness. Figure 4.30 is the boundary of the refined mesh -- PLATE-fine. This mesh consists of 1280 elements, 6293 nodes, and 8 elements through the thickness of the plate.

Figure 4.31 shows the color contours of SED on the surface of a subobject of PLATE-coarse (a few elements around the hole are extracted to create this subobject). A similar subobject is extracted from PLATE-fine; figure 4.32 shows the contours of SED on its surface. As the mesh is refined, the volume of material with high SED and SED gradients decreases. This arises directly from the fact that SED tends to converge from above the exact solution whereas displacements tend to converge from below. The neutral axis is clearly seen in the refined analysis as a color band representing values of SED near zero.

Figure 4.33 shows the locations of the two plot lines -- A1B1 and A2B2 -- along which unaveraged responses are plotted. Figures 4.34 to 4.36 are plots of SED, effective stress, and sigma-x (the stress in the global X direction) along A1B1. The observations drawn from these plots include: (a) The "kinks" and inter-element discontinuities occur in the region of high SED and SED gradient. (b) PLATE-coarse has 3 elements through the thickness of the plate. The element that contains point A1 spans a third of the thickness. The boundary that this element shares with the next one (along A1B1) is far from the region of high SED and, hence, almost no kink or jump is seen here. On the other hand, PLATE-fine has 8 elements through the thickness. Thus the first inter-element boundary (along A1B1) is close to A1. Hence, the solution of the refined mesh shows jumps at this boundary. (c) Through the rest of the thickness of the plate, a fairly good solution is obtained even with the coarser mesh.

Figures 4.37 to 4.39 are plots of SED, effective stress, and sigma_x along A2B2. This plot line is far from the concentrated region of high SED and SED gradients at the corner of the hole. The problems associated with the location of the first plot line (A1B1) are avoided here. The observations (a), (b), and (c) made on the plots along A1B1 for the ANGLE problem can also be made in this case. Again, the plots are seen to be symmetric about the center of the plot line. A sharp kink is seen at this point for PLATE-coarse. PLATE-fine attempts to equalize response slopes on either side of this plane of symmetry. Near the top and bottom edges of the plate (at A2 and B2) 3-D edge effects and the proximity of the corners create discontinuities in both the fine and coarse analyses; but these are smaller for the fine analysis. It is evident that refinement near the corner is necessary to capture these edge effects more accurately.

4.4.3.3 Summary

A number of heuristic conclusions are drawn from the observations made above, and from experience with a variety of small and large problems analyzed with the displacement method. In general, 2-D plots of unaveraged responses on the surface of structures can be used as a qualitative indication of the validity of the analysis under review. They appear to exhibit local stability in the senses that discontinuities tend to vanish with local convergence and that the addition of degrees of freedom does not result in a change in the behavior of the sensor in regions that have attained convergence [36]. The effectiveness of this sensor in an interactive computer graphics postprocessing environment has been demonstrated, wherein visual explorations can be performed rapidly and subjective evaluations quickly achieved. The subjectivity is useful because the engineer can choose to accept or reject apparent discretization errors provided convergence is evident in the particular zones of interest. Jumps and kinks in low-stress zones can be ignored; these may be quantitatively large in a local sense but trivial in the larger context.

A flexible, interactive environment has been developed in POSTPRO3D for the rapid extraction of 2-D plots along any line on the surface of a structure. The emphasis is on speed of execution and flexibility rather than on the use of the most accurate method for the extraction of values along plot lines. The interpretation of the plots is subjective; hence the desired accuracy must be viewed in this light. More "exact" extraction of values along a plot line will render the process non-interactive in the case of general, curved 3-D continua.

There are cases where caution must be exercised in interpreting the 2-D plots from the point of view of the quality of the analysis. (a) If plots of unaveraged responses along many directions on the surface of a structure are found to be smooth, then the analysis can be said to have attained convergence. Isolated cases of smooth plots, or plots with discontinuities and kinks cannot be effectively used to gauge the quality -- local or global -- of the analysis. (b) At planes of symmetry, kinks and not inter-element jumps are seen. (c) At a corner or sharp edge of a structure (where only a single finite element exists the 2-D plot offers no answers to the question of quality as there is no inter-element boundary. (d) If a single finite element spans a large section of the structure encompassing a region of high SED gradient, nothing can be said about the quality of the analysis within this element -- see observation (b) on the plots along A1B1 for the PLATE problem discussed in the last section. Within the element, the extracted values of the response can only be smooth; this is the assumption on which the formulation of all finite elements is based. However, the existence of a high SED gradient within an element is a well established indicator that refinement is necessary in the region spanned by the element.

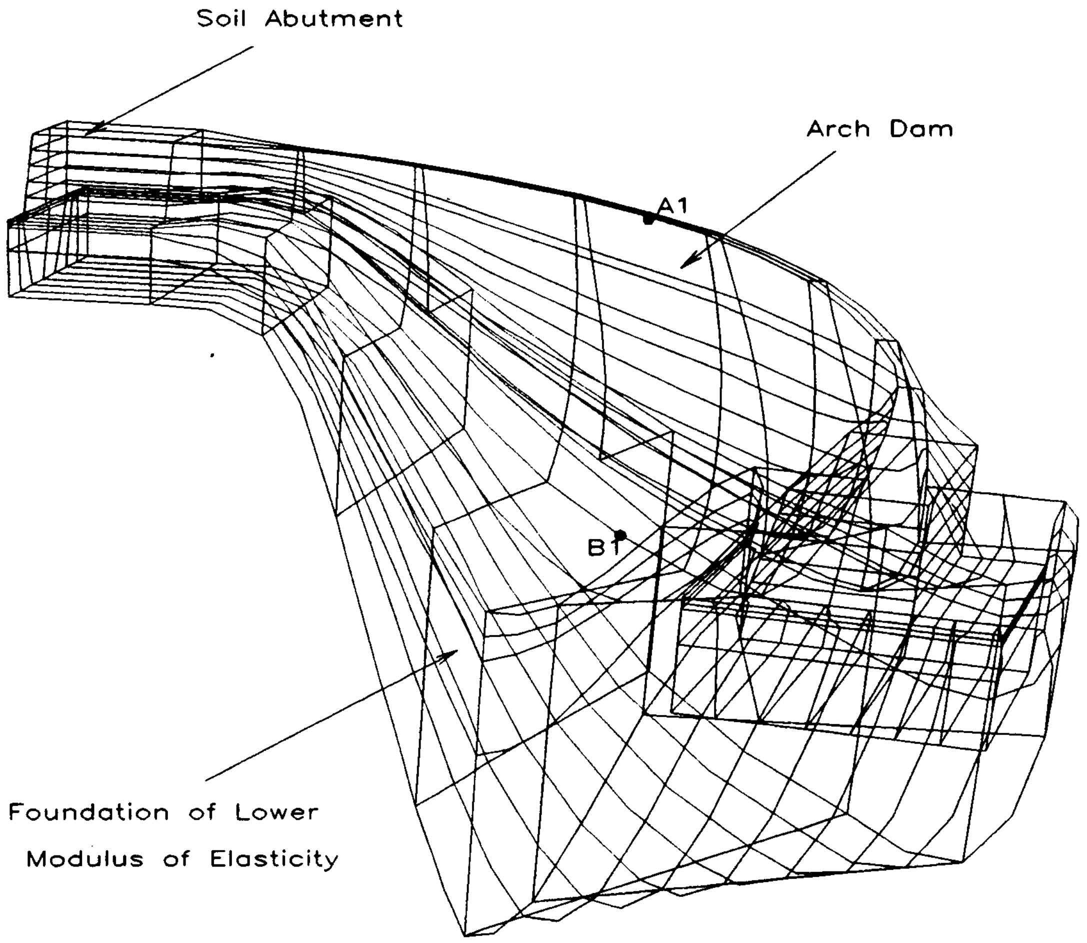


Figure 4.1 Arch Dam - Boundary of the finite element mesh

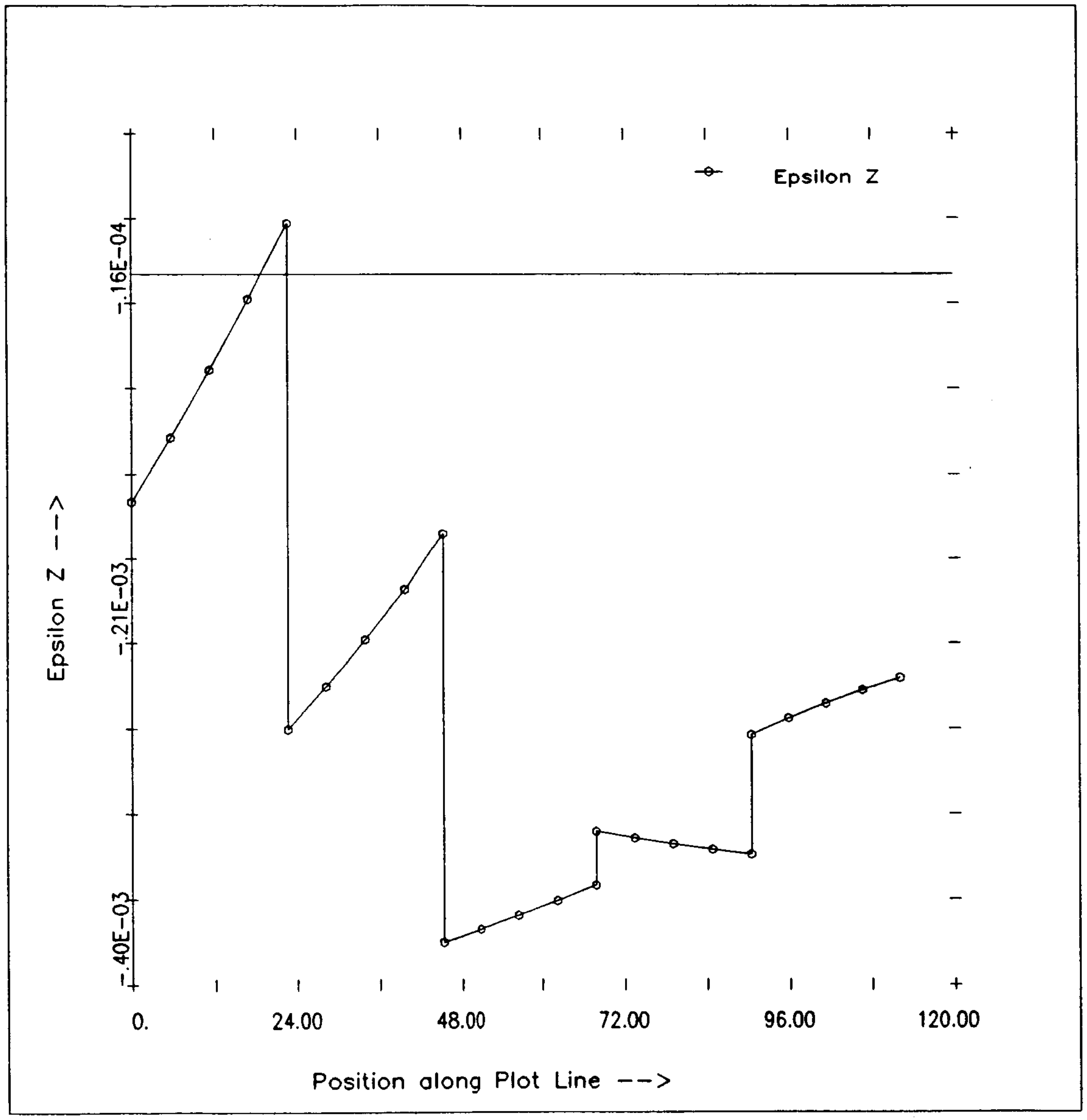


Figure 4.2 Arch Dam - Plot of Epsilon Z

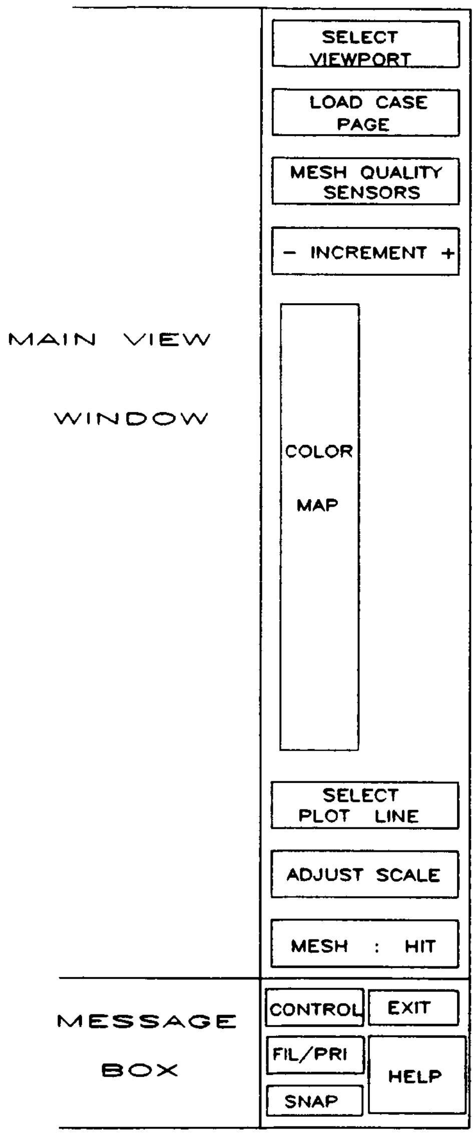


Figure 4.3 The MESH QUALITY menu page

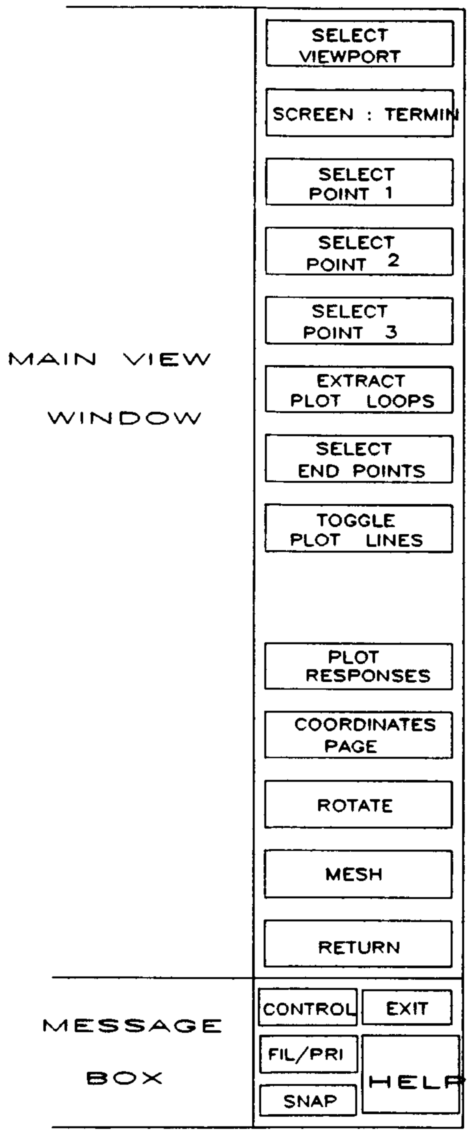


Figure 4.4 The SELECT PLOT LINE menu page

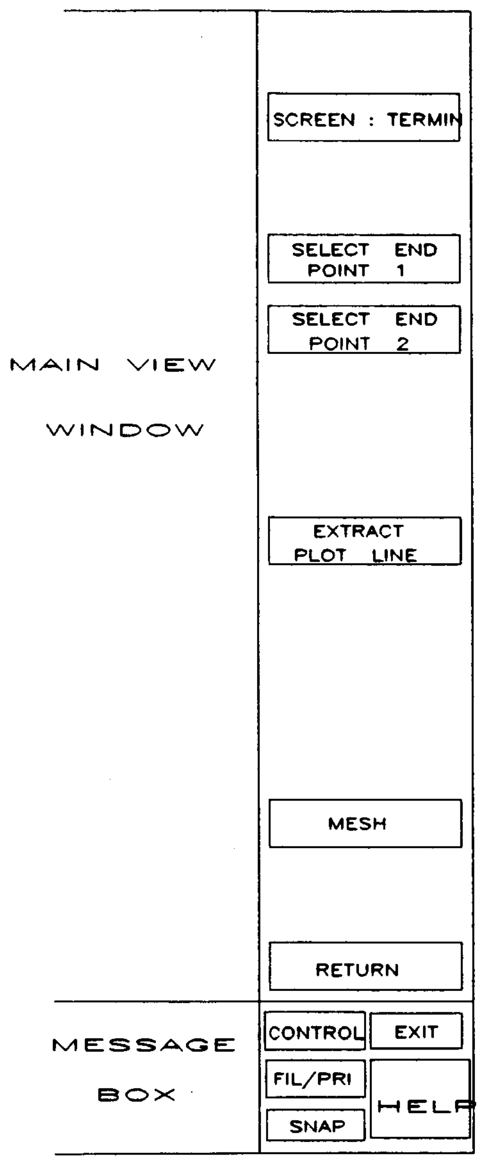


Figure 4.5 The SELECT LINE END POINTS menu page

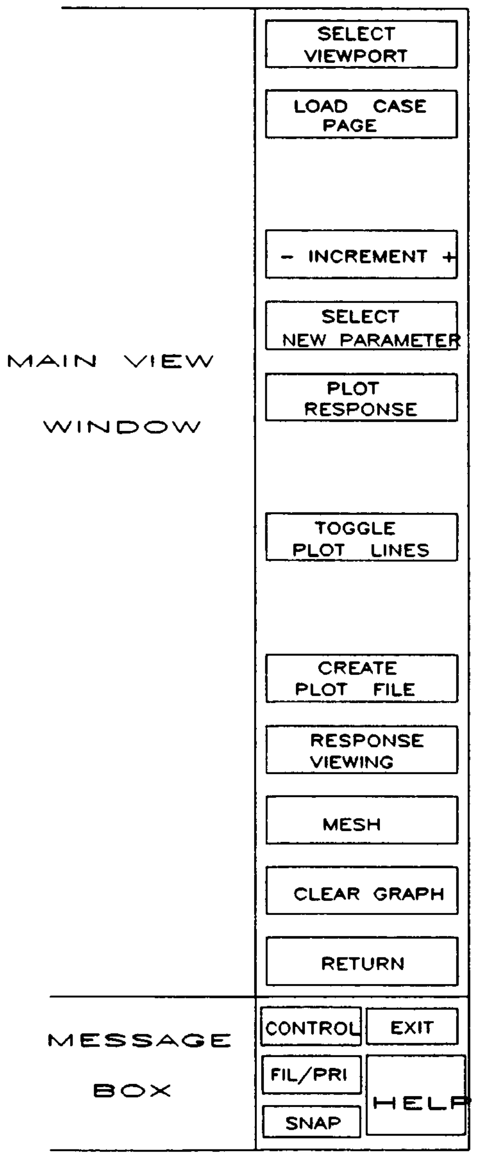
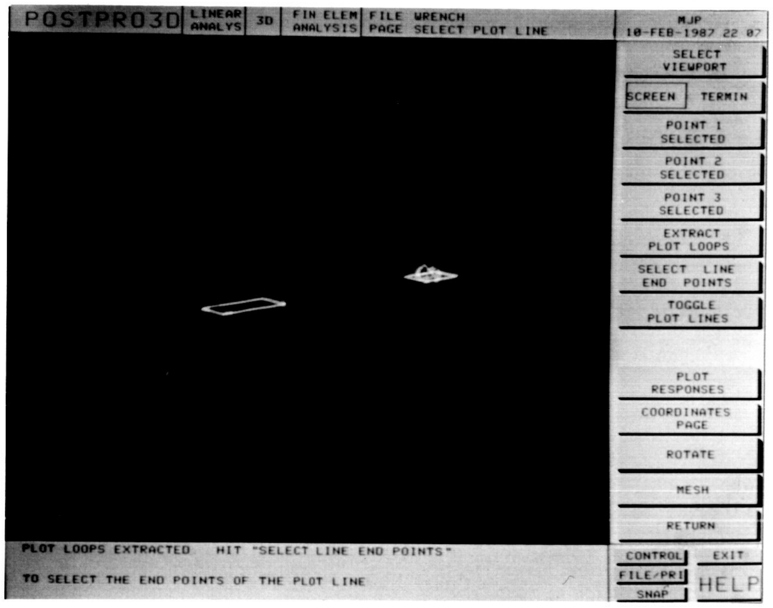


Figure 4.6 The PLOT RESPONSES menu page

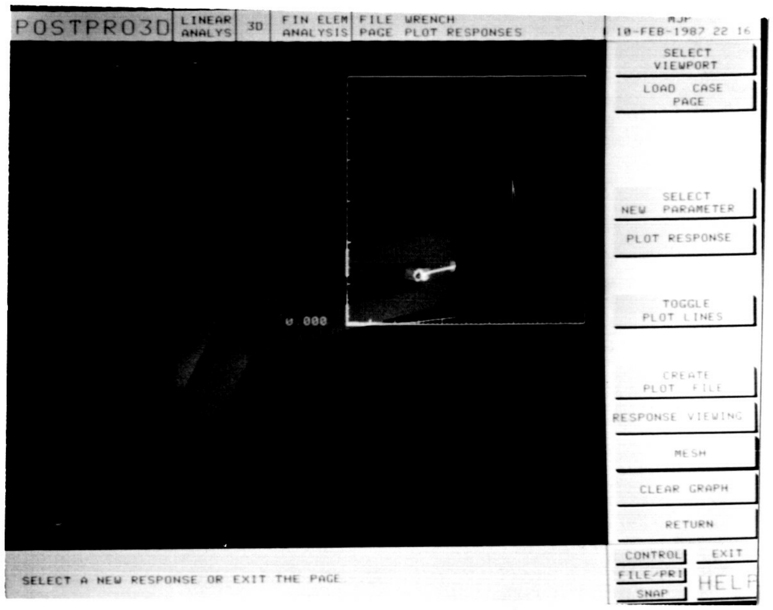
ORIGINAL PAGE
COLOR PHOTOGRAPH



PLOT LOOPS EXTRACTED   HIT "SELECT LINE END POINTS"

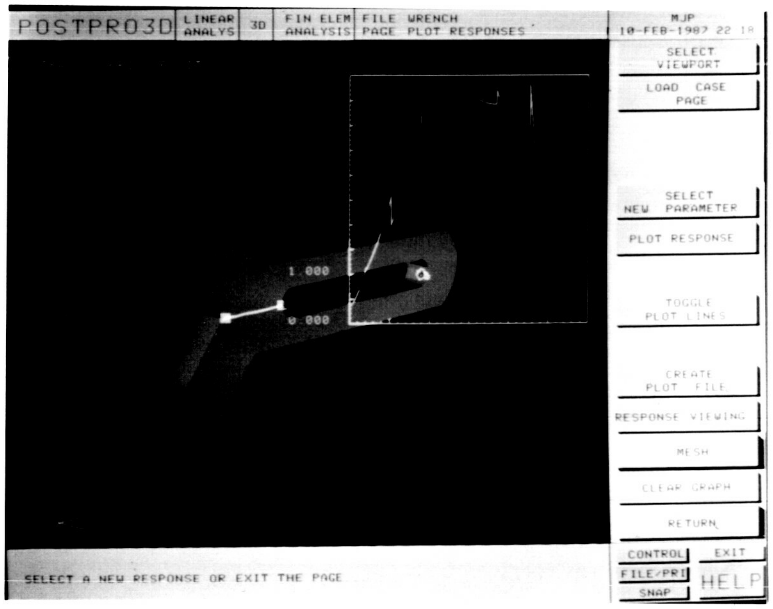
TO SELECT THE END POINTS OF THE PLOT LINE

Figure 4.7 Wrench:- Extraction of multiple plot loops



SELECT A NEW RESPONSE OR EXIT THE PAGE

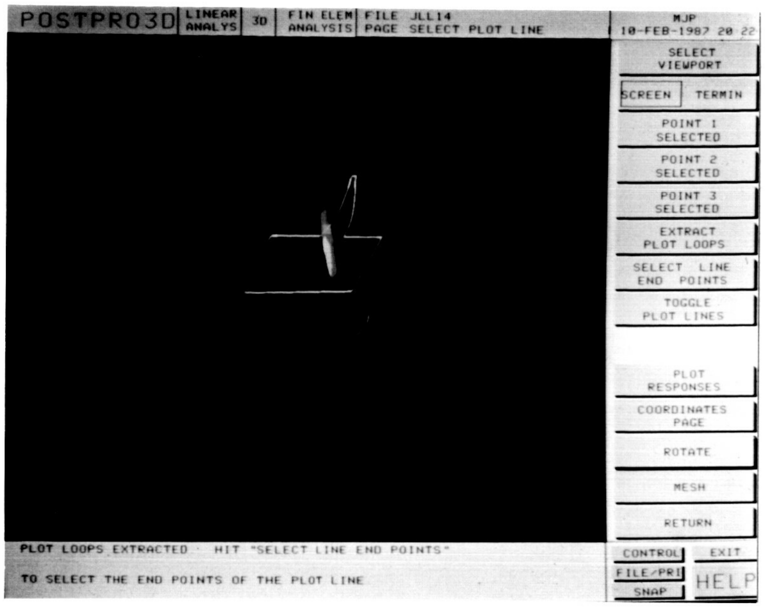
Figure 4.8 Wrench:- Plot of strain energy density along loop 1



SELECT A NEW RESPONSE OR EXIT THE PAGE

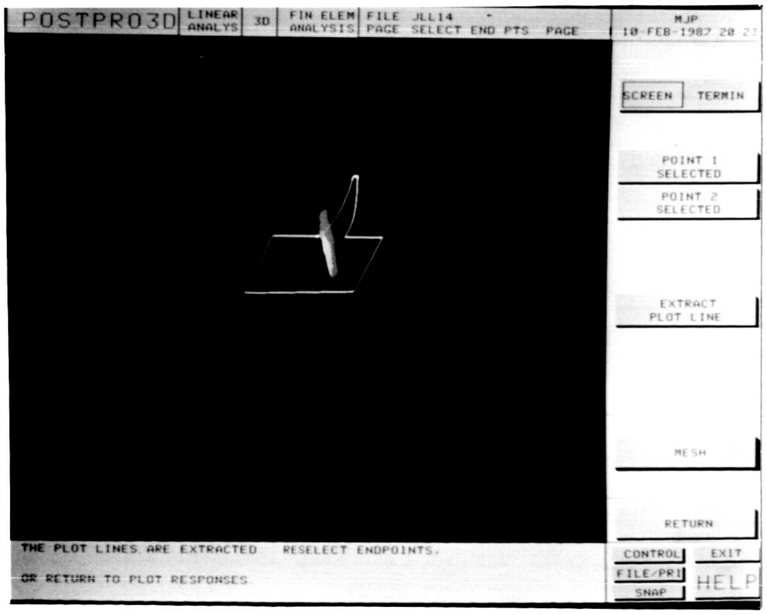
Figure 4.9 Wrench:- Plot of strain energy density along loop 2

ORIGINAL PAGE
COLOR PHOTOGRAPH



PLOT LOOPS EXTRACTED   HIT "SELECT LINE END POINTS"
TO SELECT THE END POINTS OF THE PLOT LINE

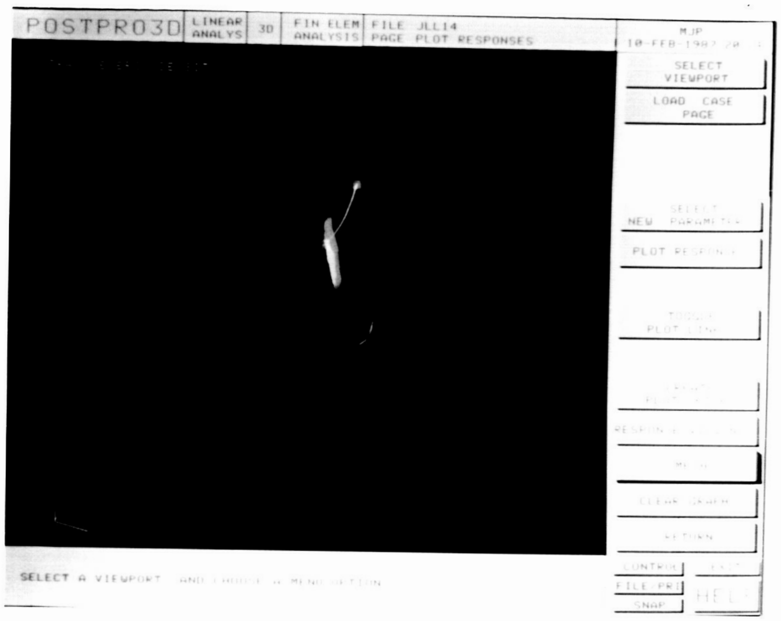
Figure 4.10 Arch dam:- Extraction of plot loops



THE PLOT LINES ARE EXTRACTED   RESELECT ENDPOINTS,
OR RETURN TO PLOT RESPONSES

Figure 4.11 Arch dam:- Selection of plot line end points

ORIGINAL PAGE
COLOR PHOTOGRAPH



SELECT A VIEWPORT AND CHOOSE A MENU OPTION

Figure 4.12 Arch dam:- Selection of responses for plotting

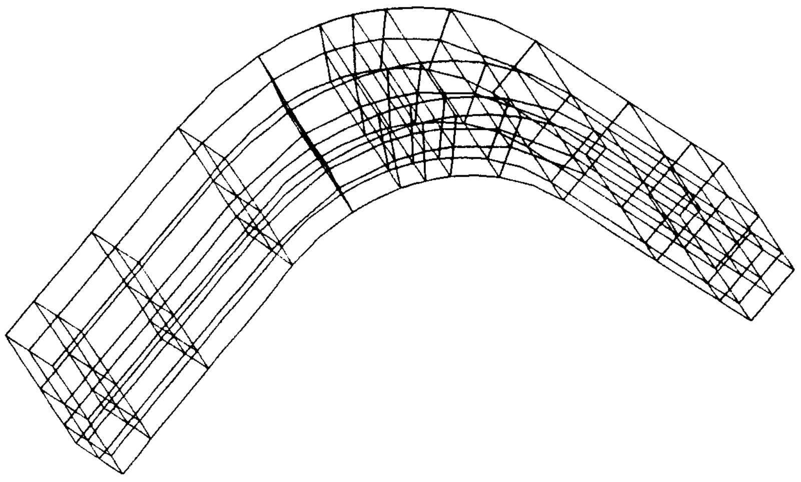


Figure 4.13 ANGLE:- Geometry

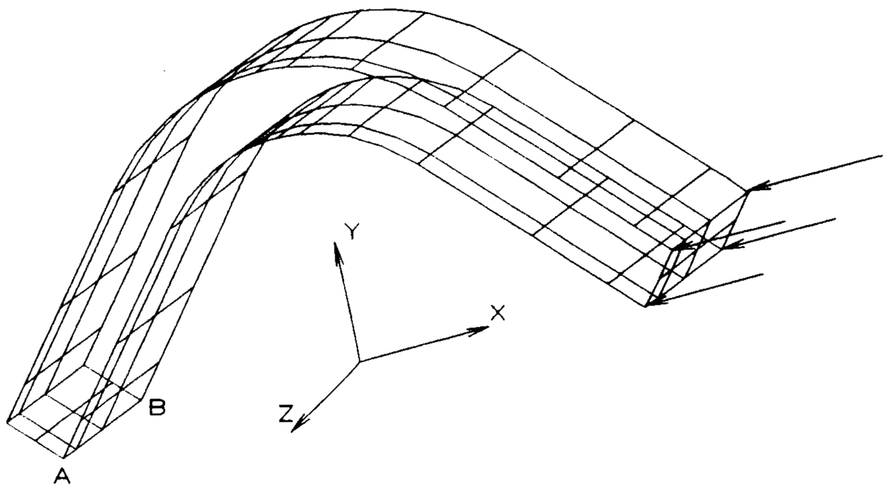


Figure 4.14 ANGLE:- Boundary conditions and loads

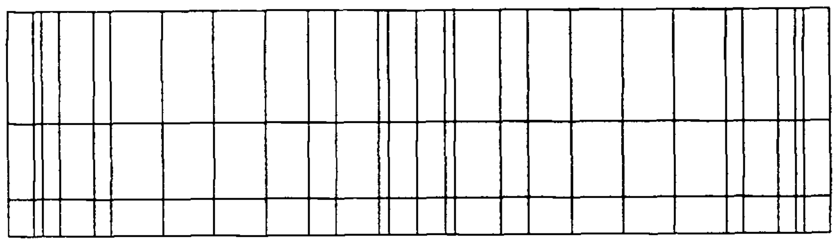


Figure 4.15 ANGLE-coarse:- Plan view

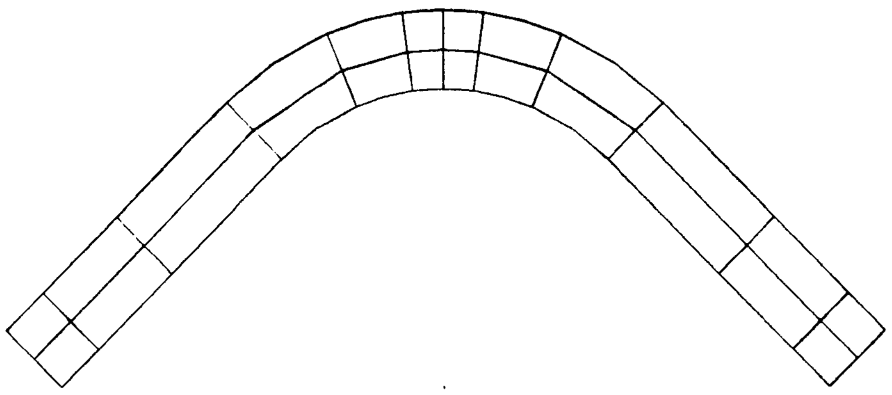


Figure 4.16 ANGLE-coarse:- Sectional view

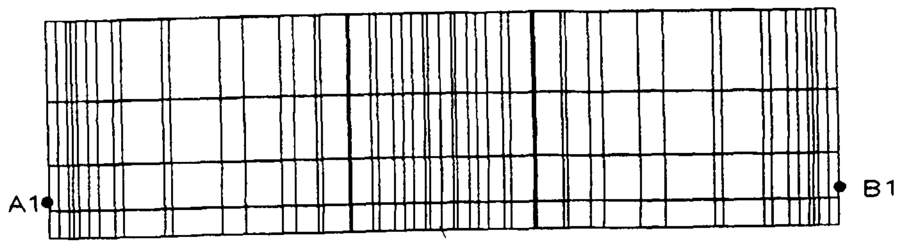


Figure 4.17 ANGLE-fine:- Plan view and plot line A1B1

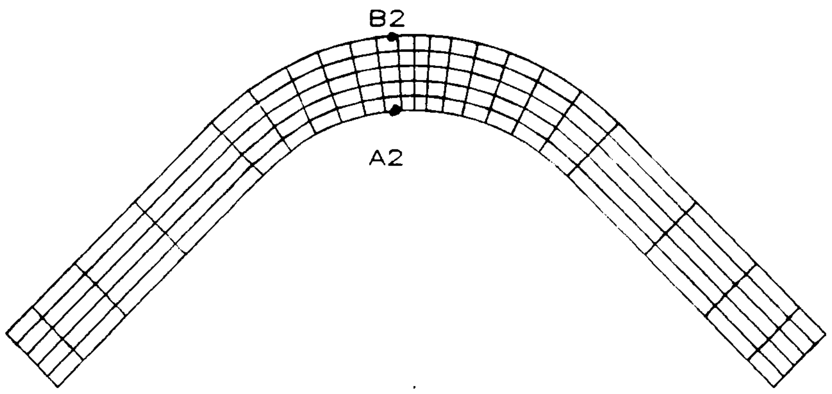
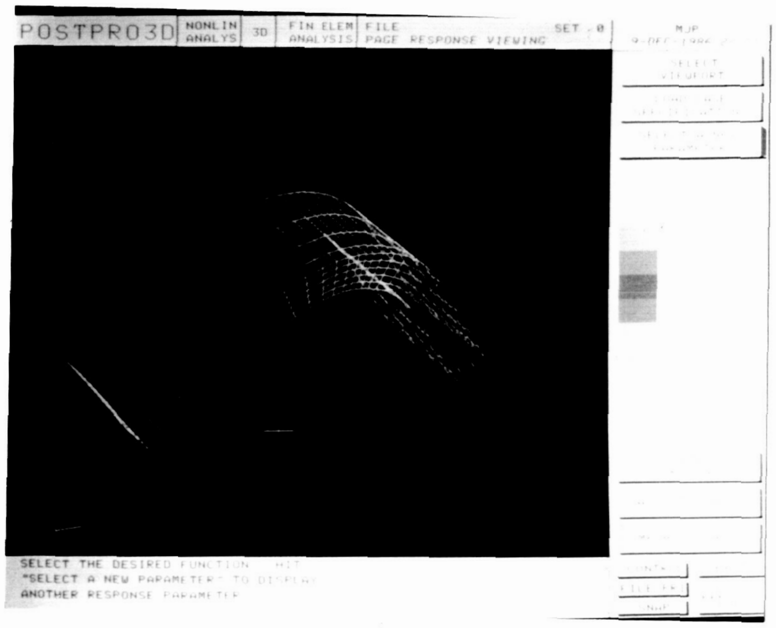


Figure 4.18 ANGLE-fine:- Sectional view and plot line A2B2



SELECT THE DESIRED FUNCTION HIT
"SELECT A NEW PARAMETER" TO DISPLAY
ANOTHER RESPONSE PARAMETER

Figure 4.19 ANGLE:- Finite element mesh and displaced shape

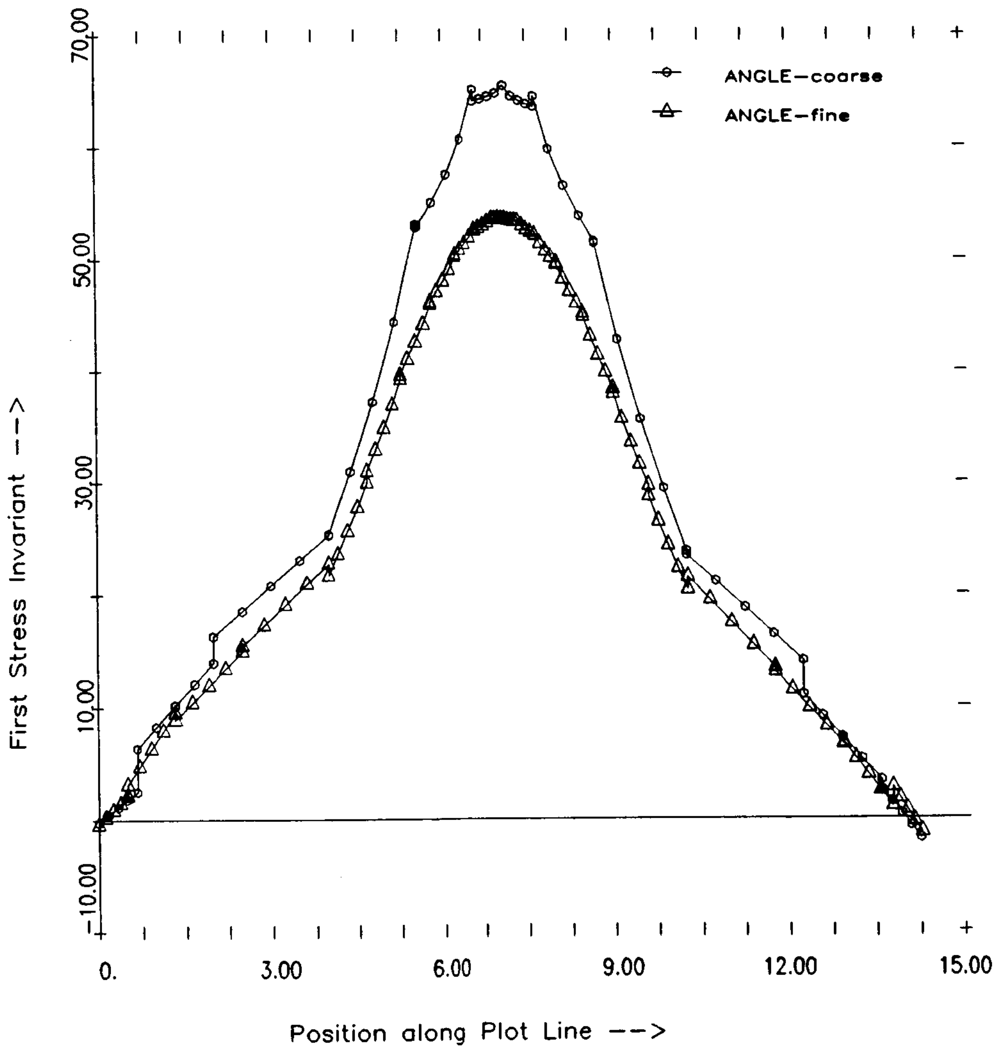


Figure 4.20 ANGLE:- Plot of INV-I along A1B1 - Load Step 0

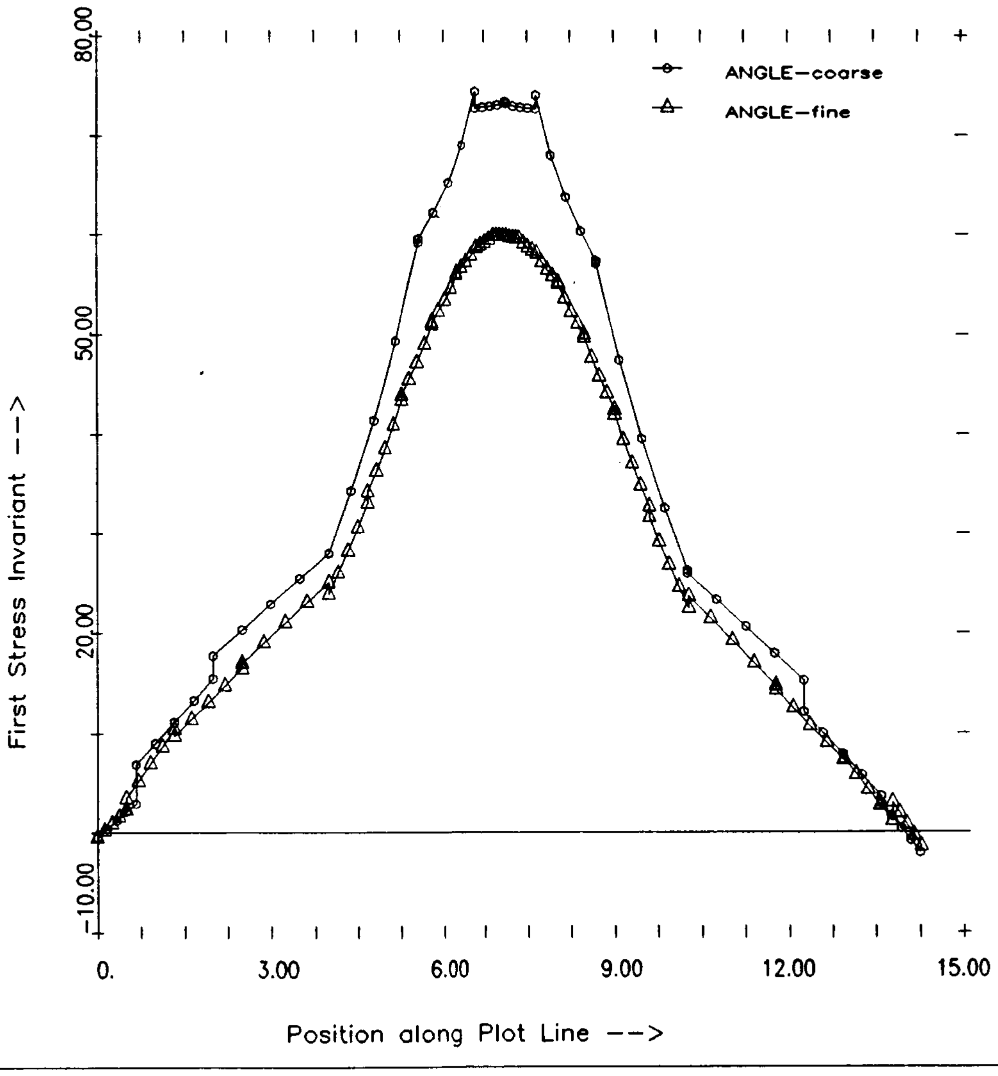


Figure 4.21 ANGLE:- Plot of INV-I along A1B1 - Load Step 1

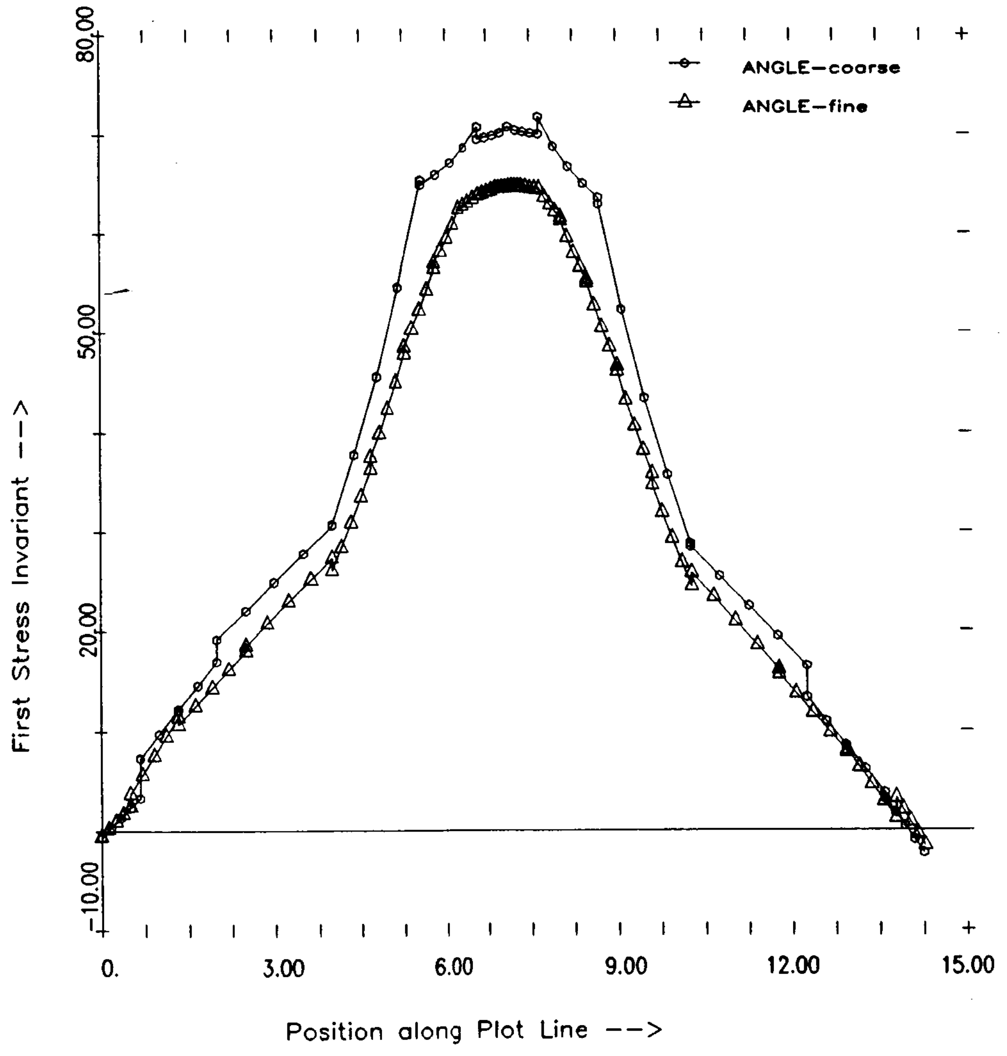


Figure 4.22 ANGLE:- Plot of INV-I along A1B1 - Load Step 2

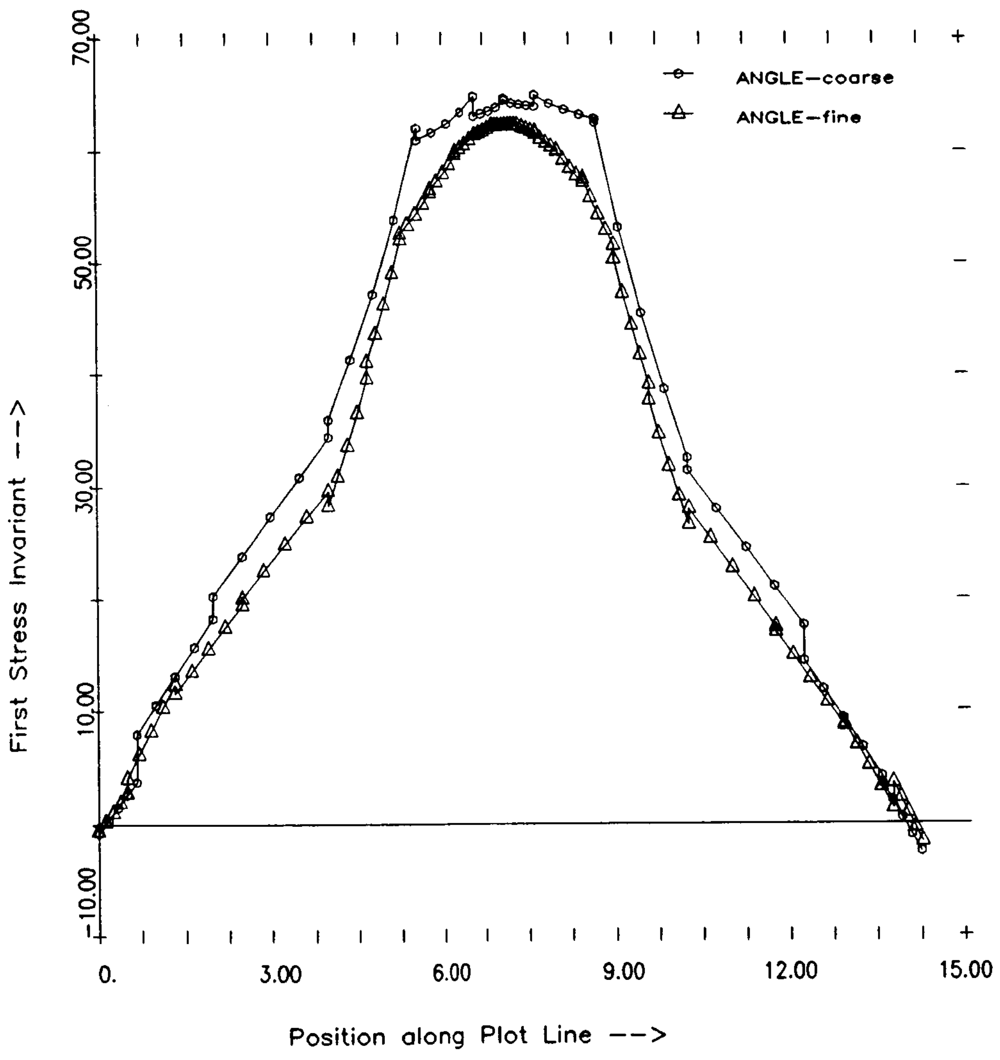


Figure 4.23 ANGLE:- Plot of INV-I along A1B1 - Load Step 3

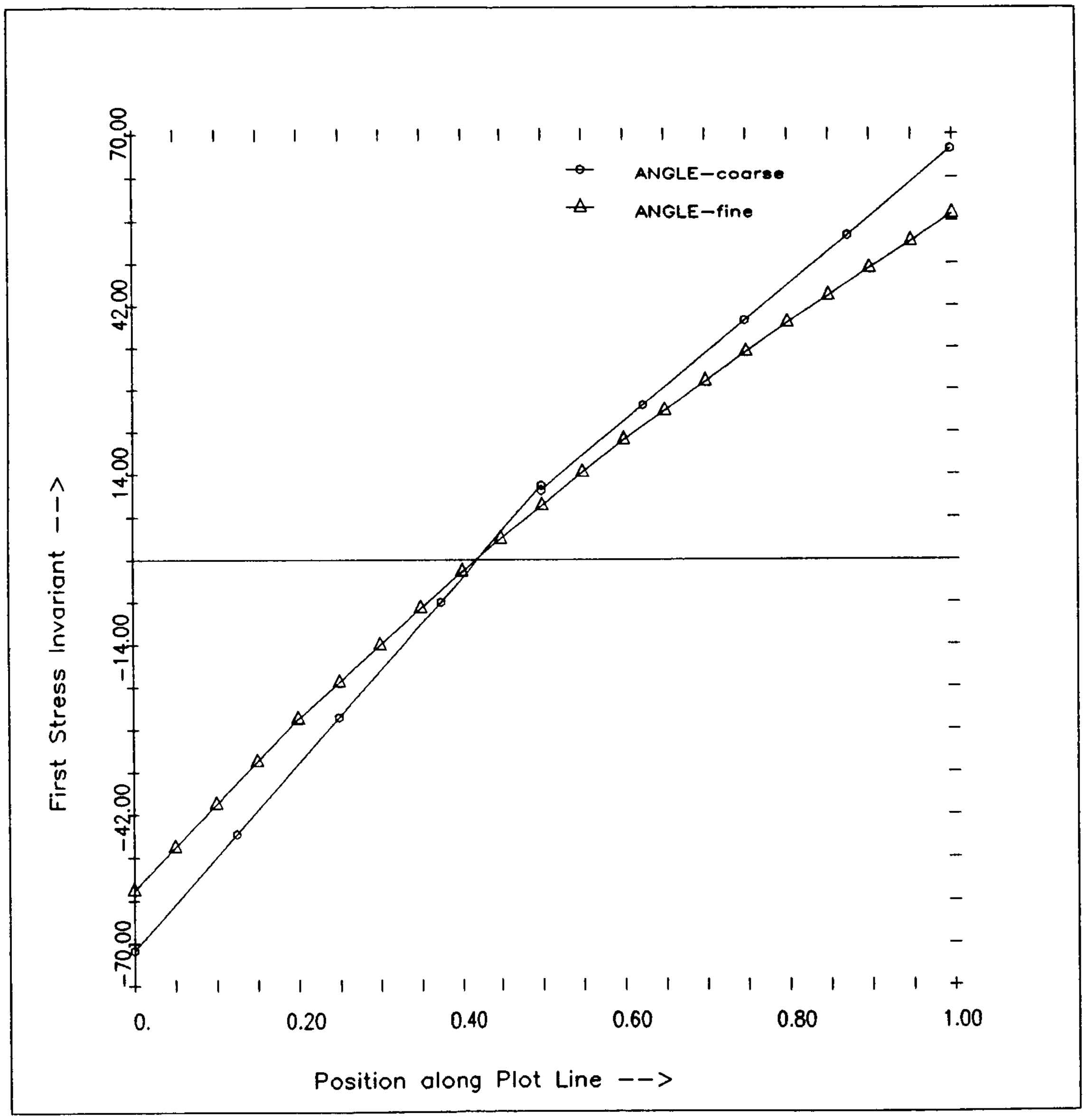


Figure 4.24 ANGLE:- Plot of INV-I along A2B2 - Load Step 0

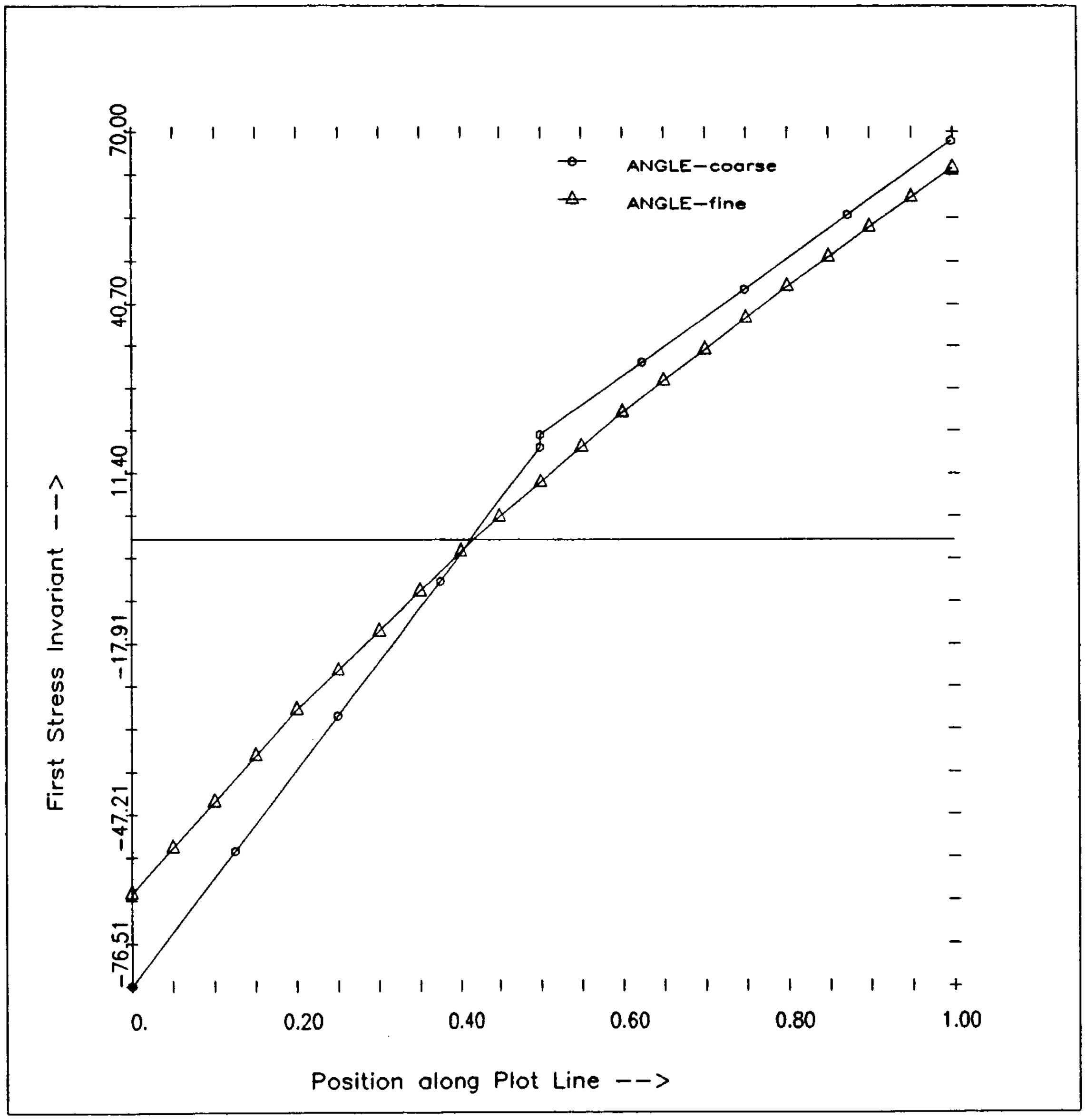


Figure 4.25 ANGLE:- Plot of INV-I along A2B2 - Load Step 1

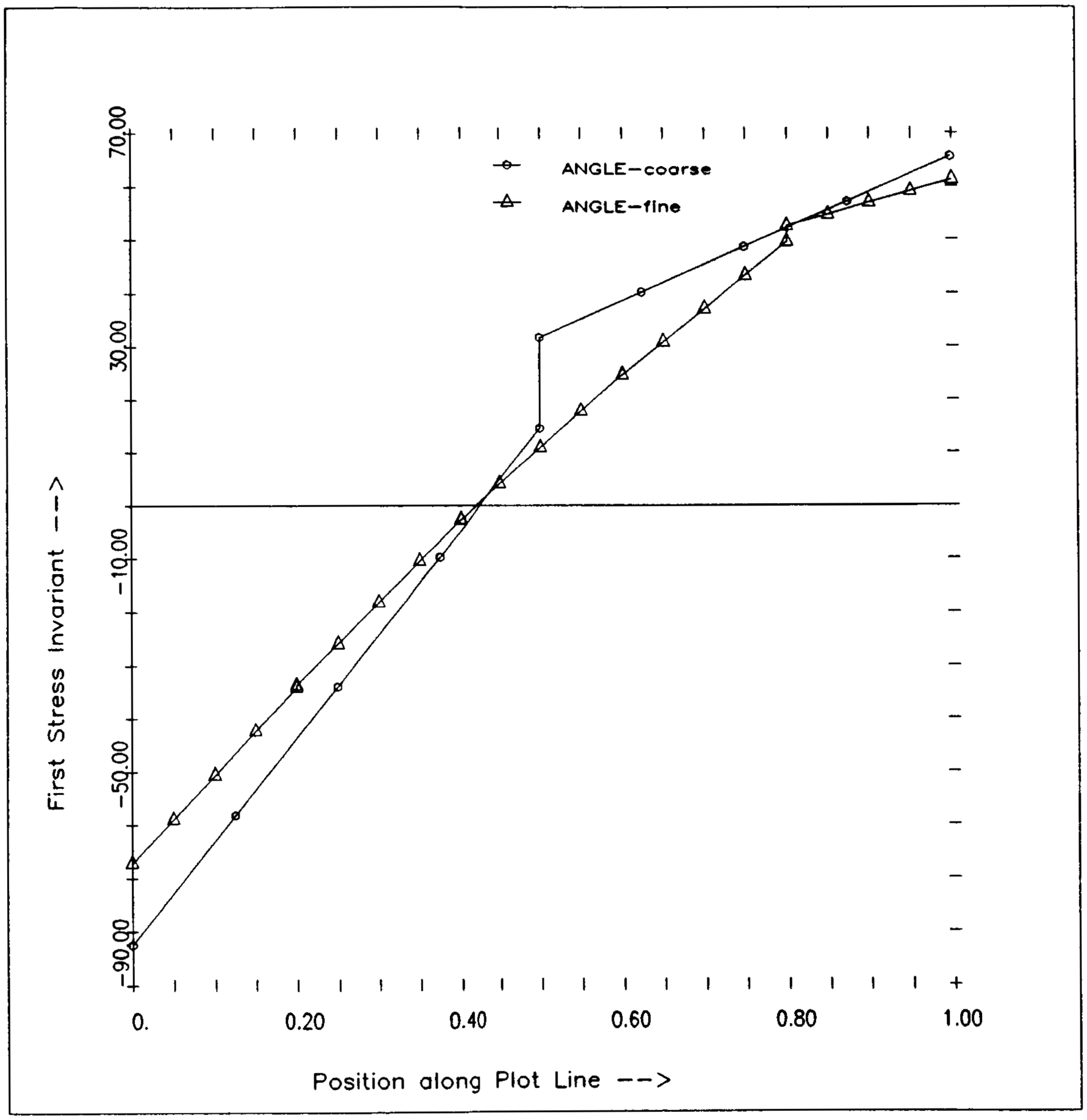


Figure 4.26 ANGLE:- Plot of INV-I along A2B2 - Load Step 2

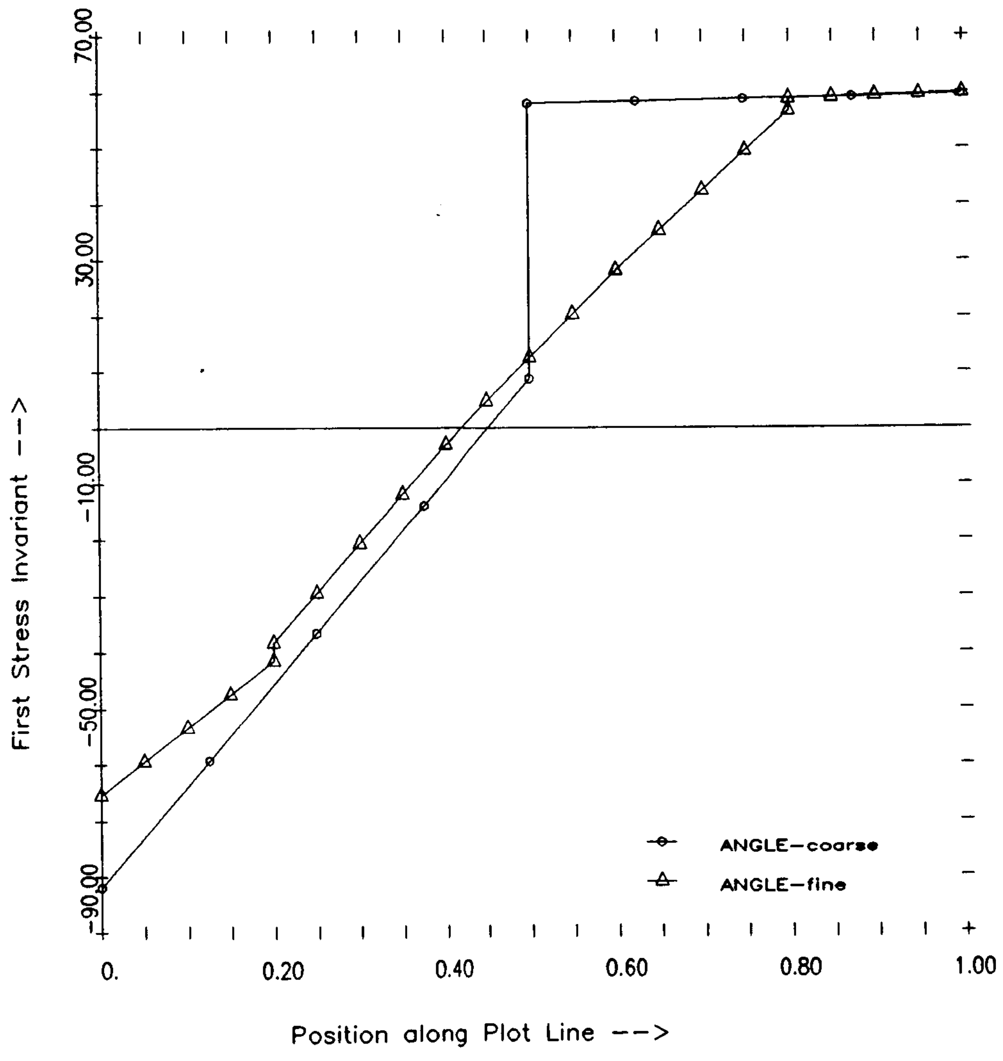


Figure 4.27 ANGLE:- Plot of INV-I along A2B2 - Load Step 3

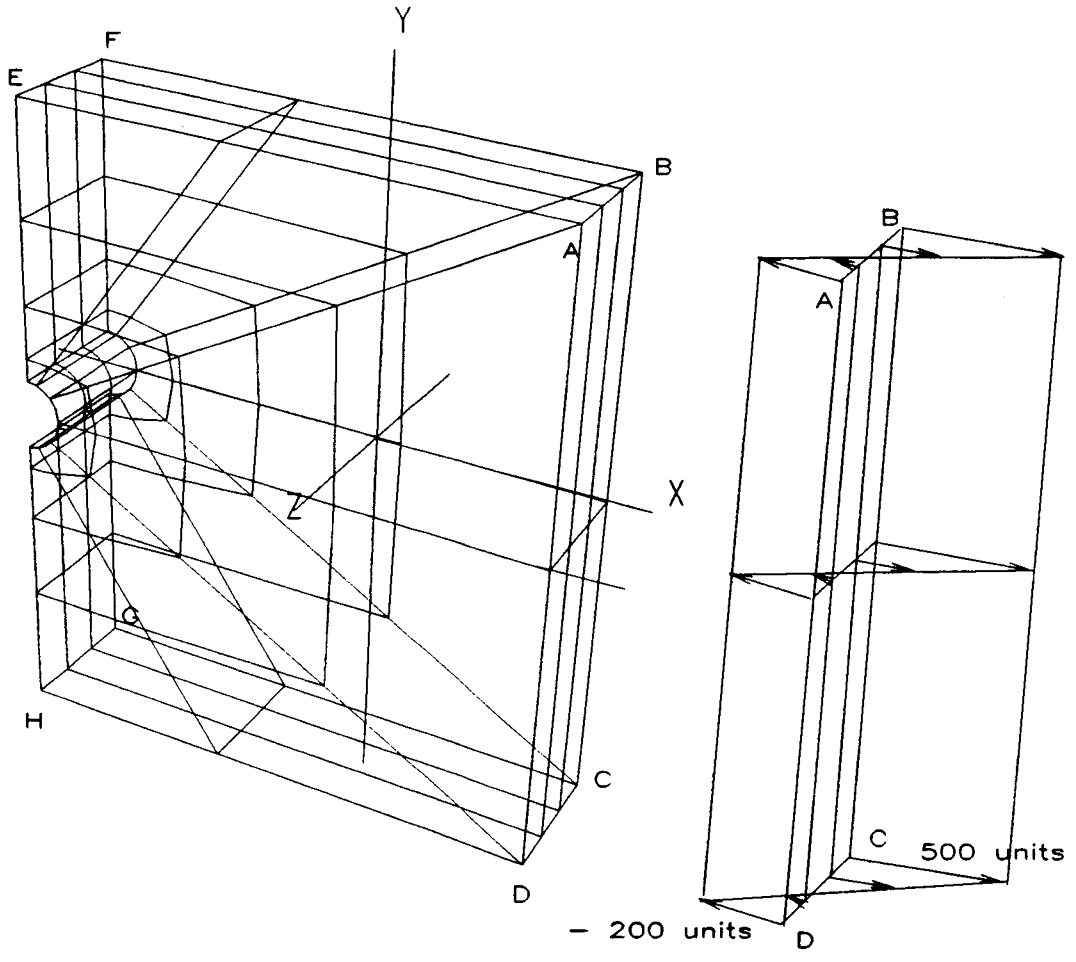


Figure 4.28 PLATE:- Geometry and loads

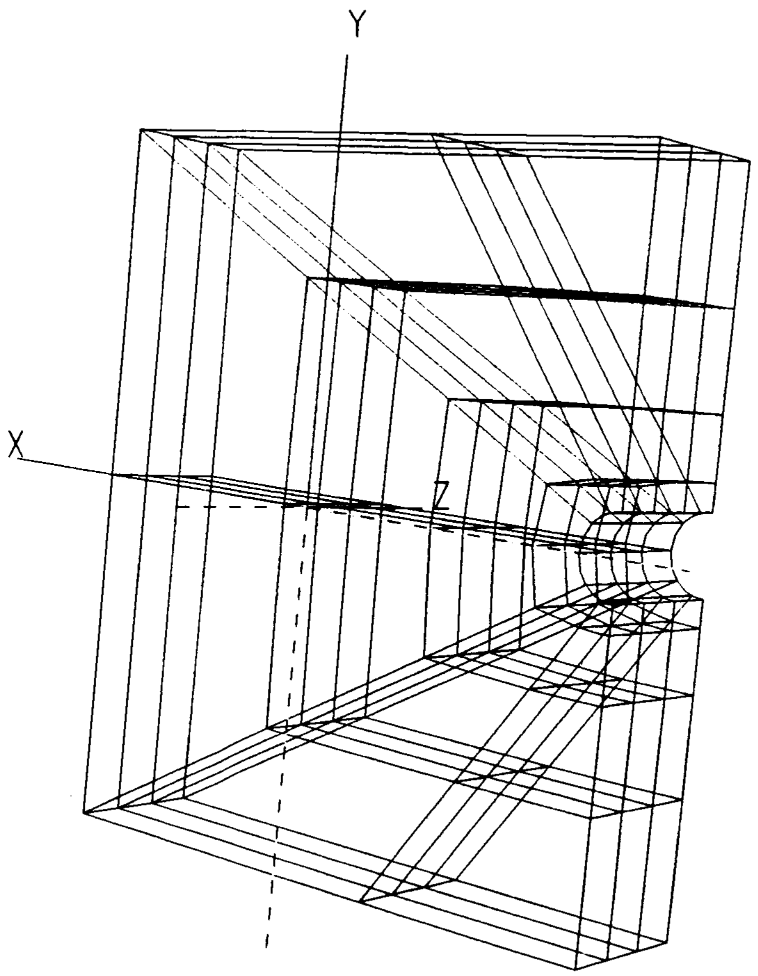


Figure 4.29 PLATE-coarse:- Boundary of the finite element mesh

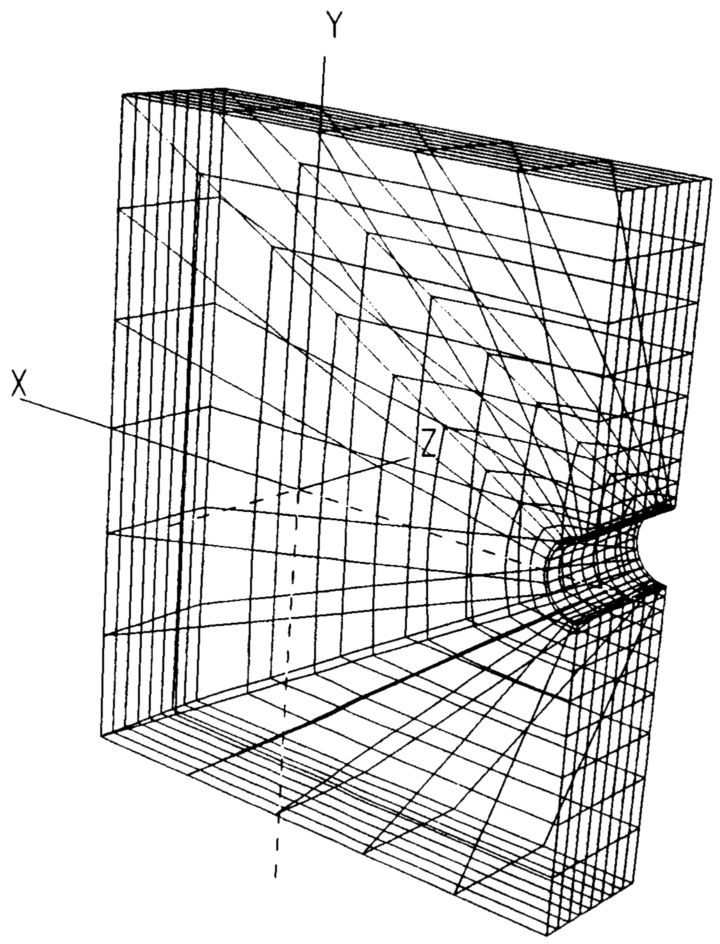
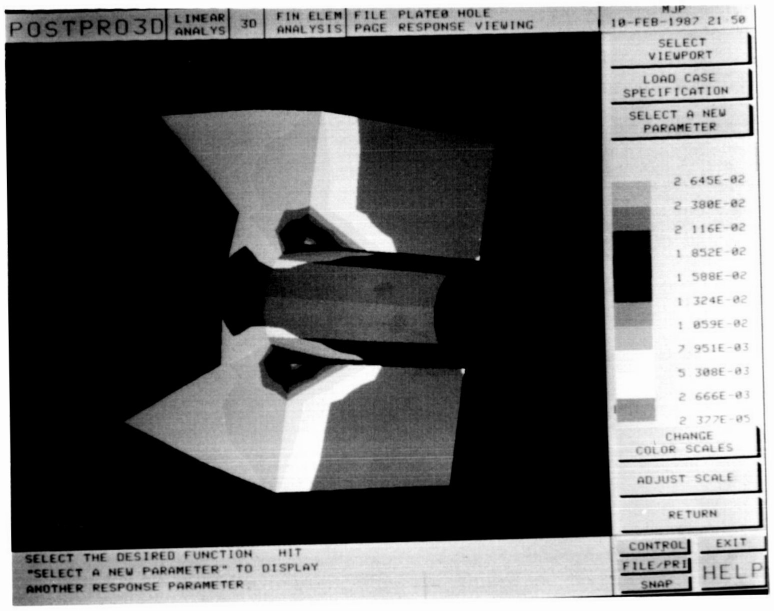


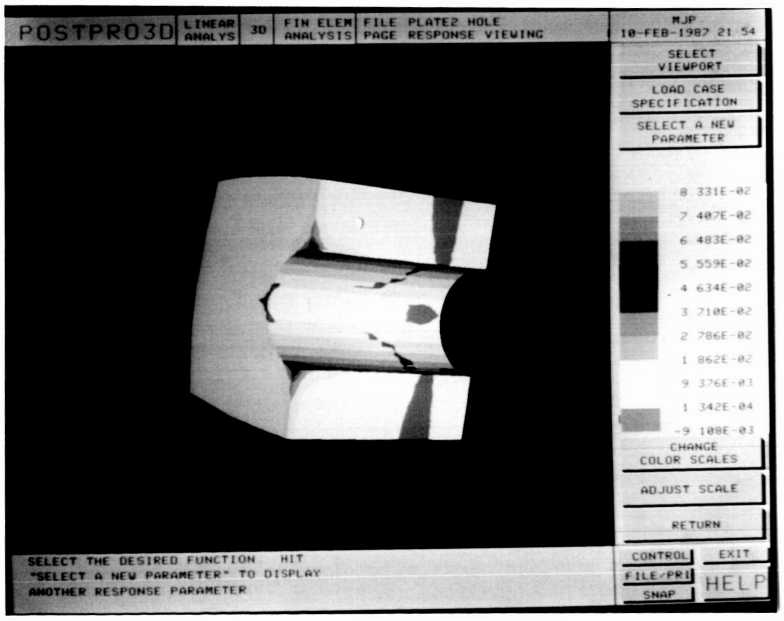
Figure 4.30 PLATE-fine:- Boundary of the finite element mesh

ORIGINAL PAGE
COLOR PHOTOGRAPH



SELECT THE DESIRED FUNCTION HIT
"SELECT A NEW PARAMETER" TO DISPLAY
ANOTHER RESPONSE PARAMETER

Figure 4.31 PLATE-coarse:- Contours of SED on a subobject



SELECT THE DESIRED FUNCTION HIT
"SELECT A NEW PARAMETER" TO DISPLAY
ANOTHER RESPONSE PARAMETER

Figure 4.32 PLATE-fine:- Contours of SED on a subobject

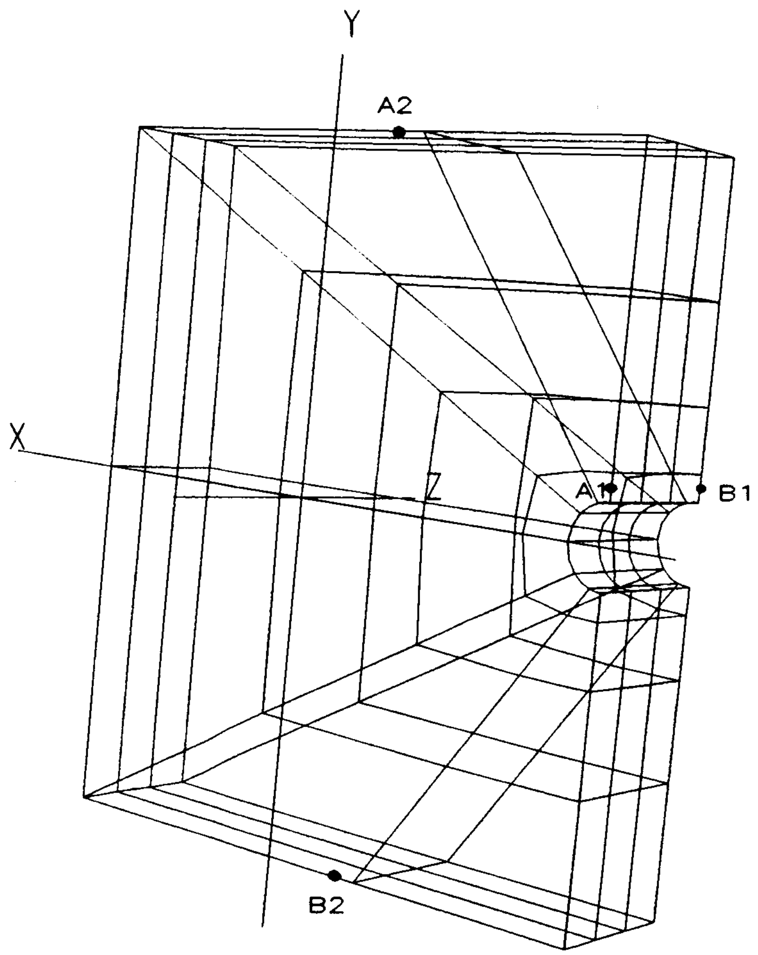


Figure 4.33 PLATE:- Plot lines A1B1 and A2B2

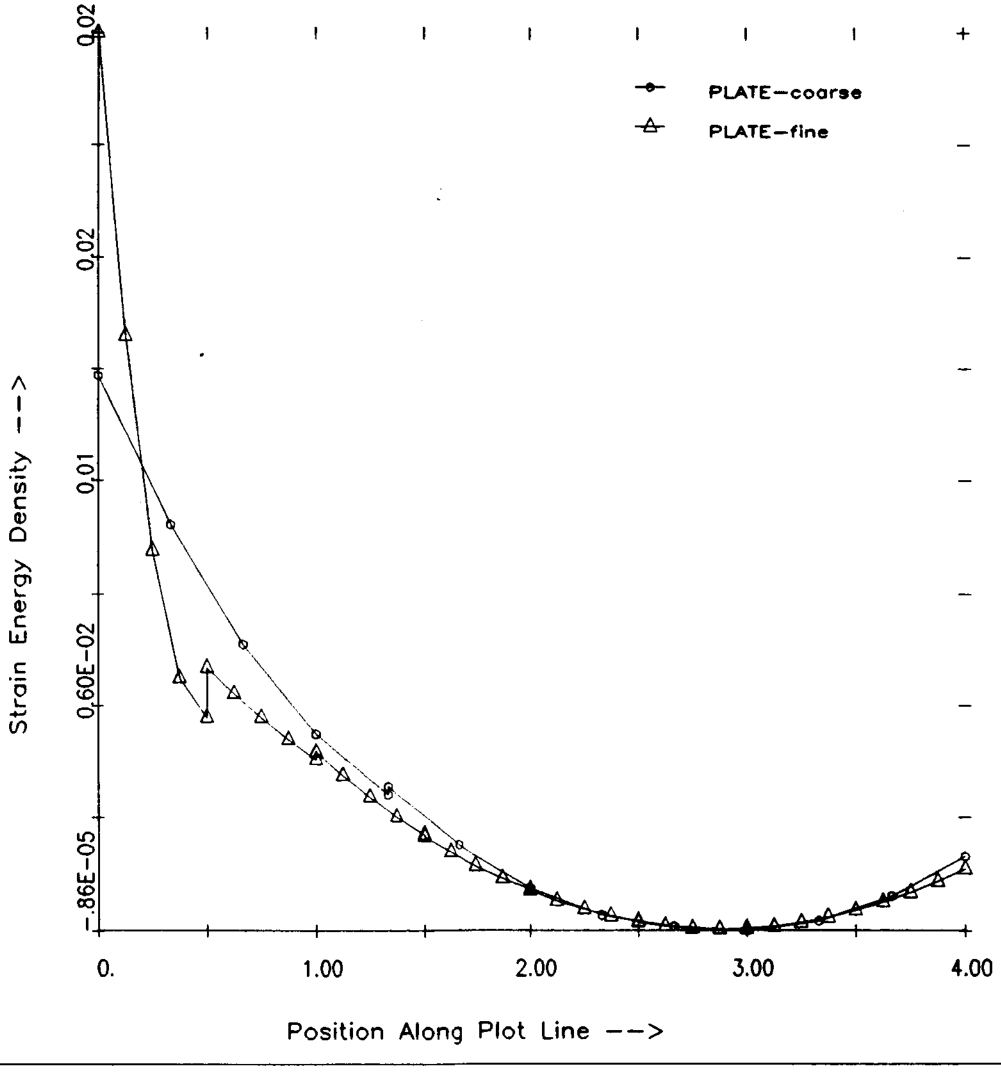


Figure 4.34 PLATE:- Plot of SED along A1B1

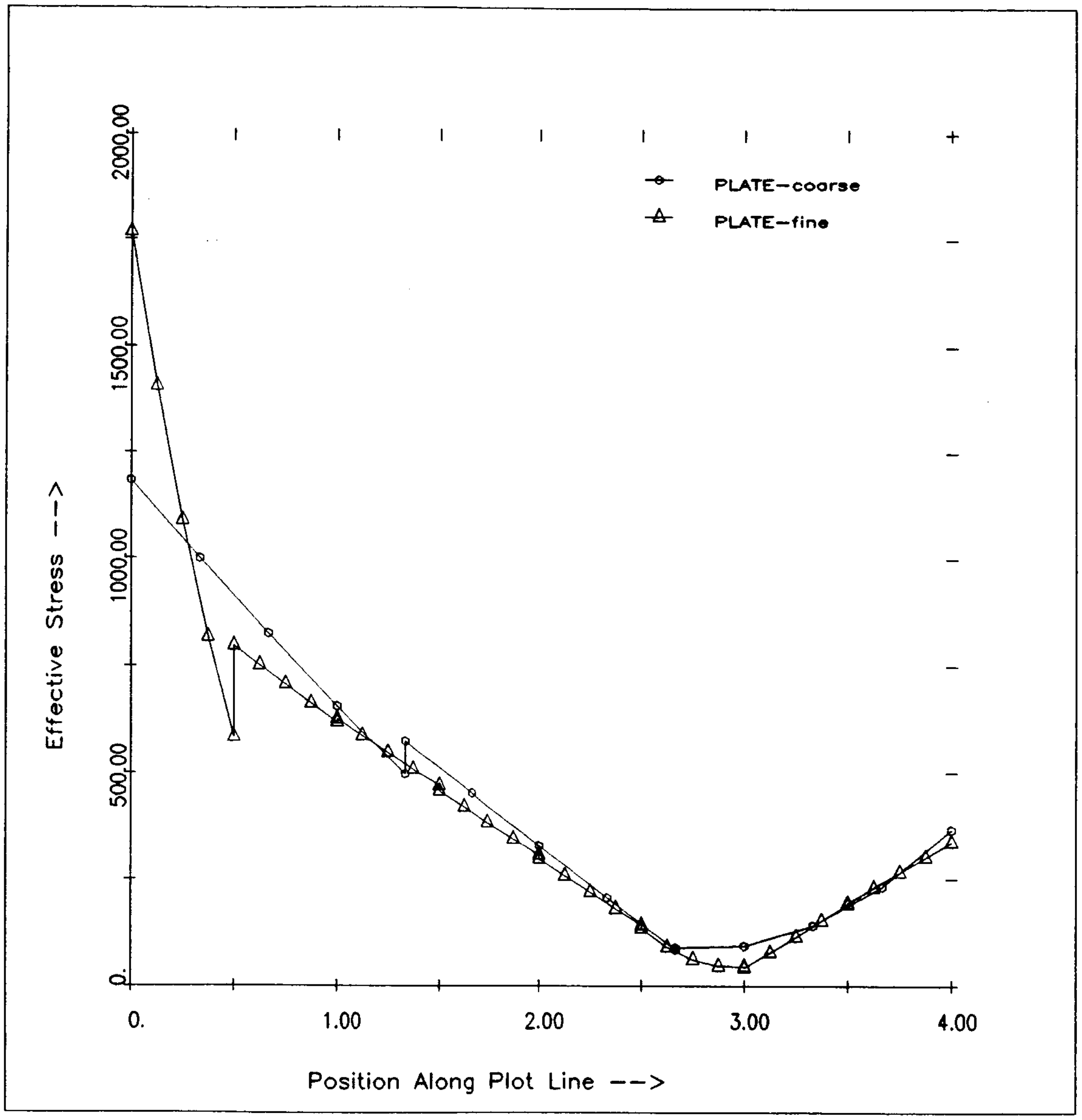


Figure 4.35 PLATE:- Plot of Effective Stress along A1B1

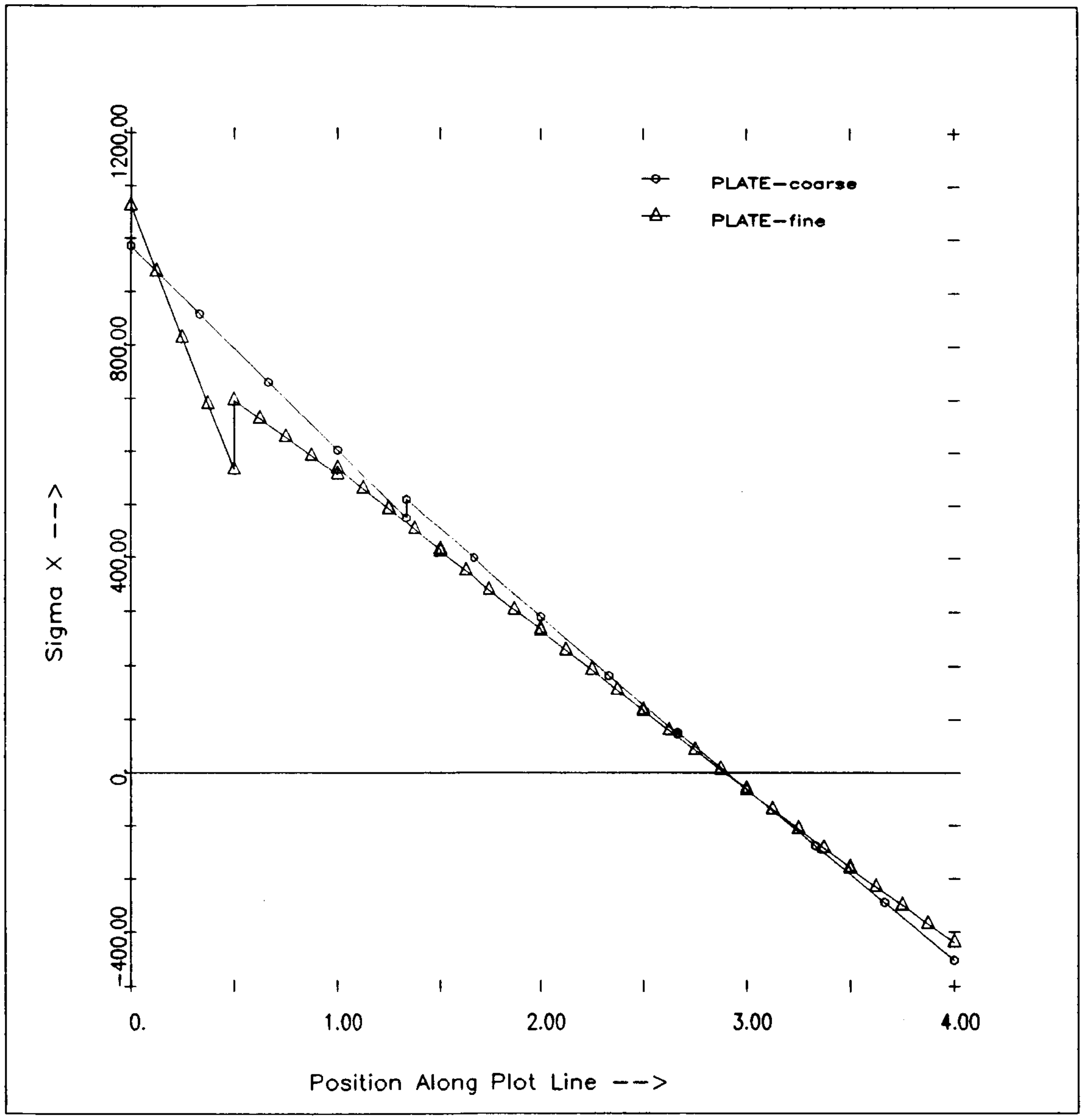


Figure 4.36 PLATE:- Plot of Sigma X along A1B1

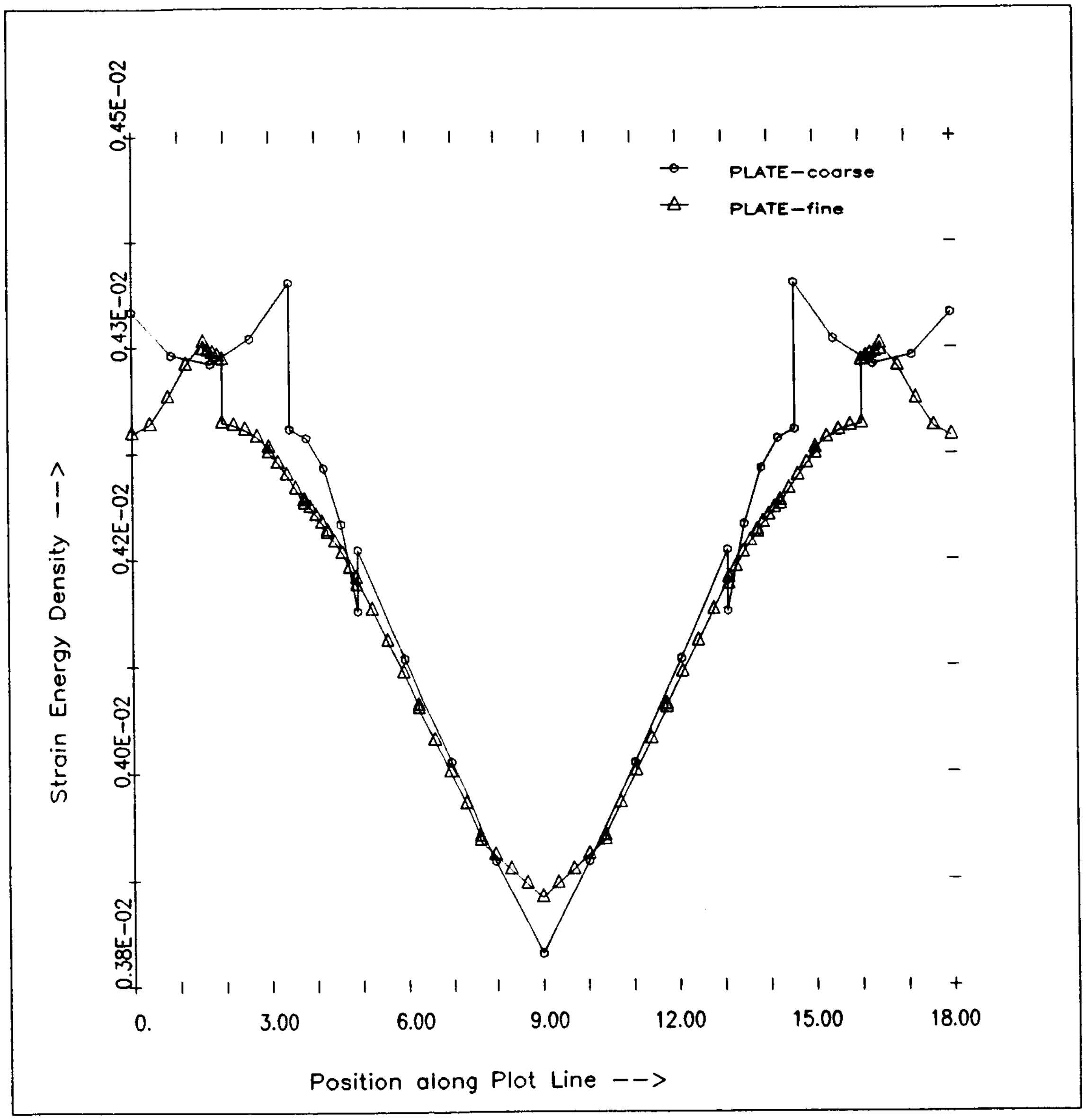


Figure 4.37 PLATE:- Plot of SED along A2B2

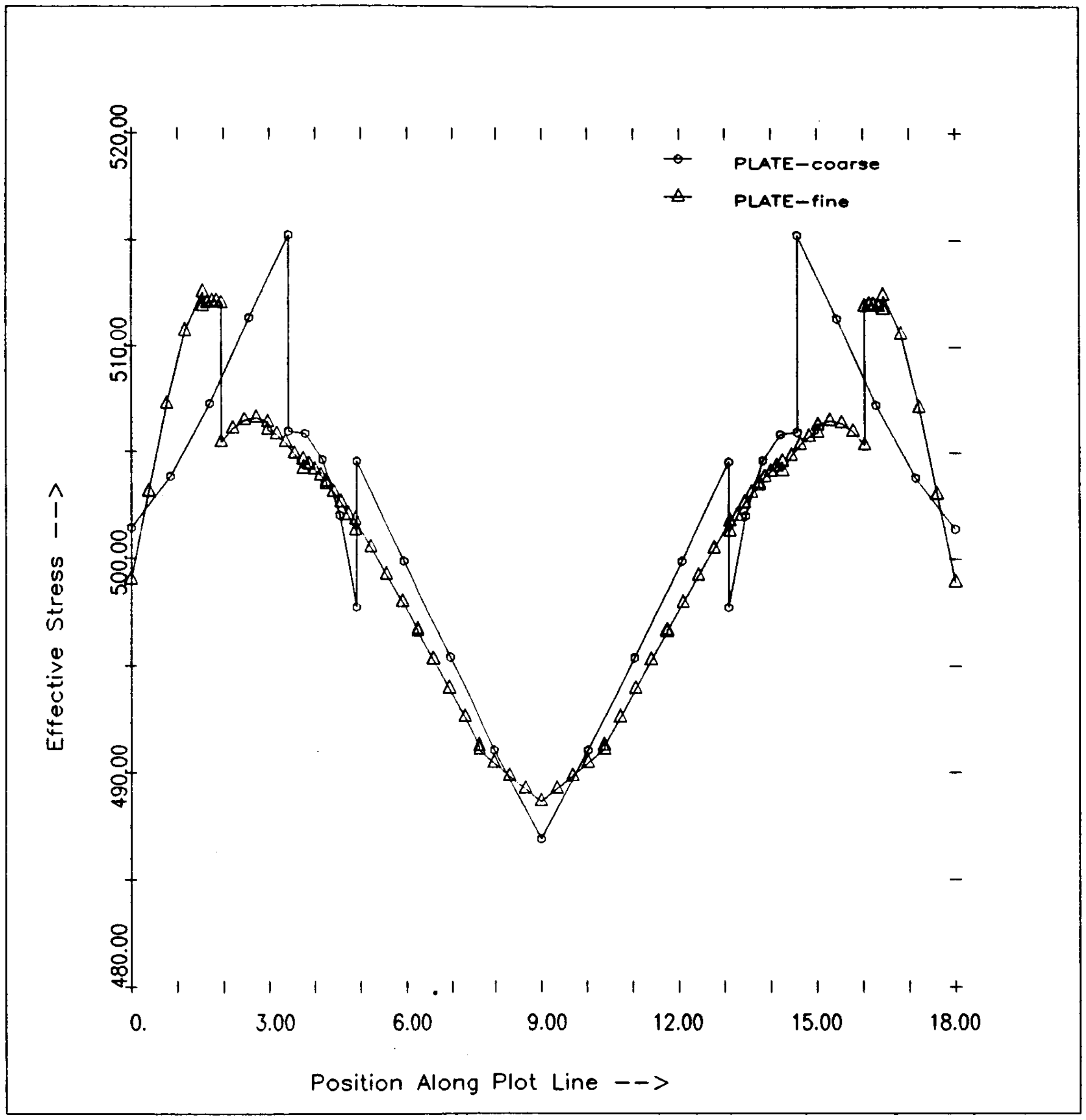


Figure 4.38 PLATE:- Plot of Effective Stress along A2B2

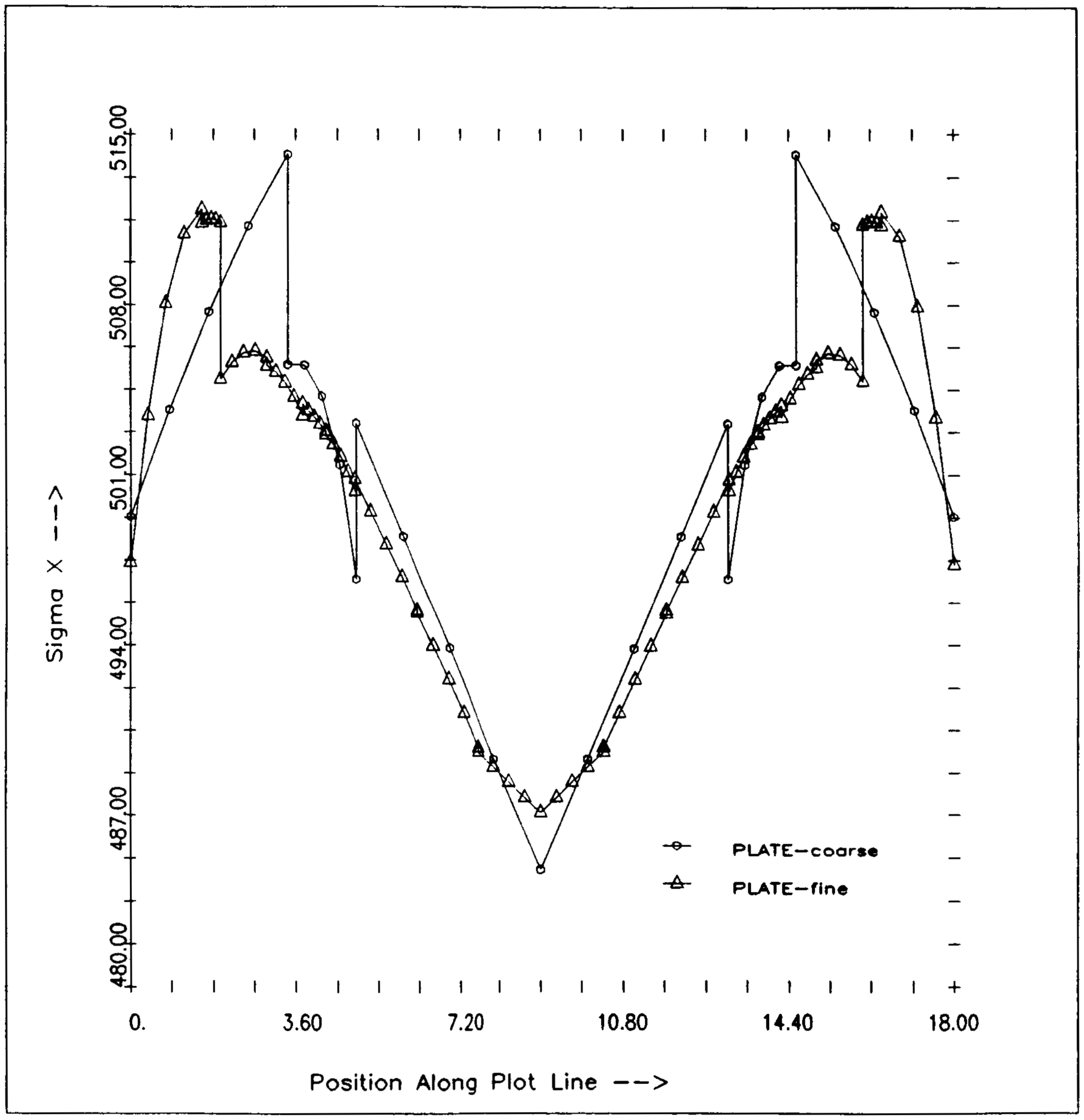


Figure 4.39 PLATE:- Plot of Sigma X along A2B2

CHAPTER 5

DESIGN OF A NEW POSTPROCESSING ENVIRONMENT

The eventual developments that may emanate from a research project that spans a number of years are difficult to forsee at its inception. But the experience gained from the effort normally enables the researcher to envision a new or different path at the conclusion of research which is quite different than that which would have been conceived at the outset. As this is the final year of a three-year research program in the area of engineering postprocessing, there now exists a clearer picture of how the required user-functionality can be re-designed in a way to facilitate a more powerful, flexible, user-friendly database and device independent environment. This chapter discusses the design features, and suggested scope and functionality of a new postprocessing environment. The basic changes in design philosophy that are suggested here reflect the experience gained over three years of development of postprocessing techniques and code. Also, suggestions are made for use of new graphics tools as a "Graphics

Manager" that can form the basis for the integration of the engineering design process.

The new design is now being implemented at the Program of Computer Graphics in Cornell University. The H.P. Series 9000/320 raster graphics workstations are the target display devices (Section 2.1.4 discusses the hardware specifications and the performance of these devices).

5.1 The Necessity For a Redesign

There are a number of motivations for the design of a new postprocessing environment. The existing program has begun to experience problems that stem from the implicit limitations of its design (see Appendix B for a discussion of memory associated problems). Also, it is not suitable for achieving the new objectives discussed in the following paragraphs.

The existing program uses translators to convert finite element, boundary element, and finite difference analysis databases to a single postprocessing database. To be able to view the contents of a new analysis database, a translator program must be written. This is a tedious and expensive process. It is also restrictive in a research environment generating continuously evolving analysis databases that need to use the same postprocessing capabilities. Further, database independence is essential to the process of integrating engineering design -- one of the long-term objectives of this research.

The advanced features of the new generation of high-level raster graphics workstations need to be exploited. These features -- 3-D transformations, diffuse and specular shading, and hit-testing (all done in hardware); and the existence of overlay planes, double buffers for "smooth" transformations, depth buffers and automatic scan-line hidden surface algorithms, dithering, and segmented display lists in virtual memory -- can now be considered a standard part of the hardware of new graphics workstations. In the existing program, these functions are software driven and, hence, much slower.

At present, the design of menu layouts and the grouping of functions on a menu page are done by the applications programmer. The program takes care of formatting the buttons, highlighting and unhighlighting them, and controlling the flow of the program. It is desirable to have a flexible environment that allows the analyst -- by means of a Menu Description Language -- to dynamically define menu layouts and associated functionality. Similarly, it is desirable to allow the analyst the option to change and create new color maps and define the "names" of response parameters being viewed. These features are hard-coded into the present program.

Recent years have witnessed a rapid increase in the computational speed of serial digital computers. But the speed of these machines is theoretically limited by the finite speed of light and the serial execution of commands (a single instruction is executed during each machine cycle) on a single processor. Future devices will have multi-processor, parallel architectures to exploit the greater power of parallel computations. The existing code has a certain degree of modularity built into it. But it is not suited for a multi-processor environment. A much higher degree of modularity needs to be built into the program if the individual modules are to be placed in the different processors of a multi-processor machine and linked over a local area network.

The development of large graphics engineering programs is time consuming. A disproportionate percentage of this time is spent in developing basic graphics tools. The use of general, high-level graphics software tools reduces this significantly. At present, the program is in complete control of the flow of information to and from the database, the electronic tablet and pen, and the display device. It is desirable to relinquish this to an external "manager." Also, software tools that process and manage lists, potentiometers, and sliders are useful. These are functions that are now handled explicitly by the postprocesor. The use of these tools, in addition to streamlining graphics applications programs, reduces their development time.

Finally, the present program has not been suitably designed to accomodate an indefinite increase in the size of analyses problems or the continued enhancement of postprocessing functionality. The lack of suitable global memory allocation and management has resulted in unforseen problems with memory intensive processes like the sectioning of large structures. The speed of execution of some important postprocessing functions -- like contouring and rotation -- leaves much to be desired. Appendix B discusses these problems in greater detail.

5.2 Basic Objectives of the New Environment

The new environment aims at solving the problems outlined above. To facilitate this, some general design philosophies are proposed.

The overriding design principle is one that believes that the personal preferences of each analayst must be supported. This concept of a Personalized Postprocessing Environment governs the design. The analyst must have the freedom to define and maintain a unique environment that is flexible and simple to modify. The new design also aims at simplifying the process of the addition of new postprocessing functions. Display-oriented tasks that are database independent (contouring, hit-testing, selection of points on the object surface, etc.) are handled by an Application Independent Data-Processing Module that can be accessed by all databases.

Another important feature is database and device independence. Database structures and the hardware charactersitics of display devices are constantly evolving, and their evolution must not be restricted by the lack of flexible postprocessing capabilities. The basic modular layout of the new design is seen in Figure 5.1. The main modules of the program are described in Section 5.3.

The new environment makes use of high-level graphics tools. A Menu Manager assumes control of the program and takes over the tedious task of menu formatting. It also allows simple and efficient user-defined menu redesign. Control of the color maps and potentiometers are other functions of the Menu Manager. A List Processor is used to maintain and interactively control lists of information. One of the important advantages of graphics tools like these is that they can be used by any graphics applications programs.

Strings defining the "names" of response parameters, and the colors of the discrete color maps used by the program are user-defined and can be modified simply. This gives the analyst greater control of the environment. The "Display Options" of each of the parameters, too, are user-defined; e.g. it would be possible to specify that the displaced shape can be viewed using the displaced mesh as well as a shaded image of the displaced structure.

Finally, it is proposed to create eventually a graphics environment that contains the various stages of the design process -- preprocessing, analysis, postprocessing, and mesh redesign. The creation of a Graphics Manager will facilitate the integration of these various stages.

5.3 A Modular Design.

All the features of the new design depend upon the strict maintainance of modularity. This feature is essential for a number of reasons.

Maintainance is a necessary part of large graphics applications programs. It involves not only "bug" fixing anf porting of the program to other devices but also the addition of new functionality. These tasks are facilitated by modular coding. The portability of a program is determined not only by the differences between the devices, but also by how modular it is.

One of the objectives of the new design is to be able to exploit the speed of parallel architectures. This is more feasible if the various sections of the program are modularized. It is important to be able to control the information paths between modules and to ensure that no memory is shared by them. This scenario allows each module to be placed on a seperate processor. Intensive data processing tasks -- sectioning, updating a contoured image -- can then be performed in the "background" while the user continues with other design functions. Besides the use of parallel architectures, a similar conceptual use for the modular design is that of "pipelining" of information between processes (a UNIX feature). In a serial computer operating in a UNIX environment, each module can be a seperate process connected by "pipes" to the other processes. These pipes act as information buffers; while data are being fed into one end, the module at the other end can begin operating on the data stream leaving that end of the pipe. This is as close as one can possibly get to asynchronous operations on a serial machine and can improve the speed of execution of data and display intensive tasks.

The contents of the various modules of the program are outlined below.

Application Independent Data Processing (AIDP): The Application Independent Data Processing Module contains code that is database and application independent. It communicates between the database, the Menu Manager, and the Device Independent Graphics I/O (DIGI). This is the device and database independent "Graphics Manager." The AIDP handles tasks that are not database specific -- for example, the interactive selection of points on the structure, the creation of nonlinear color maps, etc. It controls and maintains the display list -- creating new segments, switching segments on and off, clearing the image plane or overlays, etc. The routines of the AIDP can be used by any graphics applications program. It is envisioned that the AIDP will form the control core of the integration of the design process (see Section 5.7 for details).

Application Dependent Data Processing (ADDP): The Application Dependent Data Processing Module contains database dependent code. The basic functions include extracting information from the database, converting it to an appropriate form, and transferring information back to the

AIDP. These are the database-specific querying and modifying routines. Each database will have its own ADDP code. Subroutines that perform common utility functions will be kept in a shared ADDP Library that can be accessed by various databases; this prevents unnecessary proliferation of code. The speed of execution of functions like contouring, sectioning, material extraction, etc. will depend on the structure of the database and the efficiency of the ADDP.

View-Specification Module (VSM): The View-Specification Module is a sub-module of the AIDP and controls all aspects of view specification. This is one of the most important aspects of graphical postprocessing (or any other graphics applications program) . This module communicates with both the AIDP and the ADDP -- for example, the current concatenated transformation matrix can be sent to the ADDP upon request. Object transformations, controlling the display list, switching on and off objects, changing the background and light sources, and switching on and off object display modes (shaded image, wireframe outline, etc.) are the tasks handled by the VSM.

Device Independent Graphical I/O (DIGI): The Device Independent Graphical I/O Module serves as a buffer that communicates between the AIDP and the graphics library supported by the target display device. This module contains two sections: a device independent section of code, and device dependent drivers consisting (code that calls the graphics library supported by specific target display devices). The second section acts as a "buffer zone" between the first and the Device Dependent Graphical I/O Module -- figure 5.2 is a schematic representaion of the DIGI. When porting the program to another display device, the design of the first section of the DIGI ensures that the AIDP, ADDP and VSM code that communicates with the DIGI remains unchanged. The device specific calls in the second section are changed to match the graphics library of the new device.

Device Dependent Graphical I/O (DDGI): The Device Dependent Graphical I/O Module is the graphics library supported by the specific display device. In the case of the H.P. workstations, the library is called Starbase.

Menu Manager (MM): The Menu Manager is a high-level graphics software tool under development at the Program of Computer Graphics in Cornell University. When used, it assumes control of any menu-driven graphics package. Menu-layout formatting, keeping track of the cursor, drawing the menus on the screen, associating function calls with specific buttons, and the control of tools like list-processors and valuators (potentiometers and sliders) are some of the functions performed by the MM. The MM controls the peripheral user-interface devices like the tablet/pen, mouse, dials, terminal, etc. Menu Manager functions can be called by the other operating modules -- e.g. menu buttons and menu pages may be activated or deactivated, and menu buttons may be renamed.

5.4 Database Design for Numerical Analysis Applications

The speed of execution of an engineering graphics applications program depends mainly on: (a) the characteristics of the display hardware and (b) the efficiency of the database modifying and querying routines. The functions of any graphics program can be generalized to be: (a) operations on a graphics database and display of its contents interactively using the graphics database and (b) interactive modifications to the model and computational databases. The important common aspect is interactivity. As the computational and display complexity of the operations increases, speed, and hence interactivity, decreases. The analyst generally has no control over the characteristics of the display hardware but rather influences the design of the application database. Hence, the design of the database structures and the querying routines that extract information from the database is an important aspect of any graphics-oriented engineering program such as the postprocessor.

5.4.1 The Existing Database

The existing program uses translators to convert analysis databases to a single postprocessor database (PostD). The PostD consists of two packed strings of contiguous information -- the first, the {A} array, contains integer data, and the second, the {B} array, contains floating point data. Pointers delineate the start of various segments of the data [1,8]. At any given moment, the PostD contains response information for a single response set. If the analyst desires to view a different response set, the response data of the previous set is removed and that of the new set is explicitly "read" into the PostD.

One of the problems related to the PostD is the efficiency of extraction of information. There are database management routines that handle the tasks of inserting and deleting whole segments of the {A} and {B} arrays, but no routines whose primary function is the extraction of information from the database. At present, information is extracted from the database by using pointers as in the following example:

```
    DESIRED_NODE = NODE_NUMBER
    POINTER = PB$NODAL_COORDINATES + (DESIRED_NODE-1) * 3

    DESIRED_X_COORDINATE = B(POINTER +1)
    DESIRED_Y_COORDINATE = B(POINTER +2)
    DESIRED_Z_COORDINATE = B(POINTER +3)
```

in which

PB$NODAL_COORDINATES is a global pointer to a segment of the **{B}** array that contains nodal-coordinate information.

Assignment statements like these are randomly distributed throughout the menu control, display, and data processing sections of the program. This implies that changes to the database structure would necessitate changes throughout the program. This is an undesirable aspect of the present database organization.

A second problem with the PD is the lack of nodal, element, edge, or boundary face contiguity information. The PD stores element data in the form of nodal connectivities. There is no concept of linked or doubly linked lists to provide local proximity information. This information is useful in functions like hit-testing for surface information, creation of plot loops and plot lines on the surface, extracting response information along a plot line, extracting node numbers, element numbers and material properties, and sectioning the structure. As the size of the problems being analyzed increases, the potential improvement in the speed of execution with the use of proximity information becomes more apparent.

A third problem concerns database management. At present, the translators convert analysis data into the five postprocessor files -- .OUL, .BOU, .COR, .PAR, .GPV (refer to Chapter 1 for details). At run-time, there is a further translation phase to read these files into a form that fits into the {A} and {B} arrays. Thus "reading" in a problem can take up to 10-15 seconds depending on the size. This time can be eliminated if data is stored in a form that all the applications programs can directly access.

5.4.2 Suggestions For a New Database

The new database design aims at solving the problems associated with the existing PD. Data management will be an explicit function handled by a set of querying and modifying routines. The new environment attempts to remove the data translation phases that now exist between the various programs. Data will be put into a form that all the applications programs can directly use. Pointers into the database obviate the need to "read in" the information into a specific postprocessor format.

The above changes deal with the management of data. Though the database structure affects the performance of the querying routines, these changes are essentially independent of it. The lack of geometrical contiguity information is a problem that is more difficult to solve and involves the entire structure of the database.

At present, the origin of the "contiguity problem" for 3-D finite elements lies in the preprocessing program that generates the initial geometric data. Although this program uses contiguity information for generating meshes during the lofting process [37], the output data -- the .GEO, .INP, .DAT files -- do not contain it. At the postprocessing stage, this information can be generated from the nodal connectivity data but this is an expensive process. This problem must be solved during the preprocessing stage.

Extensive research has been conducted in the area of boundary-representational graphical databases. There are a number of existing database structures that contain elemental contiguity information of the form discussed above. Linked and doubly-linked lists are the basic format of these structures. One such database is the Winged-Edge Database (WED) and modifications of it [38-40]. This database structure has been found to be useful for 2-D problems [41] but would need to be modified if it is to be efficiently used for 3-D structures.

5.5 Personalizing the Postprocessing Environment

Personalization is one of the main design features of the new environment. Experience in the area of engineering postprocessing has shown that the requirements of individual analysts vary. Also, the development of graphics code is time consuming. Hence there is a need for an environment that allows analysts to satisfy individual requirements with minimal effort. The resulting postprocessing environment would grow as the needs of the user-group grow. At the same time, as it is to be implemented on workstations, each executable image of the program (one on each of the workstations) could be different -- adapted to the particular analyst's database design and requirements.

5.5.1 Database Independence

There has been extensive research in the area of device-independent graphics languages, but the area of database independence has not received sufficient attention. Postprocessors are essentially graphics programs that facilitate rapid viewing of engineering analysis data. The final communication with the display device is a visual representation of data and is always a set of polygons, vectors, or text. The nature of the database does not alter these graphics primitives. Thus, a postprocessor is potentially database independent.

The AIDP has no concept of how the database is structured. The designer of the database must also design a set of querying and modifying routines that "operate" on the database. The set of necessary querying calls are the same for all databases as the information needed by the AIDP (if the functionality remains unchanged) is the same. Each database would have its own implementation of these calls in the ADDP. Common utility modules may be built into an ADDP Library shared by different databases.

This feature has a number of advantages, especially in a constantly evolving research environment. A new database can use all the functionality of the postprocessor. The only programming task that the analyst faces is the development of the database querying and modifying routines. Also, a researcher can modify the structure of an existing database with no effect on a large part of the program -- only certain ADDP routines directly linked to the modified part of the database would need to be rewritten.

5.5.2 String Parsing

At present, the names of the responses, analysis types and other analysis-specific "string" data like element types, response types, etc., are hard-coded into the program. The new design eliminates this restrictive design.

A text file containing information about response parameters is created by the analyst. This file contains the "strings" that are the names of all the response parameters that can be viewed using a particular database. Associated with each of these is a code that informs the AIDP about which "display modes" are allowed for the parameter. Examples of diplay modes are discrete contouring, smooth contouring, smooth shading, and vectors. At present, each response parameter is associated with a single display mode that is hard-coded into the program. This is overrestrictive. For example, the analyst may decide to view principal stresses as either discrete contours, smooth contours or vectors. This would be a simple task in the new environment provided the AIDP supports these rendering modes. As new applications create the need for different display modes, these can be incorporated into the AIDP and become available to the other databases.

A second text file created by the analyst contains information about the kind of elements with which a particular database deals. This file also contains data on the number of nodes, and the number of degrees of freedom at each node of the elements. At present, the postprocessor supports four element types -- 6-noded wedge, 8-noded brick, 15-noded wedge and 20-noded brick finite elements. A new element cannot easily be incorporated into the postprocessor database, especially if it has a variable number of degrees of freedom at its nodes -- which is the case in some shell elements. The new environment allows the analyst to introduce new elements of any kind into the program without affecting the code -- only this text file needs to be modified to introduce the elements.

5.5.3 Color Map Design

A third file contains the color maps. At present, there are six color maps (each containing 10 colors) that the user can choose from. These are hard-coded into the program. The visual perception of users can differ; it is useful to be able to build suitable color maps according to individual needs. This file contains default color maps that may be retained if necessary. There is no restriction on the number of possible maps the user may build into this file or the number of colors each map contains. Information for smooth contour maps would also be stored in this file.

5.5.4 Menu Layouts and Evolving Functionality

The use of the Menu Manager and a Menu Description Language facilitate the easy design and redesign of menu-layouts. The AIDP and the ADDP are modular and allow the analyst to "move" individual postprocessing functions between menu pages. This removes the tedious task of menu design and formatting from the applications programmer, and allows individual analysts the freedom to change the menu structure. The use of the List Processor and Valuators further simplify the task of the programmer. These high-level software tools are general and can be used by any graphics applications program.

The Menu Description File, written in the Menu Description Language, allows the designer to name buttons, specify their type -- prompt, potentiometer, message box, etc. -- and location, and associate a function call with each. The designer may leave it to the MM to format these buttons in a given space or may specify their exact locations.

New functions can easily be added to a menu page. The modularity of the ADDP and AIDP, and the simple structure of the Menu Description File ensure that this is a simple task. Any new postprocessing feature that is a useful, general tool becomes available to all users through the AIDP.

5.6 The Display List

The Rastertech Model One/380 has a segmented display list that uses a fixed section of the memory of the frame buffer whose size is restricted to one megabyte. In contrast, the H.P. workstations retain the display list in virtual memory, and thus the size is restricted only by the virtual memory capacity of the system, a minimum of 30 megabytes, and may be as large as 4 gigabytes. Current and future workstation technology can be expected to configure display lists in this manner.

There are a number of advantages to this configuration. The large size is an obvious advantage. Separate segments can contain the mesh, the outline, a shaded view, a contoured image and any other high-level graphics primitives like plot lines, plots, etc. Individual segments can be switched on or off without necessitating recomputation. Transformations on these segments can now be controlled by the hardware. One of the advantages of a segmented display list is that selected segments can be transformed without affecting others. At present, for example, every time the user desires to switch on the mesh, the coordinates of all the vectors composing the mesh are extracted from the database and "sent" to the display device. In the new environment, geometric information of high-level primitives like the mesh, can be stored in the dynamic display list every time a new problem is "read in." This information will not change for the given problem; during the interactive session, the segment containing it can be switched on and off. This would improve interactivity.

The AIDP maintains control of the display list. The ADDP communicates with the AIDP using Display ID's (DID) to identify the segment of interest. The ADDP maintains a view status of the various segments. This status keeps track of what high-level graphics primitives have been sent to the device, where they are displayed (image plane or overlays), and whether the segment is currently on or off. (Both the ADDP and the AIDP can control switching segments on and off.) As the AIDP handles all viewing operations and transformations, the "view" information needs to be sent to the ADDP in the form of the transformation matrices.

5.7 Future Integration of the Engineering Design Process

At present the design process is usually fragmented into its individual components. Each stage can only accept input data in a specific form and generates output data in another form. Translation programs are used to convert output data from one program into the input format that the next program can accept. This is a wasteful and expensive approach which could be alleviated if all the programs were integrated through use of the same database and if the design concepts for the new postprocessing environment were applied to the entire design and analysis process. As more information is generated, the overall design database is modified. Database modifying routines perform this task, while database querying routines perform the task of extracting information from the database.

Figure 5.3 shows the envisioned integrated design environment. The Menu Manager is used to manage the flow of control for all the applications programs. For a given analysis type, say finite element, all the programs share a common database. There are no translation phases between the various segments of the design process. The geometric information is built during the preprocessing stage. The analysis phase adds response information to the database. Postprocessing includes mesh quality evaluation and is then tied-in with preprocessing for mesh redesign. This environment would realize the goal of creating a true "design loop."

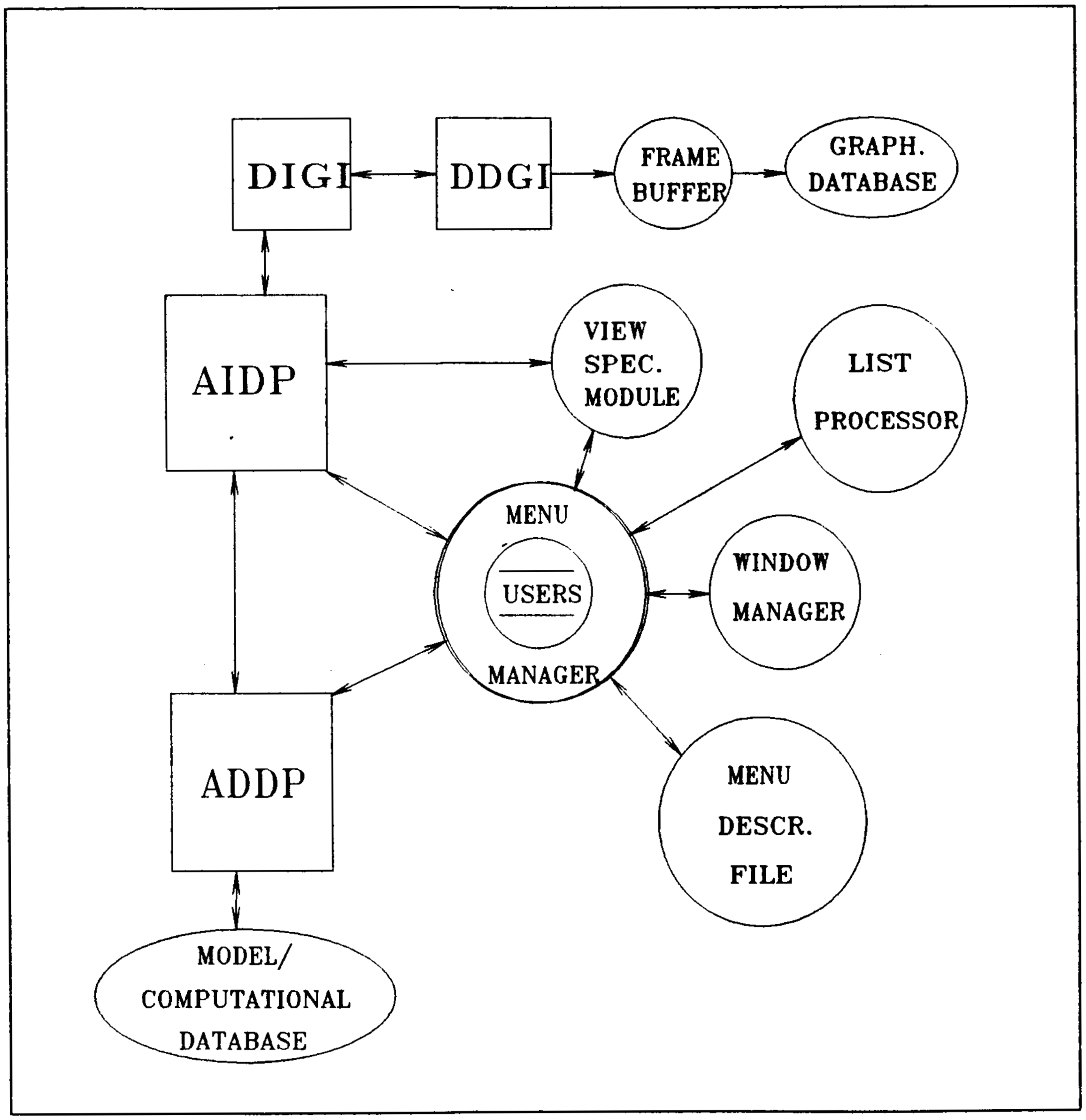


Figure 5.1 Modular layout and inter-process communication

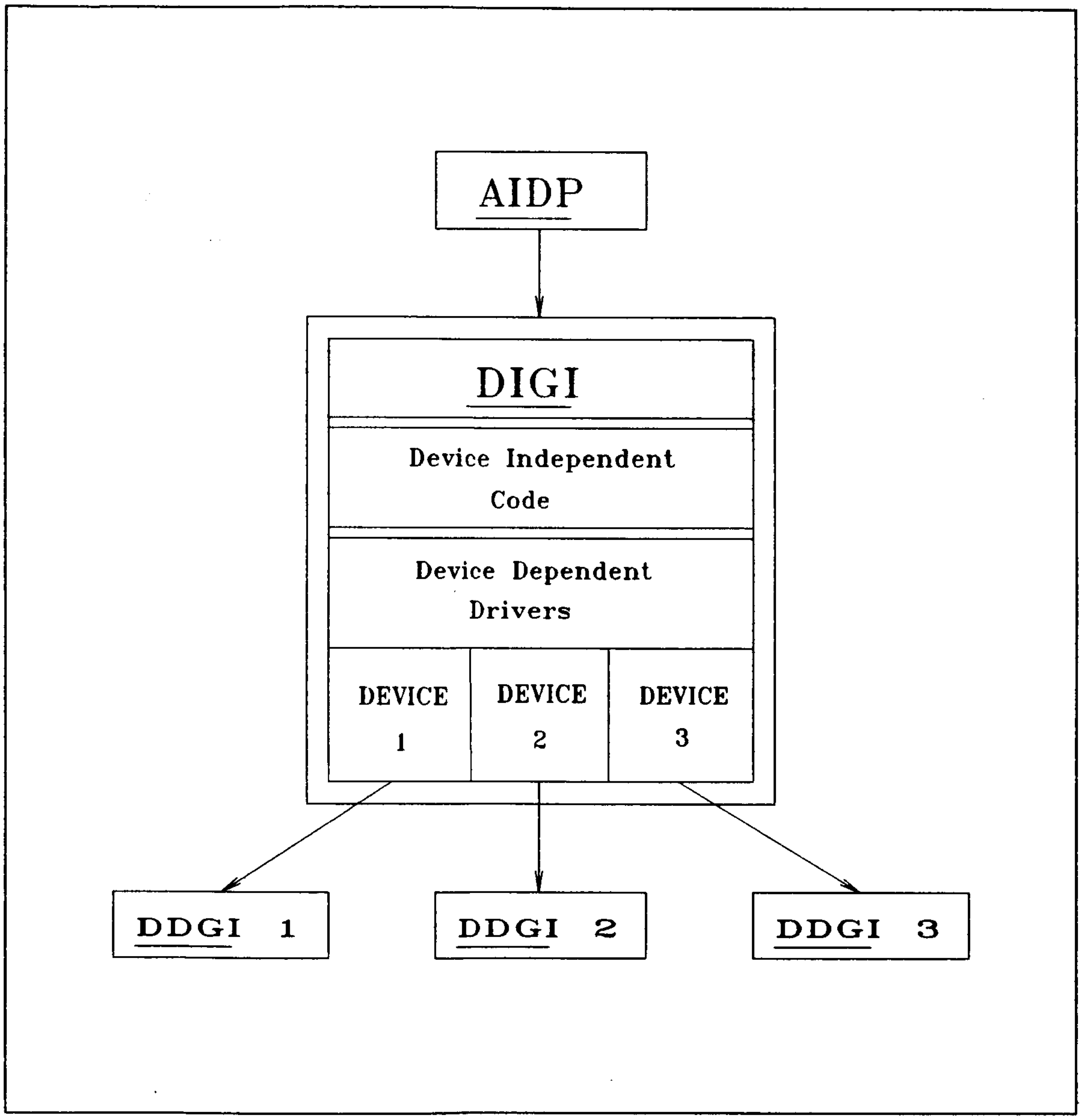


Figure 5.2 Schematic design of the DIGI module

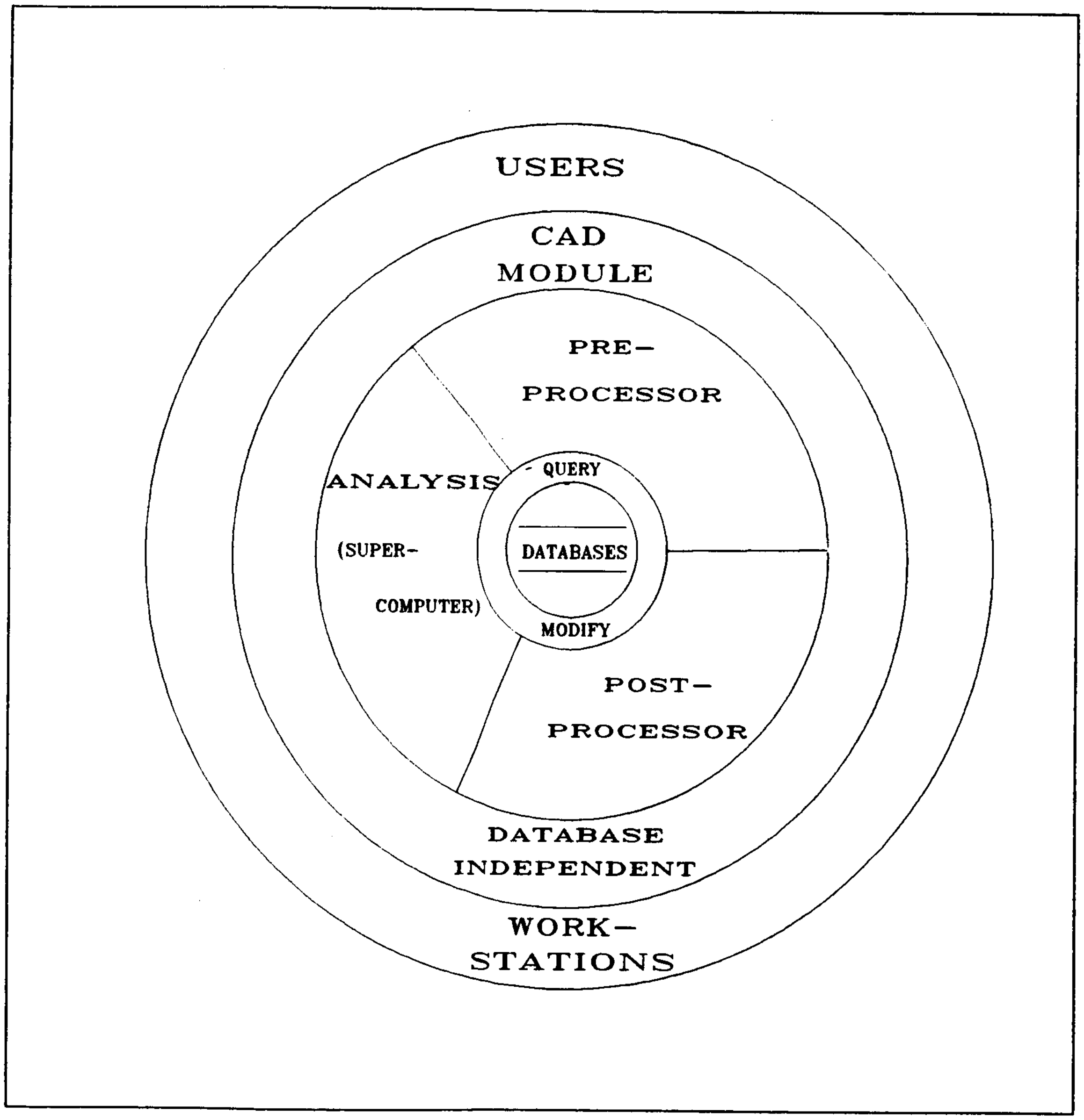


Figure 5.3 Integrated CAD environment

CHAPTER 6

CONCLUSION

The purpose of this research is to develop new techniques to facilitate color postprocessing, including the development and design of new postprocessing environments using high-speed color graphical workstations. Graphical tools for interactive finite element mesh quality evaluation from a single analysis are also developed.

This chapter summarizes work during the third and final year of research and speculates on future software and hardware environments that will affect the nature of graphical postprocessing.

6.1 Summary and Conclusions

The general design objectives stated during this three-year project [1,8] have been achieved -- generality of application, control of image manipulation, complete response and attribute viewing, interactive graphical capability, and potential for transportability of software. New computing environments have necessitated modifications to some of these, and have added new objectives. The detailed objectives for this phase of the work stated in section 1.2 have been achieved during the course of this research.

The existing postprocessing program is ported from the low-resolution Lexidata System 3400 frame buffer and monitor to the high-resolution Rastertech Model One/380. Hardware differences between these devices necessitated the redesign of the following algorithms: (a) hit-testing (b) contrasting specific base colors using the color maps, and (c) changing the color map used for contouring.

The minor refinements made to the existing program are too numerous to detail or list here. The most significant new postprocessing features added to the program include: (a) The process of "Material Extraction" is made more visual. Each material is assigned a different color and the analyst can interactively choose materials from a color map or the main view window of the display screen. (b) An Annotation menu page is added. Here, the analyst can add annotation to an image. This is useful for reproduction of images during demonstrations, and in technical papers or reports. (c) The existing database is modified. Unaveraged boundary response information for rapid qualitative error analysis is included. Software color maps are stored in the database to simulate a one-channel color look-up table. Pore water pressure is a new response parameter that can be viewed using the program, and it is now possible to contour cartesian stress quantities for boundary element problems. (d) An improved technique for response smoothing for 15-noded wedge isoparametric elements is implemented.

An important postprocessing task is the evaluation of the quality of the mesh -- "How valid is the analysis?" A literature survey is conducted in the area of **a posteriori** error analysis for finite element meshes, and a bibliography compiled. The efficiency of various mesh quality sensors is discussed. Response discontinuities across element boundaries are used as a qualitative measure of the accuracy of the mesh. Interactive graphical tools are developed to facilitate 2-D plots of unaveraged responses along any user-defined line on the surface of the structure. Multiple plot loops can be extracted and the analyst can select plot-line end points on any of the extracted loops. Also, the use of strain energy density contours as a qualitative mesh sensor is discussed. Example problems are used to demonstrate the efficacy and limitations of these qualitative techniques.

Experience with the use of 2-D plots as mesh quality sensors has led to the following conclusions: (a) they are useful as a good initial feel for the local and global quality of the analysis under the limitations described in section 4.4.3.3, (b) the sensor is found to be more efficient in 2-D than in 3-D, (c) it does not give a quantitative estimate of error, and (d) its efficiency as a mesh quality sensor depends on the method used to extract responses along plot line. For automatic mesh redesign, an energy-based quantitative measure (such as the energy norm suggested in [44]) is recommended.

Many engineering computing environments of the future will consist of networked color graphical workstations around a supercomputer core. Postprocessing, as any other graphics-oriented computational task, is affected by the advent of high-performance workstations. This research examines the effect of the new hardware on engineering postprocessing. A new postprocessing environment is designed to exploit the enhanced features of these workstations such as high-level graphics tools. Suggestions are made for the use of other geometric and computational databases that are more efficient from the point of view of the integration of the engineering design process.

6.2 Suggestions for Future Research

Rapidly evolving hardware allows the analyst to examine problems of increasing complexity. Postprocessing techniques that are not interactive with existing technology would become feasible. With the advance of supercomputing and networks with higher bandwidth, graphics workstations will be used as windows that can monitor analyses in real time. Graphical tools such as multiple viewports -- that allow the visualization of multiple views of a single problem or different problems -- enhance postprocessing capablilities. Real time rotation of not only shaded polygonal images but also contoured images would be possible as rendering speed improves.

The third year of research has advanced postprocessing capabilities, and raised a number of questions about the existing techniques and algorithms in the light of advancing hardware environments. The following is a brief list of the major areas that require further research.

Mesh Quality Evaluation: The error measures that are implemented are qualitative. It is necessary to quantify error to be able to close the design loop automatically. Other appropriate error quantities would need to be implemented and tested. Hierarchical elements and p-type mesh refinement procedures are an intriguing alternative to h-type refinement principles, and would need to be examined. Research in the area of 3-D error measures is ongoing.

Integration of the Design Process: Section 5.7 contains suggestions for the use of a unified geometric, computational, and postprocessing database to facilitate the integration of the design process. Research in the area of database design -- for efficient databases with element, edge, and nodal contiguity information, useful in the case of problems with changing topologies -- needs to be conducted. This would be essential for efficient integration of the design process.

Multiple Viewports: Existing monitors have the required screen resolution to facilitate the implementation of multiple viewports. This is a feature that would significantly enhance the ability of the analyst to examine the behavior of a structure, and to be able to study different analyses that are related -- say, mesh refinement studies of the same structure, separate parts of a machine component, etc.

Transparency: This is a useful feature that can facilitate viewing the "inside" of a structure. The analyst can select a desired reponse value and contour an "equi-value," single-color surface through the volume of the structure. Coupled with animation techniques, this would aid in the study of the propagation of elastic waves, or plastic surfaces in a structure over time.

Response Animation of Time-dependent Problems: As rendering speeds of workstations increase, response animation of time-dependent (or pseudo-time dependent) problems becomes feasible. This feature would enable the analyst to visulize the variation of responses over the domain through time. Essential factors would be speed of rendering, and computational efficiency. Efficient database and database querying routines will be necessary for rapid evaluation and "sifting" of analysis information in real-time.

Enhancement of "Sectioning" Techniques: The existing techniques to extract and view a portion of the structure are slow and inadequate. New techniques need to be designed and implemented. These would allow the analyst to specify a combination of a range of different "sectioning criteria" -- including arbitrarily oriented enclosures, use of a response range (a 3-D analogue to contrasting), a response range coupled with materials and enclosures (for example, "...extract all elements in this enclosure that have effective stress values between 'x' and 'y' and belong to material type 'n'...".) Extraction by element index values or along mesh-lofting lines could also be implemented. The database must be efficient enough to allow these operations to be conducted on the fly. At present, massive duplication of data occurs when a subobject is "extracted" from a parent object. If the new techniques can be performed on the fly, only the parent data needs to be stored at any time. "Learning" functions would allow the analyst to return to a defined status at any time; a view status is stored in the form of a "path", and data is not duplicated.

Parallel Processing: Research is necessary in the area of the application of parallel processing to postprocessing. There are a number of computationally intensive processes that may be executed asynchronously while the interactive session proceeds. Also, the various modules of the program may reside on different processors of a parallel machine (this aspect is discussed in Section 5.3.)

Database Design: An efficient database design is vital for the implementation of most of the features described above. This is an area of research that requires immediate attention; it affects all sections of the design process from pre- to post-processing. The key aspects are the data structures and the database management system employed to extract information from the database. A geometric modeling system would define the geometry -- thus "freeing" it from being tied to the discretization information (the mesh).

APPENDIX A

DATA FLOW AND FILE MANAGEMENT SCHEMES

The computer programs mentioned in this appendix have been developed and are in use at the Program of Computer Graphics, Cornell University. The BOUNDARY EXTRACTOR, OUTLINE EXTRACTOR and translator programs for POSTPRO3D use information from the preprocessor (PREPRO3D) and analysis programs (FACS, ALICE, or GNOME for finite element solutions, and BEM3D for boundary element solutions). This appendix describes the flow of data between these programs. This flow is indicated in Figures A.1 and A.2 for finite element and boundary element analyses, respectively.

A.1 Description of Relevant Data Files

The following is a brief description of the important contents of the data files generated by the programs mentioned above. A detailed record-by-record description of the important files is available in internal documentation.

1) Finite Element Applications:

Preprocessing: PREPRO3D is a general-purpose, 3-D finite element preprocessor [37].

GEO:

* Coordinates of the nodes of the structure

INP:

* Analysis type
* Parameters controlling non-linear analyses
* Number of nodes, elements, load cases, etc.
* Nodal fixities
* Material data -- thicknesses, properties, etc.
* Element data -- element type, material type, connectivities, etc.

Analysis: A number of analysis programs are used -- FACS [37], ALICE, and GNOME (this program is under development at the Program of Computer Graphics in Cornell University.)

DSP:

* Three cartesian displacement components at each node

STR:

* Gauss point data -- number of points, coordinates, etc.
* Six cartesian stress and strain components for each load case or non-linear load step, at each gauss point of each element

PWP:

* Number of gauss points per element
* Pore water pressure for each load case or non-linear load step, at each gauss point of each element

Translator1 (TRANS1): TRANS1 is the first of the two translator programs for POSTPRO3D. It is used only for finite element analyses.

COO:

* Number of materials, nodes, elements, etc.
* Coordinates of the nodes
* Nodal connectivities and material type of each element

Boundary Extractor: This is a program that extracts the boundary faces from the finite element connectivities. The MAP and MAT files written only if the problem is a subobject or if there is more than one material type -- in these cases, there is a renumbering of some nodes.

MAT:

* Number of new nodes
* New element connectivities
* New node numbers and coordinates
* The node map - that maps the new node numbers to old ones

MAP:

* Number of nodes
* The node map - that maps the new node numbers to old ones

BOU:

* Number of boundary faces
* Face connectivities, element number, materila type, etc.
* Face normals

2) Boundary Element Applications:

The program used is BEM3D. This is a 3-D boundary element preprocessor, analysis, and postprocessor (postprocessing is performed on vector refresh Evans and Sutherland devices, and capabilities are limited [43])

BEM:

* Coordinates of the nodes
* Element connectivities
* Applied displacements and tractions (boundary conditions)
* Material information

DTS:

* Computed displacements and tractions at the nodes

STR:

* Cartesian stress and strain components at the nodes

SIF:

* Stress Intensity Factors (for all modes) along the crack fronts.

3) Programs Used by Finite/Boundary Element Applications:

Outline Extractor: The OUTLINE EXTRACTOR extracts the outline for a given structure. The outline is used by POSTPRO3D to "rotate" the structure.

OUL:

* Number of outlines
* Nodal connectivities
* Angle between adjacent faces

Translator2: TRANS2 is the second of the two translator programs for POSTPRO3D for finite element analyses. (TRANS_BEM is the translator used for boundary element analyses -- Figure A.2 shows the flow of data for boundary element analyses.) The output files of these translators are the POSTPRO3D files -- COR, PAR, and GPV.

COR:

* General analysis-type data
* Number of scalar, vector, and tensor responses
* Number of nodes, elements, materials, load cases, etc.
* Non-linear analysis data
* Coordinates of the nodes
* Element data -- element type, material type, connectivity, etc.
* Node-number maps that connect orginal numbers to POSTPRO3D numbers
* Element-number maps
* Renumbered nodes' data
* Parent data (for sub-objects) -- number of nodes, elements, etc.
* Material data -- types, properties, etc.

PAR:

* Number of scalars, vectors, tensors, load sets, materials, etc.
* Scalar response data for all load sets
* Vector response data for all load sets
* Tensor response data for all load sets
* Unaveraged response data at the nodes of each boundary face for all load sets

GPV:

* Number of elements
* Element gauss point response data - number of gauss points, and cartesian stress and strain tensor components for all load sets

A.2 File Management Schemes

Figure A.1 shows the file management scheme for finite element applications (only those files relevant to this discussion are shown in the figure.) The MAP and MAT files created by the BOUNDARY EXTRACTOR are deleted after being read by the OUTLINE EXTRACTOR and TRANS2, respectively. TRANS1 and TRANS2 are the two translator programs that convert data files from PREPRO3D, and the analysis programs to the final POSTPRO3D data files -- OUL, BOU, COR, PAR, and GPV. These POSTPRO3D data files have the same format for every problem, irrespective of the type of analysis used in the solution process.

Figure A.2 shows the file management scheme for boundary element applications. In this case the elements constitute the boundary and, hence, the BOUNDARY EXTRACTOR is not used. The displacements and tractions stored in the DTS file are used to compute cartesian tensor quantities at the nodes; these are stored in the STR file. If the problem has cracks embedded in it, the stress intensity factors are stored in the SIF file and read into the Boundary Element Translator (BEM_TRANS).

Section A.1 outlines the contents of the data files. There is a significant duplication of data because each of the programs uses a different database. An integrated CAD environment with a common shared database would eliminate this problem -- this is discussed in section 5.7.

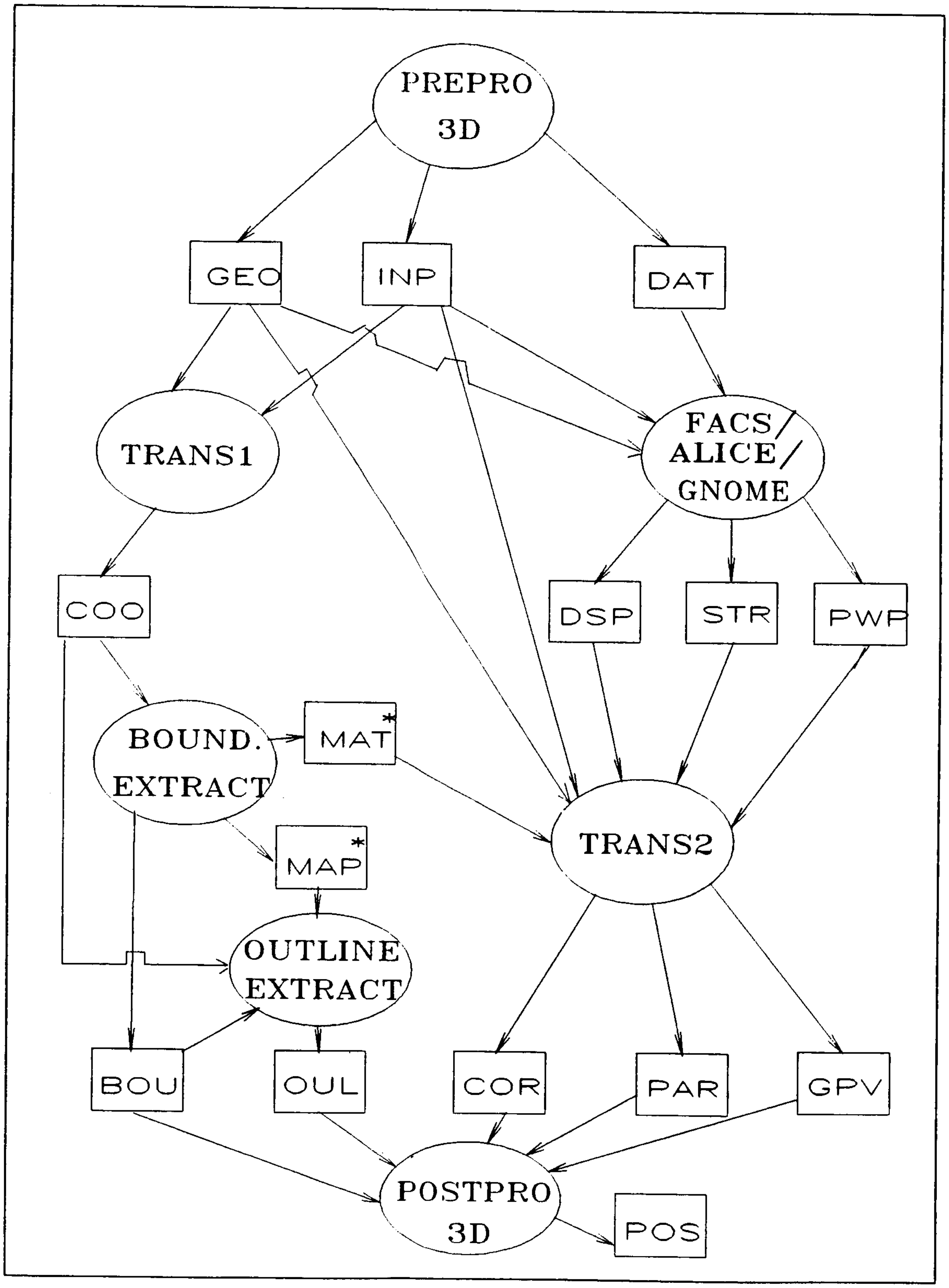


Figure A.1 File management scheme for finite element analyses

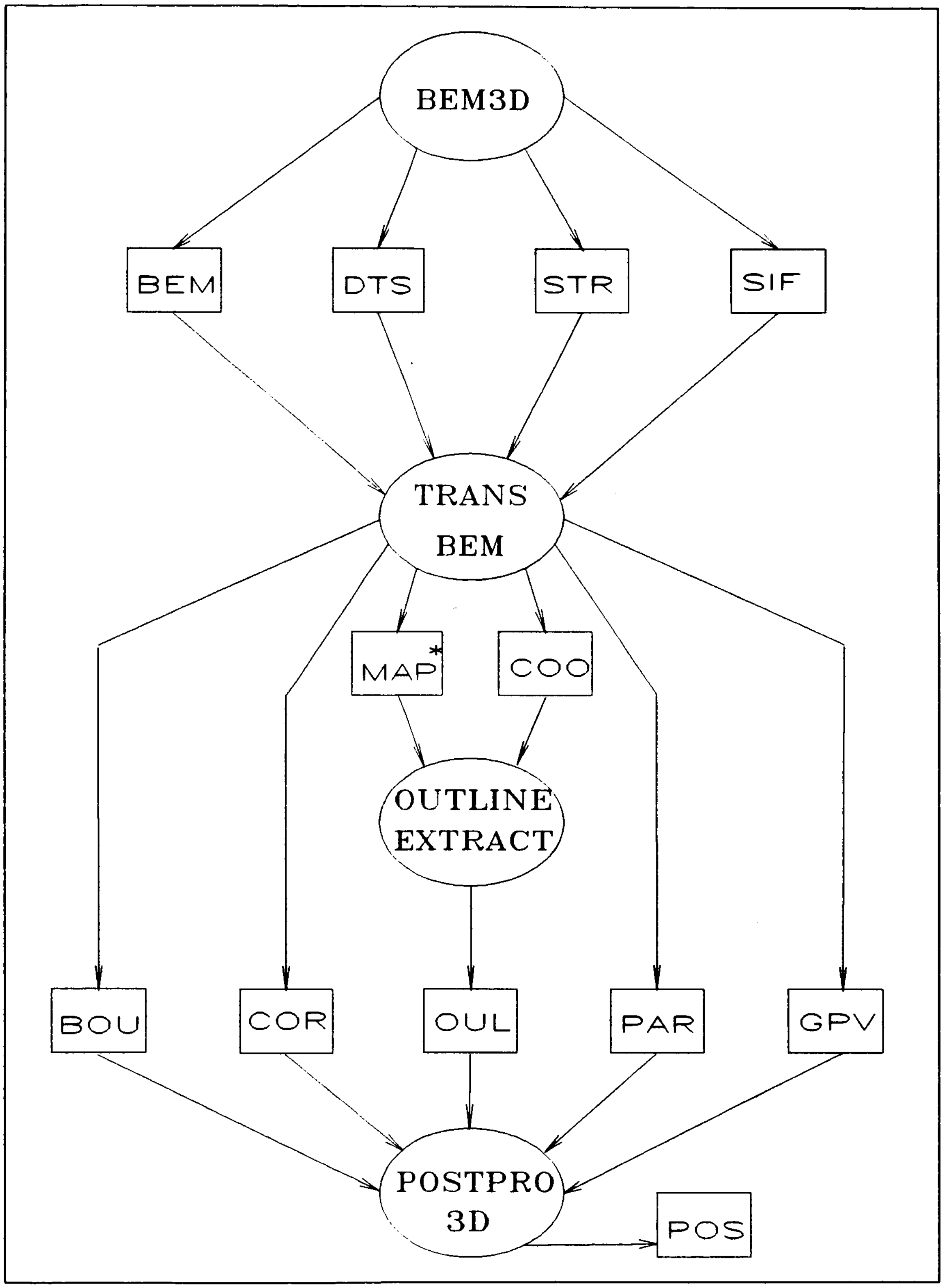


Figure A.2 File management scheme for boundary element analyses

APPENDIX B

MEMORY ALLOCATION AND RELATED PROBLEMS

A lack of sufficient memory space is one of the problems of most interactive graphics applications programs. Interactivity governs program execution and database management decisions; there is always a trade-off between the amount of information stored in a database (i.e. the amount of memory space required for the database) and the interactive performance of the program. Even if the machine has a large amount of addressible virtual memory, an excessively large database results in page faulting that drastically slows down interactive performance. On the other hand, a skeletal database (most required data are computed on the fly) relies on the computational power and I/O efficiency of the workstation for interactivity. This is a dilemma faced by most graphics applications programmers, and it is unfortunate that its solution depends mainly on machine characteristics and configuration.

The existing postprocessor has begun to experience problems that are "memory-based." As a program grows in application, so does its size. POSTPRO3D is written in VAX Fortran (specific to the VAX/VMS operating system run on machines manufactured by the Digital Equipment Corp.) In all versions of Fortran, there is a distinct difference between statically and dynamically declared memory. Variables "local" to a subroutine are normally statically declared -- especially if they are scalars or small arrays. As the number of subroutines increases, the statically declared portion of the memory also increases. At present, there are 403 subroutines in the program. Each has statically declared scalars and small arrays. The program is currently used at the Program of Computer Graphics in Cornell University in a minicomputer-host/ frame buffer environment, with approximately 2.5 gigabytes of addressible virtual memory space. But it is a time-sharing environment with various dynamic and static memory, and page faulting quotas -- a common computing scenario. For example, the statically declared memory quota for an interactive program is 2.5 megabytes; and the program has now run into problems associated with this quota. A trivial solution is to increase the memory quota available to each interactive program; this has a potential impasse as there is always a finite amount of available memory. This is a poor temporary solution to a problem that can be solved from a program-design point of view.

At present, the large amount of memory required by the program to store the {A} and {B} arrays, the contrasting information [Section 2.2.3.2], and "scratch" space for on-the-fly computations is dynamically declared. Local "scratch" variables are statically declared. This local memory must be controlled and managed by a central Memory Managing Module; no statically declared memory must exist. At program start-up, a block of virtual memory is allocated for local scratch variables. The Memory Manager takes control of allocating this memory, upon request. A similar approach is used by Gattass [42] in the development of a program for large displacement, dynamic analysis of steel frames. In this manner, memory is used efficiently -- space no longer needed is "reclaimed" by the Memory Manager for re-use by other subroutines.

The Fortran concept of "volatile" memory can be exploited. By default, statically declared memory is preserved; volatile memory is re-usable by other subroutines -- decisions to re-use "volatile" memory are made by the operating system. But usually, more memory is volatile than needs to to be preserved; so the default ought to be "volatile" and not preserved memory.

In a single-user workstation environment, this problem is less severe. The differentiation between statically and dynamically declared memory becomes less distinct -- the advantage of the latter being that it can be re-used when necessary. In this environment, the problem of page faulting becomes important. At present, hardware used for memory access off hard disks in workstations is usually inferior to that used on the larger disks attached to mainframes or mini-computers. Thus I/O with these disks is slow and excessive page faulting drastically reduces interactivity. As processors become faster, it is advantageous to store less information in the database, and compute more of the required data on-the-fly.

An problem associated with these suggestions is that the choice of the solution depends on the characteristics of the computing environment. This is a consequence of the fact that hardware environments lack sufficient standards. High-level graphics workstation manufacturers now recognize this problem, and it is reasonable to expect more hardware standards for these machines in the future.

REFERENCES

1. Bailey, B. C., "Unification of Color Postprocessing Techniques for Three-dimensional Computational Mechanics," M.S. Thesis, Cornell University, Ithaca, New York, August 1985, University Microfilms International, Ann Arbor, Michigan.

2. Shephard, M. S., "Finite Element Grid Optimisation - A Review," Finite Element Grid Optimisation, Eds., Shephard, M. S., and Gallagher, R., ASME Special Publication PVP-38, pp. 1-13, N.Y., 1979.

3. Peano, A., and Riccioni, R., "Automated Discretisation Error Control in Finite Element Analysis," World Congress on Finite Elements, October 1978.

4. Babuska, I., "The Self Adaptive Approach in the FEM," II MAFELAP Conference, Brunel University, 1975.

5. Crandall, S. H., Engineering Analysis, A Survey of Numerical Procedures, McGraw Hill Book Co., New York, 1956, pp. 171-173, 269-270.

6. Babuska, I., Rheinboldt, W. C., "Analysis of Optimal Finite Element Meshes in R1," Mathematics of Computations, Vol. 33, pp. 435-463, 1978.

7. Ramey, G. E., and Krishnamurthy, N, "Error Estimates for Conforming Finite Element Solutions," Computers and Structures, Vol 4, 1974, pp 1207-1222.

8. Hajjar, J. F., "General-purpose Three-dimensional Color Postprocessing for Engineering Analysis," M. S. Thesis, Cornell University, Ithaca, New York, January 1985, University Microfilms International, Ann Arbor, Michigan.

9. Anderson D. C., "Closing the Gap : A Workstation-Mainframe Connection," CIME, Vol. 4, No. 6, May 1986.

10. Barris, W. C., Riley, D. R., "Programmer-friendly Graphics Libraries," CIME, Vol. 5, No. 1, July 1986.

11. Golub, G. H., Van Loan, C. F., Matrix Computations, publ. The Johns Hopkins University Press, Baltimore, Maryland, 1983.

12. State-of-the-Art Surveys on Finite Element Technology, Ch. 9, Eds. Noor, A K, and Pilkey, W D, pubs. ASME, pp 297-324.

13. Kelly, D. W., Gago, J. P. de S. R., and Zienkiewicz O. C., "A Posteriori Error Analysis and Adaptive Processes in the FEM: Part 1 - Error Analysis," IJNME, Vol 19, pp 1593-1619, 1983.

14. Kelly, D. W., Gago, J. P. de S. R., and Zienkiewicz O C, "A

Posteriori Error Analysis and Adaptive Processes in the FEM: Part 2 - Adaptive Mesh Refine," Int. j. numer. methods eng., Vol 19, pp 1593-1619, 1983.

15. State-of-the-Art Surveys on Finite Element Technology, Ch. 3, Eds. Noor, A. K., and Pilkey, W. D., pubs. ASME, pp 297-324.

16. Babuska, I., and Rhienboldt, W. C., "On the Reliability and Optimality of the FEM," Symposium on Future Trends in Computerised Structural Analysis and Synthesis, Wash. D C, Oct 30, 1978.

17. Babuska, I., and Rhienboldt, W. C., "Error Estimates for Adaptive F. E. Computations," SIAM J. Numerical Analysis, Vol 15, No 4, Aug 1978, pp 736-754.

18. Babuska, I., and Rhienboldt, W. C., "On the Reliability and Optimality of the FEM", Computers and Structures, Vol 10, 1979, pp 87-94.

19. Babuska, I., and Miller, A., "The Postprocessing Approach in the FEM - Part 3: A Posteriori Error Estimates and Adaptive Mesh Selection," Int. j. numer. methods eng., Vol. 20, pp 2311-2324, 1984.

20. Zienkiewicz, O. C., and Morgan, K., Finite Elements and Approximations, Ch. 8, pubs John Wiley and Sons, N.Y., 1983.

21. Melosh, R. J., and Killian, D., "Finite Element Analysis to Attain Prescribed Accuracy," 2nd Nat. Sym. on Computers in Structural Analysis and Design, Washington Univ, March 1976.

22. Melosh, R. J., and Marcal, P. V., "An Energy Basis for Mesh Refinement of Structural Continua," Int. j. numer. methods eng., Vol 11, No 7, pp 1083-1092, 1977.

23. Peano, A., "Self-Adaptive Convergence at the Crack_tip of a Dam Buttress," Proc. Int. Conf. on Numerical Methods in Fracture Mechanics, Swansea, Wales, Jan 1978.

24. Peano, A., "Self-Adaptive F. E. in Fracture Mechanics", Computer Methods in Applied Mechanical Engineering, Vol. 11, No. 4, 1978.

25. Mehta, A., "P-Convergent F. E. Approximations in Fracture Mechanics", Ph.D. Thesis, Washington Univ., May 1978.

26. Shephard, M. S., "Finite Element Grid Optimisation with Interactive Computer Graphics," Ph.D. Thesis, Dept. of Structural Engineering, Cornell Univ., Jan 1979, University Microfilms International, Ann Arbor, Michigan.

27. Shephard, M. S., et al., "The Synthesis of Near-Optimum F. E. Meshes with Inter-active Computer Graphics," Int. j. numer. methods eng., Vol 15, 1021-1039, 1980.

28. Carey G. F., and Humphrey D. L., "Mesh Refinement and Iterative Solution Methods for F. E. Computations," Int. j. numer. methods eng., Vol 17, pp 1717-1734, 1981.

29. Carey G. F., "Adaptive Refinement in Nonlinear Fluid Problems", J. of Computational Methods in Applied Mechanics and Engg., Vol. 17/18, Part III, pp. 541-560, March 1979.

30. Carey G. F., and Humphrey D. L., "Adaptive Mesh Refinement Algorithm Using Element Residuals," TICOM Report 78-1, 1978, Texas Inst. for Computational Mechanics, Univ. of Texas.

31. Barnhill, R. R., et al., "Computable F E Error Bounds For Poisson's Equation," Short Communications, Int. j. numer. methods eng., Vol 11, pp 593-603, 1977.

32. Reddy C. T., "Discussion of - Computable Error Bounds For Poisson's Equation," Letters To the Editor, Int. j. numer. methods eng., Vol 11, pp 1629, 1977.

33. Barnhill R. R., Reply to the above letter, Int. j. numer. methods eng., Vol 11, pp 1629, 1977.

34. Steven G. P., "A Post Solution Error Estimator Using a Least Squares F. E. Solution Criterion," Proc. of the Int. Conf. on FEM, Beijing, China, pubs. Gordon and Breach, Science Publishers, NY, 1982.

35. Haber R., "A Consistent F. E. Technique for Recovery of Distributed Reactions and Surface Tractions", Int. j. numer. methods eng., Vol. 21, No. 11, pp. 2013-2025, 1985.

36. Abel, J. F., Panthaki, M. J., Wawrzynek, P. A., "Interactive Graphics and Analysis Accuracy," Reliability of Methods for Engineering Analysis, Proc. of the International Conference, Swansea, U.K., July 9-11, pp. 305-317, 1986.

37. Han, T., "Adaptive Substructuring and Interactive Graphics for Three-dimensional Elasto-plastic Finite Element Analysis," Ph.D. Thesis, Dept. of Structural Engg., May 1984, University Microfilms International, Ann Arbor, Michigan.

38. Baumgart, B. G., "A Polyhedron Representation for Computer Vision," AFIPS Proc., Vol. 44, pp. 589-596, 1975.

39. Weiler, K., "Edge-based Data Structures for Solid Modeling in Curved-surface Environments," IEEE Computer Graphics and Applications, Vol. 5, No. 1, pp 21-40, 1985.

40. Woo, T. C., "A Combinatorial Analysis of Boundary Data Structure Schemata", IEEE Computer Graphics and Applications, Vol. 5, No. 3, pp 19-27, 1985.

41. Wawrzynek, P. A., "Interactive Finite Element Analysis of Fracture Processes: An Integrated Approach," M.S. Thesis, Dept. of Structural Engg., Cornell Univ., May 1987, University Microfilms International, Ann Arbor, Michigan.

42. Gattass, M., "Large Displacement, Interactive-adaptive Dynamic Analysis of Frames," Ph.D. Thesis, Dept. of Structural Engg., Cornell Univ., May 1982, University Microfilms International, Ann Arbor, Michigan.

43. Perucchio, R., "An Integrated Boundary Element Analysis System with Interactive Computer Graphics for Three-dimensional Linear-elastic Fracture Mechanics," Ph.D. Thesis, Dept. of Structural Engg., Cornell Univ., January 1984, University Microfilms International, Ann Arbor, Michigan.

44. Zienkiewicz, O. C., Zhu, J. Z., "A Simple Error Estimator and Adaptive Procesure for Practical Engineering Analysis," Int. j. numer. methods eng., Vol. 24, No. 2, February, 1987.