Final Report

Hierarchical Strategy for Rapid Analysis Environment

NASA Grant NAG-1-01080
February 1, 2001-January 31, 2003

John Whitcomb
Center for Mechanics of Composites
Department of Aerospace Engineering
Texas A&M University
May, 2003

Final Report

Hierarchical Strategy for Rapid Analysis Environment

NASA Grant NAG-1-01080
February 1, 2001-January 31, 2003

John Whitcomb
Center for Mechanics of Composites
Department of Aerospace Engineering
Texas A&M University
May, 2003

Table Of Contents

Table of Figures 3

Introduction 4

Background 5

The HS4RAE System – Overview 10

HS4RAE – Concepts 16

HS4RAE – Implementation 20

HS4RAE Graphic User Interface 25

Example I 30

Example II 37

Conclusion and Future work 40

Appendix 41

HS4RAE Scripting 41
Mesh Data File Format 47

References 48

Summary of hyperlinked documentation 51

Table of Figures

Figure 1: Inheritance tree and its different views 12
Figure 2: Interface to external software 14
Figure 3: Mesh association (Inheritance of geometry) 17
Figure 4: Joining components using digital glue 18
Figure 5: Buffering of solutions 19
Figure 6: Class diagram of the hierarchical definition module 21
Figure 7: Data flow chart 22
Figure 8: HViewer screenshot 28
Figure 9: VB GUI screenshot 29
Figure 10: Example I – load and constraint conditions 30
Figure 11: Model LS Analysis – Stress Plot (σ_{xx}) 31
Figure 12: Model LC Analysis – Stress Plot (σ_{xx}) 31
Figure 13: Model H1 analysis results 32
Figure 14: Screenshot of Plot2002 GUI (with model H2 results) 33
Figure 15: Stress Plot of component F1c (σ_{xx}) 33
Figure 16: Screenshot of Plot2002 GUI (with Model F2c) 34
Figure 17: Model script for Example-I 35
Figure 18: Model Handler script for Model H1 36
Figure 19:Example II – load and constraint conditions 37
Figure 20: Screenshot of GUI with fuselage panel 38
Figure 21: Model and component stress plots for the refined model of the panel (σ_{xx}) ... 39

Introduction

A new philosophy is developed wherein the hierarchical definition of data is made use of in creating a better environment to conduct analyses of practical problems. This system can be adapted to conduct virtually any type of analysis, since this philosophy is not bound to any specific kind of analysis. It provides a framework to manage different models and its results and more importantly, the interaction between the different models. Thus, it is ideal for many types of finite element analyses like global/local analysis and those that involve multiple scales and fields.

The system developed during the course of this work is just a demonstrator of the basic concepts. A complete implementation of this strategy could potentially make a major impact on the way analyses are conducted. It could considerably reduce the time frame required to conduct the analysis of real-life problems by efficient management of the data involved and reducing the human effort involved. It also helps in better decision making because of more ways to interpret the results. The strategy has been currently implemented for structural analysis, but with more work it could be extended to other fields of science when the finite element method is used to solve the differential equations numerically.

This report details the work that has been done during the course of this project and its achievements and results. The following section discusses the meaning of the word hierarchical and the different references to the term in the literature. It talks about the development of the finite element method, its different versions and how hierarchy has been used to improve the methodology. The next section describes the hierarchical philosophy in detail and explains the different concepts and terms associated with it. It goes on to describe the implementation and the features of the demonstrator.

A couple of problems are analyzed using the demonstrator program to show the working of the system. The two problems considered are two dimensional plane stress analysis problems. The results are compared with those obtained using conventional analysis.

The different challenges faced during the development of this system are discussed. Finally, we conclude with suggestions for future work to add more features and extend it to a wider range of problems.

Background

The term "hierarchical" is used to refer to different types of ideas in the literature. It is important to know what we mean by this in order to understand the uniqueness of this philosophy and the differences when compared to other methodologies out there. This section talks about the relevance of hierarchy, the finite element method in general and some of its various versions especially those that deal with some sort of hierarchy. It mentions the bottlenecks faced when conducting an analysis and other common problems facing the finite element method. Attached to the end of this report is a brief introduction to the hyperlinked version of the documentation that is being developed. This electronic form of the documentation will be more comprehensive than is practical with a "paper oriented" report.

There is an aspect of hierarchy/inheritance in both science and almost every thing we see in day-to-day life. The whole theory of evolution of life itself is based upon inheritance. In the same way, the human race can be considered a huge hierarchy with every human being related to one another in some way albeit along a rather long path. The most common programs that we use these days are built using object–oriented programming, which is based on the idea of inheritance. Fields such as pattern recognition, artificial intelligence and computer networks make use of hierarchy extensively [1]. Hierarchical networks are also implemented in a class of control systems that can be used for space navigation [2].

Oden et al. [3] introduced the concept of hierarchical modeling as an approach to overcome the difficulties of multiscale modeling. In this methodology, a hierarchy of descriptions of the physics of the problem is first set up, ranging from the coarsest possible description to the most detailed description contained in the class of models. Thus in this context, it deals with scale hierarchy.

In this work, the term hierarchical strategy is used to convey the idea that analysis models can be organized and managed hierarchically in order to rapidly setup a new analysis model. New models are derived from the base model whose information is either inherited or overridden by the new model. Thus, we are dealing with a hierarchy of models.

The global/local analysis method is inherently hierarchical in nature if you consider the local model is actually a more refined part of the global model. Thus, it makes sense to manage data in a hierarchical form to tap into the full potential of the global/local analysis method.

Much of the analyses carried out by industry and academia uses global/local Analysis. The global/local technique in some sense has been around since before the finite element method was developed. This technique comes in very handy when designing large complex structures such as aircraft and automobiles where large finite element models of these structures are utilized. These models are useful in obtaining a more accurate response to load conditions or for optimizing different characteristics. But there is a practical limit to the amount of refinement that these models can hold simply due to the fact that the computational cost for such an endeavor would be too much. In such cases, separate analysis is done on localized regions of the structure where the refinement is high enough to obtain a reliable design. Global/local analysis is also used in fracture mechanics to calculate the stress intensity factor for cracks [4]. This technique is also used to compute the effective properties of composite laminates [5]. Iterative global/local finite element analysis is found to be less taxing on computer memory requirements [6]. The global local method is also used in the failure analysis of textile composites [7].

A number of strategies have been developed to enhance finite element solutions in the regions of high gradients. In the h-method, the finite element mesh is refined by keeping the elements of the same order and subdividing them. In the p-method [8], the same mesh is retained but the order of the interpolation function (approximation) is increased. The third method (h-p) is a combination of the first two strategies. The h-p version uses a simultaneous increase of the polynomial degree and mesh refinement. The rates of convergence for the h-, p- and h-p versions in terms of the number of degrees of freedom have been identified and quantified by Babuska and Szabo [9].

The hierarchical finite element method (or HFEM) [10, 11] belongs to the p-version of the FEM. In HFEM, the order of approximation (interpolation) functions are hierarchically increased – the new order of approximation is based on the lower order that is previously constructed.

A version of the finite element called the s-version, where s stands for superpositioning, was introduced by Fish [12]. The basic idea of this method is that a portion of the finite element mesh in which steep gradients are indicated by the solution is overlaid by a patch of higher-order hierarchical elements. Fish and Guttal [13] developed the s-version of the finite element method for laminated plates and shells. In their technique, the global domain is idealized using a 2-D Equivalent Single Layer (ESL) model and the location of the critical regions where a Discrete Layer (DL) model is needed is identified using Dimensional Reduction Error (DRE) indicators. These regions are superimposed by a stack of 3D elements (DL model) and thus both the local and global effects are predicted. Fish [14] also used the s-version to hierarchically model discontinuous fields such as for crack propagation. This is achieved by overlaying portions of the finite element mesh where discontinuities need to be embedded with a finite element mesh that is discontinuous across the crack. This also proves to be computationally efficient due to the hierarchical nature of the method where the base mesh can be fixed and only the super-imposed mesh needs to be modified.

Another strategy to enrich finite element solutions is by adding special shape functions that are known to approximately model the behavior of the exact solution. This idea was introduced by Mote [15] who developed a global-local Finite Element where a combined global and local dependant variable representation couples the conventional and finite element Ritz methods.

Voletia et al [16] address the use of global/local FEM to analyze large-scale periodic structures made up of multi-material composite systems. They explain two different techniques – the specified boundary method and the multi-point constraint method. The global/local FEM techniques prove to be faster especially as the size of the problem increases.

Sun and Mao [17] proposed a refined global local finite element analysis method which involves 3 steps to improve the efficiency of the analysis. The global analysis of a coarse mesh provides a displacement solution, which is then used in the local analysis for computing detailed stresses using refined meshes. Finally, a refined global analysis is conducted to improve the accuracy of both displacements and stresses.

Whitcomb [18] described the process of iterative global/local finite element analysis where the accuracy is retained by using an iterative procedure to enforce equilibrium between the global and local regions. Babuska et al [19], Reddy and Robbins [20] have done some work on the reliability, convergence and accuracy aspect of global/local techniques. N.F. Knight, Jr. et al [21] present a global/local analysis methodology for obtaining the detailed stress state of structural components. Noor et al [22] describes two predictor-corrector procedures for the accurate determination of the global as well as detailed response characteristics of plates and shells.

Ghosh et al [23] proposed a hierarchical multi-scale computational model for damage in composite materials. Analysis is done at the structural and micro-structural scales. The microscopic analysis is conducted using a voronoi cell finite element model (VCFEM) while a conventional displacement based FEM code executes the macroscopic analysis. A simple plate with a hole problem was analyzed hierarchically using three different refinements levels.

Noor et al [24] used hierarchical sensitivity analysis to identify the parameters that have the most effect on the non-linear response of composite structures. Their modeling approach used for multilayered panels can be divided into different categories that cover a wide range of length scales from local to global structural response: detailed micromechanical three-dimensional continuum models, quasi-three-dimensional models, and two-dimensional plate and shell models. The nonlinear response of the structure is dependent on a hierarchy of interrelated geometric and material parameters at these different categories. The sensitivity of the response to variations in these parameters at each level provides insight into the importance of the parameters and helps in the development of materials to meet certain performance requirements. Ransom and Knight [25] discuss a methodology for the global/local stress analysis of composite panels. Noor et al [26] discusses different global/local methodologies and their application to non-linear analysis.

J. Fish et al [27] developed a hierarchical version of the composite grid method (denoted as HFAC), which exploits the solution of the shell model in studying local effects via a 3D model solid model. It is hierarchical in the sense that information from the analysis of

an equivalent single layer (ESL) model is exploited in the resolution of local effects using a discrete layer (DL) model. The multigrid and composite grid methods [28, 29, 30, 31, 32, 33] are a widely used hierarchical global/local strategy. Some works on methods based on hierarchical decomposition of the approximation space are described in [34, 35, 36, 37, 38].

Even though there are various versions and methodologies of the finite element method, there are several basic issues that can cause bottlenecks during an analysis that does not deal specifically with a kind of methodology but with the generic finite element method. As the complexity of the problem increases, analysis models increase in size and the amount of data that needs to be handled becomes overwhelming. When designing a structure, it is common to make frequent modifications to the model during the process. A number of analyses are conducted before adequate information can be obtained to make a good decision regarding a final design. In such cases, the ability to use data from different models simultaneously becomes a major advantage. Thus, data management and control is a big issue that needs to be dealt with. This calls for interaction between models and at different detail levels. Also, time is an important factor and it is always advantageous to be able to setup analysis model quickly and efficiently. One way to achieve this is by using the computer to automate as many steps as possible that are involved in generating an analysis model. Some of these functions are boundary detecting and matching. But letting the computer do this work instead of the user manually entering the information, a lot of time can be saved and analyses can be conducted efficiently. It is these and other problems that we try to address in this project. We develop an environment for rapid analysis using hierarchical description of models and efficient and robust data control mechanisms to solve problems quicker as well as reliably.

The HS4RAE System – Overview

The term HS4RAE stands for Hierarchical Strategy for Rapid Analysis Environment. From here on, it is also referred to as the Hierarchical System. In this section, an overview of the hierarchical strategy and its philosophy is given. Some of the challenges addressed in this effort are also mentioned. Comparison is also made to other software that deals with this problem.

The hierarchical system consists of the following key components:

- Hierarchical definition module: this module is made up of the different classes and functions that implement the inheritance and storage of hierarchical data.
- Visualization tool: for interpreting the results of the analysis.
- Scripting Language: it is used to describe the relationships between different models. The user can issue commands to the system using the scripting language. It can also be used to maintain persistence of data.
- Graphical User Interface (GUI): this is another way for the user to interact with the hierarchical system. Using the GUI, the user can interact with the hierarchical system in real-time with the help of the keyboard and mouse.
- Import/Export functions: these set of functions allow the transfer of data between the hierarchical system and other external software like mesh generators and FEA programs like FEMAP, ABAQUS etc.
- Solver: This is a set of classes and functions that are used to numerically solve the set of equations defined by the finite element model.

At the core of the Hierarchical Strategy is the inheritance tree. It is used to describe how different analysis models can be organized and managed hierarchically to rapidly create a new analysis model. The hierarchy exists only to express the inheritance relationships between models. New models are derived from the base model whose data is inherited, overridden or expanded by the new model. This way the derived model can have new characteristics that are different from its base models. Data in this context could mean anything from geometric mesh information and load conditions or boundary conditions to even solutions of analysis models. This additional or differentiating data that is used to derive a new model from its base model is defined to be a component. At present, we

have implemented only inheritance of geometry and therefore it makes more sense to use the word component to define the differentiating data that results in the creation of a new model.

Each node in the inheritance tree is a model. This model is described in terms of all the components along its model path. The model path is the shortest route which links the parent node in the inheritance tree to the current model. Figure 1 illustrates the meaning of model path of a model. In this case, the model path for the model FRCd is the route traced by the models -LC, H2, H1, H1C, H1RC and finally FRCd. Any other route would require retracing through a model that had already been covered by the path. Thus, the derived model FRCd is a combination of the component at FRCd and the components of all the models in its model path just mentioned. The figure shows two different views of the inheritance tree. The component view shows the corresponding components at the nodes in the inheritance tree. In the model view, each node in the tree is associated with a complete model.

Existing commercial finite element analysis software that used some kind of hierarchy was compared to the philosophy described in this work. In DesignSpace by AnSys [39], the ability to combine components to build models is limited in the sense that each hierarchy (or 'tree structure') represents a single model. While in the Hierarchical System, each node in the model hierarchy defines a complete model and not just a component. The hierarchical system provides a framework that has the 'intelligence' for building a hierarchy of models. SIMBA (Simulation Manager and Builder for Analysts), developed by Sandia Labs [40], also builds FE models from various components but it does not address data flow between different models in the hierarchy. The hierarchical system deals with a collection of models that have the ability to interact, communicate, and pass information with each other.

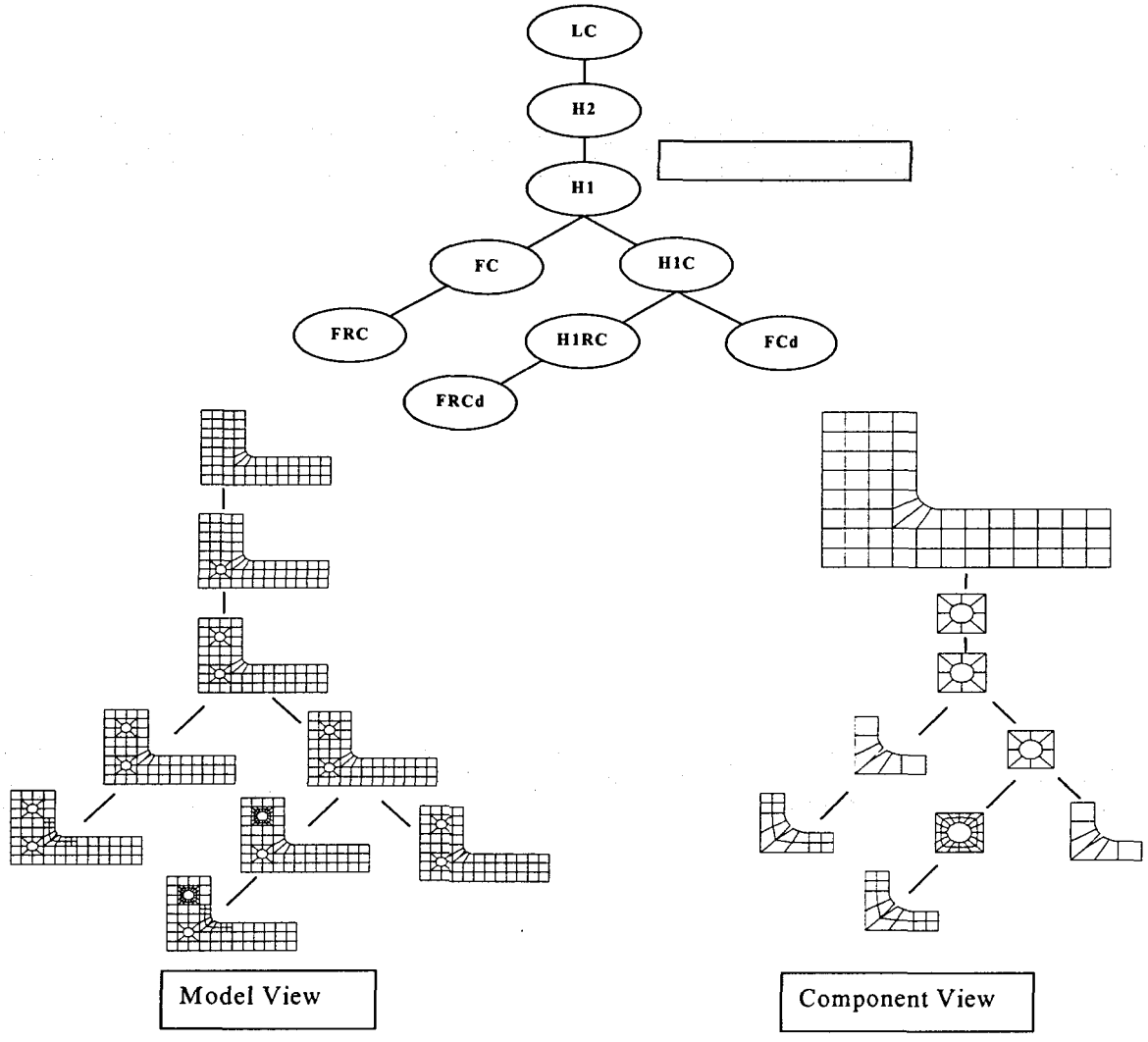

Figure 1: Inheritance tree and its different views

The ability to easily pass data between models makes it ideal for adapting it to different types of analysis methodologies. This hierarchical strategy can be used for different approaches such as iterative global/local analysis, submodeling, substructuring, multiscale and multi-physics analysis. These schemes require different types of communication between the global and local models. The important distinction is that this philosophy provides the framework for managing hierarchical models and efficient data flow between analysis models. Therefore, this framework has the potential to be used to implement various types of analysis problems. Since it imposes no restriction on the type of problem that can be analyzed, the system can be used for any problem that can

be solved by a methodology such as FEM to conduct structural analysis, progressive failure analysis, repair or optimization.

One of the initial tasks of this project was to identify and develop hierarchical data structures to define models by incorporating inheritance. This has been implemented and the different classes and functions that were developed will be explained in the later sections. This kind of hierarchical organization and the ability to communicate between models in the hierarchy helps reduce the difficulty in setting up and running models as well as managing the results for the various cases, thus reducing the timeframe and human workload in conducting analyses and digesting its results.

A major task for the hierarchical system is information management. Since the system would be managing a hierarchy of models, it would have to store the information for all models. At the same time, there is no restriction on the size of the model. Thus, the system should have a robust and efficient data management strategy that can handle numerous as well as large datasets.

Another important feature is the ability to interface with other software. There is a lot of legacy software that are present in the market and each of them might have its own forte. It would be unwise to assume that all kinds of analysis work can be conducted within this environment itself. It would be advantageous to make use of special features that other software possesses. This is possible by exporting data into another format that can be understood by other commercial FEA software using some interface function. Presently, it is possible to export mesh data and other model information to the commercial software FEMAP by writing FEMAP Neutral files. In this way, the advanced mesh generation, analysis and post-processing features of the FEMAP software can be made use of. This capability can be extended to export analysis model information to other commercial FEA software like NASTRAN, ANSYS or software that use the EXODUS [41] Database Format. When exporting an analysis model to external software that is typically not hierarchically defined, a model has to be set up by joining the components in such a way that the external software is able to 'understand' because all the data is stored in a hierarchical form within the system. Similarly, when external software is used to conduct the analysis, the results are passed back into the hierarchical system through the interface,

which maps the results back into the hierarchical format. This way the user is not forced to use only this system and can make use of other features that are available in commercial software. Figure 2 gives a block diagram illustrating how the hierarchical system interfaces with the external software. ALPHA-HS denotes the solver resident within the Hierarchical System.

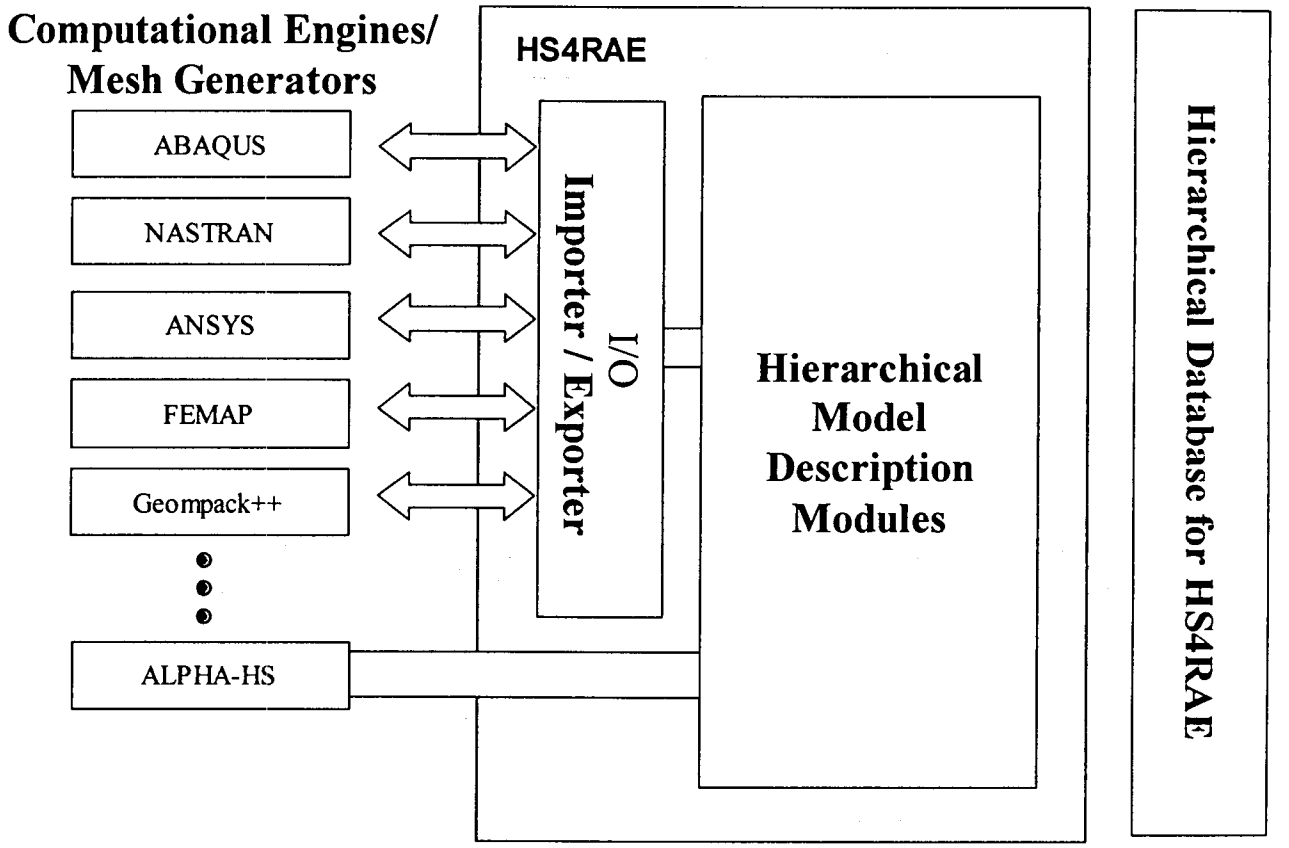

Figure 2: Interface to external software

One of the key components in an analysis environment is the visualization of the models and the results of the analyses. The visualization tool in the hierarchical system is called Plotter2002. It is capable of plotting the hierarchically defined models. The tool has various visualization options to display the results to the user. The user also has the ability to select elements in the model using the mouse. A more detailed explanation of the tool and its features is given in the section HS4RAE – Implementation. In addition to the existing features, the capability to compare results of analyses would be useful in making judgments for the design of a structure. The system should be able to provide

various options for comparing results such as change in stress distributions between multiple models. Another useful feature would be to allow the user to enter formulas for visualization of a dataset, or maybe a combination of more than one dataset. These features have not been implemented as yet.

The scripting language is an important feature that forms the link between the hierarchical system and the user. The user inputs data and issues commands to the system using the scripts. Like almost all software, the need arises to save the data to the disk for later reference or for resuming the work at a certain point in time. This can be achieved using the scripting language. The hierarchical models can be saved to the disk using the scripts and the scripts can be read by the system later on to load the models back into the memory. The syntax of the scripting language can be found in Appendix- HS4RAE Scripting.

The following section gives more in-depth information related to the different concepts and implementation of the Hierarchical System.

HS4RAE – Concepts

This section explains the different concepts involved in this philosophy. One of the major tasks involved in this effort is the design and implementation of the inheritance mechanism. Another important task is boundary detection and matching. It also discusses about the 'digital glue' that is used to join the components to form a complete analysis model. When analyses are conducted on different models in the same hierarchical tree, it is common to share a component between two or more models. This brings up the issue of managing the results of the analysis and mapping them back the components. This is achieved by buffering of solutions and is explained later in this section.

Considerable effort went into designing a robust as well as efficient mechanism for implementing inheritance, which is the essence of the hierarchy. That means the ability to derive models by inheriting, overriding and expanding the information defined in the base model. Efficient data flow mechanisms are required and complex recursive functions were developed that traverse the hierarchical tree to implement this. This kind of recursive strategy can be used to access data at any node. The beauty of the recursive strategy is that it can be used to perform any task on the models or the tree without making modifications to the mechanism. This mechanism gives a model in the hierarchy the ability to 'interrogate' another model for information.

In keeping with the main objective of this project, that is to conduct rapid analysis, it is desired to automate as many functions as possible.

One of the functions that plays an important role is the boundary detection and matching of a mesh. This is a major part in the process of deriving a new model from a base model. When a component is used to derive a new model from a base model, the component mesh replaces the corresponding region in the mesh of the base model. This is explained in Figure 3 using a simple case involving only two components in the model path. The new component deactivates all the elements (and nodes) that it would replace in all the components in its model path. This brings up the use of a term called 'collective mesh'. The collective mesh is used to describe the mesh of the derived model, which is built from the collection of components in its model path. The sub-routines to detect and match boundaries on two-dimensional meshes have been developed and implemented.

When dealing with meshes in three-dimensional space, the problem becomes more complicated. For now, we are dealing with 2-dimensional meshes and the implemented functions perform well for these cases.

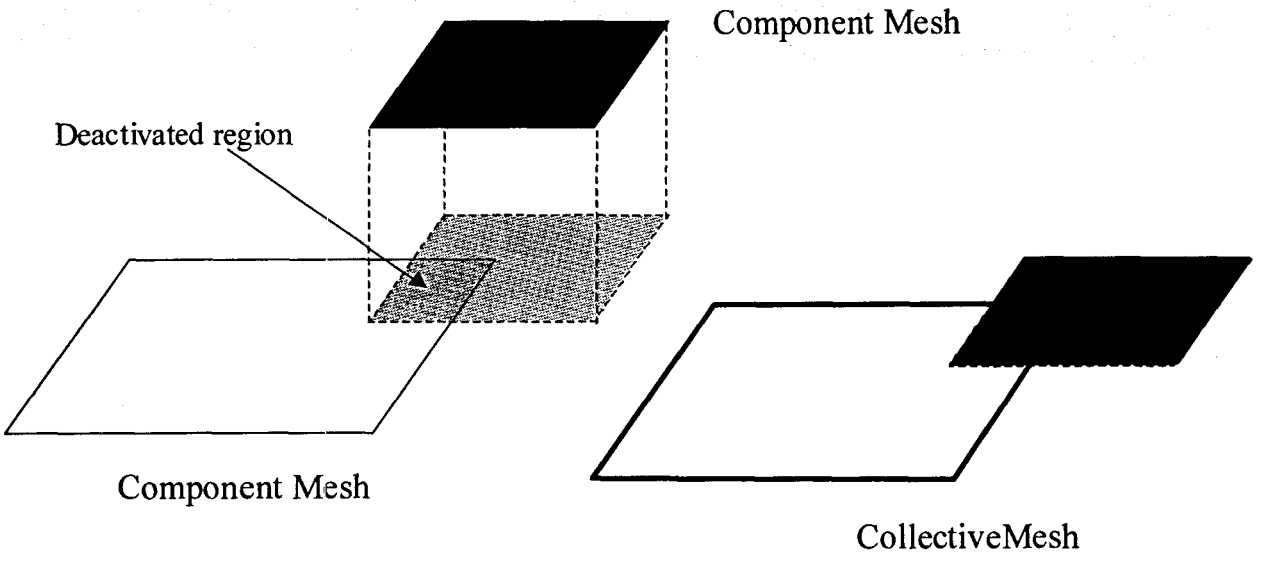

Figure 3: Mesh association (Inheritance of geometry)

In order to setup a complete analysis model, we have to 'join' the different components in its model path to create a combined model. There are a few methods in the literature to achieve this. One of the methods is to use multi point constraints [16] to constrain the degrees of freedom on the boundaries of the child model to the parent model. This is an exact satisfaction of compatibility using the interpolation functions. Another method is to use interface elements (developed by Ransom, Aminpour et al [42-44]), which uses a variational method to approximately satisfy compatibility at the interface. The first method has been implemented in this system. The multi point constraints act as a 'digital glue' to join the different components in the model tree. Figure 4 illustrates the use of digital glue to combine three components. It can be seen from the figure that a node in a model that is situated bottom most in the inheritance tree can be constrained to a node in a model multiple 'generations' up the hierarchy and not necessarily to the immediate parent model. This is implemented by using recursive functions to traverse the model path.

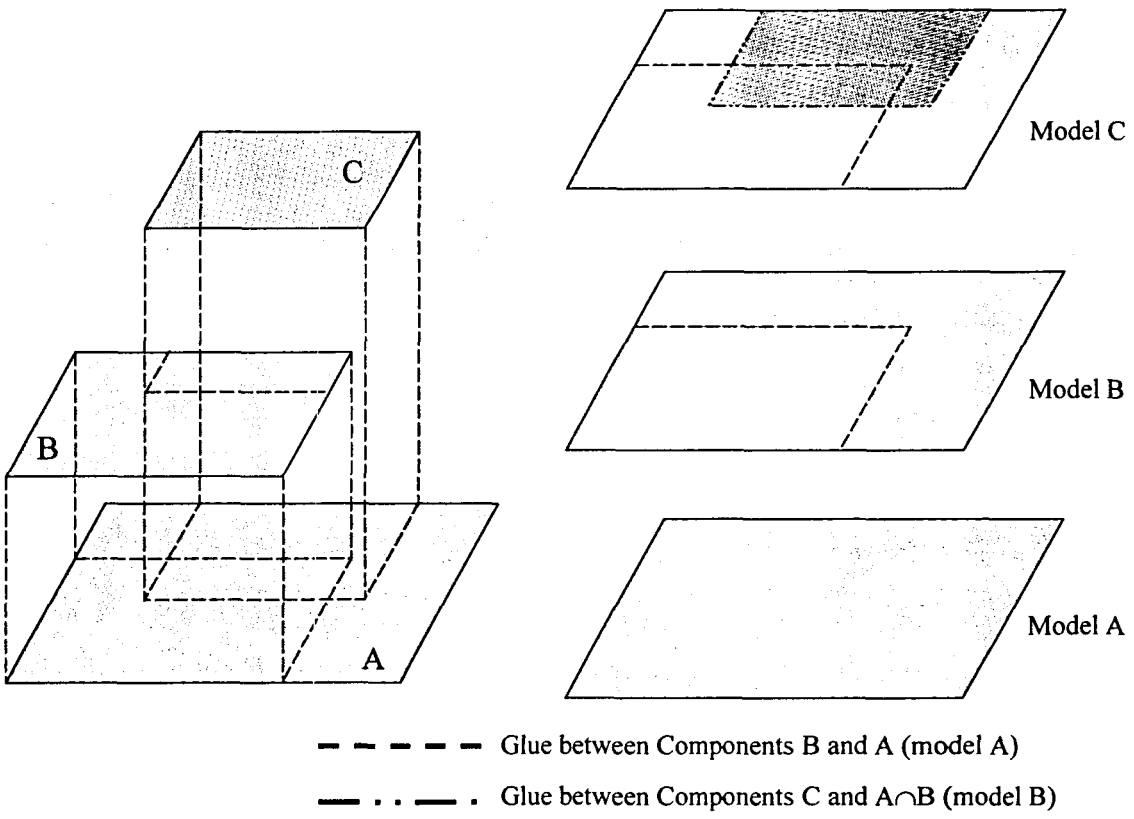

Figure 4: Joining components using digital glue

When a model is analyzed, certain data in the model path must be specialized for that particular model. An example of such a type of data in the current implementation is the degree of freedom (dof) map, which is required to map the results back to the components. This becomes a problem when a component can be a part of two models, which can easily be the case as shown in Figure 5. In this case, Model B is used by Model C and Model D. Therefore, the need arises to set only one model as 'active' at any point in time. Thus, each time a model is made active, the dof maps stored in its components have to be refreshed, since the existing dof maps in those components could be the dof map for the previously active model. In this way, the information in a component is dynamically updated by the currently active model and the component has a dynamic buffer storing the dof map for the active model. At present, this is not a major limitation. In fact, it saves on memory and book-keeping of this data by retaining the data in the component and making use of the hierarchy. The need might arise in the future for more than one model to simultaneously require their specialized data, which currently

resides in the component. In that case, we would need to maintain the data with the model rather than store it in the component.

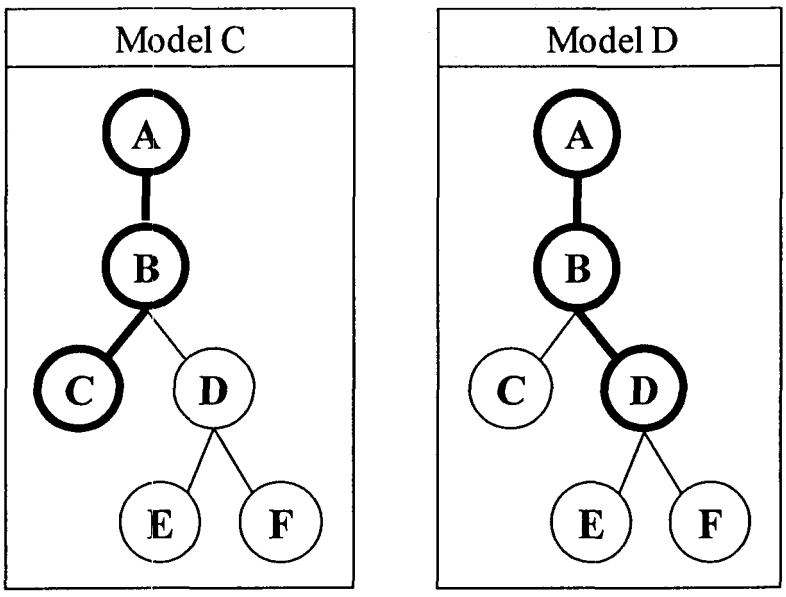

Figure 5: Buffering of solutions

HS4RAE – Implementation

As mentioned earlier, this kind of an environment requires a number of tools to successfully analyze a problem. Each of the tools developed will be now discussed in detail.

The C++ programming language and Visual Basic were used to develop the Hierarchical Environment. The object oriented-ness of the programming language was made use of extensively to implement the inheritance and hierarchical nature of the models.

The hierarchical definition module is a collection of classes and functions in the form of a dynamically linked library (DLL) called hs4rae.dll. All the classes and functions are written in C++ and compiled using Microsoft Visual Studio 6.0. The library contains classes that bring the 'hierarchical' character to the system. The two classes that are central to this system are the Model class and the ModelHandler class. The hierarchical definition module has the solver incorporated into it. It was developed by modifying a conventional in-house finite element code called Alpha.

The hierarchical inheritance tree is implemented in the form of a linked list and each node in the tree is a Model object. Each model has a link to its parent model and links to the models derived from it. Recursive functions have been coded that traverse the model path to assemble the collective mesh of the model, find the boundaries in the model and match the boundaries of components. A recursive strategy is built to access data at any tree node. The Model object processes the main scripting language commands. It is used to execute commands on a model, for example to derive a new model and to save its state by storing the commands. The syntax of the scripting language is explained in detail in the appendix.

The Model object handles only inheritance of geometric data. The inheritance of all other non-geometric information (for example, load conditions) is to be handled by the ModelHandler class but this has not been implemented as yet.

A ModelHandler pointer is a data member of the Model Class. It is the executive for the finite element analysis, results retrieval and visualization. The handler associated with a model provides the glue that combines the components to assemble a complete analysis

model such as by using multi point constraints (MPC). It manages the storage of and access to the results. The handler can be created/modified using the model handler scripting language or the GUI. A model can have more than one handlers associated with it depending on the type of analysis. Analysis-specific handlers can be derived from the base ModelHandler class depending on the requirement, for example, different handlers would be needed for linear and non-linear analyses. The handler holds the analysis related data such as load conditions, constraints and material properties. All data can be inherited from the parent model's handler or selectively overridden using the script or the GUI.

Figure 6 gives a class diagram of the hierarchical definition module. It can be seen that the geometric data such as the mesh data resides directly in the model object where as the other information such as load conditions, constraint condition and material properties are stored in the ModelHandler, which in turn is part of the Model class.

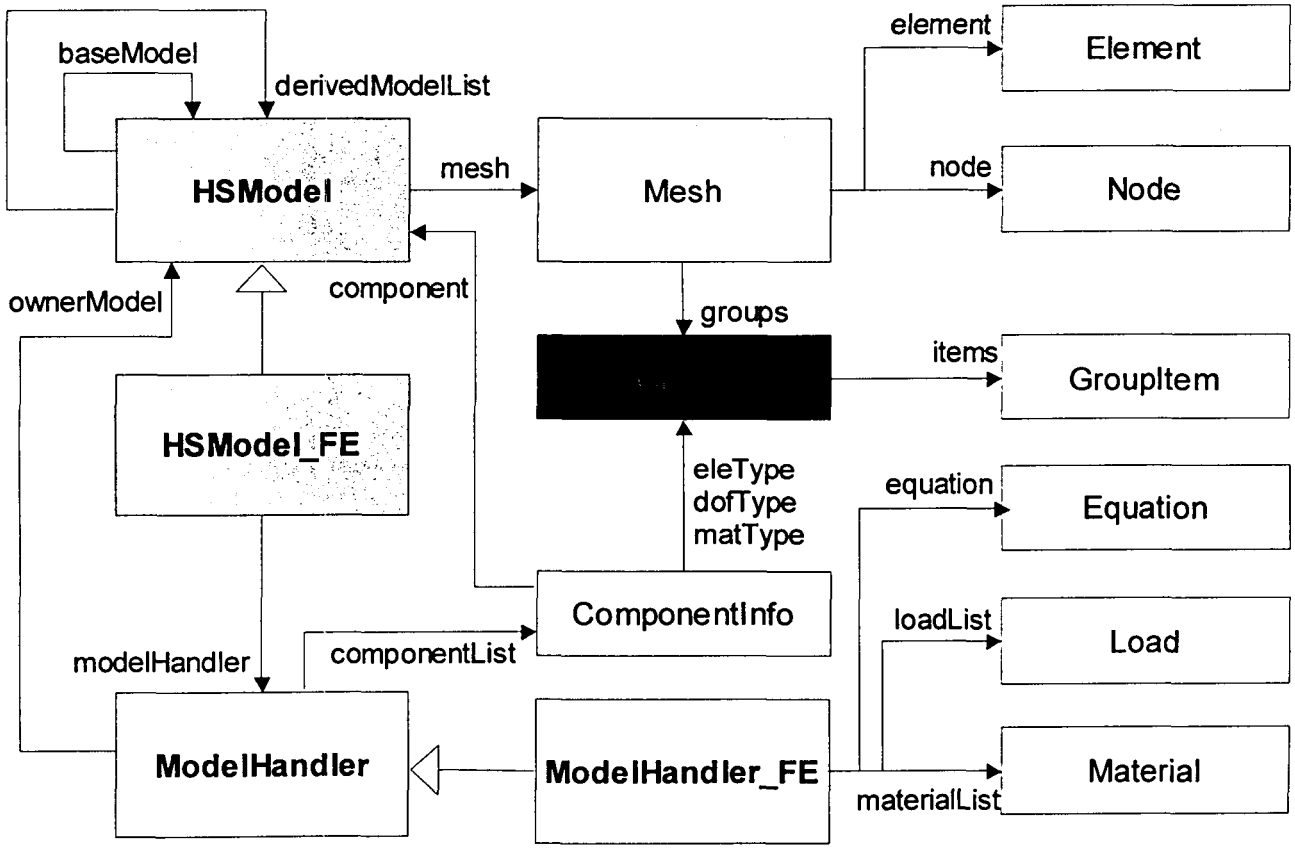

Figure 6: Class diagram of the hierarchical definition module

The equation solver is incorporated in the ModelHandler class. It must be noted that the solver does not solve the analysis problem hierarchically and is just a conventional solver. Therefore, the ModelHandler class constructs the complete set of equations as a conventional model and asks the solver to solve it. There is a potential for a hierarchical solver but this idea has not been pursued as yet.

Figure 7 illustrates the flow of data in the Hierarchical System.

The system is developed using C++, which is a high-level object-oriented programming language and allows efficient management of memory resources. The models are stored in the memory as objects that are dynamically created and managed using pointers. As a

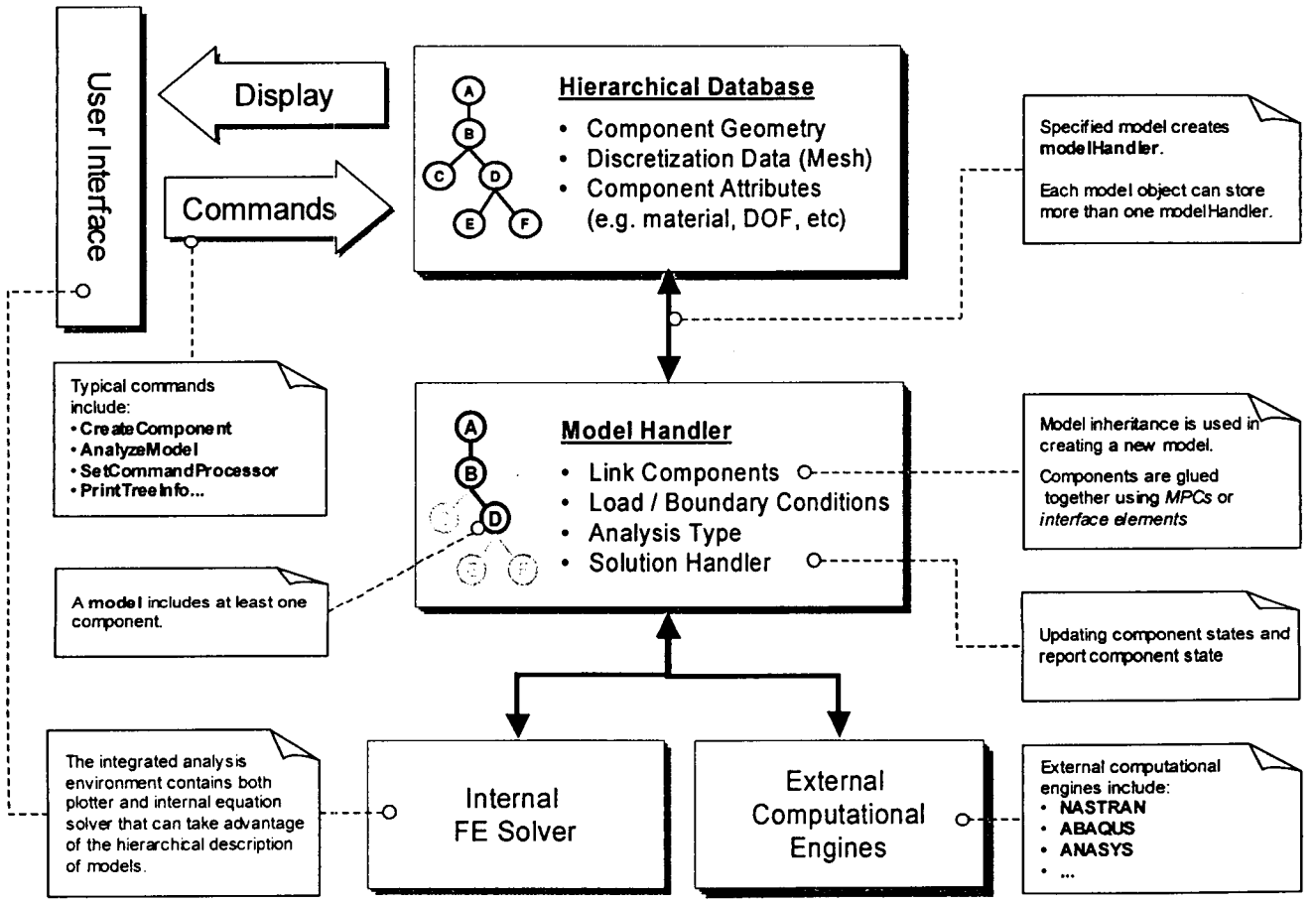

Figure 7: Data flow chart

result, the only restrictions imposed on the system with respect to the number of models and their sizes are the memory on the computer and the limits set by the programming language.

Data persistence is achieved by the capability to write all the data to the disk for re-use. A model scripting language has been developed which defines the hierarchical tree and a model handler script for analysis control of a model in the tree. These scripts can be used to store the state of analysis so that one can revisit it later. A lot of the other model data is stored in the format defined by the existing in-house FEA code.

The system uses the same solver as in the existing in-house FEA code. Since Alpha has been modified and added to the hierarchical definition module, the hierarchical library has the FE solver capability within itself.

The visualization tool called Plotter2002 is a modification of an existing in-house tool called Plotter2000. It is written in C++ and uses the Microsoft Foundation Classes. The Plotter has two versions – a stand-alone version and an ActiveX control version. Both these versions are built upon the same underlying functional code that is in the form of a DLL called the PlotInterface.dll. The plotter has three different options of outputting results – in GIF Format, PostScript format or directly onto the monitor screen using OpenGL. The Plotter has been designed in such a manner that it can be easily adapted or enhanced to display different types of data without the need for major code re-writing. This is made possible by making use of an interface class called the PlotInterface class that provides an interface between the data to be displayed and the output mode. In this way the Plotter need not 'know' what it is plotting. For example, if a model is to be plotted, the major work that is needed is to add a function to the model object to plot using the PlotInterface.

The plotter is able to read the mesh files that are used by Alpha and the Hierarchical System. The plotter can also read result data from analyses and give contour plots of displacement and stresses. It has various visual features that make it more effective and easy to interpret the results of an analysis.

The plotter has the basic features that most mesh viewers possess such as labeling of nodes and elements and highlighting different elements based on user input. It is also

possible to zoom in to observe the plot in detail. Another feature is the ability to rotate the mesh to get another perspective of the plot. The plotter employs its own hidden line removal algorithms to plot three-dimensional meshes or complex two-dimensional meshes.

For more complex analysis of results, it is possible to select and plot different elements from a mesh based on its material group. Other useful features include making specific elements transparent by directly specifying the element numbers or defining a region in the mesh. Contour plots can be tweaked by modifying the contour ranges and the magnification scale factor.

HS4RAE Graphic User Interface

The GUI for the hierarchical system is built on the existing visualization tool called Plot2000. The existing code was modified to be able to link with the Hierarchical definition module and thereby 'understand' the new classes that were developed for the Hierarchical system. New dialog boxes were added to let the user interact with and visualize the hierarchical model. The GUI makes it possible for the user to create and analyze hierarchical models on the fly as opposed to using a scripting language. However, the scripting language is very important because making the program write a script can save the current state of the model. This script can then be used by the program to load the hierarchical model back into the memory.

One of the most important components used to help the user visualize the hierarchical model is the tree control. This control is added to the main dialog box of the GUI and shows the hierarchical tree in the form of the directory structure in the hard disk. The different nodes in the tree denote each model in the hierarchical tree. Right clicking on a model gives a number of options to choose from. One of the options is to choose which model-related view is to be plotted in the plot window. There are basically two kinds of views- the complete model or the component denoted by the node in the hierarchical tree. Two additional plots that are related to the model are the basemodel copy and the basemodel. The basemodel is the component view of the parent model of the currently selected model. The basemodel copy is the region in the parent model that is replaced by the current component.

There are a number of list boxes that show the different entities associated with a model like the element groups and node groups. On selecting any existing element group or node group, the corresponding entities in the mesh are highlighted on the plot window. It is also possible to create new groups using the GUI. One of the new features added to the plotter is the ability to use the mouse to select and highlight elements in the plot window. This selection can be saved to a group. Alternatively, you can also use the keyboard to specify which elements/nodes need to be in the group.

The main dialog box also shows the different result sets that are available to view i.e. the results of the different analyses that were conducted on the models. To go with that are

different result visualization options, for example, whether to view the results for the component or the complete model, or whether to view the displacements or the stresses.

The GUI can be used to add or delete models in the hierarchical tree. This can be done by choosing the corresponding menu item in the pop-up menu of the tree control. When adding a model, the region in the current model that will be replaced by the new component has to be defined. This can be done by simply selecting the region with the mouse. There are two ways to do this. One is to simple point and click to select an element and repeat the process to select more elements. Clicking on the same element again will deselect the element. Another option is to choose the rectangle tool in the toolbar. This tool lets you define a rectangle with two clicks – denoting any two opposing corners of the rectangle. And this lets you select all the elements that fall within this rectangle. The 'select' and 'unselect' button in the tool bars let you use the rectangle tool to correspondingly select or unselect buttons. The add model menu option brings up a dialog box that asks for a name for the new model and a number of options to specify the mesh file for the new component. One option is to use an external mesh generator to create a mesh file. The path to the mesh generator can be entered in the textbox provided and the program can be launched. One thing to remember is that once the mesh has been created, it has to be converted to a format that can be understood by the hierarchical system. Another option makes use of an external mesh generator called GeomPack++ that has been interfaced with the hierarchical system. TheGeomPack++ utility is an object- oriented program that runs on the command line and does not have a graphical user interface. This mesh generator can be used to refine the region that has been selected. The Hierarchical system is also interfaced to the FEMAP software. Using this option exports the selected region to FEMAP and launches the program. The extensive mesh generating tools of the FEMAP program can now be used to create the component for the new model. Once the mesh is created, it can be exported to a NEUTRAL file, which can then be read by the hierarchical system. Finally, there is also the option of directly providing the path to the mesh file if it exists already. Choosing the 'delete model' menu item deletes the selected model and all the models that are derived from it.

Another important menu item is the 'Model Settings'. This option lets the user define the parameters required for conducting the analysis that would typically go into the

ModelHandler script when using scripting. This brings up a comprehensive dialog box that collects information about the components, materials, constraints, loads and analysis and output options. The element group, node group and material group to be used for the analysis are selected from the list of groups that have been defined for the component. This process is repeated for all the active components in the model path of the current model. The material tab of the dialog box displays the material library and lets the user add or delete materials to the library. Currently, only the PlaneStress material can be entered using the GUI. The Constraint tab displays the number of constraints on the model and also allows the user to add or delete constraints. Similarly, the Load tab lets the user add, delete or modify the load conditions acting on the model. Only the PointLoad and Point Constraints can be entered using the GUI for now. The Analysis tab let the user choose which type of solver to use for the analysis, for example, Sparse, Olaf or Profile. Other parameters that can be included in this tab are the optional outputs, but this has not been implemented. For now, if the GUI is used to conduct the analysis, all the possible output files are created.

After all the parameters required for the analysis are defined, the system is now ready to conduct the analysis. To do this using the GUI, the user can choose the 'Analyze Model' menu item. Another option the user has is to export the model to FEMAP and let the FEMAP software do the analysis. This is made possible by interfacing the hierarchical system with the FEMAP software through the NEUTRAL files. It should also be possible to import the results back from FEMAP into the Hierarchical system after the analysis has been conducted but this has not been implemented.

Another version of a GUI called Hviewer was developed which is very useful in illustrating the features of the Hierarchical Definition module and the boundary matching routines. This version is also written in C++ and uses the Microsoft Foundation classes but is built from scratch as compared to the other GUI, which is a modification of the existing plotter. HViewer uses the built-in hidden line removal features of OpenGL for plotting where as the Plotter uses in-house algorithms for this purpose.

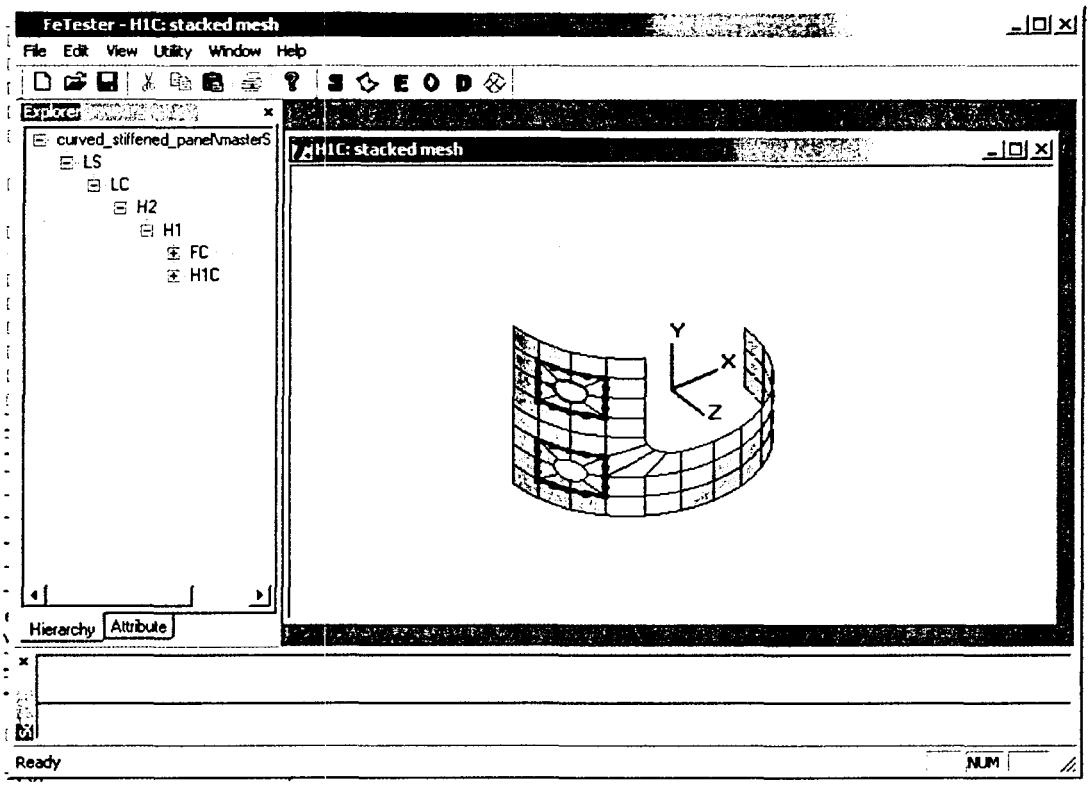

Figure 8: HViewer screenshot

Like the Plotter, the HViewer is also linked to the Hierarchical Definition Module and can understand all the hierarchical classes developed for this system. The HViewer also makes use of a tree control to visualize the Hierarchical tree and can generate different views of the models like the model view, component view and the basemodel copy. The HViewer does not have the capability to setup an analysis model interactively but uses the scripting language. The HViewer's forte is its plotting features for visualizing hierarchical models. It can generate stacked mesh plots of a hierarchical model and outline boundaries including those between components that are matched. The HViewer uses the external Plotter program to view the results of an analysis like the displacement and stress contour plots.

A Visual Basic version of a GUI was also developed early on during the project in order to check the performance of the ActiveX control version of the Plotter. It did not have the ability to setup an analysis model interactively either. The GUI interfaced with the Hierarchical definition module using exported functions. Like the other GUIs, this one also used a tree control to display the hierarchical tree.

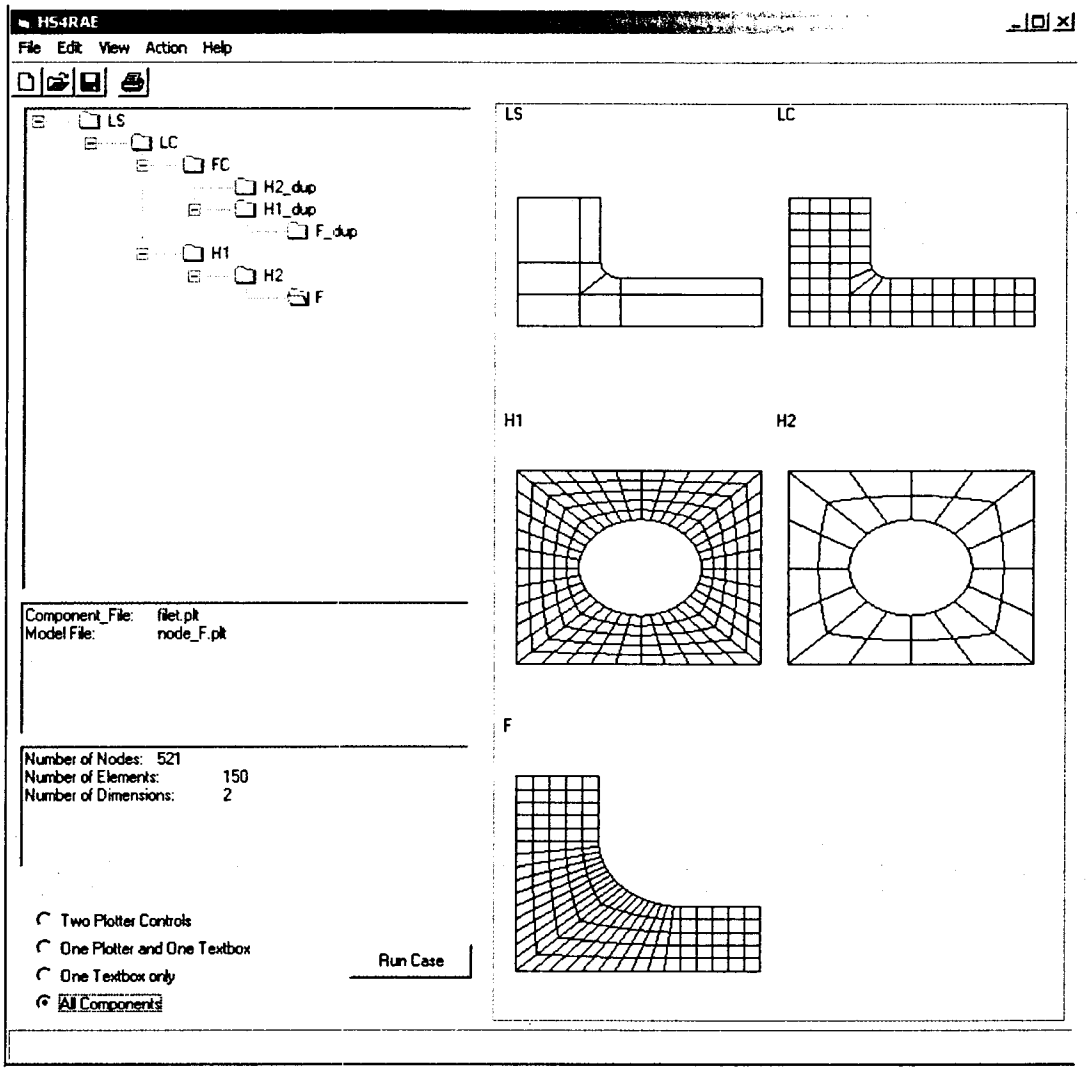

Figure 9: VB GUI screenshot

It had four different visualization options – use the plotter controls to plot a complete model view as well as a component view, plot the model view and display the numerical mesh data in a textbox, display the numerical data alone and lastly plot all the components in the model path of the currently selected model. The VB GUI made use of dynamic creation of ActiveX controls to create a number of plotter controls during run-time as per the number of models in the model path. The GUI also accesses information in the hierarchical model through the library functions and displays mesh information for the currently selected model in a textbox.

Example I

The purpose of this example is to demonstrate and test the capabilities of the developed system. It showcases the key functions and concepts of the philosophy such as inheritance of models, boundary detecting and matching and visualization of results.

A simple problem is considered wherein FE analysis is conducted on a structure. The geometry of the structure is then progressively modified to add a couple of holes. The analysis begins with a coarse model and then goes on to refine the model to get a better feel for the stress distribution in the structure.

The GUI is used to create the hierarchical models and the analysis proceeds based on the results of the previous analysis model. The structure is made up of a material with the following properties:

E11 =2.82 GPa

E22=2.82 GPa

Nu12=0.395

G12=1.01 GPa

Plane Stress elements are used to model the structure. The constraint and load conditions are kept the same through out all the hierarchical models and are illustrated below

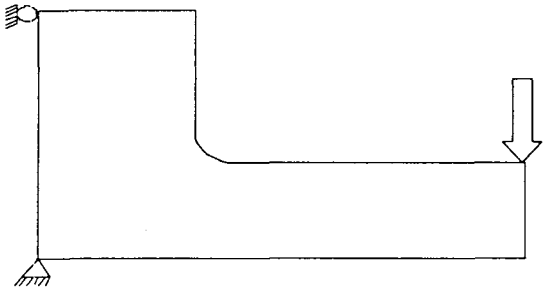

Figure 10: Example I – load and constraint conditions

For this first example, the mesh data for the components used are generated externally. In the second example, we go on to use the hierarchical system in conjunction with FEMAP to conduct the entire FE analysis. The initial model (called LS) is loaded and the parameters are supplied using the GUI to create a complete analysis model. The analysis

is conducted and it is found that the model is too coarse to realize the stress distribution in the different parts of the structure (see Figure 11). The model is then refined externally and a new model is derived called LC. The inheritance of the geometry is automatically taken care of and the whole parent model is deactivated since the new component overlaps the entire region occupied by the parent. The inheritance of the other data such as material properties, constraints and load conditions are not yet implemented therefore the analysis model must be set up again by entering these data using the GUI. Upon analysis of the model, it is found that this refinement gives a pretty good resolution of the stress and displacement distribution for practical purposes (see Figure **12**).

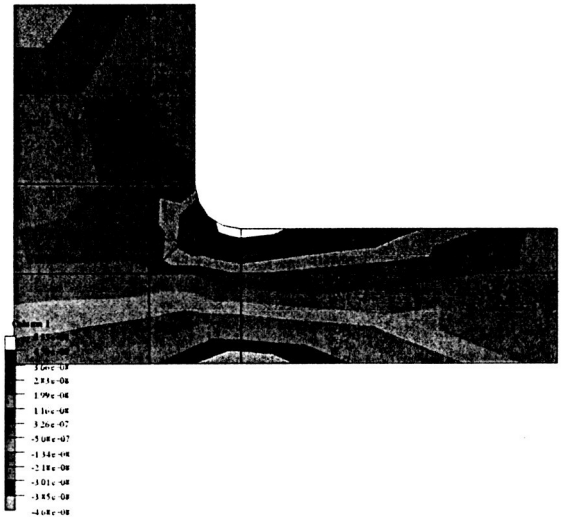

Figure 11: Model LS Analysis – Stress Plot (σ_{xx})

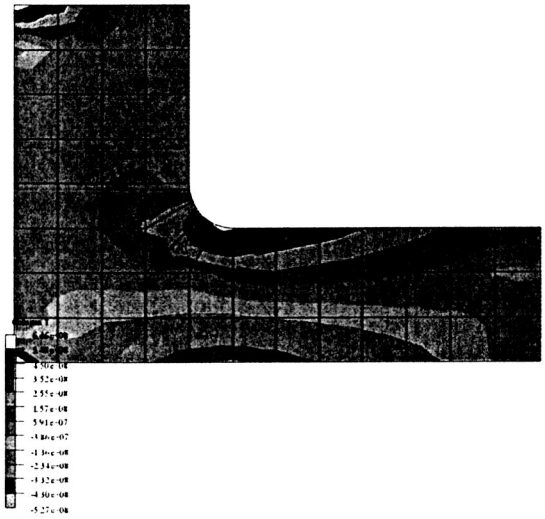

Figure 12: Model LC Analysis – Stress Plot (σ_{xx})

Next a hole is added to the structure in a position as shown in Figure 13. The new model H1 is derived from the previously analyzed model LC. In order to do this, the region in the parent model that will be replaced by the hole is highlighted using the GUI and the command to add a new model is issued. This asks for the mesh data file for the new component. The component replaces and deactivates the corresponding region in the parent model. The component is a simple 7-element mesh of a hole. Again the analysis model is setup and the run. The system automatically detects and matches the boundaries of the components and uses MPC's to 'join' them into a single analysis model. As seen in

Figure 13, it is possible to view the results of the model as well as all the active components in the model path.

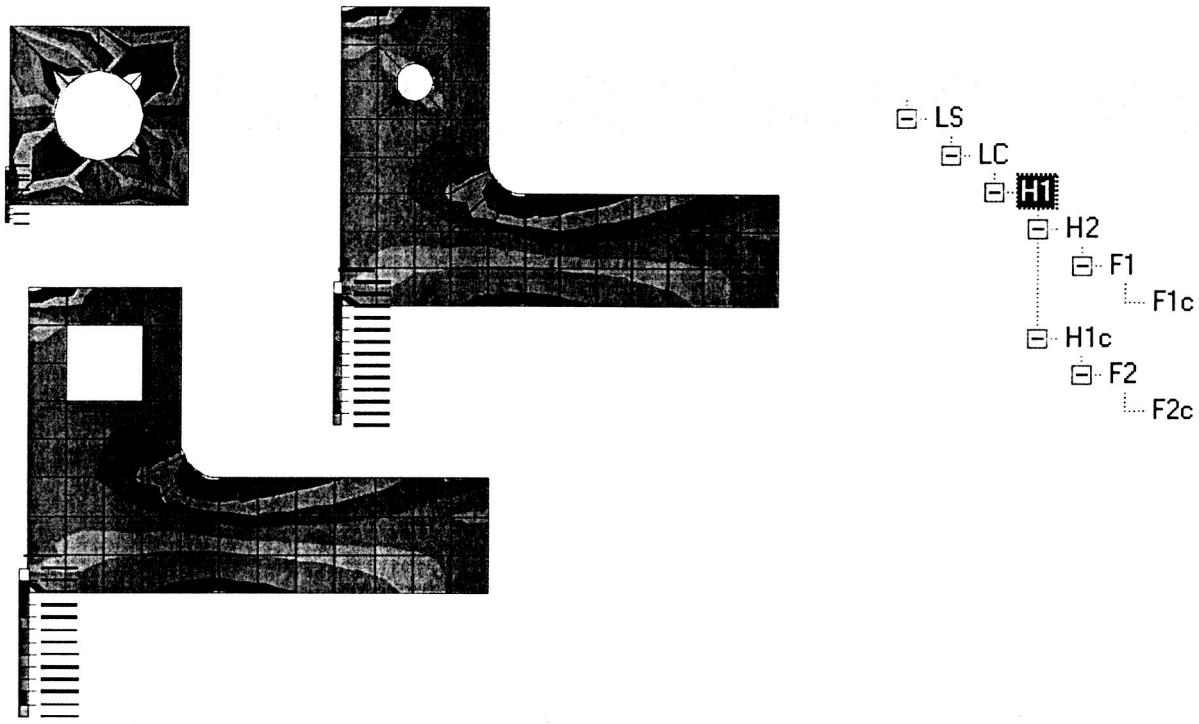

Figure 13: Model H1 analysis results

Another model (H2) is derived by adding another hole as shown in Figure 14. And the analysis is conducted as before. In this way more models can be derived and the hierarchical tree gets larger. We go on to derive model (F1) by refining the fillet and then derive another model (F1c) by introducing a crack in the fillet. The analysis of the model shows very high stresses in the region surrounding the crack tip (see Figure 15).

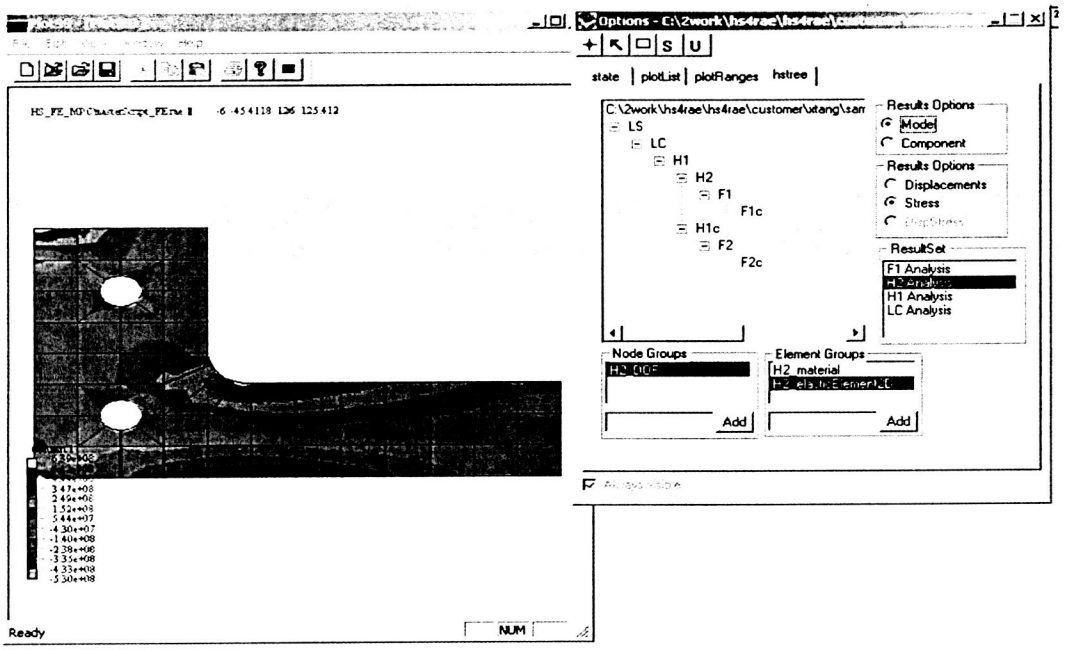

Figure 14: Screenshot of Plot2002 GUI (with model H2 results)

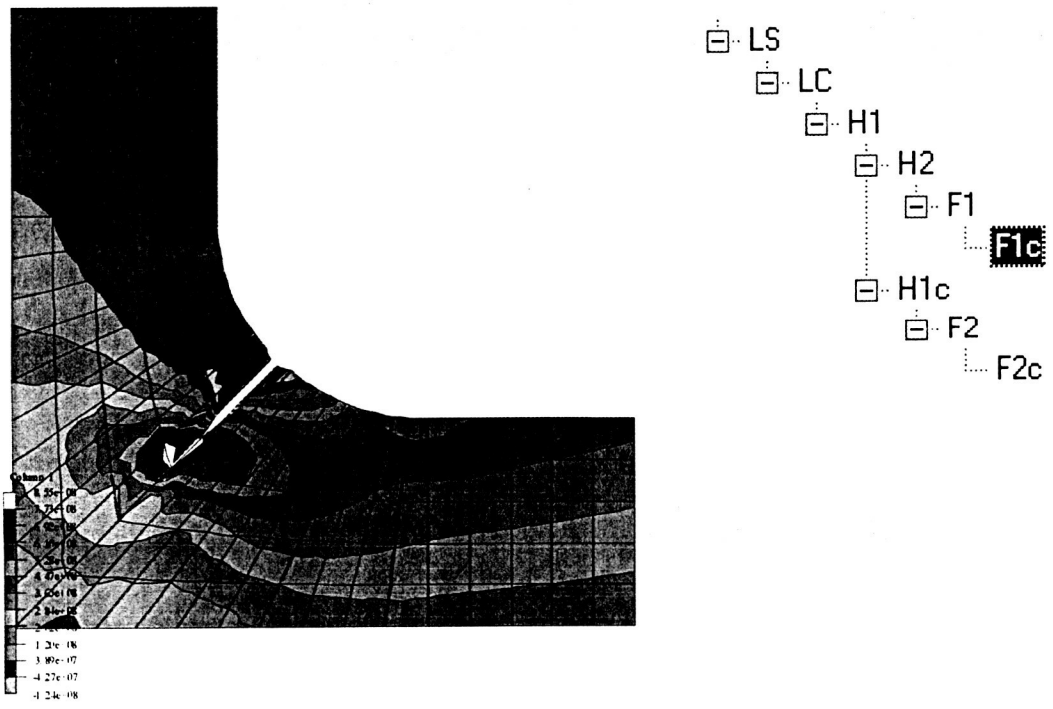

Figure 15: Stress Plot of component F1c (σ_{xx})

So far, the hierarchical tree is just a single line of inheritance from model LS to model F1c. we can start another branch to the hierarchical tree by deriving another model (H1c) from model H1 by refining the first hole and introducing a crack. Thus we start another line of analysis that does not include the second hole. It can be seen how easy it is to construct new models and make modifications. Model F2 is obtained from Model H1c by refining the fillet. And Model F2c is derived from Model F2 by introducing a crack to the fillet. Figure 16 shows a screenshot of the GUI after the analysis of the last model F2c in the tree is completed. The tree structure in the GUI dialog box depicts the inheritance tree.

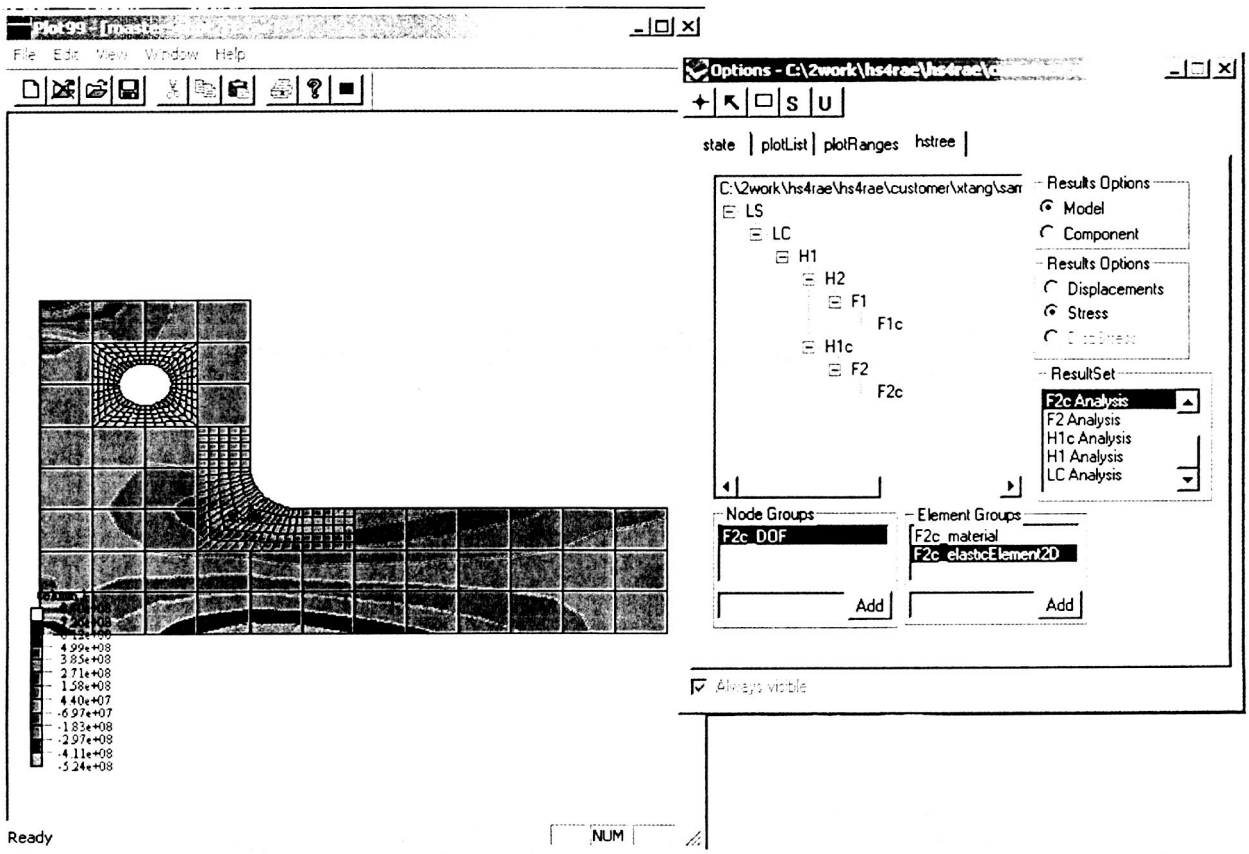

Figure 16: Screenshot of Plot2002 GUI (with Model F2c)

Scripts can be written to reproduce this state of the hierarchical tree and can be easily modified to add more models and conduct analyses. This is what the model script for Example-I would look like:

```
CreateModel LS lug_s.plt
setModel LS
  CreateModel LC  lug_c.plt LC_region_def.def
        setModel LC
        DefineElementGroup LC_material all  1
        DefineNodeGroup LC_DOF all  2
        DefineElementGroup LC_elasticElement2D all  1000
        analyzeModel LC  LC_model.rae 0

        CreateModel H1  hole_1c.plt H1_region_def.def
        H1::DefineElementGroup H1_material all 1
        H1::DefineNodeGroup H1_DOF all  2
        H1::DefineElementGroup H1_elasticElement2D all  1000
        analyzeModel H1  H1_model.rae 0

        H1::CreateModel H2  hole_2c.plt H2_region_def.def
        H2::DefineElementGroup H2_material all 1
        H2::DefineNodeGroup   H2_DOF all  2
        H2::DefineElementGroup H2_elasticElement2D all  1000
        analyzeModel H2  H2_model.rae 0

        H2::CreateModel F1  filet.plt F1_region_def.def
        F1::DefineElementGroup F1_material all 1
        F1::DefineNodeGroup   F1_DOF all  2
        F1::DefineElementGroup F1_elasticElement2D all  1000
        analyzeModel F1  F1_model.rae 0

        F1::CreateModel F1c  filet_cracked.plt F1c_region_def.def
        F1c::DefineElementGroup F1c_material all 1
        F1c::DefineNodeGroup   F1c_DOF all  2
        F1c::DefineElementGroup F1c_elasticElement2D all  1000
        analyzeModel F1c  F1c_model.rae 0

        H1::CreateModel H1c  hole_1_cracked.plt H1_region_def.def
        H1c::DefineElementGroup H1c_material all 1
        H1c::DefineNodeGroup   H1c_DOF all  2
        H1c::DefineElementGroup H1c_elasticElement2D all  1000
        analyzeModel H1c  H1c_model.rae 0

        H1c::CreateModel F2  filet.plt F1_region_def.def
        F2::DefineElementGroup F2_material all 1
        F2::DefineNodeGroup   F2_DOF all  2
        F2::DefineElementGroup F2_elasticElement2D all  1000
        analyzeModel F2  F2_model.rae 0

        F2::CreateModel F2c  filet_cracked.plt F1_region_def.def
        F2c::DefineElementGroup F2c_material all 1
        F2c::DefineNodeGroup   F2c_DOF all  2
        F2c::DefineElementGroup F2c_elasticElement2D all  1000
        analyzeModel F2c  F2c_model.rae 0

  ExitModel
  PrintModelTreeInfo
  OutputAllModels
ExitModel
end
```

Figure 17: Model script for Example-I

Remember that each analysis model would require an individual Modelhandler script. The model handler script for the model H1 would look like this (Refer the appendix for the syntax of the Scripting language.

```
// ModelHandler script file
// ------------------------
setStorageMethod
SPARSE

ComponentSettings  LC
  material LC_material
  element  LC_elasticElement2D
  DOF    LC_DOF
exitComponentSettings

ComponentSettings  H1
  material H1_material
  element  H1_elasticElement2D
  DOF    H1_DOF
exitComponentSettings

setComponentDofMap

ReadMaterials
PlaneStress
1 "Sample"
2.82e9 2.82e9 0.395 1.01e9 0 0
exitReadMaterials

setComponentMPCGlue glueC2B

ReadConstraints
PointConstraint
LC 0  1 0
LC 88 1 1
exitPointConstraint
exitReadConstraints

readLoads
PointLoad
Point 120 30 0 -1e9  2
exitPointLoad
exitReadLoads

DoAnalysis

doOptionalOutput

displacement
stress

displacement LC LC_dof
stress     LC LC_elasticElement2D

displacement H1 H1_dof
stress     H1 H1_elasticElement2D

exitOptionalOutput

end
```

Figure 18: Model Handler script for Model H1

Example II

The purpose of this example is to demonstrate the interfacing capability of the system with external software. The mesh generation features of GeomPack++ and FEMAP were used to generate the models used in this example. It also shows the ability of the system to handle real-life problems. In this example we look at a more practical problem where analysis of a side panel of an aircraft fuselage is conducted.

We start with a coarse global model of a section of the fuselage with 3 square holes intended for the windows. The load conditions are approximated to be the hoop stress and the longitudinal stress experienced by the fuselage due to the internal pressure in the cabin (see Figure 19).

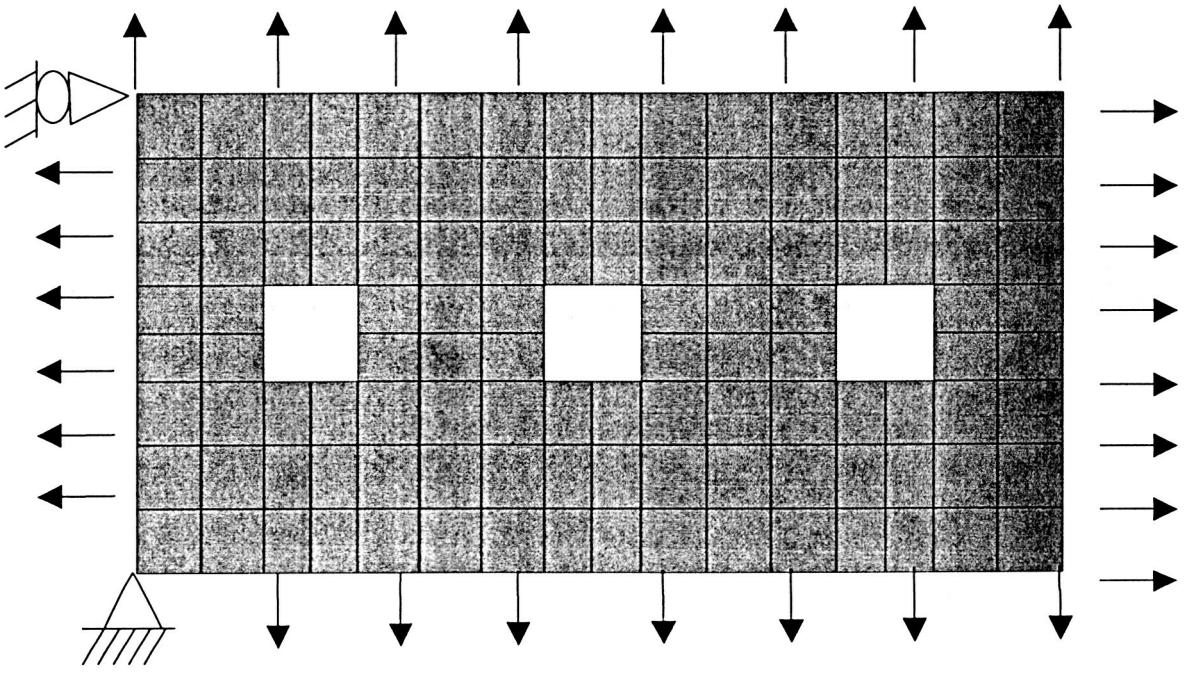

Figure 19: Example II – load and constraint conditions

First, the coarse model is refined and then more models are derived by making modifications to the second hole. The boundary of the second hole is made rounded to make it like an actual airplane side panel. Further localization is done by adding small holes to the periphery of the window in the middle. This is to model the holes for rivets that go into the fuselage to hold the actual windows.

The analysis results show the regions in the panel where there is the most chance for a crack to initiate. In this example, we stop at this point but further analysis can be done by

adding cracks in the panel and studying the its propagation in the panel. The results of the analysis can be used to calculate the stress intensity factor for studying the fracture mechanics of the model. This example shows that the system can be developed for solving practical real-life problems.

Figure 20 shows a screenshot of the GUI with the stress plot of a slightly refined panel. Figure 21 gives the component and model stress plots of the refined model of the panel with holes around the window. It can be seen that the stress is highest in the region near the holes at the top and bottom of the window.

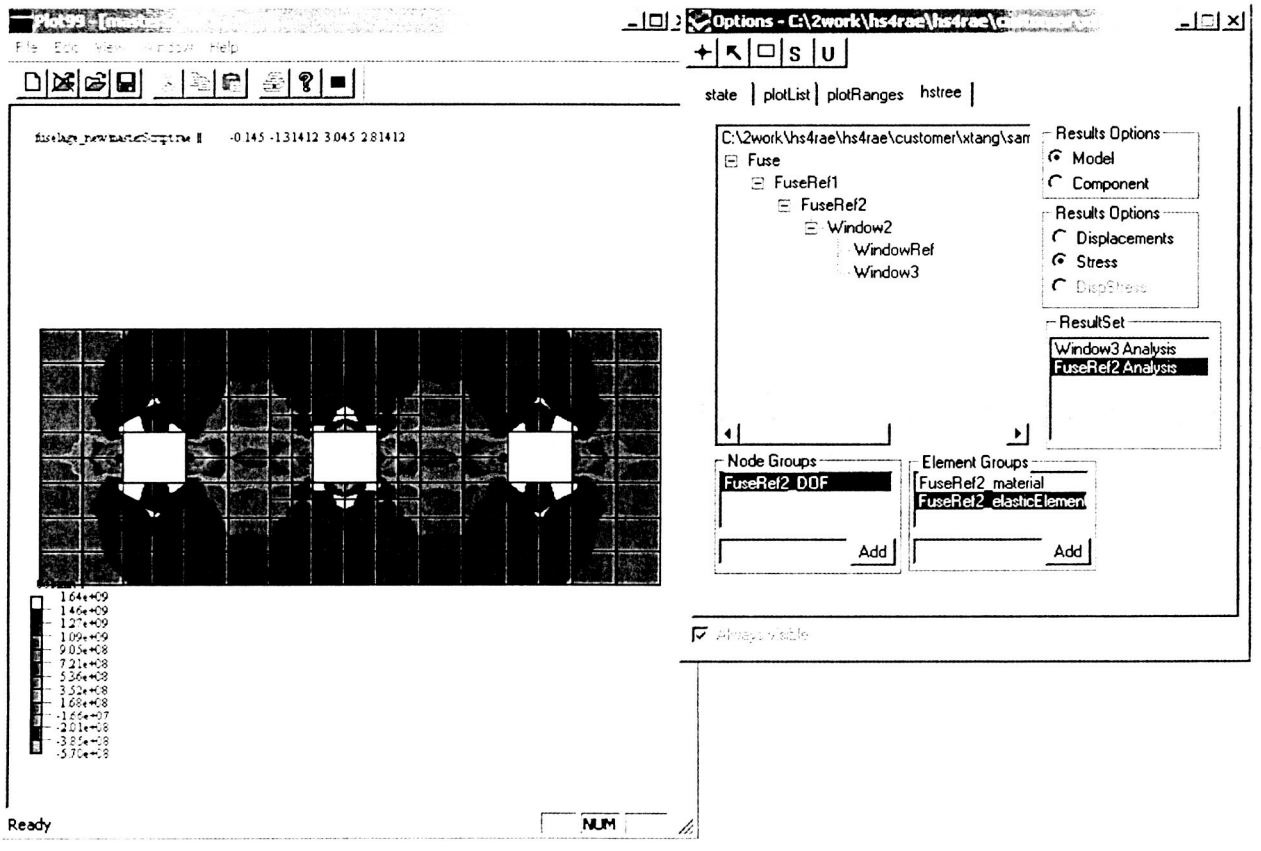

Figure 20: Screenshot of GUI with fuselage panel

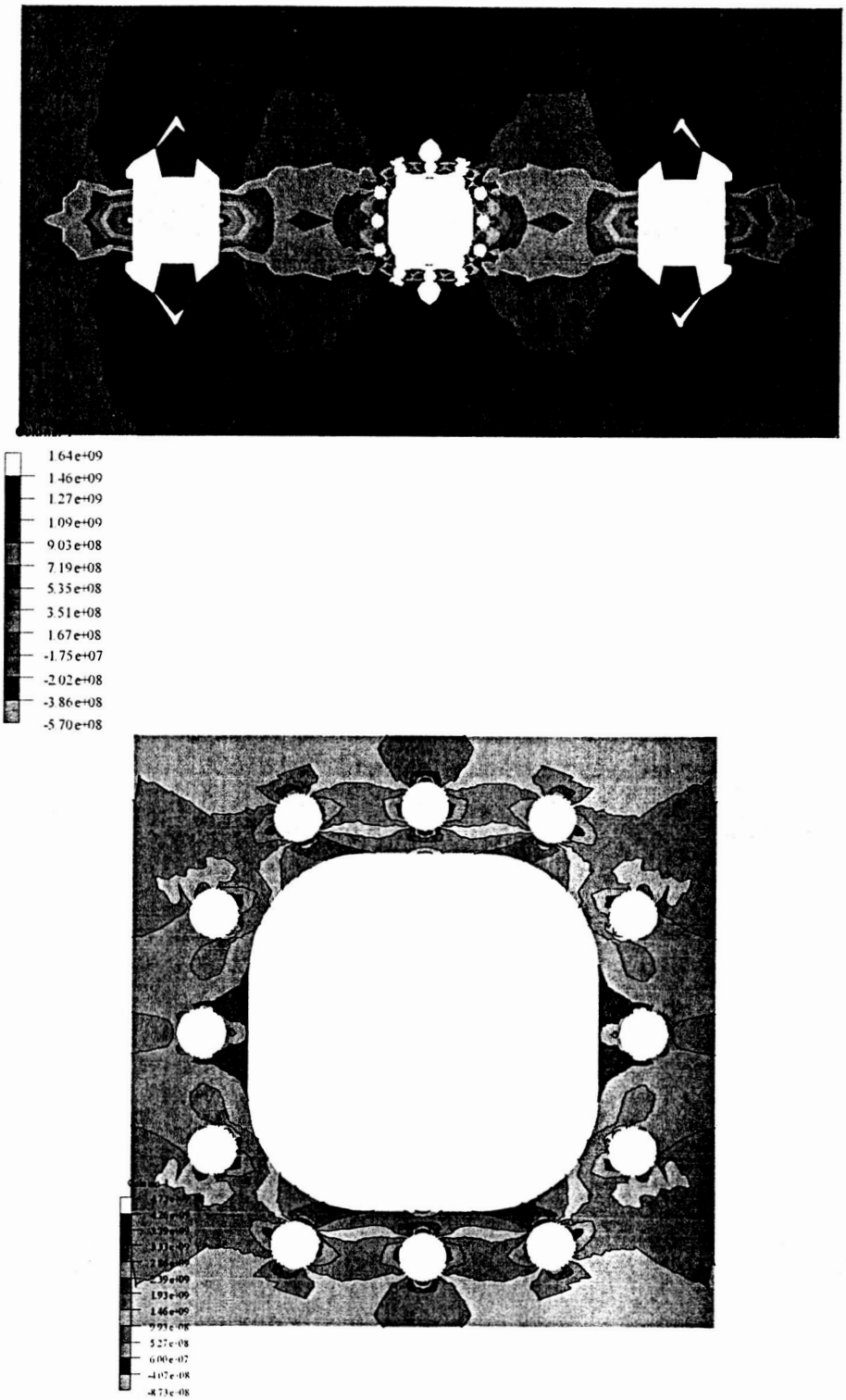

Figure 21: Model and component stress plots for the refined model of the panel (σ_{xx})

Conclusion and Future work

The philosophy using a hierarchy description seems to have a big potential for solving finite element analysis problems. As shown by the examples, it can easily reduce the human effort in generating new models and making modifications to existing models. It is especially ideal for global/local analysis due to the inherent hierarchical nature of the method. This kind of an environment involves many tools such as mesh generators, finite element solver, visualization tools and the hierarchical definition module. Each of these has a major impact on how 'rapid' the process is.

The utilities developed for this project are meant for illustrating this philosophy and cannot compete with commercial FEA software that has been developed of a long period of time. With more effort, many useful features can be included in the hierarchical system to make it more effective and enable rapid finite element analysis.

More features could be added to the visualization tool to assist the user in making decisions. In addition to using conventional MPC's, the interface technology [39] can be used to 'combine' components forming the complete analysis model. Currently the developed system can handle only two-dimensional cases. It would not be complete without robust boundary detection routines for three-dimensional geometries. This work could also be extended to handle problems that involve multiple scales and multi-physics.

Appendix

HS4RAE Scripting

This section explains the scripting language and its syntax. There are three kinds of scripts that can be written. One is the model scripting language, which is used to define the hierarchical tree. This script does not contain the information needed to conduct a complete analysis. The model script for a hierarchical model 'points' to the corresponding model handler script for a particular model in the tree, which has the detailed information needed for the system to conduct an FEA analysis. The last kind of script is the region definition script file that is a set of commands used to define a region in a mesh.

All scripts are terminated with the 'end' keyword. The following are the keywords in the model scripting language:

DefineModelName (or DefineName): Defines the name of a tree node.
Format:
[execModel::]DefineModelName|DefineName model modelName
If "model" is presented, "execModel" is discarded. Otherwise
"execModel" or the current model is to be assigned "modelName".

DefineModelMesh (or DefineMesh): Reads mesh for component.
Format:
[execModel::]DefineModelMesh|DefineMesh modelName meshFileName

CreateModel: Creates a tree node.
Format:
[execModel::] CreateModel modelName [meshFileName] [Region Definition FileName]

OutputModel (or OutputNodalModel) :outputs a conventional mesh for the complete model association with the tree node.
Format:
[execModel::]OutputModel [ModelName] MeshFileName

OutputComponent :outputs mesh for component.

Format:
[execModel::]outputComponet [modelName] outputName active|deactive

SystemCall: to run system commands from within the script.

Format:
[execModel::]systemCall

command

PrintModelTreeInfo: prints graphic tree, info for each tree node and outputs model component for each tree node and creates 'julian' NOD file (for VB GUI).

Format:
PrintModelTreeInfo

setModel (or setReader or setCommandProcessor) : Set current tree node. Pairs with an 'ExitModel' or 'exitUseModel' keyword.

Format:
setModel | setReader | setCommandProcessor modelName

...

...

...

ExitModel | exitUseModel

moveModelTo: moves position of model in tree.

Format:
[execModel::]moveModelTo [fromModelName] toModelName

toModel is the new baseModel of fromModel

fromModel can not be the rootModel

createModel_reuseMesh (or duplicateModel or copyModel) :duplicates a component and adds into tree, eventually add offsets, transformations , mirroring

Format:

[execModel::]duplicateModel | duplicateModel | copyModel newModel duplicatedModel

outputAllModels (or outputAllNodalModels) :outputs all models in tree.

Format:

outputAllModels | outputAllNodalModels

DefineElementGroup: defines a group of elements in a model

Format:

[execModel]::DefineElementGroup groupName selection attribute

DefineNodeGroup: defines a group of nodes in a model

Format:

[execModel]::DefineNodeGroup groupName selection attribute

ModelHandler: To create a ModelHandler and attach it to a tree node.

Format: [?::]ModelHandler execModel modelScriptFileName [HandlerType]

"execModel" must be specified.

If "HandlerType" if presented, it will be used to generate appropriate Handler.

The ModelHandler Scripting language is used to control the analysis of a model and contains the necessary data for conducting a finite element analysis. The ModelHandler script typically specifies the required parameters/information in the following order:

1. Solver
2. For all components, specify
   - Element type
   - Node DOF
   - Material
3. Material Library
4. Constraints
5. Loads
6. Generate DOF map
7. Glue (MPC)
8. Analysis command
9. Output options

The syntax for specifying each of the above mentioned parameters are given below:

1. Solver:

SetStorageMethod

SPARSE | OLAF | PROFILE

2. Specify Component information:

A data block with this format is written for each component in the model:

ComponentSettings <Component Name>

material <Material GroupName>

element <Element GroupName>

DOF <Node GroupName>

exitComponentSettings

3. Material Library: a data block with the following format is written for each material to be added to the library. Within each block the format of the information would differ depending on the type of material being defined. The example below gives the format for a Plane Stress material.

ReadMaterials

PlaneStress

<Material Library Index> <MaterialName>

<E11> <E22> <nu12> <G12> <a11> <a22>

exitReadMaterials

4.Constraints: this section starts with 'ReadConstraints' and is terminated with 'exitReadConstraints'. A data block is written for each Constraint type to be added to the constraint Set. Within each block the format of the information would differ depending on the type of constraint being defined. The example below gives the format for a Point Constraint.

ReadConstraints

PointConstraint

<Component name> <node number> <1 | 0> [<1 | 0> <1 | 0> ... for each DOF]

exitPointConstraint

exitReadConstraints

5. Loads: this section starts with 'ReadLoads' and is terminated with 'exitReadLoads'. A data block is written for each Load type to be added to the Load Set. Within each block the format of the information would differ depending on the type of Load being defined. The example below gives the format for a Point Load.

readLoads

PointLoad

<Component Name> <Node Number> <magnitude> <direction>

exitPointLoad

exitReadLoads

6.Generate DOF map: this directive is issued by the command 'setComponentDofMap'

7.Glue (MPC):

setComponentMPCGlue <Glue type>

8. Analysis command: this directive is issued by 'DoAnalysis'

9.Output options: this section starts with 'doOptionalOutput' and is terminated with 'exitOptionalOutput'. A line is written for each additional output option such as displacement and stress. Examples are given for both types of outputs.

displacement [<component Name> <nodegroup Name>]

stress [<component Name> <element groupName>]

If the optional information is not supplied, then the output for the whole model is obtained.

The Region Definition files are used to define a region in a mesh. The program reads and interprets the commands in the region definition files to create a list of elements that define the region. The different commands and its syntax is explained below:

1. **DefineRectangleRegion**: this command uses the bottom left corner and top right corner of a rectangle to add a rectangular region to the defined region.

   Syntax:

   DefineRectangleRegion

   <X1 Y1 (bottom left coordinate)>

   <X2 Y2 (top right coordinate)>

2. **AddElement:** this command adds an element of a specified component to the list of elements in the defined region

   Syntax:

   AddElement <component name> <element number>

3. **RemoveElement:** this command removes an element of a specified component to the list of elements in the defined region

Syntax:

RemoveElement <component name> <element number>

4. **AddElementList:** this command adds a list of elements to the defined region

Syntax:

AddElementList

<Component name> <element number>

...

...

End

Mesh Data File Format

The mesh data is one of the most important pieces of information in a finite element analysis. Finite element software has different formats for storing the mesh data on disk as well as in the memory. The Hierarchical system as well as the plotter uses the same formats to store and plot mesh information. It is pretty simple to interface this format with other commercial software, which has been done in this project with GeomPack++ and FEMAP. The following is the format for mesh information file used in the hierarchical system (which usually has the extension .PLT):

Each line denotes a record and the subscript at the end of a record denotes the number of such records and a subscript at the end of a field denotes the number of such fields.

<Number of Nodes> <Number of Elements> <Number of Dimensions>

[<NodeNumber> <x-coordinate> <y-coordinate> [<z-coordinate>]]$_{\text{(Number of nodes)}}$

[<ElementNumber> <nodes per element> <NodeNumber>$_{\text{(nodes per element)}}$]$_{\text{(Number of Elements)}}$

References

1. Pinkerton, Steven D., "Optimization of hierarchical structures", Journal of Information Processing and Cybernetics, v 29, n 4, 1993, p 221-231.
2. Stilman, B., "Hierarchical networks for space navigation", Applications of Artificial Intelligence in Engineering, 1994, p 339-348.
3. Oden, J. T., Vemaganti, K. and Moes, N., "Hierarchical modeling of heterogeneous solids", Computer Methods in Applied Mechanics and Engineering Vol. 172, 1999, pp.3-25.
4. Saito, K., Araki, S., Kawakami, T., Moriwaki, I., "Global-local finite element analysis of stress intensity factor for a crack along the interface of two phase material", : Proceedings of the 1995 10th International Conference on Composite Materials, Aug 14-18 1995, p 261.
5. Whitney, J.M., "Effective thermo-elastic constants of angle-ply laminates containing 90 degree ply cracks", Journal of Composite Materials, v 35, n 15, 2001, p 1373-1391.
6. Whitcomb, John D., Woo, Kyeongsik, "Evaluation of iterative global/local stress analysis", American Society of Mechanical Engineers (Publication) NDE, v 10, Enhancing Analysis Techniques for Composite Materials, 1991, p 201-205
7. Woo, Kyeongsik, Whitcomb, John, "Global/local finite element analysis for textile composites", Journal of Composite Materials, v 28, n 14, 1994, p 1305-1321.
8. I. Babuska, B. A. Szabo and I. N. Katz, "The *p*-version of the finite element method", SIAM J. Numer. Anal., 18 (1981), p 515-545.
9. I. Babuska and B. Szabo, "On the rates of convergence of the finite element method", International Journal for Numerical Methods in Engineering, v 18, 1982, p 323-341.
10. Zhu, D. C., "Development of Hierarchical Finite Element Methods at BIAA," Proceedings of the International Conference on Computational Mechanics, Tokyo I, 1986, pp.123-128.
11. Zienkiewicz, O. C. and Taylor, R. L., *The Finite Element Method*, 4th ed., McGraw-Hill, London, 1991.
12. J. Fish, "The s-version of the finite element method", Comput & Struct 43, 539-547 (1992).
13. J. Fish and R. Guttal, "The s-version of finite element method for laminated composite shells, Internat. J. Numer. Methods Engrg. 39, n 21, Nov 15, 1996, p 3641-3662.
14. J. Fish, Hierarchical modeling of discontinuous fields, Comm. Appl. Numer. Methods 8 (1992) 443-453.
15. C.D Mote. "Global-Local Finite element", International Journal for Numerical Methods in Engineering, v 3, 1971, p 565-574.

16. S. R. Voletia, N. Chandraa, and J. R. Millerb, "Global-local analysis of large-scale composite structures using finite element methods", Computers & Structures, v58, 1996, p453-464.

17. K.M Mao and C.T Sun, "A refined global-local finite element analysis method", internat J. Numer. Methods Engrg. 32 (1991) 29-43.

18. J.D. Whitcomb, Iterative global-local finite element analysis, Comput. & Structures 40 (1991) 1027-1031.

19. Babuska, I., Strouboulis, T., Upadhyay, C.S., Gangaraj, S.K., ", International Journal for Numerical Methods in Engineering, v 38, n 24, Dec 30, 1995, p 4207-4235.

20. J.N. Reddy and D.H. Robbins, "Theories and computational models for composite laminates", Appl. Mech. Rev 47 (1994) 147-169.

21. N.F. Knight, Jr., J.B Ransom, O.H. Griffin and D.M. Thompson, "Global/local methods research using a common structural analysis framework, Finite Elem. Anal. Design 9 (1991) 91-112.

22. A.K. Noor, W.S Burton and J.M Peters, "Predictor- corrector procedures for stress and free vibration analyses of multilayered composite plates and shells", Comput. Methods Appl. Mech. Engrg. 82 (1990) 341-363.

23. Lee, K., Moorthy, S., Ghosh, S., "Multiple scale computational model for damage in composite materials", Computer Methods in Applied Mechanics and Engineering, v 172, n 1-4, Apr, 1999, p 175-201.

24. Noor, Ahmed K., Starnes, James H. Jr., Peters, Jeanne M., "Uncertainty analysis of composite structures", Computer Methods in Applied Mechanics and Engineering, v 185, n 2-4, May, 2000, p 413-432.

25. Ransom, J. B., Knight, N. F. Jr., "Global/local stress analysis of composite panels." Computers and Structures, v37, 1990, p375–395.

26. A.K. Noor , Global-local methodologies and their application to nonlinear analysis. Finite Elements Anal. Design **2** (1986), pp. 333–346.

27. Fish, J., Suvorov, A., Belsky, V., "Hierarchical composite grid method for global-local analysis of laminated composite shells", Applied Numerical Mathematics, v 23, n 2, Mar, 1997, p 241-258.

28. J. Bramble, R. E. Ewing, J. E Pasciak and A.H. Schatz, "A preconditioning technique for the efficient solution of problems with local grid refinement, Compu. Methods Appl. Mech. Engrg 67 (1988) 149-159.

29. A. Brandt, "Multi-level adaptive solutions to boundary-value problems", Math. Comp. 31 (1977) 333-390.

30. J.Fish and V. Belsky, " Multigrid method for periodic heterogeneous media. Part 2: Multiscale modeling and quality control in multidimensional case", Comput. Methods Appl. Mech. Engrg. 126 (1995) 17-38.

31. J. E Flaherty, P.K. Moore and C. Ozturan, Adaptive overlapping methods for parabolic systems, in: J.E Flaherty, P.J Pslow, M.S. Shephard and J.D Vasilakis, eds., Adaptive Methods for partial Differential Equations (SIAM, Philadelphia, PA, 1089)

32. S.D. McCormick, Multilevel Adaptive Methods for Partial Differential Equations (SIAM Frontiers, III,l 1987).

33. S.F McCormick and J.W. Thomas, the fast adaptive composite grid (FAC) method for elliptic equations, Math. Comp. 46 (1986) 439-456.

34. T. Belytschko, J. Fish and A. Bayliss, "The spectral overlay on finite elements for problems with high gradients", Comput. Methods Appl. Mech. Engrg. 81 (1990) 71-89.

35. J. Fish and R. Guttal, "The p-version of finite element method for shell analysis, Comput. Mech Internat. J. 16 (1995) 1-13.

36. Fish, J., Markolefas, S., "Adaptive global-local refinement strategy based on the interior error estimates of the h-method", International Journal for Numerical Methods in Engineering, v 37, n 5, Mar 15, 1994, p 827-838.

37. J. Fish and S.Markolefas, R. Guttal and P. Nayak, "On adaptive multilevel superposition of finite element meshes for linear elastostatics", Appl. Numer. Math. 14 (1994) 135-164.

38. H. Yserentant, "On multilevel splitting of finite element spaces", Numer. Math (1986) 379-412.

39. ANSYS, "DesignSpace User Manual for Release 5.x", July 1999.

40. Paul Nielan, Sandia Lab News Vol. 54, Special Issue, Feb 2002.

41. Schoof, L., Yarberry, V., "Exodus II – A Finite Element Data Model", http://endo.sandia.gov/SEACAS/Documentation/exodusII.pdf, SAND92-2137, Printed November 1995.

42. Ransom, J. B., McCleary, S. L., and Aminpour, M. A., "A New Interface Element for Connecting Independently Modeled Substructures," AIAA Paper Number 93-1503, 1993.

43. Ransom, Jonathan B., "Interface technology for geometrically nonlinear analysis of multiple connected subdomains", Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, v 3, 1997, p 1862-1872

44. Wang, John T., Ransom, Jonathan B. , "Application of interface technology in nonlinear analysis of a stitched/RFI composite wing stub box", Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, v 3, 1997, p 2295-2310.

Summary of Hyperlinked Documentation

In addition to this final report, an online documentation is being prepared that convey the essence of the hierarchical system and at the same time delves into more detail explaining the different concepts introduced when developing this system.

The online version of the report explains the different components of the system and includes videos that demonstrate the use of this system. It also goes through the unique classes, the key algorithms and modules that were developed during the course of this project work.

The following page contains an overview of the online documentation. Most of the bullets will be hyperlinks that will take you to a more detailed explanation.

Hierarchical Strategy for Rapid Analysis Environment (HS4RAE)

Because much of what we do proceeds in a hierarchical way, it seemed natural to consider whether stress analysis could be performed more efficiently if the tools were designed with a hierarchical structure. The work documented in these hyperlinked pages was supported by NASA Langley through NASA Grant NAG-1-01080. Dr. Jonathan Ransom was the technical monitor. It should be noted that not all hierarchies involve inheritance. However, inheritance is the essence of the hierarchical structure developed. This page outlines the concepts studied and the tools developed during this grant. Follow the hyperlinks to obtain more detailed information.

- Generic components of stress analysis suite of tools
- Examples of inheritance
  - Nature
  - The way we think and work
  - Stress analysis
- Current analysis tools that share similarities with components of HS4RAE
  - Model builder
    - DesignSpace (Ansys)
    - SIMBA (Sandia)
    - NextGrade (NASA)
  - Model Handler
    - Commercial FE packages
    - New methods in research stage
- Characteristics of HS4RAE
  - Inheritance based description of models and the results
  - Inheritance / hierarchy tree
  - Two classes form the core
    - Model class (Inheritance based model builder)
    - ModelHandler class (Inheritance based model handler)
  - Script to describe relationships
  - Digital glue (Interface technology, multipoint constraints)
- Programs / Modules
  - HS4RAE
  - Visualization
    - Plot2002
    - HViewer
  - Import/Export functions

  - PlotInterface

- Algorithms developed
  - Inheritance / hierarchy tree
  - Tree traversal mechanism
  - Boundary detection, sorting and matching

- Classes developed
  - Model class
  - ModelHandler class
  - BoundaryFaceList
  - BoundarySegment
  - MatchedBoundarySegment.
  - MPCGlueList
  - ComponentInfo
  - CompoundMesh
  - Group/ GroupItem

- Programming techniques explored
  - Object oriented programming
  - ActiveX plotter control for FEA visualization
  - C++/MFC and Visual Basic Interface to HS4RAE

- Visualization techniques developed
  - Exploded view to display inheritance relationships
  - Display of component boundaries split between multiple layers of inheritance

- Examples
  - Simple example (Example I) to demonstrate basic concepts of philosophy
    - Inheritance
    - Boundary matching
  - More complex example (Example II)
    - Capability/potential to handle real-life practical problems
    - Interfacing with external software