Modal Test Optimization using the Virtual Environment for Test Optimization

Scott E. Klenke, Garth M. Reese, Larry A. Schoof and Craig Shierling (RE/SPEC)
Sandia National Laboratories
Albuquerque, New Mexico

Abstract:

We present a software environment integrating analysis and test-based models to support optimal modal test design through a Virtual Environment for Test Optimization (VETO). The VETO assists analysis and test engineers to maximize the value of each modal test. It is particularly advantageous for structural dynamics model reconciliation applications.

The VETO enables an engineer to interact with a finite element model of a test object to optimally place sensors and exciters and to investigate the selection of data acquisition parameters needed to conduct a complete modal survey. Additionally, the user can evaluate the use of different types of instrumentation such as filters, amplifiers and transducers for which models are available in the VETO. The dynamic response of most of the virtual instruments (including the device under test) are modeled in the state space domain. Design of modal excitation levels and appropriate test instrumentation are facilitated by the VETO's ability to simulate such features as unmeasured external inputs, A/D quantization effects, and electronic noise. Measures of the quality of the experimental design, including the Modal Assurance Criterion, and the Normal Mode Indicator Function are available[1]. The VETO also integrates tools such as Effective Independence[2] and minamac[1] to assist in selection of optimal sensor locations. The software is designed about three distinct modules:

1. a main controller and GUI written in C++,
2. a visualization model, taken from FEAVR[3], running under AVS*, and
3. a state space model and time integration module built in SIMULINK †.

These modules are designed to run as separate processes on interconnected machines. MATLAB's external interface library is used to provide transparent, bidirectional communication between the controlling program and the computational engine where all the time integration is performed. Data from the finite element model is downloaded to the MATLAB engine where the SIMULINK model is automatically created and executed. MATLAB GUI elements are used to simulate the data acquisition environment including response traces, over-range indicators, and full scale voltage ranges.

Introduction:

This paper presents an innovative test/analysis tool, called the Virtual Environment for Test Optimization (VETO), which is aimed at integrating testing earlier in the product design cycle through experimental design optimization. Traditionally, the role of testing in the product realization process has been limited to the end of the design cycle; after hardware has already been produced. As a result, data analysis and test requirements for a component have only been considered when the hardware is scheduled for testing. A goal in

---

*AVS is a trademark of Advanced Visual Systems, Inc., Waltham, MA.

†MATLAB and SIMULINK are trademarks of the Math Works, Inc., Natick, MA.

This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

developing this software tool is to provide test and analysis organizations with a capability of mathematically simulating the complete test environment within a computer. Derived models of test equipment, instrumentation and hardware can be combined within the VETO to provide the user with a unique analysis and visualization capability to evaluate new and existing test methods. By providing engineers with a tool that allows them to optimize an experimental design within a computer environment, pre-test analysis can be performed using analytical models to rapidly evaluate components before manufacturing has occurred. The benefits of using this type of experimental design tool can be very extensive. The user can evaluate the use of different types of test instrumentation and equipment as well as investigating new testing techniques for system identification to be used in analysis/ experimental model validation.

The VETO also plays a crucial role in support of an on-going Sandia National Laboratories program called Knowledge Based Testing. The Knowledge Based Testing program incorporates aspects of design, analysis, and test with rapid prototyping to optimize test based product information from an experiment. Combining the use of the VETO (to optimally design an experiment) with the use of rapid prototyping techniques for generating component parts makes the processes of model updating and validation much more efficient. There is a critical need for providing test based information to produce confidence in the predictive capabilities of computational models. A test design tool such as the VETO can make a large impact on this need.

The initial step in the Knowledge Based Testing program is generation of a computer aided design model which represents the geometry of the component to be tested. This geometric model is then used to assist component visualization, to generating a computational model used for dynamic analysis, and to produce a rapid prototype component through a stereolithography or fastcast process*. The developed computational model of the component is combined with analytical and/or experimentally derived models of the test instrumentation within the VETO to determine an optimal test design. The VETO is used to simulate the experiment and to maximize the test based information for computational model validation. The next step in the Knowledge Based Testing program is performance of the experimental test on the rapid prototype component based on the VETO test design. The results of this experiment are then be used to validate the computational model.

A major objective of this software development effort is flexibility. Because the virtual environment is a prototype software system, a primary concern for its design is that the code be easy to prototype. To minimize our effort, existing software tools are used wherever possible, provided that the necessary functionality and flexibility are available. Another significant design objective is to provide a final software system that can be used by a variety of individuals who have not been involved in its development. As is described below, our design integrates several commercial tools to meet these objectives.

The major tasks involved in our effort include: 1) database management, 2) visualization of the device under test, 3) utility functions (such as those providing additional information about any of the instruments, or interconnecting them), and 4) time integration of the system. A key element within the VETO environment is the use of virtual instruments to simulate dynamic behavior of real instruments. Each virtual instrument may require a different data representation within the different VETO modules. For example, the device under test requires a geometric definition under the visualization module, while the model

---

*Fastcast uses a stereolithography part as a mold for an automatic metal casting system.

is reduced to state space ABCD matrices for use in the time integration module. The required data for the different analysis environments are shown in Table 1.

Virtual Instrument	Environment		
	visualization	database	time integration
Structure Under Test	geometry eigenvectors eigenvalues	eigenvectors eigenvalues	ABCD matrices
sensors & actuators	grid number orientation	connections state model name parameters	connections state model name parameters
filters & amplifiers	N/A	connections state model name parameters	connections state model name parameters
Front End system	N/A	N/A	constructed based on other instruments

Table 1. Data Requirements in Each Environment

Program Integration - Unix™ Socket Connected Modules:

The VETO program is divided into three main sections: 1) user interface, database and utilities, 2) visualization, and 3) time integration. The user interface and utilities are written in C++ with the database implemented using the netCDF library*. Visualization is performed using previously developed custom AVS networks[3]. Use of existing visualization software immediately made available a wealth of tools permitting visualization of extensive finite element results, including modes of vibration, strain energy densities and static response data. The time integration section uses MATLAB and its SIMULINK toolkit. MATLAB was also used to construct the state space models for many of the virtual instruments (such as amplifiers and filters).

Communication between these different sections is performed using unix sockets. MATLAB is released with a set of interprocess communication tools (the "external interface"), by which data may be easily transferred between the programs. Data transfer with AVS was more complicated but accomplished in a similar fashion. AVS is distributed with example code permitting execution in a "server" mode. Commands and parameters may be readily transferred to AVS, but only the results printed to standard output may be retrieved. This was a significant limitation of the software, and limits its applicability in a general purpose code. Most of the structural data was shared through EXODUS II [4] files used by the AVS visualization software. The random access data features of this format was quite important to implementation of many of the analysis tools discussed below.

In addition to permitting rapid implementation of the virtual environment concepts, separation of the VETO software into these three sections had important side benefits. Unix sockets are network transparent, permitting us to run the application segments on different

* netCDF is a public domain, machine independent data format available from the Unidata Program Center in Boulder. unidata.ucar.edu

machines. For example, AVS could typically be run on an SGI machine specifically designed for visualization problems, while MATLAB ran on a more general purpose computer platform. Program development and debugging were also facilitated by the complete separation of these processes.

User Interface:

Database, integration, utility and user interface functions are performed in the `vetomain` module, Figure 1. The "File" option of the `vetomain` menu bar allows users to load finite element (FE) models into the VETO software as well as previously existing virtual test files. This module also provides numerous tools to assist the engineer in understanding how the various virtual instruments interact together.

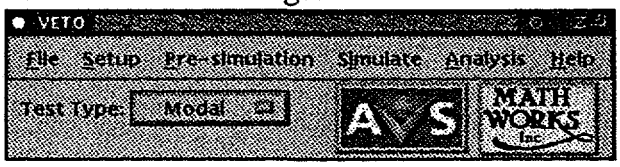

Figure 1. VETO Main User Interface

The user interface is implemented in C using Motif libraries. To provide access to AVS and MATLAB, user interface widgets were constructed by which commands could be entered and all output from the applications displayed. `Vetomain` acts as a controller, sending (or retrieving) data from the other two applications. Some limitations in the event driven model were introduced by the separation of the three application codes. Each section has its own event loop - specific to the interface events of that application. However, some events, such as communicating the node numbers from the visualized model, would be more natural if clicking on the model directly communicated back to the database program in `vetomain`. The single socket connection between programs makes this quite awkward. Action specific code was written to deal with node numbering, however, a more elegant solution might utilize an additional communication channel along with integrated event loops.

`Vetomain` is used to construct parametric models of the instruments, and to formulate interconnections between the models. Each of the virtual instruments is constructed in customized control panels. The user is able to interact with the Virtual Instruments Control Panel, selected from the "Setup/Instruments" option, to provide and view information on the devices in the simulation, Figure 2. A typical control panel used in the design of virtual actuators and sensors, those instruments that are in contact with the device under test (DUT), is shown in Figure 3.

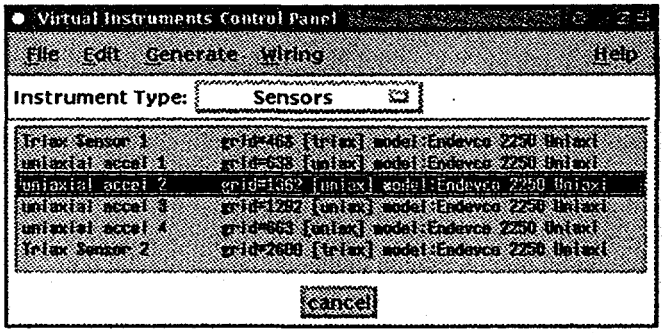

Figure 2. Virtual Instrument Control Panel for sensors

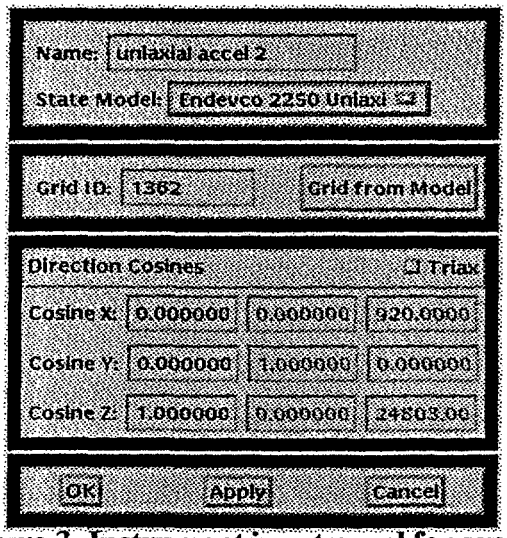

Figure 3. Instrument input panel for sensors

Before initiating a simulation run, the user also needs to set up instruments that do not directly contact the device being tested such as filters, amplifiers and the Front End data acquisition device. Two subpanes for providing parameters needed for sampling and computation of post-simulation analysis measurements for the Front End model are shown in Figures 4 and 5. There are additional subpanes to specify triggering, auto-ranging, windowing, averaging and display parameters for the simulation of the Front End device. Use of C++ classes for the instrument panels allows easier extension for new instruments. A special instrument called a "wire" is used to connect multiple instruments together. The DUT, selected virtual instruments and wire connections form the virtual test environment.

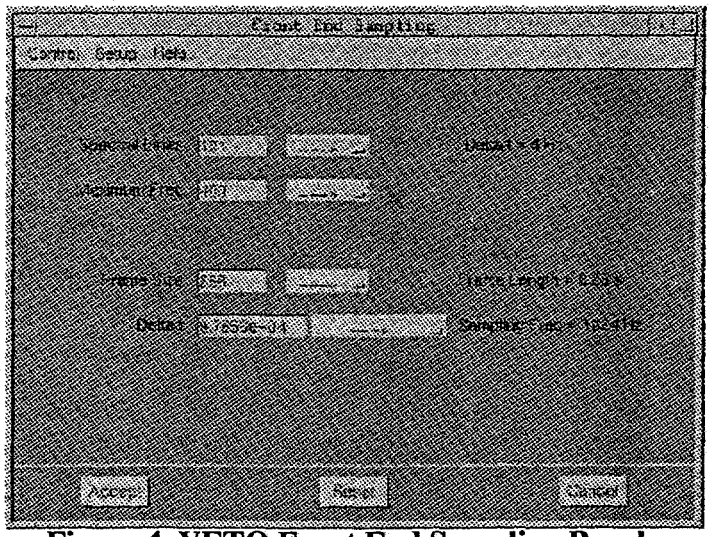

Figure 4. VETO Front End Sampling Panel

Once completed, the virtual instrument parameters and the structure of the test environment are downloaded to MATLAB, the state space model of the structure or device under test is constructed and the ABCD matrices are stored in the MATLAB workspace. SIMULINK scripts are then called to organize the virtual instrument models and data into an integration network from which the time histories of the system are computed.

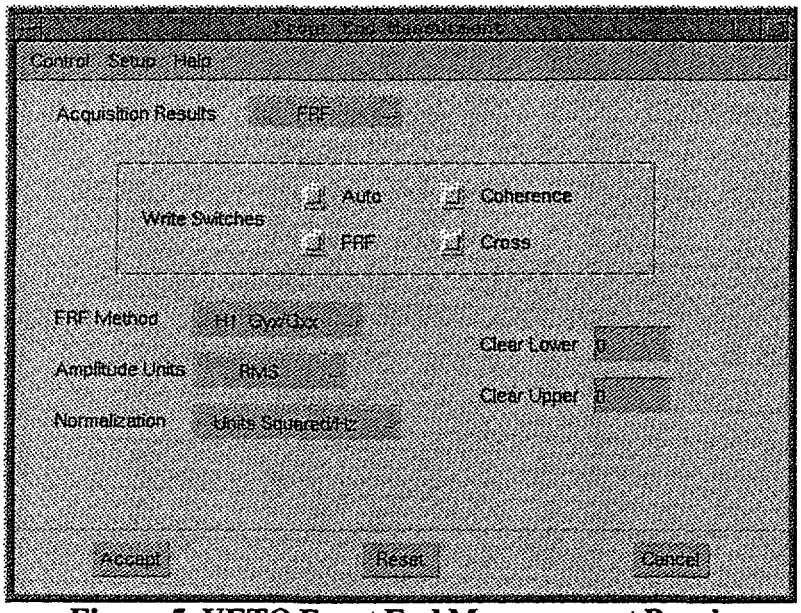

Figure 5. VETO Front End Measurement Panel

The `vetomain` module also provides numerous evaluation tools to allow the test engineer to determine the completeness of the virtual test environment. These tools can be accessed through the "Pre-simulation" menu option from `vetomain`. The modal data used for visualization is combined with selected virtual instruments to compute the Modal Assurance Criterion (or MAC), normal mode indicator functions, and driving point frequency response. The effects of the mass loading of the structure by the sensors may also be computed using a perturbation method. These and other tools guide the engineer in design of tests that will accurately identify all the desired modes of the structure.

Even with these tools, placement of sensors and actuators can be a difficult task. Tools such as the effective independence method or the `minamac` are used to automatically place sensors in locations which may help optimize the information available from the test. The virtual test environment provides the engineer with immediate visual feedback to determine the success of these methods, for which some engineering interaction is still required. Methods for automatic selection of actuator locations are currently under consideration.

Visualization with AVS Finite Element Analysis VieweR (FEAVR):

Since a finite element (FE) analysis is typically performed to predict the modes of vibration of a device, it was decided to utilize the FE model as the primary geometric representation of the device for visualization purposes within VETO. A prototype environment, FEAVR, which had been developed to provide a general purpose visualization capability for FE analyses, was selected as the graphics tool. FEAVR is an interface to the broad FE visualization functionality of AVS, incorporating networks and modules written or customized at Sandia. By using FEAVR, a user is freed from knowing the details of AVS. As an FE analysis visualization system, FEAVR provides the following capabilities:

- color the model with color fringes representing element-based (e.g., stress, strain, etc.) or node-based (e.g., temperature, displacement) scalar values.
- slice the model by showing an interior cutting plane or by removing a portion of the model that lies on one side of a slice plane.

- create an isosurface which is a surface on which an element-based or node-based scalar value is constant.
- represent a vector field (velocity, for instance) as arrows or streamlines (continuous lines that are everywhere tangent to the vector field).
- create X-Y plots of variables as they vary through time or in distance (across or through the model).
- deform the model according to a vector field, typically a displacement vector.
- create animations of mode shapes.
- "probe" the model to determine mesh-related values (i.e., nodal coordinates, node ID, element ID) and values of state variables (temperature, stress, etc.) at locations of interest.

As discussed previously, AVS is started and then connected to a Unix socket to accommodate bidirectional communication between AVS and the `vetomain` program. This allows `vetomain` to control the AVS process by issuing CLI (Command Line Interpreter) commands and also by receiving information about the model (such as node ID) from AVS. Via this mechanism, the FEAVR environment is initialized within AVS.

There were two fundamental extensions to FEAVR that were necessary for VETO. One was the ability to "attach" a virtual instrument to the model at a user-selected (via a mouse click) node point with a user-given orientation. For example, as an analyst or test engineer reviews the deformations representing a particular mode as shown in Figure 6, a virtual accelerometer can be placed at node 468 oriented parallel to the Z axis. The location and orientation of the virtual instrument is then transferred back to `vetomain` for development of the model of the DUT needed in the time integration.

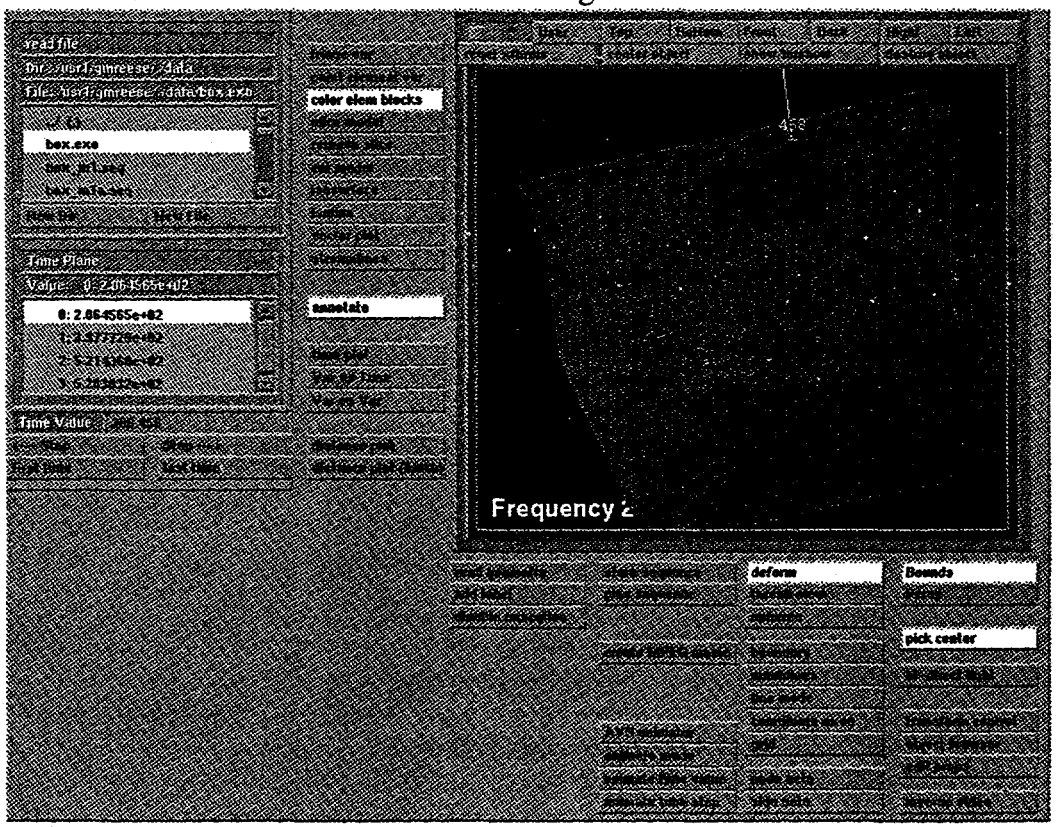

Figure 6. FEAVR display. Note virtual instrument at node 468.

The other extension to FEAVR was allowing the user to create "trace links" which are lines linking the virtual instruments on the model to create a simplified representation of the device geometry in the absence of the FE model. These are used in visualizing the simulated (or experimental) output of the virtual (or real) instruments. Figure 7 shows a deformed FE model (top) and the same model represented with just trace links and virtual instruments.

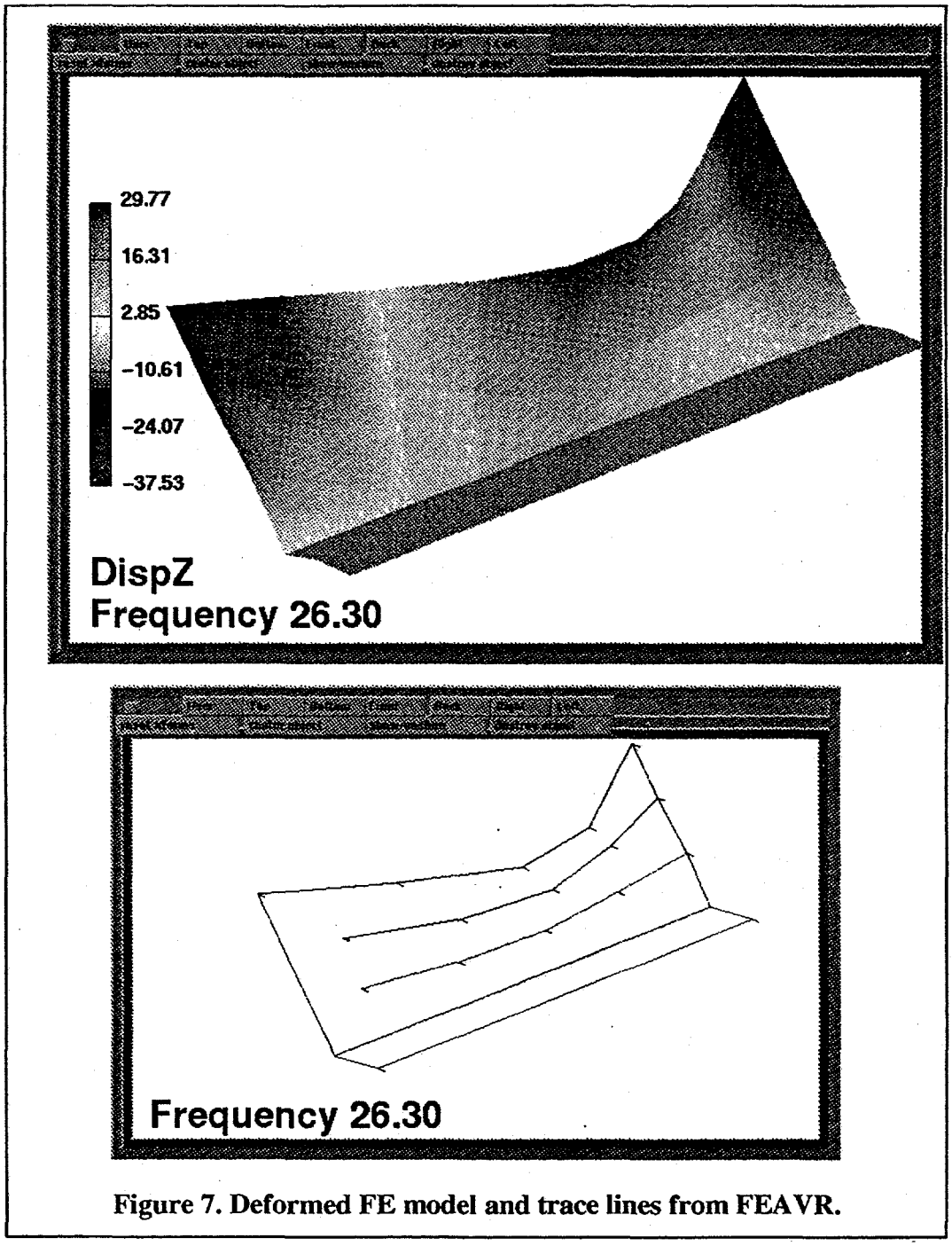

Figure 7. Deformed FE model and trace lines from FEAVR.

Simulations with MATLAB and SIMULINK:

The VETO software tool simulates the dynamic response behavior of a user defined test environment. The SIMULINK Dynamic System Simulation Software toolkit provided by MATLAB is used as the environment to assemble and ultimately integrate mathematical models of the test system. This same toolkit controls the simulation processing. Dynamic response equations are integrated by SIMULINK to provide simulated system output time histories. Within the VETO software, inputs such as type of device and interconnection of instrumentation models are combined to facilitate the rapid connection of various models (including models of test instrumentation, equipment and hardware) which comprise a given testing process. In order to achieve rapid set up of this virtual environment, models representing the instrumentation and equipment to perform the test simulation need to be developed. These models consist of a mathematical description of the dynamic response of the instruments derived either theoretically or experimentally. Most of the instruments modeled to date have been modeled in the discrete state space domain. A number of system identification tools, e.g. Power Polynomial [5] and Eigensystem Realization Algorithm with Data Correlation II [6], were used in MATLAB to generate the mathematical models. Development was based on an experimental frequency response function of the instrument or equipment.

The models of the different types of instruments and equipment (transducers, amplifiers, filters, etc.) needed to represent a complete testing environment are located in a SIMULINK Virtual Test Equipment Library (`VTELib`). When preparing for a test simulation, the selection of the desired test instrumentation from the `vetomain` is performed with the assistance of a MATLAB M-file called `lib_contents` which searches the `VTELib` for available instrument models. Optimal experimental design and simulation of the complete test environment is further facilitated by the VETO's ability to include models of external inputs and electronic instrumentation noise. In addition, complex instrumentation models, such as the Front End data acquisition system, are constructed by combining multiple submodels to simulate the dynamic response behavior of the hardware.

When "Build Simulation" is chosen from the "Simulate" menu of `vetomain`, the analysis data describing the DUT and other selected instrumentation parameters are downloaded to the MATLAB workspace. Processing control is then passed to MATLAB to construct a SIMULINK model of the test system. Construction begins with a SIMULINK "new_system" operation specifying the user's selected name for the test system. Into this new system diagram, the procedure places the device model blocks, specified by the `vetomain` data. There is a second level of block placement performed in the building process specific to the data acquisition device called the "Front End". When the "Front End" model block is placed into the new system, additional submodel blocks that simulate AC coupling and anti-alias filtering are placed within the "Front End" block based on the desired number of data simulation channels. Figure 8 shows a partial Front End block diagram as constructed by the VETO software tool.

As device blocks are added to the new system, interconnecting lines are placed between the blocks. These lines represent the flow of signals in the actual test system and are specified using the "wire" instrument in `vetomain`. Using these interconnecting lines, the input signals from the actuator devices (e.g. impact hammers) are fed to both the DUT for simulation of system excitation and to the "Front End" device for simulation of data acquisition. The "Front End" device also receives the signals from the sensors that have been attached to the DUT to simulate structural system response to the actuator input. Both

the simulated actuator and sensor signals are linked through amplifier and filtering blocks to represent preconditioning of the signals.

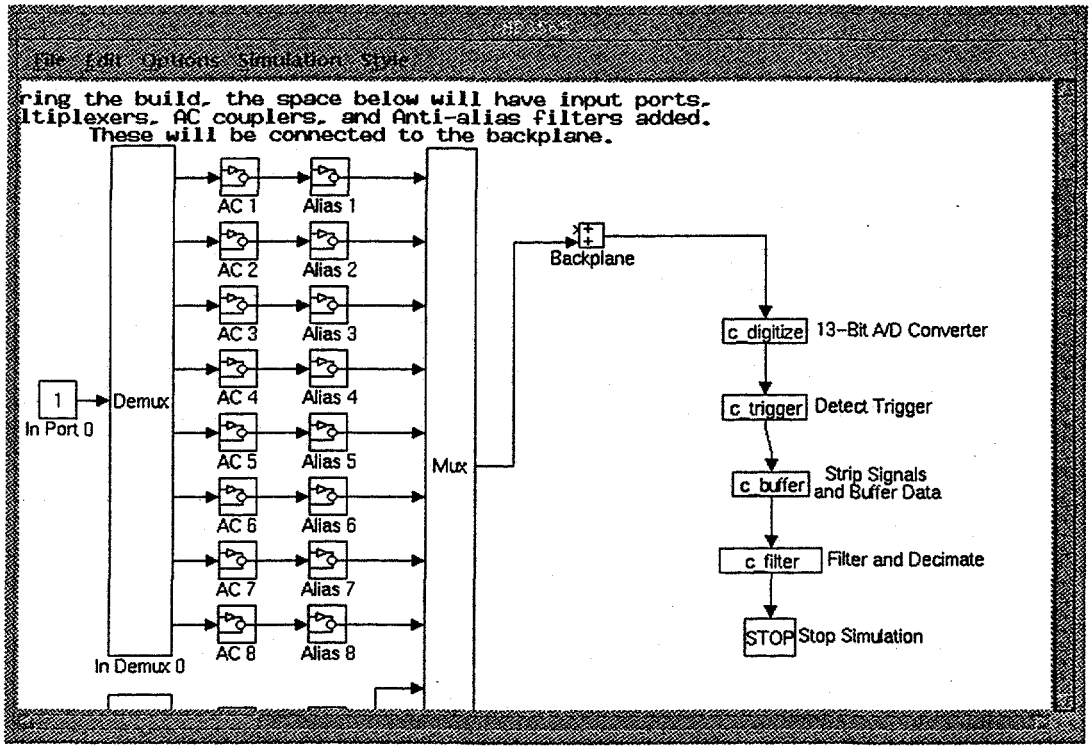

Figure 8. Partial Front End Block Diagram

The completed system is saved as a MATLAB ".m" and ".mat" file. Although it is possible for the user to modify the system built by VETO, care needs to be taken when directly modifying the SIMULINK simulation system. When changes are made to the test design from within SIMULINK, there is no mechanism for reflecting those changes in `vetomain` and in the FEAVR environment.

SIMULINK provides a number of methods for solving the set of differential equations which define the mathematical model of the test system constructed in the build phase of VETO. The VETO tool uses a Runge-Kutta fifth order ("rk45" operation) method to numerically integrate the equations for the test system. This method is considered to be a good general purpose integrator applicable to a large range of problems. It is a variable step size method with step size adjusted continuously to meet a specified relative error criterion. However, the VETO overrides the variable step size character by providing equal minimum and maximum step sizes as options when the simulation begins. The selected step size is the reciprocal of 32768 Hz; the maximum sampling rate of the HP3565 Front End device used for data acquisition and analysis. This forces SIMULINK to calculate the system responses at a constant or uniform time interval during the simulation process.

The process of simulation begins when the user selects "Run" from the "Simulate" option on `vetomain`. The data files which define the dynamics of the desired instrumentation are loaded into the test simulation system and the "Simulation Monitor" is created and displayed. This monitor allows the user to observe the estimated system response based on the numerical integration. The Simulation Monitor represents the data acquisition environment commonly used to gather data in a physical test and is a graphical interface through which the user interacts with the test simulation system. It has a set of buttons to

control the progress of the simulation and several display areas to provide visual feedback to the user. Figure 9 shows the Monitor in its initial state prior to auto-ranging. The VETO tool automatically performs auto-ranging to simulate the setting of Front End data acquisition voltage ranges on each analog-to-digital convertor required in the test simulation.

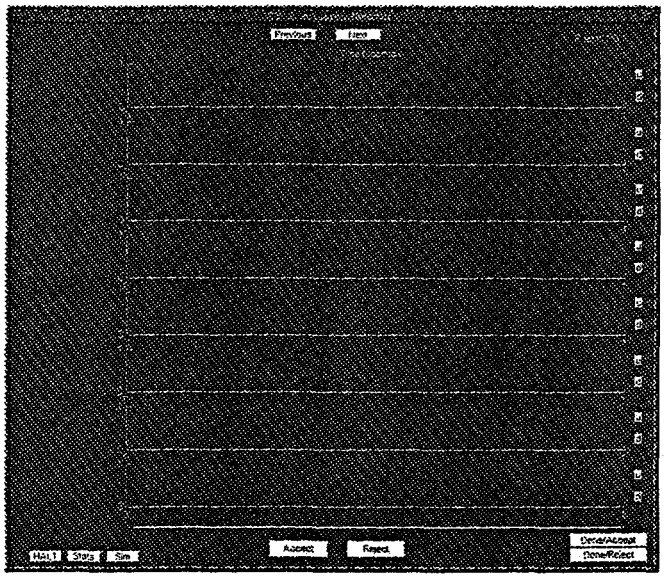

Figure 9. Simulation Monitor in its initial state.

During the simulation, the user has the option to halt the run using a button on the Monitor. Also as each frame of data is collected, the simulated response is displayed on the Monitor and the user is provided visual feedback on the test simulation results. Voltage ranges for each channel can be varied in order to maximize the signal's dynamic range before performing post-simulation analysis. Each frame can be accepted or rejected as a valid set of data using control buttons found at the bottom of the Monitor. A second set of buttons will accept or reject and also end the data collection phase. These buttons will also display a window which will provide an interface to analysis routines for computing desired measures such as frequency response functions, power spectral densities and coherences.

Application of the VETO to Structural Dynamics Test Simulation:

The VETO software environment currently integrates analysis and test-based models to support optimal modal or structural dynamic test design. The structural dynamics testing environment was selected as the initial VETO environment to enable the project team to prototype this tool to investigate areas of design/analysis/test interfaces, visualization, versatility and repeatability. This initial VETO effort has focused on assisting engineers through the development of a test design tool to maximize the value of modal tests. As was mentioned earlier, this environment plays a very critical role in the Knowledge Based Testing program particularly for structural dynamics model validation problems.

A weapon component housing was selected as the test case hardware for application in the VETO environment. The VETO software simulation tool was used to design an optimal experiment for the housing component. The goals of performing this test design optimization were to select an appropriate set of instrumentation (including sensors and

actuators) to perform a modal experiment within the VETO environment, to simulate a modal test on the housing component, and then to compare the results of the simulation to actual experimental data. A finite element model of the housing structure was loaded into the VETO environment for use in the modal test simulation. The test design was performed over a frequency band which included the first five vibration modes of the housing structure. The experimental test was performed on a steel component housing based on the VETO test design.

The outcome of the VETO test design "Setup" was to excite the structure at a single location using an impact hammer and to measure 51 acceleration responses on the housing component to characterize the dynamic behavior of the component, Figure 10. Small accelerometers, Endevco 2250s, were selected in the test design in order to minimize the mass loading effects during the experimentation. A large number of accelerometer locations were selected in the test design to make the process of analytical/experimental mode comparison more feasible. Other instrumentation such as the signal conditioning amplifiers and the Front End data acquisition system were also set up with the use of `vetomain` in preparation for the test simulation. Data acquisition parameters for sampling, averaging and desired analysis measurements were also selected for use in the post-simulation analysis.

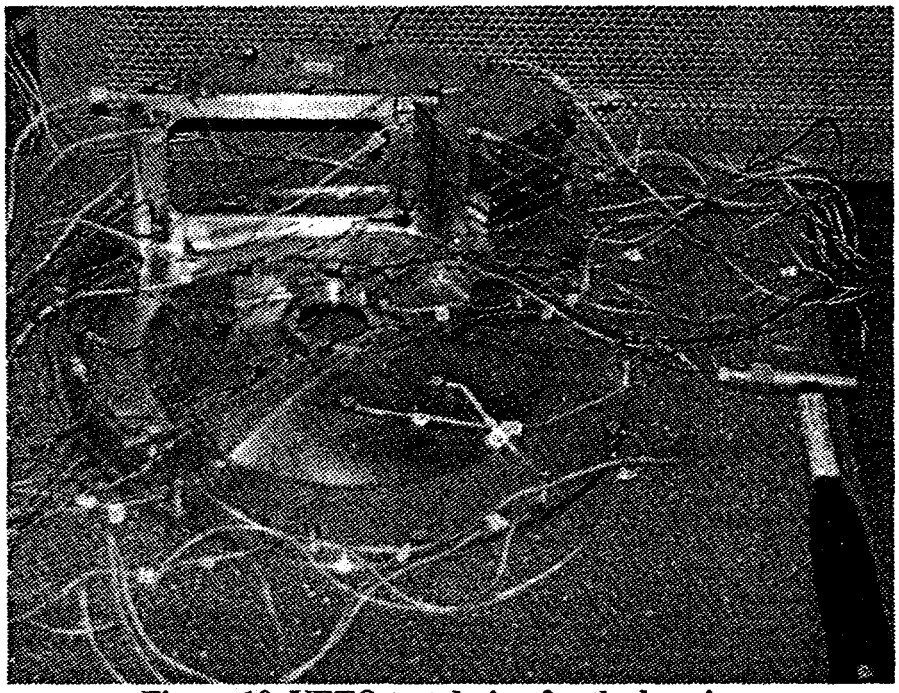

Figure 10. VETO test design for the housing.

A number of "Pre-simulation" tools were used to determine the completeness of the test design. First, the effects of mass loading the component housing were calculated given the test design sensor set, Figure 11. This figure showed that small changes in the frequencies of vibration would be experienced during the experimental test, based on the number of Endevco 2250 accelerometers chosen in the test design. Second, a normal mode indicator function and a driving point frequency response function were viewed before conducting the test simulation in order to assess whether the selected sensor and actuator (selected impact location) set would accurately identify all the desired modes of interest on the component housing, Figures 12 and 13. Initially, a location near the center of the housing,

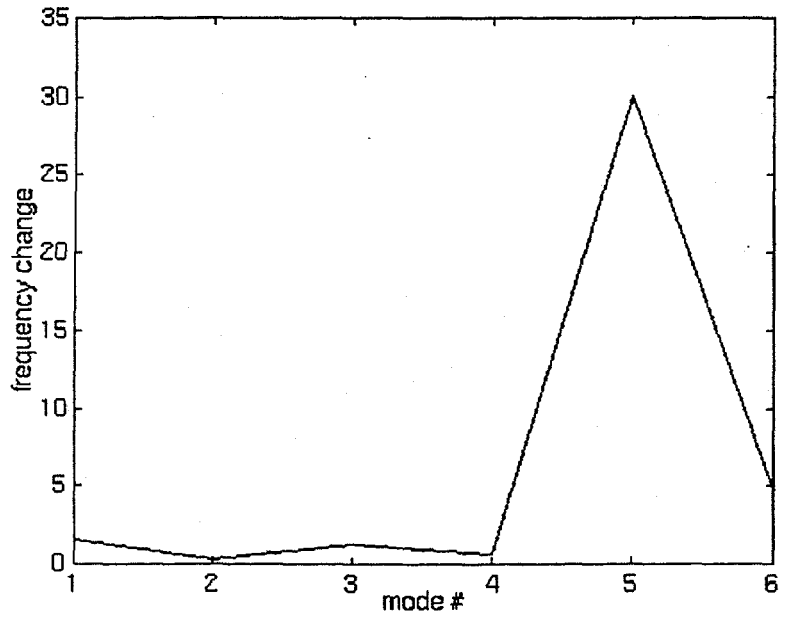

Figure 11. Estimated frequency shifts due to mass loading the housing.

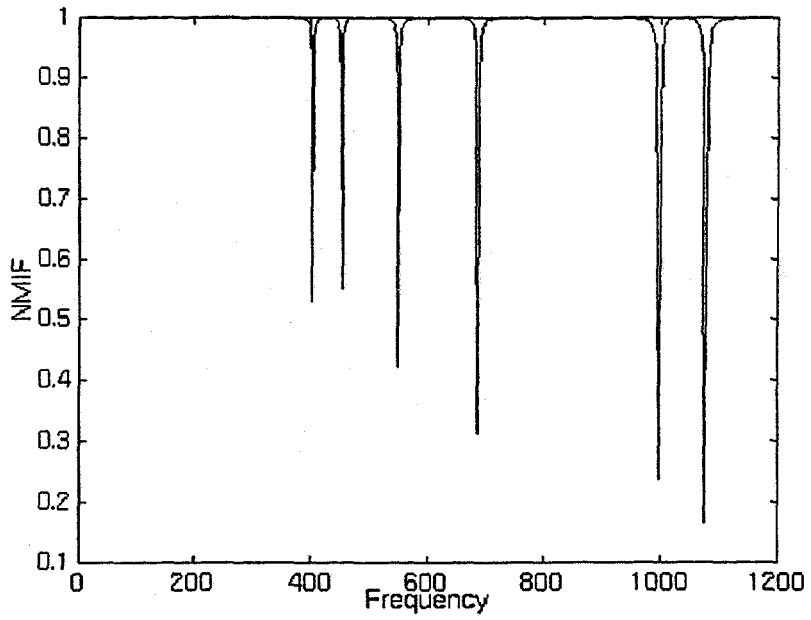

Figure 12. Normal Mode Indicator Function for the housing

on the top of the dome, was selected for the excitation of the structure. By using the normal mode indicator function, it was determined that an input location at the edge of the housing would excite the first two modes of the structure more strongly than exciting at the center of the housing. Finally, the Modal Assurance Criterion (MAC) was calculated for the test design to determine if the modes of vibration of the structure could easily be distinguished

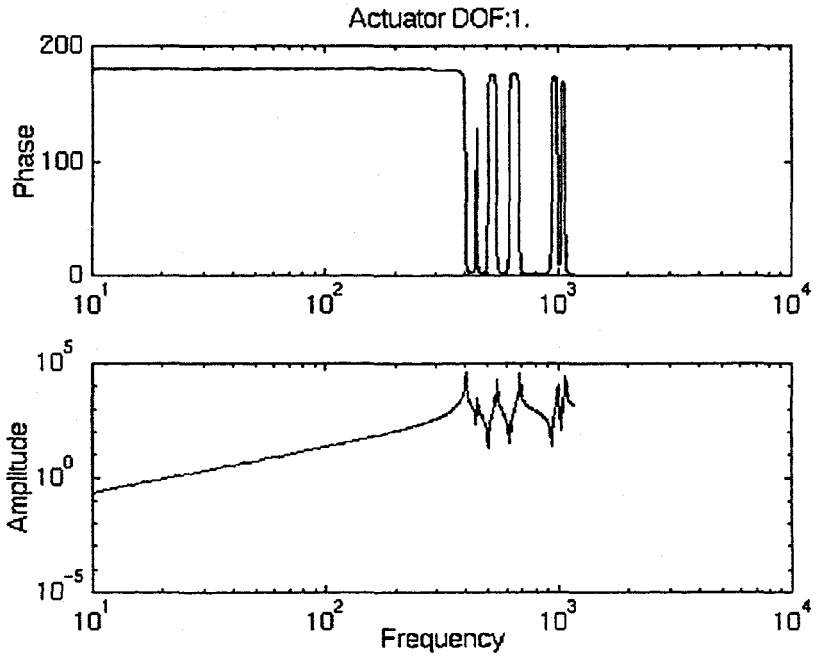

Figure 13. Driving point frequency response for housing.

from one another given the selected sensor set. Small values on the off-diagonal terms of this MAC matrix, Figure 14, indicate the relative independence of the modes of vibration thus making correlation with analysis data more attainable.

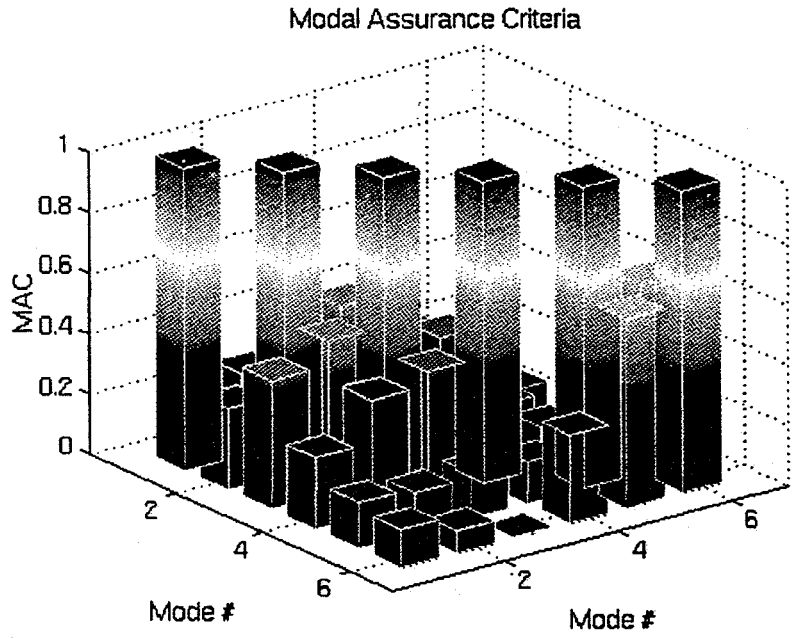

Figure 14. Modal Assurance Criterion (MAC) of the housing

Once the test design had been completed within the VETO environment, a SIMULINK block model of the test environment was automatically generated to support the simulation

of the modal test. Figure 15 shows a partial block model of this SIMULINK environment.

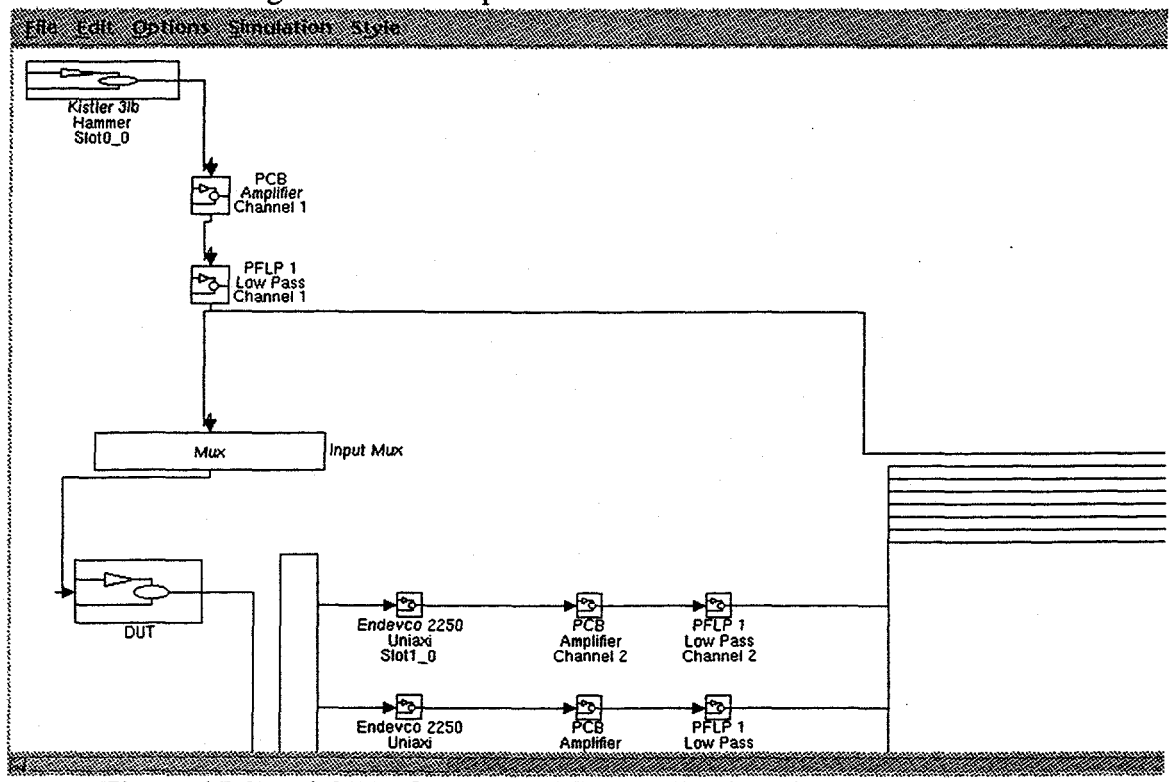

Figure 15. Partial Block Model of SIMULINK environment.

The next step in the modal test simulation is the numerical integration of the mathematical models within SIMULINK to estimate the 51 system responses. Using the Simulation Monitor, these responses are observed for each set or frame of data to be collected, Figure 16. Once the required amount of data is gathered to support the desired

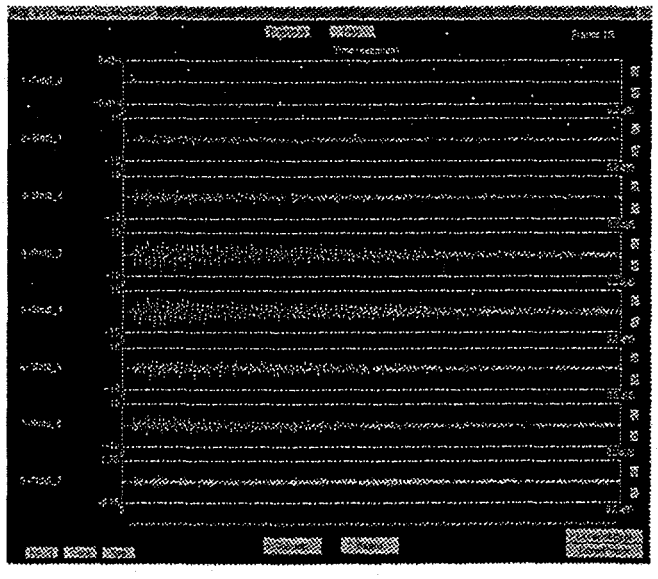

Figure 16. Collected Responses from Simulation Monitor

measurement set, the test simulation within SIMULINK is concluded. A window which

provides an interface to the post-simulation analysis routines is then used to download the data for measurement analysis. A comparison of frequency response functions for the simulated data and the experimental data for this test case is shown the following figures, Figures 17. The results of the comparison between the simulated and the actual experimental data show the need for computational model validation. The simulated data, which is based on the FE dynamic analysis, has lower modal frequencies than does the experimental results. By using this experimental data in conjunction with the FE model, the computational model can be updated and used as a predictive tool. However, even with an inaccurate finite element model, the test was complete and well designed. Placement and sensors and actuators, as specified in the VETO, resulted in data from which the required modes could be extracted unambiguously.

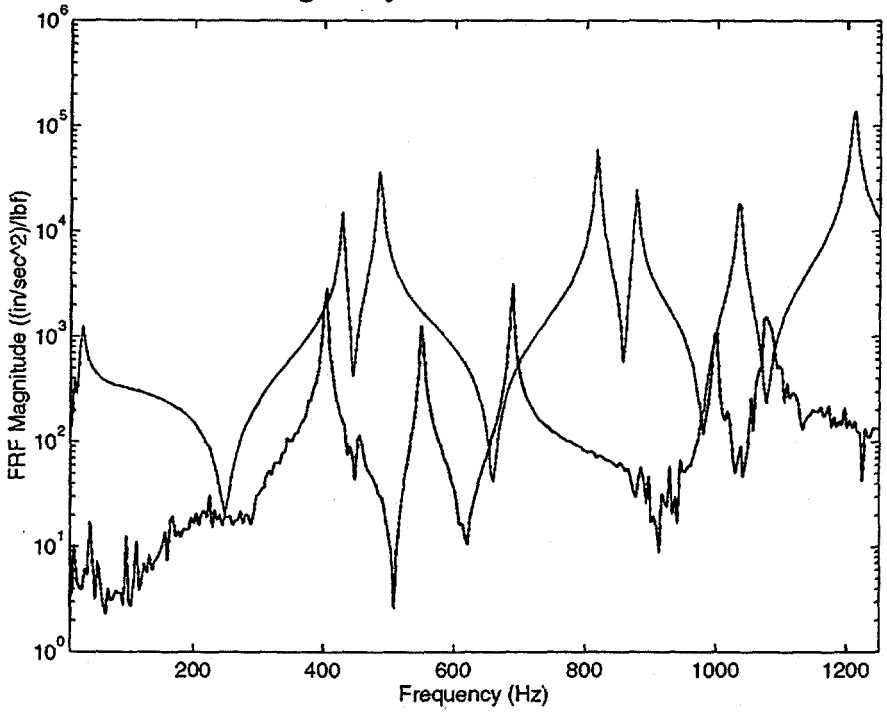

Figure 17. Comparison of Frequency Response Functions at node 3563

Conclusions:

The results of this weapon housing modal test design using the VETO environment clearly shows the benefit of this software tool. Within this software environment, engineers were able to investigate the testing of this component without the existence of any hardware. The effects that different instrumentation or equipment had on the results of the experiment were observed and the selection of appropriate analysis parameters were also studied. Another benefit to this VETO software was the optimal design of the experiment within the computer in order to provide experimental modal data for analysis model reconciliation. This tool assisted the engineers in the selection, placement and orientation of the instrumentation to maximize the information to be gathered from the experiment. Also, this tool allowed the visualization of results while iterating the test set up before committing to the actual test series. This test simulation tool, as previously described, plays an important part in the design of experiments for the purpose of computational model validation.

References:

[1] T. G. Carne and C. R. Dohrman, "A Modal Test Design Strategy for Model Correlation", Proceedings of the 13th International Modal Analysis Conference, 1995.

[2] D. C. Kammer, "Sensor Placements for On-Orbit Modal Identification and Correlation of Large Space Structures", Journal of Guidance, Control, and Dynamics, V14(2), 1991.

[3] L. A. Schoof, "Finite Element Analysis ViewER (FEAVR)". unpublished.

[4] L. A. Schoof and V. R. Yarberry, "EXODUS II: A Finite Element Data Model", Sandia Report SAND92-2137.

[5] P. S. Barney, "Power Polynomial Analysis Code". unpublished.

[6] J. P. Lauffer, "Eigensystem Realization Algorithm with Data Correlation II". unpublished.