SAND95-1566C
CONF-960353--2

Aerospace Modal Test Optimization Using VETO (Virtual Environment for Test Optimization)

S. E. Klenke, G. M. Reese, L. A. Schoof and **C. L. Shierling**
Sandia National Laboratories
Albuquerque, New Mexico 87185-0557

BIOGRAPHY

Scott Klenke is a member of the Experimental Structural Dynamics Department at Sandia National Laboratories. He was responsible for development of test instrumentation and equipment models for the test simulation. Garth Reese is a member of the Structural Dynamics Department at Sandia National Laboratories. He performed the FEM modeling in addition to developing the main graphical interface used in this software environment. Larry Schoof is a member of Computational Mechanics and Visualization Department at Sandia National Laboratories. He was responsible for the development of the visualization environment used in the software. Craig Shierling works for RE/SPEC Inc. He supported this project through programming and system integration work.

ABSTRACT

We present a software environment integrating analysis and test based models to support optimal modal test design of aerospace components through a Virtual Environment for Test Optimization (VETO). A goal in 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. The VETO assists analysis and test engineers in maximizing 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 an aerospace component 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.

KEYWORDS

Simulation, modeling, test optimization, virtual test environment, visualization

INTRODUCTION

This paper presents an innovative test/analysis tool, called the Virtual Environment for Test Optimization (VETO), which reduces test instrumentation iteration, producing better modal tests. Communication between test and analysis engineers is enhanced early in the design cycle. Traditionally, the role of testing in the product realization process is 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 are only considered when the hardware is scheduled for testing. Thus, the full benefit of the analysis in guiding the test is not realized. A goal in 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

---

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

‡MATLAB and SIMULINK are trademarks of The Math Works, Inc., Natick, MA.

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.

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 develop. 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 is reduced to state space ABCD matrices for use in the time integration module.

PROGRAM INTEGRATION - UNIX™ SOCKET CONNECTED MODULES

The VETO program is divided into three main modules: 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 module 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 modules 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; there is no direct access to internal data structures. This is a significant limitation of the software. Most of the structural data were shared through EXODUS II [4] files used by the AVS visualization software. The random access data features of this format were 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 modules had important side benefits. Unix sockets are network transparent, permitting us to run the application segments on different machines. For example, AVS could 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, see Figure 1. The "File" option of the **vetomain** menu bar allows users to load finite element (FE) models and previously defined virtual test files into the VETO software. This module also provides numerous tools to assist the engineer in understanding how the various virtual instruments interact together.

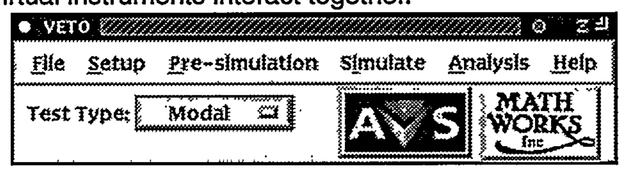

Figure 1. VETO Main User Interface

The user interface is implemented in C and 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 and 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 module 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

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

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, see Figure 2. A typical control panel used in the design of the virtual actuators and sensors, which are to be in contact with the device under test (DUT), is shown in Figure 3. Before initiating a simulation run, the user also needs to define additional instruments, 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. A special instrument called a "wire" is used to connect multiple instruments together. Once assembled, the DUT, selected virtual instruments and wire connections form the virtual test environment.

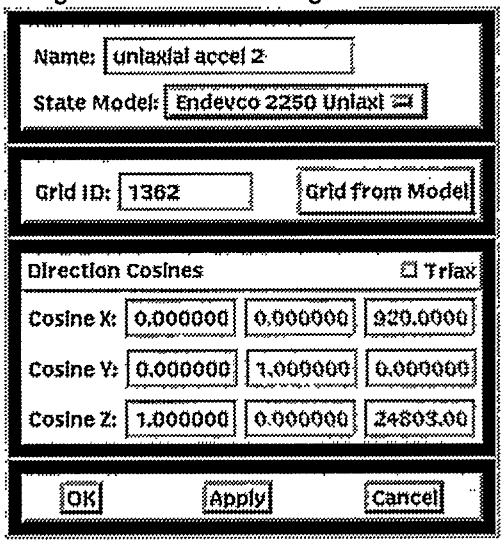

Figure 2. Sensor input panel

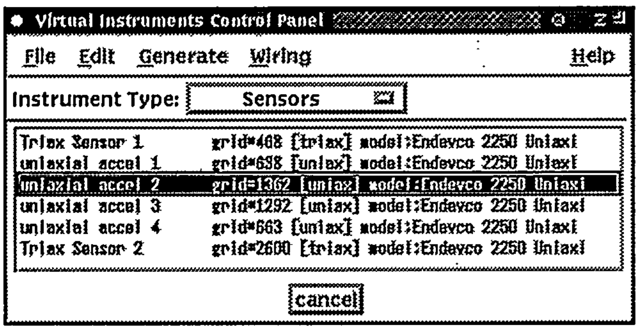

Figure 3. Virtual Instrument Control Panel for sensors

Data defining the virtual instruments 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.

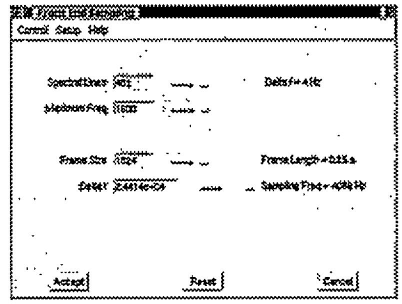

Figure 4. VETO Front End Sampling Panel

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†, 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 the design of tests that will accurately identify all the desired modes of the structure.

Even with these tools, placement of sensors 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

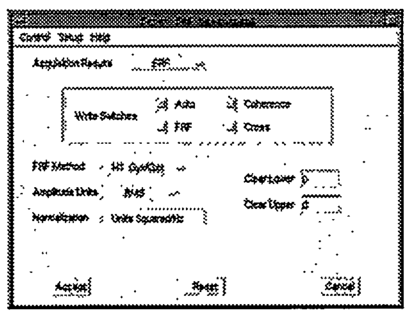

Figure 5. VETO Front End Measurement Panel

---

†The MAC is a normalized measure of mode orthogonality. It is defined by,

$$MAC_{ij} = \frac{\left(\varphi_i^T \cdot \varphi_j\right)^2}{\left(\varphi_i^T \cdot \varphi_i\right)\left(\varphi_j^T \cdot \varphi_j\right)}$$

where φ_j is the eigenvector of the j^{th} mode. The MAC is most often used to determine correspondence between test and analysis. In this effort, it identifies completeness of the modes. The MAC is sampled only at sensor locations, hence an incomplete sensor set results in large off-diagonal terms.

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 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 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 node point (via a mouse click) with a user specified orientation. For example, as an analyst or test engineer reviews the mode shape 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.

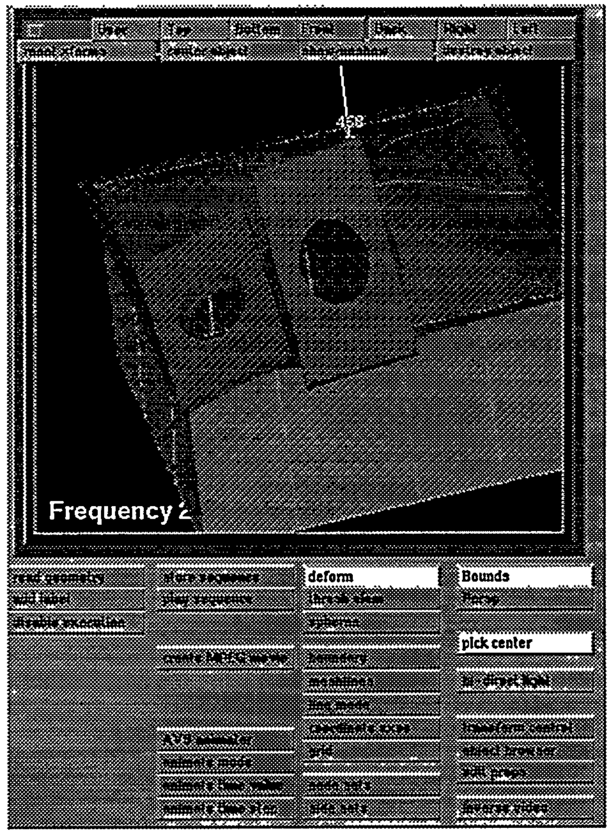

Figure 6. FEAVR display. Note virtual instrument at node 468.

The second extension to FEAVR was the capability 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 trace links and virtual instruments.

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 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 test equipment need to be developed. These models consist of a mathematical description of the dynamic response of the instruments derived either theoretically

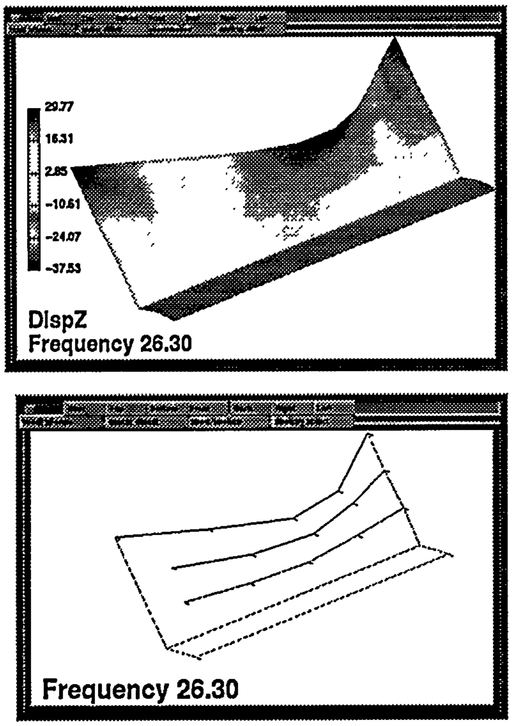

Figure 7. Deformed FE model and trace lines from FEAVR.

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.

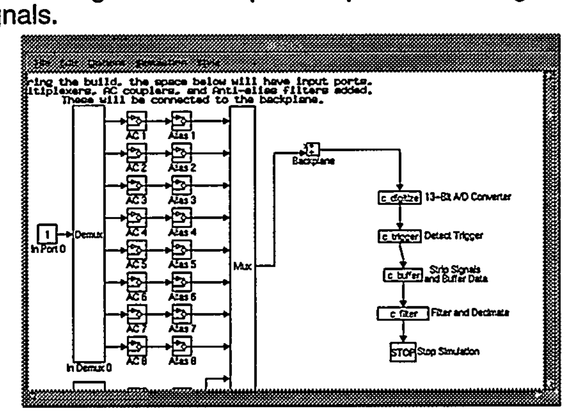

Figure 8. Partial Front End Block Diagram

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 fourth 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. 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.

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 also activate a window providing 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 OF AN AEROSPACE COMPONENT

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 for investigation into areas of design/analysis/test interfaces, visualization, versatility and repeatability. This initial VETO effort has focused on assisting engineers to maximize the value of modal tests.

An aerospace component was selected as a test case for application in the VETO environment. The VETO software simulation tool was used to design an optimal experiment for this mock reentry vehicle, Figure 9. The goal of performing this test design optimization was to observe the vibration modes of interest and to study the interaction of the support flanges with the reentry vehicle housing. The initial steps in the test design were to select an appropriate set of instrumentation (including sensors and actuators) to perform a modal experiment within the VETO environment and to simulate a modal test on the aerospace component. A symmetric finite element model of the structure was loaded into the VETO environment for use in the modal test simulation. The test design was performed over a frequency band, up to 250 Hz, which included fifteen vibration modes of the reentry vehicle.

The outcome of the VETO test design "Setup" was to excite the structure using an impact hammer and to measure acceleration responses on the reentry vehicle at 40 different locations in order to characterize the dynamic behavior of the component. Approximately

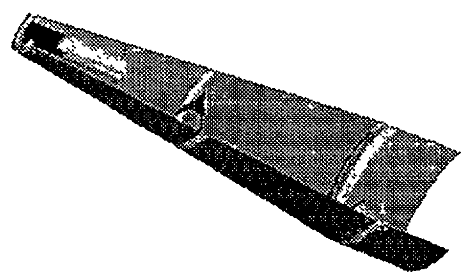

Figure 9. Mock Reentry Vehicle Model.

half of the response locations were automatically selected by using the **minamac** code to optimize the sensor locations. Some care was taken in utilizing this code to ensure that redundant or closely spaced response locations on the structure were not used in the simulation. A large number of accelerometer locations were selected in the test design to make the future 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 acquiring the desired analysis measurements were also selected for use in the post-simulation analysis.

A number of "Pre-simulation" tools were used to determine the completeness of the test design. First, the effects of mass loading the component were calculated given the test design sensor set. Small accelerometers, Endevco 2250s, were selected in the test design in order to minimize the mass loading effects during the experimentation. This analysis showed that very small changes in the frequencies of vibration (approximately 0.1%) would be experienced during an 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 aerospace compo-

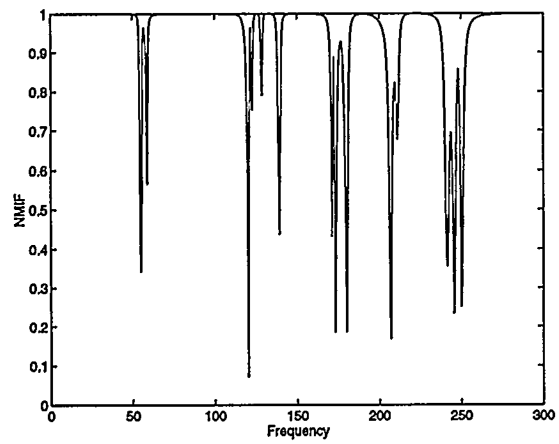

Figure 10. Normal Mode Indicator Function for the aerospace component

nent, Figures 10 and 11. By using the normal mode indicator function, it was determined that a single input location at the nose of the reentry vehicle would not excite all of the modes of the structure. Therefore, two

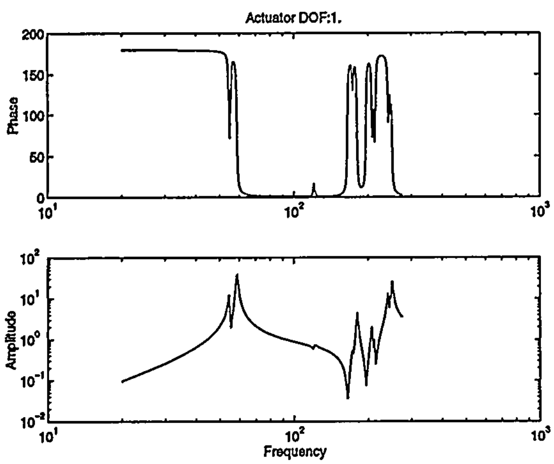

Figure 11. Driving point frequency response for the aerospace component

additional excitation locations were included in the test setup so that all the modes of vibration of the reentry vehicle could be observed. 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 from one another given the selected sensor set. Small values on the off-diagonal terms of this MAC matrix, Figure 12, indicate the relative independence of the modes of vibration, thus facilitating correlation with analysis.

With the complete test design within the VETO environment, a SIMULINK block model of the test environment is automatically generated to support the simulation of the modal test. Figure 13 shows a partial block model of the SIMULINK environment.

The next step in the modal test simulation is the numerical integration of the mathematical models

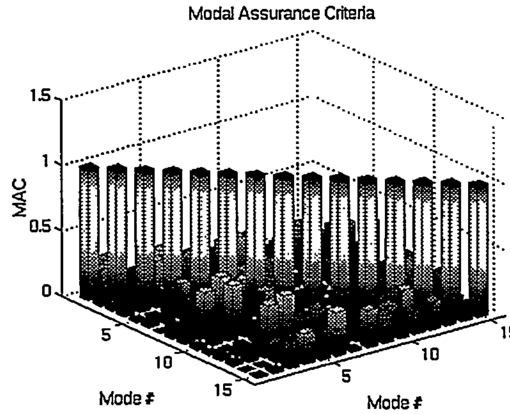

Figure 12. Modal Assurance Criterion (MAC) of the aerospace component

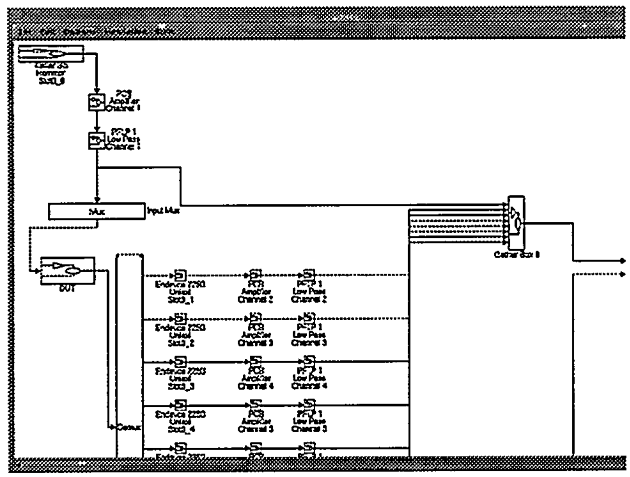

Figure 13. Partial Block Model of SIMULINK environment.

within SIMULINK to estimate the system responses. Using the Simulation Monitor, these responses are observed for each set or frame of data to be collected, Figure 14. Once the data are gathered to support the desired 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 number of analysis routines for computing desired measures such as frequency response functions, power spectral densities and coherences are available. The simulated data, which are based on the FE dynamic analysis, are used to generate frequency response functions, Figure 15. Placement of sensors and actuators, as specified in the VETO, results in a test design from which the required modes can be extracted unambiguously.

CONCLUSION

The results of this modal test design using the VETO environment clearly show the benefit of this software tool. Within this software environment, engineers were able to simulate the testing of this aerospace component without the existence of any hardware.The

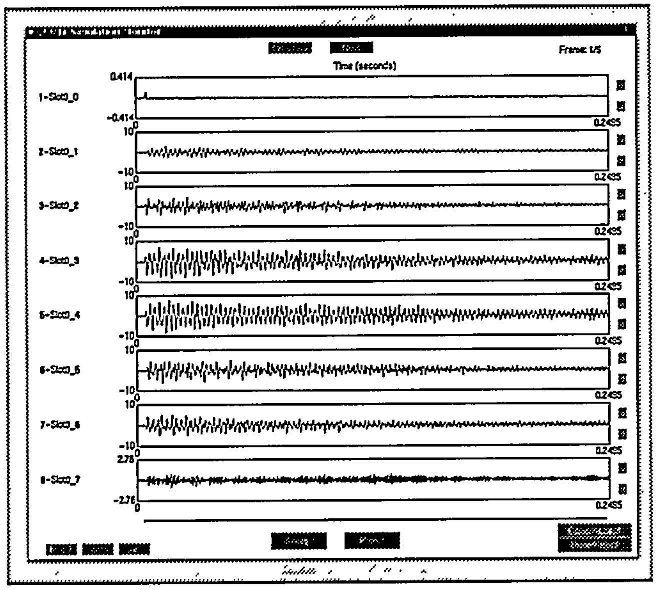

Figure 14. Collected Responses from Simulation Monitor

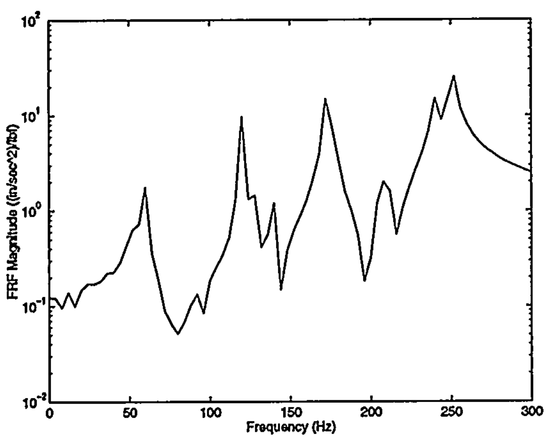

Figure 15. Simulated Frequency Response Function

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. 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 setup before committing to an actual test series. This test simulation tool, as previously described, plays an important role in the design of experiments for the purpose of computational model validation.

ACKNOWLEDGEMENT

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

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 ViewеR (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.