SAND2021-13012C

Rattlesnake: An Open-Source Multi-Axis and Combined Environments Vibration Controller

Daniel P. Rohe and Ryan Schultz

Sandia National Laboratories*
P.O. Box 5800 - MS0557
Albuquerque, NM 87123
dprohe@sandia.gov

Norman Hunter

Genuen Test Solutions
585 Owls Nest Rd.
Sequim, WA 98382
nfhunte@sandia.gov

ABSTRACT

The field of multi-axis vibration has grown significantly in the past few years, with new control algorithms being developed rapidly to provide more accurate responses with less force, voltage, and current requirements. Additionally, engineering efforts are increasingly considering combining environments, as components will often undergo simultaneous, combined environments in-service. For example a component may experience a multi-axis shock at the same time it is subject to multi-axis vibration. As no such "combined environments" controller currently exists, Sandia National Laboratories has developed its custom in-house vibration control system into a combined environments controller, which has been named Rattlesnake. Using a generalized environment interface, Rattlesnake allows an arbitrary number of environments to be included in a single test and allows users to set up a test timeline to automatically trigger testing events such as starting or stopping environments, removing the human-in-the-loop that often exists in combined environments tests. Rattlesnake currently has implemented MIMO random, MIMO transient, and Time History Generation environments, and users can readily extend Rattlesnake to additional environments. Within each of those environments, users can also easily modify the control strategies being used, or implement their own. Rattlesnake can run using NI or HBK LAN-XI hardware, or alternatively can be run in fully synthetic mode using a finite element model or state space representation to define the system. This work describes the typical Rattlesnake workflow and presents testing applications where the software has been successful.

Keywords: Multi-axis; Vibration; Control; Combined Environments;

*Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Figure 1: Rattlesnake Logo

1 Introduction

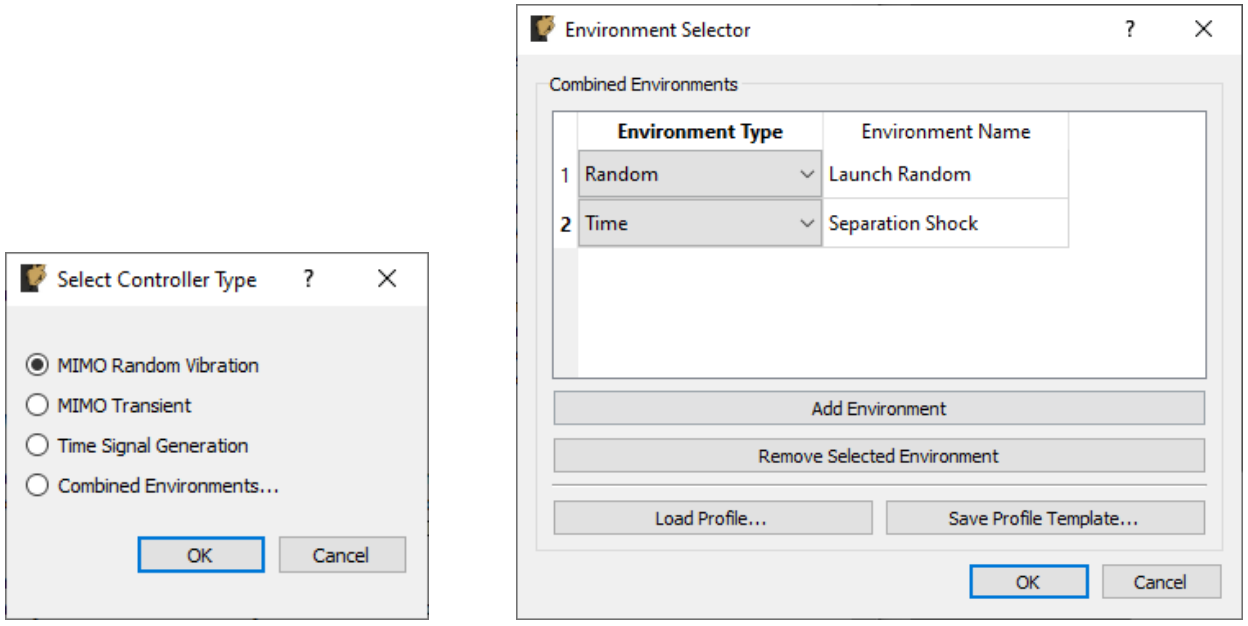
Multi-input, Multi-output (MIMO) vibration testing has seen an enormous growth in the past few years, as it has been shown to produce a more accurate test than traditional single axis vibration testing [1–3]. Additionally, engineers are beginning to realize the advantages of "combining environments" during testing. For example, during a rocket launch there are simultaneous acceleration, vibration, thermal, and shock loads that might affect a given component. Some laboratories have begun to implement combined environments in their test facilities, this could be as simple as covering a vibration shaker with a shroud of insulation so thermal testing can be performed simultaneously, or as complex as a "Vibrafuge" that puts vibration capabilities at the end of a centrifuge arm. As combined environments testing increases the complexity of current tests, the current paradigm of running a separate control system for each environment becomes more difficult. Therefore, Sandia National Laboratories has modified its in-house vibration control system [4] to accommodate multiple environments simultaneously. Nicknamed Rattlesnake (Figure 1), this software has been released open-source and can be downloaded at https://github.com/sandialabs/rattlesnake-vibration-controller. This paper will provide a high-level overview of the design, implementation, and applications of the software. More complete instructions for downloading and using the software can be found in the User's Manual, which can be found at the above GitHub website.

2 Controller Overview

Rattlesnake currently supports three environments: MIMO Random Vibration, MIMO Transient, and a simple, open-loop Time History Generator. These environments can be run individually or simultaneously in a combined environments fashion. When running with combined environments, users can set up a test timeline to specify when certain events, such as starting or stopping an environment, changing test level, etc. should occur. Rattlesnake supports two sets of data acquisition hardware: National Instruments (NI) through the NI-DAQmx interface, and LAN-XI through the OpenAPI interface. Additionally, Rattlesnake can perform synthetic control on a linear time-invariant system defined using either state space formulation or results from a finite element eigensolution.

When the controller is first opened, it prompts the user to select an environment (Figure 2a). The user can select a single environment or select `Combined Environments...` to combine multiple environments together in a single test (Figure 2b).

Once the environments are known, the main user interface appears, consisting of a tabbed window where the next tab becomes available after the previous tab is completed. The first tab requires the user to input their channel table and select data acquisition parameters (Figure 3). It is here that the data acquisition hardware is selected. Different hardware devices will have different parameters that can be set, so the graphical user interface (GUI) will update itself accordingly. For example, Figure 3 is currently set up for synthetic control using a finite element solution, so an `Integration Oversample` setting is shown, and portions of the channel table not required by the synthetic control are not filled out. Also on this tab is an environment table that allows users



(a) Selecting an environment in Rattlesnake **(b)** Selecting environments for a Combined Environments test.

Figure 2: Selecting environment in Rattlesnake

to specify which channels are active for each environment. For example, if the random environment uses different shakers than the shock environment, the channels can be set so that the random environment only utilizes the channels associated with its shakers.

Once the global data acquisition parameters are initialized, the parameters for each environment are initialized. For a combined environments test, there will be one sub-tab on this page for each environment (Figure 4). This tab is where all the signal processing and control parameters will be specified, in addition to the specification that the controller will attempt to match. The Transient and Random environments will also allow users to upload code that defines custom control laws.

At this point, the controller is completely defined, so a system identification can be performed (Figure 5). Note that the open-loop Time History Generator does not require a system identification, so no sub-tab exists for that environment. During this phase, Rattlesnake will output random signals and the responses from those signals will be used to create a relationship between the control channels and the output signals.

With system identification performed, the controller will use the user-defined control law to generate a test prediction, shown on Figure 6. The user can use this information to determine whether the test is feasible and accurate enough for their purposes prior to proceeding with the test.

The last setup step for a combined environments test is to specify a test timeline on the `Test Profile` tab (Figure 7). This is an optional step; without a test profile, the user can still manually start and stop the environments. For each event, the user will specify when the event happens (e.g at 10 seconds), to which environment the event will occur (e.g. Launch Random), and the operation to perform on that environment (e.g. Start Control). Some events, such as setting the test level, require additional data to be specified.

Finally the user can run the test. First, the data acquisition is armed, which starts outputting zero signals. The user can then start or stop the environments on the sub-tabs manually or run the test timeline specified on the previous page. There are various options to stream data to disk as well.

Data is written to the disk in netCDF file format, which can be read and analyzed using Python or Matlab. This output file completely describes the test; all metadata and test parameters set in the Rattlesnake software are stored along with the test data.

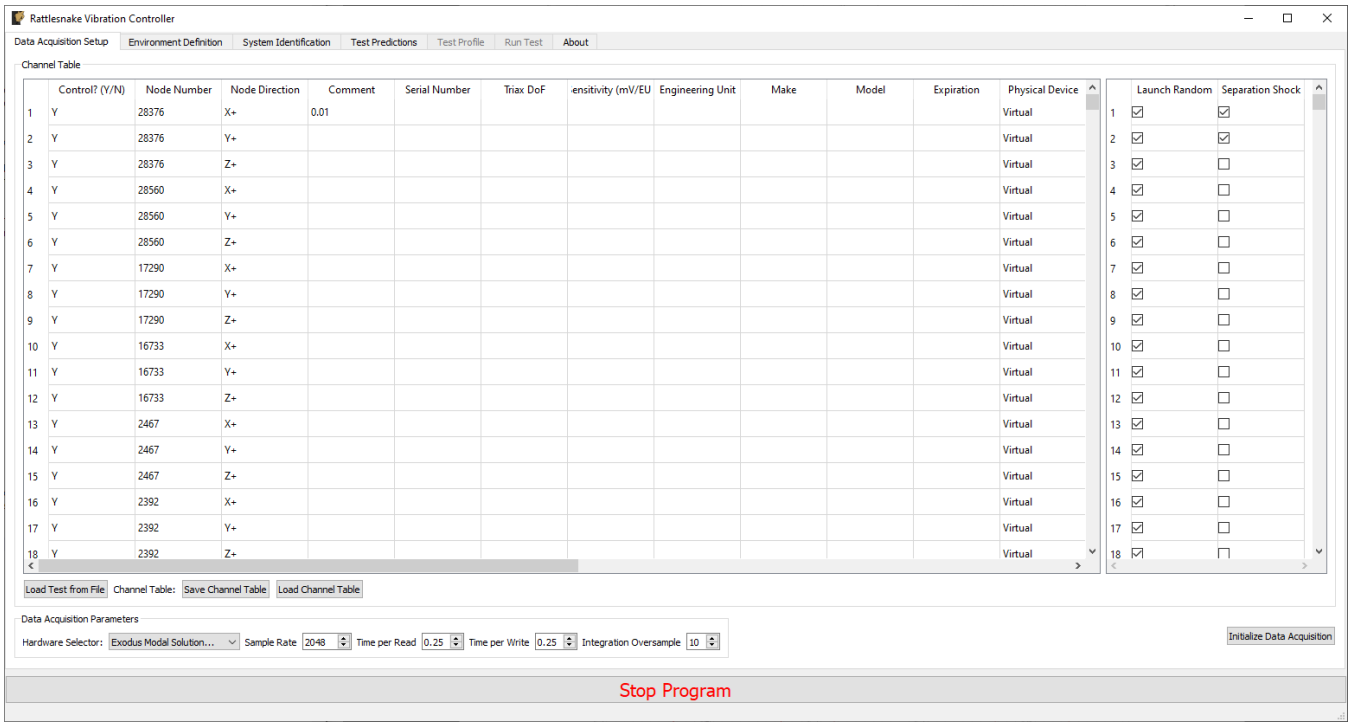


Figure 3: Setting data acquisition settings and channel table on the `Data Acquisition Setup` tab of the Rattlesnake controller

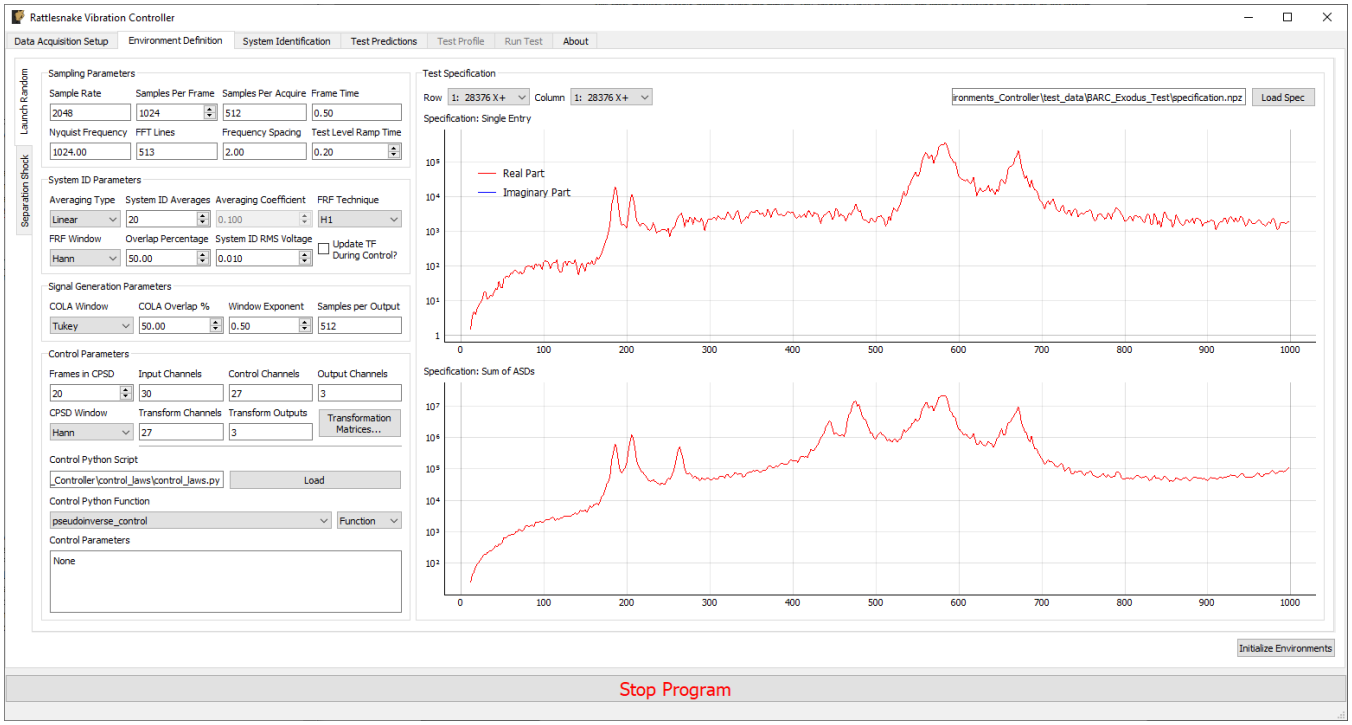
The Rattlesnake User's Manual on the GitHub page gives a more complete description of using the software.

3 Software Overview

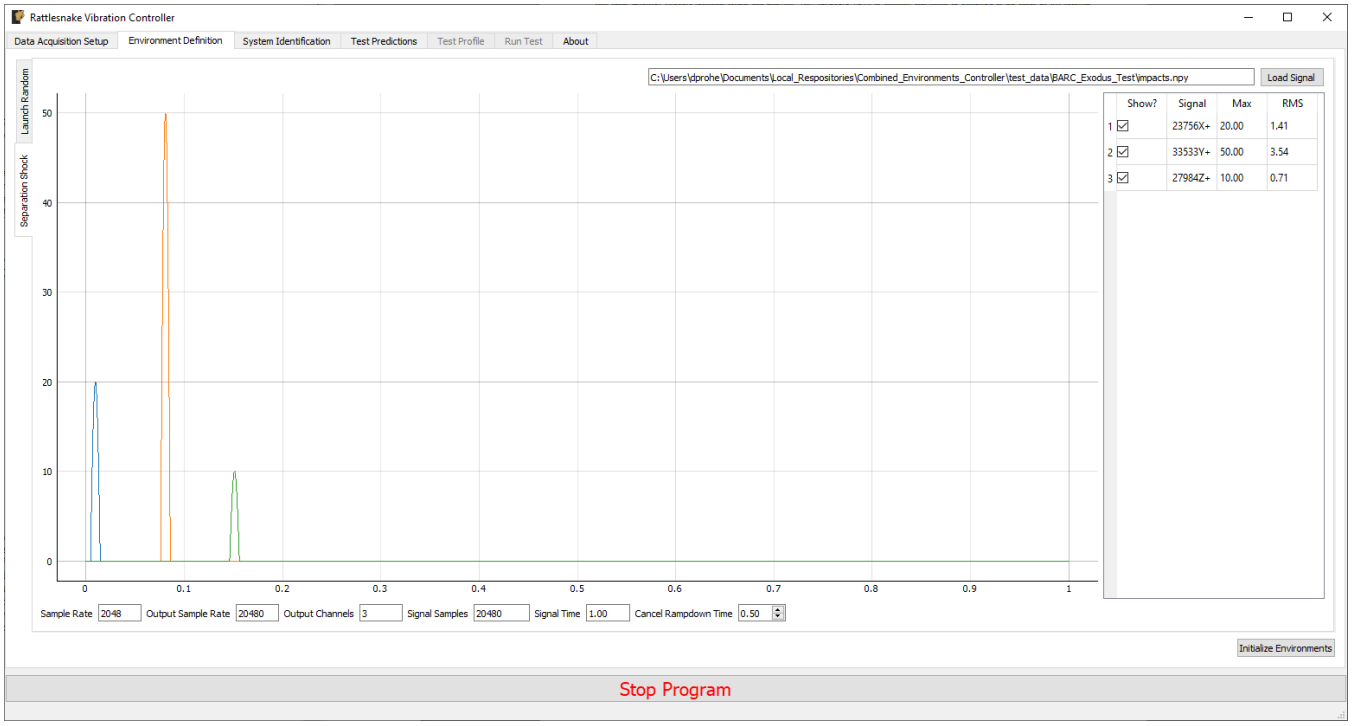
The Rattlesnake software is written in the Python programming language and designed to be flexible and extensible for future testing needs. It was designed particularly for research applications, as users can easily define new control strategies in the software via short Python functions or classes. For more complex additions, users can extend Rattlesnake to new environments or new hardware devices. For reference, a control law for Rattlesnake will generally be on the order of 10 lines of code, while a new environment will generally be on the order of 1000 lines of code. An overview of the Rattlesnake software is shown in Figure 9.

The key to Rattlesnake's flexibility is its abstracted interfaces to data acquisition hardware and environments. From Rattlesnake's perspective, all environments are identical. Rattlesnake will give the environment the previously acquired data, and the environment will give Rattlesnake back the next output data. From Rattlesnake's perspective, the implementation details of how a given environment generates the next output signal is not important, only that one is provided. Similarly with the data acquisition hardware interfaces, it does not matter exactly how the hardware interface gets data from or puts data to the data acquisition system, only that it can. When a user wishes to implement a new environment or hardware interface, they need not significantly modify the Rattlesnake source code; they need only to implement an interface to their code that is compatible with Rattlesnake.

Rattlesnake uses multiple processes to enable parallel computations of environments, acquisition and output, and streaming to disk. This means that the number of environments that can be used in a given test depends largely on the computational complexity of each environment, as well as the number of processors available on a computer. The user interface will generally communicate via the separate processes by passing messages and accompanying data to each process via queues. Each process has a "command queue" that it will read for instructions. In addition, each environment will have a "data in queue" where it listens for recently acquired data and a "data out queue" where it puts newly generated signals.



(a) Defining the random vibration environment



(b) Defining the time history generator environment

Figure 4: Defining the environment parameters on the `Environment Definition` tab of the Rattlesnake controller

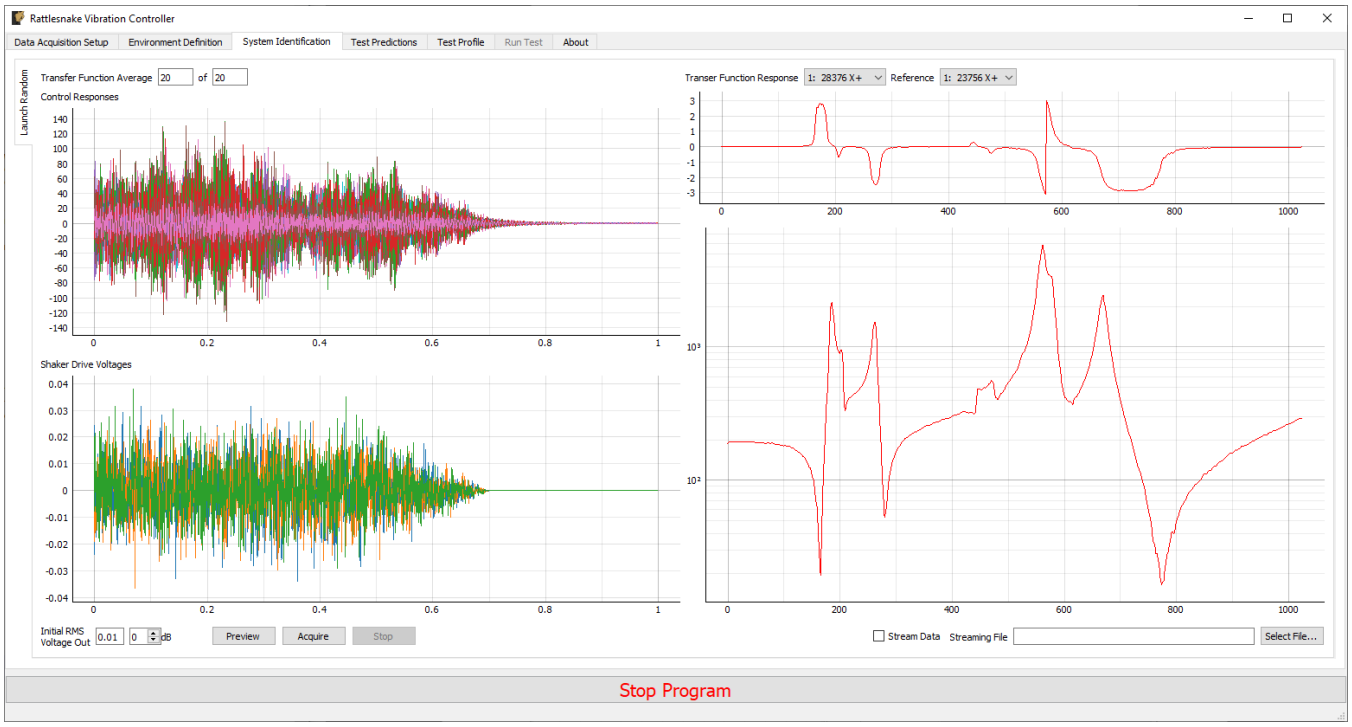


Figure 5: Running a system identification on the `System Identification` tab of the Rattlesnake controller

Figure 6: Visualizing the test predictions on the `Test Predictions` tab of the Rattlesnake controller

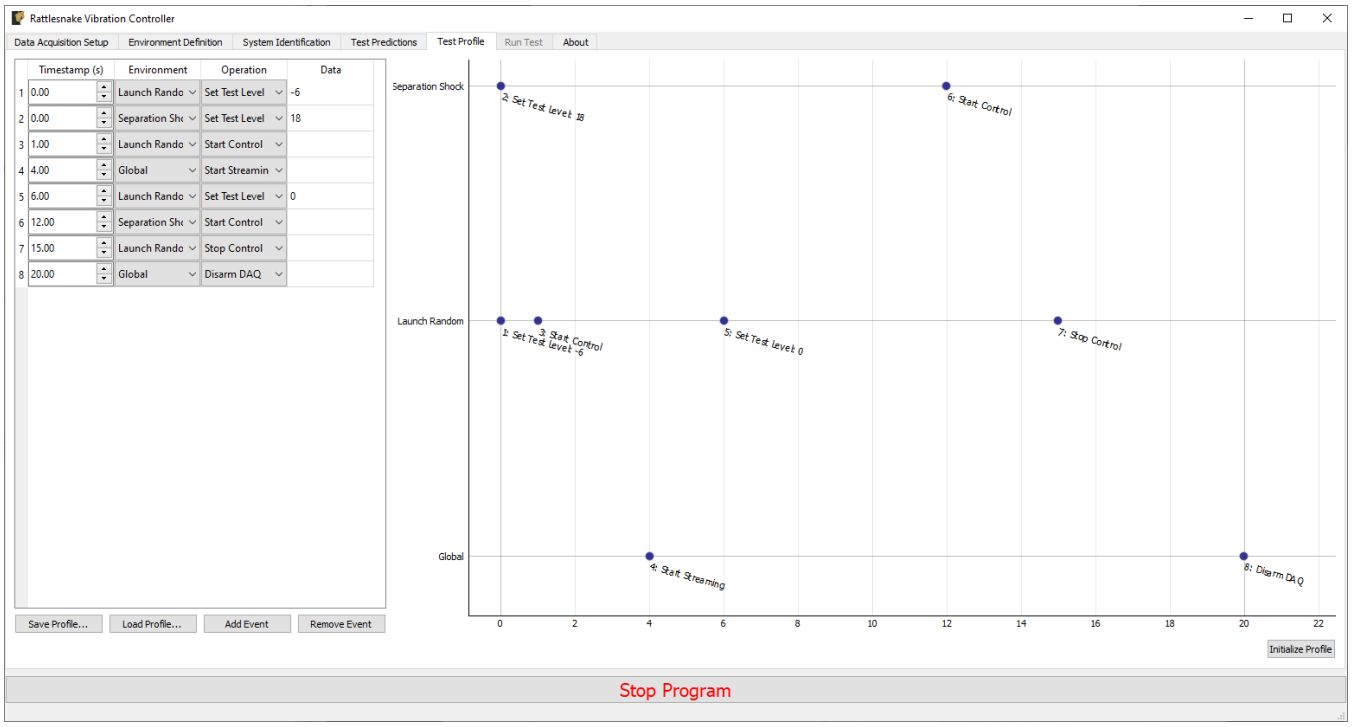


Figure 7: Setting up a test timeline on the `Test Profile` tab of the Rattlesnake controller

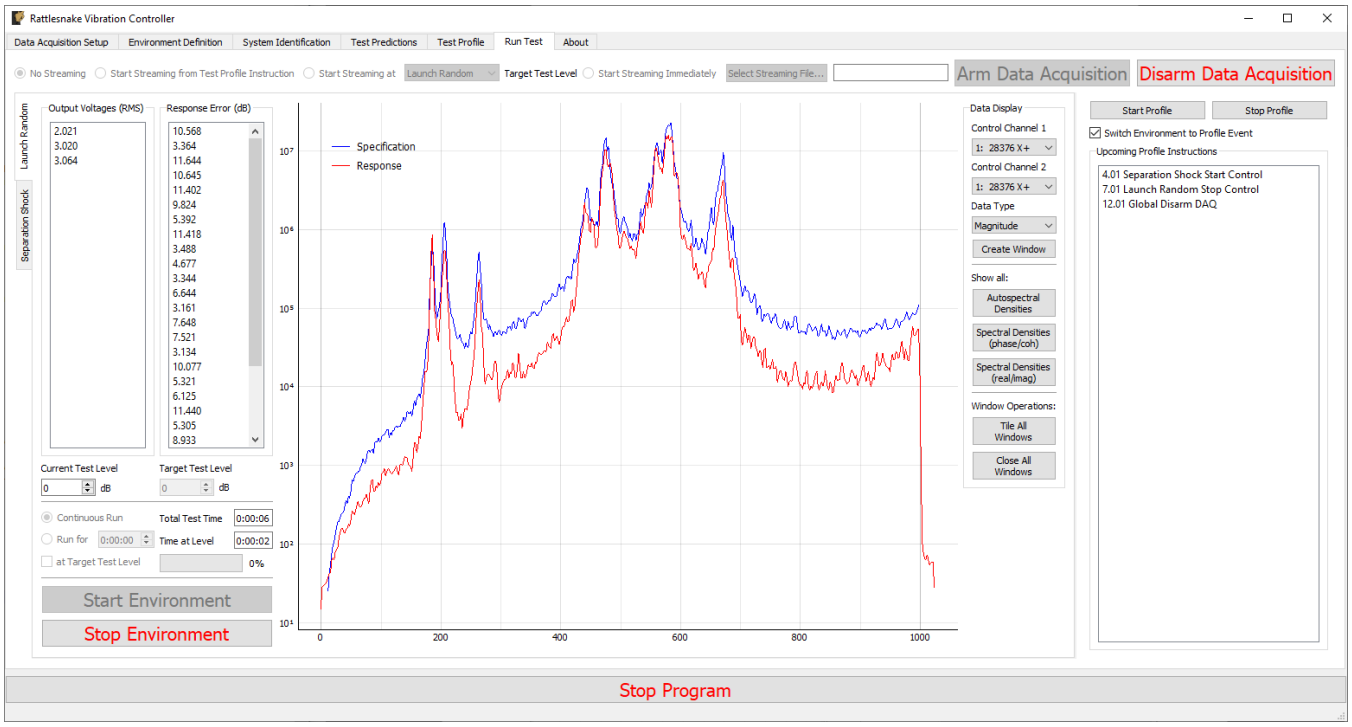


Figure 8: Running the test on the `Run Test` tab of the Rattlesnake controller

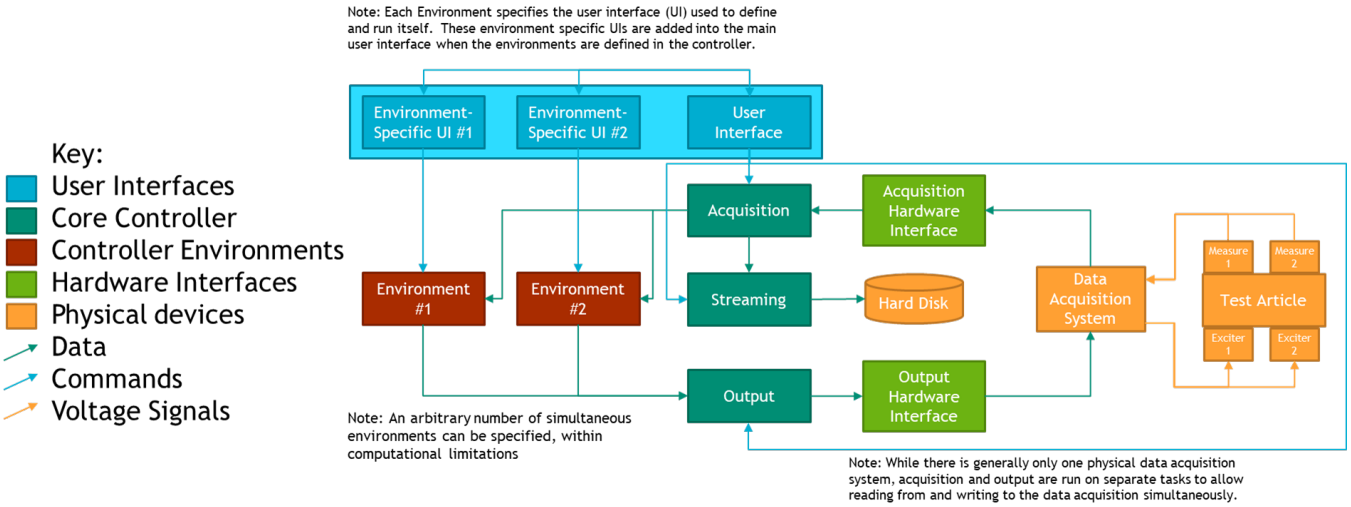


Figure 9: Rattlesnake software overview

3.1 Environments

Rattlesnake currently implements three environments, which are described more fully in the subsequent subsections.

3.1.1 MIMO Random Vibration

The first environment implemented in Rattlesnake is the MIMO Random Vibration environment. This environment takes as its specification a Cross-Power Spectral Density (CPSD) matrix and attempts to create responses on the test article that match that CPSD matrix. The random vibration environment operates several sub-processes to enable parallel computations of the updated response and output CPSD matrices, transfer function matrices, control calculations, and generation of the next time signal.

Control is performed using a user-defined control law loaded into the controller at run-time. These control laws can be defined as a Python function, generator, or class, and are generally reasonably short (<20 lines of code). Different control laws allow the test engineer to tailor the controller to the requirements of a given test, for example to maximize the test level or minimize the test error. The user defined control law must return an output CPSD matrix from which time histories will be generated. Closed loop control capabilities can be implemented by utilizing the previous drive or response signals to generate the next output CPSD matrix. A constant overlap and add (COLA) procedure is used to ensure no discontinuities between each time realization of the output CPSD matrix [5].

3.1.2 MIMO Transient Vibration

The next environment in Rattlesnake is a MIMO Transient environment. This environment takes as its specification a set of time history signals and attempts to create excitation signals that will force the control channels on the test article to match the specified time histories. The transient environment creates subprocesses during the system identification phase, so transfer functions can be computed in parallel with the signal generation. During control, the environment will not attempt to update the transfer function relationships, though control can still be updated using the previously acquired responses to the signal.

Similarly to the random environment, the transient environment allows user-defined control laws to be loaded into the controller at run-time. These control laws can also be defined as a function, generator, or class. The major difference is that the transient control laws must actually generate the output time signal that will be played to the exciters. The custom control law must also handle oversampling of the output if required by the data acquisition hardware.

3.1.3 Time History Generator

The final environment implemented in Rattlesnake is a Time History Generator environment. This environment simply plays a specified signal to the exciters and measures responses for a specified amount of time. No system identification is required, as the environment does not require any estimation of system dynamics to play an output signal.

3.2 Hardware Implementations

Rattlesnake currently implements interfaces to four data acquisition systems, two of which are real hardware, and two of which are synthetic hardware.

3.2.1 NI-DAQmx

Rattlesnake is able to interface with NI hardware that uses the NI-DAQmx[1] programming interface. This includes a number of NI devices, from standalone USB devices to large PXI/PXIe chassis devices. Since this was the first hardware device implemented in Rattlesnake, the hardware interface in Rattlesnake closely matches the NI-DAQmx programming interface.

3.2.2 LAN-XI

The second hardware device implemented in Rattlesnake is HBK LAN-XI hardware using the OpenAPI[2]. This interface communicates with the hardware via HTTP requests over an Ethernet cable. The LAN-XI hardware uses discrete sampling rates, and has a minimum output rate of 16,384 Hz. Therefore, Rattlesnake must oversample its output if the acquisition sample rate is set to 4,096 Hz or 8,192 Hz. Another difference between the LAN-XI and NI implementations is the ability to use multiple processes to acquire data off of the data acquisition system. Through experimentation, it was found for large channel count tests ($>$100 channels), the LAN-XI interface implemented in Rattlesnake struggled to pull down all data on one process fast enough, so a multi-process implementation was written to allow users to specify the number of processes used to pull data off the LAN-XI cards.

3.2.3 Synthetic Control using State Space Matrices

In addition to real hardware devices, Rattlesnake also offers the ability to perform control synthetically. This allows users to use Rattlesnake without requiring any actual hardware such as shakers or instrumentation. This can be valuable for developing new control laws; if the control law has the potential to go unstable, utilizing a synthetic setup can eliminate risk to expensive hardware such as shaker tables or test articles. Additionally, a given test can be investigated synthetically to determine feasibility prior the effort of actually setting up the test and attempting to run it.

To run synthetic control, Rattlesnake needs some kind of system defined. Utilizing state space matrices is one approach to define the system in Rattlesnake. The typical state space matrices are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2}$$

where $\mathbf{A}$ is the state matrix, $\mathbf{B}$ is the input matrix, $\mathbf{C}$ is the output matrix, $\mathbf{D}$ is the feedthrough matrix, $\mathbf{x}$ is the state vector, $\mathbf{u}$ is the input vector, and $\mathbf{y}$ is the output vector. Equation (1) defines how the state changes with time $\dot{\mathbf{x}}$ given the current state $\mathbf{x}$ and inputs $\mathbf{u}$ to the system. Equation (2) describes how the output degrees of freedom $\mathbf{y}$ are constructed from the current state and inputs to the system. When used within Rattlesnake, the output signals from Rattlesnake will be supplied to the system as $\mathbf{u}$, and the output degrees of freedom $\mathbf{y}$ will be acquired by Rattlesnake. The channel table in Rattlesnake should be set up consistently with the loaded state space matrices.

[1] See `https://www.ni.com/en-us/support/downloads/drivers/download.ni-daqmx.html` for NI-DAQmx documentation and driver downloads.
[2] See `https://github.com/hbk-world/` for documentation on the OpenAPI.

Rattlesnake then integrates the linear time-invariant equations of motion for the system using the `lsim` function[3]. Users of Rattlesnake can specify an oversample factor for the integrator to ensure that the integration is performed accurately enough. Because Rattlesnake will be integrating in real time, it is helpful to limit the sizes of the state space matrices. Since the state space equations are linear, it is generally advisable to convert a large finite element model to modal degrees of freedom prior to integration.

3.2.4 Synthetic Control using Finite Element Results

Rattlesnake can also produce a system from finite element model eigensolution results in Exodus file format [6]. Rattlesnake will construct state space matrices from the modal equations of motion, using the natural frequencies from the finite element model to populate the modal stiffness term and the mode shapes to transform physical forces from Rattlesnake into modal forces, as well as to transform the modal responses back into physical responses. Modal damping can also be specified. Once Rattlesnake develops a system of state space matrices from the finite element results, integration is performed identically to that in Section 3.2.3.

4 Applications

This section will briefly present successful applications of Rattlesnake.

4.1 Combined Environments Testing with Vibrafuge

Many aerospace components and systems undergo combined acceleration and vibration, such as a rocket launch, a capsule reentering the atmosphere, or an aircraft pulling a high-g maneuver. A number of test facilities have developed so-called "Vibrafuge" capabilities, which combine vibration and centrifuge acceleration for precisely these environments.

To investigate Rattlesnake's usefulness for these types of tests, it was fielded on a small centrifuge at Sandia National Laboratories. For this test, a small beam was was excited using piezoelectric actuators. These actuators were mounted to the centrifuge arm, with the excitation signal passing through the centrifuge's slip rings. An accelerometer was mounted to the beam for vibration control, and its signals were also passed back through the slip rings to the data acquisition system. Figure 10 shows the test setup.

The centrifuge drive motor controller accepts a voltage that is proportional to the rotational speed of the centrifuge, so this was controlled using the open-loop Time History Generator environment. The vibration environment was then controlled with the MIMO Random Vibration environment in Rattlesnake. This test utilized the test timeline capabilities in Rattlesnake to automatically start the vibration when the centrifuge spin reached full level. Historically, a test like this would have been run with multiple control systems each manually started; however, with Rattlesnake, the entire test can be run with one mouse click, enabling precise control of event timing throughout the test. A representative data set acquired from this test is shown in Figure 11 including the centrifuge control signal, piezo drive signals, and vibration levels in three axes.

4.2 Random Vibration using 6-DoF + Modal Shakers

A 6 degree of freedom (6-DoF) shaker has the ability to excite a structure in three rotational directions and three translations through the base of the structure. However, for some environments there may be other sources of excitation on the structure. For the example of the rocket launch, a base-mounted component might receive the majority of its excitation through its base, but there also may be acoustic excitation which does not originate from the base of the component. Therefore, it might be advantageous to supplement a 6-DoF shaker test with additional excitation sources such as speakers or smaller shakers to target motions from these additional excitation sources.

Two small modal shakers were attached, along with a test article [7–9], to a 6-DoF shaker at Sandia National Laboratories. Initial attempts using commercial vibration control software were unsuccessful, and without the ability to examine the code to see where the issues were occurring, the test engineer's ability to diagnose the issues were limited. The test engineer then

[3] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lsim.html`

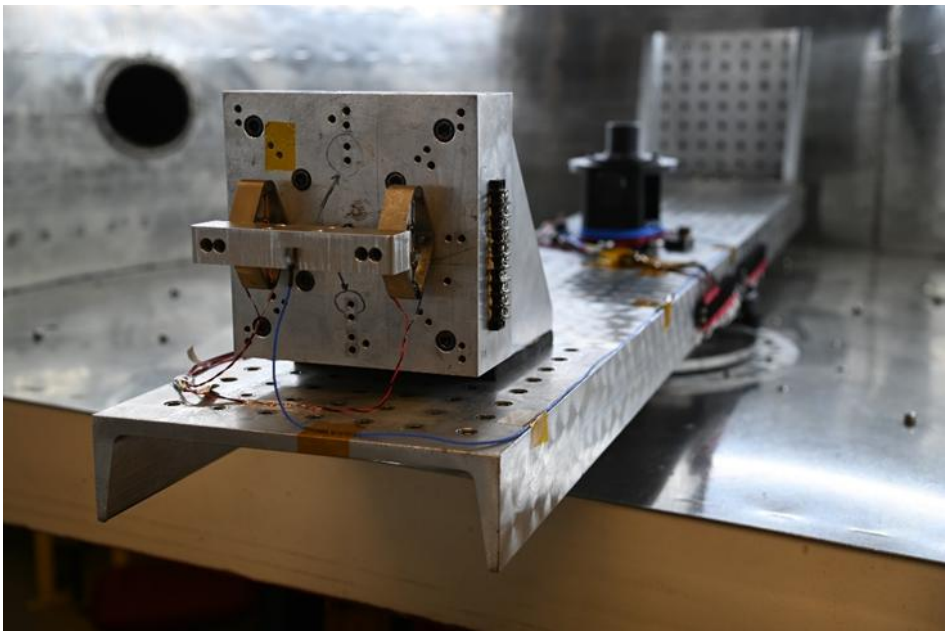


Figure 10: Vibrafuge test setup with beam mounted to piezoelectric actuators attached to the centrifuge arm.

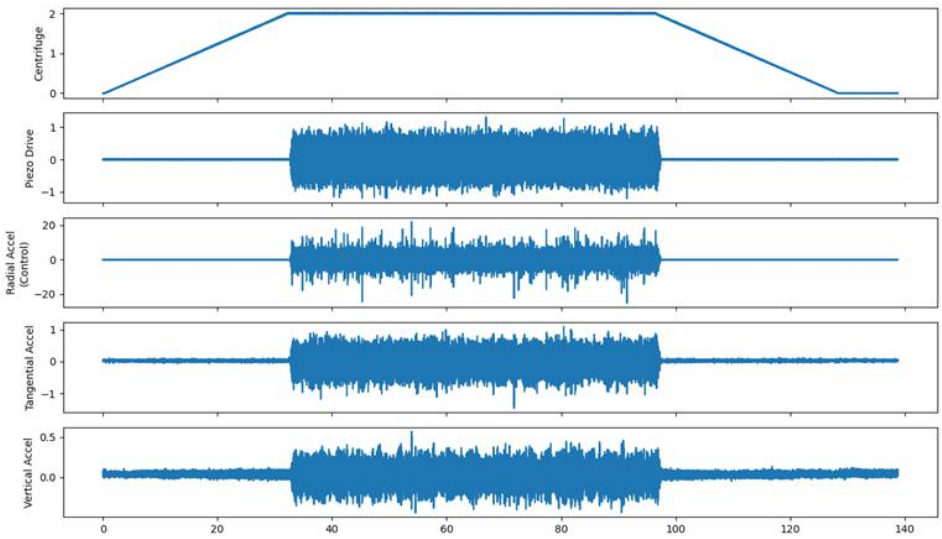


Figure 11: Time history data from the Vibrafuge test.

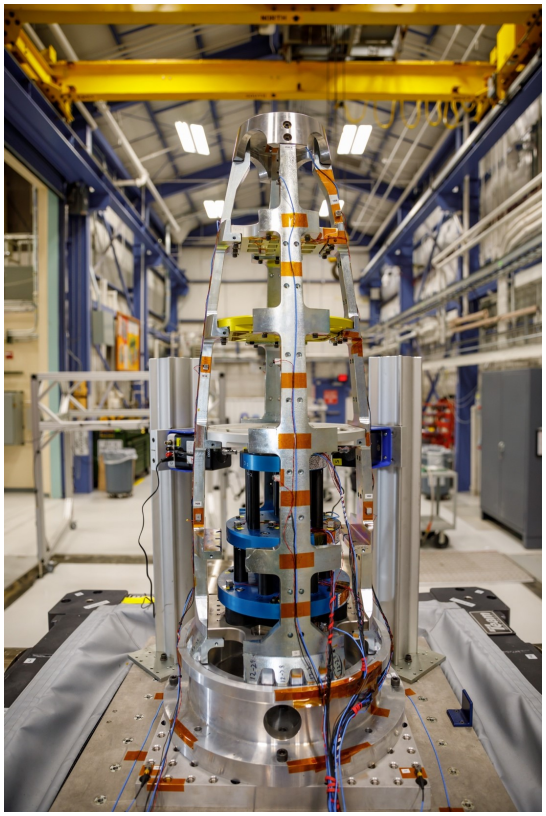


Figure 12: Test article on the 6-DoF shaker with two small modal shakers directly exciting the test article.

fielded Rattlesnake, and a custom control law was written to run the test. The test was able to be run successfully, and it was found that addition of the modal shakers improved the match to flight data over just the 6-DoF shaker.

5 Conclusions

This paper has introduced Rattlesnake, an open-source, combined environments control software, that has shown great promise to improve testing for complex testing scenarios. It can run multiple environments simultaneously. Existing environments in the software can run user-defined control laws, facilitating research by allowing users to rapidly iterate without writing a new controller from scratch or requiring a vendor to implement their ideas in existing commercial software. Rattlesnake is also extendable to new environments. Rattlesnake has been used successfully on a number of research projects that include novel excitation approaches and large channel counts. By releasing Rattlesnake open-source, the authors hope that it will become a tool that the entire vibration community can use to improve vibration research and development.

References

[1] P. Daborn, P. Ind, and D. Ewins, "Enhanced ground-based vibration testing for aerodynamic environments," *Mechanical Systems and Signal Processing*, vol. 49, no. 1, pp. 165 – 180, 2014.

[2] P. M. Daborn, "Scaling up of the impedance-matched multi-axis test (IMMAT) technique," in *Shock & Vibration, Aircraft/Aerospace, Energy Harvesting, Acoustics & Optics, Volume 9* (J. M. Harvie and J. Baqersad, eds.), Conference Proceedings of the Society for Experimental Mechanics Series, (Cham), pp. 1–10, Springer, 2017.

[3] R. L. Mayes and D. P. Rohe, "Physical vibration simulation of an acoustic environment with six shakers on an industrial structure," in *Shock & Vibration, Aircraft/Aerospace, Energy Harvesting, Acoustics & Optics* (A. Brandt and R. Singhal, eds.), vol. 9 of *Conference Proceedings of the Society for Experimental Mechanics Series*, (Cham), pp. 29–41, Springer, 2016.

[4] D. P. Rohe and R. Schultz, "A custom multi-axis vibration controller with flexible control strategies," in *Proceedings of the 39th International Modal Analysis Conference*, (Virtual), Feb. 2021.

[5] D. O. Smallwood and T. L. Paez, "A frequency domain method for the generation of partially coherent normal stationary time domain signals," *Shock and Vibration*, vol. 1, no. 1, pp. 45–53, 1991.

[6] L. A. Schoof and V. R. Yarberry, "EXODUS II: A finite element data model," Sandia Report SAND92-2137, Sandia National Laboratories, Sept. 1994.

[7] B. C. Owens, R. L. Mayes, M. Khan, D. G. Tipton, and B. Zwink, "Flight environments demonstrator: Part I – experiment design and test planning," in *Proceedings of the 37th International Modal Analysis Conference*, Society for Experimental Mechanics, 2019.

[8] B. R. Zwink, D. G. Tipton, B. C. Owens, R. L. Mayes, and M. Khan, "Flight environments demonstrator: Part II – ground trials of a sounding rocket experiment for characterization of flight," in *Proceedings of the 37th International Modal Analysis Conference*, Society for Experimental Mechanics, 2019.

[9] D. Fowler, R. Schultz, B. Zwink, and B. Owens, "Flight environments demonstrator: Part III – sensitivity of expansion to model accuracy," in *Proceedings of the 37th International Modal Analysis Conference*, Society for Experimental Mechanics, 2019.

A Acknowledgments

The authors would like to acknowledge David Siler, who was the principal investigator of the test described in Section 4.1, and Kevin Cross, who was the principal investigator of the test described in Section 4.2.