Chapter 7
Extending Rattlesnake Vibration Controller to MIMO Sine Control and Other Significant Improvements

Cody M. Langston and Dan Rohe

Abstract With the field of multi-axis vibration exploding in recent years, new control strategies and algorithms are being simultaneously developed to provide more accurate test responses with less force, voltage, and current required. Since the test articles often undergo simultaneous environments while they are in service, significant research is being performed to combine these environments together for more realistic laboratory testing. The Rattlesnake Vibration Controller was developed as an open controller framework to allow researchers to perform multiple-input, multiple-output, combined environments research and development without being constrained to closed-source commercial software. Since its initial presentation four years ago, Rattlesnake has become the workhorse for MIMO vibration testing at Sandia National Laboratories. Recently, Rattlesnake has introduced a new control environment, the MIMO Swept Sine environment. The goal of this environment is to sweep through frequencies and excite the structure so that the response at specified control points matches a sine wave with a predetermined amplitude and phase. Rattlesnake uses a MIMO swept sine algorithm that combines both feedforward and feedback control to generate a drive signal that produces the desired response. This algorithm can adjust to nonlinearities in the structure and noise during measurement to produce consistent responses. This paper outlines this algorithm and provides an example workflow on how to use Rattlesnake to implement MIMO Swept Sine tests for vibration testing. Additionally, this paper gives an update on Rattlesnake's development with an emphasis on transitioning Rattlesnake into a production quality software and expanding Rattlesnake's open-source community.

Keywords Multi-Input Multi-Output · Swept Sine · Rattlesnake · Open-Source · Vibration Testing

Introduction

Multi-input, Multi-output (MIMO) control is becoming a widely adopted alternative to single-axis testing for environmental qualification and modal tests. For vibration tests, MIMO control allows engineers to test how structures respond to inputs at multiple degrees of freedom simultaneously which is a better representation of the environments the structure will experience [5, 4]. An added benefit of testing the structure in multiple axes at a time is that it reduces the number of tests required to fully quantify the structure resulting in both speed and cost benefits [2].

Rattlesnake is the in-house MIMO vibration controller software that Sandia uses to perform environmental testing. Rattlesnake is open-source and free to download at https://github.com/sandialabs/rattlesnake-vibration-controller. Rattlesnake communicates with existing data acquisition hardware to collect data from input devices (accelerometers, load cells, etc.), processes that data to calculate output signals, and sends those output signals to output devices (shakers, motors, etc.). The software currently supports National Instruments, Lan-XI, and Data Physics data acquisition hardware as well as Exodus, Sdynpy, and State Space virtual hardware. Rattlesnake is structured around multiple distinct testing environments which can be run individually or together to support tests such as random vibration, hammer, or sine sweep tests. Five different

Cody M. Langston
University of Georgia, College of Engineering
e-mail: cml34135@uga.edu

Dan Rohe
Sandia National Labs, Experimental Structural Dynamics
dprohe@sandia.gov

Kristin B. Zimmerman, PhD (eds.), *Special Topics & Experimental Techniques in Structural Dynamics, 2026, Vol. 5*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.1007/978-8-743-81444-3_7

environments have been implemented in the current iteration of Rattlesnake: MIMO Random Vibration, MIMO Transient, MIMO Swept Sine, Modal, and Time History environments.

The most recent improvement to the Rattlesnake software is the addition of the MIMO Swept Sine environment. The goal of MIMO swept sine control is sweep through forcing frequencies to excite the structure in a way so that its response matches a sine wave with predetermined amplitude and phase. The sensors used to excite the structure are called drive signals and the sensors which respond with the desired sine waves are called control signals. One advantage of this testing technique is that the amplitudes of the measured responses are predetermined which allows engineers to design tests with high excitation levels and low signal to noise ratio resulting in higher quality FRFs. The aerospace industry regularly uses MIMO swept sine control when testing fully assembled structures as it allows them to guarantee that the structures will respond at safe levels [2]. This testing requires the usage of both open and closed loop control where the output signals are initially predicted by performing a system identification on the structure, and then adjusted during the test when the structure's response deviates from the desired amplitude and phase. The closed loop portion of the control can be used to classify non-linear behavior in the structure as it adjusts for differences between the response of the structure and the prediction made with linear assumptions. Some non-linear behavior is common during these tests due to their ability to safely excite the structure at higher levels than other tests.

Background

Rattlesnake's MIMO Swept Sine algorithm follows the structure outlined by [4]. This algorithm can be separated into four steps: specifying the desired response at the control channels, performing a system identification to predict the drive signals, choosing a method to track the amplitude and phase of the signal during the test, and performing the feedback control to adjust the drive signal when the control signal deviates from the desired signal.

The first step is to define the number of control channels, l, and number of drive channels, m, that will be used in this test. The desired signal is defined by specifying an amplitude and phase at each frequency for each control channel as well as the sweep rate between frequency breakpoints that determines the instantaneous frequency over time. Once the desired signal is defined, a system identification step must be performed where initial FRFs are computed with a low amplitude random vibration test. These initial FRFs are used to predict the drive signal at each frequency using the following relationship:

$$y = Hu, \tag{1}$$

where $y \in \mathbb{C}^{1\times l}$ is the control response, $H \in \mathbb{C}^{l\times m}$ is the FRF matrix at that given frequency, and $u \in \mathbb{C}^{m\times 1}$ is the drive signal. Given a desired response, the required drive signal can be calculated with:

$$u_{opt} = \hat{Z}r, \tag{2}$$

where $u_{opt} \in \mathbb{C}^{1\times m}$ is the predicted drive signal, $\hat{Z} \in \mathbb{C}^{l\times m}$ is the inverse of the FRF matrix, and $r \in \mathbb{C}^{l\times 1}$ is the desired control channel response. $\hat{Z}$ is usually computed with a Moore-Penrose pseudoinverse since the FRF is not always a square matrix. Various regularization and weighting parameters may be defined to tune the inversion process.

Due to non-linearities in the structure and error in the initial FRF calculation, the predicted drive signal will not generate the desired control signal. To account for this, the drive signal is adjusted in real time using a feedback control loop. The goal of this feedback control is to minimize the error:

$$e = r - y. \tag{3}$$

The desired response, r, is specified before hand, but the actual response, y, needs to be computed by tracking the amplitude and phase of the signal during data collection. A common way of doing this is by using an on-line frequency filter such as the harmonic estimator or digital tracking filter to estimate these values [4]. Another potential method of tracking the amplitude and phase is by using an Overlapped Vold-Kalman Filter outlined in [3, 6, 1].

The feedback control loop adjusts the drive signal by solving the following least squares optimization problem:

$$\min_{u} \quad e^H e = u^H H^H H u - u^H H^H r - r^H H u + r^H r. \tag{4}$$

This optimization problem is typically solved using a gradient descent method which results in the step:

$$u_{n+1} = u_n + \alpha H^H e_n, \tag{5}$$

where α is the convergence coefficient which determines the step size. [4] showed that this feedback control is guaranteed to be Bounded-Input Bounded-Output stable when α is calculated as:

$$\alpha_{opt} = \frac{2}{\sigma^2_{max}}, \tag{6}$$

where σ^2_{max} is the maximum eigenvalue resulting from the single value decomposition of the FRF matrix, H, which was calculated during the system identification of the structure.

Unlike many commercial sine control systems, Rattlesnake was designed from the start for "frame-based" data acquisition and output, similar to the ones that exist in standard Random Vibration Controllers. Instead of performing control on a sample-by-sample basis, Rattlesnake reads data in blocks (the size of which can be specified in the software). Additionally, when using the Vold-Kalman Filter, the measurement block may be constrained to a larger size due to processing and end-effects of the filter. Therefore, Rattlesnake will generally not be as responsive as other commercial control systems.

The current feedback implementation in Rattlesnake is a simple error correction routine. The complex error e at each frequency line in a measurement block is multiplied by $\hat{Z}$ to create a drive correction factor for that block. This is added to a running block drive correction factor after being scaled by the maximum drive correction Eq. 6. Future drive signals are modified by this correction factor.

Rattlesnake MIMO Sine Environment

Rattlesnake works as a wrapper around this control logic to allow MIMO Swept Sine tests to be set up and run seamlessly. To setup a MIMO Swept Sine test in Rattlesnake, choose the MIMO Sine Vibration option on the Select Controller Type dialog box shown in Fig. 1.

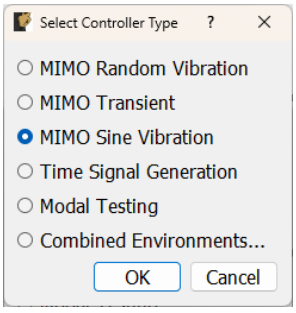

Fig. 1 Selecting MIMO Sine Vibration environment in Rattlesnake

Next, the user will setup their data acquisition hardware and its corresponding channels in the Data Acquisition Setup tab shown in Fig. 2. The Rattlesnake User's Manual found on GitHub has a comprehensive guide on how to set up a channel table. For this test, the channels that are defined with a Feedback Device will be used as the drive channels for the test.

The control channels are specified in the Control Channels table in the center of the Environmental Definition tab shown in Fig. 3. The relationship between the control and drive channels can be further defined by clicking on the Transformation Matrices button. The desired response from the control channels is defined with the Breakpoint table. This table consists of a series of breakpoints containing the desired amplitudes and phases of the control channels at the specified frequency. The desired response is generated by connecting these breakpoints through linear or logarithmic interpolation over time. Optionally, the warning and abort tables are used to define signals which will warn the user and abort the test if the system's response goes above those levels. During the test, the drive signals will start at the first breakpoint and stop at the last breakpoint. Additional sine tones are added by clicking the plus button on the tab group to the left of the Breakpoint table. These tones will run simultaneously and be superimposed on each other during the test. The Specification Plots at the bottom of the window are used to track which tones will be playing at a given time.

The filter used to track the amplitude and phase of the sine tones during the test is chosen with the Tracking Filter Parameters group. Rattlesnake currently supports a Digital Tracking Filter and an Overlapped Vold-Kalman Filter. After selecting a filter, the parameters required to define that filter will also be in this group. Clicking the Explore... button will allow the user to vary the filter settings and apply the filters to the specification signal to visualize the results, including in the presence of noise. This can allow the user to dial in the filter settings without actually running the test.

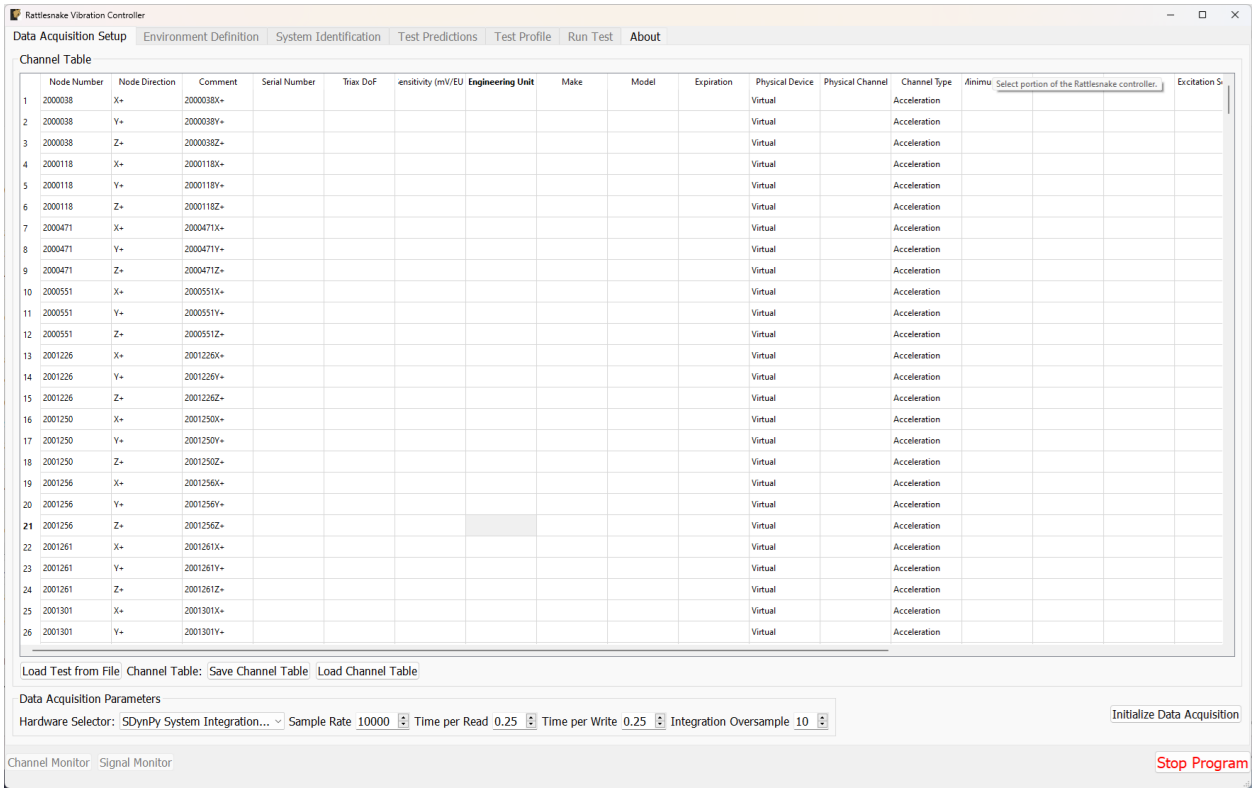

Fig. 2 Setting up data acquisition hardware and channels on the DATA ACQUISITION SETUP tab of Rattlesnake

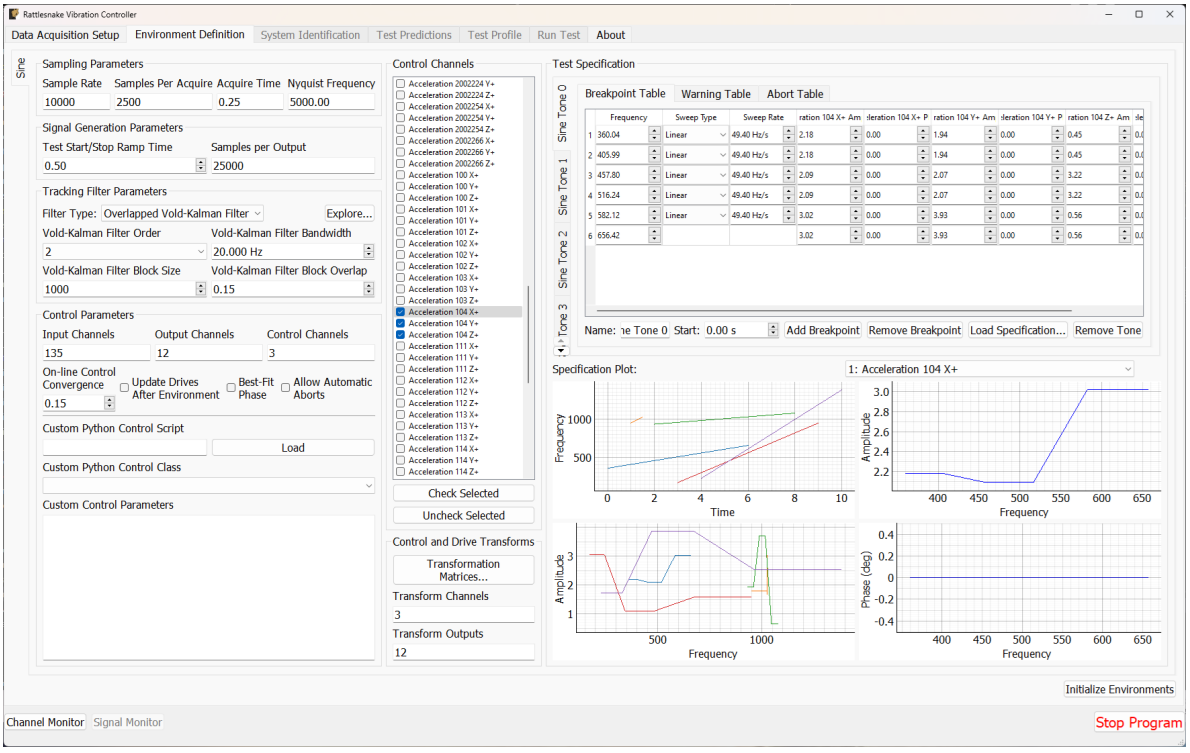

Fig. 3 Defining the desired sine signals and tracking filter on the ENVIRONMENT DEFINITION tab of Rattlesnake

The next step is to perform a system identification on to the SYSTEM IDENTIFICATION tab shown in Fig. 4. Here, the user will set up the random vibration test used to generate the initial FRFs. Once the test is defined, hit the Start button to perform the random vibration test and compute the initial FRFs.

Similar to the MIMO Random Vibration and Transient environments in Rattlesnake, the Swept Sine environment allows users to load in custom control laws defined as Python classes with specific methods defined (__init__, system_id_update, initialize_control, update_control, generate_signal, and finalize_control), fundamentally changing how the Sine environment operates without having to modify the environment itself. These Python scripts can be assigned to the test by clicking the LOAD button within the CUSTOM PYTHON CONTROL SCRIPT section.

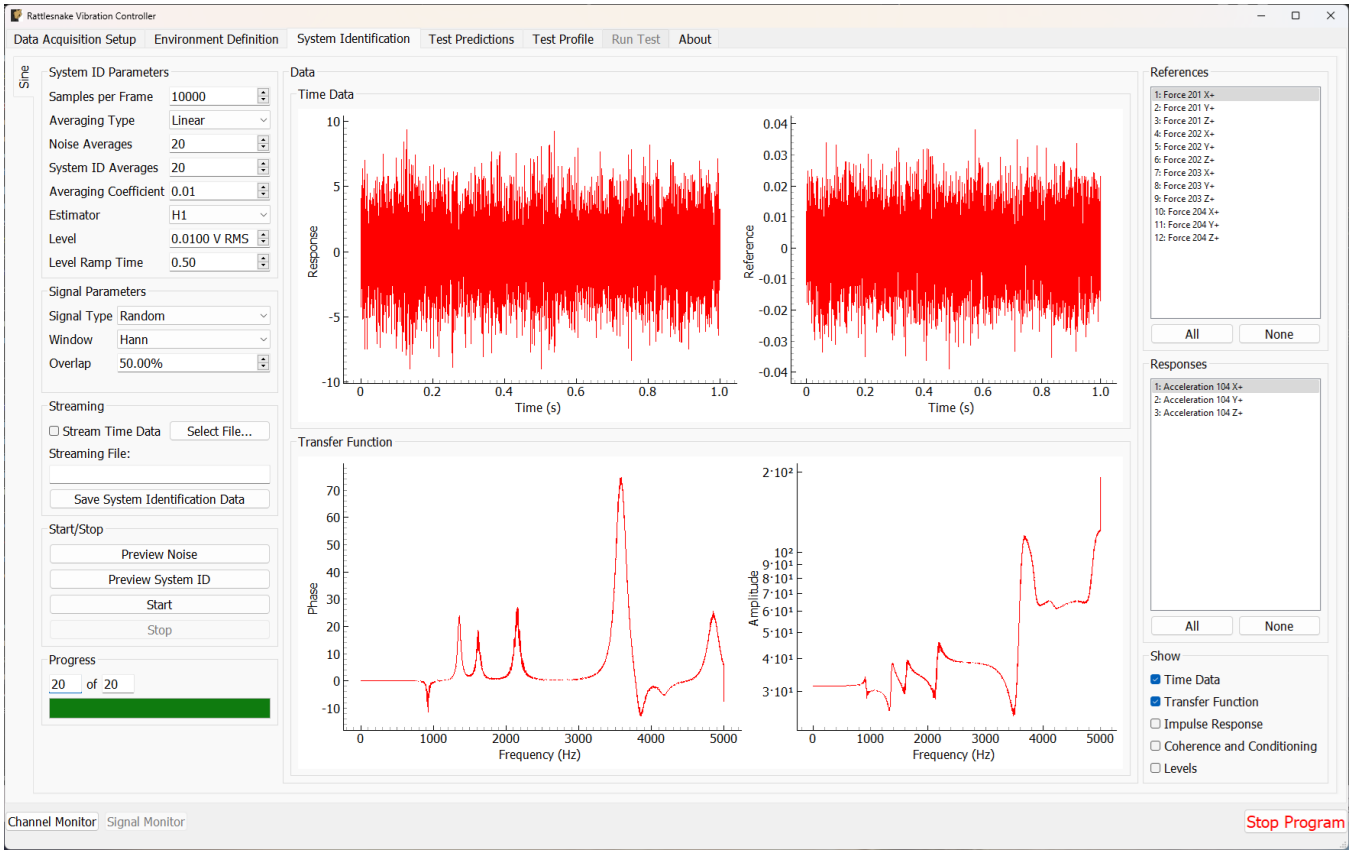

Fig. 4 Performing the system identification on the SYSTEM IDENTIFICATION tab of Rattlesnake

Performing a system identification will populate the TEST PREDICTIONS tab shown in Fig. 5. This tab is gives the user an idea of the accuracy expected from the current test setup and acts as a sanity check to make sure that the test will not cause harm to the structure or devices.

If the predicted signals meet the required specifications, the software is ready to run the full test using the RUN TEST tab shown in Fig. 6. This is done by clicking on the ARM DATA ACQUISITION and START ENVIRONMENT buttons. As the test runs, the amplitude of the sine waves at each control channel will be displayed with the desired signal for that control channel. The real (x-axis) and imaginary (y-axis) adjustments made to the predicted drive signal by the feedback control loop are shown on the CONTROL UPDATES plot in the upper left corner. After a complete sine sweep, Rattlesnake provides the ability to update the feed-forward drive signal to make corrections for the next pass, allowing a users to run the entire sweep at a lower level to dial in the response prior to the full-level test. The drive and control signals from the test can be saved to a NetCDF4 file for further processing.

The MIMO Swept Sine environment has been shown to be very functional, especially in conjunction with the combined environments capabilities in Rattlesnake. Recently, the capability was utilized to run a mixed mode test controlling to 9 separate sine sweeps each controlling unique X,Y,Z table translations for a 6-DoF shaker (27 total sine signals), while simultaneously controlling to a random vibration profile in the background. Rattlesnake is able to synchronize these sine components within a single MIMO Sine Environment, allowing the instantaneous amplitudes and phases to be accurately computed for all sine tones. The implementation of the Vold-Kalman filter for order tracking allows for these sine tones to

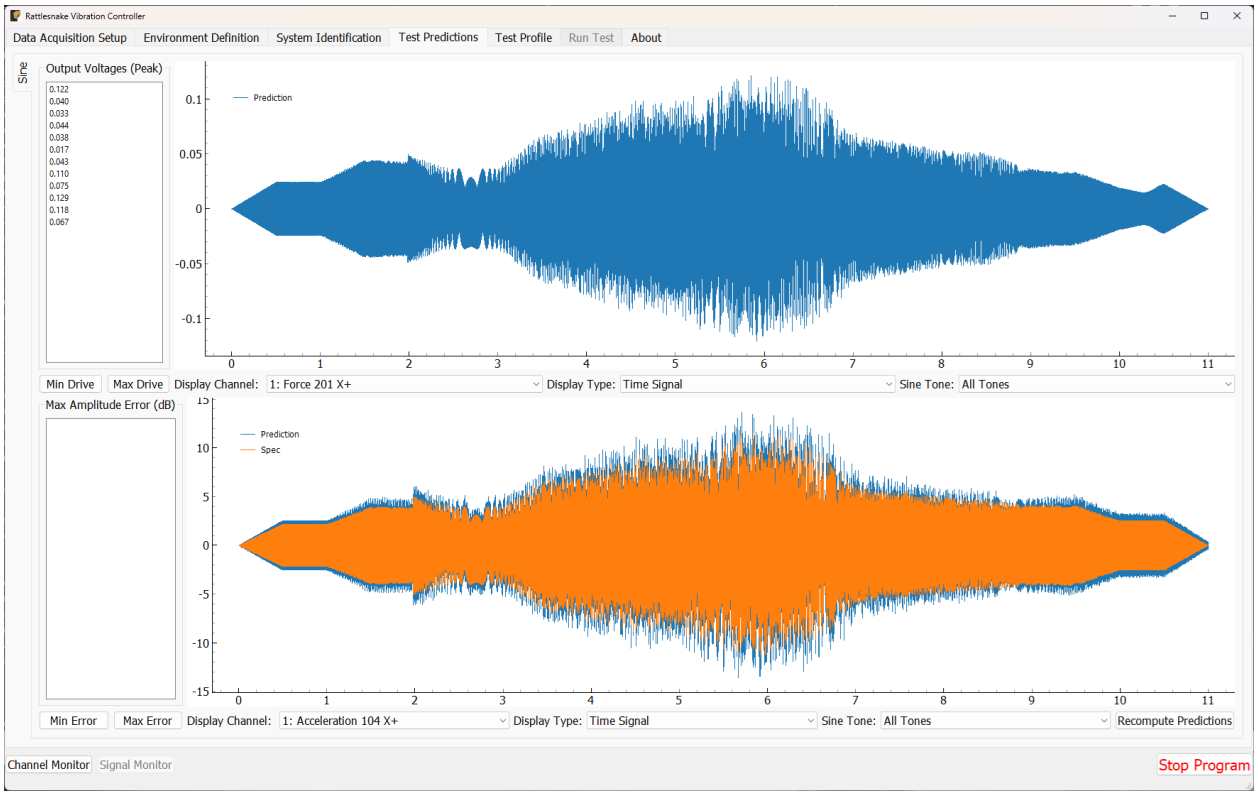

Fig. 5 Plottted test predictions on the TEST PREDICTIONS tab of Rattlesnake

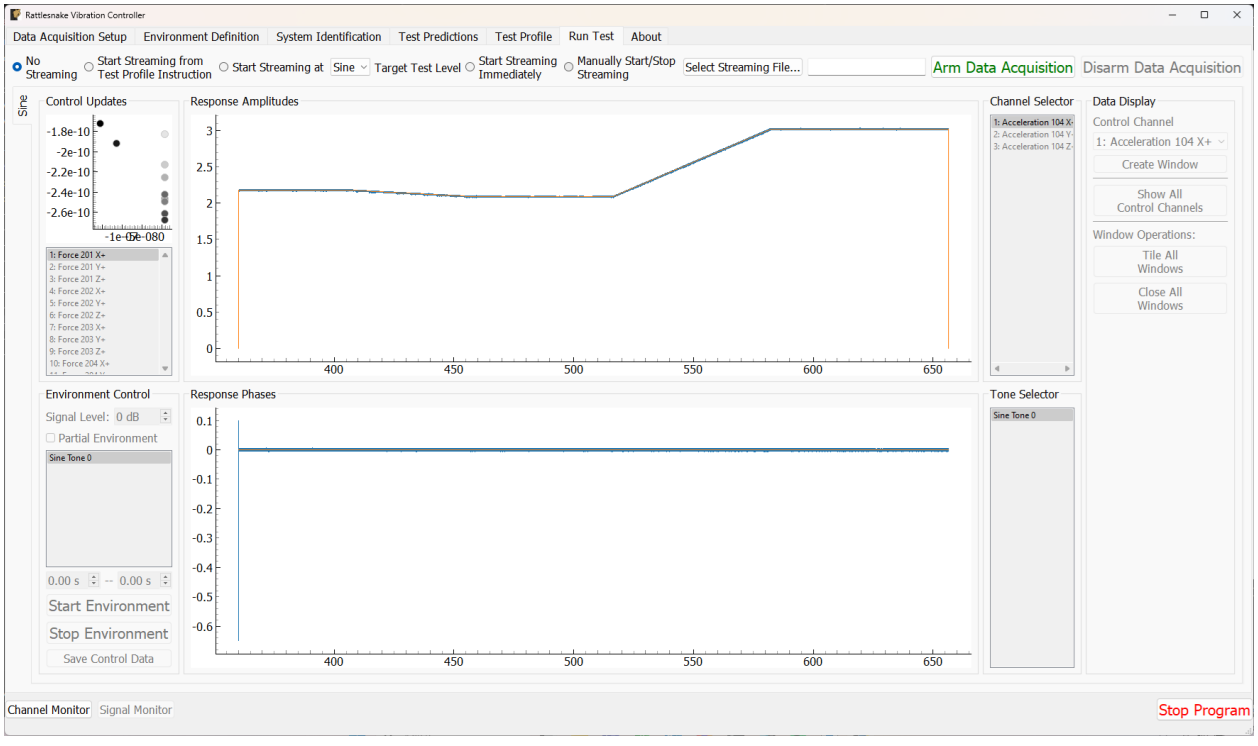

Fig. 6 Running the MIMO Swept Sine test on the TEST PREDICTIONS tab of Rattlesnake. The desired signal is orange and the measured siganl is in blue.

cross in frequency without significantly impacting the ability to extract the amplitude and phases of the tones. Rattlesnake's flexibility allowed this challenging test to successfully be performed.

Rattlesnake Development Updates

Rattlesnake is an open-source codebase written in Python that allows the user to customize it to their specific needs. The current structure of Rattlesnake is that it collects data from a Hardware class which defines how to communicate with the data acquisition hardware. The collected data is sent out to Environment classes which process the data and determine the drive signals if necessary. The Environment classes send the drive signals back to the Hardware class where they are combined and output to the data acquisition hardware. This structure is key to Rattlesnake's usability for research applications as it allows the user to focus on programming their own control laws or environments, while using the Rattlesnake's existing implementation for aspects such as hardware communication, data streaming, etc.

In the past year, the Rattlesnake development team has made significant changes to transition Rattlesnake into a production quality product. These changes include a major overhaul to the development cycle and improvements to the ease of use, stability, and capabilities of Rattlesnake. Previously, Rattlesnake has been developed in house at Sandia with the codebase being pushed to the public GitHub whenever significant changes were made to the software. To avoid the complexity of maintaining two separate codebases, Rattlesnake will now use the public GitHub as its main repository. This change means that external users will get updates sooner and opens up the potential for collaboration without having to merge changes between two separate codebases.

With this transition, Rattlesnake is moving to a Continuous Integration/Continuous Delivery (CI/CD) development pipeline where changes to the software will be frequently committed to a central development branch. This CI/CD pipeline includes the introduction of unit tests which automatically run isolated portions of the software to detect possible bugs introduced when committing to the codebase. These tests must be maintained as they act as a safety net that allow Rattlesnake to be iterated on frequently with confidence that the changes will not introduce new issues into the software. Currently, these unit tests cover 45% of Rattlesnake's codebase.

Rattlesnake's transition to a production quality codebase also includes improvements to the software's ease of use. Rattlesnake has been added to the pip package manager to make it easier to install. Additionally, Rattlesnake is being designed to validate the user's inputs which improves error messaging and prevents the user from crashing the software. With this validation, the Rattlesnake UI is now capable of suggesting valid inputs for values such as data acquisition hardware, channel names, and voltage ranges. The Rattlesnake documentation now has its own website with improved search functionality. The documentation can be found at https://sandialabs.github.io/rattlesnake-vibration-controller/book/.

The development team hopes that with these changes, it will be easier to implement Rattlesnake into the workflow of other laboratories. The open-source nature of Rattlesnake allows others to make meaningful contributions to the software that will help others. The transition to the public GitHub and introduction of CI/CD development streamlines this process improving version control and bug tracking. It is now easier than ever to join the Rattlesnake open-source community.

Conclusion

This paper outlines the recent addition of MIMO Swept Sine control to the Rattlesnake vibration controller. MIMO Swept Sine control allows engineers to design the response of a structure to match specifications required for their testing purposes. This allows the test to reach higher excitation levels without the risk of damage to the structure or sensors. Rattlesnake streamlines the process of conducting a MIMO Swept Sine test by allowing the user to define their drive and control channels, shape the desired response at those control channels, and define parameters that are required for the test such as the tracking filter and system identification. This workflow has been used to conduct MIMO Swept Sine tests at Sandia National Laboratories. This paper also provides updates on Rattlesnake's development cycle with particular focus on improvements made to transition Rattlesnake into a production quality software.

Acknowledgments Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges

References

1. J.R. Blough. Understanding the kalman/vold-kalman order tracking filters' formulation and behavior. 05 2007.
2. G. Gloth and M. Sinapius. Analysis of swept-sine runs during modal identification. *Mechanical Systems and Signal Processing*, 18(6):1421–1441, 2004.
3. H. Herlufsen, S. Gade, H. Konstantin-Hansen, and H. Vold. Characteristics of the vold-kalman order tracking filter. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 6, pages 3895–3898 vol.6, 2000.
4. Umberto Musella, Bart Peeters, Francesco Marulo, and Patrick Guillaume. Multi-input multi-output swept sine control: A steepest descent solution for a challenging problem. In Nikolaos Dervilis, editor, *Special Topics in Structural Dynamics & Experimental Techniques, Volume 5*, pages 83–94. Springer International Publishing, 2020.
5. Daniel Rohe, Ryan Schultz, and Norman Hunter. Rattlesnake: An open-source multi-axis and combined environments vibration controller. Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 10 2021.
6. Håvard Vold and Jan Leuridan. High resolution order tracking at extreme slew rates using kalman tracking filters. *Shock and Vibration*, 2:507–515, 01 1995.