# *OpenSD 2025 – Proceedings*

Ljubljana, 16. – 17. June 2025

*Edited by:*
**Janko Slavič**
**Martin Česnik**

# *OpenSD 2025*

Ljubljana, 16. – 17. June 2025
**CONFERENCE PROCEEDINGS**


**Edited by:**

Janko Slavič
Martin Česnik

**Reviewers:**

Gregor Čepon
Martin Česnik
Janko Slavič
Klemen Zaletelj

**Publisher:**

University of Ljubljana, Faculty of mechanical engineering

Aškerčeva 6, 1000 Ljubljana

June, 2025

**Pricing:**

Proceedings are freely available on the conference homepage.

# *Contents*

# PyMMM - Python Multi Model Manager

Matej Anko  Janko Slavič *

*University of Ljubljana, Faculty of Mechanical Engineering*

**Abstract**

Solving new problems with machine learning (ML), often requires development of several different ML models, before the optimal one is designed. Working on multiple models can quickly lead to scatter of models between different files in project folder. In addition to that, PCs used for developing and preliminary testing are usually not as powerful as workstations designed with training of ML models in mind, so it is desired that training of models can be paused and later resumed easily.

To overcome these challenges, the Python Multi Model Manager (pyMMM) package has been developed as an open-source Python framework for managing multiple ML model pipelines. The framework currently supports storing and loading different PyTorch models, have complete control over the training, such as start, stop, pause and resume training, storing the complete training history, such as epoch loss, weights at every *n* epoch, how long the training has been going on, evaluate them and in future optimize hyperparameters for desired models. In addition to that, as models may use different datasets, datasets can be stored and loaded as well.

By providing customizable framework, pyMMM significantly simplify managing multiple models, and integrates seamlessly with the other ML packages like PyTorch. It should be mentioned, that currently pyMMM is in the early development phase, and has not been released yet.

**Keywords:** Open Source, Python, Machine learning, Model manager

## 1 Statement of need

Solving new problems with ML, often requires development of several different ML models, before the optimal one is designed. When working on projects, where multiple models are designed, one must define project structure well, otherwise this can lead to scattered notebooks, no clear naming conventions (filenames like “model1.ipynb”, “test_model_3.ipynb”, or “final_model.ipynb” offer no clarity) and no directory organizations (data, models, scripts and results are often mixed together). Reproducibility issues can also occur, because of lack of version control for data or models, additionally when coding, cells in notebooks might be run out of order, which can produce misleading results. Tracking of model status is also a good practice, to have clear indication if the model was fully trained or not and having automatic checkpointing for systematic saving and loading of model parameters. When programming often paths to files are hardcoded and in case, when organizing folders, if some files are moved, the whole ML pipeline can fail. Experiment tracking, like evaluating and comparing different models can also be challenging, since there is no centralized logging to compare metrics like

*Corresponding author, Email address: janko.slavic@fs.uni-lj.si

accuracy or loss. Most of the time, metrics are collected and copied manually into a spreadsheet or left in markdown cells. Additionally hyperparameters for different models are often scattered across the whole notebook, changed manually and not tracked. These are only couple of bad examples what can quickly happen when working on projects requiring multiple ML models.

The Python Multi Model Manager (pyMMM) addresses this challenges by providing an open-source Python framework for managing multiple models and storing information about the whole ML pipeline, including model parameters, training steps, evaluation metrics, etc. Currently everything mentioned is supported only for PyTorch [1] models, but it is planned to also support other popular ML packages like Tensorflow.

# 2 Overview of components

PyMMM framework for managing multiple models is designed modularly, with two core classes - `Model` and `MultiModelManager`.

## 2.1 Model

The `Model` class is designed to encapsulate all relevant components of a single model pipeline, including model weights, training and test datasets, optimizer, loss function, save interval and other related elements. In addition, it provides custom implementations of the training and testing loops tailored for standard machine learning tasks (one input-output pair in the model). For scenarios requiring specialized training or testing procedures, the `train` and `test` methods should be overridden accordingly.

## 2.2 MultiModelManager

The `MultiModelManager` class, when first initialized, creates database. The `sqlite3` module from the Python Standard Library [2] is used to interface with a lightweight SQL database [3]. The database is used to store model information, including training history with epoch loss.

Methods, such as `add_model` and `remove_model` enables simple model manipulations. When calling the method `start_training`, all the models in the model manager that have not been trained yet, begin to train in succession. If for some reason the training is paused, it is possible to resume it the next time from the last checkpoint. All the historical information required for continuation of training is stored in one folder and loaded automatically for simpler use.

# 3 Demonstration of PyTorch and pyMMM workflow

Basic workflow using PyTorch an pyMMM is presented on Fig. 1. Several models can be defined, together with their dataset, model parameters (weights), hyperparameters (e.g. optimization, model architecture, training, loss function, etc.), and description. This is then packed into `Model` class and connected with `MultiModelManager` via `add_model` methods.

pyMMM then enables us to efficiently train and evaluate different models and store results in database.

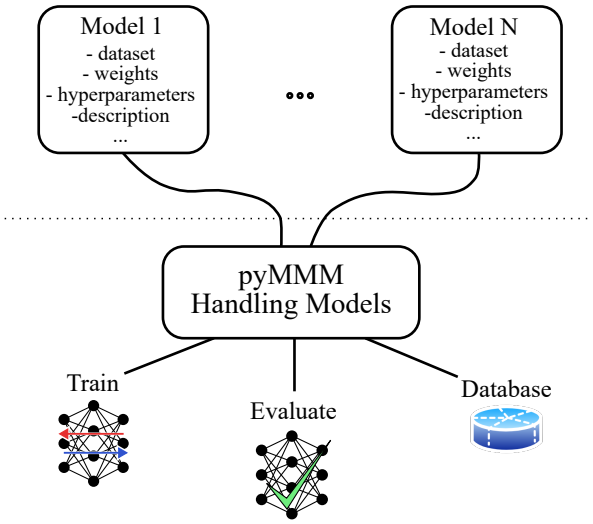



**Fig. 1:** Simple workflow when working with PyTorch models and pyMMM.

## 4 Conclusion

pyMMM is a Python framework, for dealing with multiple models in ML projects, keeping track of learning progress and model results. The idea for a package came due to the needs described earlier and a lack of open-source solution. In the future, our goal is to polish the package; create user interface and logger, write complete documentation, and provide several examples of how to use the package. In addition to that, automatic testing must be implemented for easier contributions by others, before releasing the package to PyPI.

As an open-source framework, pyMMM is aiming for ML beginners, working on project requiring work and comparison between multiple models.

## References


[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[2] P. S. Foundation, *Python*, 2024. Version 3.10.

[3] SQLite Consortium, "Sqlite." `https://sqlite.org`.

# MATLAB Functions for the Reconstruction of Vibration Fields from Event Camera Acquisitions

Sofia Baldini[1]*, Riccardo Bernardini[1], Andrea Fusiello[1], Paolo Gardonio[1], Roberto Rinaldo[1]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine, (IT)*

**Abstract**

This paper is focussed on a new approach for the measurement with event-cameras of flexural vibrations of distributed structures. The working principle of these cameras differs significantly from that of classical frame-based cameras. In fact, rather than a sequence of coherent images, here, each pixel of the optical sensor generates a binary information every time the light rises or falls by a fixed step. Thus, overall, the optical sensor produces an asynchronous flow of information, which is called flow of events. Indeed, each pixel delivers a very simple, low size, binary flow of digital information, plus/minus delta, with very small time-latency. Overall, these cameras offer a very appealing solution for the recording of very fast processes such as the vibration of flexible structures, which, particularly for vibro-acoustic measurements, cover a wide range of frequencies that normally exceed a few kHz. To start with, the paper briefly recalls the working principle of these cameras. Then it provides a short overview of the signal and image processing techniques developed to transform offline the streams of events coming from the photosensor into a sequency of classical images with a very high frame rate. Finally, it briefly recalls the implementation of 3D-Point Tracking and triangulation techniques developed to reconstruct the flexural vibration of a cantilever beam from measurements taken with two event cameras. The paper provides insights on both the background theory and the MatLab codes developed for the image reconstruction from the flow of events as well as for the 3D-Point Tracking and triangulation techniques implemented to reconstruct the flexural vibrations of the beam.

**Keywords:** event cameras, image reconstruction from events, 3D-point-tracking, multi-view triangulation

## 1 Introduction / Statement of Need

Among the latest advancements in optical measurement systems, event cameras [1–4] have attracted growing interest due to their unique working principle. Unlike traditional frame-based cameras, which capture images at fixed intervals, event cameras operate asynchronously, recording only changes in brightness at the pixel level. This leads to significantly reduced data redundancy, lower latency (on the order of microseconds), and the ability to capture high-speed phenomena beyond the capabilities of standard high-speed cameras [5]. Additionally, event cameras enable the inspection of a wide range of vibration frequencies, from a few Hz to several kHz, making them particularly suitable for vibro-acoustic applications that demand precise high-frequency measurements.

A critical component in event-based vibration measurement is 3D triangulation, a well-established technique in photogrammetry and computer vision. By capturing images from multiple viewpoints, triangulation allows for the reconstruction of three-dimensional motion data. This

principle has been extensively used in traditional stereo vision systems [6–8] and structured light scanning but is now being adapted for event-based imaging [9,10]. The process involves tracking discrete markers placed on the vibrating structure and computing their 3D coordinates by solving a set of geometric constraints [11–14]. Advances in bundle adjustment and direct linear transform (DLT) algorithms have significantly improved the accuracy and robustness of this approach in dynamic scenarios.

This paper presents an open-source framework for event-camera-based vibration measurement, leveraging state-of-the-art computational tools to process event streams and reconstruct vibration fields. The proposed methodology involves:

- Event Data Processing: converting raw event streams into intensity images using open-source tools.
- Camera Calibration: implementing intrinsic and extrinsic calibration using established photogrammetry methods.
- Marker Tracking and 3D Triangulation: using open-source algorithms for point tracking and vibration reconstruction.
- Vibration Analysis: applying modal analysis techniques on a cantilever beam to quantify vibration characteristics.

The implementation relies on open-source Python libraries, MATLAB-based routines [15], and pre-trained neural network E2VID [16] ensuring reproducibility and accessibility for the research community. Additionally, it compares the event-camera-based results with those obtained using a high-precision laser vibrometer to assess accuracy and reliability. By providing an open-source approach to high-speed vibration measurement, this work aims to foster collaboration and innovation in the field of structural dynamics and optical metrology.

## 2 Open-Source Tools for Event-Based Vibration Measurement

Intensity images are reconstructed from the raw event stream using E2VID, a pre-trained neural network designed for event-based vision processing. E2VID is a recurrent, fully convolutional model inspired by UNet [17], consisting of multiple encoder and decoder layers connected by skip links. The architecture includes depth-wise convolutions, ConvLSTM [18] units for temporal dependencies, and bilinear up-sampling techniques to refine the final intensity image. Batch normalization and ReLU activation are applied throughout the network to ensure stability during reconstruction. E2VID can operate in two distinct modes: constant frame rate and variable frame rate. In the constant frame rate mode, intensity images are generated at predefined intervals, ensuring frame synchronization, this mode is particularly suitable for multi-camera setups and triangulation applications. In the variable frame rate mode, frames are generated dynamically based on event density, optimizing data usage for scenarios with sporadic motion. Selecting an appropriate frame rate is crucial, as a low frame rate may result in an excessive accumulation of events, while a high frame rate could lead to insufficient event data, causing loss of detail.

The Computer Vision Toolkit for MATLAB [11,15] (it runs on Octave too) provides a comprehensive set of tools for 3D reconstruction, camera calibration, and multi-view geometry and it does not have any external dependency on toolboxes or packages. It is particularly useful for refining extrinsic and intrinsic calibration, bundle adjustment, and triangulation processes necessary

for obtaining structure displacements due to vibrations. The choice of these open-source tools over proprietary software is motivated by several key factors:

1. Flexibility and Customization: Unlike commercial solutions, open-source tools allow researchers to modify algorithms, optimize performance, and tailor processing pipelines to specific experimental needs.
2. Community Support and Continuous Development: Open-source projects benefit from active developer communities that contribute to ongoing improvements, bug fixes, and feature enhancements.

Leveraging open-source tools guarantees that the methodologies presented in this study are fully reproducible and can be easily adapted or extended by other researchers. The following sections will detail the implementation of these tools in the context of event-based vibration measurement, covering aspects such as data acquisition, marker tracking, and 3D triangulation.

# 3 Event Stream Processing, Camera Calibration, and 3D Point Tracking

For this study, two event cameras of the DVXplorer model by Inivation were used (documentation available at: https://docs.inivation.com/hardware/current-products/dvxplorer.html). The spatial resolution of these cameras is 640 × 480 pixels, which is relatively low compared to conventional sensors. However, this limitation is primarily due to the data handling approach of the sensor itself and the required electronics, which currently make it challenging and costly to miniaturize while increasing the number of available pixels.

Processing the raw stream of events into usable data involves multiple steps, starting with event accumulation to create meaningful intensity images. Since event cameras do not capture frames in a traditional sense but they work as shown in Figure 1, events are accumulated over short time windows to reconstruct an approximation of the scene. The selection of an appropriate accumulation time is crucial, as it affects the resolution and accuracy of motion detection. It is necessary to comply with the Nyquist-Shannon sampling theorem to prevent aliasing while also ensuring that a sufficient number of events are accumulated so that the reconstructed image can be processed effectively using triangulation algorithms designed for frame-based images.

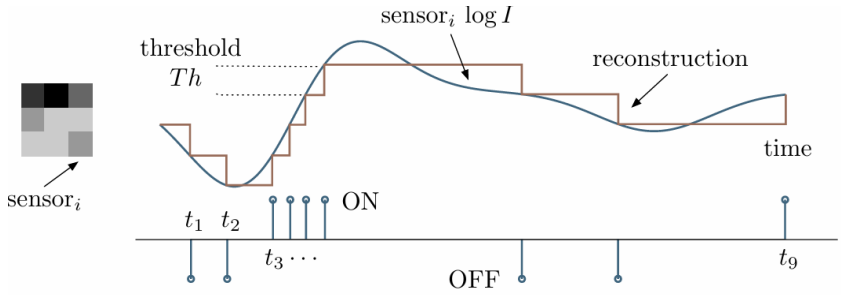



Fig. 1: Operation principle of an event camera.

For accurate camera calibration and 3D tracking, intrinsic and extrinsic parameters must be determined using standard photogrammetry techniques [19,20]. Calibration is performed using checkerboard patterns and optimized through bundle adjustment algorithms from the Computer Vision Toolkit for MATLAB [15]. This ensures minimal distortion in the reconstructed images and

precise alignment between multiple cameras. To obtain the calibration images, an example of which is shown in Figure 2a along with the different positions of the event camera for calibration in Figure 2b, the event stream was processed through the E2VID neural network. For the grayscale image reconstruction during the experiment recording, a timestamp of 1000 Hz was selected. While not necessary for accurate signal reconstruction, this choice was made to explore the limits of temporal resolution starting from the raw event camera data, providing an insight into the potential frequency resolution achievable during modal analysis of structures using event cameras.

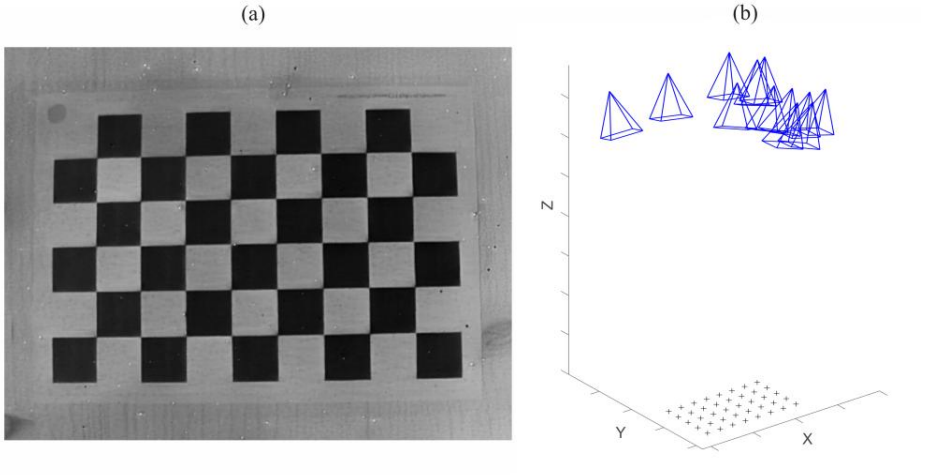



Fig. 2: calibration image reconstructed by E2VID(a), positions of the camera to perform calibration (b).

To validate these open-source techniques in a relevant vibration study case, recordings were performed on a cantilever beam made of steel having length $L = 210$ mm, width $h = 30$ mm, thickness $s = 2$ mm. On the surface of the cantilever beam, a regular grid of 9 ×3 markers was applied in order to perform the marker tracking procedure. Following calibration, the recordings on the beam were performed and after the frame reconstruction procedure, marker detection and tracking was implemented. In general, the perspective image of a circle appears as an ellipse. To simplify the dot detection task, a rectification process is applied to eliminate the perspective distortions in the images captured by the cameras. To determine the homography H that rectifies each of the K images, the user must manually select four points (in this study the centre positions of the dots at the four corners of the grid were used) with known coordinates in the first frame of the video sequence.

In the rectified image, the centres of the dots were identified through template matching, which involved locating the maxima of the normalized cross-correlation between the image and a small patch. It should be noted that some outliers may arise from clutter in the images, often appearing as bright spots. To address this issue, an assignment problem was formulated, defined as follows: Given two sets *A* and *B* of equal size and a function *C(a,b)* that measures the cost of matching *a* with *b*, the goal is to find the bijection φ that minimizes the cost function: $\sum_{a \in A} C(a, \phi(a))$. The cost is the Euclidean distance. Set *A* contains the nominal positions of the dots, whereas set *B* contains the coordinates of the dots detected in the images.

To improve precision, subpixel refinement of the dot centres was performed by fitting a parabola to the point of maximum correlation and its two neighbouring points along the axes, with the vertex providing the refined position. Finally, the inverse of H was applied to transform the coordinates back to the original image space. These tracked points are then triangulated using the Direct Linear Transform (DLT) algorithm, allowing for accurate 3D reconstruction of the marker positions $M_j(t) = [X_j(t), Y_j(t), Z_j(t)]$ over time. The bundle adjustment refinement step further minimizes errors by iteratively optimizing both camera parameters and 3D point positions. This combination of event stream processing, calibration, and tracking enables highly precise vibration measurement with minimal data redundancy, making it ideal for high-speed applications. The next section will discuss the application of these techniques to a real-world case showing the main results and comparing them against traditional laser-based measurements.

# 4 Event Stream Processing, Camera Calibration

Vibration measurements generally span a broad range of frequencies, typically from 2 to 5 kHz for standard engineering applications. Therefore, to ensure the accuracy of the proposed method for reconstructing the vibrating field from camera data, it is essential to utilize cameras with a high frame rate. The beam was excited by a shaker at 35 Hz, corresponding to the first flexural natural mode. Subsequently, grayscale images were reconstructed using the E2VID neural network. An example of a reconstructed frame is shown in Figure 3b. The reconstruction timestamp was set to 1000 Hz, and care was taken to ensure synchronous measurements between the two cameras. Once the frames were obtained, the marker detection algorithm was applied to visualize the trajectory followed by each marker over time.

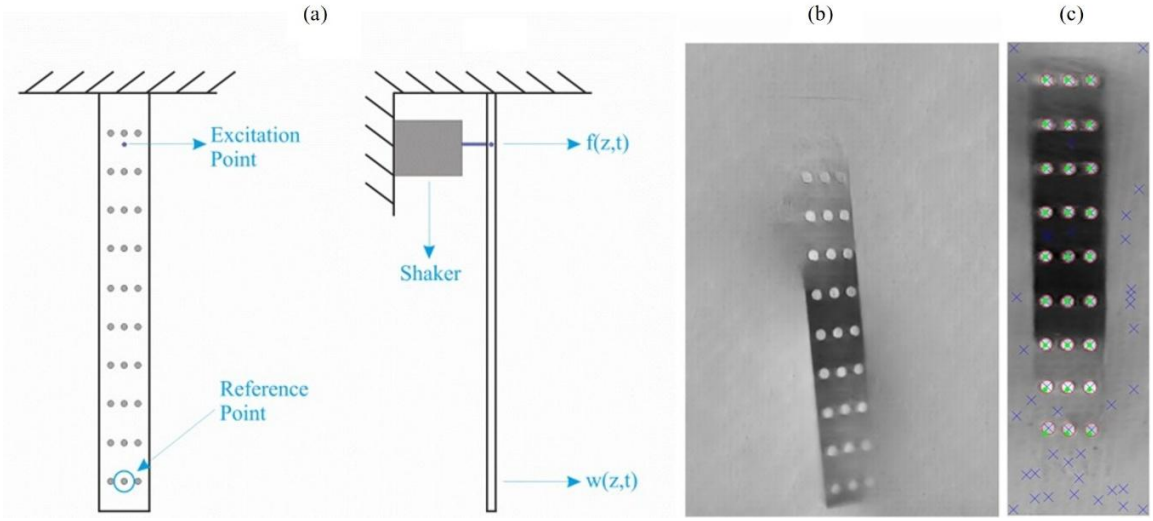



Fig. 3: (a) cantilever beam model problem (b) frame reconstructed by E2VID (c) marker detection, red circles represent chosen marker by the algorithm.

After computing the cost function that ensures minimum Euclidean distance between the detected marker positions and the real ones, the result consist of a set of coordinates in time in the reference

system of the camera. An example of the correctly detected markers after solving the cost equation is shown in Figure 3c (red circles).

For a cantilever beam exhibiting small-amplitude flexural vibrations, the transverse displacement at the *j-th* grid point can be estimated from its *Z(t)* coordinate, obtained through the triangulation process. Once the flexural displacements *w(x,y,t)* are computed at each marker position *(x,y,z)* using camera measurements for a given time-harmonic excitation with circular frequency $\omega$, the time-harmonic transverse displacement of the *j-th* marker point can be expressed as follows:

$$w_j(t_k) \cong W_j e^{i\omega t_k}$$

where *Wj* represents the complex amplitude of the displacement and $t_k$ is the *k-th* time sample. This formulation allows for the analysis of the vibrational response of the beam under harmonic excitation, facilitating further spectral and modal analysis. The derived displacement data are then validated in accuracy against traditional reference techniques, such as the laser vibrometer.

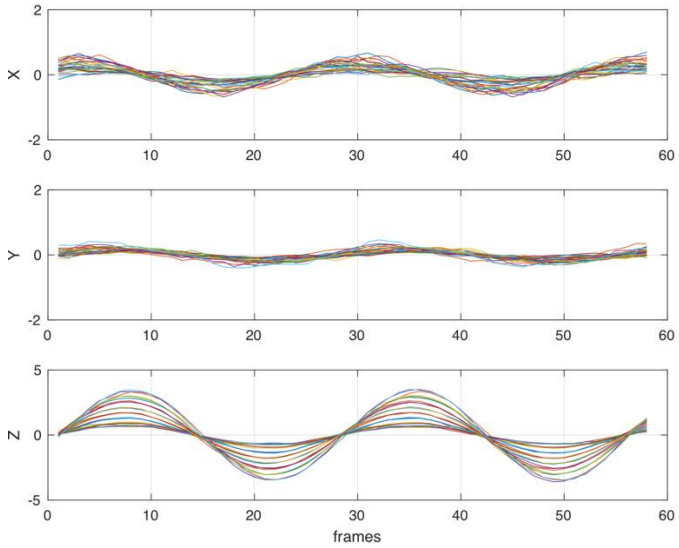



Fig. 4: 3D trajectory for the tracked points in X, Y and Z direction respectively.

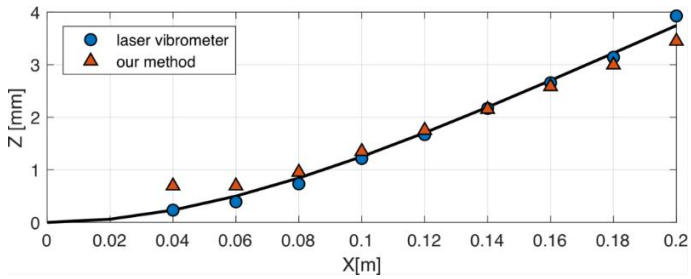



Fig. 5: Comparison of the average transverse displacement measured with event cameras (orange triangles) and with a laser vibrometer (blue dots). The black line represent the theoretical deflection shape of the cantilever beam.

After reconstructing the trajectories of all 27 markers within the frame sets of each camera, the triangulation procedure was performed. Through the use of bundle adjustment, the RMS reprojection error was reduced from an initial value of 1.03 pixels to a final value of 0.133 pixels. The bundle adjustment process yields 3D trajectories for the tracked points. The *X*, *Y*, and *Z* displacements of these points are illustrated in Figure 4. Their motion follows sinusoidal patterns with identical frequency and phase but varying amplitudes. The *Z* component corresponds to the transverse displacement, representing the flexural vibrations of the beam. Meanwhile, the *X* and *Y* components indicate in-plane displacements, which are significantly smaller in magnitude but still exhibit a periodic time-history. Additionally, the experimental setup includes a Polytec PSV-500-A laser Doppler vibrometer (LDV), which serves as a reference instrument for measuring the transverse vibration field of the cantilever beam. The transverse displacements of the 27 grid points are quantified as the average of the extreme *Z* values recorded over each vibration period (in this study, two periods were considered). The residual RMS error compared to the LDV measurements was found to be 0.27 mm. However, when excluding the first row of point, those nearest to the clamped end, the error decreased to 0.23 mm. In this region, the transverse displacement remains minimal, below 0.5 mm, resulting in fewer detected events. Consequently, the reconstructed intensity image appears dimmer, leading to a reduction in tracking accuracy. Figure 5 presents the average transverse displacements measured with our method compared to those obtained using the LDV. The figure also illustrates the theoretical deflection shape, which has been fitted to the LDV measurements. This theoretical shape is determined based on the boundary conditions and the physical properties of the beam, including its material characteristics, geometry, and support conditions.

# 5 Conclusion

This study presented an open-source framework for high-speed vibration measurement using event cameras, demonstrating their potential for structural dynamics and optical metrology. Unlike conventional frame-based cameras, event cameras operate asynchronously, capturing changes in brightness at the pixel level, significantly reducing data redundancy while enabling the recording of high-speed phenomena with minimal latency. Experimental validation was performed using a steel cantilever beam subjected to harmonic excitation. The reconstructed displacement fields were compared against high-precision laser Doppler vibrometer (LDV) measurements, revealing a residual RMS error of 0.27 mm, which decreased to 0.23 mm when excluding the first row of markers near the clamped end. This discrepancy was attributed to the lower event density in regions of minimal displacement, affecting the accuracy of the reconstructed intensity images. Additionally, the application of bundle adjustment reduced the RMS reprojection error from 1.03 pixels to 0.133 pixels, highlighting the effectiveness of the optimization process in enhancing tracking accuracy. The results confirm that event cameras, when combined with appropriate processing algorithms, offer a viable alternative for high-speed vibration analysis, particularly in applications requiring minimal data redundancy and high temporal resolution. The open-source nature of the implemented tools ensures reproducibility and adaptability, fostering further research and development in event-based optical measurement techniques. Future research could aim to refine the methodology by directly processing the event stream without the intermediate step of generating grayscale images via a neural network.

## Acknowledgments

The authors would like to thank the ESPERT project of the University of Udine for the fundings provided.

## References

[1] Benosman, R., *et al.*, 2014, "Event-Based Visual Flow," IEEE Trans. Neural Netw. Learn. Syst., 25(2), pp. 407–417.

[2] Gallego, G., *et al*., 2022, "Event-Based Vision: A Survey," IEEE Trans Pattern Anal Mach Intell, 44(1), pp. 154–180. https://doi.org/10.1109/TPAMI.2020.3008413.

[3] Gallego, G., *et al*., 2018, "Event-Based, 6-DOF Camera Tracking from Photometric Depth Maps," IEEE Trans. Pattern Anal. Mach. Intell., 40(10), pp. 2402–2412.

[4] iniVation, A. G., 2020, "Understanding the Performance of Neuromorphic Event-Based Vision," Tech. Rep.

[5] Amir, A., *et al*., 2017, "A Low Power, Fully Event-Based Gesture Recognition System," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, pp. 7388–7397. https://doi.org/10.1109/CVPR.2017.781.

[6] Baldini, S., *et al*., 2025, "3D Sound Radiation Reconstruction from Camera Measurements," Mech Syst Signal Process, 227. https://doi.org/10.1016/j.ymssp.2025.112400.

[7] Gorjup, D., *et al*., 2019, "Frequency Domain Triangulation for Full-Field 3D-Deflection-Shape Identification," Mech. Syst. Signal Pr., 133 106287, pp. 143–152.

[8] Gardonio, P., *et al*., 2022, "Reconstruction of the Sound Radiation Field from plate Flexural Vibration Measurements Taken with multiple Cameras," *Proceedings of ISMA 2022*, Katholieke Universiteit Leuven (B).

[9] Sofia Baldini, *et al*, 2024, "Vibration Measurement with Event Cameras," *Proceedings of ISMA 2024*, Katholieke Universiteit Leuven, Leuven.

[10] Sofia Baldini, *et al*, 2024, "Measuring Vibrations with Event Cameras," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Brescia. https://doi.org/https://doi.org/10.5194/isprs-archives-XLVIII-2-W7-2024-9-2024.

[11] Fusiello, A., 2024, *Computer Vision: Three-Dimensional Reconstruction Techniques*, Springer Cham.

[12] Faugeras, O., 1993, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, The MIT Press, Cambridge, MA.

[13] Hartley, R., and Zisserman, A., 2003, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge.

[14] Trucco, E., and Verri, A., 1998, *Introductory Techniques for 3-D Computer Vision*, Prentice-Hall.

[15] Fusiello, A., 2022, "Computer Vision Toolkit for Matlab."

[16] Rebecq, H., *et al*., 2019, "High Speed and High Dynamic Range Video with an Event Camera," IEEE Trans Pattern Anal Mach Intell, 43, pp. 1964–1980. [Online]. Available: https://api.semanticscholar.org/CorpusID:189998802.

[17] Ronneberger, O., *et al*., 2015, "U-Net: Convolutional Networks for Biomedical Image Segmentation," pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28.

[18] Xingjian Shi, *et al.*, 2015, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *Advances in Neural Information Processing Systems*.

[19] Sturm, P. F., and Maybank, S. J., 1999, “On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications,” *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, pp. 432–437. https://doi.org/10.1109/CVPR.1999.786974.

[20] Zhang, Z., 2000, “A Flexible New Technique for Camera Calibration,” IEEE Trans Pattern Anal Mach Intell, 22(11), pp. 1330–1334. https://doi.org/10.1109/34.888718.

# A Population-Based SHM Framework

D.S. Brennan, T. Gowdridge, J. McCulloch, J. Poole, T.R. Windus-Smith, T.J. Rogers, K. Worden

*Dynamics Research Group, School of Mechanical, Aerospace, and Civil Engineering, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK*

**Abstract**

Structural Health Monitoring (SHM) is an engineering discipline centred around determining the integrity of a structure; ultimately to establish a prognosis of remaining useful lifespan of said structure. Population-based Structural Health Monitoring (PBSHM) is an emerging technique within the SHM field, that aspires to enhance the available knowledge of a given structure by transferring knowledge from established similar structures, thus improving the effectiveness of SHM.

While PBSHM provides a clear advantage in understanding the health of structures, it inevitably comes with its own set of technical problems and hurdles. Published work to date has already introduced the three shared-data domains for PBSHM: *network*, *framework*, and *database*. While the existing publications have outlined the initial database choice and software stack, this paper focusses on the additional technical work undertaken to achieve the current version of the PBSHM software: the *PBSHM Framework*.

This paper outlines the architecture of the PBSHM Framework and details the technical decisions made to overcome the problems faced when architecting a framework built to cope with varying data sources, accuracy, and security requirements. The open-source nature of the framework is also highlighted, showcasing how third-party modules can be developed to extend its core functionality. Additionally, the paper explores database issues such as data uniqueness, and the traceability from raw data, through to features, and finally transferred data. Ensuring data traceability within the framework not only improves error detection but also allows for result caching, significantly reducing unnecessary computational overheads.

**Keywords:** PBSHM, Framework, Database, Security, Module Development

## 1 Introduction

The goal of Structural Health Monitoring (SHM) [1] is to determine if a structure—or system—is damaged or undamaged; this can be achieved via a range of methods, most commonly, via the processing of damage-sensitive features extracted from the structure. While this approach is 'tried and tested', it often suffers from a lack of sufficient data to reliably determine whether these features indicate damage or not. Population-based Structural Health Monitoring (PBSHM) [2, 3, 4, 5, 6] is a recent derivative of SHM which aims to overcome this data limitation by sharing data between structures which exhibit some degree of similarity. The idea is that by pooling data from related structures, it becomes possible to gain richer insights into each individual structure than would be achievable with data from a single structure alone.

Despite its potential advantages, PBSHM introduces new challenges that need to be addressed. Aside from building the algorithms required to establish a base similarity between structures—the

*population*—and then transfer existing knowledge between the established structures, there is a technical shared domain problem which, if not resolved, will hamper the progress of PBSHM. To solve this problem, Brennan et al. [6] introduced three shared-data domains for PBSHM data: *network* where the similarities between structures are established, *framework* where the computational and algorithmic logic occurs, and *database* where data is stored.

If handled poorly, PBSHM could actually cause more harm than good, via a process known as *negative transfer*, where the transfer of knowledge results in worse SHM predictions than if the structure were considered in isolation. Selecting genuinely similar structures remains challenging and mistakes in this process could have serious consequences. The framework, and its associated storage mechanisms and logic aims to perform the most optimal analyses.

The Framework provides secure encapsulation of data, modules, and security measures, offering native support for population-based analysis. The Framework comes with integrated software that forms the backbone of population-based analyses, aiming to mitigate the issues surrounding data scarcity, and algorithms which select appropriate structures to undergo knowledge transfer. In doing so, The Framework's guiding principle is to ensure a fully deterministic and traceable process from raw data, through to feature extraction and all inferences made via transfer learning.

The remainder of this paper provides essential background and highlights related work previously published on the Framework. The paper then outlines design decisions related to data storage, followed by an overview of the software layout and the module logic applied to the stored data, including the prebuilt modules that ship with the framework. Subsequent sections address data security considerations and the continuous integration pipelines used to automate development processes.

## 2 Background

The term 'Framework' is not new; in computer science, it refers to the encapsulation of data and the software that operates on it. Within this context, the ROSEHIPS Framework has two main components: storage and logic. The Framework uses the NoSQL database MongoDB to store data, and uses Python to implement its logic. The Framework is online and makes extensive use of common third-party Python packages, most notably Flask, a lightweight web-development framework [7]. The Framework's graphical user interface (GUI) is built using a more traditional web development software stack: HTML, JavaScript, and CSS, with Jinja2 [8] employed for templating.

Many of the topics reviewed in this summary paper have been explored in greater detail on an individual basis [6, 9, 10, 11, 12, 13]. Part V in the 'Foundations of PBSHM' [6] series outlines the initial *PBSHM Schema* to regulate and standardise Channel and IE-model data within the *database* domain. Further work by Gowdridge et al. [9] and McCulloch et al. [10] introduced additional elements to explore and process this data.

However, this is the first paper that brings all of these elements together into a cohesive 'PBSHM Framework' and provides an overview of how they all function together. As such, this paper's focus is on describing the framework as a unified software platform, and on detailing features not previously covered in earlier publications—such as data siloing, the module builder, and the use of differential privacy within the framework.

The requirement for a Framework in PBSHM is at the core of PBSHM, as standardisation among families of structures is essential. For example, the MongoDB database follows a defined schema, which outlines the exact structure of a document within a collection. Ensuring data conformity across

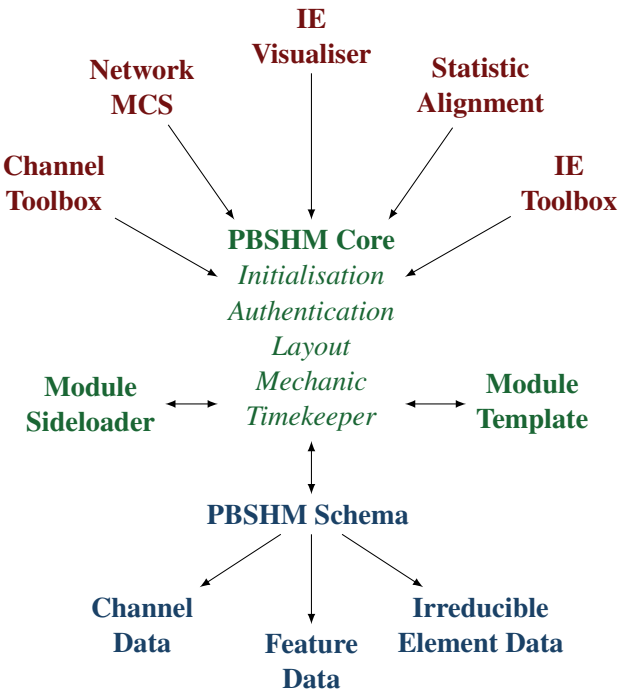



**Fig. 1:** The hierarchical structure of the PBSHM Framework; PBSHM Modules interacting with the PBSHM Core and subsequently the PBSHM Core then interacting with the PBSHM Database to fulfill the requests from the PBSHM Modules.

all structures is the first step in enabling information to be leveraged from one structure to another, supporting information sharing. Another important element that the Framework relies on to support PBSHM is the use of IE models. These models encode topological connections of structures with attributes and can be assessed by graph-matching algorithms [14] to identify structural similarities.

Fig. 1 provides a simplified overview of key components in the Framework. Elements shown in red are modules, containing the logic to process the data. Green elements are members of the *framework domain*, which handle security, usability, GUI and related features. In blue are elements of the *database domain*, showing the types of data stored in the framework and the form of each data entry.

## 3 Database Domain

Supporting population-based analysis in Structural Health Monitoring needs well-defined and standardised mechanisms for the managment and storage of raw sensor data across varying formats and sources. The PBSHM Framework addresses this need by the use of JSON schemas, which validate all data entries. These schemas define the desired structure for a range of SHM data types, allowing for consistent interpretation and processing. There are currently three knowledge areas within the *database* domain, each with their own associated schema: *channel*s, *feature*s, and *irreducible element model*s. Each knowledge area is applicable within one or more of the three

outlined database domains: *time*, *frequency*, and *system* [9].

At the most fundamental level, each document stored within the database must satisfy the following requirements (as outlined in the PBSHM Schema) to be a valid PBSHM document, regardless of which knowledge area is being stored:

`version`: the version of the PBSHM Schema against which the document is compliant.

`population`: a unique name for the population in the database that represents the structure.

`name`: a unique name for the structure within the population.

`timestamp`: (*only applicable for time domain knowledge areas*) data must be recorded on a global timescale with high accuracy, down to nanosecond granularity, to support the analysis of acoustic measurements. All timestamps are therefore measured as the number of nanoseconds since epoch time [15].

`source`: (*only applicable for system domain knowledge areas*) provides the unique and identifiable source allowing the deterministic and reproducible generation of the data.

## 3.1 Channels

Raw sensor data within the framework are structured according to the Channel Schema, which formalises how time-series data from individual sensors are stored. For clarification, data taken from sensors may be converted in base units and still be within the *channel*s' knowledge area; however, any processing of the data requires the subsequent data to be stored within the *feature*s' knowledge area.

## 3.2 Features

The storage of processed data poses greater challenges in maintaining traceability and uniqueness, as every action performed on the data must be tracked. Indexing processed data also becomes particularly tricky when converting from the time domain to the frequency domain, because of the many-to-one mapping into the frequency domain.

The proposed solution is to store an item known as a *selection object* [9]. This object contains enough detail, including: the selected time range, channels, and the processes applied to the data, which fully describe how the values were derived and is then used to assist indexing the feature data, ensuring both uniqueness and determinism. Data within the *feature*s' knowledge area is governed by the Feature Schema.

## 3.3 Irreducible Element Models

IE models provide an organised method for representing real-world structures using predefined rules, ensuring consistency across different creators. Fundamentally, IE models are graphs that define a structure's topology, along with associated attributes such as profile geometry, material properties, and relationships between elements. These models represent structures as data structures, forming the basis for graph-matching analyses.

Graph matching is used to identify similarities between subsets of structures, which is a useful step in determining where knowledge transfer is viable. The specifics of IE models have been discussed in far greater detail elsewhere. Additional information on the latest version of the IE model langunage, can be found in the IE Model Schema and previous publications [6].

# 4 Framework Domain

The Framework is structured so that its core functionality is handled by the PBSHM Core, which manages initialisation, authentication, GUI layout, and database configuration and connection. The PBSHM Core is a minimal installation designed for users who wish to build their own PBSHM pipelines from scratch.

The PBSHM Framework extends the PBSHM Core, providing prebuilt modules and simulated data from the PEAR database [11]. By creating an ecosystem that ships with built-in module functionality, the framework allows experts in their respective fields to develop and contribute analyses, which can then be distributed as part of the system.

## 4.1 Modules

While the storage of data has been loosely discussed, it is the accompanying logic, delivered through module functionality, that truly defines the framework. These modules extend the stored data, enabling meaningful interaction with it.

The Framework natively includes several modules designed to modify, analyse, or visualise data. However, these modules are not exhaustive; new modules can be developed by experts to introduce custom logic or specialised analyses. Modules can provide any functionality, and the native ones support various aspects relevant to PBSHM. Currently, the Framework includes five native modules: Channel Toolbox, IE Toolbox, Network MCS, IE Visualiser, and Statistical Alignment.

### 4.1.1 `pbshm-channel-toolbox`

The PBSHM channel toolkit is designed to help users interact with data belonging to the channel knowledge area of the PBSHM Schema. The toolkit is broken down into two distinct packages: autostat and cleanse.

The autostat package serves as a quick and easy way to browse channel data stored within the *framework*'s associated database, and provides a simple breakdown of basic statistical information on a dataset. The cleanse package provides the database driven tools to clean an exsisting dataset of any inconsistencies and irregularities across the population and produce a consistent and normalised dataset for use within other tools.

### 4.1.2 `pbshm-ie-toolbox`

The PBSHM IE Toolkit is designed to help users interact and design IE models for use within the *framework*. The toolkit enables IE models to be either created within the online text editor or upload existing PBSHM Schema-compliant JSON files.

Each IE model is placed within a private users sandbox, each model then goes through two steps of verification. Step 1: Validate the syntax of the IE model against the latest version of the PBSHM Schema. Step 2: Validate that the logic of the IE model holds true (e.g. it is if a grounded model, there should be at least one ground element).

Once a model has passed both stages of validation, this model can then be included in the global IE-model catalogue of the system.

### 4.1.3 `pbshm-network-mcs`

The Network Maximum Common Subgraph module uses the Jaccard Index with Maximum Common Subgraph to generate a similarity score between IE models in the *network*. Within this module, one can select any IE models that are loaded into the *framework*'s associated database (given that they meet the specification outlined in PBSHM Schema) and generate a similarity matrix (1 = similar, 0 = dissimilar) for the selected models.

### 4.1.4 `pbshm-ie-visualiser`

The PBSHM IE Visualiser is designed to help users design and visualise IE models for use within the *framework*'s associated database. The visualiser enables IE models to be created in the online 3D model builder and then saved as PBSHM Schema-compliant JSON files in the database. Existing IE models within the database can either be viewed in a read-only online 3D tool or edited in the 3D model builder.

### 4.1.5 `pbshm-statistic-alignment`

The PBSHM Statistic Alignment package [9] demonstrates basic transfer functionality using statistic alignment [16] between a source and a target domain. The idea is that by aligning the statistics of the two domains over specified time intervals, class labels can be transferred from the label-rich source to the information-poor target.

### 4.1.6 Module Template

A Module Template was created using the CookieCutter Python package [17], which allows for the creation of customised repositories. This useful tool streamlines the process of creating modules within the Framework ecosystem, eliminating the need for the user to repeat boilerplate code and manually set up database connections. As a result, developers can focus on the core logic of the module rather than handling non-essential setup tasks.

### 4.1.7 Module Sideloader

The Module Sideloader enables dynamic management of framework modules, allowing users to upload, register, install, and uninstall modules at runtime. The Sideloader is configured using JSON files, which define each module's package source, namespace, import path, and routing. The Module Sideloader easily allows modules to be enabled or disabled, and seemlessly integrates them into the application's GUI.

# 5 Security

The engineering data required for a PBSHM Framework is considered sensitive intellectual property (IP) by asset owners. As a result, convincing well-established companies to participate can be challenging, as the security of their IP must be guaranteed in the face of potential adversarial attacks on the database.

Two types of security measures are implemented: database segregation and mathematical obfuscation. Of the two, mathematical obfuscation [12] is particularly notable, as it enables the sharing of patterns in data between competing companies without revealing their specific details.

## 5.1 Data Silos

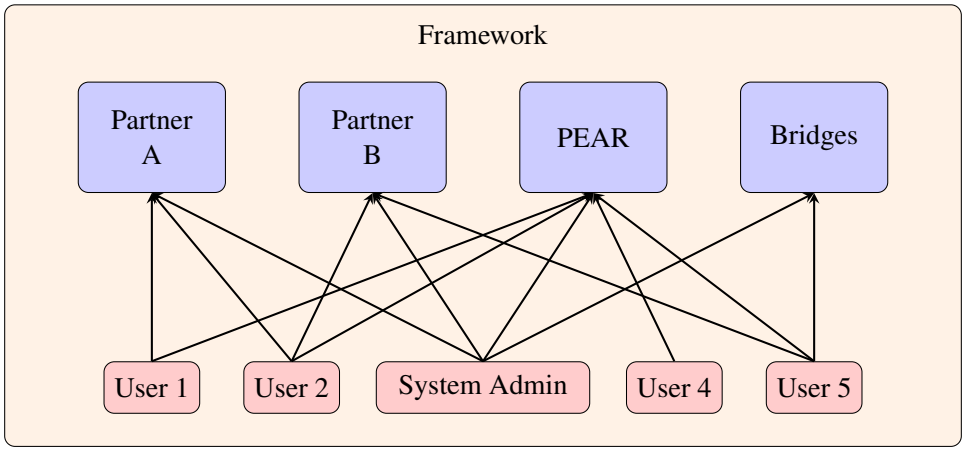



**Fig. 2:** Secure Collection Access in the PBSHM Framework

A problem with any software system that relies upon and encourages expansion of itself through the development of new modules, is how to ensure new additions meet existing practises. This complication is compounded when considering the security of commercially and intellectually sensitive corporate data. Any internally built security practises to ensure data is only accessible to the correct users, would need to be honoured in new modules and would inevitably require vetting of each module before inclusion within the framework. This approach is not only time-consuming, but prone to human error. Instead, a system was decided upon for the *PBSHM Framework* that sits outside the framework, to regulate user data access through the credentials used to access the database. If the database enforces the access, then no matter how error-prone an included module is, data will only be returned to the module that the logged in database user has access to.

Collections in MongoDB act as containers for related documents. Within the PBSHM Framework, collections are used to group structural data according to user access levels, ensuring data isolation so that sensitive company information is not accessible by default. Access is controlled via resource restricted 'Role-Based Access Control' in MongoDB, with each user assigned one or more roles and each role limiting which resources — collections — are permitted access. For example, the PEAR dataset (a dataset curated specifically for PBSHM research) is openly accessible to all roles, while raw data from a specific company is restricted to users with the necessary permissions. An example of this secure data siloing is shown in Fig. 2.

## 5.2 Differential Privacy

From a population-based perspective, basic data siloing provides only a rudimentary level of security, as it prevents any information exchange between private users. A more sophisticated and research-driven approach to privacy in the PBSHM Framework involves mechanisms that enable secure data

sharing while protecting sensitive information from unauthorised access.

Two main strategies are being considered to address this challenge: Federated Learning (FL) and Differential Privacy (DP). FL is a decentralised approach in which edge devices train machine-learning models locally, sharing only learned weights with a central server for aggregation [18]. This outcome eliminates the need to transfer raw data and has shown strong performance even in non-independent and identically-distributed scenarios where data distributions differ across clients [19].

DP, predating FL, provides a more foundational guarantee of privacy [20]. It limits the influence of any single database entry on the result of a query, thereby protecting individual identities. This is achieved by injecting carefully calibrated random noise into query outputs via a sanitisation mechanism. The guarantee of DP is formalised as:

$$P(\mathscr{M}(D) \in \mathscr{R}) \leq \varepsilon \cdot P(\mathscr{M}(D') \in \mathscr{R}) + \delta \tag{1}$$

where $\mathscr{M}$ is the DP mechanism, $D$ and $D'$ are neighbouring databases differing by one entry, $\mathscr{R}$ is a possible query result, and $\varepsilon$, $\delta$ control the privacy guarantees. DP ensures that even with auxiliary knowledge, adversaries cannot infer whether a particular entry is in the database. Moreover, DP can be integrated with FL, where noise is added to model weights during each aggregation step [21], further strengthening privacy protections.

# 6 Continous integration

By the use of GitHub Actions, two continuous integration (CI) pipelines have been established to run automatically upon any modification to the main branch of the repository: one for packaging and another for testing.

The packaging pipeline supports the release process by automating the build and distribution of the framework. As the PBSHM Framework is distributed via the Python Package Index (PyPI), any new release triggers the pipeline to compile the latest version and publish it to PyPI. This process allows users to install the framework using standard Python tooling (`pip install pbshm-framework`) without needing to clone or build the project manually. The automation ensures version consistency and reduces the risk of human error during the release process.

The testing pipeline is critical for maintaining the robustness of both the PBSHM Core and the broader PBSHM Framework. Unit tests are run to validate the behaviour of individual modules, while integration tests ensure that all components of the framework interact as expected. This pipeline provides immediate feedback to developers, helping catch erroneous code or compatibility issues early in the development cycle.

# 7 Conclusion

This paper has introduced a software framework designed to support the implementation of PBSHM. The framework directly addresses technical and operational challenges associated with PBSHM, such as standardising heterogeneous data sources, ensuring traceability of data transformations and analyses, and maintaining privacy and security when handling sensitive information.

The framework's open-source, modular design allows users to extend its functionality through custom-developed modules. This extensibility enables support for diverse applications and encourages collaboration among researchers, developers, and asset owners, incorporating field-specific knowledge into the modules.

Using the PBSHM Framework helps realise the full value of population-based insights, it supports better decision-making by improving data richness via knowledge transfer between structures, and enhances the effectiveness of structural health monitoring across assets.

# 8 Acknowledgements

The authors would like to gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) via grant reference EP/W005816/1. For the purposes of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

## References


[1] C. Farrar and K. Worden, "An introduction to structural health monitoring," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2007.

[2] L. A. Bull, P. A. Gardner, J. Gosliga, T. J. Rogers, N. Dervilis, E. J. Cross, E. Papatheou, A. E. Maguire, C. Campos, and K. Worden, "Foundations of population-based SHM, part I: Homogeneous populations and forms," *Mechanical Systems and Signal Processing*, vol. 148, p. 107141, 2021.

[3] J. Gosliga, P. A. Gardner, L. A. Bull, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part II: Heterogeneous populations–graphs, networks, and communities," *Mechanical Systems and Signal Processing*, vol. 148, p. 107144, 2021.

[4] P. Gardner, L. A. Bull, J. Gosliga, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part III: Heterogeneous populations–mapping and transfer," *Mechanical Systems and Signal Processing*, vol. 149, p. 107142, 2021.

[5] G. Tsialiamanis, C. Mylonas, E. Chatzi, N. Dervilis, D. Wagg, and K. Worden, "Foundations of population-based SHM, part IV: The geometry of spaces of structures and their feature spaces," *Mechanical Systems and Signal Processing*, vol. 157, p. 107692, 2021.

[6] D. S. Brennan, J. Gosliga, E. J. Cross, and K. Worden, "Foundations of population-based SHM, part V: Network, framework and database," *Mechanical Systems and Signal Processing*, vol. 223, p. 111602, 2025.

[7] A. Ronacher, "Flask: A lightweight wsgi web application framework." `https://flask.palletsprojects.com/`, 2010. Version 2.2.5, accessed April 2025.

[8] A. Ronacher, "Jinja2 templating engine," 2008. Accessed April 2025.

[9] T. Gowdridge, J. McCulloch, J. Poole, T. J. Rogers, K. Worden, and D. S. Brennan, "Embedding an initial transfer-learning technology within a population-based SHM framework," *e-Journal of Nondestructive Testing*, vol. 2024, July 2024.

[10] J. McCulloch, T. Gowdridge, K. Worden, and D. S. Brennan, "An implementation of a generalised irreducible element model builder for use within a population-based framework," *e-Journal of Nondestructive Testing*, vol. 2024, July 2024.

[11] C. O'Higgins, T. Gowdridge, D. Hester, K. Worden, and D. S. Brennan, "Population-based structural health monitoring engineered asset resource (PEAR): A benchmark PBSHM dataset," *Structural Health Monitoring*, 2025.

[12] T. Windus-Smith, K. Worden, and T. J. Rogers, "On conserving privacy in structural health monitoring," *e-Journal of Nondestructive Testing*, vol. 29, 07 2024.

[13] D. S. Brennan, C. T. Wickramarachchi, E. J. Cross, and K. Worden, "Implementation of an organic database structure for population-based structural health monitoring," in *Dynamics of Civil Structures, Volume 2*, pp. 23–41, Springer International Publishing, 2022.

[14] D. Brennan, T. Rogers, E. Cross, and K. Worden, "On calculating structural similarity metrics in population-based structural health monitoring," *Data-Centric Engineering*, vol. 6, p. e24, 2025.

[15] International Organization for Standardization, "ISO 8601 - Date and Time Format," 2017.

[16] J. Poole, P. Gardner, N. Dervilis, L. Bull, and K. Worden, "On statistic alignment for domain adaptation in structural health monitoring," *Structural Health Monitoring*, vol. 22, no. 3, pp. 1581–1600, 2023.

[17] A. R. Greenfeld, D. R. Greenfeld, and R. Pierzina, "cookiecutter," Feb. 2024.

[18] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[19] H. Nguyen, P. Wu, and J. M. Chang, "Federated learning for distribution skewed data using sample weights," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 6, pp. 2615–2626, 2023.

[20] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[21] M. Moshawrab, M. Adda, A. Bouzouane, H. Ibrahim, and A. Raad, "Reviewing federated learning aggregation algorithms; strategies, contributions, limitations and future perspectives," *Electronics*, vol. 12, no. 10, p. 2287, 2023.

# PMD - A Planar Multibody Dynamic Open Source Simulation Software

Giacomo Cangi[1,2] * Massimiliano Palmieri[1] Filippo Cianetti[1]

[1] *University of Perugia, Department of Engineering, Via G. Duranti 93, 06125 Perugia, Italy*
[2] *Terex Italia S.R.L., Via Cassoletta 76, 40053, Valsamoggia (BO), Italy*

**Abstract**

`PMD` (Planar Multibody Dynamics) is an open-source Python library developed to model, simulate, and analyze planar multibody dynamic systems. Using the principles of object-oriented programming, `PMD` provides a modular and intuitive framework that allows users to define bodies, joints, forces, and constraints with ease. The package also supports the integration of user-defined forces and motion constraints through custom functions, allowing an high degree of customization. The library architecture is built around a solver that manages interactions between components and computes the dynamic response of the system. `PMD`'s use of Python, a widely adopted and accessible programming language, ensures seamless integration with scientific libraries like `NumPy` and `SciPy`, enhancing computational performances and enabling advanced numerical analyses. By simplifying the complexities of planar multibody dynamics and offering a flexible and extensible platform, `PMD` empowers users to focus on problem solving without the overhead of proprietary software limitations.

**Keywords:** Multibody, Open Source, Structural Dynamics, Python

## 1 Statement of Need

The use of multibody codes for the analysis and optimization of the mechanical systems is a common practice in both the industrial and academic sectors [1]. These codes enable the simulation of virtually any mechanical system and are particularly useful for tasks such as design, health monitoring, and optimization [2, 3]. In any mechanical system, it is crucial to understand the behavior of individual components and how they interact over time, as these interactions can be highly intricate. The dynamics of such systems are often determined by complex relationships that involve relative motion and joint forces between components [4, 5, 6]. Multibody dynamic models play a key role in this context, as they allow engineers to analyze the behavior of mechanical systems under various operating conditions. By simulating the interactions between interconnected components, these models help predict system responses, optimize performances, and enhance safety. Accurate representation of system dynamics also makes it possible to identify potential issues early and improve equipment reliability. A wide range of commercial software tools are available for performing multibody simulations, such as `Ansys Motion`, `Adams`, `Recurdyn`, as well as packages based on proprietary runtimes [7]. However,

*Corresponding author, Email address: giacomo.cangi@dottorandi.unipg.it

these solutions typically require a license and are not developed in an open-source, collaborative manner. As a result, users are often confined to environments with limited graphical interactions, which can hinder more complex or large-scale analyses [8].
In contrast, the `PMD` open-source software library enables the modeling and simulation of various planar mechanical systems under different operating conditions. This capability allows prediction of system behavior, which supports research efforts and contributes to the development of safe and reliable machines. `PMD` can be accessed through its GitHub repository, which is open to the community. Users are encouraged to access, explore, and contribute to the development of the library. Comprehensive documentation is readily available within the repository to assist users in understanding and using `PMD` effectively. The software is entirely built upon two widely used Python packages: `NumPy` [9], which handles the array operations during simulation, and `SciPy` [10], which performs numerical integration of the system of ODE and DAE generated during the model resolution. Bodies in `PMD` are modeled as rigid by simply specifying their mass, moment of inertia, center-of-mass position and orientation, with respect to the global reference frame. A variety of joints can be defined using a dedicated Python class, enabling the simulation of a wide range of dynamic systems. Since joints and forces must be applied at specific locations, a specialized class is provided to allow users to define all the necessary points for connecting bodies and applying loads. Users can also define custom functions to simulate non-linear behaviors during the simulation process. To use the `PMD` package, users must define all the bodies, points, joints, and forces involved in the system. Once defined, the software automatically handles the creation of the complete model. The `PMD` library supports multiple types of analysis, including static, kinematic, and dynamic simulations, allowing users to explore different aspects of system behavior, depending on the specific application or design need. A simple example of

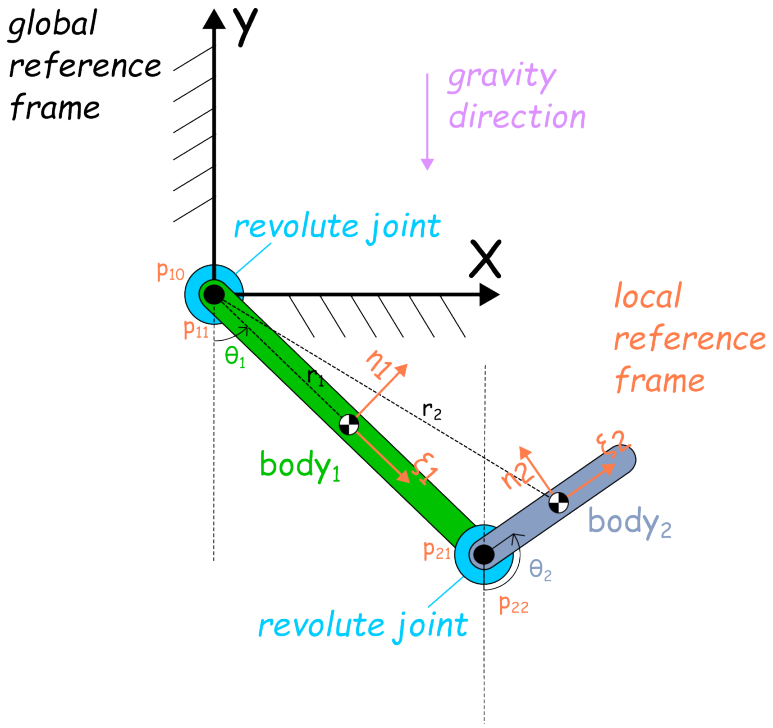



**Fig. 1:** Double compound pendulum schematic.

a system that can be analyzed using the PMD library is illustrated in Fig.1, which shows the schematic representation of a compound double pendulum. To model this mechanical system, it is necessary to define two bodies and two revolute joints, which require a total of four reference points. Subsequently, the external forces acting on the system must be specified; in this simple example, only the gravitational force is considered. The corresponding code, to perform a dynamic simulation, looks as follows:

```
# ... bodies ...
b1 = Body(m=1, J=0.0833, r=[0.0,0.5], p=0.0)
b2 = Body(m=2, J=0.375, r=[0.75,1.0], p=0.0)

# ... points ...
p10 = Point(Bindex=0, sPlocal=np.array([0.0, 0.0]))   # index = 0
p11 = Point(Bindex=1, sPlocal=np.array([0.0, -0.5])) # index = 1
p21 = Point(Bindex=1, sPlocal=np.array([0.0, 0.5]))   # index = 2
p22 = Point(Bindex=2, sPlocal=np.array([-0.75, 0]))   # index = 3

# ... joints ...
# revolute joint between body 1 and the ground
j1 = Joint(type="rev", iPindex=0, jPindex=1)

# revolute joint between body 1 and body 2
j2 = Joint(type="rev", iPindex=2, jPindex=3)

# ... forces ...
# only weight force, acting along -y axis
s3 = Force(type="weight")

# ... double pendulum model creation and simulation...
double_pendulum = PlanarMultibodyModel()
time, solution = double_pendulum.solve(method="LSODA", analysis="
    dynamic")
```

**List. 1:** Code for double pendulum model creation and simulation through PMD library.

In this example, the objective is to solve the system's dynamics, which involves addressing the following mathematical problem:

$$\begin{bmatrix} \mathbf{M} & -\mathbf{D}' \\ \mathbf{D} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{c}} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \mathbf{h}^{(a)} \\ -\dot{\mathbf{D}}\dot{\mathbf{c}} \end{Bmatrix} \tag{1}$$

where the matrix $\mathbf{M}$ represents the mass of the system, including the masses and inertias of the bodies in the multibody system, the matrix $\mathbf{D}$ contains the partial derivatives of the constraint equations with respect to the generalized coordinates, while $\mathbf{D}'$ is the transpose of the constraint matrix. The vector $\ddot{\mathbf{c}}$ represents the generalized accelerations of the bodies in the system, the vector $\lambda$ represents the Lagrange multipliers, which indicate the constraint forces in the system. The vector $\mathbf{h}^{(a)}$ includes the generalized active forces, i.e., the external forces applied to the bodies. Finally, the term $-\dot{\mathbf{D}}\dot{\mathbf{c}}$ represents the constraint forces arising from the velocities of the bodies. The integration method used to solve these equations of motion can be specified directly via the `solve()` method. PMD supports all integration methods available in the `solve_ivp()` function of the SciPy library [10], such as RK45, RK23, Radau, BDF, and others. This allows users to adapt the numerical solution process to the specific

characteristics and stiffness of the model being analyzed. The program adopts the International System of Units (SI), specifically the MKS system, as default. However, users can switch to the MMKS system by specifying it as an input argument when initializing the `PlanarMultibodyModel` class. Angles must be provided in radians by default; nevertheless, users can input angles in degrees by explicitly setting the corresponding attribute. When the simulation starts, the program provides textual output to inform the user about the successful creation and execution of the model. Additionally, the user is prompted to input the simulation end time and time step, if required, directly via the terminal, as this interaction is currently handled through a command-line interface. Once the calculations are completed, the results can be saved for post-processing. Alternatively, the data can be visualized immediately using standard Python plotting libraries such as `matplotlib` [11] or `plotly` [12]. For the model shown in Fig.1, it is possible to extract from the simulation results the x and y coordinates of the center of mass of both bodies, as well as the angle $\psi$ around the z-axis, as illustrated below:

```
# ... double pendulum model creation and simulation...
double_pendulum = PlanarMultibodyModel()
time, solution = double_pendulum.solve(method="LSODA", analysis="
    dynamic")

# ... plot ...
fig, ax = plt.subplots(2, 1, figsize=(10, 12), sharex=True)

ax[0].plot(time, solution[:, 0], label=r'$x_1$', color='b', linestyle=
    '-', linewidth=2)
ax[0].plot(time, solution[:, 1], label=r'$x_2$', color='r', linestyle=
    '-', linewidth=2)
ax[0].set_ylabel('Displacement (m)', fontsize=12)
ax[0].legend()

ax[1].plot(time, solution[:, 3], label=r'$y_1$', color='b', linestyle=
    '-', linewidth=2)
ax[1].plot(time, solution[:, 4], label=r'$y_2$', color='r', linestyle=
    '-', linewidth=2)
ax[1].set_ylabel('Displacement (m)', fontsize=12)
ax[1].legend()

fig.suptitle('System responses', fontsize=14)
plt.show()
```

**List. 2:** Code for visualizing the responses of the double compound pendulum in terms of x and y coordinates of the bodies center of mass, using the matplotlib package.

Since `PMD` is an open-source library coded in Python, it enables seamless integration with other programs without relying on commercial software. Furthermore, the package offers comprehensive documentation and a collection of examples with further usage applications.

## 2 Validation

Several tests were conducted to ensure the accuracy of the developed Python package. To demonstrate its reliability, a comparison between the simulation conducted through the `PMD` library and the simulation of the analytical model of the double compound pendulum, illustrated in Fig.1, is presented

below. The equations of motion for the double pendulum were manually derived using the Lagrangian method, as shown in Eq.2.

$$\begin{cases} J_a\ddot{\theta}_1 + J_x\cos(\theta_1-\theta_2)\ddot{\theta}_2 + J_x\sin(\theta_1-\theta_2)\dot{\theta}_2^2 + \mu_1\sin\theta_1 = 0 \\ J_b\ddot{\theta}_2 + J_x\cos(\theta_1-\theta_2)\ddot{\theta}_1 - J_x\sin(\theta_1-\theta_2)\dot{\theta}_1^2 + \mu_2\sin\theta_2 = 0 \end{cases} \tag{2}$$

where:

$$J_a = \frac{1}{3}m_1 l_1^2 + m_2 l_2^2 \tag{3}$$

$$J_b = \frac{1}{3}m_2 l_2^2 \tag{4}$$

$$J_x = \frac{1}{2}m_2 l_1 l_2 \tag{5}$$

$$\mu_1 = \left(\frac{1}{2}m_1 + m_2\right) g l_1 \tag{6}$$

$$\mu_2 = \frac{1}{2}m_2 g l_2 \tag{7}$$

Once the equations of motion, expressed in terms of the system's degrees of freedom ($\theta_1$ and $\theta_2$, representing the angles of the first and second bodies relative to the $y$ axis, respectively), were obtained, the `x` and `y` coordinates of the center of mass for both bodies were back-calculated. These results were then compared with those obtained from the same model using the developed `PMD` library. Both simulations employed the same integration method (Radau). As shown in Fig.2, there is a perfect match between the two simulations, confirming that `PMD` is a valid simulation program and validating the routines developed for managing multibody models.

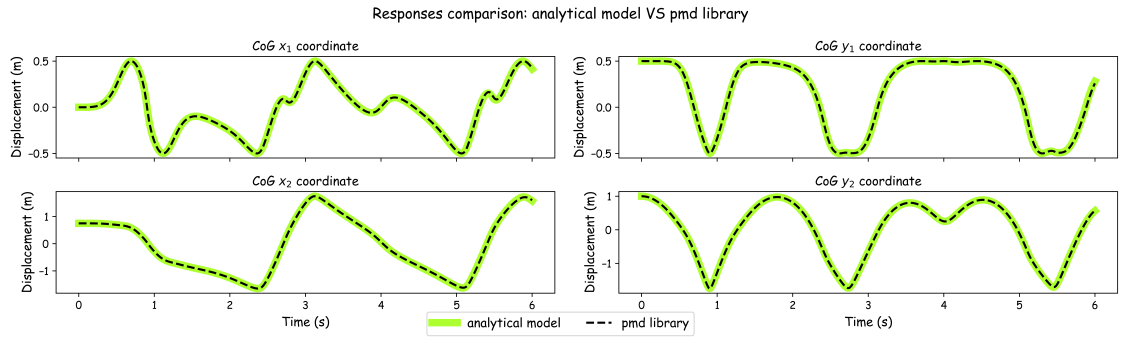



**Fig. 2:** Comparison of results between the analytical model and the `PMD` library model of the double compound pendulum. The initial conditions used for the simulation are $\theta_1 = \pi$ and $\theta_2 = \frac{\pi}{2}$.

## 3 Conclusion

The use of multibody dynamic models is essential for understanding the behavior of mechanical systems under various operating conditions. These models allow engineers to simulate complex interactions between different components, predict system responses, and identify potential issues before

they arise. The PMD Open-Source library provides an accessible and collaborative solution for modeling and simulating these systems. With its seamless integration into Python and the use of packages like NumPy and SciPy, PMD enables efficient simulation management and easy visualization of results. This integration allows for rapid prototyping and iterative testing, making it easier to refine models and improve accuracy. Additionally, the open-source nature of PMD fosters a collaborative environment where the community can actively contribute to its development, share insights, and build upon each other's work. Looking ahead, several future developments are planned for the PMD library. These include optimizing internal routines to enhance performance and efficiency, as well as integrating flexible bodies into the simulations. This will further improve the accuracy and realism of the models, making PMD an even more powerful tool for engineers and researchers.

## References


[1] F. Cianetti, R. Garzia, M. Palmieri, F. Ambrogi, and C. Braccesi, "An estimation model of suspension loads in explicit multibody simulation," *IOP Conference Series: Materials Science and Engineering*, vol. 1038, p. 012042, Feb. 2021.

[2] W. Schiehlen, "Multibody system dynamics: roots and perspectives," *Multibody system dynamics*, vol. 1, pp. 149–188, 1997.

[3] A. A. Shabana, "Flexible multibody dynamics: review of past and recent developments," *Multibody system dynamics*, vol. 1, pp. 189–222, 1997.

[4] A. A. Shabana, *Computational dynamics*. John Wiley & Sons, 2009.

[5] P. E. Nikravesh, *Planar Multibody Dynamics: Formulation, Programming with MATLAB®, and Applications*. Boca Raton London New York: CRC Press, Taylor & Francis, second edition ed., 2019.

[6] G. Cangi, A. Angeletti, M. Palmieri, and F. Cianetti, "A multibody mathematical model to simulate the dynamic behavior of aerial work platforms using python," *Engineering Proceedings*, vol. 85, no. 1, p. 36, 2025. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[7] J. R. Dye, Y. Y. Tay, and H. M. Lankarani, "Development and application of planar computational general-purpose constrained multibody simulations in matlab with simple graphical/visualization capability," in *Volume 4B: Dynamics, Vibration, and Control*, (Houston, Texas, USA), p. V04BT04A002, American Society of Mechanical Engineers, Nov. 2015.

[8] R. Timbó, R. Martins, G. Bachmann, F. Rangel, J. Mota, J. Valério, and T. Ritto, "Ross - rotordynamic open source software," *Journal of Open Source Software*, vol. 5, p. 2120, Apr. 2020.

[9] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. Van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with numpy," *Nature*, vol. 585, pp. 357–362, Sept. 2020. Publisher: Springer Science and Business Media LLC.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. Van Der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. Van Mulbregt, and SciPy 1.0 Contributors, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, Mar. 2020. Publisher: Springer Science and Business Media LLC.

[11] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[12] P. T. Inc., "Collaborative data science," 2015.

# Fiducial Marker-Based Motion Tracking and Correction: Implementation into pyIDI

Lorenzo Capponi* Klemen Zaletelj Janko Slavič

*Faculty of Mechanical Engineering, University of Ljubljana*

**Abstract**

Optical methods offer a powerful alternative to traditional sensors for the analysis of rotating structures, enabling non-contact, spatially dense measurements in scenarios where contact-based instrumentation is impractical or invasive. This study introduces an image-based approach that leverages fiducial markers to track and correct rigid in-plane motion. By estimating and inverting frame-to-frame transformations, the method effectively restores spatial alignment across image sequences, laying the groundwork for accurate subsequent analysis. A detailed uncertainty analysis confirms sub-pixel transformation accuracy under various experimental conditions, demonstrating the method's precision and reliability. Although the current implementation focuses on rotational motion, the approach is readily extendable to more complex transformations, such as Euclidean, affine, and homography, broadening its applicability across diverse structural analysis tasks. Originally developed for infrared thermography, the technique is equally suited to visible-spectrum imaging, supporting applications such as video stabilization and motion compensation in optical measurements of moving components. To encourage reproducibility and foster collaborative development, the entire processing workflow is provided as an enhancement to an existing open-source package.

**Keywords:** Fiducial Markers, Open Source, Computer Vision, Optical methods

## 1 Statement of Need

In the experimental analysis of large-scale moving structures, traditional sensor-based methods often face limitations due to the challenges of mounting sensors on moving components [1]. Optical methods offer a powerful, non-contact alternative, providing spatially dense measurements without interfering with the system under test [2]. This is especially valuable when contact-based instrumentation is impractical, enabling real-time monitoring and diagnostics in applications such as structural health monitoring and vibration fatigue analysis.

One of the key challenges in this context is compensating for structural motion, which can distort the captured data and hinder further investigations [3, 4, 5]. This study introduces a method that uses fiducial markers to track in-plane motion during image-based measurements [6, 7]. While simple motion tracking may suffice in some cases, the proposed method becomes particularly valuable when motion compensation is required for further analysis. By estimating and reversing frame-to-frame transformations, whether due to rotation or other types of rigid motion, the method compensates for

*Corresponding author, Email address: lorenzo.capponi@fs.uni-lj.si

structural movement, restoring alignment across frames and enabling accurate data interpretation. The robustness of the approach is demonstrated through uncertainty analysis, which reveals sub-pixel transformation errors under various experimental conditions. This supports its reliability and suitability for real-world applications. Although originally developed for rotational motion, the method can be extended to handle more complex transformations, such as Euclidean, affine, and homographic changes, broadening its applicability in structural analysis tasks.

Originally designed for infrared-range optical measurements and large-scale motion compensation, such as de-rotation, the technique is equally applicable to visible-spectrum imaging. This opens up opportunities for broader use cases, including video stabilization and motion compensation in optical-based analysis of moving components. By enabling the reconstruction of consistent pixel-intensity time histories at specific structural locations, even as they move across frames, the method ensures data integrity for downstream analyses. To promote open-source collaboration and support further advancements, the complete processing workflow is provided as an enhancement to an existing package [8]. This effort encourages transparency, reproducibility, and innovation within the optical measurement and structural dynamics communities.

## 2 Implemented Methodology

The geometric relationship between corresponding points in a pair of images can be modeled using homogeneous coordinates as:

$$\mathbf{P}' = \mathbf{Q}\mathbf{P}, \tag{1}$$

where $\mathbf{P}, \mathbf{P}' \in \mathbb{R}^3$ are homogeneous coordinates of corresponding points, and $\mathbf{Q} \in \mathbb{R}^{3\times3}$ is the planar transformation matrix. A general $\mathbf{Q}$ describes a homography, encompassing rotation, translation, scaling, shearing, and perspective distortion [9]. When perspective effects are negligible, $\mathbf{Q}$ simplifies to an affine or Euclidean form. In the latter case, $\mathbf{Q}$ reduces to:

$$\mathbf{Q} = \begin{bmatrix} \cos\theta & -\sin\theta & r_i \\ \sin\theta & \cos\theta & r_j \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where $\theta$ is the rotation angle and $r_i, r_j$ are the translation components. The inverse transformation restores a point to its reference frame:

$$\hat{\mathbf{P}} = \mathbf{Q}^{-1}\mathbf{P}', \tag{3}$$

where hat symbol in $\hat{\mathbf{P}}$ describes back-projected coordinates of a transformed point into reference frame. This framework tracks and compensates for rigid motion in a sequence of $K$ images, captured in either visible or infrared range. For each frame $k$, the backward transformation matrix $\mathbf{Q}_k^{-1}$ is estimated using $M$ fiducial markers, such as ArUco markers [6]. Any fiducial marker family can be used in place of ArUco. For infrared images, the markers must be properly manufactured to ensure sufficient contrast, leveraging high-emissivity and high-reflectivity materials. Fiducial markers offer robust frame correspondence, even with perspective distortions or partial occlusions, due to their unique binary grid identification, minimizing the need for complex feature matching. Let $\mathbf{P}_{m,c}$ and $\mathbf{P}'_{m,c}$ denote the homogeneous coordinates of the $c$-th corner of marker $m$ in the reference frame ($k = 0$) and the $k$-th frame, respectively. The transformed (back-projected) coordinates are obtained as:

$$\hat{\mathbf{P}}_{m,c} = \mathbf{Q}_k^{-1}\mathbf{P}'_{m,c}, \quad \forall m, c. \tag{4}$$

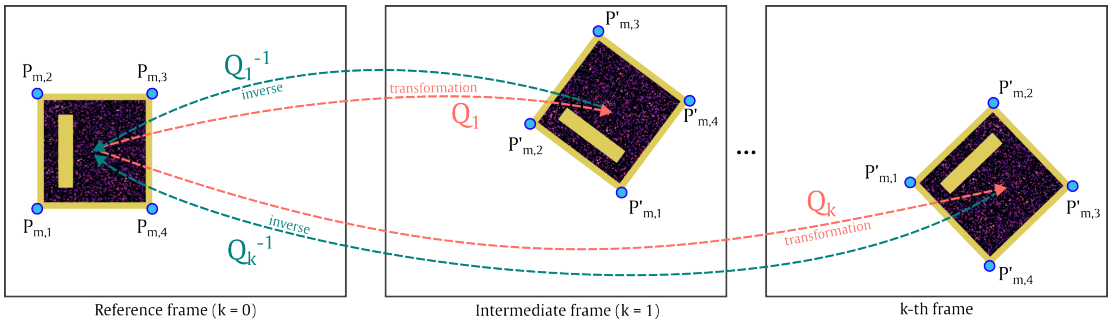



**Fig. 1:** ArUco marker-based transformation.

If the inverse matrices are applied to all points of interest, full motion compensation of the observed target can be achieved. The accuracy of the transformation is evaluated by the per-frame mean Euclidean error $\mu_k$ and its standard deviation $\tau_k$:

$$\mu_k = \frac{1}{MC} \sum_{m=1}^{M} \sum_{c=1}^{C} \left\| \mathbf{P}_{m,c} - \hat{\mathbf{P}}_{m,c} \right\|_2 \tag{5}$$

$$\tau_k = \sqrt{\frac{1}{MC} \sum_{m=1}^{M} \sum_{c=1}^{C} \left( \left\| \mathbf{P}_{m,c} - \hat{\mathbf{P}}_{m,c} \right\|_2 - \mu_k \right)^2} \tag{6}$$

The overall mean and standard deviation are then computed from the frame-to-frame errors using the results from the previous equations.

## 3 Conclusion

This study introduces a method for compensating rigid in-plane motion in optical measurements of rotating and moving structures using fiducial markers. By estimating and reversing frame-to-frame transformations, the method restores alignment for accurate analysis, achieving sub-pixel transformation accuracy across various experimental conditions. The technique, initially designed for rotational motion, can also handle more complex transformations such as Euclidean, affine, and homography, making it suitable for a wide range of structural analysis tasks. While developed for infrared measurements, it can be applied to visible-spectrum imaging, enabling motion compensation and video stabilization. All computations and image transformations described in this methodology are implemented in the `pyIDI` Python package. The contribution provides an open-source implementation of the processing workflow and is compatible with both infrared and visible imaging systems. It also supports a variety of fiducial marker families, allowing the method to be applied to different marker types depending on the application.

## 4 Acknowledgements

The authors acknowledge financial support from the European Union's Horizon-WIDERA-2023-TALENTS-02-01 topic, under the Marie Skłodowska-Curie grant agreement id 101180595, and the Slovenian Research and Innovation Agency (research core funding No. P2-0263).

## References


[1] Z. L. Mahri and M. S. Rouabah, "Fatigue estimation for a rotating blade of a wind turbine," *Journal of Renewable Energies*, vol. 5, pp. 39–47, 6 2002.

[2] P. J. Schubel, R. J. Crossley, E. K. Boateng, and J. R. Hutchinson, "Review of structural health and cure monitoring techniques for large wind turbine blades," *Renewable Energy*, vol. 51, pp. 113 – 123, 3 2013.

[3] R. Huňady, P. Pavelka, and P. Lengvarský, "Vibration and modal analysis of a rotating disc using high-speed 3d digital image correlation," *Mechanical Systems and Signal Processing*, vol. 121, pp. 201–214, 4 2019.

[4] Y. Chen and D. T. Griffith, "Experimental and numerical full-field displacement and strain characterization of wind turbine blade using a 3d scanning laser doppler vibrometer," *Optics and Laser Technology*, vol. 158, p. 108869, 2 2023.

[5] S. Hwang, Y. K. An, and H. Sohn, "Continuous line laser thermography for damage imaging of rotating wind turbine blades," *Procedia Engineering*, vol. 188, pp. 225–232, 1 2017.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, pp. 2280–2292, 6 2014.

[7] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, 8 2018.

[8] K. Zaletelj, D. Gorjup, and J. Slavič, "ladisk/pyidi: Release of the version v0.23," Sept. 2020.

[9] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. 2004.

# Analysis of Rotating Bladed Disk Vibrations Using Digital Image Correlation

Alessandra Cesaretti[1] Serena Occhipinti[1] Paolo Neri[2] Christian Maria Firrone[1]
Daniele Botto[1] Davide Mastrodicasa[3] Chiara Cardillo[1]

[1]*Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24*
[2]*Department of Civil and Industrial Engineering, Università di Pisa, Largo L. Lazzarno 1*
[3]*Siemens Industry Software NV, Interleuvenlaan 68, 3001, Leuven, Belgium*

**Abstract**

The analysis of rotating structures can be challenging because traditional sensors (such as accelerometers and strain gauges) typically require data transmission via telemetry. In addition, traditional sensors can only provide measurements at a few discrete points and introduce disturbances due to additional mass, stiffness, or damping that can affect the dynamics of the system. Therefore, a non-contact measurement approach such as Digital Image Correlation (DIC) offers potential advantages. DIC is an optical technique for measuring deformation parameters from digital images by correlating measurement points in reference and deformed images. This study proposes an original full-field approach to capture the dynamics of a rotating bladed disk by DIC. The proposed approach was tested on a simplified bladed disk using a dedicated spin rig.

**Keywords:** Disk Dynamics, Optical Technique, DIC, Down-sampling

## 1 Introduction

Monitoring the structural health of turbomachinery is crucial for preventing failures and ensuring optimal performance. Components such as disks and blades are highly susceptible to vibration damage due to the high natural frequency density and wide frequency range of the external force. Therefore, early detection of structural issues like excessive vibrations or fatigue can significantly improve the reliability and durability of these systems. Traditional sensors, such as strain gauges and accelerometers, are commonly used for vibration measurement, but they have significant limitations: they can interfere with the dynamics of the system because of their mass, provide measurements only at a few discrete locations, and require complex setups for power supply and data transmission. As a result, there is a growing interest in the development of non-contact measurement techniques. Blade tip timing (BTT) allows the monitoring of turbomachinery vibration [1], but it is essentially a single point technique, typically located at the blade tip. In this scenario, digital image correlation (DIC) provides a non-contact, full-field approach to measuring displacement and deformation of the entire structure. Digital images are recorded by two synchronized cameras. A speckle pattern has to be applied to the surface, enabling DIC to track the position of a grid of measurement points. While DIC is widely used to measure quasi-static and dynamic displacements of fixed structures, it faces challenges in studying

the dynamics of rotating structures. These challenges arise from the initial point position estimate [2], leading to inaccurate measurements when dealing with large displacements. Alternatively, when low displacements between two consecutive frames occur, incremental DIC algorithm [3] and Rigid Body Rotation (RBR) compensation techniques [4] can track vibrations effectively. This study focuses on the application of incremental DIC to the experimental analysis of a bladed disk of simplified geometry, rotating in a dedicated rotating system and excited by permanent magnets.

## 2 Methodology

The measurement process can be divided into three main steps:

- First, image reordering is performed to mitigate cumulative errors due to incremental DIC, which involves updating the reference image whenever correlation is lost as a result of large rotations. Each update introduces an initial estimation error, progressively affecting the displacement and trajectory of tracked points. For reducing the number of updates the image set is sorted by estimating the angular velocity for each frame and computing the corresponding phase of the bladed disk.
- Second, RBR is removed to isolate the dynamic behavior of the bladed disk. Two different methods are employed: the Point Cloud Overlapping (PCO) and the Hybrid (Hy) approach [4]. PCO is applied to incremental DIC results and involves computing the rotation parameters on low-vibration points to compensate RBR. The Hy approach integrates the PCO technique with the requirement of acquiring images of the bladed disk at its equilibrium position across multiple rotational angles, which are then processed before performing DIC analysis.
- Finally, vibration analysis is conducted on the displacement data obtain through image reordering and incremental DIC to extract operational deflection shapes, using Siemens Testlab [5].

## 3 Experimental test and results

The test rig and the DIC hardware are shown in figure 1. The tested bladed disk is made of aluminum and has twelve identical rectangular section blades whose length and width are 150 mm and 25 mm respectively, with a thickness of 5 mm. The outer diameter is 400 mm. Twelve cylindrical permanent magnets are glued at the tip of each blade. The disk was tested in a laboratory spinning rig and was fixed at the top of the shaft between two flanges. The excitation system employs cylindrical permanent magnets that interact with those mounted on the rotating disk. As the disk spins, this interaction produces impulsive repulsive forces, resulting in a non-contact periodic excitation. Two stationary coaxial disks hold the magnets at fixed angular positions. The number of magnets have to match the engine order (EO) of the simulated excitation. A vibrational mode with a specific nodal diameter (ND) can be excited by different EO values. For a disk with N blades, the EO that can excite a mode with nodal diameter ND are given by:

$$EO = k \cdot N \pm ND, \quad \forall k = 0, 1, 2, 3, ... \tag{1}$$

In order to excite the $ND = 4$, sixteen permanent magnets were equally spaced around the test rig. A linear sweep was performed, varying the rotational velocity between 0 and 600 rpm in 11 minutes.

The excitation frequency $\omega$ is related to both the $EO$ and the rotational frequency $\Omega$ as:

$$\omega = \Omega \cdot EO \tag{2}$$

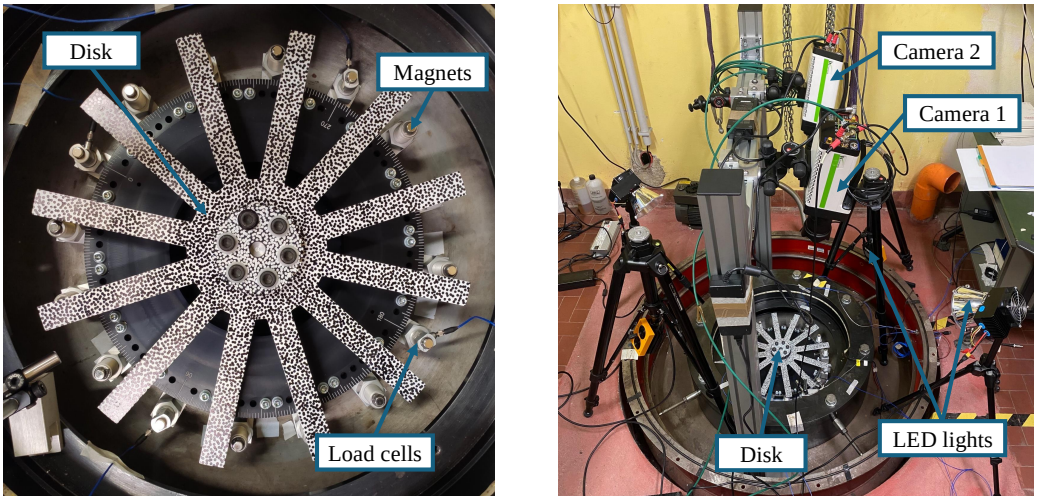



**Fig. 1:** Test rig (left) and DIC hardware (right)

The overall results are shown in Figure 2. The Hy approach with reordered images proves to be the most effective, as it nearly eliminates RBR and enables the most accurate extraction of deflection shapes. This improvement is due to fewer updates of the reference image in incremental DIC when images are reordered, reducing errors and leading to a more precise reconstruction of the motion.

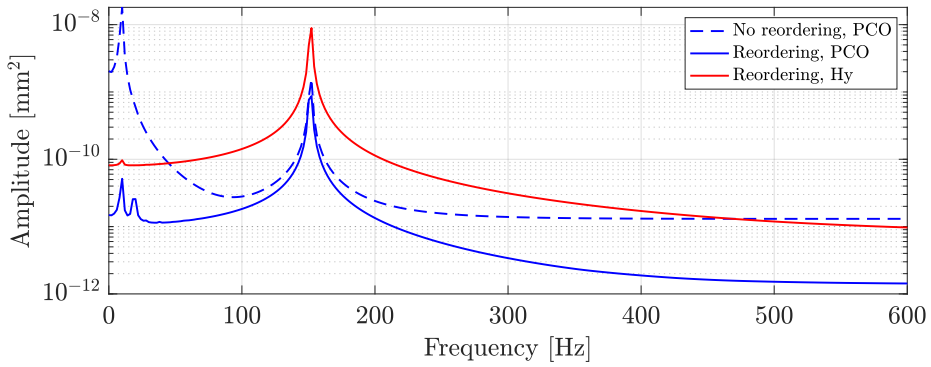



**Fig. 2:** Comparison between different RBR compensation approaches

The Campbell diagram shown in figure 3 is computed by using the Short-time Fourier Transform to investigate the excitation phenomena. The response amplitude reaches its maximum at the intersection of the resonance frequency and the EO lines that excite the bladed disk. Resonance phenomena are observed at EOs that are integer multiples of the number of magnets.

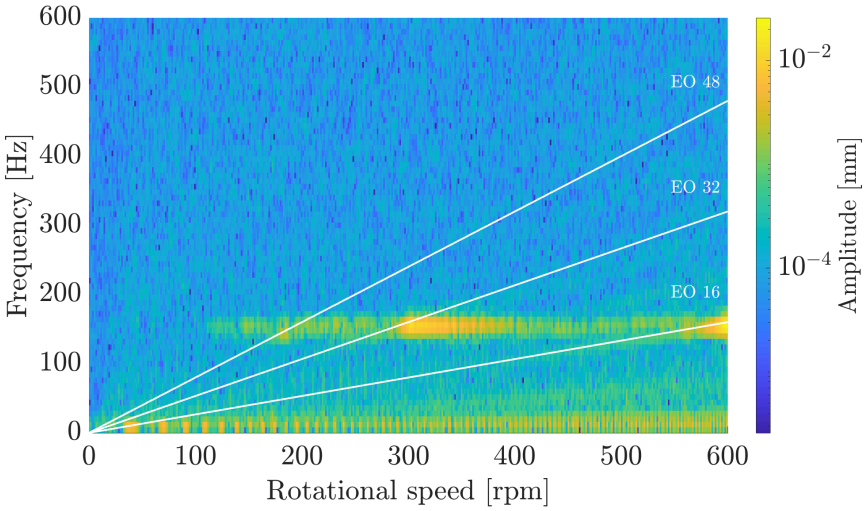



**Fig. 3:** Campbell diagram for the run-up from 0 to 600 rpm

## 4 Conclusion

This study demonstrates the reliability of an original DIC algorithm for vibration analysis of rotating bladed disks. The combination of image reordering and incremental DIC, along with the removal of RBR, allows for an accurate characterization of the dynamic behavior of the bladed disk. Experimental results show that the Hy approach, combined with image reordering and RBR compensation, produces the most accurate operational deflection shapes, effectively mitigating the errors typically introduced by large rotational displacements. This approach highlights the potential of DIC as a powerful, non-contact, full-field measurement technique for dynamic analysis of rotating structures, providing a valuable tool for turbomachinery monitoring and vibration analysis.

## References

[1] G. Battiato, C. Firrone, and T. Berruti, "Forced response of rotating bladed disks: Blade tip-timing measurements," *Mechanical Systems and Signal Processing*, vol. 85, pp. 912–926, Feb. 2017.

[2] J.-C. Passieux and R. Bouclier, "Classic and inverse compositional gauss-newton in global dic," *International Journal for Numerical Methods in Engineering*, vol. 119, no. 6, pp. 453–468, 2019.

[3] D. Minervini, D. Mastrodicasa, T. Geluk, and E. D. Lorenzo, "Novel methodology for isolating rotational phenomena in tire testing," *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, vol. 268, no. 7, pp. 1845–1856, 2023.

[4] S. Occhipinti, D. Mastrodicasa, S. Manzato, and E. D. Lorenzo, "Application of digital image correlation in operational modal analysis of rotating structures," *ISMA 2024 International Conference on Noise and Vibration Engineering Proceedings*, pp. 2544–2555, 2024.

[5] Siemens, "Simcenter testlab." `https://plm.sw.siemens.com/it-IT/simcenter/physical-testing/testlab/`, 2025.

# A Python toolbox for advanced time series analysis and generation

Giulio Curti [*] Massimiliano Palmieri Filippo Cianetti

*University of Perugia, Department of Engineering, Via G. Duranti 93, 06125 Perugia (Italy)*

**Abstract**

In the contemporary world, the term "sustainability" is gaining increasing interest. Within the domain of mechanical design, this phenomenon can be addressed by promoting the concept of structural design efficiency, thereby ensuring the effective utilization of materials during the design phase. However, the efficacy of this approach is contingent upon the establishment of a precise load hypothesis and correct life estimation. This involves the consideration of complex load histories, including non-gaussian and non-stationary scenarios. The prevailing methodologies for handling these loads are often plagued by time inefficiency and inconsistent outcomes.
This article presents an open-source Python toolbox that enables users to analyze and generate both simple and complex signals by employing several theories documented in the literature. Further research is necessary to enhance the accuracy of damage assessment on such loads, and the development of an open-source package could provide the scientific community with a foundational understanding of the current state of the art in generating such complex signals. The open-source nature of the toolbox allows community contribution by enabling members to compare and potentially augment the existing set of generating methods.
In summary, the Python package has been developed as a valuable resource for researchers, providing the tools to generate a substantial signal dataset for the testing and development of novel methodologies for the damage assessment on such complex loads.

**Keywords:** Signal processing, Signal synthesis, Non-Gaussianity, Non-Stationarity, Open Source, Python

## 1 Statement of Need

A common problem in mechanical design is load assessment. This challenge is particularly pronounced in scenarios where structures are exposed to time-varying loads and the assessment of fatigue life is necessary. In such instances, resonance phenomena can lead to an amplification of the applied load, consequently reducing the component's fatigue life.
Many standards from automotive, to aerospace to military [1] assume that complex time-varying time histories are globally stochastic and, thanks to the Center Limit Theorem, are simply describable as a series of gaussian processes via the spectral distribution of the variance of the load. This is often referred to as the Power Spectral Density (PSD) distribution [2]. The synthesis of these signals for testing purposes is a relatively straightforward process [3].

[*] Corresponding author, Email address: giulio.curti@dottorandi.unipg.it

However, it is important to note that real signals frequently do not exhibit these properties [4], assuming deformed distributions that are commonly characterized via higher order central moments [5]. Specifically, indicators such as skewness and kurtosis are frequently employed in this context.
It is critical to include such deformations in the load evaluation, as odd indicators such as kurtosis represent variation in the form of extreme values and, if not included, will result in an underestimation of fatigue damage [6].
A large number of non-Gaussian signal synthetization methods have been developed over the last century, each with its own differences. Notably, these methodologies are predicated on the stationarity hypothesis and the utilization of global higher-order central moments in their process-based analyses [7]. Also standards are considering the presence of non-Gaussianity during the testing procedure with the same hypothesis [1].
As the field of analysis has evolved, the development of analytical tools has been undertaken to gather the spectral distribution of parameters such as kurtosis [8].
The implementation of these recently gathered information into a synthesis method is intended, but to the best of the authors' knowledge, this has never been done.
Finally, the stationarity hypothesis is frequently erroneous. This phenomenon arises from the existence of multiple processes with distinct characteristics within a single time history [9]. The presence of these processes can lead to significant deviations in the statistical distribution, which, if treated as a simple non-gaussianity, can result in a high degree of error. This is due to the fact that the deformation of the distribution is not uniquely defined.
Methods for extracting valuable information regarding non-stationarity have been developed [10], [11], [12] and some methods for non-stationary signal synthesis are available in literature [13], [14], [15]. However, further research is ought to be done in this field.
All the cited literature regarding signal analysis and synthesis of complex time histories has never been enclosed into an Open Source package named *SignalForge* [16]. Therefore, pursuing the creation of such package would spring new researchers to develop skills around these fundamental problem that is highly meaningful not only to machine design but to many other fields.
Researchers of the field of signal processing could take advantage of the structure of this package for potential improvements on existing methods or the implementation of new approaches analyzing or synthesizing histories. Moreover, researchers on machine design could use this package to both analyze measured signals and design a set of complex loads that could spring new researches on the field of damage estimation considering non-stationarity and non-gaussianity.

## 2 SignalForge package structure

The proposed package is written in Python with an object-oriented structure. The architecture comprises a main class called SingleChanSignal that holds many signal post processing methods available in literature.
Subsequently, three child classes are designed to divide generable time histories into the three main macro areas defined in the statement of need section. Therefore, three classes are available to generate signals and their name is descriptive of the main property of the synthetic signal:

- class StationaryGaussian → Stationary gaussian signal generator
- class StationaryNonGaussian → Stationary non-gaussian signal generator
- class NonStationary → Non-stationary signal generator

Being child classes, they inherit all attributes and methods of the parent class SingleChanSignal.
Next sections will briefly explain the functionalities of each class.

# 3 Signal processing with SingleChanSignal

The SingleChanSignal class is generated by the user via the definition of the time history 'x' and the sampling frequency 'fs'. Optional attributes may be input, including the name of the signal, the variable name, and the unit of measure that is used for plotting analysis results.

SignalForge.SigleChanSignal (x, fs, dfpsd = 0.5, name = “”, var = “x”, unit = “m/s^2”)

Each method implemented in the class is written in a get_function() format to allow the user to manipulate the analysis results and with a plot_function() format to easily visualize results.
In addition to the fundamental functionalities encompassing the estimation of the probability density function, the calculation of the power spectral density, and the determination of the Fourier transform, the implementation encompasses several advanced analysis functionalities.

- Short Time Fourier Transform (STFT) [17]
- Spectrogram [17]
- Spectral kurtosis [8]
- Kurtogram [18]
- Non stationarity index [11]
- Wavelet transform [3] [19]

Profound research has been carried out regarding implemented signal analysis tools. A comprehensive review of the cited literature provides relevant information and applications.
The available methods of the class are shown in the package open source repository. [16]

# 4 Signal design

## 4.1 Generating stationary gaussian signals with StationaryGaussian

Stationary gaussian signals are simple to generate using the simple formulation proposed by Newland [3] with the following command.

SignalForge.StationaryGaussian(fpsd, psd, T)

The synthesis procedure involves obtaining the amplitudes of the Fourier transform of the signal from the PSD and then imposing a uniformly random phase. Consequently, the time history is generated through an inverse Fourier transform procedure (Fig 1).

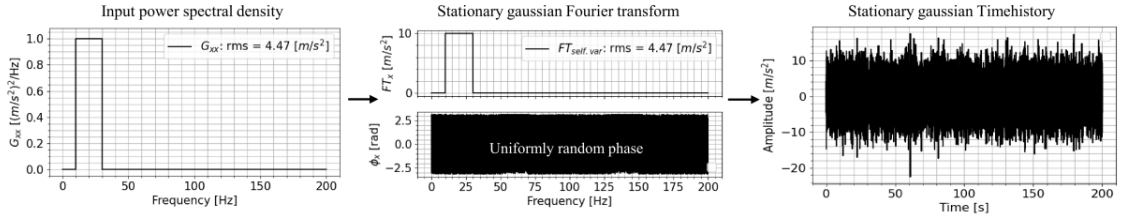



Fig. 1: Newland’s procedure of generating stationary gaussian signals

## 4.2 Generating stationary non-gaussian signals with StationaryNonGaussian

Generating a stationary non-gaussian signal with the StationaryNonGaussian class is carried out via a simple class definition with the intended parameters of skewness, kurtosis and the method to be used

for the synthetization process. An initial assumption of spectral distribution of the signal's power, and therefore its overall energy, must be provided in terms of PSD.

SignalForge.StationaryNonGaussian(fpsd, psd, T, kurtosis, skewness = 0, method = 'winter')

Often, since the transformations are non-linear, methods rely on an optimization procedure to obtain the parameters that control the nonlinear transform function. These are returned to the user after the optimization process.

Many methods are present in literature regarding the synthetization of non-gaussian signals. These methods can be grouped into three categories: memoryless nonlinear transformations, phase modification and filtered Poisson processes [7].

The memoryless nonlinear transformation methods implemented in this package are the following:

- Hermite-Winterstein ('winter') [20]
- Cubic polynomial ('cubic') [21]
- Hyperbolic transform – Zheng ('zheng') [22]
- Sarkani ('sarkani') [23]
- Improved nonlinear transform ('zmnl') [24]

Memoryless transformations are characterized by their simplicity and widespread utilization due to their ease of implementation. These transformations involve the pointwise transformation of a Gaussian process via a non-linear transformation function. This function is monotonic, and its shape varies with the method employed. (Fig. 2)

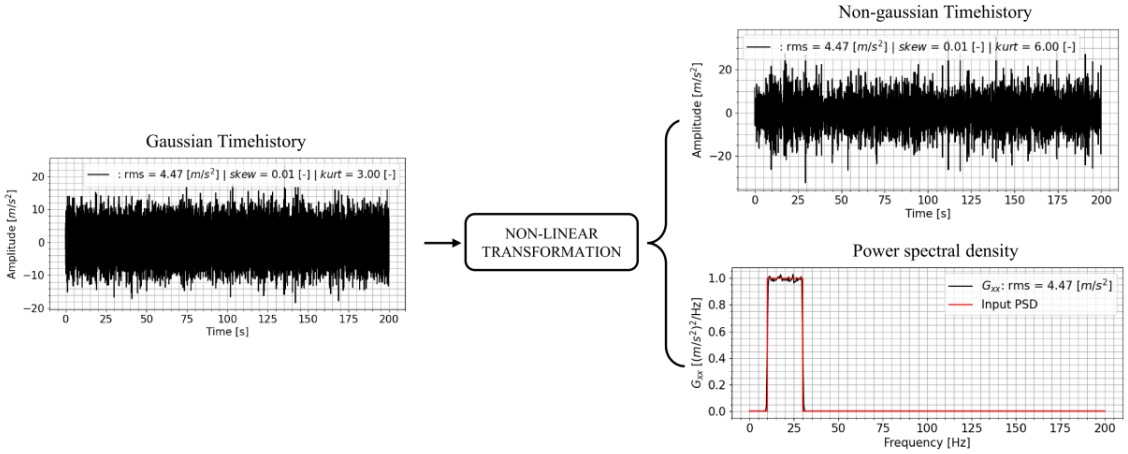



Fig. 2: Standard procedure of memoryless non linear transformation for adding non gaussianity

The phase modification methods implemented are the following:

- Steinwolf ('steinwolf') [25]

With this methods only the Fourier transform of the signal is used since the phase is considered to be responsible for non-gaussian behavior (Fig. 3). The relationship between phases goes against randomness and therefore deforms the statistics of the final signal while maintaining the amplitudes of the frequency content.

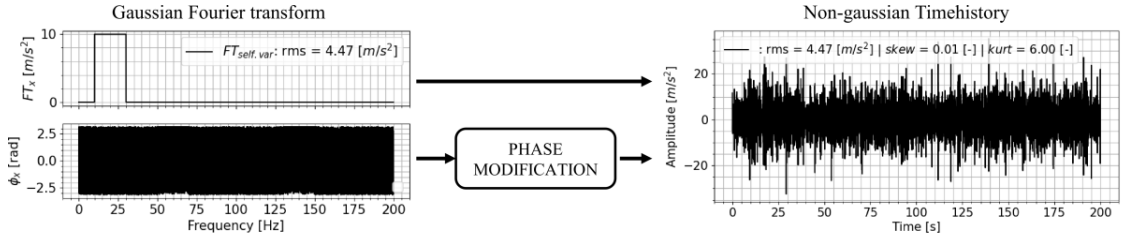



Fig. 3: Phase modification procedure to generate non-gaussian signals

The Filtered Poisson processes methods implemented are the following:

- Smallwoood Poisson shots ('smallwood') [26]
- Van Baren ('vanbaren') [27]

These methods rely on the construction of the signal as a series of impulses whose frequency content is coherent with the final intention. The frequency of the shot impulses is controlled by an exponential distribution that is the parameter to be optimized (Fig. 4). It is noteworthy that in instances where the signal is extensive and the objective high-order parameters are elevated, the optimization procedure can prove to be computationally onerous.

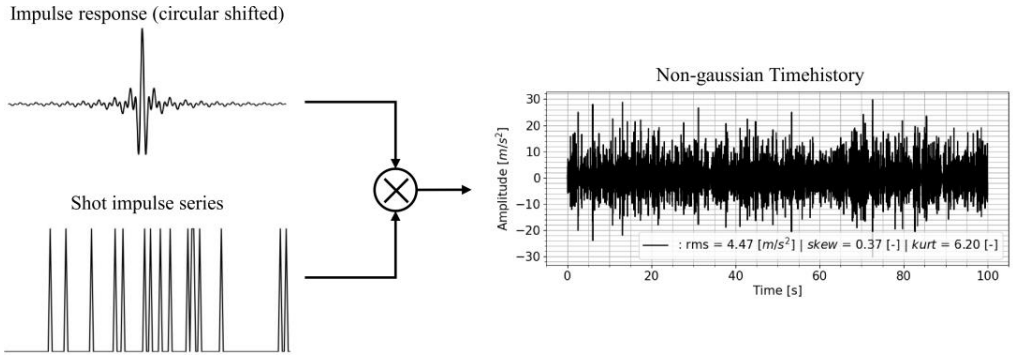



Fig. 4: Procedure for generating Filtered Poisson non gaussian processes

### 4.3 Generating non-stationary signals with NonStationaryNonGaussian

Non-stationarity is a complex problem in terms of both analysis and reproduction. Frequently, this condition arises due to the presence of multiple processes within the same time history. Each process contributes a segment of the signal that is challenging to quantify a priori. Consequently, global statistics are frequently erroneous due to insufficient sampling, which fails to adequately represent the process.To generate the process, an input PDS and the time length of the signal must be provided along with the method and relative parameters with the following command:

SignalForge.NonStationaryNonGaussian(fpsd, psd, T, method, params = None)

Common synthetization methods can be categorized into two distinct classifications: the addition of amplitude-modulated and frequency-modulated non-stationarities.
The following methods for generating amplitude modulated non-stationary signals have been implemented:

- Beta amplitude modulation ('beta_am') [14]
- Gaussian amplitude modulation ('trapp_am') [13]

- Rayleigh amplitude modulation ('ray_am') [14]

The common practice in amplitude-modulated synthetic time histories is to introduce non-stationarity to a stationary Gaussian history by locally varying the overall power of the signal (Fig. 5). The carrier function, which modulates the original Gaussian signal, is designed using different approaches depending on the method used. However, irrespective of the method employed, the frequency content of the signal remains constant.

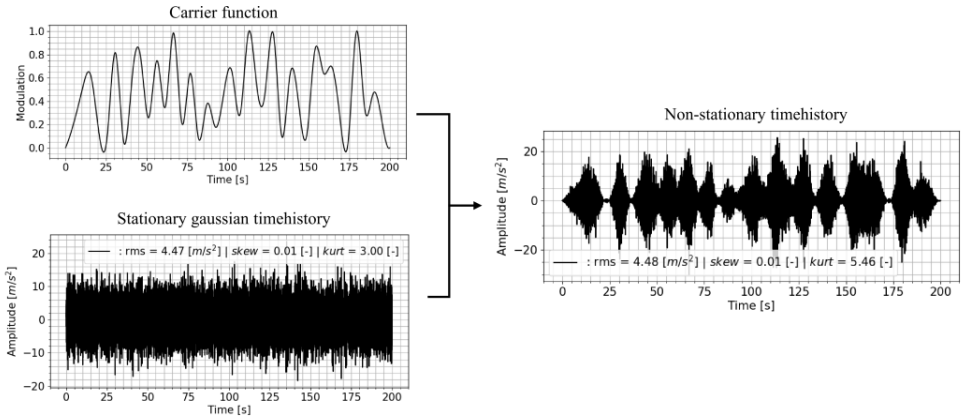



Fig. 5: Standard procedure for adding non stationarity via amplitude modulation

Regarding frequency modulation the following method has been implemented:

- Frequency modulation via STFT ('fm') [28]

Another prevalent form of non-stationarity pertains to the time-varying frequency content of the signal. This phenomenon can occur when forced harmonics excite a structure, such as a motor or an inverter. Algorithms for obtaining a signal from a STFT already exist; therefore, designing one with time-varying frequency content is a conceivable undertaking. The frequency shift is imposed via a user-defined carrier function, with which the STFT is designed, and the original frequency content is given via a PSD. The final stage of the process involves generating the resulting signal. (Fig. 6)

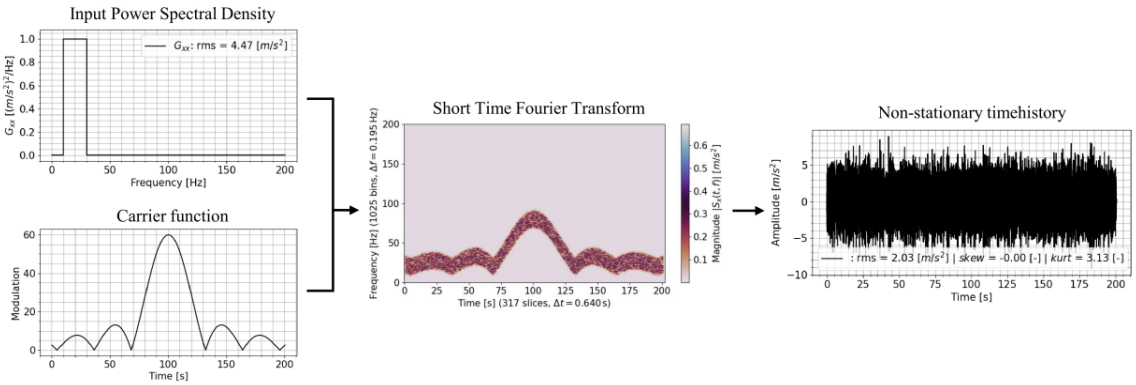



Fig. 6: Procedure for generating a frequency modulated non-stationary signal

# 5 Conclusions and future improvements

This article is intended to serve as a foundational starting point for a more extensive project. A substantial number of analysis and synthesis methods have been implemented, many of which are well-known and widely utilized. However, it should be noted that there are numerous additional methods documented in the relevant literature. The open-source nature of this project is intended to encourage contributions from individuals seeking to enhance the package.
The planned improvements to the package are focused on its expansion to accommodate multiple-channel signals, a subject that has recently garnered significant interest. Furthermore, existing methods for generating correlated signals with non-Gaussian properties should be integrated into the package, as this addition is likely to be of significant value [22], [26], [29], [30].

# References


[1] US Department of defense, *MIL-STD-810H - ENVIRONMENTAL ENGINEERING CONSIDERATIONS AND LABORATORY TESTS*, 2019.

[2] A. G. Piersol e J. S. Bendat, *Random data: analysis and measurement procedures*. Hoboken, N.J.: Wiley, 2013.

[3] D. E. Newland, *An introduction to random vibrations, spectral & wavelet analysis*, 3. ed., Unabridged republ. Mineola, NY: Dover Publ, 2005.

[4] M. Decker, «Vibration fatigue analysis using response spectra», *Int. J. Fatigue*, vol. 148, p. 106192, lug. 2021, doi: 10.1016/j.ijfatigue.2021.106192.

[5] C. L. Nikias e A. P. Petropulu, *Higher-order spectra analysis: a nonlinear signal processing framework*. in Prentice Hall signal processing series. Englewood Cliffs, N.J: PTR Prentice Hall, 1993.

[6] M. Palmieri, M. Česnik, J. Slavič, F. Cianetti, e M. Boltežar, «Non-Gaussianity and non-stationarity in vibration fatigue», *Int. J. Fatigue*, vol. 97, pp. 9–19, apr. 2017, doi: 10.1016/j.ijfatigue.2016.12.017.

[7] R. Zheng, G. Chen, e H. Chen, «Stationary non-Gaussian random vibration control: A review», *Chin. J. Aeronaut.*, vol. 34, fasc. 1, pp. 350–363, gen. 2021, doi: 10.1016/j.cja.2020.10.005.

[8] J. Antoni, «The spectral kurtosis: a useful tool for characterising non-stationary signals», *Mech. Syst. Signal Process.*, vol. 20, fasc. 2, pp. 282–307, feb. 2006, doi: 10.1016/j.ymssp.2004.09.001.

[9] D. Benasciutti e R. Tovo, «Frequency-based fatigue analysis of non-stationary switching random loads», *Fatigue Fract. Eng. Mater. Struct.*, vol. 30, fasc. 11, pp. 1016–1029, nov. 2007, doi: 10.1111/j.1460-2695.2007.01171.x.

[10] J. Antoni, «Cyclostationarity by examples», *Mech. Syst. Signal Process.*, vol. 23, fasc. 4, pp. 987–1036, mag. 2009, doi: 10.1016/j.ymssp.2008.10.010.

[11] L. Capponi, M. Česnik, J. Slavič, F. Cianetti, e M. Boltežar, «Non-stationarity index in vibration fatigue: Theoretical and experimental research», *Int. J. Fatigue*, vol. 104, pp. 221–230, nov. 2017, doi: 10.1016/j.ijfatigue.2017.07.020.

[12] P. Wolfsteiner, «Fatigue assessment of non-stationary random vibrations by using decomposition in Gaussian portions», *Int. J. Mech. Sci.*, vol. 127, pp. 10–22, lug. 2017, doi: 10.1016/j.ijmecsci.2016.05.024.

[13] A. Trapp, M. J. Makua, e P. Wolfsteiner, «Fatigue assessment of amplitude-modulated non-stationary random vibration loading», *Procedia Struct. Integr.*, vol. 17, pp. 379–386, 2019, doi: 10.1016/j.prostr.2019.08.050.

[14] D. Smallwood, «Vibration with Non-Gaussian Noise», *J. IEST*, vol. 52, fasc. 2, pp. 13–30, ott. 2009, doi: 10.17764/jiet.52.2.gh0444564n8765k1.
[15] A. Meynard e B. Torrésani, «Synthesis-based time-scale transforms for non-stationary signals», *Appl. Comput. Harmon. Anal.*, vol. 65, pp. 112–136, lug. 2023, doi: 10.1016/j.acha.2023.02.001.
[16] G. Curti, *SignalForge*. Python. [Online]. Disponibile su: https://github.com/GiulioCurti/SignalForge
[17] K. Gröchenig, *Foundations of Time-Frequency Analysis*. in Applied and Numerical Harmonic Analysis. Boston, MA: Birkhäuser Boston, 2001. doi: 10.1007/978-1-4612-0003-1.
[18] J. Antoni, «Fast computation of the kurtogram for the detection of transient faults», *Mech. Syst. Signal Process.*, vol. 21, fasc. 1, pp. 108–124, gen. 2007, doi: 10.1016/j.ymssp.2005.12.002.
[19] M. Rhif, A. Ben Abbes, I. R. Farah, B. Martínez, e Y. Sang, «Wavelet Transform Application for/in Non-Stationary Time-Series Analysis: A Review», *Appl. Sci.*, vol. 9, fasc. 7, p. 1345, mar. 2019, doi: 10.3390/app9071345.
[20] S. R. Winerstein, «Nonlinear Vibration Models for Extremes and Fatigue», *J. Eng. Mech.*, vol. 114, fasc. 10, pp. 1772–1790, ott. 1988, doi: 10.1061/(ASCE)0733-9399(1988)114:10(1772).
[21] S. R. Winterstein, C. H. Lange, e S. Kumar, «Fitting: Subroutine to fit four-moment probability distributions to data», SAND--94-3039, 10123319, gen. 1995. doi: 10.2172/10123319.
[22] R. Zheng, H. Chen, e X. He, «Control Method for Multiple-Input Multiple-Output Non-Gaussian Random Vibration Test: MIMO Non-Gaussian Random Vibration Test», *Packag. Technol. Sci.*, vol. 30, fasc. 7, pp. 331–345, lug. 2017, doi: 10.1002/pts.2303.
[23] S. Sarkani, D. P. Kihl, e J. E. Beach, «Fatigue of welded joints under narrowband non-Gaussian loadings», *Probabilistic Eng. Mech.*, vol. 9, fasc. 3, pp. 179–190, gen. 1994, doi: 10.1016/0266-8920(94)90003-5.
[24] G. Wise, A. Traganitis, e J. Thomas, «The effect of a memoryless nonlinearity on the spectrum of a random process», *IEEE Trans. Inf. Theory*, vol. 23, fasc. 1, pp. 84–89, gen. 1977, doi: 10.1109/TIT.1977.1055658.
[25] A. Steinwolf, «Random vibration testing with kurtosis control by IFFT phase manipulation», *Mech. Syst. Signal Process.*, vol. 28, pp. 561–573, apr. 2012, doi: 10.1016/j.ymssp.2011.11.001.
[26] D. O. Smallwood, «Generation of Stationary Non-Gaussian Time Histories with a Specified Cross-spectral Density», *Shock Vib.*, vol. 4, fasc. 5–6, pp. 361–377, 1997, doi: 10.1155/1997/713593.
[27] P. Van Baren, «System and method for simultaneously controlling spectrum and kurtosis of a random vibration», US20070185620
[28] M. Clerc e S. Mallat, «Estimating deformations of stationary processes», *Ann. Stat.*, vol. 31, fasc. 6, dic. 2003, doi: 10.1214/aos/1074290327.
[29] N. F. Hunter, K. R. Cross, e G. Nelson, «The Cross Spectrum in Multiple Input Multiple Response Vibration Testing», in *Topics in Modal Analysis & Testing, Volume 9*, M. Mains e B. J. Dilworth, A c. di, in Conference Proceedings of the Society for Experimental Mechanics Series. , Cham: Springer International Publishing, 2019, pp. 91–102. doi: 10.1007/978-3-319-74700-2_10.
[30] R. Zheng, H. Chen, D. Vandepitte, e Z. Luo, «Multi-exciter stationary non-Gaussian random vibration test with time domain randomization», *Mech. Syst. Signal Process.*, vol. 122, pp. 103–116, mag. 2019, doi: 10.1016/j.ymssp.2018.12.013.

# Acoustic field solution in a pipe using CAFE

Daniele Fabbri * Fabio Bruzzone Carlo Rosso

*Politecnico di Torino*

**Abstract**

The use of the finite element method in the acoustic field has grown in recent years, as has the desire to reduce noise in an environment. Understanding how acoustic waves propagate inside a particular structure and how they interact with the surrounding space is possible using numerical methods often implemented in commercial software. This work aims to demonstrate how acoustic problems can be solved using open-source resources. A crucial step in numerical simulations is the creation of a mesh that accurately represents the geometry of interest. Since considerable effort is often dedicated to mesh generation using tools such as Gmsh, this study leverages an existing mesh file for acoustic analysis. In CAFE, it is possible to assign a specific material and build matrices for acoustic mass, stiffness, and damping, depending on the required boundary condition. This work focuses on studying the propagation of sound waves inside a rectangular section pipe. The acoustic source is represented by the imposed velocity boundary condition assigned to one of the four sides. A perfectly matched layer theory is used to simulate an infinite spatial domain in the numerical case. To validate the model, this work compares the results from CAFE with those from commercial software.

**Keywords:** Acoustic Structure Interaction, Open Source, Structural Dynamics, Python

## 1 Introduction

Acoustic-Structure Interaction (ASI), often termed vibroacoustics, refers to the two-way coupling between sound waves in a fluid medium and vibrations of solid structures. In simple terms, it is the branch of multiphysics which study the effects of a structure in a fluid domain (usually air) understanding how the pressure fluctuations interact with a flexible structure.
In order to study acoustic-structure interaction problems, it is essential to solve the acoustic and structural fields coupled or not. For complex geometries, numerical methods such as the finite element method are commonly employed. While the finite element discretization of structural components is broadly explained in a lot of work as [1] and several open-source implementations are available online, this is not the case for the acoustic domain. According to the authors' knowledge and after an extensive literature review, few widely available open-source software was found that solves the acoustic field using the finite element method within a programming environment as widely used as MATLAB, it was therefore decided to develop CAFE (Computational Acoustic Finite Element), which uses open source resources to discretize geometries and solve the acoustic field on it. Moreover, the authors are working to introduce the vibro acoustic module as soon as possible. The software workflow will be

*Corresponding author, Email address: daniele.fabbri@polito.it

explained in section 2.

The starting point is the governing equation for the pressure in an acoustic medium. The frequency domain is governed by the Helmholtz equation:

$$\nabla^2 P + \frac{\omega^2}{c_a^2} P = -i\rho_a \omega q \qquad in\ \Omega_a \tag{1}$$

where: $P$, $\rho_a$ and $c_a$ are the pressure, density of the acoustic domain and the speed of sound in the analysis domain $\Omega_a$. The angular frequency and the wavenumber are denoted by $\omega$ and $k$, respectively. The Helmholtz equation can be solved using the appropriate boundary conditions, and in this paper different boundary conditions will be used.

In this paper the focus is on the use of finite element to discretize the acoustic domain following the theory described in [2, 3, 4].
The starting point is writing the pressure $P$ using the standard shape functions matrix $N_p$:

$$\hat{P} = \sum_{j=1}^{n_p} \mathbf{N_p} P_j \qquad j = 1...n_p \tag{2}$$

here $n_p$ are the number of prescribed shape functions for the pressure. After using the field variable approximation in the weighted residual formulation of the Helmholtz equation the solution of the pressure field is obtained solving this system of linear equations :

$$\left(\left[\mathbf{K}\right] + i\omega\left[\mathbf{C}\right] - \omega^2\left[\mathbf{M}\right]\right)\{P\} = \{F_a\} \tag{3}$$

where $\mathbf{K}$, $\mathbf{C}$, $\mathbf{M}$ are the stiffness, damping and mass matrices, $F_a$ is instead is the acoustic force vector.

In the following section, the working principles of CAFE will be described and benchmark results will be presented compared to commercial software.

## 2 CAFE Workflow and results

To run a simulation on CAFE it is mandatory to have a geometry discretized with finite element, to do it the Python open source code called GMSH [5] is used. Once the mesh is imported in CAFE, appropriate boundary condition must be applied. For the sake of clarity, these conditions are not described in detail here, but are simply listed: *Pressure Soft, Prescribed Pressure, Hard Wall, Prescribed Velocity, Prescribed Impedance, Perfectly Matched Layer (PML)*. Although it is not strictly a boundary condition, the *PML* domain is treated as such, since some numerical software adopts this convention and does not display the *PML* domain in the final results. The software provides a visual representation of the boundary conditions applied before solving the system of equations, as shown in Figure 2.

This case study was specifically developed to evaluate whether the software produces results consistent with those obtained from established commercial solvers such as COMSOL. In CAFE it is possible to use both linear and quadratic 2D elements, and the user can also choose whether to perform a modal analysis or directly solve the system described in Equation 3. Since the objective of this work is to evaluate the performance of CAFE under operational conditions, a comparison between the

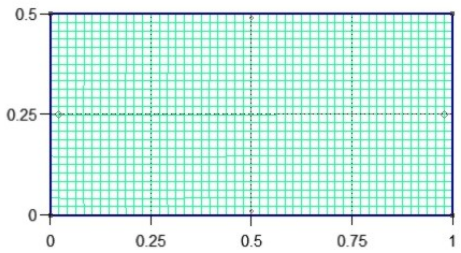



**Fig. 1:** Mesh of the geometry

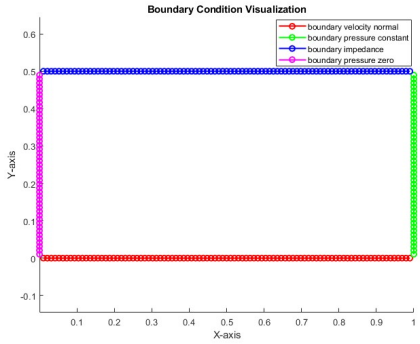



**Fig. 2:** Boundary condition visualization.

natural frequencies obtained using CAFE and those computed with COMSOL will not be presented. Nevertheless, it is worth noting that the average percentage error remains below 5%. In the example presented in this paper, different boundary conditions have been applied to the geometry. On the side highlighted in magenta in Figure 2, the acoustic pressure is set to zero. The red side represents a boundary condition with imposed velocity, while the green side is subjected to a constant pressure of 2 Pa. Finally, an impedance boundary condition of 1000 $Pas/m$ is applied to the blue side. The direct approach has been adopted to solve the problem, in the frequency range between 20 Hz and 800 Hz, with a frequency step of 20 Hz. The focus is on the low-frequency range, as this is where acoustic-structure interaction is more likely to occur, representing the primary application area for CAFE. In line with the concept of *h-p* refinement [6] in finite element simulations, CQUAD8 elements are employed to reduce the total number of elements while maintaining accuracy. Applying the rule of thumb of 5 to 6 elements per minimum wavelength, the geometry is discretized into approximately 2000 elements, resulting in a model with 16000 degrees of freedom.
For the sake of brevity, this work considers only the acoustic pressure comparison between the results obtained with CAFE and COMSOL, shown in Fig.3. The results show that, except for a slightly different trend in the very low frequency range (e.g. 100-200 Hz),are in agreement for both software, reproducing the same trend for the acoustic pressure , particularly in the mid-frequency range. Since this is a benchmark case aimed at verifying the correct implementation of the elements formulations and boundary conditions, the results are certainly encouraging. This is further supported by the relatively low errors observed in the modal analyses, which could be leveraged through a modal solution approach.

## 3 Conclusions

This paper presents the results obtained using CAFE, a newly developed software designed to support acoustic-structure modeling in an open and collaborative way. The project aims to encourage cooperation within the academic community by providing a shared and transparent tool. Future work will include a deeper understanding of the differences observed in the low-frequency trend, the implementation of more advanced boundary conditions and the development of the vibro-acoustic module.

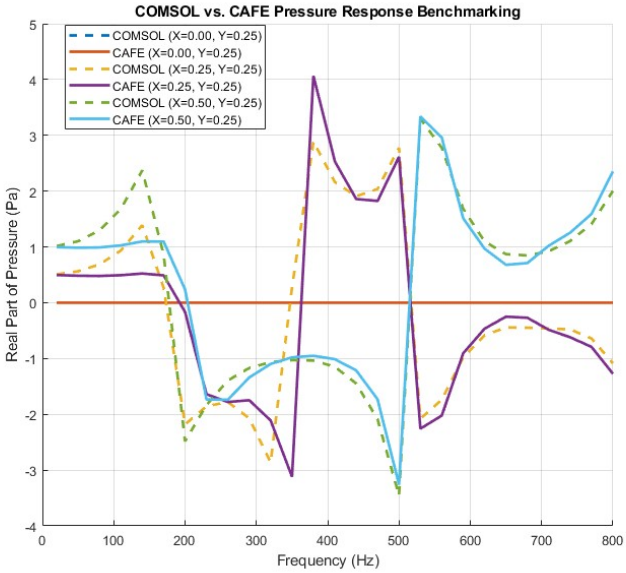



**Fig. 3:** Acoustic Pressure Comparison between CAFE and COMSOL, Frequency range: 20-800 Hz

## 4 Acknowledgments

This publication is part of the project PNRR-NGEU which has received funding from the MUR – DM 117/2023

## References

[1] R. H. Macneal, *Finite Elements: Their Design and Performance*. 270, Madison Avenue, New York: DEKKER, 1994.

[2] W. Desmet and D. Vandepitte, "Finite element method in acoustics," 1999.

[3] F. Ihlenburg and I. Babuška, "Finite element solution of the helmholtz equation with high wave number part i: The h-version of the fem," *Computers Mathematics with Applications*, vol. 30, no. 9, pp. 9–37, 1995.

[4] A. G. Prinn, "A review of finite element methods for room acoustics," *Acoustics*, vol. 5, no. 2, pp. 367–395, 2023.

[5] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, pp. 1309 – 1331, 09 2009.

[6] I. Babuška, "The p and h-p versions of the finite element method: The state of the art," in *Finite Elements* (D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt, eds.), (New York, NY), pp. 199–239, Springer New York, 1988.

# Sound Power Radiation from Vibration Measurements with Optical Methods: MATLAB functions

Paolo Gardonio[1*] Sofia Baldini[1] Domen Gorjup[2] Janko Slavič[2] Roberto Rinaldo[1]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine, (IT)*

[2] *University of Ljubljana - Faculty of Mechanical Engineering, Askerceva 6 1000 Ljubljana (SI)*

**Abstract**

This paper presents a new methodology for the measurement of both the narrow band and third-octave band spectra of the total vibration kinetic energy and total sound power radiation due to the flexural vibration of closed shells. To start with, the paper briefly recalls the computer vision techniques used to reconstruct the displacements at a regular grid of points on the surface of the shell structure from multi-view video acquisitions taken with a single high-speed and high-resolution camera. Digital image correlation combined with frequency domain triangulation is implemented, which is particularly suited for the derivation of spectral measurements. Then, the paper presents the derivation of the narrow band spectra of the vibration kinetic energy and sound power radiation, which are calculated from discretised versions of two integral formulations, in particular the Kirchooff–Helmholtz integral formulation for the sound radiation. The third-octave band spectra of the vibration kinetic energy and sound power radiation are then calculated straightforwardly by appropriate integration over frequency-bands of the narrow band spectra. The paper provides both the mathematical formulation for the derivation of the vibration and sound radiation narrow band and third-octave band spectra and a MATLAB function for the calculation of the sound radiation matrix, which is the kernel for the derivation of the total sound power radiation. Also, it reports the narrow band and third-octave band spectra of the vibration kinetic energy and sound power radiation estimated for a baffled cylinder model-structure.

**Keywords:** single view videogrammetry; frequency-domain triangulation; sound power radiation narrow band spectrum; sound power radiation third-octave band spectrum

## 1 Introduction

The measurement of the total flexural vibration and sound radiation by structures is of great importance for the analysis of Noise, Vibration, and Harshness (NVH) of transportation vehicles as well as for the Noise and Vibration (N&V) classification of domestic appliances and structural components of buildings (principally partitions and windows) [1,2]. In general, acoustic measurements are implemented either in situ (e.g. with far-field microphones array, nearfield holography, sound intensity probes, acoustic cameras) or in large anechoic or reverberant rooms. Both solutions present a few drawbacks linked to background and flanking disturbances, accuracy and repeatability of the measurement, cost of the equipment and of the infrastructure (for more details see Ref. [3]). For this reason, a new approach based on the reconstruction of the vibration of the structure from full-field camera measurements has been recently explored by Gardonio *et. al.* [4-6]. The

[*] Paolo Gardonio, Email address: paolo.gardonio@uniud.it

principal advantage of this new approach is given by the fact that it does not require special acoustic facilities and measurement setups to mask the effect of noise and flanking sources and it relies on full field acquisitions taken over significantly short periods of time, such that the measurement is not influenced by variations of measurement conditions (e.g. temperature, tensioning effects due to operation of the machine, etc.). For this reason, there is quite a lot of interest on the derivation of both vibration and sound radiation form full field camera measurements of structures and machines. In this context, this paper is focussed on the derivation of narrow band and third-octave band spectra of the flexural kinetic energy and sound power radiation of a baffled cylinder model-structure from video measurements taken with a single camera. Both the narrow band and third-octave band spectra of the vibration kinetic energy and sound power radiation derived from the camera measurements are validated against measurements taken on the cylinder rig with a laser vibrometer and a microphones array respectively.

## 2 Model-structure and measurement setups

As shown in Figure 1, the proposed methodology for the reconstruction of the flexural vibration and sound radiation by closed shell has been tested on a model-structure made by a thin-walled cylinder, equipped with two rigid baffle extensions and excited by a radial force exerted by an inner shaker. This peculiar model structure was adopted because it allows the closed form derivation of the sound radiation from flexural vibration field estimated from camera measurements [7]. The cylinder wall is made of steel and has thickness $h = 1$ mm, hight $H = 300$ mm and inner radius $R = 150$ mm. The flexural vibration and sound radiation have been reconstructed from closed form integral expressions over the surface of the cylinder, which have been approximated into a summation over a $62 \times 20$ grid of quadrangular elements (20 elements along the length and 62 along the circumference of the cylinder). Both flexural vibration and sound radiation measurements were taken respectively with a camera and laser vibrometer and with an array of microphones. As shown in Figure 1a, the camera measurements were taken with a single high-resolution and high-speed camera (Photron FASTCAM SA-Z), which, however, was set to acquire multi-views of the cylinder. More precisely, the camera was kept fixed whereas the cylinder was rotated around its principal axis by fixed angular steps. The Digital Image Correlation (DIC) technique [8,9] was thus implemented using the multiple views

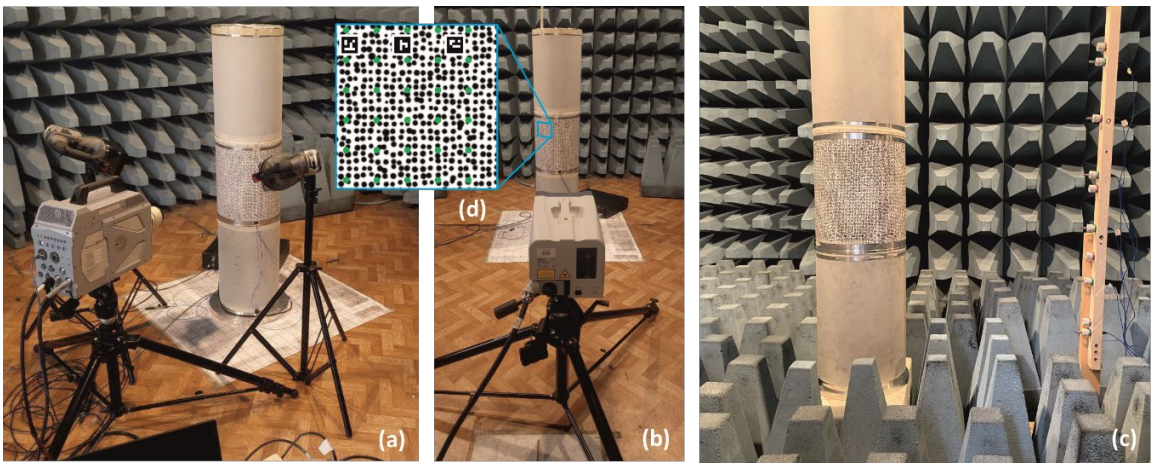



Fig. 1: Cylinder rig with (a) camera setup, (b) laser vibrometer setup, (c) microphones setup. (d) details of the speckle pattern for the camera (black dots) and laser (green dots) measurements.

camera acquisitions, which, as highlighted in Figure 1d, had been covered by a thin film painted with a dense speckle pattern of black dots. As depicted in Figure 1b a scanner laser vibrometer was employed too to measure the flexural vibration field. Here the measurements were taken at the centres of the $62 \times 20$ grid of quadrangular elements, which, as shown in Figure 1d, had been highlighted with green dots on the thin film glued onto the cylinder. Finally, as can be seen in Figure 1c, the sound radiation was measured with an array of microphones. To derive the total sound power radiation, the measurement was inspired to the International Organization for Standardization (ISO) criteria for the measurement of sound radiation (e.g. see ISO 3740:2019). As can be seen in Figure 1c, the measurement setup consisted of a vertical array of 6 microphones that covered the high of the cylinder $H = 300$ mm extended on the top and bottom by $\Delta_H = 150$ mm. The measurements were repeated 8 times with $\pi/4$ rotations of the baffled cylinder. Thus, overall a $6 \times 8$ grid of sound pressure measurements were taken on a cylindrical measurement surface having radius $R_m = 750$ mm and height $H_m = 600$ mm, which was aligned with the vibrating cylinder.

# 3 Reconstruction of flexural vibration and sound radiation

The video acquisitions were used to reconstruct the flexural vibration of the cylindrical wall at the centre points of the cylinder $62 \times 20$ grid of quadrangular elements. More specifically, Digital Image Correlation [8,9] combined with frequency-domain triangulation [10,11] of multi-view video acquisitions were employed to derive receptance Frequency Response Functions (FRF)

$$\alpha_{r,j}(\omega) = \frac{u_{rj}(\omega)}{f_r(\omega)}\,, \tag{1}$$

which are given by the ratio of the complex radial displacements $u_{rj}(\omega)$ at the $62 \times 20$ grid of point and the complex amplitude of the radial force excitation $f_r(\omega)$. Here, time-harmonic displacements and a time-harmonic force were considered, given in the forms $u_{rj}(t) = Re\{u_{rj}(\omega)\exp(j\omega t)\}$ and $f_r(t) = Re\{f_r(\omega)\exp(j\omega t)\}$, where $j = \sqrt{-1}$ and $\omega$ is the circular frequency. According to Ref. [10], the complex amplitudes of the radial displacements $u_{rj}(\omega)$ were derived from the following matrix relation:

$$\mathbf{u}'_{cj}(\omega) + \mathbf{x}'_{cjREF} = \frac{1}{w_{REF}}\mathbf{P}\,(\mathbf{u}_{cj}(\omega) + \mathbf{x}_{cjREF})\,, \tag{2}$$

where, the vector $\mathbf{u}'_{cj}(\omega)$ contains the complex amplitudes of the $j$-th marker point time-harmonic displacement in the image plane of the camera, $\mathbf{x}_{cjREF}$ gives the rest position of the $j$-th marker point. Also, $\mathbf{P}$, $w$ are respectively the projection matrix and perspective scale factor that defines the 2D image perspective of the camera via the non-linear expression $\mathbf{x}' = \frac{1}{w}\mathbf{P}\,\mathbf{x}$, where $\mathbf{x}'$ and $\mathbf{x}$ are respectively the homogeneous vectors with the camera-2D and spatial-3D coordinates $\mathbf{x} = (x, y, z, 1)^T$, $\mathbf{x}' = (x', y', 1)^T$. Here, the vector $\mathbf{x}_{cj}(\omega) = \mathbf{u}_{cj}(\omega) + \mathbf{x}_{cjREF}$ contains the complex amplitudes of the three components of the marker position in the Euclidean space, which, as shown in Ref. [10] can be used to implement a multi-view triangulation in the frequency-domain that leads to an univocal displacement vector $\mathbf{u}_{cj}(\omega)$. The complex amplitude of the radial component of the displacement can then be extracted from the vector $\mathbf{u}_{cj}(\omega)$ with the following projection

$$u_r(\mathbf{x}_{cj}, \omega) = \frac{\mathbf{u}_{cj}(\omega) \cdot \mathbf{n}_j}{|\mathbf{n}_j|}\,. \tag{3}$$

Reference [12], provides the details of the frequency-domain triangulation from multi-view video acquisitions employed in this study, whereas Refs. [10,13] give an exhaustive overview of the theory behind this approach.

Assuming time-harmonic vibrations, the time-averaged total flexural kinetic energy of the cylinder wall was then derived by summing the kinetic energies due to the radial displacements of the $N_e = 62 \times 20 = 1240$ quadrangular elements [12], such that:

$$\bar{K}(\omega) = \omega^2 \frac{M_c}{4N} \sum_{j=1}^{N_e} \left|\alpha_{r,j}(\omega)\right|^2 F_r^2(\omega) \,, \tag{4}$$

where $M_c$ is the mass of the cylinder and $F_r(\omega) = |f_r(\omega)|$ is the amplitude of the excitation. Likewise, the total sound power radiation was derived by summing the sound radiations produced by the radial displacements of the cylinder quadrangular elements [12], such that

$$\bar{P}(\omega) = \omega^2 \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} \alpha_{r,i}^*(\omega) R_{ij}(\omega) \alpha_{r,j}(\omega) F_r^2(\omega) \,, \tag{5}$$

where $^*$ stands for complex conjugate and the radiation resistances $R_{ij}(\omega) = \mathrm{Re}\{Z_{ij}(\omega)\}$ are derived from the acoustic radiation impedance functions [12]

$$Z_{ij}(\omega) = -j \frac{\omega \rho_0 Se}{2\pi^2} \sum_{n=0}^{N} \varepsilon_n \cos\left(n\left(\theta_i' - \theta_j\right)\right) I_n\left(z_i' - z_j\right) \,, \tag{6}$$

where

$$I_n\left(z_i' - z_j\right) \approx \sum_{m=0}^{M} \frac{\cos\left(m\Delta k_z\left(z_i' - z_j\right)\right)}{\sqrt{k^2 - (m\Delta k_z)^2} R} \frac{H_n^{(2)}\left(\sqrt{k^2 - (m\Delta k_z)^2} r_i'\right)}{H_n^{(2)'}\left(\sqrt{k^2 - (m\Delta k_z)^2} R\right)} \Delta k_z \,. \tag{7}$$

These expressions are defined with reference to cylindrical coordinates $(r, \theta, z)$ at the centre of the cylinder. Also, $\varepsilon_n = 1$ for $n = 0$, $\varepsilon_n = 2$ for $n > 0$, $\rho_0$ is the density of air, $c$ is the speed of sound in air, $k = \omega/c$ is the acoustic wavenumber, $k_z$ is the projection of the acoustic wavenumber into the longitudinal direction of the cylinder and $k_r = \sqrt{k^2 - k_z^2}$. Finally, $H_n^{(2)}$, $H_n^{(2)'}$ are the 2nd-kind Hankel function and its derivative respectively [14]. The limits of the two summations $N$, $M$ and the $\Delta k_z$ step were selected in such a way as to guarantee convergence of the closed form integral version of these equations (see detailed discussion in [12]).

To have a reference value, the total sound power radiation was also derived from measurements of acoustic impedance-like FRFs:

$$Z_m(\omega) = \frac{p_m(\omega)}{f_r(\omega)} \,, \tag{8}$$

which are given by the ratio of the complex sound pressure $p_m(\omega)$ at the $6 \times 8$ microphone positions and the complex amplitude of the radial force excitation $f_r(\omega)$, assuming $p_m(t) = Re\{p_m(\omega)\exp(j\omega t)\}$ and $f_r(t) = Re\{f_r(\omega)\exp(j\omega t)\}$. In this case the total sound power radiation resulted given by

$$\bar{P}(\omega) = \frac{1}{2\rho_0 c} \sum_{m=1}^{N_m} |p_m(\omega)|^2 = \frac{1}{2\rho_0 c} \sum_{m=1}^{N_m} |Z_m(\omega)|^2 F_r^2(\omega) \,, \tag{9}$$

# 4 Narrow and third-octave band spectra of K and P

Figures 2 and 3 present the narrow band spectra of the total vibration kinetic energy and total sound power radiation per unit force excitation. Figure 2 shows a rather good matching between the spectra of the vibration kinetic energy derived from the laser vibrometer measurements and the camera measurements. In general, the resonance peaks in the spectrum derived from the camera measurements are less pronounced than those in the spectrum derived from the laser vibrometer measurements. This is principally because the spectrum from the camera measurements has been derived with a comparatively coarser frequency sampling.

Figures 4 and 5 show third-octave band plots of the power spectral densities of the Vibration Kinetic Energy Level (KEL) and Sound Power Level (PWL), which were derived from the following relations for the *n*-th band [15]:

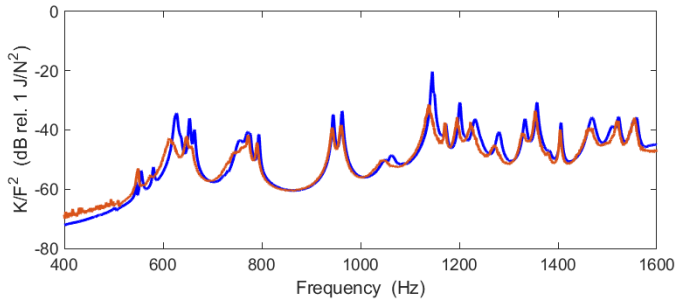



Fig. 2: Narrow band spectrum of the total vibration kinetic energy per unit force excitation from measurements with camera (red line) and laser vibrometer (blue line).

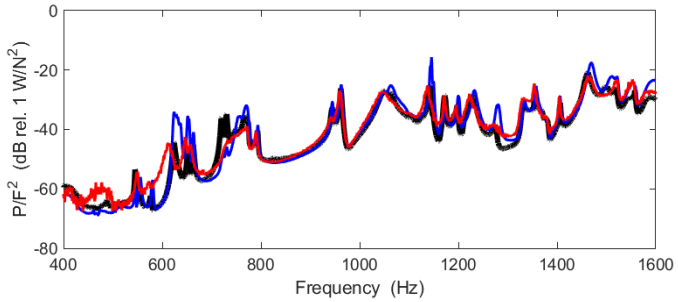



Fig. 3: Narrow band spectrum of the total radiation sound power per unit force excitation from measurements with camera (red line), laser vibrometer (blue line) and microphones (black line).

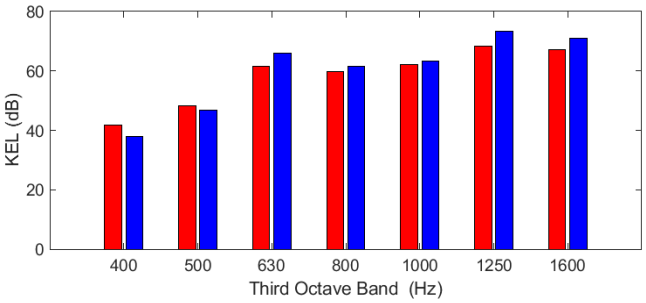



Fig. 4: Third-octave band spectrum of the total vibration kinetic energy from measurements with camera (red bar) and laser vibrometer (blue bar) – dB ref. $10^{-12}$ W.

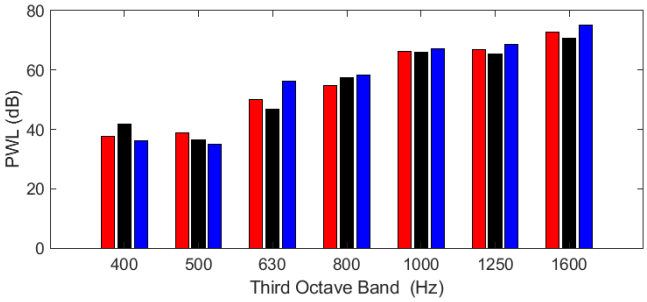



Fig. 5: Third-octave band spectrum of the total radiation sound power from measurements with camera (red bar), laser vibrometer (blue bar) and microphones (black bar) – dB ref. $10^{-12}$ J.

$$\mathrm{KEL}_n = 10\log_{10}\frac{\bar{K}_n}{K_r} \quad , \quad \mathrm{PWL}_n = 10\log_{10}\frac{\bar{P}_n}{P_r} \tag{10,11}$$

where $K_r = 10^{-12}$ J = 1 pJ and $P_r = 10^{-12}$ W = 1 pW are reference values for the kinetic energy and sound power and

$$\bar{K}_n = \frac{\Delta B_n}{N_i}\sum_{i=i_1}^{i_2}\bar{K}(\omega_i) \quad , \quad \bar{P}_n = \frac{\Delta B_n}{N_i}\sum_{i=i_1}^{i_2}\bar{P}(\omega_i) \tag{12,13}$$

are the band-limited values, which were calculated using the spectral values $\bar{K}(\omega_i)$, $\bar{P}(\omega_i)$ given in Eqs. (4), (5). Here $\Delta B_n$ is the width of the $n$-th third-octave band and $N_i = i_2 - i_1$, where $i_1$, $i_2$ are the indices of the frequency-samples at the lower and maximum frequency of the third-octave band [2].

Figure 5 contains the PWL derived from the measurements taken with the microphones too, which was derived considering the ISO guidelines for the measurement of sound radiation [2], such that:

$$\mathrm{PWL}_n = \mathrm{SPL}_n + 10\log_{10}\frac{S_m}{S_0}, \tag{14}$$

where $\mathrm{SPL}_n$ is the band-limited sound pressure level and $S_m = 2.8$ m$^2$ is the measurement surface and $S_0 = 1$. The $\mathrm{SPL}_n$ has been derived from the following relation:

$$\mathrm{SPL}_n = 10\log_{10}\frac{1}{N_m}\sum_{m=1}^{N_m}\frac{\bar{p}_{n,m}^2}{p_r^2}, \tag{15}$$

where $p_r = 20\ \mu$Pa and the band-filtered time-average square pressure $\bar{p}_{n,m}^2$ has been derived from measurements of the FRFs defined in Eq. (8), such that

$$\bar{p}_{n,m}^2 = \frac{\Delta B_n}{2N_i}\sum_{i=i_1}^{i_2}|p_m(\omega_i)|^2 = \frac{\Delta B_n}{2N_i}\sum_{i=i1}^{i2}|Z_m(\omega_i)|^2\, F_r^2(\omega) \tag{15}$$

Figure 4 shows that there is quite a good agreement between the third octave band $\mathrm{KEL}_n$ derived from the camera video acquisitions (red bars) and that derived from direct measurements taken with the laser vibrometer (blue bar). In general, the measurements based on the camera acquisitions are between 1 and 3 dB lower than those obtained from the measurements with the laser vibrometer. As noticed above, this is principally caused by the fact that the spectrum reconstructed from the camera measurements is characterised by lower resonance peaks than the spectrum derived from the laser vibrometer measurements.

Figure 5 shows a reasonably good agreement of the third octave band $\mathrm{PWL}_n$ derived from the camera video acquisitions (red bars) and from direct measurements taken with the laser vibrometer (blue bar) and the $\mathrm{PWL}_n$ measured directly with the array of microphones (black bars). In general, the spectra reconstructed from measurements taken wither with the cameras (red bars) and the laser vibrometer (blue bars) differ by 1 to 3 dB except for the 630 Hz third octave band where there is a difference of about 5 dB. Here, besides the differences of the resonance peaks noticed in the narrow band spectra of Figure 3, there are probably two other aspects that affect the measurements. The first one depends on the spatial distribution of the flexural displacements reconstructed from the camera measurements and measured directly with the laser vibrometer that could lead to different estimates of the sound radiation narrow band spectra and thus third-octave band spectra. The second one depends instead on the fact that the spectrum derived from the measurements taken with the microphones provides an approximate estimate only, which, moreover, is affected by the quality of the anechoic room, that is from the reverberation time and the background noise.

# 5 CONCLUSIONS

At best knowledge of the authors, this paper has first proved the feasibility of classical narrow band and third-octave band spectral measurements of the vibration kinetic energy and sound power radiation by a flexible structure using optical camera video acquisitions. The spectra reconstructed from the camera measurements overlap well with those derived from direct measurements taken with a laser vibrometer and an array of microphones arranged according to ISO standards.

# Appendix: MATLAB code for radiation acoustic impedances

```
function p=zij(r,theta,z, a, thetas, zz, f0, c0, rho0, S, v, M, tol, Madd)
% Computes the contribution of a single source element % at a, thetas, zz to the target at r,theta,z
% (possibly on the same cylinder, with r=a)
% r, theta, z coordinates of the target point a, thetas, zz coordinates of source point, with a the cylinder radius
% f0 frequency, c0 speed of sound, rho0 fluid density, S area element, v velocity of source element
% M number of integration points between 0 and k (default M=180)
% tol tolerance to stop integration (default tol=1e-4)
% Madd additional terms for kz beyond k - imaginay argument (default Madd=1000).
% If Re[Zpq] is needed, this function is called with Madd=0.
if nargin==11
   M=180;
   tol=1e-4;
   Madd=1000;
end
omega=2*pi*f0;
k=omega/c0;
delta=k/M;
p=-1i*rho0*omega*S*v*delta/(2*a*pi^2);
small_disp=delta/20;
n=0;
kz=(0:M+Madd-1)*delta+small_disp;
arg1=(sqrt(k^2-kz.^2)*a);
arg2=(sqrt(k^2-kz.^2)*r);
idxr=find(real(arg1)~=0);
idxi=find(real(arg1)==0);
add=sum((cos(kz(idxr)*(z-zz)).*besselh(n,2,arg2(idxr))./...
   (arg1(idxr).*dbesselh(n, 2, arg1(idxr))));
add=add+sum((cos(kz(idxi)*(z-zz)).*besselk(n,abs(arg2(idxi)))./...
   (abs(arg1(idxi)).*dbesselk(n, abs(arg1(idxi)))) );
en=2;
while (1)
   n=n+1;
   addi=en*cos(n*(theta-thetas))*sum( (cos(kz(idxr)*(z-zz)).*besselh(n,2,arg2(idxr))./...
      (arg1(idxr).*dbesselh(n, 2, arg1(idxr))) );
   addi=addi+ en*cos(n*(theta-thetas))...
      *sum( (cos(kz(idxi)*(zz)).*besselk(n,abs(arg2(idxi)))./...
      (abs(arg1(idxi)).*dbesselk(n, abs(arg1(idxi)))) );
   add=add+addi;
   if ((abs((addi))<tol*abs(add)) || (n>50)) % limit sum to n=50
      break;
   end
end
p=p*add;
end
```

## References


[1] R. F. Barron, "Industrial noise control and acoustics," Dekker, 2003.

[2] D. A. Bies, C. H. Hansen, "Engineering noise control : theory and practice," Crc Press, 2017.

[3] F. J. Fahy, "Measurement of audio-frequency sound in air", in Fundamentals of Sound and Vibration, Ed. F. J. Fahy and D. J. Thompson, CRC Press Taylor & Francis Group, 2015

[4] P. Gardonio, et. al., "Multi-view videogrammetry for the estimate of plate sound radiation," Proceedings of the 14th Int. Conference on Vibration Measurements by Laser and Noncontact Techniques, 2021. doi:10.1088/1742-6596/2041/1/012013

[5] P. Gardonio, et. al., "Free-field sound radiation measurement with multiple synchronous cameras," Measurement 188, 2022 110605. https://doi.org/10.1016/j.measurement.2021.110605.

[6] P. Gardonio, et. al., "Reconstruction of the sound radiation field from flexural vibration measurements with multiple cameras," Mechanical Systems and Signal Processing, 195, 2023 https://doi.org/10.1016/j.ymssp.2023.110289.

[7] M.C. Junger, D. Feit, "Sound, structures, and their interaction," The Mit Press, Imp, 2010.

[8] M. N. Helfrick, et. al., "3D digital image correlation methods for full-field vibration measurement", Mechanical Systems and Signal Processing, 25, 2011, 917–927.

[9] J. Baqersad, et. al., "Photogrammetry and optical methods in structural dynamics – A review," Mechanical Systems and Signal Processing, 86, 2017, 17–34. https://doi.org/10.1016/j.ymssp.2016.02.011.

[10] D. Gorjup, et. al., "Frequency domain triangulation for full-field 3D operating-deflection-shape identification," Mechanical Systems and Signal Processing, 133, 2019, 143–152.

[11] J. Javh, et. al., "Measuring full-field displacement spectral components using photographs taken with a DSLR camera via an analogue fourier integral," Mechanical Systems and Signal Processing, 100, 2018, 17–27

[12] S. Baldini, et. al., 3D Sound Radiation Reconstruction from Camera Measurements, Mechanical Systems and Signal Processing, Volume 227, 2025, https://doi.org/10.1016/j.ymssp.2025.112400

[13] D. Gorjup, et. al., "Still-camera multiview Spectral Optical Flow Imaging for 3D operating-deflection-shape identification," Mechanical Systems and Signal Processing, 152, 2021, 107456. https://doi.org/10.1016/j.ymssp.2020.107456.

[14] E. Skudrzyk, "The Foundations of Acoustics," Springer-Verlag, New York, 1971

[15] L.L. Beranek and T.J. Mellow, "Acoustics, Sound Fields and Transducers," Academic Pres - Elsevier, Amsterdam, 2012

# Ensuring Robustness in Open-Source Structural Dynamics Tools: Lessons from Debugging and Testing pyFRF

Domen Gorjup * Janko Slavič

*University of Ljubljana, Faculty of Mechanical Engineering*

**Abstract**

In open-source software development, ensuring the reliability of scientific computing tools is critical for their adoption and trustworthiness, especially in fields close to industrial applications, like structural dynamics. This paper presents the process of identifying and correcting a subtle phase inversion bug within pyFRF, a Python package for computing frequency response functions (FRFs) in experimental modal analysis. The error arose due to discrepancies in signal processing method formulae found in various literature sources, compounded by an incomplete unit testing framework. By tracking the issue, we illustrate the importance of rigorous testing and clear documentation for open-source projects. We discuss how the bug was resolved, and how a robust suite of unit tests was developed to prevent future errors. Through this case study, we highlight best practices for maintaining the reliability and accuracy of open-source scientific tools, including continuous integration, automated testing, and collaborative development practices. These practices serve as a model for future contributors to pyFRF and similar tools in structural dynamics.

**Keywords:** pyFRF, open-source, unit testing, EMA

## 1 Statement of Need

Structural health monitoring and modal analysis are critical for ensuring the safety, functionality, and longevity of mechanical and civil engineering structures. This drives the increasing need for robust Operational Modal Analysis (OMA) and Experimental Modal Analysis (EMA) techniques, which can extract modal parameters from ambient or experimental vibration data.

However, the development of signal processing algorithms is not solely a mathematical exercise. A critical review of prior literature is essential to avoid redundant implementations, rediscovering known limitations, or overlooking important edge cases. Foundational works such as those by Maia and Silva [1] offer well-tested derivations and empirical insights that should inform any new implementation.

Equally important is the integration of rigorous unit testing during software development. Modal analysis routines involve complex linear algebra operations and numerical sensitivity. Without systematic verification of core routines, the integrity of the analysis pipeline is compromised. Unit tests allow each component to be independently validated, which is crucial in ensuring that FRF estimation methods produce reliable and reproducible results.

*Corresponding author, Email address: domen.gorjup@fs.uni-lj.si

# 2 Methodology

We consider a linear time-invariant system under ambient excitation, modeled as a Multi-Input Multi-Output (MIMO) system. The goal is to estimate the system's Frequency Response Functions (FRFs) based on measured inputs and outputs.

## 2.1 $H_1$ Estimator Derivation

The $H_1$ estimator is commonly used when input forces are measurable and noise is primarily present in the output signals. Its derivation is outlined below.

Let the output vector $X$ and input vector $F$ be related through the FRF matrix $[H]$:

$$X_{m\times 1} = [H]_{m\times n} \cdot F_{n\times 1} \tag{1}$$

$$X^T_{1\times m} = F^T_{1\times n} \cdot [H]^T_{n\times m} \tag{2}$$

Pre-multiplying (2) by the complex conjugate of the excitation vector, $F^*$, we get:

$$[S_{FX}] \quad = [S_{FF}] \cdot [H]^T \tag{3}$$

$$\Rightarrow [H_1] \quad = \left([S_{FF}]^{-1} \cdot [S_{FX}]\right)^T \tag{4}$$

Therefore, the MIMO $H_1$ estimator for the FRF is:

$$[H_1]_{m\times n} = \left([S_{FF}]^{-1}_{n\times n} \cdot [S_{FX}]_{n\times m}\right)^T \tag{5}$$

Notice that an important assumption was made at this step: by convention, we define the cross-spectra and auto-spectra of excitation and response as:

$$[S_{FX}] \quad = F^* \cdot X^T, \tag{6}$$

which is equivalent to the definition in [1] and also how `scipy.signal.csd` [2], used in the background of our FRF estimator code, compute cross-spectra.

### 2.1.1 The phase inversion bug in pyFRF

This convention is at the root of the phase inversion issue encountered in the case of pyFRF. In some signal processing literature, the cross-spectrum is defined using the complex conjugate of the second signal rather than the first. This seemingly minor difference leads to subtle but important changes in the resulting FRF formulae — most notably, a phase inversion. When one part of the code estimates the FRF using one convention, and another part computes the cross-spectrum using a different one, the mismatch can go unnoticed, as each part appears correct within its own context. This makes such errors difficult to detect. It is therefore essential to define such mathematical conventions explicitly and apply them consistently across the entire codebase. Every component of the implementation must be clearly understood and tested to avoid unintended discrepancies.

## 3 Software Development Considerations

Implementing modal analysis algorithms therefore requires more than mathematical correctness—it also demands software engineering discipline. Given the modular nature of EMA pipelines, poor implementation in a single function (e.g., FFT windowing, spectral estimation, etc.) can invalidate the entire analysis.

We advocate for:

- **Automated unit tests**: All crucial components of your implementation, as well as the end result, should be verified against known benchmarks.
- **Test-driven development**: Writing tests before implementing the functionality ensures precise API behavior and promotes modular design.
- **Documentation of assumptions**: Each analytical step (e.g., assuming noise-free inputs for $H_1$) must be clearly documented to guide users and future developers.

These practices also facilitate code reuse and ensure compliance with open science and reproducibility standards.

## 4 Unit Test Case: Synthetic MIMO Validation

To verify the correctness of the implemented Frequency Response Function (FRF) estimator, multiple synthetic unit test were added th the existing test suite of pyFRF. The goal is to compare the FRFs computed from the code with analytically known ground truth FRFs generated from a simulated linear MIMO system, not only in amplitude, but also in phase. This allows for a full validation.

As an example, an FRF estimation unit test, based on synthetic MIMO data, involves the following steps:

- Generate the ground truth FRF matrix over a defined time and frequency grid.
- Simulate random broadband Gaussian excitation signals with flat power spectral density.
- Compute system responses using the known FRF.
- Estimate FRFs from simulated excitation-response data using the implemented H1 estimator.
- Compare estimated FRFs against the ground truth in both magnitude and phase.

## 5 Results and Discussion

The synthetic MIMO test confirms the estimator's correctness under controlled conditions. The estimated FRFs closely match the true system response, both in amplitude and phase. Tests like this were added for both the $H_1$ and $H_2$ FRF estimators.

Importantly, the implemented unit tests can highlight implementation pitfalls, such as inconsistencies in signal processing conventions or incorrect matrix dimensions. The goal is to detect such issues early through disciplined use of unit tests and rigorous validations of both numerical magnitude and phase, before pushing code updates to the end users.

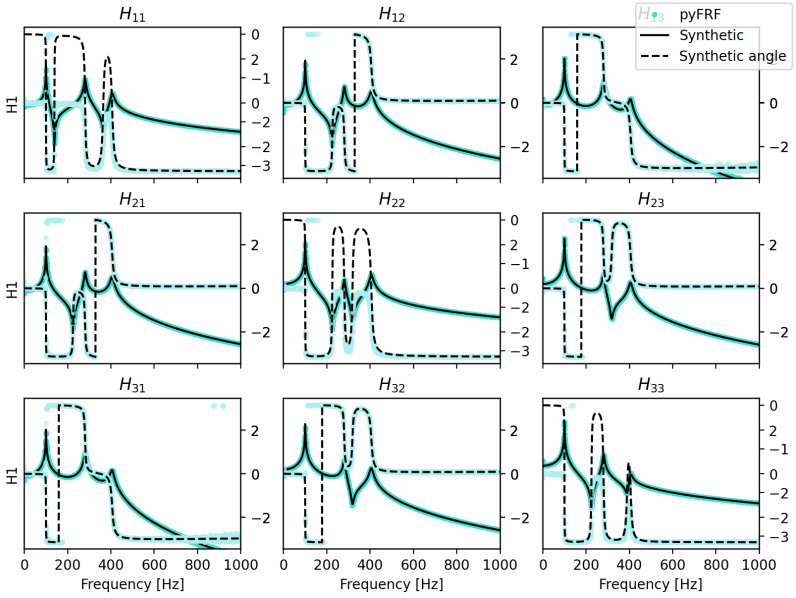



**Fig. 1:** Comparison of estimated and true FRFs (magnitude and phase) for multiple DOFs.

## 6 Conclusion

This study demonstrates the application and validation of the $H_1$ estimator for FRF computation in a synthetic MIMO context. Beyond correctness of results, the process underscores three key points:

1. **Mathematical consistency** is critical. Even minor differences in conventions (e.g., order of conjugation in cross-spectral computations) can produce significant effects like phase inversions.

2. **Programming discipline** is necessary to prevent and catch errors. This includes proper documentation, clear separation of assumptions, and adherence to conventions across modules.

3. **Unit testing** is an indispensable tool in signal processing codebases. Rigorous tests not only catch bugs but also clarify behavior under edge cases and help maintain trust in computed results.

As computational modal analysis continues to evolve, integrating strong software engineering practices into numerical research workflows becomes essential for ensuring the reliability and reproducibility of results, particularly in the open-source community.

## References

[1] N. M. M. Maia and J. M. M. Silva, *Theoretical and Experimental Modal Analysis*. Research Studies Press, 1997.

[2] P. Virtanen, R. Gommers, T. E. Oliphant *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 3 2020.

# SoftPEAR: On Constructing Finite Element Models from Irreducible Element Models

T. Gowdridge[1], C. O'Higgins[2], K. Worden[1], and D.S. Brennan[1]

[1]Dynamics Research Group, School of Mechanical, Aerospace, and Civil Engineering, The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK
[2]School of Natural and Built Environment, Queen's University of Belfast, Stranmillis Road, Belfast, BT9 5AG, UK

**Abstract**

Population-Based Structural Health Monitoring (PBSHM) is an emerging technique that leverages data from populations of structures to enhance understanding of an individual structure via knowledge transfer from others. Irreducible Element (IE) models, a technique for representing real-world structures as attributed graphs, have demonstrated utility in identifying topological similarities between structures within a population-based framework. However, the interpretability of these models can diminish when represented through nested JSON-based structures comprising thousands of lines.

This work presents novel software that translates JSON-based IE models into fully functional finite element models within the bridge analysis software LUSAS. This integration allows users to load, manipulate, and analyse models according to specific requirements, closing the gap between abstract data representations used for finding similarities within the population and practical engineering applications used for simulating structural responses under loading.

The development of this tool marks a crucial step in establishing the first simulated population-based structural dynamics database, known as the Population-based SHM Engineered Asset Resource (PEAR). PEAR aims to generate semi-realistic structural models and associated dynamic data to support the advancement of PBSHM methodologies. By facilitating seamless model generation and analysis, this work contributes to improving the accessibility and interpretability of population-based approaches in structural dynamics, ultimately promoting PBSHM innovation.

**Keywords:** PBSHM, Irreducible-Element Models, Finite-Element Models, Simulation

# 1 Introduction

The need for this work arises from a growing shift towards Structural Health Monitoring (SHM) from a population-based perspective [1, 2, 3, 4, 5]. In the field of Population-Based SHM (PBSHM), significant research has focussed on graph matching algorithms [6], geodesic-based flows [7], and transfer learning techniques [8]. However, these studies have relied on their own datasets, which introduces certain limitations.

The software presented in this work, along with accompanying papers [9, 10], outlines a method for generating a standardised benchmark dataset to support the testing and devlopment of population-based approaches. To achieve this, families of structures are generated in their Irreducible Element (IE) model form [5], with varying topologies and geometries based on realistic engineering principles [9]. While the IE models, expressed in JSON format, are well suited for programmatic generation and algorithmic processing, they are not directly usable for generating Finite Element (FE) models or running simulations.

The software developed for this work maps IE models to their corresponding FE models; extending what can be expressed through the structural representation. Although both IE and FE models describe the same underlying structure, they serve different purposes. IE models enable easy programmatic generation and graph-based analysis but lack detail regarding structural response and behaviour. Conversely, FE models provide detailed structural insight but do not support the generative and the abstract nature required for similarity comparisons. Therefore, creating a validated and reliable mapping between the two structure representations allows both sets of advantages to be leveraged.

The dual IE and FE representation of a structure offers form-specific benefits. The primary motivation for this work is the creation of a population-based dataset, internally referred to as the Population Engineered Asset Resource (PEAR). Inside PEAR all data regarding IE models, and the structural response and parameters simulated via the FE model are stored, allowing a large, high-information database, with the intent of catalysing the development and validation of population-based technologies

Although the original aim of the project was to create a tool for generating FE models from programmatically defined IE models, the resulting software is more versatile. Other very natural uses of the IE to FE conversion software may also include:

1. Creating a coarse FE representation of a real-world structure from a previously curated IE model. The resulting FE model can then be simulated under different load cases to support information transfer.

2. Providing a validation step to ensure an IE model is correctly defined, as intuition can be shrouded in the JSON structure.

The remainder of the paper follows the chronology of the software's operation, highlighting key design features. It concludes with results from the 2000 generated structures and the validation steps applied.

## 2 Software Chronology

The main walkthrough of this software is taking the IE model in JSON form, extracting and extrapolating information for each element, rearranging into a form more readily applicable for FE APIs.

Following this preprocessing, as outlined in a more engineering-focused sister paper [9], the simulated dataset is partitioned into populations and subclasses. A structure identifier of a certain form is a required feature of the PEAR dataset, where each IE JSON file name must match a predefined regular expression with sufficient information to identify its population and subclass, as shown in Fig. 1.

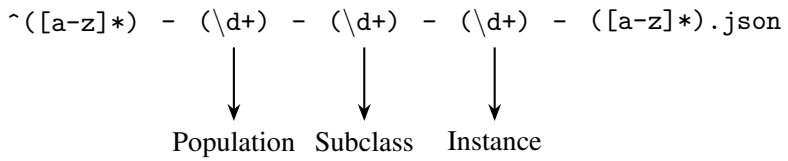



**Fig. 1:** Extraction regex for the structure identifier.

Based on the extracted structure identifier, the appropriate instance of the FE software is invoked. This includes specific FE construction logic tailored to the structure type, ensuring that the resulting FE models are both realistic and grounded in sound engineering principles. For the purposes of this paper, only the bridge population is discussed, specifically the beam-and-slab (`1-1-x`) and ladder deck (`1-2-x`) subclasses, which are simulated and assessed.

Once invoked, the software initiates a new LUSAS [11] project and proceeds to create the model wireframe. It then assigns essential attributes such as material properties, geometric profiles, according to the rules defined by the subclassed model, and ground conditions. Once the model geometry is fully defined, the structure is meshed.

Subsequently, predefined scenario scripts, scripts used to automatically modify the FE models within LUSAS, are used to apply loading conditions within LUSAS, simulating relevant load cases. Once simulations are complete, both the data and models are saved, concluding the pipeline.

The following subsections provide a more detailed breakdown of the key modelling stages and their respective nuances.

### 2.1 Preprocessing step

Different FE software packages offer specific advantages. For example, LUSAS is particularly suited to modelling bridges, with extensive libraries and features tailored in this domain. Other software packages, such as ABAQUS, ANSYS, and SolidWorks, offer different specialisations.

To support generalisation across different FE software packages, the IE model information is first extracted into a generalised Python representation. Whilst the IE model form is well defined [5], it needs to be transformed into an abstract representation applicable to any FE software package; in this case, LUSAS.

The need for this Python preprocessing layer is that the IE model form is very much created with the intent processing data in a graph domain, so is more concerned about capturing the essence of the structure, and less concerned about the finite geometrical details. However, The FE model is more geared towards the geometry of the elements, and thus the coordinates defining them. Therefore, each coordinate defining an element's geometry are to be extracted from the localised coordinate system of

```
{
        "name": "deck-1"
        "element_type": "regular",
        "isground": False,
        "coordinates": [(0,0,0), (0,1,0), (1,1,0), (1,0,0)],
        "geometry": "Surface",
        "ground_coordinate": (),
        "profile": {
                "thickness": 0.1,
                "width": 1,
                "profile": ("rectangular", "plate")
        },
        "material": ("concrete", {
                "youngsModulus": 35e9,
                "poissonsRatio": 0.2,
                "density": 2400,
                "linearThermalExpansionCoefficient": 1e-5
            }
        )
}
```

**Fig. 2:** Example element, `deck-1`, in the Python preprocessing step.

the IE models. Thus the preprocessing obtains the information in a more elementwise intrinsic form, where all information is contained within one element.

There are several non-trivial aspects involved in extracting IE model information into the tabulated Python format. For instance, both the relationships and elements sections of the IE model provide only a single coordinate. In the elements section, this coordinate represents the origin of the bounding box, while in the relationships section, it denotes the point at which two elements are joined. However, when constructing line, surface, or volume elements in the FE software, the full set of points defining the geometry is required. This means all relevant geometric information must be extracted from the IE model in a form that can be passed into the FE software.

The preprocessing step transforms the IE model into a tabulated Python dictionary, indexed by element names. Each dictionary entry contains a value resembling a C-style struct, also represented as a Python dictionary, as shown in Fig. 2. Whilst this form is largely a reordering of the IE information, there is also extrapolated information, this being the coordinates defining each element's geometry. Preprocessing at this stage ensures standardisation across information being passed into the choice FE software.

To avoid issues with rounding and calculation errors, the wireframe for the preprocessed Python layer is first extracted from the relationships section of the IE model. In cases where two line geometries, or more commonly a line and a surface, are connected, the relationship coordinate is given as the centre of the line span. The line's endpoints then need to be extrapolated along its length, which is fully solvable using the information in the relationships section.

However, there are cases where the relationships section does not provide sufficient information. This occurs when two surface elements, or higher-dimensional elements, are connected. Their re-

lationship is defined by a single point, but there are two or three axes along which extrapolation is required. In such cases, the elements section must be used to infer the edge points of the geometry based on the shape's defined properties.

## 2.2 LUSAS Instantiation

The bulk of the LUSAS logic is handled by a base class, created to handle core functionality such as launching a LUSAS session, saving and loading models, and managing interactions with the FE API, as well as more general aspects of model construction, such as generating the structure's wireframe, assigning materials, and adding ground elements. This base logic is then inherited by child classes, which implement the specific geometric rules required to accurately construct the FE model for each structure type.

The first step that the base class performs is the instantiation of the LUSAS caller, from this the base class then sets up a LUSAS database instance used for saving the model, at this point some global variables are set such as the axis orientation, model dimensions and units.

## 2.3 Wireframe

During the development of this software, it became clear that a significant portion of the runtime was spent on API calls. Therefore, when creating the wireframe, the higher-order geometries were created first; in the order of volumes, surfaces, lines, then points. This approach was chosen because lower-dimensional elements are often likely to be faces of higher-dimensional ones. For instance, creating a single cube volume also generates six surface elements, twelve line elements, and eight point elements. At the time of creation of each wireframe element, its and all of the children geometries' LUSAS-based name and coordinates are stored in the Python-side hashmap for fast $O(1)$ lookups. Then, when creating the lower-order elements, the software first checks whether they already exist in the Python dictionary. If they do exist, the element name is retrieved via lookup, significantly speeding up model generation, effectively bypassing the costly LUSAS API in many cases.

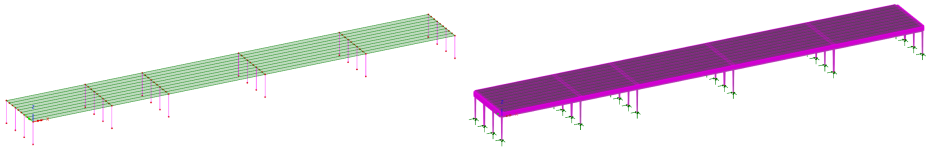


**Fig. 3:** Beam and Slab Models

There are two important orientation-related issues that must be addressed at this stage: the direction of surface normal vectors, and the orientation of the bridge deck perimeter.

When defining surface elements, the lines connecting successive input points must not intersect. To ensure no line intersection, the software assumes each surface forms a convex hull and uses an algorithm that successively selects the nearest point from the coordinate set until all points are exhausted. While this provides a solution, it does not preserve a consistent orientation; cycles around the surface perimeter may proceed either clockwise or anticlockwise.

This inconsistency causes issues later, particularly when applying surface eccentricities to shift surfaces to their intended locations. Surface normals are determined by the ordering of points, and may therefore point in either direction. Since surfaces are expected to have normals in the positive Z


**(a)** UM **(b)** YE

**Fig. 4:** UM and YE beams showing their nonsymmetric profiles.

direction, the software identifies any downward-facing surfaces and flips them to the correct orientation.

A second issue arises with non-symmetric precast beam elements, specifically those from the UM and YE families, as shown in Fig. 4, where it is required that the flat side faces outward. The software accounts for this by adjusting the orientation based on the specific beam type to ensure this condition is satisfied.

## 2.4 Software subclassing for geometry specifics

The authors recognise that there will never be a one-size-fits-all solution for generating all structures from the IE models. There are simply too many modelling nuances that fall outside the scope of the information captured by IE models. Therefore, FE construction classes have been developed, each tailored to a specific structure type. This section outlines some of the differences in modelling approaches across the broader populations, with a focus on the beam-and-slab and ladder deck bridge subclasses.

When generating bridge models, as previously discussed, the highest-order geometries are created first; in this case, the surface elements of the bridge deck. The supporting beams are then derived from the faces of these surfaces. Initially, all elements are created in a planar arrangement; however, for a realistic structural configuration, the bridge deck must sit above and be flush with the supporting beams. Therefore, the FE model requires shifting the deck upwards by the height of the cross/main beams. Such domain-specific engineering logic has been explicitly embedded in the IE-to-FE bridge conversion software and is inherited by the respective subclass models, enabling the most faithful representation of structures.

Beyond bridges, the PEAR database currently includes two structural populations: wind turbines and masts. Each of these, and their respective subclasses, demand tailored FE modelling assumptions to produce realistic results. This portion of the software is designed specifically to embed domain-specific structural knowledge to most accurately realise the FE simulations.

### 2.4.1 Bridges - Beam and Slab

The beam-and-slab configuration serves as the default logic case, requiring minimal specialised handling. In this representation, only the previously discussed upward shifting of the bridge deck is necessary to ensure an accurate structural model.

### 2.4.2 Bridges - Ladder Deck

Ladder deck bridges comprise longitudinal main beams and transverse cross beams. These beam types are parameterised with different heights, with cross beams being shorter than the main beams. Therefore, the cross beams must be shifted upward so that their top surfaces align flush with the underside of the bridge deck; ensuring modelling realism.

## 2.5 Meshing

The meshing philosophy begins by defining the mesh on the lowest-dimensional geometry and then propagates it to higher-dimensional elements. In the case of bridges, a line mesh is defined first, using an element spacing of 0.5 metres to avoid generating elements with large aspect ratios. The mesh intervals on these lines are then inherited by surface elements to form a regular 2D grid, which in turn propagates to volume elements, giving a structured 3D mesh.

## 2.6 Scenarios

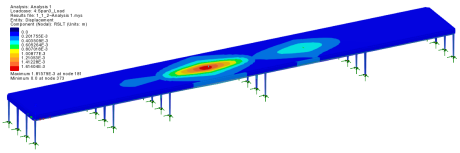


**Fig. 5:** Simulation of a point load applied to the third span of a five span beam-and-slab bridge.

For the bridge models, two load case categories are applied: the self-weight (dead load) of the bridge structure, and a vertical point load of 40 kN applied at the centre of each bridge span.

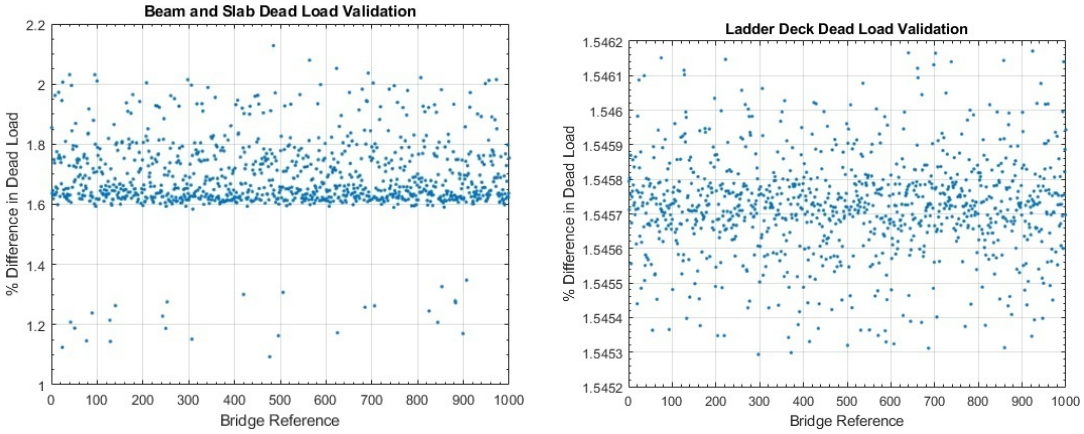
From a software perspective, these load cases are implemented through programmatically generated scenario scripts, which are directly callable within LUSAS. Additionally, these loadcase scripts can be defined independently of the software and FE models, allowing for user-tailored loading scenarios. Once the load cases are applied, the simulation proceeds, and the resulting structural deflections are computed and extracted at each node in the FE mesh.

# 3 Validation

To ensure that the generated FE models are well behaved, two validation checks are performed by comparing the FE output to expectations derived from the corresponding information in the IE models. These checks evaluate both the structural mass and the deformation response of the bridge.

The first validation check assesses the total dead load of the bridge. This check is done by computing the total mass of all individual elements in the IE model, estimated as the product of each element's length, cross-sectional area, and material density. This total is then compared to the sum of the vertical reaction forces at the supports obtained from the FE simulation. While the IE-based mass may slightly overestimate the true mass, because of potential overlaps between structural elements, this discrepancy is typically small and does not compromise the validity of the check.

From the mass validation results shown in Fig. 6, it appears that the ladder-deck error is consistently smaller, with much less variation. This improvement is likely the more accurate estimation of beam section properties. For beam-and-slab bridges, many of the beam types used, such as Y, YE,




**(a)** Beam-and-slab **(b)** Ladder-deck

**Fig. 6:** Bridge mass validation errors for two bridge types.

M, and UM sections, are nonparametric, meaning their exact geometric properties used by LUSAS and the validation checks may have varied, leading to less precise mass estimates. In contrast, for ladder deck bridges, the exact parametrisations of the beam sections are known and used directly in the calculations. This information results in more accurate mass estimates and significantly reduces both the magnitude and variability of the validation error.

The second validation check compares the simulated vertical deflection to that predicted by beam bending theory. Each bridge span is assumed to be a simply supported beam, and a point load is applied to its centre. The resulting deflections from the FE model are then compared to analytical solutions, allowing for a sanity check on the mechanical behaviour of the model.

# 4 Conclusion

This paper introduced a software tool that bridges the gap between abstract IE models and executable FE representations, enabling the automated translation of JSON-based data structures into fully functional LUSAS models. This development not only improves the interpretability and practical utility of IE models, but also serves as a critical component in the simulation pipeline of the PEAR dataset. By enabling large-scale, consistent FE model generation, the software supports the evaluation and curation of PBSHM methodologies.

Looking ahead, future work will focus on expanding the range of structural populations and subclasses supported by the tool, to further populate and diversify the PEAR dataset. Additionally, efforts are underway to integrate this software into the broader PBSHM Framework currently being developed at the University of Sheffield, where data will be generated and uploaded seamlessly into the database.

# 5 Acknowledgements

The authors would like to gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) via grant reference EP/W005816/1. For the purposes of open

## References


[1] L. A. Bull, P. A. Gardner, J. Gosliga, T. J. Rogers, N. Dervilis, E. J. Cross, E. Papatheou, A. E. Maguire, C. Campos, and K. Worden, "Foundations of population-based SHM, part i: Homogeneous populations and forms," *Mechanical Systems and Signal Processing*, vol. 148, p. 107141, 2021.

[2] J. Gosliga, P. A. Gardner, L. A. Bull, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part ii: Heterogeneous populations–graphs, networks, and communities," *Mechanical Systems and Signal Processing*, vol. 148, p. 107144, 2021.

[3] P. Gardner, L. A. Bull, J. Gosliga, N. Dervilis, and K. Worden, "Foundations of population-based SHM, part iii: Heterogeneous populations–mapping and transfer," *Mechanical Systems and Signal Processing*, vol. 149, p. 107142, 2021.

[4] G. Tsialiamanis, C. Mylonas, E. Chatzi, N. Dervilis, D. Wagg, and K. Worden, "Foundations of population-based SHM, part IV: The geometry of spaces of structures and their feature spaces," *Mechanical Systems and Signal Processing*, vol. 157, p. 107692, 2021.

[5] D. S. Brennan, J. Gosliga, E. J. Cross, and K. Worden, "Foundations of population-based SHM, part v: Network, framework and database," *Mechanical Systems and Signal Processing*, vol. 223, p. 111602, 2025.

[6] D. S. Brennan, T. J. Rogers, E. J. Cross, and K. Worden, "On calculating structural similarity metrics in population-based structural health monitoring," *Data-Centric Engineering*, vol. 6, p. e24, 2025.

[7] T. Dardeno, L. A. Bull, N. Dervilis, and K. Worden, "Transfer learning via intermediate structures," *e-Journal of Nondestructive Testing*, vol. 2024, July 2024.

[8] T. Gowdridge, J. McCulloch, J. Poole, T. J. Rogers, K. Worden, and D. S. Brennan, "Embedding an initial transfer-learning technology within a population-based SHM framework," *e-Journal of Nondestructive Testing*, vol. 2024, July 2024.

[9] C. O'Higgins, T. Gowdridge, D. Hester, K. Worden, and D. S. Brennan, "Advancing PEAR: Development of a bridge benchmark datasets for pbshm research," *SHMII*, 2025.

[10] C. O'Higgins, T. Gowdridge, D. Hester, K. Worden, and D. S. Brennan, "Population-based structural health monitoring engineered asset resource (PEAR): A benchmark PBSHM dataset," *Structural Health Monitoring*, 2025.

[11] LUSAS, "LUSAS Bridge Analysis Software," 2025. `https://www.lusas.com/products/bridge/`.

# Prediction of Cutting Edge Wear Using Machine Learning

Miha Kodrič Jure Korbar Miha Pogačar Gregor Čepon *

*University of Ljubljana, Faculty of Mechanical Engineering*

**Abstract**

Sustainable machining demands efficient tool condition monitoring (TCM) to maximize tool utilization and reduce environmental impact. Existing TCM solutions range from high-cost multi-sensor systems to ultra-low-cost alternatives with limited accuracy. This research bridges the gap with a resource-efficient, standalone TCM system for on-site tool wear estimation. The system integrates a PVDF-based accelerometer, Raspberry Pi 4, and a data acquisition card. A multi-level software architecture is designed to fully leverage this hardware, optimizing real-time signal processing while supporting both machine learning model training and inference. The proposed method employs two elementary models: $k$-means clustering for machining phase segmentation and ridge regression for tool wear estimation. A case study on an industrial lathe established a linear correlation between tool wear and surface roughness. Data from three tool inserts over their lifetimes proved sufficient for training machine learning models, achieving promising prediction accuracy. This research advances standalone TCM solutions tailored for manufacturing sectors seeking a balance between affordability and performance.

**Keywords:** tool condition monitoring, surface roughness, tool vibration, turning, tool used life

## 1 Introduction

The growing demand for sustainable, high-precision machining highlights the persistent challenge of efficient tool management. Traditional strategies replace inserts at fixed intervals—often 50–80% of their expected life—without accounting for actual wear [1], leading to premature replacements or unexpected failures that can cause up to 20% of downtime [2]. These inefficiencies have driven interest in tool condition monitoring (TCM) systems [3, 4].

Flank wear, the most commonly monitored wear mechanism due to its impact on surface finish, is typically tracked using indirect methods (e.g., vibration, force, or temperature) when direct observation is impractical [7, 6]. Such systems generally combine sensor data with AI models to estimate tool wear [8].

However, a clear gap remains for low-cost TCM solutions that use minimal sensors and standard hardware—features vital for SMEs. To address this, we propose a compact TCM system that estimates used life (UL[1]) using a single uniaxial PVDF accelerometer. The setup includes a Raspberry Pi 4 and

*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si

[1] UL is defined as the percentage of a tool's expected life that has already been consumed based on the number of parts produced. Surface roughness is used as the tool wear benchmark.

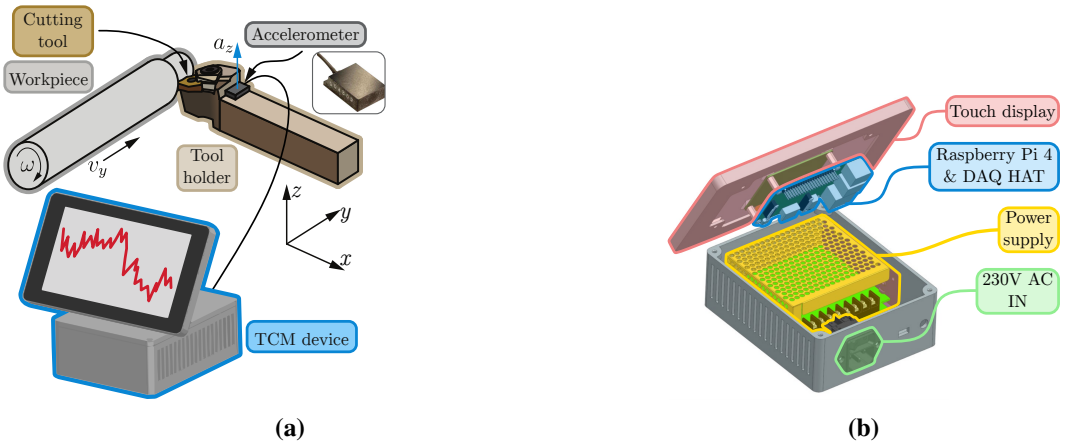
Digilent MCC 172 DAQ, with a layered software stack handling real-time acquisition, preprocessing, and local storage—eliminating the need for cloud infrastructure.

The system employs k-means clustering to identify machining stages and ridge regression for life estimation. Validated on an industrial lathe across three tool edges, it showed strong correlation between estimated tool life and surface roughness. This approach balances cost, simplicity, and performance, making real-time TCM accessible for a broader range of machining environments.

## 2 TCM Hardware Setup

This study proposes a cost-effective, standalone TCM hardware system that prioritizes simplicity, performance, and industrial usability. To reduce complexity, a single-sensor approach is adopted, focusing on vibration monitoring. Vibration signals offer reliable tool condition indicators and allow for easy sensor mounting without modifying the tool holder—unlike force sensors.

A uniaxial PVDF-based accelerometer is used due to its high-frequency sensitivity, low noise floor, and affordability compared to conventional IEPE sensors. The sensor is mounted directly on the tool holder, measuring vibrations in the cutting direction ($z$-axis), as illustrated in Fig. 1a.




**(a)** **(b)**

**Fig. 1:** TCM device: (a) schematic representation of the TCM device connected to the accelerometer during turning; (b) rear view of the TCM device model without housing top.

The processing unit consists of a Raspberry Pi 4, chosen for its balance of cost and computational capability. Vibration signals are acquired using a Digilent MCC 172 DAQ HAT, offering two channels and a sampling rate of 51.2 kHz. A Raspberry Pi Touch Display is included to support real-time monitoring and local control. A custom graphical interface enables measurement control, signal visualization, model training, and tool life tracking.

## 3 Methodology for Tool Used Life Estimation

This section outlines the methodology for estimating the used life of a cutting tool using vibration signals and machine learning. The approach was validated through a case study on an industrial lathe in serial production. A high-level overview of the process is shown in Fig. 2.

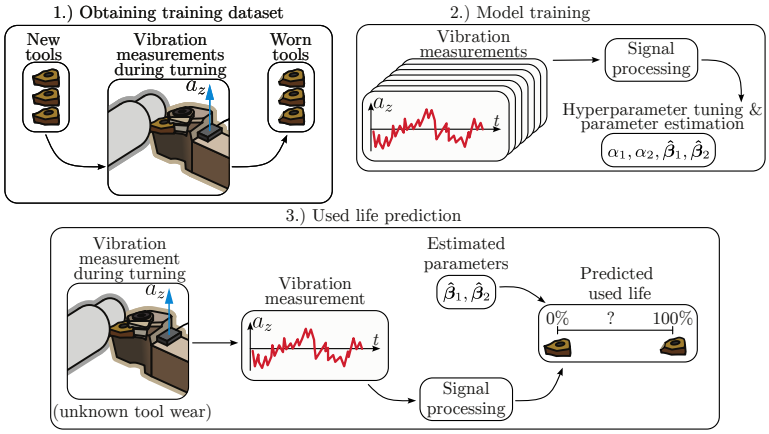
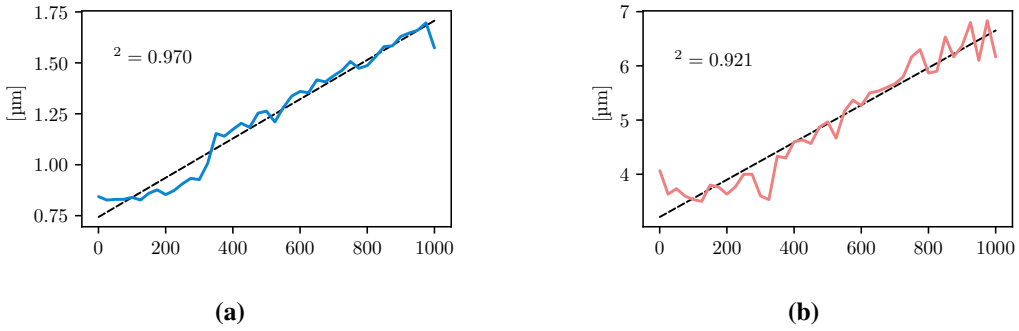



**Fig. 2:** Flowchart describing the use of vibration measurements for used life estimation.

## Surface Roughness and Tool Life

In a preliminary study, surface roughness was tracked over the full life of a cutting tool. Measurements of $R_a$ and $R_z$ were taken every 25 parts using a Mitutoyo SJ 400. As shown in Fig. 3, both parameters showed a strong linear relationship ($R^2 > 0.9$) with the number of produced parts. Based on this, tool life is treated as a linear function of the number of parts, and roughness measurements were omitted in later experiments.




**(a)** **(b)**

**Fig. 3:** Surface roughness vs. number of produced parts: (a) $R_a$, (b) $R_z$.

## Signal Processing and Operation Clustering

During actual production, vibrations were recorded for four tools (from new to worn) at 51.2 kHz. Fig. 4 shows an example time series, where active cutting is marked by high-amplitude regions. These were extracted using a simple amplitude threshold. To differentiate between multiple cutting operations per part, the signal segments were grouped using $k$-means clustering. Each cluster corresponded to a distinct operation ($s_1$, $s_2$), enabling operation-specific analysis.

Each vibration segment was transformed into the frequency domain using the discrete Fourier transform, limited to 10 kHz. Spectra were averaged into 10 Hz bins and normalized, resulting in fixed-length feature vectors. These were labeled with the relative tool life (ratio of produced parts to total tool life).

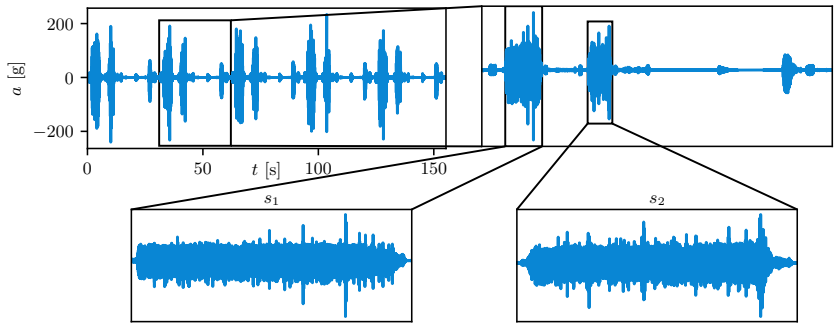



**Fig. 4:** Isolation of vibration time series for $s_1$ and $s_2$ operations.

**Used Life Estimation via Ridge Regression**
Tool life was predicted using ridge regression, chosen for its robustness to high-dimensional data and regularization properties. A separate model was trained for each operation using grid search and cross-validation to optimize the regularization parameter. Model training and evaluation were implemented using scikit-learn [9]. To reduce variability in predictions, the final used life estimate was computed as a moving average over the last five parts.

## 4 Results

Machine learning models were trained on vibration data from three tools and tested on a fourth. Separate models were used for two turning operations ($s_1$ and $s_2$). Initial estimates for each operation captured the wear trend but were noisy, but combining averaging process on $s_1$ and $s_2$ results and applying a moving average with 5 samples significantly improved prediction quality (Fig. 5). An alert threshold at 80% of the tool life ensures timely replacement. Ridge regression enables fast, real-time estimation with minimal data, showing strong correlation with surface roughness as a wear indicator.

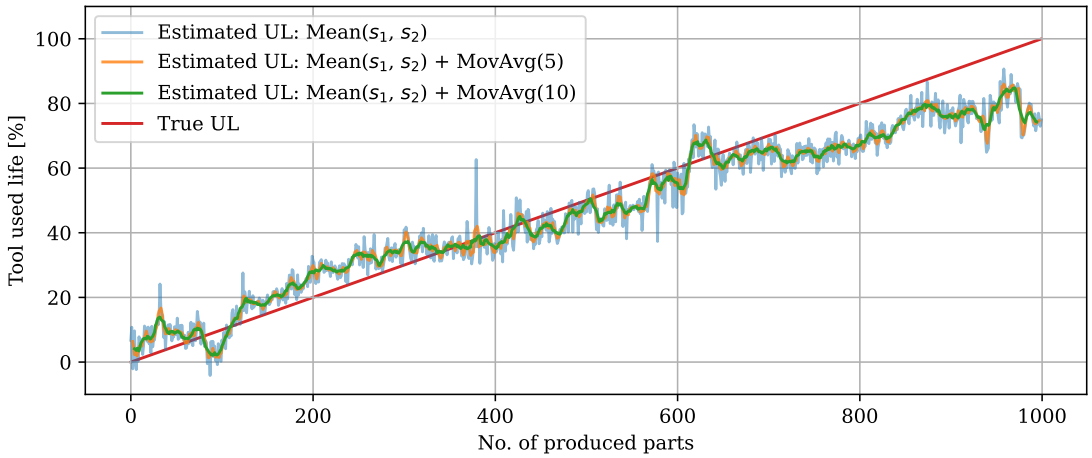



**Fig. 5:** Tool used life estimation with moving average.

## References


[1] H. Wiklund, "Bayesian and regression approaches to on-line prediction of residual tool life," *Quality and Reliability Engineering International*, vol. 14, no. 5, pp. 303–309, 1998.

[2] S. Kurada and C. Bradley, "A review of machine vision sensors for tool condition monitoring," *Computers in industry*, vol. 34, no. 1, pp. 55–72, 1997.

[3] T. Mohanraj, S. Shankar, R. Rajasekar, N. Sakthivel, and A. Pramanik, "Tool condition monitoring techniques in milling process—a review," *Journal of Materials Research and Technology*, vol. 9, no. 1, pp. 1032–1042, 2020.

[4] S. Swain, I. Panigrahi, A. K. Sahoo, and A. Panda, "Adaptive tool condition monitoring system: A brief review," *Materials Today: Proceedings*, vol. 23, pp. 474–478, 2020.

[5] C. Nath, "Integrated tool condition monitoring systems and their applications: a comprehensive review," *Procedia Manufacturing*, vol. 48, pp. 852–863, 2020.

[6] M. Bhuiyan, I. Choudhury, and M. Dahari, "Monitoring the tool wear, surface roughness and chip formation occurrences using multiple sensors in turning," *Journal of Manufacturing Systems*, vol. 33, no. 4, pp. 476–487, 2014.

[7] T. Mikołajczyk, K. Nowicki, A. Kłodowski, and D. Y. Pimenov, "Neural network approach for automatic image analysis of cutting edge wear," *Mechanical Systems and Signal Processing*, vol. 88, pp. 100–110, 2017.

[8] L. Colantonio, L. Equeter, P. Dehombreux, and F. Ducobu, "A systematic literature review of cutting tool wear monitoring in turning by using artificial intelligence techniques," *Machines*, vol. 9, no. 12, p. 351, 2021.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

# LDAQ: A Lightweight, Python-Based Toolkit for Modular Data Acquisition in Structural Dynamics

Gašper Krivic Tilen Košir Klemen Zaletelj Janko Slavič *

*University of Ljubljana, Faculty of Mechanical Engineering*

**Abstract**

Experimental structural dynamics frequently depends on proprietary, inflexible data acquisition and signal generation systems that are difficult to integrate with custom hardware. These limitations hinder the adaptability of experimental workflows and complicate reproducibility and automation, particularly in academic and prototyping contexts.

To overcome these challenges, the Lightweight Data Acquisition (LDAQ) toolkit has been developed as an open-source Python framework for modular signal acquisition, generation, and real-time visualization. LDAQ supports various hardware platforms, including National Instruments devices, serial microcontrollers, and visual and thermal cameras. Its modular, object-oriented architecture enables entire measurement workflows to be defined in code, device interfaces to be customized, new hardware to be integrated with minimal effort, and real-time signal transformations to be applied during acquisition.

A representative use case is presented in which LDAQ is employed for the dynamic characterization of a structure through synchronized signal generation and acquisition, with real-time visualization in both time and frequency domains. This example demonstrates the capability of LDAQ to replicate typical commercial measurement workflows within a flexible, transparent, and scriptable environment.

By providing a customizable, code-based alternative to proprietary tools, LDAQ lowers the barrier to implementing reproducible and adaptable experiments in structural dynamics. It integrates seamlessly with the broader Python scientific ecosystem and promotes open-source practices in experimental mechanics.

**Keywords:** Data Acquisition, Open Source, Structural Dynamics, Python

## 1 Statement of Need

Experimental structural dynamics frequently depends on proprietary and inflexible data acquisition systems, which constrain flexibility, integration, and reproducibility. Commercial tools are often closed-source, costly, and difficult to adapt to evolving research requirements—particularly when custom sensors or novel test configurations are involved.

The Lightweight Data Acquisition (LDAQ) toolkit addresses this gap by providing an open-source Python framework for modular signal acquisition, generation, and real-time visualization. It supports a range of hardware platforms—including National Instruments devices, serial microcontrollers, and thermal cameras—and enables complete experimental workflows to be defined and controlled in code.

*Corresponding author, Email address: janko.slavic@fs.uni-lj.si

LDAQ offers a flexible and extensible platform for the development of customizable and reproducible measurement workflows, making it well-suited to diverse structural dynamics applications.

# 2 Overview of Acquisition, Generation & Visualization Capabilities

LDAQ implements a unified and extensible Python framework for signal acquisition and generation in structural dynamics experiments. Its architecture is composed of modular, object-oriented components that facilitate the configuration of custom workflows across heterogeneous hardware devices. Designed to be lightweight yet powerful, LDAQ separates core functions into acquisition, generation, and visualization modules, which are integrated within the `LDAQ.Core` Python class. Additional information is available in the GitHub repository [1] and the accompanying documentation [2].

## 2.1 Acquisition Modules

LDAQ supports a range of acquisition sources, each implemented as a dedicated Python class:

- National Instruments (NI): High-performance data acquisition via the NI-DAQmx driver, implemented in `LDAQ.national_instruments.NIAcquisition`.
- Digilent Analog Discovery: Portable, cost-effective analog acquisition, implemented in `LDAQ.digilent.WaveFormsAcquisition`.
- Serial Communication Devices: Data acquisition from Arduino, ESP, or custom sensors via USB/serial protocols, implemented in `LDAQ.serial_communication.SerialAcquisition`.
- Basler and FLIR Cameras: Visual and thermal imaging supported through vendor SDKs, implemented in `LDAQ.basler.BaslerCamera` and `LDAQ.flir.FLIRThermalCamera`.

Each acquisition device is encapsulated within a Python class that defines its specific acquisition method. Custom acquisition classes can also be implemented by subclassing `LDAQ.acquisition_base.BaseAcquisition`, which serves as a template. Abstract methods in this base class must be overridden to enable data acquisition from new sources.

LDAQ provides built-in support for acquisition triggering, allowing measurements to commence automatically based on predefined signal conditions on a specified channel. The triggering mechanism is configurable per acquisition object and facilitates event-based data collection in applications such as impact testing or synchronized multi-device capture.

## 2.2 Generation Modules

Signal generation is primarily supported through National Instruments hardware, implemented in the `LDAQ.national_instruments.NIGeneration` class. Excitation signals are defined as arbitrary waveforms using standard Python tools (e.g., NumPy arrays [3]), which are passed to LDAQ for buffered output and trigger control.

Custom generation modules can be implemented by subclassing `LDAQ.generation_base.BaseGeneration`, following the same design pattern as the acquisition modules. Acquisition and generation components can be executed concurrently within a single `LDAQ.Core` instance, enabling tightly synchronized excitation–response experiments.

### 2.3 Visualization Modules

Real-time signal visualization is implemented in the `LDAQ.visualization.Visualization` class, which provides a flexible and efficient interface for displaying acquired or computed signals during runtime. The visualization system is built on top of the high-performance pyqtgraph [4] library, enabling smooth updates even for high-frequency data streams.

Multiple subplots arranged in a grid layout are supported. Each subplot can display one or more signal traces, linked to individual data channels—either raw input or computed virtual channels. Appearance parameters such as axis labels, line styles, plot limits, and update intervals are fully customizable.

In addition to raw time-domain signals, the visualization module supports real-time data transformations, including Fourier transforms and user-defined Python functions. These transformations are applied on-the-fly, enabling the monitoring of both physical signals and derived features during ongoing experiments.

## 3 Demonstration of a Representative Structural Dynamics Workflow Using LDAQ

This section demonstrates a representative experimental workflow in structural dynamics using LDAQ. The objective is to excite a structure with a logarithmic sine sweep, acquire acceleration signals from two sensors, visualize both time-domain and frequency-domain responses in real time, and save the results for post-processing.

The experimental setup is defined entirely in Python, as shown in Lis. 1, and consists of several modular steps. First, NI tasks for both acquisition and signal generation are created using LDAQ's high-level interface. Two acceleration channels are configured to capture the excitation and structural response within the acquisition object. A logarithmic sine sweep signal, ranging from 50 Hz to 5000 Hz, is generated and assigned to the output channel in the generation object. Real-time visualization is configured to display both acceleration signals, along with a frequency-domain representation via a live Fourier Transform plot in the visualization object, as illustrated in Fig. 1. Finally, the complete configuration is passed to the `LDAQ.Core` class, which executes a five-second measurement and stores the acquired data to disk for subsequent analysis. More detailed and advanced examples are available in the LDAQ documentation [2].

**Listing 1:** Python script demonstrating a LDAQ workflow.

```
import numpy as np
from scipy.signal import chirp

import LDAQ

# ----------------------------------------------------------------------
# Define Data Acquisition Task
task_acq = LDAQ.national_instruments.NITask('my_acqusition_task',
                                            sample_rate=25_600)
task_acq.add_channel('acc_shaker',
                     device_ind=1,
                     channel_ind=0,
                     sensitivity=10.17,
```

```
                        sensitivity_units='mV/g',
                        units='g')
task_acq.add_channel('acc_sample',
                        device_ind=1,
                        channel_ind=1,
                        sensitivity=9.595,
                        sensitivity_units='mV/g',
                        units='g')

# Initialize acquisition object
acq = LDAQ.national_instruments.NIAcquisition(task_acq,
    acquisition_name='my_NI_acq')

# ------------------------------------------------------------------
# Define Signal Generation Task
task_gen =
    LDAQ.national_instruments.NITaskOutput('my_generation_task',
                                                sample_rate=25_600)
task_gen.add_channel('exct_signal',
                        device_ind=2,
                        channel_ind=0,
                        min_val=-10,
                        max_val=10)

# Generate a logarithmic chirp signal
t = np.linspace(0, 1, task_gen.sample_rate)
signal = chirp(t, f0=50, t1=t[-1], f1=5_000, method='logarithmic')

# Initialize signal generation object
gen = LDAQ.national_instruments.NIGeneration(
     task_gen, signal=signal, generation_name='my_generation_task')

# ------------------------------------------------------------------
# Initialize visualization
vis = LDAQ.Visualization(refresh_rate=100)

# Configure visualization
# Add time-domain plots for both accelerometers
vis.add_lines(position=(0, 0), source='my_NI_acq',
    channels='acc_shaker')
vis.add_lines(position=(1, 0), source='my_NI_acq',
    channels='acc_sample')

# Add frequency-domain plot of the response acceleration
vis.add_lines(position=(3, 0),
                source='my_NI_acq',
                channels='acc_sample',
                function='fft')
vis.config_subplot((3, 0), axis_style='semilogy', t_span=1.0,
    xlim=(50, 5000))

```

```
# ------------------------------------------------------------------------
# Combine acquisition, generation, and visualization in a Core object
ldaq = LDAQ.Core(acquisitions=[acq], generations=[gen],
    visualization=vis)

# run the measurement for 5 seconds
ldaq.run(5)

# Save the measured data to a file
ldaq.save_measurement('my_measurement', root='.')
```

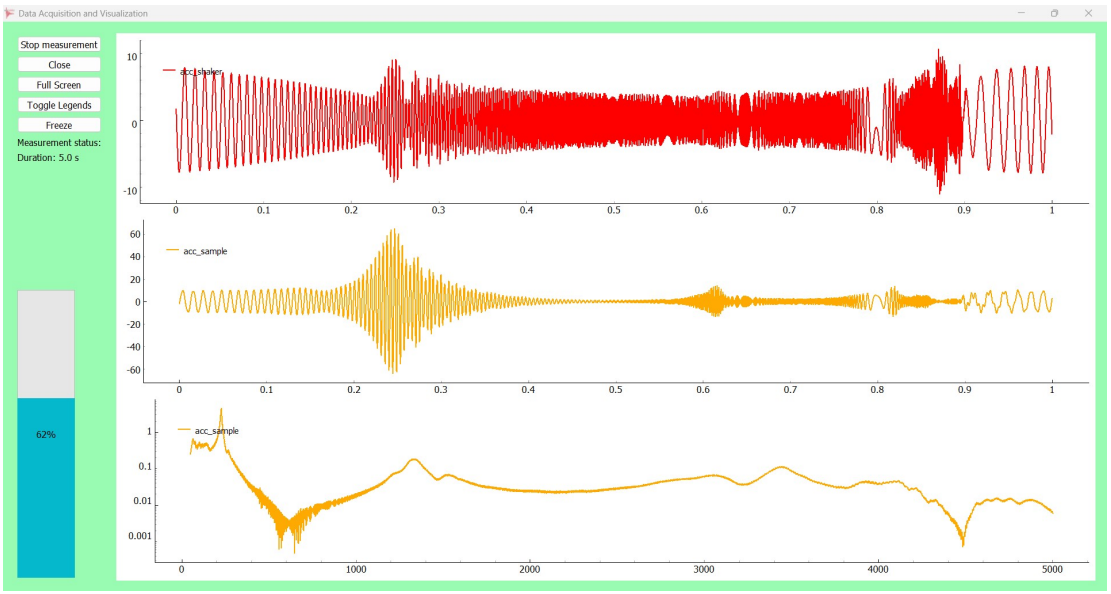



**Fig. 1:** Real-time visualization during the sine sweep test. The top two plots display the excitation and response signals, while the bottom subplot shows the magnitude of the Fourier transform of the response on a logarithmic scale.

## 4 Conclusion

LDAQ is a lightweight Python toolkit for data acquisition, signal generation, and real-time visualization, designed to support a broad range of structural dynamics experiments. It is compatible with various hardware platforms, and its modular architecture enables seamless adaptation to new measurement configurations. Experimental workflows can be fully defined and controlled in code, as demonstrated in the presented use case involving synchronized acquisition, generation, and real-time visualization.

As an open-source alternative to proprietary systems, LDAQ lowers the barrier to entry for customizable, scriptable experiments and integrates effectively with Python-based scientific computing tools.

# References


[1] T. Košir, K. Zaletelj, and J. Slavič, "Ldaq - streamlined data acquisition and generation." https://github.com/ladisk/LDAQ, 2022. Accessed: 2025-04-22.

[2] T. Košir and K. Zaletelj, "Ldaq documentation." https://ldaq.readthedocs.io/en/latest/index.html, 2022. Accessed: 2025-04-22.

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sep 2020.

[4] "Pyqtgraph: Scientific graphics and gui library for python." https://www.pyqtgraph.org/, 2021. Accessed: 2025-04-22.

# pyHarm: An Open Source Harmonic Balance Method Platform for Nonlinear Mechanical Dynamics

Quentin Mercier * Jason Armand

*Safran Tech, DST Department, Rue des Jeunes Bois, 78114 Magny-Les-Hameaux, France*

**Abstract**

pyHarm is a recently released open-source Harmonic Balance Method (HBM) solver designed for simulating nonlinear mechanical systems. Licensed under Apache 2.0 and distributed as a Python package, pyHarm aims to tackle Forced Response Frequency analysis of steady-state nonlinear mechanical systems. It leverages an harmonic balance solver enhanced with continuation methods to ensure robust and accurate results.

The underlying philosophy of pyHarm is to model mechanical systems as assemblies of elementary components (e.g. subtructures) and connectors (e.g. linear springs), allowing for independent evaluation of their contributions to the residual. The code has been mainly developed with flexibility in mind, trying to let the user define custom components easily based on the available abstract classes and using the built-in plugin system. On the performance side, the use of just-in-time compilation and autodifferentiation significantly increases computation speed.

pyHarm includes a range of physical phenomena such as gyroscopic effects, friction and gaps employing either the penalty method or Dynamic Lagrangian method (DLFT). It also offers advanced reduction techniques, a set of classical correction equations and prediction methods, as well as kinematic conditions.

In addition to concise user documentation and API documentation, pyHarm provides a set of tutorials in the form of Jupyter notebooks. These tutorials help users learn about the HBM method and classical nonlinear behaviors in mechanical dynamic systems.

The platform welcomes external contributions and is available at https://gitlab.com/drti/pyharm

**Keywords:** Harmonic Balance Method (HBM), Nonlinear Mechanical Systems, Forced Response Frequency Analysis, Open Source, python

## 1 Statement of Need

In the field of harmonic forced response predictions, the Harmonic Balance Method (HBM) has gained significant traction as a prominent solving technique. Several open-source projects have been developed with the intention of advancing numerical methods within the research community. Examples such as [1, 2, 3, 4], while they present the latest advancements in the field, often fall short in terms of modularity and flexibility, limiting their adaptability to different contexts or custom systems.

One of the most notable initiatives is NLvib [5], which offers an array of modeling features and a degree of modularity. However, its reliance on a proprietary software stack can present deployment challenges.

*Corresponding author, Email address: quentin.mercier@safrangroup.com


**Fig. 1:** pyHarm's official logo

Recognizing the necessity of a truly modular and flexible platform that serves both educational and research purposes, we propose a new initiative: pyHarm. The primary objectives of pyHarm are as follows:

- Utilize a fully open-source stack that promotes transparency and collaboration.
- Offer a modular platform suitable for both research and teaching in the field of nonlinear dynamics.
- Ensure flexibility to accommodate a wide range of applications.
- Deliver efficient run times.

With these goals in mind, pyHarm aims to provide an accessible and versatile tool for the study and simulation of nonlinear mechanical systems.

## 2 Harmonic Balance Method and pyHarm philosophy

The Harmonic Balance Method (HBM) is a frequency-domain technique for analyzing the steady-state response of dynamic systems [6]. Unlike time-domain methods, HBM focuses on representing periodic solutions as sums of harmonic functions for a given frequency, and takes the Fourier basis as ansatz for the solution. Once injected into the differential equation, one obtains the residual algebric equation to be solved. In order to estimate nonlinear terms, some techniques have been proposed in the literature including the use of taylor serie expansion [7] and the alternative Time domain — Frequency domain method [8] which has been prefered for pyHarm as it has been proven to be more stable for non-smooth nonlinearities [9].

Most existing harmonic balance softwares focuse on expressing the so-called dynamic stiffness matrix, which combines the linear system matrices in the frequency domain for a fixed frequency, adhering closely to the formalism found in the literature. However, pyHarm takes a different approach by decoupling the equation to the maximum extent and splitting each contribution in the Jacobian and residual. This means that each residual or Jacobian evaluation for the system state is performed through a loop over each elementary contribution to the equation. The full system residual or Jacobian is then assembled before entering the solver iteration process.

This approach that aims at maximizing decoupling within the code is the core of pyHarm philosophy. It allows for increased abstraction, which serves as a foundation for modularity and flexibility. By defining core interfaces independently of specific implementations, pyHarm enables seamless substitution and extension of numerical methods without altering the overall structure as long as the numerical methods are kept conform to the interfaces. This abstraction coupled with the built-in plugin system ensures that users can integrate custom components with minimal effort, fostering adaptability to diverse mechanical simulation needs. Moreover, this approach promotes code maintainability, making it easier for researchers and developers to experiment with new algorithms or adapt the framework to evolving requirements. One of the most time-consuming tasks in efficient harmonic balance codebase is the derivation of the analytical Jacobian and its implementation, which can be cumbersome and error-prone. To simplify this, pyHarm ensures interoperability with the JAX library, leveraging automatic differentiation and just-in-time compilation to maintain runtime efficiency.

## 3 Modeling capabilities of pyHarm

Regarding functionality, pyHarm comes equipped with a library of both linear and nonlinear elementary components, including polynomial contributions. The codebase also includes models for Coulomb friction and gaps, which can be implemented using either a penalized approach [10] or the dynamic Lagrangian formulation (DLFT) [11]. Each elementary component provided is designed to work for an arbitrary number of coupled directions, and the reuse of these components is facilitated to construct complex components (e.g., 3D contact with friction).

The code can handle the assembly of substructures with different degrees of freedom per node. Additionally, gyroscopic matrices and hysteretic damping matrices are available for shaft dynamics applications. A set of kinematic conditions is also provided to replicate real test boundary conditions and intermediate basis projections. Currently, the main analysis offered by pyHarm is the nonlinear forced response of the system over a range of frequencies. It employs continuation techniques over the frequency range from a set of available correction equations and prediction methods. The library is equipped with state-of-the-art reduction methods to accelerate the resolution process, such as reduction to nonlinear degrees of freedom [12], and the harmonic number reduction method described in [13].

## 4 Learning how to use pyHarm

To facilitate learning and customization of pyHarm, as well as to introduce users to the harmonic balance method, pyHarm is provided with a set of tutorials that cover a wide range of its functionalities. A first block of four tutorials helps new users understand the fundamental principles and capabilities of the harmonic balance method, including basic concepts such as prediction/correction and AFT procedures. A second block of three tutorials dedicated to the effects of friction is also included with the code. The documentation is continuously updated with every addition to the code, and automatic documentation of the API is generated to ensure comprehensive coverage of the source code.

pyHarm is open to external contributions, and a guide for contributing is available in the documentation. To support contributions, a set of unit tests is integrated into the code along with a non-regression test base to ensure that existing features continue to function correctly.

## 5 Conclusion

The pyHarm initiative aims to offer a new codebase for nonlinear harmonic balance analysis, written in the widely-known and user-friendly language of Python. Its identity is centered around the principles of modularity and flexibility, utilizing software design principles with a focus on abstraction and distributed responsibility. The inclusion of automatic differentiation for Jacobian computation and just-in-time compilation allows the code to accommodate custom components while maintaining high performance, making it an excellent platform for testing new ideas.

The platform encourages and welcomes external contributions, providing a set of tools to facilitate this process. On the application side, pyHarm has already demonstrated its capabilities in modeling and solving lumped mass models, shaft dynamics, and blade-disk interactions in an aeronautical context. The codebase is available at the following address: https://gitlab.com/drti/pyharm.

## References


[1] J. C. Slater, "Mousai: An open source harmonic balance solver for nonlinear systems," in *13th ASME Dayton Engineering Sciences Symposium*, 2017.

[2] R. Arquier, B. Cochelin, and C. Vergez, "Manlab : Logiciel de continuation interactif," *HAL open science*, 2023.

[3] M. Legrand and C. Pierre, "A compact, equality-based weighted residual formulation for periodic solutions of systems undergoing frictional occurrences," *Journal of Structural Dynamics*, vol. 2, 2024.

[4] J. Košata, J. del Pino, T. L. Heugel, and O. Zilberberg, "HarmonicBalance.jl: A Julia suite for nonlinear dynamics using harmonic balance," *SciPost Phys. Codebases*, p. 6, 2022.

[5] J. G. Malte Krack, *Harmonic Balance for Nonlinear Vibration Problems*. Mathematical Engineering, Springer Cham, 2019.

[6] Z. Yan, H. Dai, Q. Wang, and S. Atluri, "Harmonic balance methods: A review and recent developments," *Computer Modeling in Engineering & Sciences*, vol. 137, pp. 1–41, 06 2023.

[7] B. Cochelin and C. Vergez, "A high order purely frequency-based harmonic balance formulation for continuation of periodic solutions," *Journal of Sound and Vibration*, vol. 324, p. 243–262, 2009.

[8] T. M. Cameron and J. H. Griffin, "An Alternating Frequency/Time Domain Method for Calculating the Steady-State Response of Nonlinear Dynamic Systems," *Journal of Applied Mechanics*, vol. 56, p. 149–154, 1989.

[9] L. Woiwode, N. N. Balaji, J. Kappauf, F. Tubita, L. Guillot, C. Vergez, B. Cochelin, A. Grolet, and M. Krack, "Comparison of two algorithms for harmonic balance and path continuation," *MechanicalSystems and Signal Processing*, vol. 136, p. 106503, 2020.

[10] O. Poudou and C. Pierre, *Hybrid Frequency-Time Domain Methods for the Analysis of Complex Structural Systems with Dry Friction Damping*. 2003.

[11] S. Nacivet, C. Pierre, F. Thouverez, and L. Jezequel, "A dynamic lagrangian frequency–time method for the vibration of dry-friction-damped systems," *Journal of Sound and Vibration*, vol. 265, p. 201–219, 2003.

[12] Y. Colaïtis, *Stratégie numérique pour l'analyse qualitative des interactions aube/carter*. PhD thesis, EPM - École Polytechnique de Montréal, 2021.

[13] C. Gastaldi and T. M. Berruti, "A method to solve the efficiency-accuracy trade-off of multi-harmonic balance calculation of structures with friction contacts," *International Journal of Non-Linear Mechanics*, vol. 92, pp. 25–40, 2017.

# Py-Fatigue: An Open-Source Tool for Fatigue Assessment using Stress-Life Methods and Crack Growth Calculations

Ahmed Mujtaba*, Pietro D'Antuono, Wout Weijtjens

*Department of Mechanical Engineering, Vrije Universiteit Brussel, Brussels, 1050, Belgium*

**Abstract**

Py-fatigue is a well-documented open-source Python package developed for efficiently analyzing long-term strain signals collected by Structural Health Monitoring (SHM) systems. Interactive tutorials using Jupyter Notebooks and Binder enable users to easily explore and test the package online. Py-fatigue implements two primary fatigue damage assessment methodologies: stress-life methods and crack growth calculations. These assessments can be performed either directly on time series data or using rainflow `CycleCount` matrices.

Time series signals are first processed into `CycleCount` matrices using a three-point rainflow counting algorithm. Stress ranges in these matrices can then be adjusted for mean stress effects using models such as DNVGL, Goodman, Walker, and Smith-Watson-Topper, prior to fatigue assessment. The stress-life method is implemented in the damage module and supports both linear and multi-linear SN curves. In addition to the linear Palmgren-Miner damage accumulation rule, several non-linear models such as those proposed by Pavlou, Theil, Manson-Halford, Si-Jian, and Leve are also available.

The damage module also supports fatigue assessment based on crack growth calculations. Py-fatigue uses a Paris Law-based approach, modeling crack growth curves as combinations of linear segments within the stable propagation region. The geometry module includes standard geometrical correction factors for crack growth, and it also allows users to define custom geometries by specifying crack growth rates as functions of the stress intensity factor.

**Keywords:** Structural Health Monitoring, Fatigue Life, Mean Stress, Damage Accumulation, Crack Growth

## 1 Introduction / Statement of Need

The Offshore Wind Infrastructure Lab (OWI-Lab) has been actively monitoring the structural health of offshore wind turbines since 2011. As wind turbines are designed to operate over lifetimes exceeding 20 years, wind farm operators seek to evaluate the rate of fatigue damage accumulation based on continuous measurement data. A key objective is to link observed fatigue damage to operational and environmental conditions (EOCs), which are typically recorded in 10-minute data segments. Understanding these links enables more informed decisions regarding maintenance, lifetime extension, and failure prevention.

To support this analysis, there is a growing need for tools that can efficiently process and interpret long-term monitoring data. In response, py-fatigue was developed as an open-source Python package for analysing, storing, and processing strain signals collected through Structural Health Monitoring

* Corresponding author, Email address: ahmed.mujtaba@vub.be

(SHM) systems in offshore wind turbines (OWTs). The package is specifically designed to handle segmented time series, enabling users to compute and accumulate fatigue damage incrementally without requiring reprocessing of the entire time history. This functionality allows for the efficient reconstruction of the full fatigue spectrum from individual cycle count matrices.

Py-fatigue addresses the need for a unified and extensible fatigue analysis toolkit by incorporating state-of-the-art methodologies for fatigue life estimation while maintaining a modular architecture. The package provides two primary assessment approaches: stress-life (SN curve-based) and crack propagation (Paris Law-based) methods.[†]

# 2 Signal Processing and Storage

A typical py-fatigue workflow begins with the collection of a time series array or cycle-count JSON files into py_fatigue.CycleCount (CC) object instances. A time series might be any array-like sequence of values, while a cycle-count JSON file is a memory-efficient way of storing cycle-counted time-series using py-fatigue convention [1].

## 2.1 Cycle Counting of Stress Signals

The cycle-counting functionality of py-fatigue has been extensively detailed in the previous edition of OpenSD (2023) and will only be summarized here.

Once collected, multiple CC objects can be "added" together to build a cycle-count object representative of a longer time span. This is especially useful in the offshore-wind industry, where storing 10-minute-long only time series has become a *de-facto* standard, although the typical dynamics of fatigue cycles might go well beyond this time span. The word "added" has been placed within quotation marks since it namely means concatenating multiple CCs together, as well as the sequence of half-cycles of every CC, i.e., those open hysteresis loops who did not have time to complete. This half-cycle sequence can then be used to reconstruct the low-frequency fatigue dynamics [2], [3], [4], hence retrieving those long-lasting cycles which generally are also the highest in range.

## 2.2 Mean Stress Corrections

Py-Fatigue includes built-in methods for mean stress correction, allowing stress amplitudes within CCs to be adjusted based on their corresponding mean stress values. These correction techniques include common models such as Walker , Smith-Watson-Topper [5], [6], DNVGL [7], and a generic implementation of the Goodman-Haigh line, able to correct the stress amplitude from any input to any output load ratio, generalizing Goodman-Söderberg-Gerber equations which only "correct" the amplitude to fully reversed loading. The equation is:

$$\sigma_{amp,\text{out}} = \frac{1}{\left( \frac{1 - \left( \frac{1 + R_{\text{in}}}{1 - R_{\text{in}}} \cdot \frac{\sigma_{amp,\text{in}}}{\sigma_{ult}} \right)^{n}}{\sigma_{amp,\text{in}}} + \sigma_{amp,\text{out}}^{n-1} \cdot \left( \frac{1 + R_{\text{out}}}{1 - R_{\text{out}}} \cdot \frac{1}{\sigma_{ult}} \right)^{n} \right)}$$

where:

[†] Py-Fatigue v1.*.* requires Python [3.8, 3.9, 3.10], while py-fatigue v2.*.* is compatible with Python [3.10, 3.11, 3.12, 3.13]. It is a 64-bit package, hence not compatible with 32-bit Python.

- σ: stress. The subscript amp stands from amplitude, min and max stand for minimum and maximum, while in and out stand for input and output. Ult stands for ultimate.
- $R = \sigma_{min}/\sigma_{max} = (\sigma_{mean} - \sigma_{amp}) / (\sigma_{mean} + \sigma_{amp})$ is the load ratio
- n: correction exponent (e.g., 1 for Goodman or Söderberg, 2 for Gerber).

```
from py_fatigue import CycleCount
from py_fatigue.testing import get_random_data as D, get_sampled_time as T
# Sample time series definition
time_series = D(T(fs=10, duration=100), seed=41, random_type="normal")
# CycleCount instantiation
cc = CycleCount.from_timeseries(time_series, name="Example")
# Goodman mean stress correction
cc_c = cc.mean_stress_correction("goodman", r_out=0.1, ult_s=900)
```

**Listing 1:** Pseudo-code for Goodman mean stress effect correction from CycleCount

# 3 Stress Life Method

## 3.1 SN Curve Definition

The material resistance to fatigue is defined using SN curves (stress vs. number of cycles to failure) for stress-life fatigue assessment. In py-fatigue, SN curves can be defined using s*lope-intercept pairs* or through *knee points* for piecewise linear or multi-linear curves in py_fatigue.SNCurve class. This flexibility allows users to accurately model both simple and complex material fatigue behaviours, making the package suitable for various materials and load scenarios. The two snippets below illustrate how to instantiate SN curves either through "the classic approach" or through knee points.

```
from py_fatigue import SNCurve
sn_1 = SNCurve(slope=[3, 5], intercept=[12.592, 16.320],
               norm="DNV-RP-C203", environment="Air", curve="C")
sn_2 = SNCurve.from_knee_points(knee_stress=[339.36, 73.11, 29.11],
                                knee_cycles=[1.0E+5, 1.0E+7, 1.0E+9],
                                norm="DNV-RP-C203", environment="Air",
                                curve="C")
```

**Listing 2:** DNV-RP-C203 C (Air) SN curve defined from slope-intercept pairs or knee points

## 3.2 Damage Estimation

Stress-life damage is calculated by combining the CC and the defined SN curve using py_fatigue.damage.stress_life module. Py-fatigue supports both ***l****inear damage accumulation* using the Palmgren-Miner rule [8], and *non-linear damage accumulation models* such as Manson-Halford, Pavlou, and Theil [9], [10], [11]. Before calculating the damage, the stress histogram can be scaled for stress concentration effects via multiplication of the CC by a scaler. Some of these non-linear damage accumulating models require a non-linear solver which has been implemented with Numba.

```
SCF = 2.5
dmg, cum_dmg, _, _= CycleCount.get_nonlinear_damage_with_dca(
    damage_rule="Pavlou", cycle_count=cc_c * SCF, sn_curve=sn_1,
    damage_bands=[0, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0],
    ultimate_stress=900  )
```

**Listing 3:** Non-linear damage accumulation example using Pavlou's model

# 4 Crack Growth Calculations

Crack growth calculations in py-fatigue follow a preprocessing workflow similar to that described in Section 2, including cycle counting and optional mean stress correction. These processed signals serve as the input for crack propagation analysis.

## 4.1 Crack Growth Curve

Py-fatigue implements fatigue crack growth analysis using Paris or Walker Laws, relating the crack growth rate to the range of the stress intensity factor. Similarly to SN curves, a crack growth curve can be defined using slope-intercept pairs or knee points for a piecewise definition.

```
from py_fatigue import ParisCurve
from numpy import array
# pipe-separated knee points
k_str = "29,7e-12|30,9e-11|35,5e-10|50,3e-9|800,2e-4|1_200,2e-3|1_300,3e-2"
k = array([list(map(float, l.split(','))) for l in k_str.strip().split("|")])
pc = ParisCurve.from_knee_points(knee_sif=k[:,0], knee_growth_rate=k[:,1],
                                 norm="Simile-NASGRO", curve="Interpolated")
```

**Listing 4:** Crack growth curve definition from knee points by using Paris' law

## 4.2 Crack Geometry and Crack Growth

Initial crack geometry plays a critical role in defining the initial stress intensity factors (SIFs). Py-fatigue provides built-in crack geometry definition on through-crack on infinite plates and surface cracks on hollow cylinders. These cases are commonly found in offshore and rotating equipment, but the package's modular architecture allows simple definition for virtually any additional crack geometry.

```
from py_fatigue import geometry
geo = geometry.HollowCylinder(initial_depth=3., thickness=15., height=100.,
                              outer_diameter=50., width_to_depth_ratio=3.,
                              crack_position="external")
geo.plot()
```

**Listing 5:** Crack geometry (external crack on a hollow cylinder)

Finally, a crack propagation study can be carried out by using the cycle-count, crack growth curve, and geometry defined.

```
from py_fatigue.damage import crack_growth
cg = crack_growth.get_crack_growth(cc_c, pc, geo, express_mode=True)
print(f"Cycles to end: {int(cg.final_cycles)}")
print(f"Final length: {cg.crack_depth:.4f} mm")
```

**Listing 6:** Crack growth analysis for CC

# References


[1] P. D'Antuono.; W. Weitjens, 'py-Fatigue: Efficiently process, store and analyse load data for fatigue assessments', in *Proceedings of OpenSD 2023*, Jun. 2023, pp. 34–37.

[2] C. Amzallag, J. P. Gerey, J. L. Robert, and J. Bahuaud, ‘Standardization of the rainflow counting method for fatigue analysis’, *International Journal of Fatigue*, vol. 16, no. 4, pp. 287–293, Jun. 1994, doi: 10.1016/0142-1123(94)90343-3.
[3] G. Marsh *et al.*, ‘Review and application of Rainflow residue processing techniques for accurate fatigue damage estimation’, *International Journal of Fatigue*, vol. 82, pp. 757–765, Jan. 2016, doi: 10.1016/j.ijfatigue.2015.10.007.
[4] N. Sadeghi, K. Robbelein, P. D’Antuono, N. Noppe, W. Weijtjens, and C. Devriendt, ‘Fatigue damage calculation of offshore wind turbines’ long-term data considering the low-frequency fatigue dynamics’, *J. Phys.: Conf. Ser.*, vol. 2265, no. 3, p. 032063, May 2022, doi: 10.1088/1742-6596/2265/3/032063.
[5] K. N. Smith, T. Topper, and P. Watson, ‘A stress–strain function for the fatigue of metals (stress-strain function for metal fatigue including mean stress effect)’, *J Materials*, vol. 5, pp. 767–778, Jan. 1970.
[6] Gyoko Oh, ‘Effective stress and fatigue life prediction with mean stress correction models on a ferritic stainless steel sheet’, *International Journal of Fatigue*, vol. 157, 2022, doi: https://doi.org/10.1016/j.ijfatigue.2021.106707.
[7] DNVGL, *Fatigue Design of Offshore Steel Structures: DNVGL-RP-C203*. in DNV-GL. DNV GL AS, 2019. [Online]. Available: https://books.google.fr/books?id=i4qOzQEACAAJ
[8] M. A. Miner, ‘Cumulative damage in fatigue journal of applied mechanics 12 (1945) no. 3, pp’, *A159-A164*, 1945.
[9] Norbert Theil, ‘Fatigue life prediction method for the practical engineering use taking in account the effect of the overload blocks’, *International Journal of Fatigue*, vol. 90, pp. 23–35, 2016, doi: https://doi.org/10.1016/j.ijfatigue.2016.04.006.
[10] Hectors Kris and De Waele Wim, ‘Cumulative Damage and Life Prediction Models for High-Cycle Fatigue of Metals: A Review’, p. 32, Jan. 2021.
[11] D. Pavlou, ‘A deterministic algorithm for nonlinear, fatigue-based structural health monitoring’, *Computer aided Civil Eng*, vol. 37, no. 7, pp. 809–831, Jun. 2022, doi: 10.1111/mice.12783.

# On the experimental coupling with continuous interfaces using frequency-based substructuring and pyFBS

Domen Ocepek Gregor Čepon *

*University of Ljubljana, Faculty of Mechanical Engineering, Laboratory for Dynamics of Machines and Structures, Aškerčeva 6, 1000 Ljubljana, Slovenia*

**Abstract**

In experimental dynamic substructuring, the coupling of substructures sharing a line- or surface-like interface proves to be a challenge due to the difficulties in interface modelling. Modelling a high number of degrees of freedom at the common interface can be too stringent when imposing compatibility and equilibrium conditions, thereby causing redundancy and ill-conditioning. To mitigate the effects of overdetermination and experimental errors, that can lead to a high error amplification, several techniques have been developed, proposing different reduction spaces to weaken the interface conditions. This work presents the implementation and evaluation of these reduction techniques within the open-source *python* package *pyFBS*. In particular, a comparative investigation of the established techniques—namely, the frequency-based modal constraints for fixture and subsystem and singular vector transformation—is conducted within the frequency domain. The comparative study highlights practical considerations for implementing these techniques within the frequency-domain framework of *pyFBS*.

**Keywords:** Frequency-based Substructuring, Continuous Interfaces, python, pyFBS

## 1 Introduction

In the context of experimental frequency-based substructuring (FBS) [1], the main challenge for a successful substructure-coupling implementation remains the modeling of the common interface between the substructures. Coupling of the substructures requires compatibility of the displacements along the interface between two substructures and the equilibrium of interface forces to be satisfied. Practical difficulties relate to the measurement of an appropriate number of degrees of freedom (DoFs) at which interface conditions are imposed. Coupling too many DoFs can results in redundancy and consequently bad conditioning of the interface coupling equations. Ill-conditioning combined with inevitably present measurement errors (random or systematic in nature) can lead to a high error amplification. Limiting the number of interface measurements and imposing weak compatibility avoids those problems, but can significantly deteriorate the accuracy of the coupled dynamics.

A solution can be applied by projecting the measured dynamics into a representative subspace [2]. This mitigates the effects of measurement errors since the interface conditions are now imposed in

*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si

the reduced space and are thus related to as weak. Redundant and insignificant dynamic information that is not included in the reduced subspaces is left uncoupled. This is beneficial since this part is commonly strongly affected by measurement errors, is badly controlled or badly observed. In the context of component-mode synthesis, modal constraints for fixture and subsystem (MCFS) [3] is a commonly used approach the use of which was also suggested for FBS in [4]. A flexible fixture is introduced to the coupling workflow and the reduction basis is defined as a truncated set of physical mode shapes of the fixture. A method developed with the aim to handle flexible interface behavior in the frequency domain is the singular vector transformation (SVT) [5], where force and displacement reduction spaces are extracted from the measured response models by the means of singular value decomposition.

Recently, the release of the *pyFBS python* package brought open-source implementation of the FBS and supplementary methods (such as MCFS and SVT) to the researches and acoustic engineers. All required methods, implemented using an object-orientated approach, can be used in an intuitive manner for consistent coupling predictions, as is demonstrated in this paper.

## 2 Theoretical background and notation

Consider two substructures A and B assembled at the interface DoFs $(\star)_2^{\mathrm{A}}$ and $(\star)_2^{\mathrm{B}}$, as depicted in Fig. 1. With admittances of the individual subsystems known ($\mathbf{Y}^{\mathrm{A}}$ and $\mathbf{Y}^{\mathrm{B}}$) and partitioned in internal

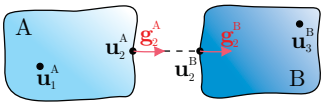



**Fig. 1:** Substructures A and B to be coupled at the common interface.

($(\star)_1^{\mathrm{A}}$ and $(\star)_3^{\mathrm{B}}$) and interface DoFs, the governing equation of motion for the uncoupled system can be written as:

$$\mathbf{u} = \mathbf{Y}^{\mathrm{A|B}} \left( \mathbf{f} + \mathbf{g} \right). \tag{1}$$

The vector $\mathbf{u}$ represents the displacements to the external force vector $\mathbf{f}$, and $\mathbf{g}$ is the vector of interface forces between the substructures that exist only at the interface DoFs, keeping the substructures together. $\mathbf{Y}^{\mathrm{A|B}}$ is a block-diagonal matrix of subsystem's admittances. All subsystems considered are assembled through the proper application of interface conditions[1]. The compatibility of the displacements at the common boundary is recast in the general formulation:

$$\mathbf{B}\,\mathbf{T}_{\mathrm{u}}\,\mathbf{u} = \mathbf{0} \quad \text{where} \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} \end{bmatrix}. \tag{2}$$

The equilibrium condition is imposed by replacing the interface forces using a set of Lagrange multiplier vectors $\boldsymbol{\lambda}$:

$$\mathbf{g} = -\mathbf{T}_{\mathrm{f}}^{\mathrm{H}}\,\mathbf{B}^{\mathrm{T}}\boldsymbol{\lambda}. \tag{3}$$

By eliminating the Lagrange multiplier vector from the set of Eqs. (1 - 3) we obtain:

$$\mathbf{u} = \underbrace{\left[ \mathbf{I} - \mathbf{Y}^{\mathrm{A|B}}\,\mathbf{T}_{\mathrm{f}}^{\mathrm{H}}\,\mathbf{B}^{\mathrm{T}} \left( \mathbf{B}\,\mathbf{T}_{\mathrm{u}}\,\mathbf{Y}^{\mathrm{A|B}}\,\mathbf{T}_{\mathrm{f}}^{\mathrm{H}}\,\mathbf{B}^{\mathrm{T}} \right)^{-1} \mathbf{B}\,\mathbf{T}_{\mathrm{u}} \right]}_{\mathbf{Y}^{\mathrm{AB}}} \mathbf{Y}^{\mathrm{A|B}}\,\mathbf{f}. \tag{4}$$

[1] Weak compatibility is considered here. For more information, an interested reader is refereed to [6]

where $\mathbf{Y}^{AB}$ is the admittance of the assembled system, $\mathbf{T}_u$ is the transformation matrix to project interface displacements to the reduced domain, and $\mathbf{T}_f$ is the transformation matrix for the interface forces which fulfill a weak compatibility in a space of lower dimension. Note that different Boolean matrices can be used to apply the compatibility $\mathbf{B}_u$ and equilibrium $\mathbf{B}_f$ conditions.

A flexible fixture (named transmission simulator or TS) can be introduced to the coupling workflow and attached to substructure B (Fig. 2). The combined (sub)structure is denoted as BTS in the following which is coupled to the substructure A from which the TS structure is then decoupled to obtain the response model for AB. The procedure is schematically depicted in Fig. 2.

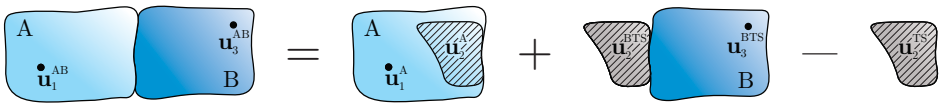



**Fig. 2:** Coupling application with a TS substructure. Common interface (presented as hatched) for all substructures if therefore extended and includes the full TS substructure where inputs and outputs can be measured at the accessible locations.

## 3 Case study

The coupling approach with the MCFS and SVT method adopted the workflow presented in Fig. 3.

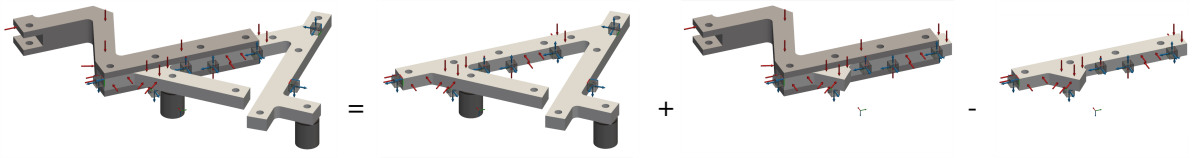



**Fig. 3:** Coupling workflow using flexible fixture; a) final assembly AB, b) substructure A, c) substructure BTS, d) substructure TS. Figure was created using *3D display* feature provided by the *pyFBS* package and shows location of the sensors and impacts used in the measurement campaign.

For the MCFS approach, a *multi-reference modal identification* implemented within the *pyFBS* was used to obtain the first four flexible output and input modes of the TS substructure. These were then added to the 6 rigid body modes, obtained using a numerical model of TS and stacked into transformation matrices $\mathbf{T}_u$ and $\mathbf{T}_f$ [6]. Finally, the response model for AB was obtained using Eq. (4)

For the SVT approach, the dominant singular modes (based on the corresponding singular values) of TS substructure at each frequency line were retained for the reduction bases. Transformation matrices $\mathbf{T}_u$ and $\mathbf{T}_f$ [6] were constructed using *SVT* implementation within *pyFBS* and the assembly was performed using Eq. (4).

The coupling prediction where the number of mode shapes and singular modes retained in the reduction step varied from 8 to 10 are presented in Fig. 4. A reasonably higher number of DoFs retained in the reduction improves prediction in a higher frequency range and vice versa.

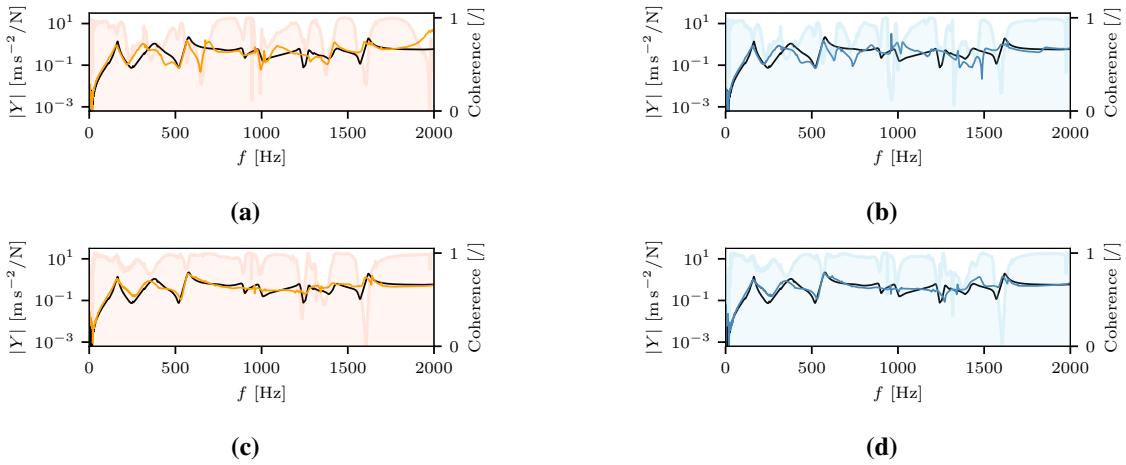



**Fig. 4:** Amplitude and coherence of a FRF of $\mathbf{Y}^{\mathrm{AB}}$ using MCFS and SVT approach with various number of mode shapes/singular modes retained in the reduction bases; a) first 8 mode shapes, b) first 8 singular modes, c) first 10 mode shapes, d) first 10 singular modes. (— reference, — TS approach, — SVT approach, — coherence on TS approach, — coherence on SVT approach)

## References


[1] D. de Klerk, D. J. Rixen, and S. Voormeeren, "General framework for dynamic substructuring: history, review and classification of techniques," *AIAA journal*, vol. 46, no. 5, pp. 1169–1181, 2008.

[2] M. S. Allen, D. Rixen, M. Van der Seijs, P. Tiso, T. Abrahamsson, and R. L. Mayes, *Substructuring in Engineering Dynamics: Emerging Numerical and Experimental Techniques*, vol. 594. Springer, 2019.

[3] M. S. Allen, R. L. Mayes, and E. J. Bergman, "Experimental modal substructuring to couple and uncouple substructures with flexible fixtures and multi-point connections," *Journal of Sound and Vibration*, vol. 329, no. 23, pp. 4891–4906, 2010.

[4] R. L. Mayes, "Tutorial on experimental dynamic substructuring using the transmission simulator method," in *Topics in Experimental Dynamics Substructuring and Wind Turbine Dynamics, Volume 2: Proceedings of the 30th IMAC, A Conference on Structural Dynamics, 2012*, pp. 1–9, Springer, 2012.

[5] F. Trainotti, T. Bregar, S. Klaassen, and D. Rixen, "Experimental decoupling of substructures by singular vector transformation," *Mechanical Systems and Signal Processing*, vol. 163, p. 108092, 2022.

[6] D. Ocepek, F. Trainotti, G. Čepon, D. J. Rixen, and M. Boltežar, "On the experimental coupling with continuous interfaces using frequency based substructuring," *Mechanical Systems and Signal Processing*, vol. 217, p. 111517, 2024.

# A Simulink Platform for the Design of Adaptive TVA

Lisa Ortis[1*] Daniel Casagrande[2] Paolo Gardonio[1]

[1] *Università degli Studi di Udine - DPIA, Via delle Scienze 206, 33100, Udine, Italy*

[2] *Universidad de O'Higgins, Instituto de Ciencias de la Ingeniería, Libertador O'Higgins 611, 2820000, Rancagua, Chile.*

**Abstract**

This paper presents a SIMULINK platform that has been developed to study the vibration response of flexible distributed structures equipped with adaptive Tunable Vibration Absorbers. The platform has been conceived specifically to design adaptive algorithms that optimally tune the vibration absorbers either to control the resonant response of a target flexural mode of the hosting structure excited by a broadband disturbance or to control the time-harmonic flexural response of the hosting structure subject to a tonal excitation. The platform can work on simple hosting structures whose modal response can be derived from analytical formulae as well as structures with complex geometries, whose modal response can be either derived numerically using finite element methods or measured with distributed (e.g. laser vibrometer, video cameras) or point (accelerometers) sensors. The platform works in the time-domain and can be used to investigate the stability and convergence properties of the tuning algorithms. Also, it provides models for simple mass-spring-damper vibration absorbers as well as for electro-mechanical absorbers or vacuum controlled absorbers. Simple tuning algorithms based on phase considerations of the absorber vibration response as well as more advanced algorithms based on energy cost functions can be investigated.

**Keywords:** tunable vibration absorbers; tonal vibration control; broadband vibration control; adaptive vibration absorbers; tuning algorithms

## 1 Introduction

Vibration absorbers have been used since the early 1900s to control either the time-harmonic response of mechanical systems subject to tonal disturbances or the broad-band resonant response of target modes of structures subject to stochastic stationary disturbances [1-4]. They are quite elementary devices that can be modelled in terms of a base mass and a seismic mass connected via a spring and damper in parallel [4-7]. Their working principle is also quite simple; in fact, their fundamental resonance frequency should be tuned to match the frequency of the tonal excitation [8,9] or to about coincide with the natural frequency of the hosting structure subject to a stochastic excitation [10-12]. To control tonal disturbances, the effect of damping should be minimised such that the absorber offers to the hosting system a large reactive impedance magnified, indeed, by the resonant response of the absorber [8,9]. Instead, to lower the target resonant response of a distributed hosting structure, the damping of the absorber should be optimized in such a way as to maximise the time-averaged vibration absorption [10-12]. In general, to have large vibration control effects, the absorber should

* Lisa Ortis, Email address: lisa.ortis@uniud.it

be designed with the largest possible seismic mass consistent with the design constraints [10]. Also, for tonal control, it should be located at the point where the mechanical system is excited, whereas, for broadband control, it should be located close to anti-nodal lines of controlled mode.

Despite their simple design and the advanced formulations developed over the years for the optimal tuning [12-21], to function properly, vibration absorbers must be finely tuned. This is quite a relevant problem, since in most applications, the frequency of tonal excitations varies over time. Also, the resonant response of distributed structure tends to vary over time, primarily because of tensioning effects, which may be caused by temperature changes or by operation conditions. For this reason, starting in the 1990s, the idea of developing adaptive Tunable Vibration Absorbers (TVAs) has been investigated [22]. Here, the research work has been focused on two aspects: on the one hand the design of electro-mechanical vibration absorbers whose fundamental resonance frequency and damping ratio can be tuned online electronically and on the other hand the development of simple tuning algorithm that can be implemented online to track changes of either the frequency of the tonal excitation or the natural frequency and damping ratio of the target mode to be controlled.

This paper provides a SIMULINK platform for the design of TVA mounted on flexible mechanical systems. The platform relies on a State-Space model of the hosting-structure and TVA assembly. Time domain analyses can be implemented for both tonal and stochastic stationary excitations. The platform relies on the modal parameters of the hosting structure (i.e. modal mass, damping and stiffness matrices, natural frequencies and modal amplitudes at relevant points), which, for simple structures, can be calculated from analytical expressions. Alternatively for structures characterized by complex shapes, they can be derived numerically, using Finite Element Method for example, or identified experimentally from vibration measurements on prototypes. The platform has been specifically conceived for on-line implementation of algorithms for the self-tuning of the TVA and, in first instance, it will be employed to study a new type of in-vacuo beam-like absorbers whose tuning can be adapted by varying the level of vacuum in the composite beam element [23,24].

## 2 Simulink Platform

The Simulink platform is based on a state space model for the flexural vibration of a hosting mechanical structure equipped with one or multiple TVA (see Figure 1a). The model is derived from an energy formulation based on Hamilton's principle for mechanical systems, which states that

$$\int_{t_1}^{t_2}[\delta(T-U)+\delta W_{nc}]dt=0\ , \tag{1}$$

where $T$, $U$, $\delta W_{nc}$ are respectively the total kinetic energy, strain energy and work done by non-conservative forces (i.e. external and dissipative forces) of the whole system, that is of the hosting flexible structure and the TVAs. Assuming the flexural vibration of the hosting structure is expressed in terms of $N$ generalized coordinates, that is

$$w(\mathbf{x},t)=\boldsymbol{\phi}(\mathbf{x})\mathbf{q}(t)\ , \tag{2}$$

where the vector $\boldsymbol{\phi}(\mathbf{x})$ contains the amplitudes of the first $N$ flexural modes of the hosting structure at position $\mathbf{x}$ and the vector $\mathbf{q}(t)$ contains the first $N$ generalized coordinates of time, Hamilton's principle leads to the following system of differential equations, which, for simplicity are cast in matrix form:

$$\mathbf{M}\ddot{\mathbf{y}}+\mathbf{C}\dot{\mathbf{y}}+\mathbf{K}\mathbf{y}=\mathbf{f}\ , \tag{3}$$

where $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$ are the global mass, damping and stiffness matrices, and the vectors $\mathbf{y}$, $\mathbf{f}$ contain the overall set of generalized displacements and generalized forces, including those relative to the degrees of freedom of the TVA seismic masses (see Figure 1b).

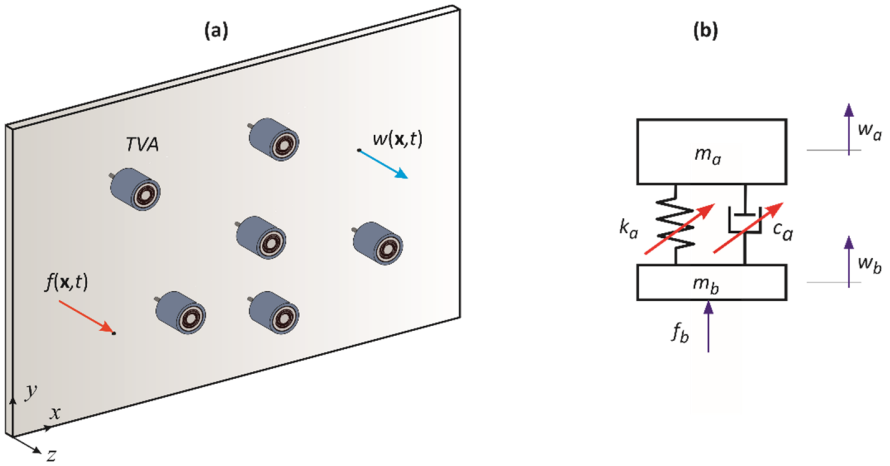



Fig. 1: (a) Structure with multiple TVA. (b) TVA lumped parameter model.

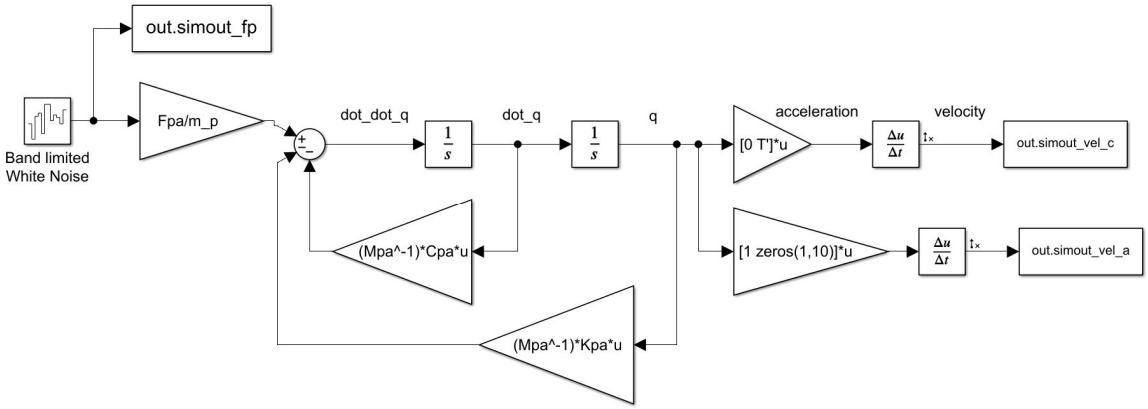



Fig. 2: Simulink State-Space platform.

The state space formulation is then derived straightforwardly with reference to the state vector

$$\mathbf{z} = \begin{bmatrix} \mathbf{y} \\ \dot{\mathbf{y}} \end{bmatrix}, \tag{4}$$

such that, the following system of first order differential equations holds:

$$\begin{bmatrix} \dot{\mathbf{y}} \\ \ddot{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \dot{\mathbf{y}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{f}, \tag{5}$$

This platform can be employed to implement several analyses. For instance, assuming time-harmonic vibrations, the global response of the hosting structure expressed with reference to the time-averaged total flexural kinetic energy, which, is given by

$$K(\omega) = \frac{1}{4} m \dot{\mathbf{q}}^H(\omega) \mathbf{q}(\omega), \tag{6}$$

where the vector $\mathbf{q}(\omega)$ contains the complex amplitudes of the generalised coordinates for the flexural vibration of the hosting structure and $m$ is the mass of the structure itself. Also, the tuning of the TVAs can be implemented with reference to the time-averaged vibration power absorbed by these devices [10,25], which, assuming time-harmonic vibrations, for one TVA is given by

$$P(\omega) = \frac{1}{2}\mathrm{Re}\{f_b^*(\omega)\dot{w}_b(\omega)\} = \frac{1}{2}c_a|\dot{w}_a(\omega) - \dot{w}_b(\omega)|^2 \,, \tag{7}$$

where, as shown in Figure 1b, $f_b(\omega)$, $\dot{w}_b(\omega)$ are the complex force and velocity at the base of the TVA, $\dot{w}_a(\omega)$ is the complex velocity of the TVA seismic mass and $m_a$, $c_a$, $k_a$, $m_b$, are respectively the seismic mass, damping, stiffness and base mass of the lumped elements composing the TVA model. Figure 2 shows a simplified version of the SIMULINK model that implements the state space formulation presented above.

## 3 Simulation Results

Figures 3, 4, 5 show simulation results generated for a model problem given by a thin rectangular plate hosting structure equipped with a single TVA set to control the resonant response of the first flexural mode. Figure 3 shows that the TVA can generate a significant reduction of the resonant response of the first mode, that can exceed about 40 dB. Also Figure 4 confirms that the optimally tuned TVA rises significantly the vibration energy absorption in a small finite band centered at the target resonance frequency. Indeed, the-graph in Figure 5a confirms that the flexural kinetic energy of the structure averaged over a 20 Hz band centered at the first resonance frequency follows an inverted bell shape as the stiffness of the TVA is raised and reaches a minimum for an optimal $k_{opt}$

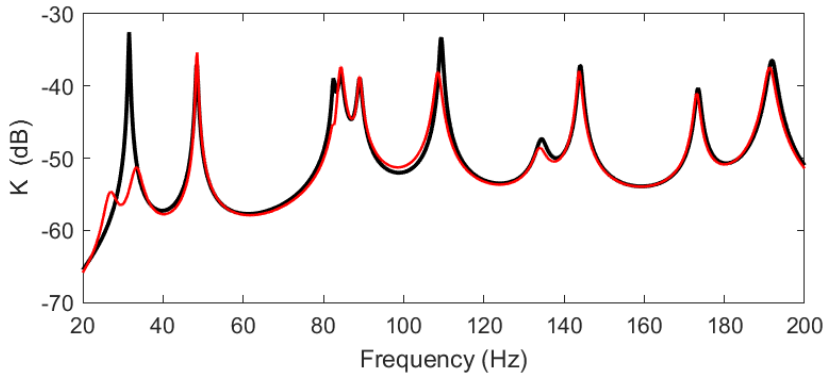



Fig. 3: Flexural kinetic energy of the structure without (black line) and with (red line) the TVA optimally tuned to control the resonant response of the first flexural natural mode.

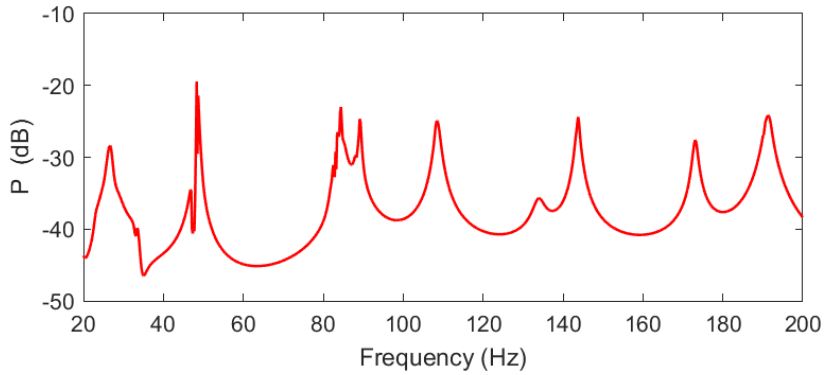



Fig. 4: Vibration power absorbed by the TVA optimally tuned to control the resonant response of the first flexural natural mode.

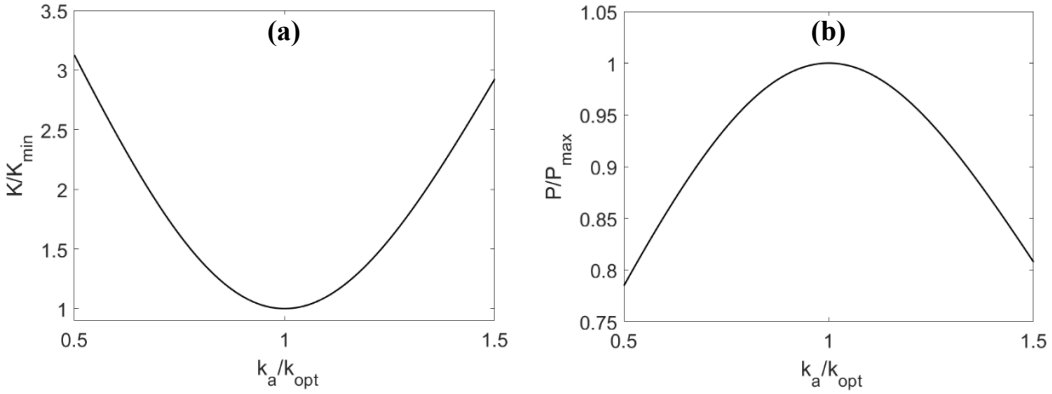



Fig. 5: Structure flexural kinetic energy and TVA absorbed vibration power integrated over a 20 Hz frequency band centered at the first resonance frequency of the structure.

value of the stiffness. Alternatively, the graph in Figure 5b shows that the vibration power absorbed by the TVA averaged in the same 20 Hz band follows a bell shape as the stiffness of the TVA is raised and reaches a maximum for the same optimal $k_{opt}$ value of the stiffness. Hence, the SIMULINK model confirms the findings presented in Refs. [10,25].

# 4 Conclusions

This paper has presented a SIMULINK platform for the simulation of on-line operation of adaptive TVA. The platform is structured in such a way as the dynamics of complex flexible structures equipped with multiple TVA can be calculated for deterministic as well as stochastic primary disturbances. Also, it is specifically designed to mimic the online implementation of self-tuning TVA, in particular the tuning algorithms as well as the electro-mechanical design of adaptive TVA as well as the sensing that provides smart cost functions that would allow the local detection of the global vibration of the hosting structure.

# References

[1] H. Frahm, "Device for damping vibrations of bodies," US989958A

[2] J. Ormondroyd and J. P. Den Hartog, "The theory of the dynamic vibration absorber," *ASME Journal of Applied Mechanics*, vol. 50, pp. 9–22, 1928.

[3] J. E. Brock, "A note on the damped vibration absorber," *J. Appl. Mech.*, vol. 3, p. A284, 1946, doi: 10.1115/1.4009588.

[4] J. P. Den Hartog, *Mechanical vibrations*. in Dover books on engineering. New York: Dover Publications, 1985.

[5] J. B. Hunt, *Dynamic vibration absorbers*. London: Mechanical Engineering Publications, 1979.

[6] D. J. Inman, *Engineering Vibration*, vol. 621. Prentice-Hall, 1994.

[7] D. J. Mead, *Passive vibration control*. Chichester: John Wiley & sons, 2000.

[8] M. J. Brennan and N. S. Ferguson, "Vibration control," in *Advanced Applications in Acoustics, Noise and Vibration*, First., F. Fahy and J. Walker, Eds., London: CRC Press, 2004, p. 656

[9] D. Casagrande, G.K. Rodrigues and P. Gardonio. Tonal vibration control with a self-tuning absorber employing the extremum seeking algorithm, *Mechanical Systems and Signal Processing*, vol 211, 2024. https://doi.org/10.1016/j.ymssp.2024.111254

[10] P. Gardonio, E. Turco, A. Kras, L. D. Bo, and D. Casagrande, "Semi-active vibration control unit tuned to maximise electric power dissipation," *Journal of Sound and Vibration*, vol. 499, p. 116000, May 2021, doi: 10.1016/j.jsv.2021.116000.

[11] P. Gardonio and E. Turco, "Tuning of vibration absorbers and Helmholtz resonators based on modal density/overlap parameters of distributed mechanical and acoustic systems," *Journal of Sound and Vibration*, vol. 451, pp. 32–70, Jul. 2019, doi: 10.1016/j.jsv.2019.03.015.

[12] M. Zilletti, S. J. Elliott, and E. Rustighi, "Optimisation of dynamic vibration absorbers to minimise kinetic energy and maximise internal power dissipation," *Journal of Sound and Vibration*, vol. 331, no. 18, pp. 4093–4100, Aug. 2012, doi: 10.1016/j.jsv.2012.04.023.

[13] O. Nishihara and T. Asami, "Closed-Form Solutions to the Exact Optimizations of Dynamic Vibration Absorbers (Minimizations of the Maximum Amplitude Magnification Factors)," *Journal of Vibration and Acoustics*, vol. 124, no. 4, pp. 576–582, Oct. 2002, doi: 10.1115/1.1500335.

[14] S. Krenk, "Frequency analysis of the tuned mass damper," *J. Appl. Mech.*, vol. 72, pp. 936–42, 2005, doi: 10.1115/1.2062867.

[15] P. Bisegna and G. Caruso, "Closed-form formulas for the optimal pole-based design of tuned mass dampers," *J. Sound Vib.*, vol. 331, pp. 2291–314, 2012, doi: 10.1016/j.jsv.2012.01.005.

[16] S. Krenk and J. Høgsberg, "Equal modal damping design for a family of resonant vibration control formats," *J. Vib. Control*, vol. 19, pp. 1294–315, 2012, doi: 10.1177/1077546312446796.

[17] G. B. Warburton, "Optimum absorber parameters for various combinations of response and excitation parameters" *Earthquake Eng. Struct. Dyn.*, vol. 10, pp. 381–401, 1982.

[18] S. Krenk and J. Høgsberg, "Tuned mass absorbers on damped structures under random load" *Prob. Eng. Mech.* Vol. 23 408–415, 2008.

[19] S. Krenk and J. Høgsberg, "Tuned mass absorber on a flexible structure" *J. Sound Vib.* Vol. 333, pp. 1577–1595, 2014.

[20] S. Krenk and J. Høgsberg, "Tuned resonant mass or inerter-based absorbers: unified calibration with quasi-dynamic flexibility and inertia correction" *proc. the Royal society A* vol. 472, 2016.

[21] P. Gardonio and M. Zilletti, "Integrated tuned vibration absorbers: a theoretical study.", *Journal of the Acoustical Society of America*, vol. 134(5), pp. 3631-3644, 2013.

[22] J. Q. Sun, M. R. Jolly, and M. A. Norris, "Passive, Adaptive and Active Tuned Vibration Absorbers—A Survey," *Journal of Vibration and Acoustics*, vol. 117, no. B, pp. 234–242, Jun. 1995, doi: 10.1115/1.2838668.2]

[23] P. Gardonio, E. Rustigh, S. Baldini, C. Malacarne and M. Perini, "In-vacuo adaptive beam element for vibration control", *Mechanical Systems and Signal Processing*, vol 224, 2005.

[24] P. Gardonio, L. Ortis, E. Rustighi, C. Malacarne and M. Perini, "Multi-Resonant TVA Formed by a Tree of Deflated Composite Beams with Core Structured Fabrics", *Smart Materials and Structures*, vol. 34, 2025. https://doi.org/10.1088/1361-665X/ad9c07.

[25] E. Turco, L. Dal Bo and P. Gardonio, "Tuning of a shunted electro-magnetic absorbes based on the maximisation of the electrical power dissipated", Proceedings of the Institution of Mechanical Engineers, Part C: *Journal of Mechanical Engineering Science*, No. 235 vol.14, pp. 2570-2586, May 2021.

# Non-contact beam bridge damage monitoring through a Digital Image Correlation-based methodology

Valentina Pasquinelli[1], Gloria Cosoli[2], Milena Martarelli[1], Paolo Castellini[1], Gian Marco Revel[1]

[1]*Marche Polytechnic University, Department of Industrial Engineering and Mathematical Sciences (DIISM), 60131 Ancona (AN), Italy*
[2]*eCampus University, Department of Theoretical and Applied Sciences (DiSTA), 22060 Novedrate (CO), Italy*

**Abstract**

Targeting the resilience of structures to emergency situations is an impacting approach which allows to efficiently alert through early warnings in case of structural damages occurrence. This paper focuses on a scaled beam bridge damage monitoring in the framework of the Horizon Europe MULTICLIMACT project (MULTI-faceted CLIMate adaptation ACTions to improve resilience, preparedness and responsiveness of the built environment against multiple hazards at multiple scales - GA n. 101123538). A laboratory test bench has been developed to stress the structure at different load levels (i.e., pre-cracking and pre-yielding) and analyse its dynamic behaviour in free oscillations. Through a vision-based system, the full-field displacement has been measured via Digital Image Correlation technique and used to identify the resonance frequency of the beam first mode. As the damage of the beam progresses, the material stiffness decreases, hence also the resonance frequency: in particular, a 9.2 Hz-decrement has been assessed across the whole test. This shift in frequency acts as an indicator of the structural health of the beam bridge. Indeed, the beam region of interest analysed via DIC is very narrow (210x100 mm) and does not include areas with crack formation, anyway, enables its detection. Therefore, the methodology presented can be applied in multiple regions of the structure, proving to be a flexible and reliable application – the test carried out has indicated a 99% Pearson's correlation with accelerometer-based data – enabling to early detect the state of damage of the structure, thus improving its resilience in case of hazards and extreme events like earthquakes.

**Keywords:** Structural Health Monitoring, soft sensing, Digital Image Correlation, early warning, bridge monitoring, pyIDI, open-source

## 1 Introduction

Non-destructive techniques are fundamental to evaluate the structural health of buildings and infrastructures and serve to define synthetic damage indices in order to have a high-level parameter tracking the status of a building during the whole life cycle [1]. This type of information can be exploited for the generation of early warnings [2]-[4] that can support the efficient and timely management of emergency situations (e.g., due to extreme climate events or earthquakes), thus enhancing the resilience of the structure and its inhabitants.

This paper presents the results of laboratory mechanical and vibrational tests carried out on ECCs in the framework of the Horizon Europe MULTICLIMACT project, aimed at improving the resilience of the built environment towards disruptive events [5]. A reinforced concrete beam, manufactured with engineered cement composites (ECCs) and containing sustainable recycled carbon-based materials for improving the self-sensing capability of the material [6], was designed in order to

replicate full-scale beam bridge vibration frequencies. An image-based inspection via high-speed camera has been applied to monitor the resonant frequency of the beam. In Section 2 the test procedure is described including the specifications of the beam tested and the vision-based monitoring setup. Section 3 reports the results along with correlation with accelerometer-based data. Finally, Section 4 reports the conclusion and future steps of the work.

# 2 Materials and Methods

The most common structural bridge form is a span of a beam bridge supported at each end. An experimental campaign has been carried out on a specimen which simulates a beam bridge span: a cement-based scaled reinforced beam (100x100x3000 mm). The specimen was realized according to the mix-design reported in Tab. 1 and includes sustainable carbon-based additions in the form of filler (biochar – BCH) and fibers (recycled carbon fibres – RCF) enhancing the self-sensing capability of the material as well as the circularity in the construction sector [7].

## 2.1 Measurement setup and test sequence description

Experimental release tests at different load levels were performed to analyze the behavior of the beam during free vibration oscillations. Six loading steps were defined to ensure the release at different strain states of the beam to analyze two different conditions: uncracked (steps 1-3) and pre-yielding (steps 4-6) of the longitudinal rebars. To perform the release tests, the specimen was placed in a self-balancing reaction frame, constrained at the ends with double-effect rollers; the loads were applied by a hydraulic jack in the middle of the beam (see Fig. 1-a). The hydraulic jack was controlled by a calibrated mechanical fuse designed to break at specific load levels (Tab. 2), ensuring the instant release of the force upon its failure.

Tab. 1: Mix-design (g/L) of the mortar with conductive carbon-based additions for the casting of reinforced beams for vibrational tests.

| Cement | BCH | RCF | Sand | Water | Water-to-cement ratio |
|---|---|---|---|---|---|
| 505 | 10 | 1 | 1400 | 303 | 0.60 |

Tab. 2 Mechanical fuse specifications.

| Step | Load level (kN) | Diameter di (mm) |
|---|---|---|
| 1 | 0.30 | 0.65 |
| 2 | 0.65 | 0.95 |
| 3 | 0.95 | 1.10 |
| 4 | 1.30 | 1.35 |
| 5 | 2.00 | 1.60 |
| 6 | 3.00 | 2.15 |

To track the beam with Digital Image Correlation, a speckle pattern was applied to the beam using white spray paint over a black painted background. A high-speed camera (model Phantom v10) was placed in front of the beam, framing the central part of the beam (field of view 210x100 mm), and triggered by the mechanical fuse break. The mechanical fuse was intercepted by metal grippers (in points A and B, see photo in Fig. 1-b and scheme in Fig. 1-c), interrupting the circuit at the fuse breaking and leading to a voltage drop from 5 V to 0 V, which is used to trigger the high-speed camera (the electrical circuit is reported in Fig. 1-c).

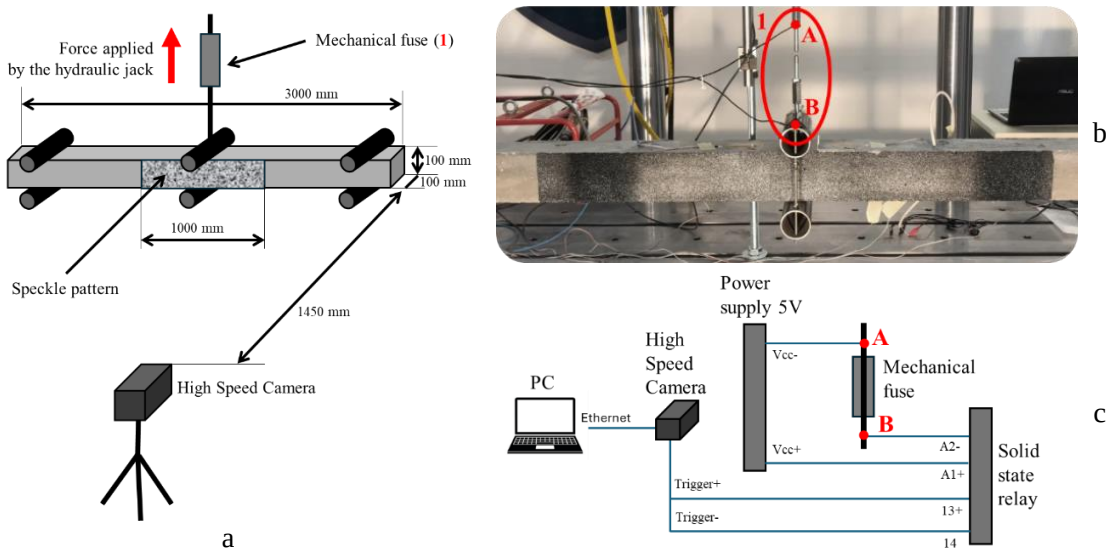



Fig. 1: Measurement setup scheme (a), photo (b) electrical scheme for triggered acquisition (c)

## 3 Results

The displacement of the beam is extracted with DIC methodology via the open-source Python package pyIDI and used to identify the resonant frequency of the beam. The implementation is based on the Lucas-Kanade algorithm that tracks only the rigid translations of the subsets [8][9].

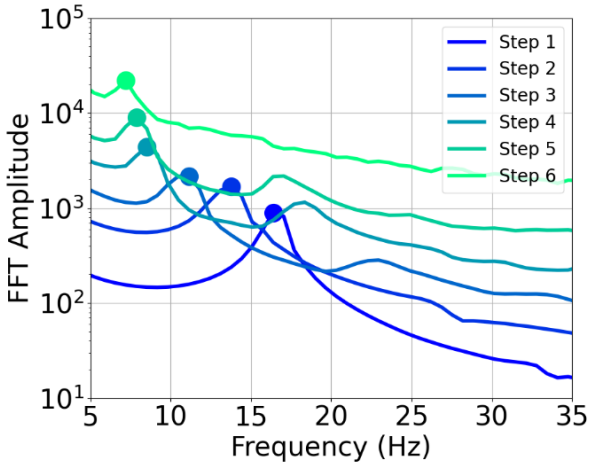



Fig. 2: Image-based FFT Amplitude

The Fast Fourier Transform (FFT) of the displacement signal (Fig. 2) highlights a peak which corresponds to the first resonant mode of the beam which decrements along the progressive loading steps of the test. Indeed, the cracks formation decreases the modulus of elasticity of the material, hence making the natural frequencies reduce. This result is correlated with the measurement obtained from an accelerometer placed on the beam – 400 mm distant from the edge – which has been used to continuously monitor the vibration of the beam during the loading tests. Pearson's correlation coefficient (Fig. 3) between the image-based and the accelerometer-based frequencies (reported in Tab. 3) is 0.99, indicating a strong linear correlation between the two techniques, proving that the vision-based technique is a valid alternative to the traditional contact-based vibrational monitoring techniques.

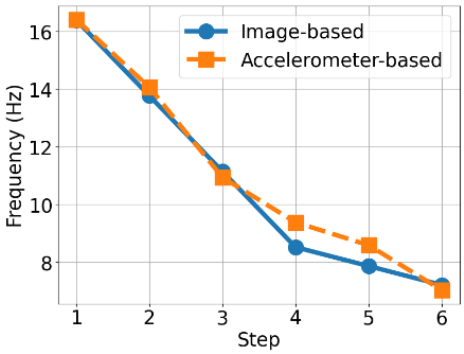



Fig. 3 Pearson's correlation

Tab. 3: Frequency decrement

| Step | Image-based *fn* | Accelerometer-based *fn* |
|---|---|---|
| 1 | 16.39 | 16.41 |
| 2 | 13.77 | 14.06 |
| 3 | 11.14 | 10.94 |
| 4 | 8.52 | 9.38 |
| 5 | 7.87 | 8.59 |
| 6 | 7.21 | 7.03 |

# 4 Conclusion

Non-contact techniques are fundamental in case of test structures difficult to reach for sensors installation, such as bridges. The methodology presented is peculiar as it enables the Structural Health Monitoring (SHM) of the beam even though not framing a part of the structure with cracks, which could be the use case of a real-life scenario, proving to be a valid alternative to the contact measurement techniques, as Pearson's correlation is 0.99. Future developments of the work could be the implementation of early warnings based on the damage status of the structure which could consider, besides the resonance frequency, other model parameters such as damping.

# 5 Acknowledgements

This work is supported by MULTICLIMACT project (MULTI-faceted CLIMate adaptation ACTions to improve resilience, preparedness and responsiveness of the built environment against multiple hazards at multiple scales - GA n. 101123538).

# References

[1] G. Cosoli, M. Martarelli, A. Mobili, F. Tittarelli, G.M. Revel, Damage Identification in Cement-Based Structures: A Method Based on Modal Curvatures and Continuous Wavelet Transform, Sensors 2023, Vol. 23, Page 9292. 23 (2023) 9292. https://doi.org/10.3390/S23229292.

[2] C. Rainieri, G. Fabbrocino, E. Cosenza, Integrated seismic early warning and structural health monitoring of critical civil infrastructures in seismically prone areas, Http://Dx.Doi.Org/10.1177/1475921710373296. 10 (2010) 291–308. https://doi.org/10.1177/1475921710373296.

[3] J. Wang, Y. Fu, X. Yang, An integrated system for building structural health monitoring and early warning based on an Internet of things approach, Int. J. Distrib. Sens. Networks. 13 (2017).

[4] L. Deng, S. Lai, J. Ma, L. Lei, M. Zhong, L. Liao, Z. Zhou, Visualization and monitoring information management of bridge structure health and safety early warning based on BIM, J. Asian Archit. Build. Eng. 21 (2022) 427–438. https://doi.org/10.1080/13467581.2020.1869013.

[5] C. Lanfranconi, C. Fuggini, G. Cosoli, G.M. Revel, R. Chirico, M. Kontogeorgos, Advancing Resilience of the Built Environment by Digital and Measurement Technologies, 2024 IEEE Int. Work. Metrol. Living Environ. MetroLivEnv 2024 - Proc. (2024) 460–464. https://doi.org/10.1109/METROLIVENV60384.2024.10615855.

[6] A. Mobili, G. Cosoli, N. Giulietti, P. Chiariotti, T. Bellezze, G. Pandarese, G.M. Revel, F. Tittarelli, Biochar and recycled carbon fibres as additions for low-resistive cement-based composites exposed to accelerated degradation, Constr. Build. Mater. 376 (2023) 131051. https://doi.org/10.1016/j.conbuildmat.2023.131051.

[7] I. Amarasinghe, Y. Hong, R.A. Stewart, Development of a material circularity evaluation framework for building construction projects, J. Clean. Prod. 436 (2024) 140562. https://doi.org/10.1016/J.JCLEPRO.2024.140562

[8] Klemen Zaletelj, Janko Slavic and Miha Boltezar, Full-field DIC-based model updating for localized parameter identification, Mechanical Systems and Signal Processing, Volume 164, February 2022, https: // doi. org/ 10. 1016/ j. ymssp. 2021. 108287

[9] Masmeijer T., Habtour E., Zaletelj, K., & Slavič, J., (2024). Directional DIC method with automatic feature selection. MSSP, https://doi.org/10.1016/j.ymssp.2024.112080

# Python package for monitoring of damping ratio in mechanical systems

Stipe Perišić  Ivan Tomac *  Jani Barle

*Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture*
*University of Split, Ruđera Boškovića 32, HR-21000 Split, Croatia*

**Abstract**

Monitoring the condition of mechanical systems subjected to dynamic loads is an important part of ensuring reliable and safe operation. One indicator that can be reliably linked to the deterioration of these systems is the damping ratio. Since changes in the damping ratio often precede failures, monitoring the damping ratio is critical for assessing system health. This paper presents a novel Python package that automates the detection of free vibration events and estimates the damping ratio. The package utilises the bootstrap technique to identify free response segments without the need for explicit excitation force measurements. Bayesian gamma-exponential conjugate prior is then used to estimate the damping ratio as a probability distribution, which provides a quantitative measure of uncertainty. The utility of the proposed approach is validated by experimental testing of the clamped beam. The results demonstrate the accuracy and robustness of the Python package, making it a valuable tool for real-time monitoring the health of various mechanical systems.

**Keywords:** Python, monitoring, damping ratio, Bayesian analysis, bootstrap

## 1 Introduction / Statement of Need

To improve the reliability and safety of a system, monitoring is essential. This involves observing a measurable system parameter and tracking its value in order to detect and monitor system degradation [1]. In this work, the damping ratio is chosen as the parameter to be monitored, as it correlates with the degradation process [2]. A novel monitoring approach based on the bootstrap technique and Bayesian analysis is demonstrated, which is applicable to systems with a dominant first mode. The method is suitable for systems that operate intermittently and where free response events are expected. This paper briefly explains the research conducted in Paper [3] and demonstrates its implementation as an open-source Python package.

Identification of damping is performed in a two-stage process. The data is measured continuously and divided into segments of adjustable length. Then each segment is tested for free response using the bootstrap (BT) [4, 5]. Once the free response is detected, damping is determined using Bayesian analysis [6], which also allows the uncertainty to be quantified. The package is published under the MIT licence and is designed as a class object, whereby the methods used for BT and Bayesian analysis are implemented as integrated SciPy functions in the stats and signal sub-packages. Lightweight Data

*Corresponding author, Email address: itomac@fesb.hr

AcQuisition (LDAQ) is used for signal acquisition. The proposed monitoring approach is tested on a real signal, an experimental testbed with a multidegree-of-freedom beam system was used.

## 2 Brief theoretical background

The proposed approach is applicable to systems with the dominant first mode. Estimating the damping value from the free response using logarithmic decrement ($\delta$) makes it possible to determine the condition of the system. Bayesian analysis is used to estimate the state and uncertainty. The Bayesian model comprises a prior distribution $\pi_0(\theta)$ and a likelihood function $f(x\,|\,\theta)$, which together give the posterior distribution $\pi_1(x\,|\,\theta)$. Sometimes an analytical solution, i.e. a conjugate prior [7], can be obtained for a certain selection of prior distribution and likelihood function. The conjugate gamma-exponential prior, Eq. (1), is used here.

$$\pi_1(\theta\,|\,T) = \frac{\kappa_\mathrm{n}^{\alpha_\mathrm{n}}}{\Gamma(\alpha_\mathrm{n})}\,\theta^{\alpha_\mathrm{n}-1}\,e^{-\kappa_\mathrm{n}\theta} \tag{1}$$

In Eq. (1) parameters $\alpha_\mathrm{n}$ and $\kappa_\mathrm{n}$ referees to hyperparameters and can be calculated by Eq.(2).

$$\alpha_\mathrm{n} = \alpha_0 + n; \quad \kappa_\mathrm{n} = \kappa_0 + \sum_{i=1}^{\mathrm{n}} x_i \tag{2}$$

where $n$ is the total number of data points, $x_i$ is individual data and $\alpha_0$ and $\kappa_0$ are prior values determined on the basis of experience or data from a similar systems. The conjugate prior (Eq. (1)) can only be applied to a free response signal. Therefore, signal processing is required to detect free response. The data used to recognise a free response is the amplitude difference. During a free response, the difference is negative as each subsequent amplitude is smaller than the previous one. By resampling these amplitudes using the BT technique, the mean and confidence intervals can be estimated as shown in [5]. The sign of the mean value and the confidence interval distinguishes events with free and non-free response. The bootstrap steps are summarised in Tab. 1. In addition to the requirement

**Tab. 1:** Steps for Bootstrap.

| | |
|---|---|
| 1. | Set the total number of resamples $N_\mathrm{r}$ |
| 2. | Generate $N$ samples from discrete uniform distribution $U(1, N)$ |
| 3. | Resample the original data set |
| 4. | Calculate new mean value |
| 5. | Repeat steps from 2 – 5 up to $N_\mathrm{r}$ |

for a negative sign, the width of the confidence interval and the length of the segment are also important parameters. For details on calculating the width of the confidence interval and determining the appropriate segment length, see [3].

## 3 Experimental verification of damping monitoring

To experimentally verify the monitoring approach, a mechanical testbed with beam system is used. In Fig. 1 damping monitoring is illustrated. The top panel shows the system response, while the

bottom panel shows the free response detection and damping estimation within the segment. The segments are divided by vertical grey lines, each segment lasting 4 seconds. Horizontal green and red dashed lines represent the Bayesian damping estimation. The green colour indicates the accepted estimation, while the red denotes the rejected one, which is further highlighted bay red "x". The estimated uncertainty is captured by 90% confidence intervals, which are shown with a horizontal green line. The black horizontal lines are reference damping values. Fig. 1 indicates that the detection

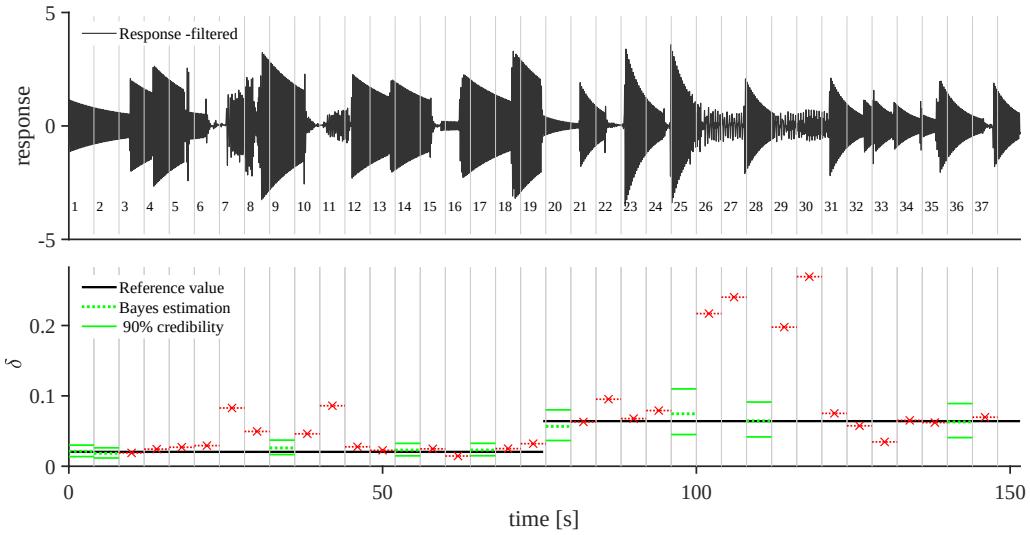



**Fig. 1:** System response (upper panel), free response detection damping estimation (lower panel).

of free response is frequent, ensuring that change in the damping value is well monitored. The damping value estimation is accurate for all accepted segments, therefore, Fig. 1 provides strong evidence supporting the efficiency of the combinations of BT and Bayesian analysis. In Fig. 2 the Bayesian damping estimation for segment 2 is presented. As can been seen, the damping estimation aligns with the reference value and the posterior distribution is narrow, indicating low uncertainty in the estimated value. The uncertainty in Fig. 1 is presented via 90% credibility intervals that can be easily derived from the posterior distribution.

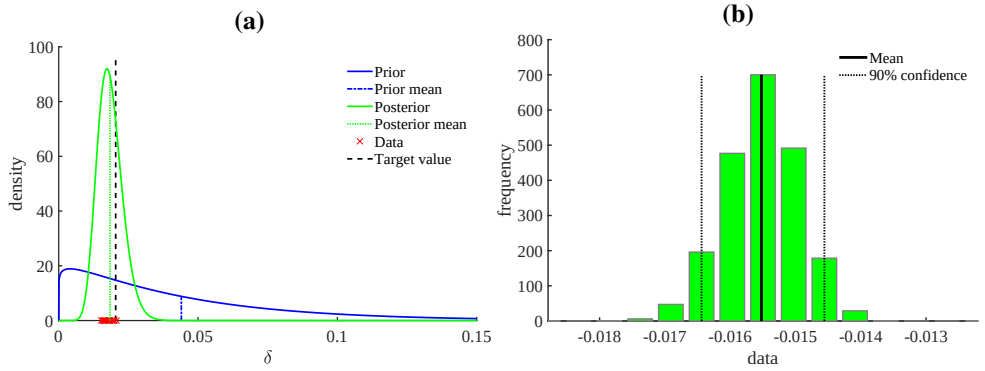



**Fig. 2:** Bayesian damping estimation a) and bootstrap b) for the segment 2 of the response.

# 4 Implemetation in Python

The initial parameters required by the user are: the natural frequency of the first dominant mode $\omega_\mathrm{d}$, the damping ratio $\zeta$ to be monitored, the segment length $k$ a number of periods on the natural frequency that is set automatically in the range of $8 \leq k \leq 15$, depending on $\zeta$ [3] and optionally the use of a bandpass filter to isolate the monitored natural frequency. The signal is divided in segments of length $t_\mathrm{seg} = 2k\pi/\omega_\mathrm{d}$. The first step is identification of peaks inside the chunk calling the SciPy built-in function:

```
scipy.signal.find_peaks(sig, height=h, distance=dist)
```

where, `h` sets the minimal height of the peak is set to 0 and `dist` is a minimal distance between peaks in samples set as: $2\pi f_\mathrm{s}/\omega_\mathrm{d} \cdot 0.9$. Then difference between peaks is calculated, that are stored in the array `peaks_dif`, which must be converted to a sequence and the SciPy function is called:

```
scipy.stats.bootstrap(peaks_dif, statistic=np.mean, method='percentile')
```

The function returns a `ConfidenceInterval`, whose mean value must be negative in case of a free response. Difference of the confidence interval should be less then threshold, which is set 1.5–2 times higher then the difference value obtained for the almost ideal free response. This leads to coarse or soft criterion for selection of the free response. Once the free response is acknowledged, the damping is to be identified together with the uncertainty interval based on the gamma distribution. It is identified from the logarithmic decrement of the identified peaks that should coincide with the gamma-distribution. The mean value of the gamma-distribution is the identified value, and uncertainty is defined as a 90% confidence interval:

- Posterior distribution is determined with:

```
posterior = scipy.stats.gamma.pdf(delta, alpha_i, scale=1/kappa_i)
```

- Confidence interval 90%:

```
CI_delta = scipy.stats.gamma.interval(0.9, alpha_i, scale=1/kappa_i)
```

where, `alpha_i` and `kappa_i` are hyper parameters defined with Eq. (2).

# 5 Conclusion

In this paper, an efficient monitoring for automatic free response detection and damping estimation is presented, specifically targeting systems with a dominant first mode that operates intermittently. Damping is selected as a key indicator since it strongly correlates with the degradation process. The monitoring integrates the bootstrap technique and Bayesian analysis, using the conjugate gamma-exponential model for damping estimation. Since this model is only applicable to free-response events, the bootstrap technique is used to detect these events before applying Bayesian analysis. As shown, the approach effectively identifies free response events and provides accurate damping estimates and is ready to use via an open source Python package.

# 6 ACKNOWLEDGEMENTS

The authors gratefully acknowledge financial support from the Croatian Science Foundation (HRZZ-IP-2022-10-8856).

## References


[1] J. Z. Sikorska, M. Hodkiewicz, and L. Ma, "Prognostic modelling options for remaining useful life estimation by industry," *Mechanical Systems and Signal Processing*, vol. 25, pp. 1803–1836, 7 2011.

[2] C. Modena, D. Sonda, and D. Zonta, "Damage localization in reinforced concrete structures by using damping measurements," *Key Engineering Materials*, vol. 167-168, pp. 132–141, 1999.

[3] S. Perišić, J. Barle, I. Tomac, and P. Djukić, "Automatic damping estimation via bootstrap technique and bayesian analysis for mechanical system condition monitoring," *Mechanical Systems and Signal Processing*, vol. 220, p. 111654, 11 2024.

[4] D. N. Politis, "Computer-intensive methods in statistical analysis: An introduction to the bootstrap, the jackknife, and other household items in a statistician's toolbox," *IEEE Signal Processing Magazine*, vol. 15, pp. 39–55, 1998.

[5] A. M. Zoubir and B. Boashash, "The bootstrap and its application in signal processing: An attractive tool for assessing the accuracy of estimators and testing hypothesis for parameters in small data-sample situations," *IEEE Signal Processing Magazine*, vol. 15, pp. 56–76, 1998.

[6] K.-V. Yuen, *Bayesian Methods for Structural Dynamics and Civil Engineering*. John Wiley and Sons, 2010.

[7] J. Barle, D. Ban, and M. Ladan, "Maritime component reliability assessment and maintenance using bayesian framework and generic data," in *Advanced Ship Design For Pollution Prevention* (C. G. Soares and J. Parunov, eds.), pp. 181–188, Taylor and Francis, 2010.

# Modal Parameter Estimation Framework with pyFBS: A Practical Example

Miha Pogačar[a] Sara Janjac[b] Gregor Čepon[a] *

[a] *University of Ljubljana, Faculty of Mechanical Engineering*
[b] *Gorenje gospodinjski aparati d.o.o.*

**Abstract**

This paper presents a case study on the modal parameter estimation of a laboratory structure, employing pyFBS as the primary analytical tool. The study highlights the practical application of pyFBS's suite of modal analysis tools, aiming to provide a comparative understanding of the available techniques for identifying key modal parameters. The case study demonstrates two distinct approaches to estimating poles and modal participation factors. Additionally, the process of identifying mode shapes is examined, with specific emphasis on mass normalization techniques. This paper offers a step-by-step exploration of pyFBS's capabilities and limitations, positioning it as a practical, adaptable toolset for researchers and engineers engaged in experimental modal analysis.

**Keywords:** Experimental modal analysis, Polyreference modal identification, pyFBS, Structural Dynamics, Python

## 1 Introduction

Dynamic substructuring has seen significant advancements in recent years, driven by growing interest in estimating the dynamic behavior of assemblies based on knowledge of their individual components. Among various methodologies, the two most commonly employed approaches are formulated in the frequency domain and the modal domain. The pyFBS framework[1, 2] originated as an open-source initiative with the aim of developing a Python package for Frequency-Based Substructuring. This effort was the result of a collaboration between the Laboratory for Dynamics of Machines and Structures at the University of Ljubljana, Faculty of Mechanical Engineering, and the Chair of Applied Mechanics at the Technical University of Munich.
As the project evolved, it became apparent that the library needed to be extended to support interactions between different substructuring domains (Fig. 1). Recently, the package has been expanded to include support functions for modal substructuring. These enhancements enable the integration of diverse numerical and experimental models, facilitating the combined application of various substructuring techniques. To support this expanded scope, the package required an open-source solution for polyreference modal identification, leading to the development of a dedicated module, `modal_id`.

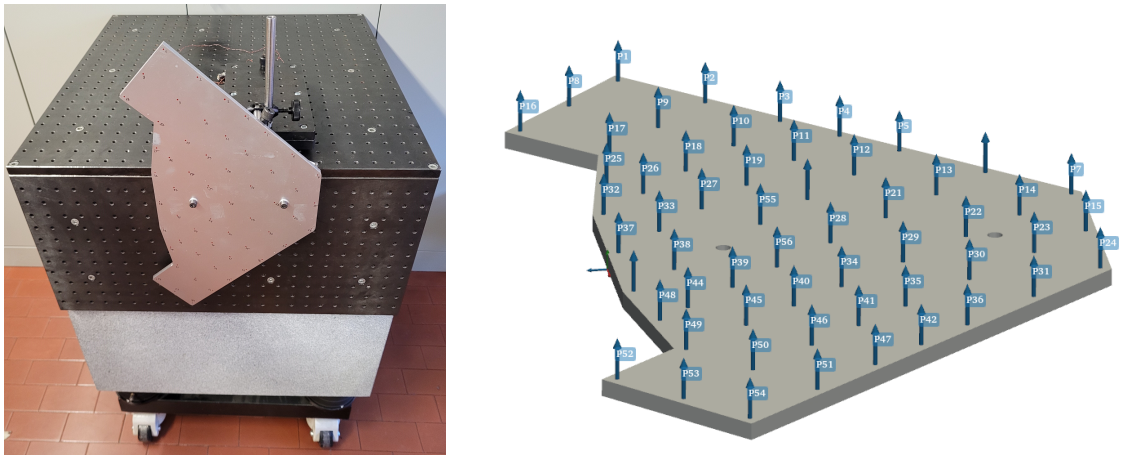
*Corresponding author, Email address: gregor.cepon@fs.uni-lj.si


**Fig. 1:** New pyFBS logo, reflecting expanded support beyond frequency-based methods.

The capabilities of the `modal_id` module are demonstrated through a case study involving a laboratory wing structure. The module includes two implementations for the identification of poles and corresponding modal participation factors: the Least Squares Complex Frequency (LSCF) method[3] and the Saanathan–Korener Rational Approximation (SKRA) family of methods[4, 5]. Additionally, the module supports Least Squares Frequency Domain (LSFD) techniques for mode shape estimation, offering both mass-normalized real mode shapes and a-normalized complex mode shapes.

## 2 Polyreference Modal Identification Using pyFBS

This case study involves polyreference experimental modal analysis of a lightly damped laboratory test structure. The specimen is a steel sheet metal structure, measuring 450 mm in height and 225 mm in width, as shown in Fig. 2a. The structure was mounted using two M6 bolts through pre-drilled holes and connected to a vibration-isolated base via aluminum bushings. Velocity response measurements were acquired at 56 locations on the front surface using a Polytec QTec Scanning Vibrometer. The measurement degrees of freedom (DoFs) are illustrated in Fig. 2b. Excitation was applied using a PCB-086E80 modal hammer at the DoFs labeled 1, 7, and 16.




**(a)** **(b)**

**Fig. 2:** Case study: (a) Photograph of the experimental setup; (b) Schematic of measurement DoFs used for modal testing.

Modal parameter estimation was performed using the polyreference modal identification module available in the open-source `pyFBS` package. The frequency range of 0–6 kHz is considered, enabling the identification of over 20 experimental mode shapes, along with their corresponding natural frequencies and modal participation factors.

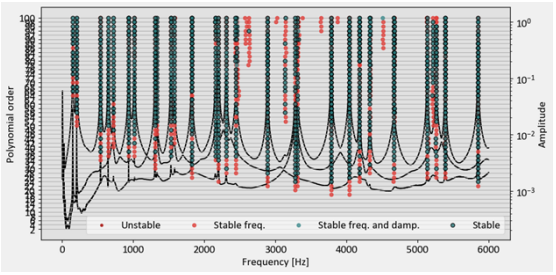
## 2.1 Identification of Poles and Modal Participation Factors

The first part of the example demonstrates the use of the `modal_id` function `pLSCF`, which implements the Polyreference Least Squares Complex Frequency (pLSCF) method [3]. This approach often benefits from higher polynomial model orders to ensure robust pole identification; in this case, a maximum order of 100 is applied. Alternatively, the recently integrated `pLSRA` function, based on the open-source `polyrat` project, employs least squares rational approximation techniques. While its iterative fitting procedures (iterative, and stabilized iterative) can be computationally more demanding, they generally require lower model orders and are performed with fewer evaluations to yield reliable results.

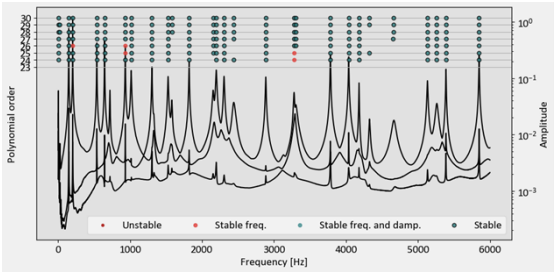
```
# Option 1: Perform identification of poles and modal participation factors with pLSCF
_id = pyFBS.modal_id(freq[:freq_limit], Y[:freq_limit])
_id.pLSCF(max_order=100)
_id.stabilization()

# Option 2: Perform identification of poles and modal participation factors with pLSRA
_id = pyFBS.modal_id(freq[:freq_limit], Y[:freq_limit])
_id.pLSRA(max_order=30, sol_type = 'iterative')
_id.stabilization()

# access poles
_id.selected_poles
# access modal participation factors
_id.selected_mpf
```

**Listing 1:** Code snippet – using pLSCF or pLSRA.

Executing the code examples below results in the stability charts shown in Fig. 3, illustrating typical outcomes for both the pLSCF and pLSRA methods. In this step, users can interactively select relevant poles from the stabilization diagram, after which the corresponding modal participation factors are automatically evaluated.




**(a)** Stability chart – pLSCF method




**(b)** Stability chart – pLSRA method

**Fig. 3:** Stability charts: a) Generated using pLSCF; b) Generated using pLSRA.

## 2.2 Identification of Mode Shapes

The second step involves the polyreference identification of mode shapes using a least squares approach, implemented via the `pLSFD` function. Two techniques are available:

- The first assumes a proportionally damped system admittance, enabling the evaluation of real, mass-normalized mode shapes.

- The second assumes a generally viscously damped system admittance, allowing for the estimation of complex, non-normalized mode shapes.

The following example demonstrates the use of the first approach, including how to access both normalized and non-normalized mode shapes. Note that the normalization step is optional and supports polyreference normalization based on a least squares fit.

```
# Perform idenfitication of real modeshapes assuming proportional damping
_id.pLSFD(frf_type = 'mobility', assume_proportional=True)

# access modal constants
_id.A.shape
# access non-normalized output modeshapes
output_shapes = _id.A[:,:,0].T
# access non-normalized input modeshapes
input_shapes = _id.A[:,0,:].T

# Mass normalize modeshapes
output_dp_ind = np.array([0,6,15])
input_dp_ind = np.array([0,1,2])
_id.normalize(output_dp_ind, input_dp_ind, check_dp=True)

# access mass normalized output modeshapes
_id.Phi_o
# access mass normalized input modeshapes
_id.Phi_i
```

**Listing 2:** Code snippet – using pLSFD.

## 3 Conclusion

The `pyFBS` package offers a flexible, open-source solution for experimental modal analysis with a strong focus on polyreference methods. Through this case study, key capabilities were demonstrated for estimating poles, modal participation factors, and both normalized and non-normalized mode shapes. By supporting multiple domain-specific algorithms – such as `pLSCF`, `pLSRA`, and `pLSFD` – `pyFBS` allows for robust, customizable workflows adaptable to various structural dynamics scenarios. This highlights its practical value for researchers and engineers engaged in component mode synthesis and structural health monitoring.

## References


[1] T. Bregar, A. El Mahmoudi, M. Kodrič, D. Ocepek, F. Trainotti, M. Pogačar, M. Göldeli, G. Čepon, M. Boltežar, and D. J. Rixen, "pyFBS: A python package for frequency based substructuring," *Journal of Open Source Software*, vol. 7, no. 69, p. 3399, 2022.

[2] "pyFBS repository." `https://gitlab.com/pyFBS/pyFBS`.

[3] P. Verboven, "Frequency-domain system identification for modal analysis," *Vrije Universiteit Brussel, Brussels*, vol. 665, 2002.

[4] J. M. Hokanson, "Multivariate rational approximation using a stabilized sanathanan-koerner iteration," *arXiv preprint arXiv:2009.10803*, 2020.

[5] "polyrat repository." `https://github.com/jeffrey-hokanson/polyrat`.

# OpenPulse: An Open Source Software for Acoustically Induced Vibration Analysis of Piping Systems

Olavo M. Silva * Jacson G. Vargas André S. Fernandes Vitor V. Slongo
Rodrigo Schwartz Gildean. N. Almeida Vinícius H. Ribeiro Gustavo C. Martins

*Federal University of Santa Catarina, Florianópolis, Brazil*

**Abstract**

Acoustically Induced Vibration (AIV) represents a critical risk to the integrity of gas piping systems pressurized by reciprocating compressors in the oil and gas industry. This article introduces *OpenPulse*, an open-source Python software developed to facilitate the numerical modeling of low-frequency AIV. With the aid of a graphical user interface (GUI), the modeling process begins with the definition of system geometry, material properties, and cross-sectional parameters, followed by the application of boundary conditions such as displacement constraints, nodal pressure, acceleration, and external forces. These loads can be either experimental or derived from theoretical models. A thermo-fluid-dynamic model of a reciprocating compressor is also implemented. Time-harmonic acoustic analysis is then performed within the one-dimensional acoustic domain, under the assumption of plane waves and using the Finite Element Transfer Method (FETM). The resulting pressure field is subsequently applied as an equivalent internal distributed load to the structural piping model, modeled based on the Timoshenko beam theory and solved using the Finite Element Method (FEM). Visualization tools, including animations, colormaps, and frequency-domain plots assist users in analyzing the computed AIV results.

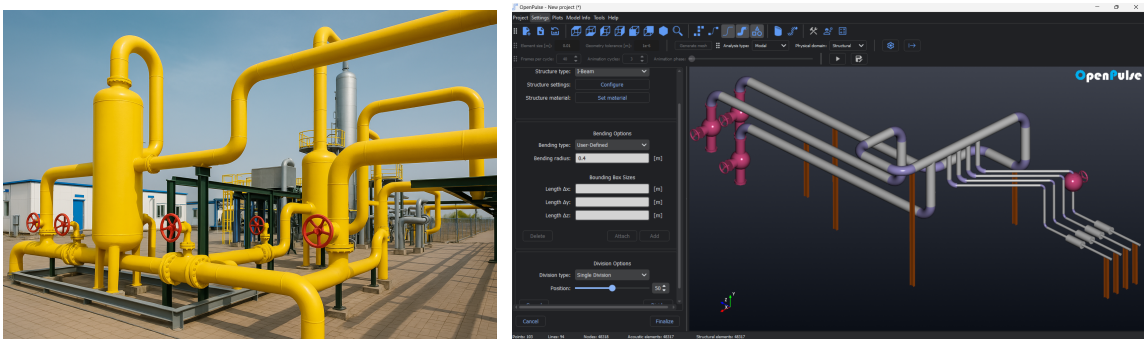
**Keywords:** Acoustically-Induced Vibrations, Low-Frequency Pulsation, Reciprocating Compressors, Finite Element Transfer Method, Piping systems, Timoshenko Beam, Finite Element Method

## 1 Introduction

Reciprocating compressors represent an important excitation source for gas piping systems in oil refinery plants. The structural behavior of the pipes can be strongly affected by the acoustic response of the internal acoustic field represented by the gas flow. This kind of excitation is known as Acoustically Induced Vibration (AIV) and is an important vibration mechanism in piping structures, which respond to both unsteady and pulsating flow fields [1]. Acoustic and structural responses of pipes arise from low-frequency internal distributed loads induced by acoustic pulsations at low-order harmonics of typical compressor discharge and suction pressure spectra.

When acoustic and structural natural frequencies coincide, vibrations can amplify dramatically, especially for reciprocating compressors where harmonic energy is concentrated below 100 Hz [2]. To mitigate these effects, standards such as the API 618 [3] guide the design and analysis of safe and reliable pipeline systems that ensure adequate structural and acoustic performance [4]. These

*Corresponding author, Email address: olavo@lva.ufsc.br



**(a)** Virtual representation of a piping system. **(b)** OpenPulse interface.

**Fig. 1:** Illustrative figures of a typical piping system and software interface. Source: authors.

analyses compare predicted values to allowable limits to prevent fatigue failures, in accordance with, for instance, the Energy Institute guidelines [2].

Refineries commonly have kilometers of piping and hundreds of components such as pulsation chambers, valves, and others, as illustrated in Fig. 1a. In this context, methodologies to predict the vibration behavior of such systems must be both simplified and representative. Approximate 1D acoustic models can be applied to calculate time-domain solutions for AIV [5]. The most common methods are the Method of Characteristics (MOC) [6] and the Finite Element Method (FEM) [7]. On the other hand, the structural response of the pipes is often obtained by FEM [8].

Assuming that the most energetic pulsation harmonics of reciprocating compressors occur at frequencies below 100 Hz, one can assume the associated structural wavelengths are long enough to represent only global vibrations, rather than local wall vibrations. It is reasonable that most structural analyses of such systems assume that the pipes are made of elastic-isotropic materials, with lengths much greater than their diameters and relatively low deflections. This fact enables one to consider the piping system as a set of beams.

Regarding acoustic-structure interaction, most engineers estimate equivalent shaking forces at elbows based on acoustic pressure, applying point forces aligned with the direction of the corresponding equivalent force area. This procedure is generally not automated and typically involves a manual calculation of the forces to be applied to the structural model.

In this article, the authors present *OpenPulse* [9], an open-source Python software that numerically models low-frequency AIV. Through a graphical user interface (GUI), the software allows users to define the pipeline route and geometry, material properties, cross sections, and loads. It performs a time-harmonic acoustic response analysis of the one-dimensional acoustic domain using the Finite Element Transfer Method (FETM)[10], and automatically applies the resulting pressure field as internal distributed loads to the structural piping system (weak coupling), which is modeled using Timoshenko beam theory and FEM [8]. A screenshot of the GUI used for pipeline input is shown in Fig. 1b.

## 2 Theoretical background

For a given pipe element (see Fig. 2) and considering acoustics only, the FETM arranges the nodal pressures $p_1$ and $p_2$, and volume velocities $q_1$ and $q_2$, obtained from the Transfer Matrix Method [10], into separate vectors and transforms the transfer matrix of the acoustic element into a mobility matrix.

This mobility matrix depends on fluid and geometric properties such as sound speed $c_f$, density $\rho_f$, and cross-sectional area $S_f$. It can be written, for a given angular frequency $\omega$, as [10]

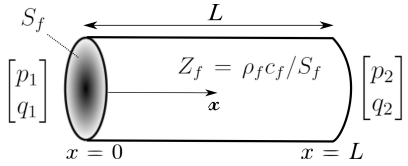



**Fig. 2:** Uniform tube, 1D acoustic element, with length $L$ (adapted from [10]).

$$\begin{bmatrix} -\mathrm{j}\cot(kx)/Z_f & \mathrm{j}/Z_f \sin(kx) \\ \mathrm{j}/Z_f \sin(kx) & -\mathrm{j}\cot(kx)/Z_f \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad (1)$$

where $k = \omega/c$ is the wavenumber, $Z_f = \rho_f c_f / S_f$ is the impedance, and j is the imaginary unit.

To enable a programmable implementation, it is advantageous to combine FETM with numerical strategies from FEM [10]. In *OpenPulse* acoustics approach, only the "concept of element" and the prescription of boundary conditions – such as nodal pressure, volume velocity, and impedance – are inherited from FEM. Meanwhile, the mobility matrices are derived analytically using FETM, based on wave propagation theory in time-harmonic response. The resulting formulation yields a two-node element with one degree of freedom per node (pressure). In *OpenPulse*, Eq. (1) is further refined by using dissipation models such as the Low-Reduced Frequency (LRF) model [11] and other models for dissipation at low Mach numbers. In addition, the software contains various "pre-modeled" elements, such as perforated plates, pulsation suppression devices, and other components. The reciprocating compressor is modeled using a transient thermodynamic approach, whose time-domain response is processed via FFT and applied as a frequency-dependent volume velocity boundary condition.

The objective of *OpenPulse* is to predict the harmonic acoustic response of piping systems using FETM, as well as their global vibration response under low-frequency distributed loads, represented by internal acoustic pulsations corresponding to the first harmonics of typical reciprocating compressor spectra. The methodology assumes that the pipes can be modeled through the well-known Timoshenko beam theory with FEM (two-noded element, with three translations and three rotations per node).

A weak coupling between structural and acoustic fields is considered, where axial stress, $\sigma_1$, radial stress, $\sigma_r$, and circumferential stress, $\sigma_c$, on the structure pipe wall, derived analytically from cross-section dimensions and internal pressure [12, 5], are converted into axial forces. In this case, for a given structural element, the axial equivalent forces acting at both ends can be found, and the additional element load vector $\mathbf{f}_p^e$, which must be added to the external forces vector, is given by, assuming $\nu$ as the Poisson coefficient of material,

$$\mathbf{f}_p^e = \big[-F_x, 0, 0, 0, 0, 0, F_x, 0, 0, 0, 0, 0\big]', \quad \text{where} \quad F_x = S_f\big[\big(\sigma_1 - \nu(\sigma_r + \sigma_c)\big)\big]. \quad (2)$$

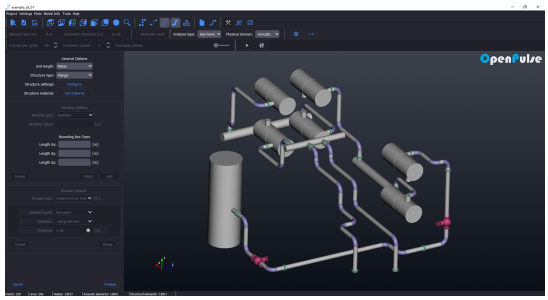
The "weak" (one-way) coupling is assumed based on the following hypotheses: the fluid is a gas; the acoustic plane wave propagates axially; the pipe is thin-walled and linearly elastic; the radial inertia of the pipe wall is neglected for low frequencies; and the acoustic wave speed $c_f$ in the gas is influenced by the mechanical compliance of the pipe wall, given by $c_f = \sqrt{\left(\frac{\rho_f}{K_f}\left(1 + \frac{D_i K_f}{Et}\right)\right)}$, where $K_f$ is the bulk modulus of the fluid, $E$ is the Young's modulus of the pipe material, $D_i$ is the internal diameter, and $t$ is the pipe wall thickness. In *OpenPulse*, the same mesh is utilized for both the acoustic analysis (solved first) and the structural analysis.

## 3 Python implementation

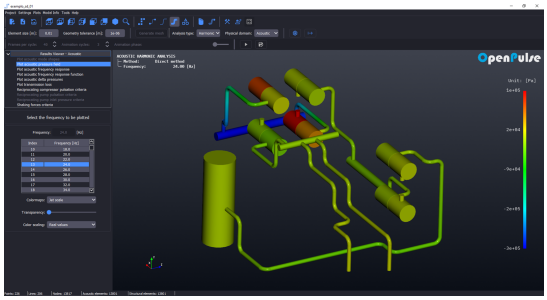
*OpenPulse* is developed in Python, leveraging a range of libraries to ensure functionality and efficiency. The GUI and application control are implemented using PySide6, while VTK is employed for 3D graphical rendering. Gmsh serves as the mesh processor, enabling the creation and manipulation. Numerical computations are handled by NumPy, SciPy, and PyPardiso. For visualization of data, Matplotlib is used to generate 2D plots, while data interfaces rely on Pandas and H5py to manage and store structured datasets. Finally, dependency management and environment configuration are performed with Poetry. The source code of *OpenPulse* is available on GitHub (https://github.com/openpulse/OpenPulse) and published under GPL v3 license. For each release, an installation file is provided, eliminating the need to interact with Python code if the user prefers to work this way.

## 4 Some screenshots of application

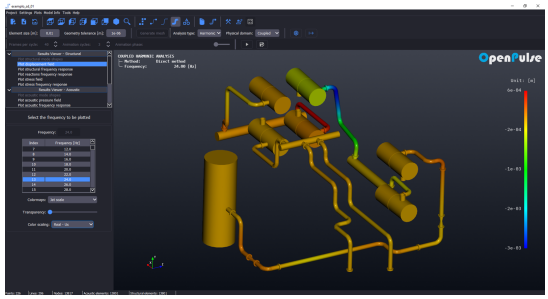
The screenshots in Fig. 3 illustrate the application of *OpenPulse* to the analysis of a piping system subjected to excitation by three-stage reciprocating compressors.



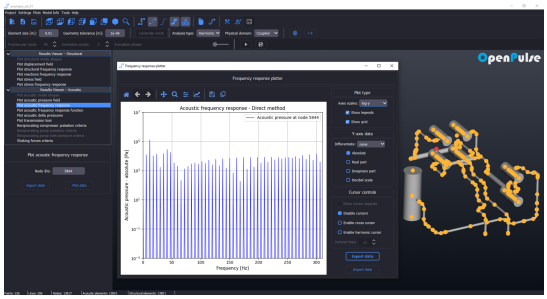
(a) Geometry and component design.



(b) Resultant pressure field.



(c) Resultant structural field.



(d) Frequency response of pressure at a node.

**Fig. 3:** Some screenshots of *OpenPulse*.

## 5 Conclusions

This manuscript introduces the main physical and methodological foundations of *OpenPulse*, an open-source software designed to analyze low-frequency AIV in piping systems. While it remains under

development, initial results obtained by partners like PETROBRAS have shown reliable accuracy and can be validated against commercial FEM software. Future research will focus on comparisons with experimental data.

## References


[1] S. Kaneko, T. Nakamura, F. Inada, M. Kato, K. Ishihara, T. Nishihara, and M. A. Langthjem, eds., *Flow-induced Vibrations*. Oxford: Academic Press, second edition ed., 2014.

[2] *Guidelines for the avoidance of vibration induced fatigue failure in process pipework*. Energy Institute, London, UK, 2nd edition ed., January 2008.

[3] "Reciprocating compressors for petroleum, chemical, and gas industry services," API 618 - 5th edition, American Petroleum Institute (API), Washington DC, United States, Dec. 2007.

[4] P. P. Shejal and A. D. Desai, "Pulsation and vibration study of reciprocating compressor according to api 618 5 th edition," 2014.

[5] O. Silva, D. Tuozzo, J. Vargas, L. Kulakauskas, A. Fernandes, J. Souza, A. Rocha, A. Lenzi, R. Timbó, C. Mendonça, and A. Brandão, "Numerical modelling of low-frequency acoustically induced vibration in gas pipeline systems," in *Proc. 29th International Conference on Noise and Vibration engineering (ISMA2020) - At: Online - Virtual Plataform*, 09 2020.

[6] S. Li, B. W. Karney, and G. Liu, "Fsi research in pipeline systems – a review of the literature," *Journal of Fluids and Structures*, vol. 57, pp. 277 – 297, 2015.

[7] A. Coulon, R. Salanon, and L. Ancian, "Innovative numerical fatigue methodology for piping systems: qualifying acoustic induced vibration in the oil & gas industry," *Procedia Engineering*, vol. 213, pp. 762–775, 2018. 7th International Conference on Fatigue Design, Fatigue Design 2017, 29-30 November 2017, Senlis, France.

[8] T. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Civil and Mechanical Engineering, Dover Publications, 2000.

[9] O. Silva, D. Tuozzo, J. Vargas, L. Kulakauskas, A. Fernandes, J. Souza, A. Rocha, A. Lenzi, R. Timbó, C. Mendonça, and A. Brandão, "Openpulse: Open source software for pulsation analysis of pipeline systems." %urlhttps://open-pulse.github.io/OpenPulse/, 2021.

[10] D. M. Tuozzo, O. M. Silva, L. V. Kulakauskas, J. G. Vargas, and A. Lenzi, "Time-harmonic analysis of acoustic pulsation in gas pipeline systems using the finite element transfer matrix method: Theoretical aspects," *Mechanical Systems and Signal Processing*, vol. 186, p. 109824, 2023.

[11] H. Tijdeman, "On the propagation of sound waves in cylindrical tubes," *Journal of Sound and Vibration*, vol. 39, no. 1, pp. 1 – 33, 1975.

[12] A. Boresi, K. Chong, and J. D. Lee, *Elasticity in Engineering Mechanics*. Wiley, 2010.

# FatigueDS: A Python Package for Fatigue Damage Spectrum and Extreme Response Spectrum Analysis

Jaša Šonc Martin Česnik Rok Pavlin Janko Slavič *

*University of Ljubljana, Faculty of Mechanical Engineering*

**Abstract**

The FatigueDS Python package provides an open–source implementation of the *Extreme Response Spectrum* (ERS) and *Fatigue Damage Spectrum* (FDS) concepts that underpin modern vibration–fatigue qualification of mechanical systems. It enables engineers to derive a frequency–domain test specification from arbitrary excitation records—random, sine, or sine–sweep—defined either as time histories or as power–spectral densities and to translate the specification into a cumulative damage metric that is directly comparable with traditional stress–life (*S–N*) data. The package offers a concise, NumPy–aware API, extensive documentation, and reference notebooks that shorten the path from raw data to decision–ready plots

**Keywords:** Fatigue Damage Spectrum, Extreme Response Spectrum, Open Source, Python, Specification development

## 1 Statement of Need

In modern engineering design, fatigue analysis is a cornerstone for predicting the life and reliability of components subjected to dynamic loading. This is particularly relevant in automotive, aerospace, civil, and mechanical engineering applications, where components are often exposed to a variety of cyclic loads. FatigueDS [2] addresses this need by offering a robust and flexible computational toolkit that supports educational, scientific and industrial use cases.

Traditional fatigue analysis tools often require significant setup effort and domain-specific expertise. FatigueDS fills a niche for accessible, Python-based tools that integrate seamlessly with scientific computing ecosystems. By leveraging Python's capabilities and an intuitive API, the package accelerates the fatigue analysis process while maintaining scientific rigor. Because FatigueDS is distributed as pure Python, the entire pipeline—from raw accelerometer signal to PDF report—may be scripted and version-controlled.

*Corresponding author, Email address: janko.slavic@fs.uni-lj.si

## 2 Theoretical Background

The Fatigue Damage Spectrum (FDS) and Extreme Response Spectrum (ERS) [1] are metrics derived from the dynamic response of single-degree-of-freedom (SDOF) systems subjected to an excitation signal. The FDS quantifies the accumulated fatigue damage a component might experience across a range of resonant frequencies. This is achieved by simulating an array of SDOF systems, each tuned to a different natural frequency and damping ratio, and processing their responses.

ERS, on the other hand, evaluates the maximum peak response across frequencies, providing insight into the structural response to transient events. Both spectra are essential for designing test profiles in shaker testing, understanding failure modes, and assessing design robustness.

Fatigue analysis relies on the rainflow counting algorithm and Miner's rule, which together translate complex loading histories into equivalent damage cycles. FatigueDS incorporates these methods with configurable fatigue parameters, such as the Basquin law coefficients $b$ and $C$, and the stiffness coefficient $K$.

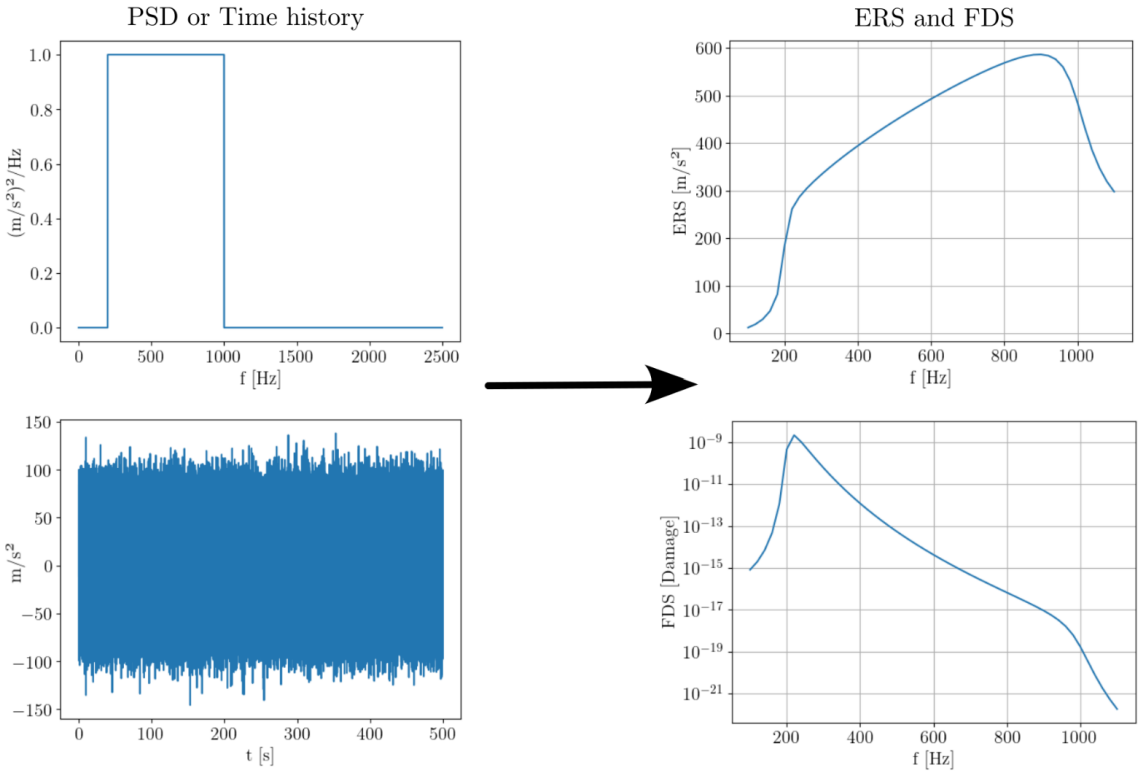



**Fig. 1:** Concept of signal evaluation using FDS and ERS.

## 3 Features and Capabilities

FatigueDS supports the analysis of:

- **Random Signals**: Defined either in the frequency domain via Power Spectral Density (PSD) or

in the time domain as time histories. Useful in simulating operational vibrations.

- **Sine and Sine-Sweep Signals**: Essential for harmonic and modal excitation tests.

Advanced capabilities include:

- Customizable damping ratios and frequency ranges.
- High-resolution FFT and filtering support for preprocessing.
- Interactive plotting with Matplotlib integration.
- Modular architecture for extension into other spectrum-based analyses.

# 4 Methodology

FatigueDS computes spectral responses by discretizing the frequency domain and simulating the response of each virtual SDOF system. The simulation involves:

1. Constructing transfer functions for each frequency bin.
2. Applying input excitation using convolution or spectral multiplication.
3. Calculating peak responses and damage using rainflow counting.

The methodology aligns with standards presented in Lalanne's work [1] and incorporates best practices from the shock and vibration testing community.

# 5 Usage Examples

This section presents three examples of FatigueDS usage for different excitation signals.

## 5.1 Random signal (PSD)

```
import numpy as np
import pyExSi as es
import FatigueDS

# generate random signal
fs = 5000  # sampling frequency [Hz]
time = 1  # time duration [s]
freq_flat = np.arange(0, fs / 2, 1 / time)  # frequency vector
freq_lower = 200  # PSD lower frequency limit  [Hz]
freq_upper = 1000  # PSD upper frequency limit [Hz]
PSD_flat = es.get_psd(freq_flat, freq_lower, freq_upper, variance=800)
# one-sided flat-shaped PSD

import FatigueDS
```

```
# instantiate the SpecificationDevelopment class
# set the frequency range (start, stop, step) and damping ratio
sd = FatigueDS.SpecificationDevelopment(freq_data=(100, 1100, 20), damp=0.05)
# set the random load
sd.set_random_load((PSD_flat, freq_flat), unit='ms2', T=3600)
# input is PSD array and frequency array

# calculate the ERS and FDS
sd.get_ers()
sd.get_fds(b=10, C=1e80, K=6.3 * 1e10)
```

### 5.2 Random signals (time history)

Determining of a ERS and FDS of a random signal, defined in the time domain, can be done with two methods:

- Convolution (directly from time history, using rainflow counting)
- PSD averaging (converting time history to PSD and then to ERS and FDS)

```
import numpy as np
_time_data = np.load('test_data/test_time_history.npy', allow_pickle=True)
time_history_data = _time_data[:,1]
t = _time_data[:,0]
dt = t[2] - t[1]

import FatigueDS

# instantiate the SpecificationDevelopment classes
sd_1 = FatigueDS.SpecificationDevelopment(freq_data=(20, 200, 5))

# set the random loads (input is time history array and time step)
sd.set_random_load((time_history_data, dt), unit='g', method='convolution')
# list of methods: ['convolution', 'psd_averaging']

# calculate the ERS and FDS
sd.get_ers()
sd.get_fds(b=10, C=1e80, K=6.3 * 1e10)
```

### 5.3 Sine and sine-sweep signals

```
import numpy as np
import FatigueDS
import matplotlib.pyplot as plt
```

```
# instantiate classes
sd_sine = FatigueDS.SpecificationDevelopment(freq_data=(0, 2000, 5),
 damp=0.1)  # sine
sd_sine_sweep = FatigueDS.SpecificationDevelopment(freq_data=(0, 2000, 5),
 damp=0.1)  # sine sweep

# set the sine and sine-sweep loads
sd_sine.set_sine_load(sine_freq=500, amp=10, t_total=3600)  # t_total
 is required only for FDS calculation.
sd_sine_sweep.set_sine_sweep_load(const_amp=[5, 10, 20],
   const_f_range=[20, 100, 500, 1000], exc_type='acc', sweep_type='log',
   sweep_rate=1)

sd_sine.get_fds(b=10, C=1e80, K=6.3 * 1e10)
sd_sine_sweep.get_fds(b=10, C=1e80, K=6.3 * 1e10)
```

## 6 Conclusion

FatigueDS is a powerful and accessible Python library for analyzing the fatigue potential of dynamic signals through FDS and ERS. It supports diverse use cases, from academic research to industrial testing, and promotes reproducibility and transparency in structural fatigue analysis.

## Availability

The FatigueDS package is distributed under the MIT license and can be accessed at: `https://github.com/ladisk/FatigueDS`

## References


[1] C. Lalanne, *Mechanical Vibration and Shock: Specification Development*, ISTE Ltd and John Wiley & Sons, 2009.

[2] J. Šonc, M. Česnik, R. Pavlin, and J. Slavič, "FatigueDS: Fatigue Damage Spectrum and Extreme Response Spectrum in Python," *GitHub repository*, Feb. 2025. [Online]. Available: `https://github.com/ladisk/FatigueDS`

# Open-source software for Multiaxial fatigue damage calculation

Jaša Šonc, Janko Slavič

University of Ljubljana, Faculty of Mechanical Engineering

**Abstract**

The open-source effort FLife was launched in 2023 as a tool for calculating fatigue damage using 20 different methods in both the time and frequency domains. Recently, it was upgraded to include functionality for determining damage caused by multiaxial stress states. Several multiaxial criteria are available to convert multiaxial stress states into uniaxial equivalent stress states. Additionally, multipoint functionality has been introduced, allowing users to import entire FEM models and interactively select critical points for damage assessment.

**Keywords:** Fatigue damage, multiaxial stress, multiaxial fatigue criteria, uniaxial equivalent stress

## 1 Statement of need

In advanced engineering applications—ranging from automotive and aerospace structures to civil —components are routinely subjected to complex, multiaxial loading histories that give rise to multiaxial stress states. Traditional fatigue-life prediction methods are based on uniaxial stress states and are therefore not suitable for more complex stress states. The multiaxial stress states therefore require multiaxial criteria to be converted into the uniaxial equivalent stress. FLife offers a complete workflow by incorporating multiaxial criteria and damage accumulation methods in a single package.

## 2 Theoretical background

Vibrational fatigue represents one of the key damage mechanisms in structures subjected to prolonged dynamic loading [1]. It occurs because of repeated stress cycles that gradually accumulate damage even at stress levels well below the material's yield strength. Situations in which the excitation frequency approaches a structural natural frequency are particularly critical, as resonance can greatly amplify the response and sharply reduce the component's fatigue life [2]. Methods for vibration–fatigue analysis fall into two main categories: time-domain and frequency-domain. For random loadings—where the input is specified not deterministically but via a power-spectral-density (PSD) function—frequency-domain methods are most widely used in practice [3]. Modal decomposition further enhances computational efficiency by substantially reducing the problem's dimensionality [4]. For uniaxial stress states, numerous spectral-moment-based approaches have

been developed; among the most commonly applied are the Tovo–Benasciutti method [5], Dirlik's model [6], the Zhao–Baker approach [7], and the Jiao–Moan probabilistic method [8]. A comprehensive review of more than twenty such techniques is provided by Zorman et al. [9].

Over the past decade, significant research has focused on non-Gaussian processes [10], non-stationary loadings [11], and multiaxial excitations [12]. It has been shown that classical uniaxial approaches often fall short in reliably estimating fatigue life under these more complex conditions, highlighting the need for advanced methods and refined damage criteria.

Since high-cycle fatigue parameters, such as S–N curves, are determined experimentally for uniaxial stress states, it is necessary—when dealing with real multiaxial stress fields—to convert them into an equivalent uniaxial loading. To this end, various multiaxial criteria have been developed that enable the assessment of fatigue damage under general stress states [13, 12]. In structural dynamics, these criteria are most often formulated in the frequency domain. Some are defined to convert the full stress-tensor PSD matrix into the PSD of an equivalent uniaxial stress, while others transform the amplitude–frequency spectrum of the multiaxial stress state into an equivalent uniaxial stress spectrum. The most widely used criterion is the frequency-domain reformulation of the von Mises criterion—commonly referred to as EVMS (equivalent von Mises stress). This approximates the static von Mises criterion for application to power-spectral-density matrices and was proposed by Preumont and Piefort [14]. Another frequently used class comprises critical-plane criteria [12]. In these methods, a critical plane for damage initiation is identified—often via iterative procedures—and, on that plane, the maximum normal, shear, or combined stress amplitude is determined. More recent criteria proposed in the last few decades include the Carpinteri–Spagnoli criterion [15], an EVMS reformulation for phase-shifted stress-tensor components [16], and Nieslony's criterion, which combines hydrostatic and von Mises equivalent stresses [17]. A comprehensive review of many multiaxial criteria was published by Carpinteri et al. [18].

## 3 Features and Capabilities

FLife provides tools for fatigue life calculation by spectral methods and for equivalent stress calculation by employing multiaxial criteria for converting multiaxial stresses into uniaxial equivalent. It supports more than 20 damage **accumulation methods**:

*Narrowband, Wirsching Light, Ortiz Chen, Alpha 0.75, Tovo Benasciutti, Dirlik, Zhao Baker, Park, Jiao Moan, Sakai Okamura, Fu Cebon, modified Fu Cebon, Low, Gao Moan, Single moment, Bands method*

FLife supports more than 10 different **multiaxial criteria**:

*Maximum normal stress on critical plane, Maximum shear stress on critical plane, Maximum normal and shear stress on critical plane, EVMS (Equivalent von Misses stress), Carpinteri-Spagnoli criterion, Frequency-based multiaxial rainflow criterion, Thermoelasticity-based criterion, EVMS adaptation for out-of-phase components, Nieslony criterion, combining EVMS and hydrostatic stresses, Equivalent Lemaitre stress, LiWI approach, COIN-LiWI method*

# 4 Usage Examples

```python
import FLife
import numpy as np

dt = 1e-4
x = np.random.normal(scale=100, size=10000)

C = 1.8e+22  # S-N curve intercept [MPa**k]
k = 7.3      # S-N curve inverse slope [/]

# Spectral data
input_dict = {'time_history': x, 'dt': dt}
sd = FLife.SpectralData(input=input_dict)

# Rainflow reference fatigue life
# (do not be confused here, spectral data object also holds the time domain data)
rf = FLife.Rainflow(sd)

# Spectral methods
dirlik = FLife.Dirlik(sd)
tb = FLife.TovoBenasciutti(sd)

print(f'          Rainflow: {rf.get_life(C=C, k=k):4.0f} s')
print(f'            Dirlik: {dirlik.get_life(C=C, k=k):4.0f} s')
print(f'Tovo Benasciutti 2: {tb.get_life(C=C, k=k, method="method 2"):4.0f} s')
```

Listing 1: Fatigue-life calculation with FLife

```python
    import FLife
import numpy as np

# Load multiaxial PSD data
test_PSD = np.load('data/test_multiaxial_PSD_3D.npy')
freq=np.arange(0,240,3)
input_dict = {'PSD': test_PSD, 'f': freq}

# Create EquivalentStress object
eqs = FLife.EquivalentStress(input=input_dict,T=1,fs=5000)

# Use multiaxial criterion
eqs.EVMS()

# manual critical point selection
eqs.select_critical_point(point_index=331)

# GUI critical point selection
```

```
#FLife.visualize.pick_point(eqs)

# Define material properties
C = 1.8e+22  # S-N curve intercept [MPa**k]
k = 7.3  # S-N curve inverse slope [/]

# Calculate fatigue life in seconds
rf = FLife.TovoBenasciutti(eqs)
fatigue_life = rf.get_life(C=C, k=k)
print(f'Fatigue life: {fatigue_life:.2f} s')
```

Listing 2: Multiaxial fatigue-life calculation with FLife

## 5 Conclusions

FLife's new multiaxial extension lets users apply both invariant- and critical-plane criteria to convert 3D stress states into uniaxial stress equivalents. Its multipoint FEM import streamlines identification of critical locations and simplifies fatigue-life estimation.

## 6 Availability

The FLife package is distributed under the MIT license and can be accessed at: https://github.com/ladisk/FLife.

# References


[1] J. Slavič, M. Mršnik, M. Česnik, J. Javh, and M. Boltežar, *Vibration Fatigue by Spectral Methods*. Elsevier, 2021.

[2] O. Bernasconi and D. J. Ewins, "Modal strain/stress fields," *International Journal of Analytical and Experimental Modal Analysis*, vol. 4, pp. 68–76, 1989.

[3] M. Mršnik, J. Slavič, and M. Boltežar, "Frequency-domain methods for a vibration-fatigue-life estimation – application to real data," *International Journal of Fatigue*, vol. 47, pp. 8–17, 2013.

[4] M. Mršnik, J. Slavič, and M. Boltežar, "Vibration fatigue using modal decomposition," *Mechanical Systems and Signal Processing*, vol. 98, pp. 548–556, 2018.

[5] R. Tovo, "Cycle distribution and fatigue damage under broad-band random loading," *International Journal of Fatigue*, vol. 24, no. 11, pp. 1137–1147, 2002.

[6] T. Dirlik, *Application of computers in fatigue analysis*. PhD thesis, University of Warwick, 1985.

[7] W. Zhao and M. J. Baker, "On the probability density function of rainflow stress range for stationary gaussian processes," *International Journal of Fatigue*, vol. 14, no. 2, pp. 121–135, 1992.

[8] G. Jiao and T. Moan, "Probabilistic analysis of fatigue due to gaussian load processes," *Probabilistic Engineering Mechanics*, vol. 5, no. 2, pp. 76–83, 1990.

[9] A. Zorman, J. Slavič, and M. Boltežar, "Vibration fatigue by spectral methods—a review with open-source support," *Mechanical Systems and Signal Processing*, vol. 190, p. 110149, May 2023.

[10] D. Benasciutti and R. Tovo, "Spectral methods for lifetime prediction under wide-band stationary random processes," *International Journal of Fatigue*, vol. 27, no. 8, pp. 867–877, 2005.

[11] L. Capponi, M. Česnik, J. Slavič, F. Cianetti, and M. Boltežar, "Non-stationarity index in vibration fatigue: Theoretical and experimental research," *International Journal of Fatigue*, vol. 104, pp. 221–230, 2017.

[12] A. Niesłony and E. Macha, *Spectral Method in Multiaxial Random Fatigue*, vol. 33. Springer Berlin Heidelberg, 01 2007.

[13] M. Mršnik, J. Slavič, and M. Boltežar, "Multiaxial vibration fatigue—a theoretical and experimental comparison," *Mechanical Systems and Signal Processing*, vol. 76, 02 2016.

[14] A. Preumont and V. Piefort, "Predicting random high-cycle fatigue life with finite elements," *Journal of Vibration and Acoustics*, vol. 116, pp. 245–248, 04 1994.

[15] A. Carpinteri, A. Spagnoli, and S. Vantadori, "Reformulation in the frequency domain of a critical plane-based multiaxial fatigue criterion," *International Journal of Fatigue*, vol. 67, pp. 55–61, 2014.

[16] M. H. A. Bonte, A. de Boer, and R. Liebregts, "Determining the von mises stress power spectral density for frequency domain fatigue analysis including out-of-phase stress components," *Journal of Sound and Vibration*, vol. 302, p. 379–386, Apr. 2007.

[17] A. Niesłony, M. Böhm, R. Owsiński, A. Dziura, and K. Czekaj, "Integrating von mises and hydrostatic stresses in frequency domain multiaxial fatigue criteria for vibration fatigue analysis," *Mechanical Systems and Signal Processing*, vol. 224, p. 112229, 2025.

[18] A. Carpinteri, A. Spagnoli, and S. Vantadori, "A review of multiaxial fatigue criteria for random variable amplitude loads," *Fatigue and Fracture of Engineering Materials and Structures*, vol. 40, p. 1007–1036, May 2017.

# PyIDI.VideoReader: a module for the support of different multimedia formats

Ivan Tomac[1] * Klemen Zaletelj[2] Domen Gorjup[2] Janko Slavič[2]

[1] *Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture University of Split, Ruđera Boškovića 32, HR-21000 Split, Croatia*

[2] *Faculty of Mechanical Engineering, University of Ljubljana Aškerčeva cesta 6, SI-1000 Ljubljana, Slovenia*

**Abstract**

PyIDI - Python Image Displacement Identification is an open-source package developed by the structural dynamics community. The package was developed with the aim of providing an open-source displacement identification from high-speed video recordings, primarily aimed at researchers working on computer vision methods in structural dynamics. Two main displacement identification methods are included, one is simplified optical flow and the second is based on Lucas-Kanade pattern matching. The package is modular, allowing other researchers to contribute.

A major drawback of the package was the limited support of the input file format to mraw video files, which is tied to a specific camera manufacturer. To enable other video formats, such as image streams typically generated by rendering engines (e.g. png, tiff, jpg, etc.), video file formats generated by common cameras (avi, mp4, mov, etc.) or even image file formats that can contain multiple images (e.g. animated gifs or tiff), an additional module is introduced. By maintaining compatibility with the mraw file format, this module allows easy recognition of displacements from different file formats. The user only needs to specify a video file with a supported extension or the first frame from the stream and can then easily access all frames and perform displacement identification. PyIDI works with monochrome images, i.e. color images are converted to monochrome images by default, but the conversion can be omitted, and a single-color channel can be selected. The use of PyIDI with a new module is demonstrated with different examples to cover most of the assumed use cases.

**Keywords:** python, open source, pyIDI, multimedia

## 1 Introduction / Statement of Need

The ”Python Image Displacement Identification” package is an open source tool developed by the structural dynamics community [1, 2]. Its goal is to provide an open-source solution for identifying displacements from high-speed video recordings, aimed specifically at researchers starting out with computer vision techniques in structural dynamics. Two main methods for identifying displacements are included, one is simplified optical flow [3] and the second is based on the Lucas-Kanade [4] pattern

*Corresponding author, Email address: itomac@fesb.hr

matching algorithm. The package is modular so that other researchers can contribute to it, however up to now was tied to the image file format developed by Phortron, a manufacturer of high-speed cameras. To this end, we have integrated a VideoReader module into the PyIDI package that manages the reading of high-speed video recordings. The video recording can be any of the supported file formats, including image streams, video files or memory map for the "mraw" file format.

## 2 Background

Structural dynamic analysis requires straightforward access to pixel data in NumPy array format from video files of various multimedia types, including file streams. This process usually consists of several steps and using a single command from a library to read video/image files, such as Imageio, is not sufficient. Image streams require even more steps. Instead, simply provide an image/video file and have access to each image with a single command.

## 3 Module VideoReader

The module is integrated into the PyIDI package and is used to access video data by the displacement identification modules. The module can also be used to access images separately. The video recording is initialised by specifying the path to the video file, `cih(x)` file from Photron or `numpy.ndarray`. For the image stream, it is sufficient to specify the path to any image file in the sequence. The images in the stream must be in the same directory and named in such a way that they can be sorted in the correct order, *e.g.* for a stream of 10000 images, the file names should be as follows `im_0000.ext, ..., im_9999.ext`. Image formats that support multiple images, such as "gif", "tif", are also supported. The images are accessed by calling the method `get_frame()` from the class `reader`, here the example for the image stream, which is located in the subdirectory `image_stream`:

```
from pyidi import pyIDI
video = pyIDI('./image_stream/image_0000.png')
video.reader.get_frame(0)
```

The reader lists all png files in the subdirectory and returns the image according to the specified index, in this case the first image. The returned image is a 2D `numpy.array (height, width)` of type `numpy.uint8` or `numpy.uint16`, depending on the bit depth of the image file, *e.g.* images with 12 bit depth are returned as `numpy.array` of type `numpy.uint16`.

When the camera header file `cih(x)` is provided, which is stored in the subdirectory `camera_recording`, a PyMRAW module is called that generates a memory map. The internal methods for identifying displacements access the images using the `get_frame()` method, just like in the previous example. However, the memory map is also available:

```
from pyidi import pyIDI
video = pyIDI('./camera_recording/beam.cihx')
video.reader.mraw[0]
```

To access the first frame. In addition, the object `numpy.array` can be specified when initialising the object `video`. The colour images are also supported, but they are converted to a monochrome format. It is assumed that the images are in RGB colour format and they are converted to YUV format using only the Y channel. The user can select any channel by adding the letter 'R', 'G' or 'B' to the argument of the `get_frame()` method in the following way:

```
from pyidi import pyIDI
video = pyIDI('./image_stream/colour_image_0000.png')
video.reader.get_frame(0, use_channel='G')
```

This skips the conversion and only returns the green channel. The same procedure applies to video file formats such as mp4, avi etc.

# 4 CONCLUSION

Support for video formats and image streams extends the use of the package. The package is based on fundamental displacement identification methods that can now be used for any video source, including simulated video with rendering engines. The package is ideal for researchers beginning to explore computer vision methods in structural dynamics and can also be used for comparison with commercial displacement identification software.

# 5 ACKNOWLEDGMENT

The authors gratefully acknowledge partial financial support from the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie Grant Agreement No. 101027829, the Croatian Science Foundation (HRZZ-IP-2022-10-8856) and the Slovenian Research and Innovation Agency (research core funding No. P2-0263).

# References

[1] D. Gorjup, K. Zaletelj, and J. Slavič, "A hands-on tutorial on open-source research in structural dynamics and image-based experimental modal analysis (#10976)," in *IMAC-XXXIX*, 2 2021.

[2] D. Gorjup, K. Zaletelj, and J. Slavič, "A hands-on tutorial for image-based experimental modal analysis (#8033)," in *IMAC-XXXVIII*, 2 2020.

[3] J. Javh, J. Slavič, and M. Boltežar, "The subpixel resolution of optical-flow-based modal analysis," *Mechanical Systems and Signal Processing*, vol. 88, pp. 89–99, 5 2017.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

# Decomposition into Gaussian Portions – A python implementation for dealing with non-stationary random vibration in structural dynamics

Josef Weber * Arvid Trapp Peter Wolfsteiner

*University of Applied Sciences Munich*

**Abstract**

Numerous structural components across various fields of application are subjected to random vibrations during operation. To account for these loading conditions during the component design phase, fatigue damage assessment is performed based on reliable and representative load assumptions capturing the in-service use. Two primary approaches exist: time-domain- and PSD-based methods. While the former can capture effects of non-stationarity, they often entail enormous computational costs for statistically robust load assumptions. In contrast, PSD-based methods on the other side operate probabilistically and within the frequency domain, offering greater computational efficiency; however, they are limited to stationary Gaussian vibrations.

To overcome this limitation, the Decomposition into Gaussian Portions (DGP) approach is proposed. The fundamental idea of this approach is based on the theory that a non-stationary process can be approximated by multiple Gaussian segments, which together yield the same fatigue damage potential as the original signal. Essential for maintaining equivalent fatigue damage potential is that the quasi-stationary replacement load has the same effects on structural dynamics. Therefore, central to the DGP approach is that the quasi-stationary decomposition is carried out in the frequency-domain. This enables a PSD-based fatigue damage estimation for each individual Gaussian segment, resulting in higher computational efficiency while capturing a wide range of non-stationary effects.

**Keywords:** Random Vibration, Non-Stationarity, Structural Dynamics, Python

## 1 Introduction

Over their operational lifespan, structural components of road and railway vehicles are frequently subjected to random vibrations, which may induce fatigue damage. Consequently, reliable fatigue assessment is essential to enable the integration of appropriate structural measures during the design phase.

A widely adopted method for fatigue damage estimation is based on time-domain analysis, in which the anticipated loads are represented as time series of random processes. The finite element (FE) model is typically evaluated via modal-transient simulation. Fatigue damage is quantified by extracting load spectra at critical nodes using rainflow counting, followed by application of a damage

*Corresponding author, Email address: josef.weber@hm.edu

accumulation rule. Although this approach is broadly applicable to various types of stochastic excitation, a significant limitation lies in the considerable computational expense associated with rainflow counting. For large-scale FE models or long-duration input signals, these computational demands may become prohibitive, potentially rendering the methodology impractical for routine use.

A computationally efficient alternative is offered by power spectral density (PSD)-based methods [1]. In this approach, the excitation is represented by a PSD, allowing the FE model to be solved entirely in the frequency domain. Fatigue damage is subsequently estimated from the response PSDs. In contrast to time-domain methods, PSD-based approaches rely on a statistical characterization of the random process, thereby eliminating the need for rainflow counting and providing an elegant means for load spectrum estimation. A fundamental limitation of these methods, however, is their applicability being restricted to stationary Gaussian processes. When applied to non-stationary excitations, they tend to significantly underestimate fatigue damage, making them inappropriate for fatigue assessment under non-stationary conditions.

# 2 Decomposition into Gaussian Portions (DGP)

To circumvent these drawbacks, the Decomposition into Gaussian Portions (DGP) method is introduced. This approach is founded on the premise that a non-stationary random process can be approximated by a series of stationary Gaussian sub-processes that collectively retain the original signals fatigue damage potential [2].

Given that fatigue damage is strongly influenced by the resonant behavior of structural components, it is essential to preserve the frequency characteristics of the original process when decomposing it into Gaussian portions. Accordingly, the implementation of the DGP method incorporates an optimization algorithm that utilizes higher-order spectral information to account for non-stationarities. Specifically, the fourth-order moments are captured by a non-stationarity matrix [3], enabling a computationally efficient representation of the same in the frequency domain.

## 2.1 Python implementation

The DGP method is implemented in the open-source Python package `pyRaTS` (*Python package for processing Random Time Series for vibration fatigue*). The core concept of the package is the `tsarr` class, which represents a random process. It provides several methods for manipulating such processes, including the `est_DGP()` function for decomposition into Gaussian portions.

In the following, the application of the DGP method, as implemented in `pyRaTS`, is demonstrated using a time realization of a non-stationary random process.

## 2.2 Decomposition of a non-stationary time realization

Figure 1 shows the time realization of a random process. The data was obtained from acceleration measurements on railway vehicle components during operation. Owing to its high kurtosis value of $\beta_2 = 31.19$, the signal can be classified as a non-stationary process, making it a suitable candidate for the application of the DGP method. In this example, the signal is decomposed into $R = 6$ Gaussian sub-processes.

As previously discussed, the dynamic behavior of a structural component is crucial in fatigue life prediction. To assess the quality of the DGP approximation, the fatigue damage potential is evaluated using the Fatigue Damage Spectrum (FDS) and compared to that of the original signal (Figure 2). The

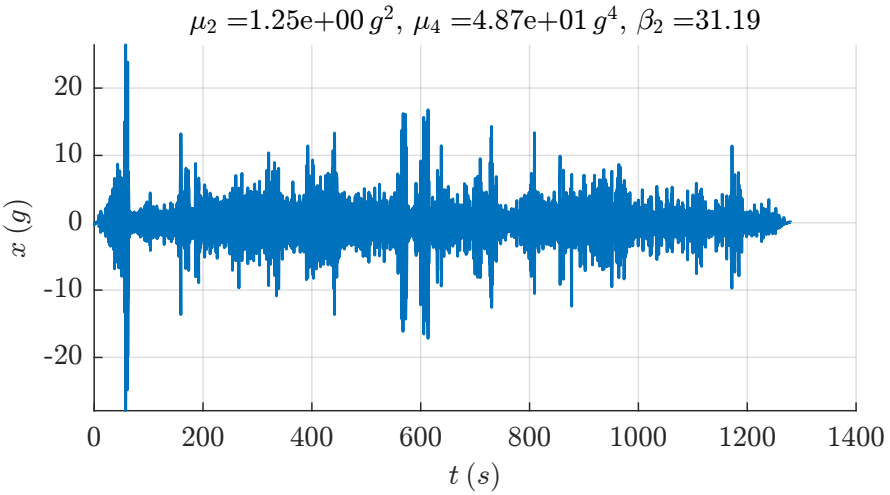



**Fig. 1:** Non-stationary time realization of a random process

reference response, obtained via transient analysis and subsequent rainflow counting of the original signal, is shown in blue. For comparison reasons, the response's damage-equivalent load amplitude is displayed. The DGP-based result is shown in yellow. Here, the damage contribution of each segment is derived from its respective PSD response and subsequently evaluated using the Dirlik method. As a further point of comparison, the damage estimate based on the average PSD is shown in red. The results clearly indicate that this approach leads to a substantial underestimation of the actual fatigue damage.

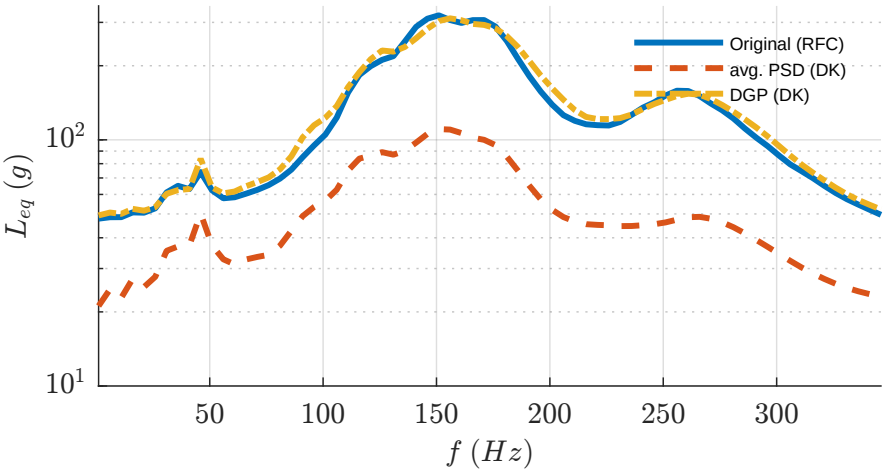



**Fig. 2:** FDS of original signal (blue, transient analysis and RFC), avg. DGP (red, PSD analysis and DK) and the DGP result (yellow, PSD analysis and DK)

This highlights the advantages of the DGP method, which enables the application of well-established PSD-based fatigue damage estimation techniques while avoiding the computational cost of cycle counting. At the same time, it yields a more accurate damage assessment compared to conventional PSD-based approaches when applied to non-stationary signals.

## References

[1] T. Dirlik and D. Benasciutti, "Dirlik and Tovo-Benasciutti Spectral Methods in Vibration Fatigue: A Review with a Historical Perspective," *Metals*, vol. 11, p. 1333, Aug. 2021.

[2] A. Trapp and P. Wolfsteiner, "Fatigue assessment of non-stationary random loading in the frequency domain by a quasi-stationary Gaussian approximation," *International Journal of Fatigue*, vol. 148, p. 106214, July 2021.

[3] A. Trapp and P. Wolfsteiner, "Frequency-domain characterization of varying random vibration loading by a non-stationarity matrix," *International Journal of Fatigue*, vol. 146, p. 106115, May 2021.

# SEP 005 : an Open-Source Effort for Unified Timeseries Data in Structural Dynamics

Wout Weijtjens[1*] Domen Gorjup[2] Janko Slavič[2]

[1]*OWI-lab, Faculty of Engineering, Vrije Universiteit Brussel, Belgium*

[2] *Ladisk, Faculty of Mechanical Engineering, University of Ljubljana, Slovenia*

**Abstract**

*Anyone who ever has worked with data acquisition hardware or software will acknowledge the task of storing data, writing a little bit of wrapper code to get the data to work in your preferred piece of software, just to discover that with a software update, a new piece of hardware, a different configuration, you can start the exercise all over again as your original code doesn't work anymore.*
*SEP005 is a small community effort inspired by conversations at the first OpenSD conference in 2023. What if we set up a small set of conventions on how timeseries data is to be structured for our usecases? Then we can build our code starting from this unified structure. Separate dedicated packages can translate common file types (.tdms, .dxd, ... ) to these conventions and can be easily maintained and updated. While the analysis packages can stay unchanged only requiring the data to comply to SEP005's rules.*

**Keywords:** Signal processing, Python, Data acquisition, Structural Health Monitoring

## 1 Statement of Need

The original concern and motivation for a unified timeseries came from experience. At OWI-lab we run structural health monitoring solutions for various civil applications. Over the years we have worked with Fiber Bragg Gratings with readout units from various suppliers (HBK, Fibersensing, FBGS,…), we've used data acquisition units from various suppliers (HBK, Dewesoft, National Instruments,…) and are more and more interacting with API's or databases (MQTT, InfluxDB,…). Let alone some homebrew file formats just used by a particular supplier or researcher.
All these systems spit out different file-formats, different conventions,… Some store information on timestamps in every sample, some are more advanced, some use historian style logic, the list goes on. Yet, when looking at structural dynamics this data, despite its original source, has to have some common properties and has to go through similar pipelines, processed in pretty much the same way. Initially we just had our code take this data in, writing wrappers to all these filetypes. But as the list kept on growing we were tripping over own names (e.g. our get_fbgs worked for Fibersensing data, but was not suited for HBK fbgs data…) , we were losing sight of recent updates, or started to dread to work on these small wrappers as it required updating and maintaining the bigger analysis package.

*We needed a way out, the input-output code had to go. We needed a way that analysis was again separated from data input-output.*

* Corresponding author, Email address: wout.weijtjens@vub.be

## 2 SEP005 - SDyPy Unified timeseries

The proposed solution to the abovementioned concern is the SEP005 proposal for SDyPy unified timeseries. The proposal document can be found here and is open for discussion; `https://github.com/sdypy/sdypy/blob/main/docs/seps/sep-0005.rst`

The currently proposed structure of a timeseries is that of Python dictionary, representing a single timeseries, with possibly multiple rows of data ("channels") with common time-sampling parameters. This simple structure was chosen to require no additional dependencies (e.g. pandas) to allow for a low-overhead solution.

```
{
      "data": np.array([[1,2,3,4], [5,6,7,8]]),        # Array of data shaped (n,) (or (m, n) for `m` channels) and `n` time-sample
      "fs": float(125),                                # Sampling frequency in Hz. Optional if `time` specified.
      "name": "structure xyz",                         # Timeseries name
      "time": np.array([1,3,4,7]),                     # Optional: for equally spaced data: time-vector in seconds (shaped (n,)).
      "channel_name": ["1_x", "1_y"],                  # Optional: Name of the measured physical channel (or list of `m` channel na
      "quantity": "a",                                 # Optional: "f", "a", "v", "d", "e", "s" for force, acceleration, velocity,
      "start_timestamp": "2023-06-29T15:00:00.000000", # Optional: timestamp in absolute time. Default format is ISO 8601.
      "start_timestamp_format": "%d/%m/%y %H:%M:%S.%f" # Optional: if custom format is used in `start_timestamp`, specify it here.
      "unit_str": "m/s²",                              # Optional: unit string (or list of `m` unit strings if `data` is of shape (
      "unit_tex": "m/s$^2$",                           # Optional: LaTeX interpretation of `unit_str`.
      "...": '...'                                     # Optional: additional data.
}
```

Figure 1 : Screenshot from [1] illustrating the basic structure of a SEP005 compliant structure

To represent multiple timeseries with potentially different time-sampling parameters or different quantities, a list of timeseries-object dictionaries should be used.

The guideline adopts a strategy of compulsory fields, that are required in the context of structural dynamics, including the obvious fields ‘data’ and ‘name’ but also engineering specific fields like a unit string and a consistent definition of time (either as a vector or a fixed sampling frequency). An additional requirement is that the length of a time vector equals the length of a data vector/array. Trivial constraints, but essential to do proper structural dynamics.

A list of additional fields is also suggested on [1] to guarantee consistency across implementations, but currently the guideline does not adopt a limited list of acceptable fieldnames, anyone is free to add fields to their needs. Opposingly, variations to the compulsory or optional fields listed in the guideline, such as variations in case and/or the incorrect use of underscores (_) are prohibited simply to avoid confusion to the human interpreter of the data.

In general, there isn’t much about it, but it creates consistency and a clear interface to exchange data. While standalone the format is a bit clunky, to e.g. read it with pandas you need a bit of code, but at its core the format is readable and straightforward. It also guarantees that all information needed for structural dynamics is properly accounted for.

## 3 Code adoption

At OWI-lab we started to embrace the SEP005 strategy, as mentioned it isn’t perfect, but it does serve its purpose. We can split the development of code input-output packages from the main body of our

codebase. Allowing people to more swiftly transfer data into the existing measurement processing code.

### 3.1 Asserting SEP005 compliance

The first piece of code was the SEP005 compliance package: sdypy-sep005 (available from pip). Which only use-case is to assert that a python structure complies with sep005's guidelines through the `assert_sep005` command;

```
from sdypy_sep005.sep005 import assert_sep005

signals = read_from_path(FILE_PATH) # Your import wrapper
assert_sep005(signals)
```

Anyone who is interested in writing a wrapper function for SEP005 for a particular file format or data structure can rely on this package to assert a correct implementation of their wrapper code. Opposingly, from the analysis side this package can be used to verify that the inbound data structure complies with SEP005 as a first form of quality control.

### 3.2 Writing wrappers

The art of creating an input-output wrapper package now becomes simple. When loading a data file, the wrapper code serves to translate the loaded data to a SEP005 compliant structure, which you can check using assert_sep005.

Currently OWI-lab has already launched some sep005 packages for various filetypes, all available on GitHub; an overview of the most relevant to this community is given in Table 1.

Table 1: Overview of some of the existing SEP005 compliant wrapper packages

| PACKAGE | FILETYPES | GITHUB |
|---|---|---|
| sep005_io_dxd | Dewesoft .dxd | https://github.com/OWI-Lab/sep005_io_dxd |
| sdypy_io_tdms | National Instruments .tdms | https://github.com/OWI-Lab/sdypy_io_tdms |
| sep005_io_fbgs | FBGS .txt | https://github.com/OWI-Lab/sep005_io_fbgs |
| sep005-io-fast | OpenFAST simulation output | https://github.com/OWI-Lab/sep005-io-fast |

### 3.3 Integrating in analysis packages

Currently we are still exploring the scope of SEP005 and how far it can be integrated in for instance SDyPy. But it can serve anyone who works with structural dynamics data, as the wrapper and assertion package are standalone it is open to anyone to integrate this interface. From it you can fall back on the existing (and hopefully growing) body of wrapper packages.

To illustrate how SEP005 can be integrated into your code, we'll take an example from our in-house (non-public) code **DYNA***wind*. **DYNA***wind* handles all data through its `SignalList` class. In the past our code would contain all sorts of wrappers to convert data from different sources into this

`SignalList` class. Today, all these wrappers are to be eliminated and replaced by a workflow where SEP005 data is assumed.

Below is an example how today Dewesoft's .dxd files are first read by `sep005_io_dxd` and then simply passed into a `SignalList` using the `SignalList.from_sep005` command.

```
from sep005_io_dxd import read_dxd
from dw_signal import SignalList

file_path = # Path to the dxd file of interest
signals_sep005 = read_dxd(file_path) # imports as sep005 using the wrapper package for dxd files

signals = SignalList.from_sep005(signals_sep005)
```

The example above can simply exchange into any other file-format (e.g. .tdms) by just changing the first 4 lines. Anything below is identical independent of which data format was used.

# 4 Conclusion

SEP005 is still at its infancy, but it has a potential to simplify workflows and enable collaboration between groups that rely on data acquisition software from multiple parties. Opening up a small doorway to SEP005 into your analysis package should allow us to more easily adopt analysis packages and eliminates the need to understand the exact formatting required to work with your packages import or export functions. At the same time you can rely on a growing body of wrapper packages.

Let us discuss if this is the way forward, at OWI-lab we'll continue with the idea to split input-output code from the main codebase, and this requires a common interface, so why not SEP005.

# References

[1] SEP5 – SdyPy Unified Timeseries; https://github.com/sdypy/sdypy/blob/main/docs/seps/sep-0005.rst

# SDyPy: Following the roadmap

Klemen Zaletelj* Domen Gorjup Janko Slavič

*Faculty of Mechanical Engineering, University of Ljubljana*

**Abstract**

The SDyPy framework is an open-source initiative designed to streamline and unify workflows in structural dynamics. Building on the SDyPy-EMA, SDyPy-FRF, SDyPy-io, SDyPy-excitation and SDyPy-io packages, the SDyPy roadmap is being followed by new packages: SDyPy-model and SDyPy-view. SDyPy-model enables users to import finite element meshes and compute mass and stiffness matrices for various element types, while SDyPy-view provides intuitive tools for visualizing structural models and modal results in 3D using PyVista. Together, these additions enhance SDyPy's functionality and support its roadmap-driven development approach.

## 1 Introduction

In the field of structural dynamics, the identification of the modal parameters is an important task. Efficient approaches have been developed and implemented in different programming languages and software packages. Among them is the SDyPy framework [1], which follows the example of scipy [2] and numpy [3], and aims to unify open-source initiatives in structural dynamics. The modal identification algorithms are implemented in the SDyPy-EMA sub-package, the successor of the pyEMA package [4].

While the SDyPy-EMA sub-package provides a set of tools for identification of the modal parameters, the SDyPy framework also provides a roadmap for the development of the structural dynamics software. The roadmap is a living document that is continuously updated and extended. The roadmap is divided into several modules, each of which covers a specific area of structural dynamics. The modules are designed to be independent, but at the same time they are interconnected, allowing the user to easily switch between them. The roadmap is opened for discussion and the users are encouraged to contribute to it. The roadmap is available on the SDyPy GitHub repository `https://github.com/sdypy/sdypy/blob/main/docs/seps/sep-0004.rst`.

Recently, the SDyPy framework was extended with the SDyPy-model and SDyPy-view sub-packages.

*Corresponding author, Email address: klemen.zaletelj@fs.uni-lj.si

## 2 SDyPy-view

While identification of the modal parameters is a crucial task, the intepretation of the results is equally important. With this in mind, the SDyPy framework was extended with the SDyPy-view sub-package, which provides a set of tools for visualizing the results of the identification process and the underlying data. It is designed to be flexible and at the same time easy to use. The goal is to provide a tool for fast implementation of the visualization of the results, with an option to implement complex visualizations if needed.

The SDyPy-view is built on top of the PyVista library [5], which is a powerful visualization library for 3D data. It provides a simple and intuitive interface for visualizing 3D data, and it is widely used.

The SDyPy-view namespace package is available for installation via pip as:

```
pip install sdypy-view
```

Plotting a mesh in 3D is a common task in stuctural dynamics. The following is a code snippet that allows the user to view a finite element mesh in 3D.

```
import sdypy as sd

nodes = ... # Nodes where the data are available
elements = ... # Connections between the nodes (forming elements)

plotter = sd.view.Plotter3D(nodes, elements)
plotter.add_fem_mesh(nodes, elements)
plotter.show(show_grid=True, show_axes=True)
```

To visualize the results of the modal identification, the user can add the identified modal parameters to the plotter, for example, a modal shape:

```
mode_shape = ... # Shape (n_nodes, 3)

plotter = sd.view.Plotter3D(nodes, elements)
plotter.add_fem_mesh(nodes, elements, animate=mode_shape)
```

The `add_fem_mesh` method also automatically supports the animation of the modal shape as well as the export of the animation. The resulting view can be seen in Fig. 1.

## 3 SDyPy-model

Another important aspect of the structural dynamics is the comparison of the experimental data and the numerical models. As stated in the roadmap, the SDyPy framework was extended with the SDyPy-model sub-package, which provides a set of tools for working with numerical models. The package enables the user to import the mesh and generate the mass and stiffness matrices. The package supports the generation of the tetrahedral non-linear elements, shell elements (MITC4) and beam elements (Euler-Bernoulli).

Currently, the SDyPy-model does not provide automatic boundary conditions support, the generated models are always free-free. The user can manually apply the boundary conditions, however, in the future, the SDyPy-model will be extended with the boundary conditions support.

The SDyPy-model namespace package is available for installation via pip as:

```
pip install sdypy-model
```

The following code snippet demonstrates how to load a gmsh-generated [6] mesh (using meshio [7]) and compute the mass and stiffness matrices for a shell element model:

```
import sdypy as sd
import meshio

mesh = meshio.read("data/L_bracket.msh")

# extract nodes and elements from mesh (gmsh format)
nodes = mesh.points / 1000
elements = []
for cells in mesh.cells:
  if cells.type == "quad":
    elements.append(cells.data)
elements = np.vstack(elements)

#  Properties
thickness = 0.001
E = 2.069e11
nu = 0.3
rho = 7829

# Compute the mass and stiffness matrices
shell_obj = Shell(nodes, elements, E, nu, rho, thickness)
K, M = shell_obj.K, shell_obj.M
```

The excellant numpy [3] and scipy [2] libraries are used for assempling matrices and computing the eigenvalues and eigenvectors. The resulting mode shape can be visualized with the SDyPy-view package and is shown in Fig. 1.

## 4 Conclusion

The SDyPy framework aims to unify the open-source initiatives in structural dynamics and streamline the standard workflows in the field. In addition to that, SDyPy aims to provide functionallity that is extendable for advanced users. While SDyPy-EMA is the basis for the modal identification, according to the roadmap, the framework is continuously extended with new modules and improvements of the existing ones. With the recently added SDyPy-model and SDyPy-view sub-packages, the framework now provides the tools for finite element model analysis and visualization of the results.

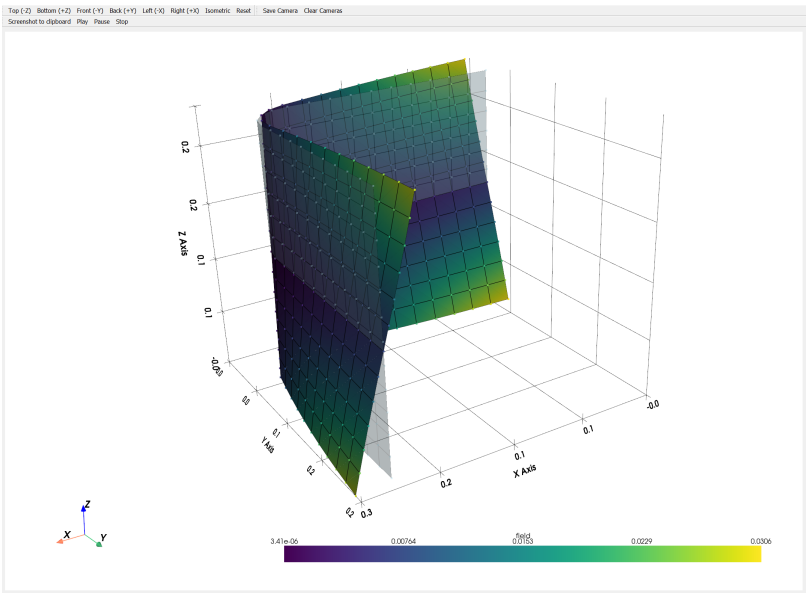



**Fig. 1:** Example of the SDyPy-model-computed response with SDyPy-view render.

## References


[1] "GitHub - sdypy/sdypy: Structural Dynamics Python — github.com." `https://github.com/sdypy/sdypy`. [Accessed 02-06-2025].

[2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sep 2020.

[4] K. Zaletelj, T. Bregar, D. Gorjup, and J. Slavi, "ladisk/pyema: v0.24," Sept. 2020.

[5] B. Sullivan and A. Kaszynski, "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)," *Journal of Open Source Software*, vol. 4, p. 1450, May 2019.

[6] Geuzaine, Christophe and Remacle, Jean-Francois, "Gmsh," 2020. Version 4.6.0, released 2020-06-22.

[7] N. Schlömer, "meshio: Tools for mesh files," 2024.

# OASIS: Bridging the Open-Source Gap Between Testing and Data Processing

Oliver M. Zobel* Johannes Maierhofer Andreas Köstler Daniel J. Rixen

*Chair of Applied Mechanics, TUM School of Engineering and Design,*
*Technical University of Munich, Boltzmannstr. 15, 85748 Garching, Germany*

**Abstract**

Open-source software brings great advantages; besides being free of charge, it allows everyone to look into the used algorithms and adapt the functionality based on individual requirements. However, the measurements themselves still rely on commercial products for data acquisition. To bridge this gap in the open-source structural dynamics analysis chain, the *Open Acquisition System for IEPE Sensors* (*OASIS*) has been developed. Combining the software side with the steadily increasing amount of open hardware allows the entire measurement and analysis to be performed solely using open tools. This contribution showcases the history of *OASIS* as well as the specifications and capabilities of the latest system, *OASIS-UROS*, available today. An exemplary procedure for open-source data acquisition and experimental modal analysis is shown using an academic test structure and the well-established Python packages *pyFRF* and *pyFBS*. While *OASIS-UROS* cannot match the full performance of the commercial system, the developed system can be a viable alternative for students, people in academia, or smaller companies with a constrained budget.

**Keywords:** Open-Source, Acquisition Hardware, Data Acquisition, Experimental Dynamics, Python

## 1 Statement of Need

Open-source software brings significant advantages; besides being free of charge, it allows everyone to look into the used algorithms and adapt the functionality based on individual requirements. For structural dynamics, many noteworthy Python packages exist today, e.g., *pyFRF*, *pyFBS*, *pyUFF*, *SDynPy*, *pyIDI*, and many more. This allows the processing and analysis of measurement data without the need for commercial solutions and enables custom post-processing steps.

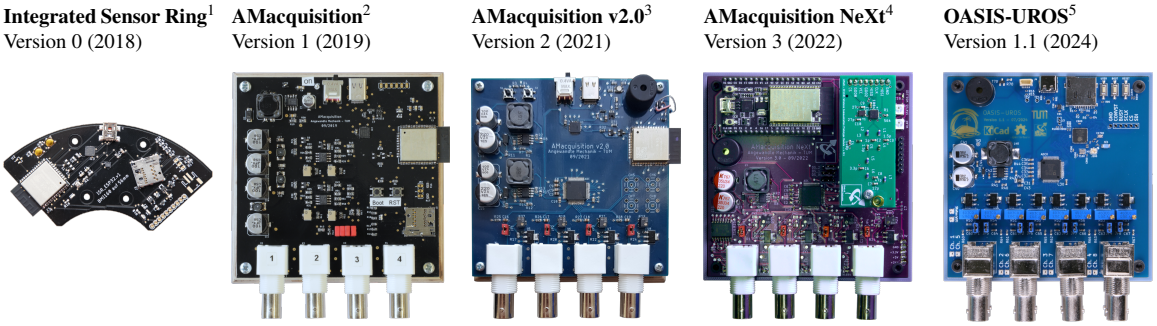
Nevertheless, the measurements themselves still rely on commercial products for data acquisition. This makes adaption to specific measurement requirements, such as self-contained measurements on drones, more cumbersome. Further, integrating costly measurement equipment into dangerous environments, like drones, also holds a substantial financial risk.

To bridge the gap in the open-source structural dynamics analysis chain and to enable a cost-effective measurement in dangerous environments, the *Open Acquisition System for IEPE Sensors* (*OASIS*) has been developed. The same benefits of open-source software apply to open hardware, allowing to build on other people's work and customizing for the task at hand.

*Corresponding author, Email address: oliver.zobel@tum.de

## 2 Development History of OASIS

The development of *OASIS* began in 2018 with the *Integrated Sensor Ring*; see also fig. 1 on the left. Originally, the idea was to develop a custom acquisition system that could be mounted on a rotor during operation. The measured vibrations, captured in a body-fixed coordinate system, should then be used for condition monitoring. Out of this first prototype, a standalone version called *AMacquisition* (*AM* as in Chair of **A**pplied **M**echanics) was created in 2019. Both systems used an *Espressif ESP32* micro-controller as their central processing unit, including the built-in analog-digital-converter (ADC). The IEPE constant current supply was realized using discrete components and transistors. In order to measure the acceleration sensor signals, which for IEPE sensors oscillate around the DC bias voltage of approximately 12 V, with the *ESP32*'s ADC, a voltage divider was used to scale the signal. This resulted in signal loss due to the resistive voltage divider and required calibration for accurate results.




Contributions: Johannes Maierhofer: [12345], Philipp Radecker: [1], Andreas Köstler: [2345], Oliver Maximilian Zobel: [345]

**Fig. 1:** Overview of *OASIS* development prototypes in chronological order.

With *AMacquisition v2.0* in 2021, the first major hardware upgrades were introduced: Instead of relying on the internal ADCs of the *ESP32*, a dedicated simultaneously sampling bipolar ADC, the *Analog Devices AD7606C-18*, was used. Starting with this board, *LT3092* constant current sources were used, the voltage divider removed, and the DC bias of IEPE sensors removed using a high-pass filter. Due to this, the signal quality was greatly improved. With this version, also the software core of today's *OASIS*, both for the firmware and control software, was created.

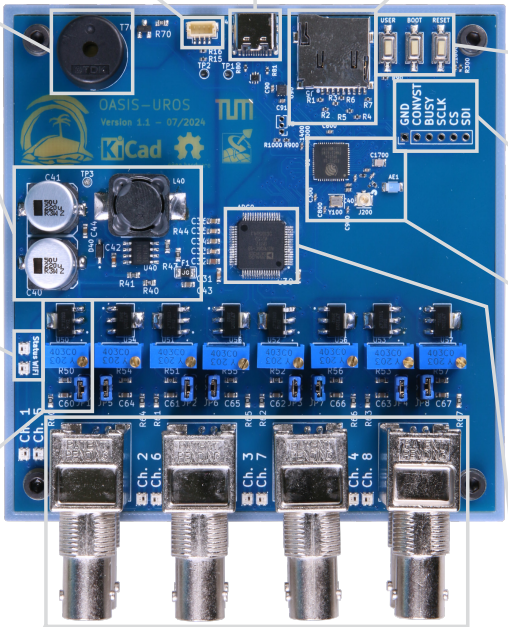
*AMacquisition NeXt* was developed in 2022 and later released as *OASIS* [1]. Notable changes are the added wireless synchronization feature and improved robustness of the IEPE supply. Significant improvements to the control software were also achieved with this version.

The latest version *OASIS-UROS* (*Open Acquisition System for IEPE Sensors - Upgraded, Refined, and Overhauled Software*) [2] was released in 2024. This version aimed at cleaning up both software and hardware to provide a more robust system, as described in the following.

## 3 Features of the 2024 Hardware – OASIS-UROS

An exhaustive documentation of *OASIS-UROS* can be found in [2]. On the software side, a major increase in robustness and performance could be achieved by caching the raw measurement data on an SD card, compared to the data streaming approach of the previous versions, and reading eight data lines simultaneously. Further, using a hardware-generated PWM signal, the sample jitter was reduced.

Figure 2 provides an overview of the key hardware components and features of *OASIS-UROS*.




**Optional I²C Interface**
Can be used for future extensions of the board

**USB-C Interface**
Used for powering the board as well as for serial communication

**microSD-Card Slot**
Supports up to 32 GB SD cards with FAT32 format to store measurements

**Piezo Buzzer**
Provides audible feedback about measurement start, end, and potential errors

**User Interface**
Used to reset the device, put into bootloader flash mode, and optionally execute custom function

**24 V Boost Converter**
Generates the voltage for the IEPE supply using an *MC34063A* chip

**ADC SPI Connector**
Can be used to debug ADC communication issues

**Indicator LEDs**
Inform about the current system status, e.g., measurement ongoing or post-processing, as well as WiFi connectivity

**Micro-controller**
The acquisition system is controlled using an *ESP32-S3* chip with an on-board WiFi antenna, enabling wireless sampling

**Acquisition Front-End**
Features adjustable constant current sources using *LT3092* chips for IEPE sensors as well as high-pass filters removing DC bias

**Analog-Digital Converter**
Core of the device is the *AD7606C-18*, bipolar input, simultaneous sampling ADC

**BNC Connectors**
Allow to connect up to 8 IEPE sensors to the board and feature status LEDs for each channel

**Core Specifications**

| | |
|---|---|
| Available channels | 8 IEPE channels with BNC connector |
| Sampling frequency @ 8 ch. | 36 kHz per channel @ 18-bit resolution |
| Voltage ranges | ±2.5 V, ±5 V, ±6.25 V, ±10 V, ±12.5 V |
| Bill of materials cost | ≈ $ 220 / 200 € |

**Fig. 2:** Overview of the *OASIS-UROS* board and outline of key features.

*OASIS-UROS* is built around the *Analog Devices AD7606C-18* 18-bit ADC and the *Espressif ESP32-S3* micro-controller. A key feature of the board is its integrated IEPE power supply, which utilizes an *MC34063* boost converter and *LT3092* constant current sources. This design enables operation with just a 5 V USB power supply; the four-layer PCB includes all essential hardware components.

The *AD7606C-18* provides eight simultaneously sampled channels with adjustable input ranges and a built-in programmable oversampling function to reduce noise. Each channel features a dedicated acquisition front-end with an *LT3092*-based constant current source, which can be fine-tuned using a potentiometer. The board directly includes BNC connectors, eliminating the need for adapters. All input signals pass through a fixed passive high-pass RC filter with a −3 dB cutoff frequency of 0.8 Hz, meaning *OASIS* operates in AC mode and attenuates frequencies below 3.5 Hz by more than 5 %. While this makes it unsuitable for low-frequency structural health monitoring applications, the open-source design allows users to modify the high-pass filter as needed.

# 4 Software Stack

The firmware of *OASIS* is written in Arduino using C++, with the latest version available on the GitLab project page[1]. With the release of *OASIS-UROS*, the control software, the *OASIS-GUI*, was also made available on the *Python Package Index*[2], and can now be simply installed using `pip install OASIS-GUI`. Once installed, the GUI can be opened from a command prompt using `oasis-gui`. A screenshot of the current version can be seen in fig. 3.

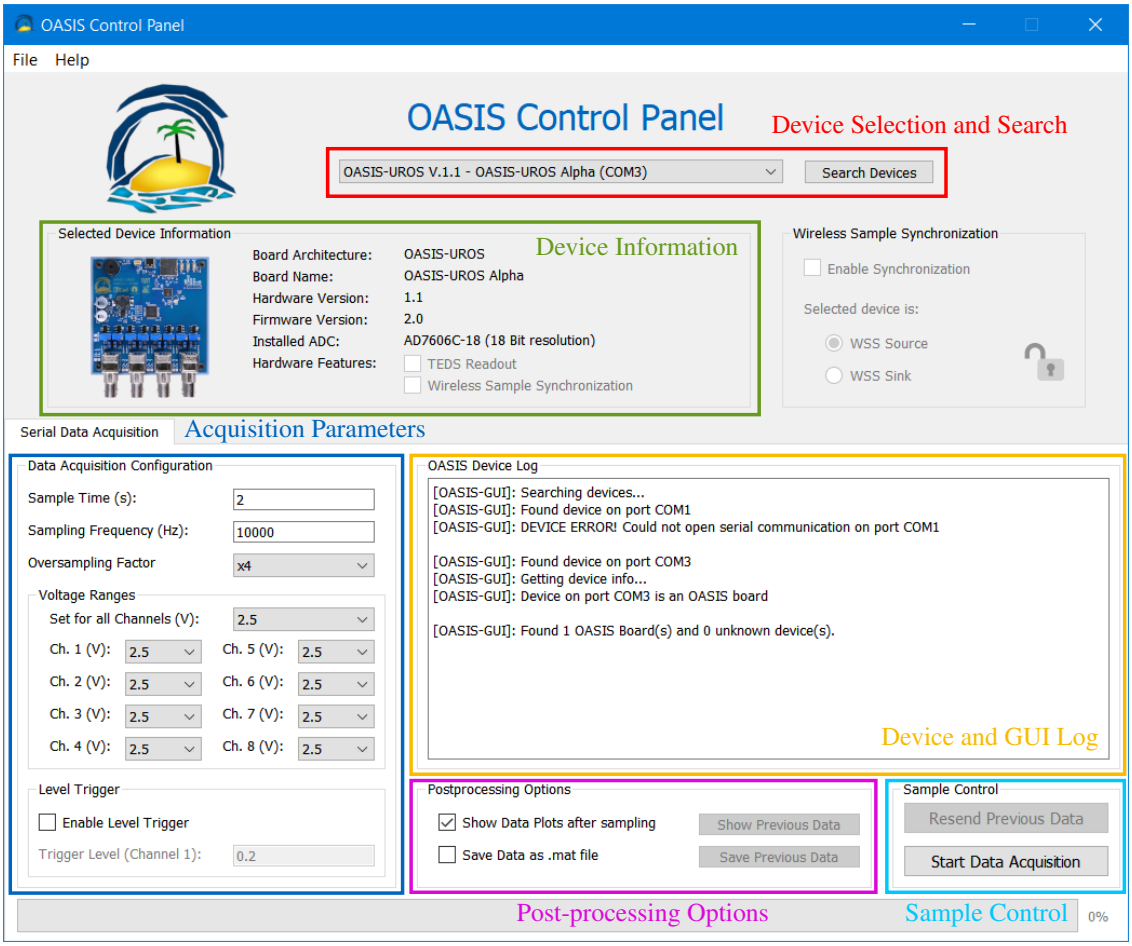



**Fig. 3:** *OASIS-GUI* after opening with an *OASIS-UROS* board connected.

The main features include the automatic detection of *OASIS* devices connected to the computer, the display of device information and log messages, and the live control of the attached *OASIS* device. Before each sample, the acquisition parameters can be adjusted. This includes the sample time and frequency, the oversampling factor (for noise suppression), the voltage ranges for each channel, and an optional level trigger. The post-processing options allow to preselect the behavior after the sampling process is finished, i.e., whether the acquired data should be automatically plotted (see also fig. 4) and/or saved as a `*.mat` file. Both options can also be invoked manually after each sample.

[1] https://gitlab.com/oasis-acquisition
[2] https://pypi.org/project/OASIS-GUI/

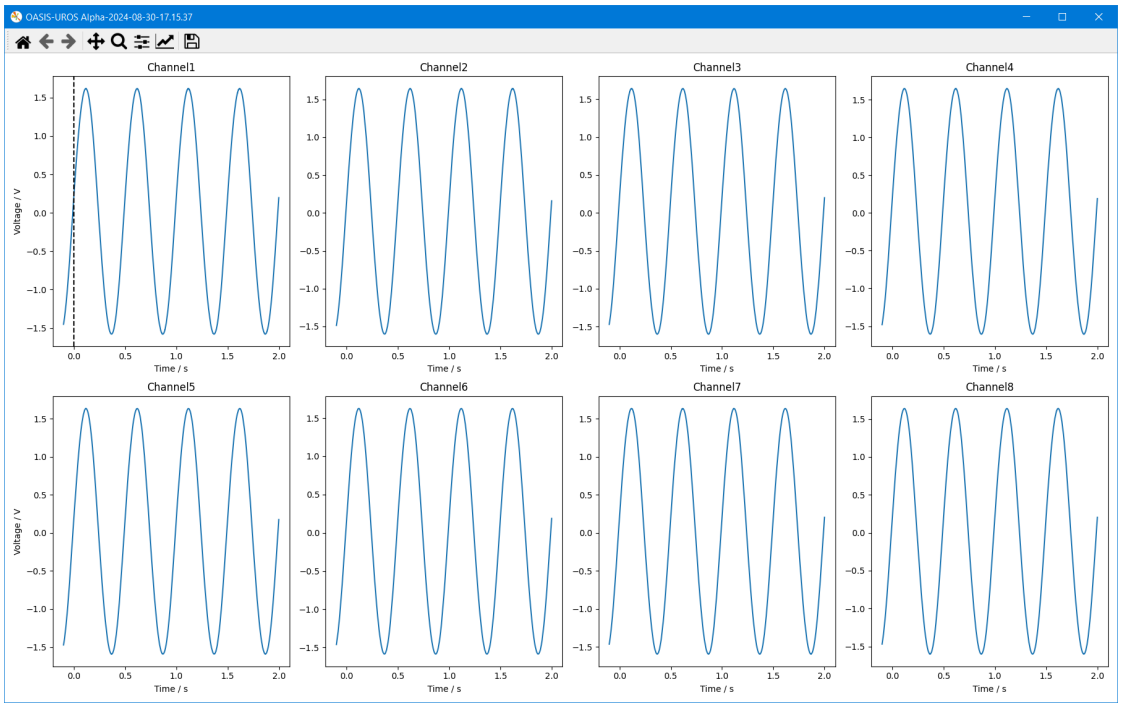



**Fig. 4:** Plot displayed by *OASIS-GUI* after the triggered data acquisition is completed.

## 5 Experimental Modal Analysis Using pyFRF and pyFBS

Since the *OASIS-GUI* only saves the raw time series of the measured voltages, any data analysis has to be performed externally. Here, an example of evaluating an experimental modal analysis using *pyFRF* [3] and *pyFBS* [4] is shown. The impact measurements performed on the academic test structure depicted in fig. 5 are available for download in [5] and are described in detail in [2].

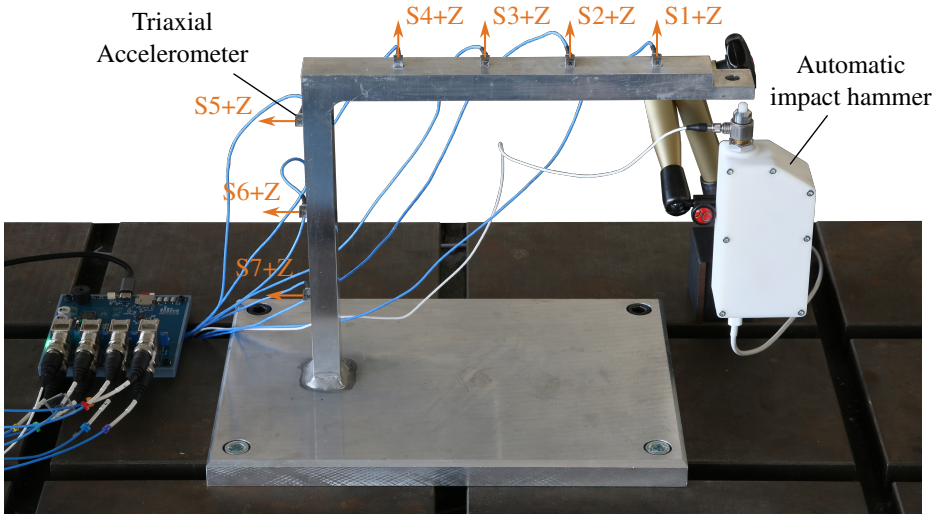



**Fig. 5:** Experimental setup of academic test structure used for validation measurements.

In the following, the content of the Python script `Modal_Analysis.py`, available for download in [5], is described. First, the required packages are imported in listing 1, including *NumPy* for basic linear algebra, *SciPy* for loading `*.mat` files as well as *pyFRF* and *pyFBS*. Additionally, the measurement files to be loaded are defined in `fileList` and the sensor calibration factors are defined.

```
1  # -*- coding: utf-8 -*-
2  """
3  Modal Analysis of OASIS-UROS measurements using pyFRF and pyFBS
4
5  """
6
7  import numpy as np
8  import matplotlib.pyplot as plt
9  import pyFRF
10 import pyFBS
11 from scipy.io import loadmat
12 import scipy as sp
13 import os
14
15 # Set working directory to script directory
16 os.chdir(os.path.dirname(os.path.abspath(__file__)))
17
18 # Measurement data to load
19 fileList = ["OASIS/OASISData-OASIS-UROS Alpha-2024-08-28-16.50.14.mat",
20             "OASIS/OASISData-OASIS-UROS Alpha-2024-08-28-16.51.37.mat",
21             "OASIS/OASISData-OASIS-UROS Alpha-2024-08-28-16.52.19.mat",
22             "OASIS/OASISData-OASIS-UROS Alpha-2024-08-28-16.52.50.mat",
23             "OASIS/OASISData-OASIS-UROS Alpha-2024-08-28-16.54.17.mat"]
24
25 # Calibration data
26 ExcConvFactor = 1/(22.88555 * 1e-3) # (mV/N * V/mV = V/N)^-1 => N/V
27 RespConvFactorZ1 = 1/(10.95 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
28 RespConvFactorZ2 = 1/(10.53 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
29 RespConvFactorZ3 = 1/(8.88 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
30 RespConvFactorZ4 = 1/(10.48 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
31 RespConvFactorZ5 = 1/(10.96 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
32 RespConvFactorZ6 = 1/(8.96 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
33 RespConvFactorZ7 = 1/(8.94 * 1e-3) # (mV/g * V/mV = V/g)^-1 => g/V
```

**Listing 1:** Import of Python packages, definition of measurement data and calibration factor.

Next, in listing 2 the measured data is loaded and stored into *NumPy* arrays. The time vector `time_vec` is loaded from the first `*.mat` file, and the sampling frequency `sample_frequency` is determined from it.. Next, the indices `start_pos` and `end_pos` for a rectangular force window are determined. The `start_pos` is defined as two samples prior to the trigger point during the measurements (where $t = 0$), and the `end_pos` is estimated at $t = 1.5\,\text{ms}$, based on the impact duration. Every sample outside these indices is set to zero in lines 58-59. Lastly, the calibration factor is applied to convert the measured voltages into physical units in lines 65-72.

```
36 # ---------- Load OASISData ----------
37
38 n_resp_channels = 7
39 MATFileData = loadmat(fileList[0])
40
41 # Retrieve data acquisition information
42 time_vec = np.squeeze(MATFileData.get("OASISTime"))
43 sample_frequency = round(1/(time_vec[-1]-time_vec[-2]))
44 start_pos = np.where(time_vec==0)[0][0] - 2
45 end_pos = np.where(time_vec>=0.0015)[0][0]
46
47 # Array preallocation
48 OASISExcitationData = np.zeros([len(fileList), len(time_vec)])
```

```
49  OASISResponseData = np.zeros([len(fileList), n_resp_channels, len(time_vec)])
50
51  # Load measurements
52  for idx,file in enumerate(fileList):
53      MATFileData = loadmat(f"{file}")
54      OASISExcitationData[idx] = MATFileData.get('OASISChannel')[0]
55      OASISResponseData[idx,:,:] = MATFileData.get('OASISChannel')[1:]
56
57  # Force windowing
58  force_window = np.zeros(len(time_vec))
59  force_window[start_pos:end_pos] = 1
60
61  for idx in range(len(fileList)):
62      OASISExcitationData[idx,:] = OASISExcitationData[idx,:]*force_window
63
64  # Apply calibration factor
65  OASISExcitationData *= ExcConvFactor
66  OASISResponseData[:,0,:] *= RespConvFactorZ1
67  OASISResponseData[:,1,:] *= RespConvFactorZ2
68  OASISResponseData[:,2,:] *= RespConvFactorZ3
69  OASISResponseData[:,3,:] *= RespConvFactorZ4
70  OASISResponseData[:,4,:] *= RespConvFactorZ5
71  OASISResponseData[:,5,:] *= RespConvFactorZ6
72  OASISResponseData[:,6,:] *= RespConvFactorZ7
```

**Listing 2:** Loading of measurements, application of force window, and conversion to physical units.

In listing 3, the FRFs are calculated using *pyFRF*. For this, a *pyFRF* object `FRFpy` is created and configured in lines 81-85, including applying an exponential window. Then, the measurement data is added to `FRFpy` using `add_data` in line 87, and in line 89, the FRFs are calculated using the $H_1$-estimator. In lines 93-95, the FRFs, the coherence, and the frequency axis are extracted from `FRFpy` and converted into *NumPy* arrays.

```
75  # ---------- Build FRF matrices ----------
76  sample_frequency = 25600
77
78  FRF_matrix = []
79  FRF_coherence = []
80
81  FRFpy = pyFRF.FRF(sampling_freq=sample_frequency,
82                    fft_len=len(time_vec),
83                    exc_type='f',
84                    resp_type='a',
85                    window="exponential:0.01")
86
87  FRFpy.add_data(OASISExcitationData, OASISResponseData)
88
89  _frf = FRFpy.get_FRF(type='H1', form='accelerance')
90  FRF_matrix.append(_frf)
91  FRF_coherence.append(FRFpy.get_coherence())
92
93  freq_axis = FRFpy.get_f_axis()
94  FRF_matrix = np.squeeze(np.asarray(FRF_matrix).T)
95  FRF_coherence = np.squeeze(np.asarray(FRF_coherence).T)
```

**Listing 3:** Calculation of FRFs using the *pyFRF* [3] package.

The frequency bandwidth considered for the following modal analysis are defined in listing 4.

```
98  # ---------- Analysis settings ----------
99
100 # Analysis settings
101 freq_start = 1 # Hertz
102 freq_end = 4000 # Hertz
```

```
103
104 f_start_idx = (np.abs(freq_axis - freq_start)).argmin()
105 f_end_idx = (np.abs(freq_axis - freq_end)).argmin()
```

**Listing 4:** Definition of modal analysis frequency bandwidth.

Next, the *pyFBS* `modal_id` module is initialized in listing 5 by providing the frequency axis `freq_axis` and the FRFs `FRF_matrix`. The latter has to be reshaped into the dimensions expected by *pyFBS*, and both are truncated to the frequency bandwidth defined in listing 4. In line 111, the poles up to a `max_order` of 64 are calculated, and line 112 generates the stabilization diagram shown in fig. 6.

```
108 # ---------- pyFBS Modal extraction ----------
109
110 FBS_object = pyFBS.modal_id(freq_axis[f_start_idx:f_end_idx], FRF_matrix[f_start_idx:
        f_end_idx,:].reshape([f_end_idx-f_start_idx,7,1]))
111 FBS_object.pLSCF(max_order=64)
112 FBS_object.stabilization()
```

**Listing 5:** Calculation of poles for stabilization diagram using *pyFBS* [4].

The stabilization diagram in fig. 6 shows the determined poles and their stability, similar to the depiction in commercial measurement system software. Here, the desired poles can be selected and then appear in the list in the top left.

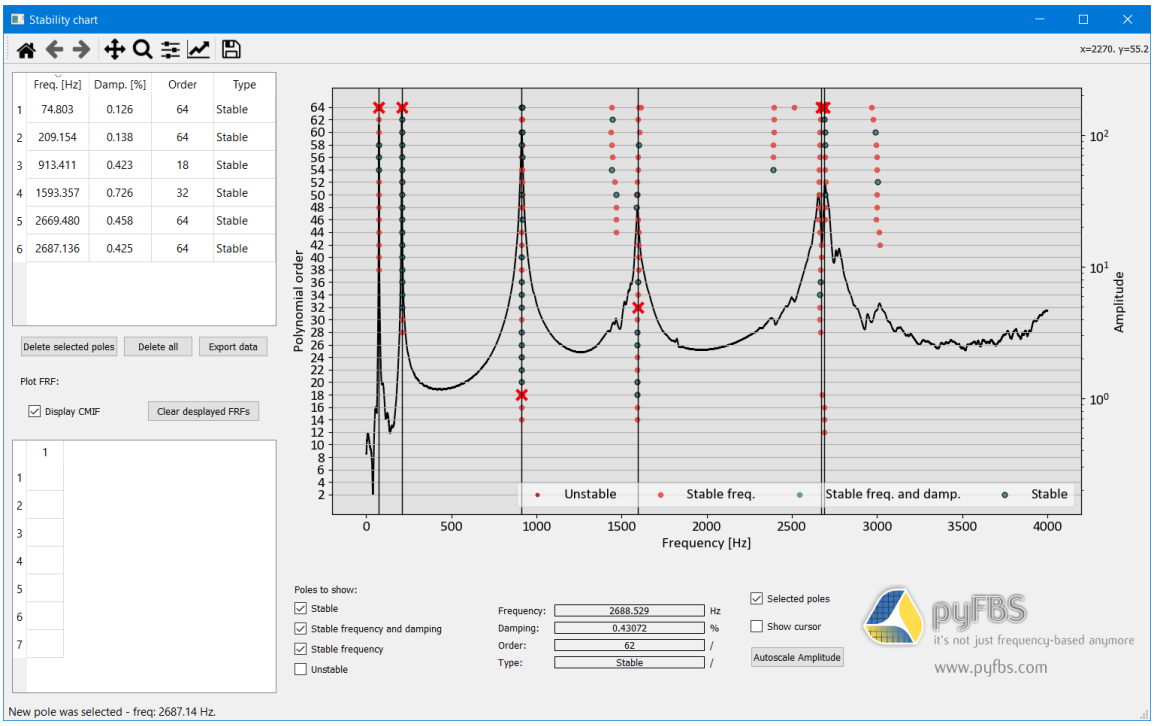



**Fig. 6:** Exemplary stability diagram with selected poles generated by *pyFBS* [4].

Lastly, to extract the eigenfrequencies, damping ratios, and mode shapes of the selected poles, the code in listing 6 has to be executed. Then, the modes are stored in `mode_shapes`; the `eigenfrequencies` and `damping_ratios` are extracted from the poles in lines 120-121.

A validation of *OASIS-UROS* against a commercial measurement system in the context of experimental modal analysis of the academic structure shown in fig. 5 can be found in [2].

```
# ---------- pyFBS Modal extraction - Post-Process ----------
# Run this cell after selecting poles in the stabilization diagram created by the cell
    above

FBS_object.pLSFD(reconstruction=True,lower_residuals=False,upper_residuals=False)
mode_shapes = FBS_object.shape
eigenfrequencies = np.imag(FBS_object.selected_poles)/(2*np.pi)
damping_ratios = -np.real(FBS_object.selected_poles)/np.imag(FBS_object.selected_poles)
```

**Listing 6:** Extraction of modal parameters from *pyFBS* [4] object.

## 6 Conclusion and Outlook

From the validation in [2], is can be seen that for frequencies up to 3 kHz, no significant differences exist in the FRFs estimated by the commercial system with proprietary software and those retrieved using open-source software and hardware. Besides some differences in the damping ratios, the experimental modal analysis validation results matched pretty closely between the commercial and the open-source system. While *OASIS-UROS* cannot match the performance of the commercial system, the authors believe that, especially when comparing the costs, the developed system is a viable alternative for students, people in academia, or smaller companies that have a constrained budget or require complete insight as well as adaptability of the hardware and software.

Future developments of the *OASIS* systems will focus on further improving the usability for day-to-day measurements. On the one hand, this concerns the control software, which today can only acquire single measurements of voltage time series and save the data in individual files. To better match the commercial system experience, the software should be extended to include channel setup and project management as well as immediate post-processing capabilities like calculating averaged spectra, FRFs, and coherences. On the other hand, previous features of the original *OASIS*, like wireless sample acquisition and synchronization, that have been removed due to time constraints should be reintegrated into the new and improved software and hardware framework.

## References


[1] O. M. Zobel, J. Maierhofer, and D. J. Rixen. “OASIS: Open Acquisition System for IEPE Sensors: For Academic Research and Teaching Purposes”. In: *Conference Proceedings of the Society for Experimental Mechanics Series* (2024), pp. 79–86. DOI: 10.1007/978-3-031-34938-6_9.

[2] O. M. Zobel, J. Maierhofer, A. Köstler, and D. J. Rixen. “OASIS-UROS: Open Acquisition System for IEPE Sensors - Upgraded, Refined, and Overhauled Software”. In: *HardwareX* (2025). ISSN: 2468-0672. DOI: 10.1016/j.ohx.2025.e00650.

[3] J. Slavič, M. Česnik, K. Zaletelj, and L. Novak. *pyFRF: Frequency response function as used in structural dynamics (v1.1.1)*. 2024. URL: https://github.com/ladisk/pyFRF.

[4] T. Bregar, A. El Mahmoudi, M. Kodrič, D. Ocepek, F. Trainotti, M. Pogačar, and M. Göldeli. *pyFBS (v0.3.1)*. 2023. URL: https://pyfbs.readthedocs.io/en/latest/index.html.

[5] O. M. Zobel, J. Maierhofer, A. Köstler, and D. J. Rixen. *OASIS-UROS Experimental Validation Measurement Data*. Dataset. 2024. DOI: 10.14459/2024mp1754305.

CIP - Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID 240253955
ISBN 978-961-7187-17-5 (PDF)