World Conference on
Timber Engineering
Oslo 2023

HYSTERESIS - A PYTHON LIBRARY FOR ANALYSING STRUCTURAL DATA

Christian Slotboom[1]

ABSTRACT: Researchers studying the design of timber structures are often required to generate and process a large amount of data from experimental and numerical studies. Hysteretic data coming from seismic tests is particularly challenging to work with, because the x/y curve will change direction through testing. This paper provides an overview of Hysteresis, a software library written in the Python programming language that can be used to quickly process and analyse structural data, including hysteretic curves.
The main structure and algorithms used in the software package are presented, including a summary of how data is represented in the package, and how it can be used. Two case studies are then presented where data is processed using the Hysteresis package. In the first, experimental and numerical data from tests on a shear wall are processed and compared in a variety of ways. The second, Hysteresis is used in an optimization analysis, where a genetic algorithm is used to fit non-linear material data to a structural element.

KEYWORDS: Hysteresis, Connection Testing, Data Analysis, Genetic Algorithm.

1 INTRODUCTION

In seismically active areas, wood structural elements and connections will be exposed to repeated loading. For these elements it's important to understand not just the initial behaviour of the specimen to load, but how the response of the system change over repeated loading in a force deformation hysteresis. Experimentalists testing structural elements and connections will often have to apply load protocols, such as the CUREE method proposed in Krawinkler et al. [1], to their test specimens to understand this behaviour. Common tasks may include finding the area under a curve, finding the back-bone curve of a hysteresis, or fitting a new curve to a back bone, such as in the Equivalent Energy Elastic-Plastic procedure outlined in ASTM E2126 [2]. These data processing tasks are often very time consuming and take away time from high level research.
Structural researchers are also frequently working with hysteretic data coming from a finite element numerical analysis. In addition to processing this data, there are many common tasks that require researchers compare output data a numerical model to those from experiment to see their similarity. Researchers also may want to fit parameters of a non-linear material model, for example Steel02 [3] or pinching4 [4] from the OpenSees [5,6] library, to their experimental data. These tasks are often completed manually, due to the difficulty in matching different hysteresis.
To address these challenges in processing experimental and numerical data, a library in the Python programming language has been developed. This library can be used to work with simple x/y curves, as well as hysteretic data that changes direction in cycles. The following paper describes the Hysteresis library, its main features, and showcases several key use-cases.

2 HYSTERESIS PACKAGE SUMMARY

2.1 BACKGROUND

Data from structural tests is generally represented by a series of (x,y) points, commonly force vs. deformation. In the case of structural data, it can be useful for curves to be divided into regions where the y values are "mostly" increasing or decreasing monotonically. For a CLT wall loaded in shear loaded may have a force deformation curve that increases to a peak value, then decreases after this value. Figure 1 demonstrates this, using the final cycle from a force deformation hysteresis tested by Drexlar et. al. [7], where the cycle has been split into regions of interest.

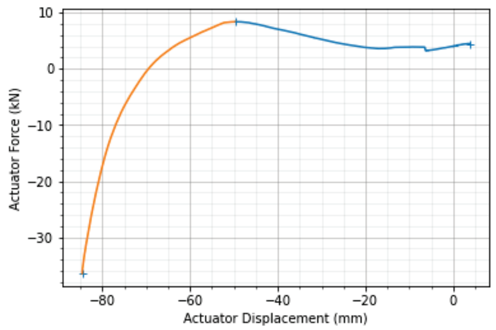


Figure 1: *A data from a shear wall split into regions that are mostly increasing (orange) or decreasing (blue).*

[1] Christian Slotboom, Graduate Engineer, Fast+Epp, cslotboom@fastepp.com

Structural data will also occur in cycles, where the x axis values are either increasing or decreasing monotonically. This commonly occurs in reverse cyclic loading for seismic testing. Figure 2 shows a hysteresis of the same shear wall in Figure 1, highlighting distinct regions where the x values are mostly unchanging. Similar to the monotonic load case, it's often useful to determine where these cycles occur in data.

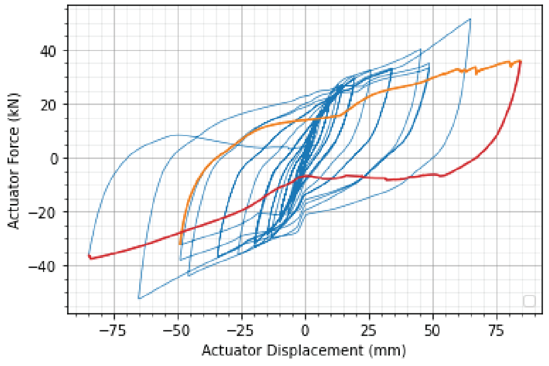


Figure 2: *A positive cycle (orange), and negative cycle (red) noted in a shear wall hysteresis.*

In both Figure 1 and Figure 2, there are small regions where the graph will decrease slightly in a roughly increasing zone. These irregularities are often not important to the data but can complicate finding data.

2.2 ABSTRACTIONS AND CLASS STRUCTURE

To represent x/y curves of structural data, the hysteresis Python package contains three main classes of curve objects:

- The "MonotonicCurve", where the x/y curve does not change in in its vertical axis and its horizontal axis.
- The "SimpleCycle", where data changes only in the y axis only.
- The "Hysteresis", where data in the x/y curve can change in both the both the x and y direction.

The structure of each curve object is nested, so that Hysteresis objects can be broken up into SimpleCycle objects, and SimpleCycle can be broken up into of MonotonicCurve objects.

Hysteresis and SimpleCycle objects divide themselves into sub-curves by finding reversal points in either the x data, or peaks in the y data. Signal filtering functions from the Python library Scipy [8] are used to find these local extreme values in the data. These functions also allow peaks/reversals to be selected based on various criteria, such as number of data points between extremes, or the prominence of the peaks/reversals from other lower points. Providing a method of filtering out smaller peaks in the data allows the main objects to ignore minor irregularities within the input data.

MonotonicCurve, SimpleCycle, and Hysteresis objects all inherit from a common "CurveBase" class that provides functionality to all objects. Included in this base class are a number of methods that can be used to numerically find properties like the curve, such as slope, cumulative change in x, arclength, or area under the curve. Figure 3 showcases some of these properties for a SimpleCycle object. By default, slope and area is calculated with second order finite difference schemes.

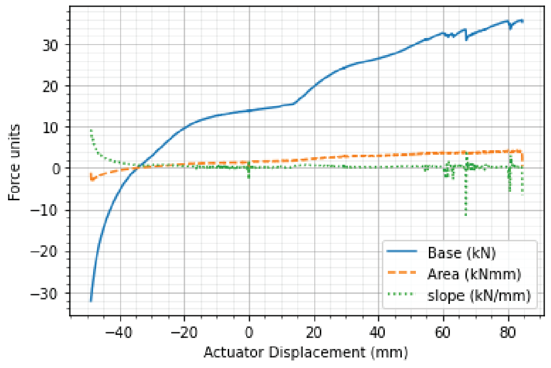


Figure 3: *A SimpleCycle object cycle with its numerically calculated slope, and area under the curve.*

The CurveBase class also includes a number of methods that allow for the curve to be easily plotted. Built in methods include: making a x/y plot of the curve, plotting the cumulative area under the curve, or plotting a curves x value against the index. Plots can also be made to show peak values or reversal points. All plots produced by the hysteresis module make use of the Matplotlib Python library [9] and can be modified with well-known commands. Figure 4 shows two of the plot functions, for the hysteresis in Figure 2 with reversal points highlighted, and the same data with x values plotted against index.

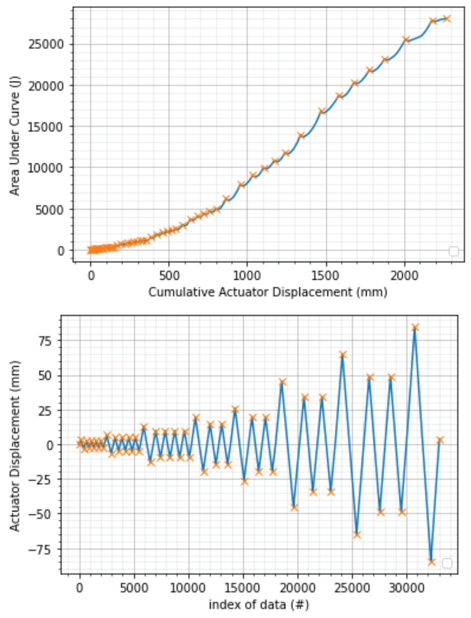


Figure 4: *Top, experiment hysteresis with reversal points highlighted. Bottom, the x displacement plotted against the index used, with reversal points highlighted.*

The Hysteresis package was designed to allow users to customize the behaviour of its key objects. At the time of object creation, functions are assigned to each object. For example, the "areaFunction" is set, reading from a environment object that contains the default functions used by all objects. The stored areaFunction will then be called later when a request is made to calculate area. Users can implement custom behaviour by either directly overwritten the desired function in individuals object, or by setting a new function in the environment variable. Changing the environment will affect the behaviour of all objects created afterwards.

Figure 5 summarizes the main classes used in the Hysteresis package, and some of their key methods. Arrows represent inheritance relationships, where one class is a subclass of another class, while diamonds represent composition where objects are stored in other objects. Hysteresis classes will contain SimpleCycles in a list of "cycles", and similarly SimpleCycles contain MonotonicCurves in a list of subcycles.

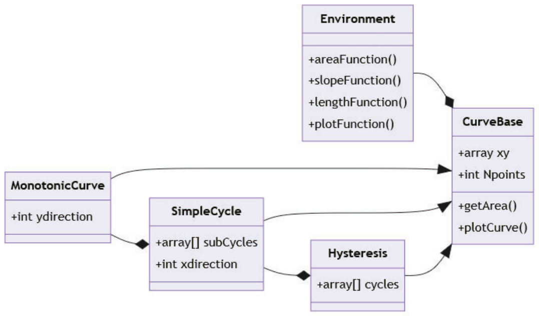


Figure 5: Class diagram for the Hysteresis package's main abstractions, and some key methods for each class.

2.3 BASIC FUNCTIONS

In addition to the basic numerical analysis and plotting functionality provided by the CurveBase class, there are a number of functions in the Hysteresis package that allows users to operate on hysteresis objects. Different object types can be combined can be combined with the "concatenate" function. This allows for more complex objects, such as the SimpleCycle or Hysteresis, to be built up from its component curves, or lists of x/y data.

Another desirable feature is the ability to take an input curve and resample it, which allows for data to be compressed or expanded. The Hysteresis package has a "resample" function, which allows for the number of x axis points to be increased or decreased using linear interpolation. When called on the MonotonicCurve or SimpleCycle Object, resample will change the total number of points in the curve, while when Hysteresis objects are resampled the number of points in each sub-curve is changed. This highlights a major benefit of the class structure used, because complex objects such as the Hysteresis are broken into simpler objects, it's possible to easily resample entire hysteresis by resampling its components. Resampling can be done based on a number of increments between the maximum and minimum x value or based on a sample change in x direction distance. Figure 9 shows the final cycle shown in Figure 1, where the curve has been resampled to only include 5 segments.

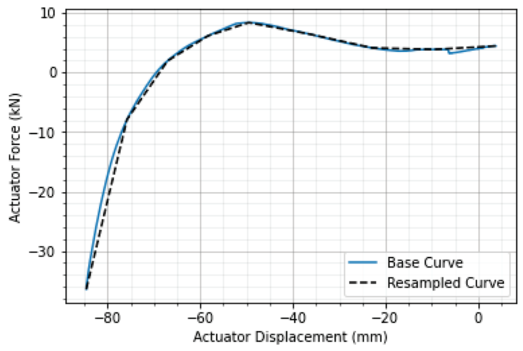


Figure 6: *The x/y curve shown in Figure 1, and a resampled version of that curve.*

A feature enabled by the resample function is the ability to various Curve objects of the same type. Two curves can be compared using the "compare" function by resampling each curve, then summing the Euler distance over all the resulting set of points. Equations (1,2) below overview this calculation for a SimpleCycle and Hysteresis respectively, where: "j" is an index that counts the number of sample point; and "i" is an index that counts the number of SimpleCycles that make up a hysteresis object. While the actual sum of the two curves is not physically significant, it's size will indicate how similar two curves are, and reduce to zero when curves coincide. Like resampling, Hysteresis curves are compared by summing the comparison between the monotonic cycles of each object. Figure 7 showcase the process comparison process, where the final output is the average distance between each point on the curve.

$$\varepsilon_0 = \frac{\sum_j^{N_j} \sqrt{(y_j - y_{j-1})^2 + (x_j - x_{j-1})^2}}{N_j} \tag{1}$$

$$\varepsilon_{hys} = \sum_i^{N_i} \frac{\sum_j^{N_j} \sqrt{(y_j - y_{j-1})^2 + (x_j - x_{j-1})^2}}{N_j} \tag{2}$$

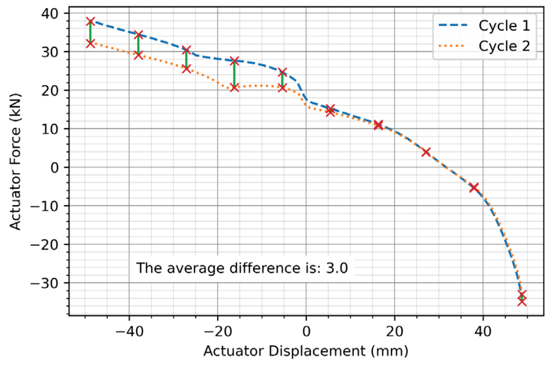


Figure 7: *The average difference in the x/y space between each point for the resampled curves.*

2.4 ENVELOP FUNCTIONS

The Hysteresis library also has functions that can be used to extract data from the envelop of a Hystersis class. A common task researchers will face is finding the backbone of a hysteresis. Because the reversal points in an input Hysteresis are automatically stored, the Hysteresis library can be used to quickly extract the backbone of data coming from a numerical or experimental hysteresis. A function "getBackbone" can be applied to the curve in question to extract the correct points from an input Hysteresis. Figure 8 shows a backbone curve extracted from a numerical model of a seismic damper [11]. By default the getBackbone function will return all reversal points, but the user can provide the number cycles each step in the load protocol has to the function to skip certain points, for example return only the first cycle.

In addition to the getBackbone function, a second function called "getBackboneAvg" can be used. This function extracts the positive and negative backbones of a Hysteresis objects, takes the absolute value of each x/y points, then an averages to the two curves.

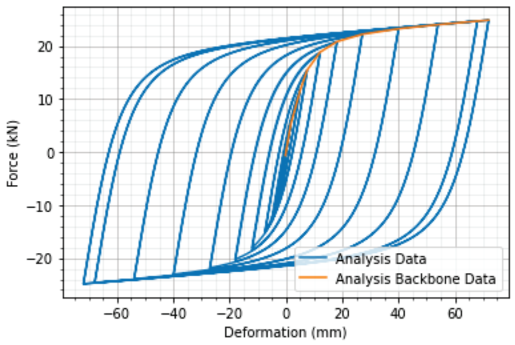


Figure 8: *The extracted backbone curve from a hysteresis.*

A specialized feature of the Hysteresis package is automatically fit an Equivalent Energy Elastic-Plastic (EEEP) Curve to a Hysteresis. In this process, the backbone of the hysteresis is first extracted using the average backbone function and rules outlined in ASTM 2126 [2]. Figure 9 shows a typical use case, where a EEEP curve is fit to the shearwall data from Figure 2.

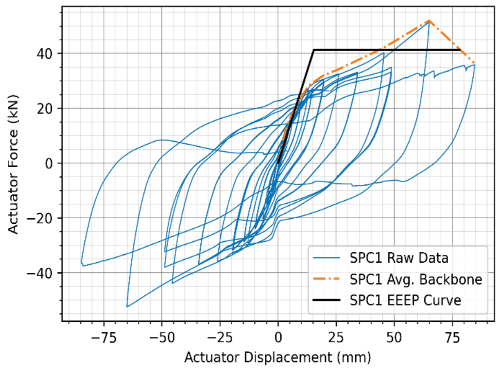


Figure 9*: Raw data from a shearwall test SPC1, overlayed with the calculated back-bone curve and fitted EEEP curve.*

3 CASE STUDIES

3.1 POST PROCESSING OF SHEAR WALL DATA

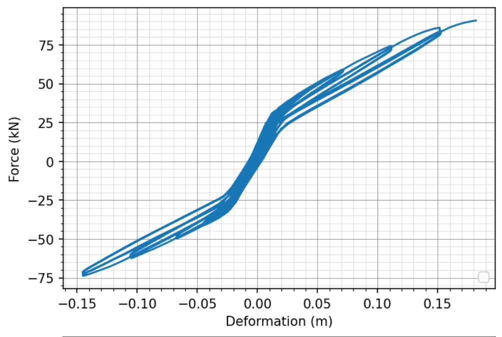
In the following example, experimental data from a self-centring CLT shear wall is postprocessed, then compared to data coming from numerical models of the shear wall. The CLT wall was approximately 4.5m tall, and a post-tensioned steel tendon was used to apply an initial compression force to the wall. Lateral force was applied to the system in a number of cycles, and the force-deformation of the wall was recorded. A detailed discussion of the experimentation can be found in Ganey [10], where it is labelled as TS1. Data for the numerical model is generated in OpenSees [5] using a models similar to those presented in Slotboom [11]. Figure 10 below shows both models, where both data sets have been "cropped", so they both end at approximately 0.15m of drift.

First the experimental data will be checked to see if reversal have been correctly extracted properly. The "plotLoadProtcol" method is used to clearly see which peaks have been picked up, and can be seen in figure XX. It's noted that that not all of the reversal points alternate between positive and negative, and one of the peaks has a value similar to the prior peak. This incorrectly detected point is removed by recalculating the reversal points using a prominence filter that will removes reversal points which do not change much compared to other points. The updated plot can be seen in Figure XX.

Often data from numerical and experimental studies are compared by plotting directly on top of each other, similar to Figure 12. While this can be used to understand general trends in the data, it is often difficult to tell how each curve compares at different points in the analysis. Using the hysteresis package, it's possible easily extract specific cycles of the curve for comparison. Two other metric can easily be employed to see how similar each curve is at different points:

- Cumulative energy observed by each model.
- A direct comparison of the "error" between each curve.

Figure 13 shows a cycle of the produced by the Hysteresis package, while Figure 14 and Figure 15 overview the cumulative energy and error of each cycle respectively. Looking at the Figure 14 it easily be observed that the experimental model absorbed much more energy per cycle than the numerical model did, something which would have been difficult to observe from Figure 12 alone. Through Figure 13, it can be concluded that the experimental hysteresis was asymmetric, and had an open hysteresis that was absorbing energy, while the numerical model was symmetric and had a purely elastic behaviour. Noting quickly noting these differences could allow the researcher make changes to their experimental setup or numerical model if needed.



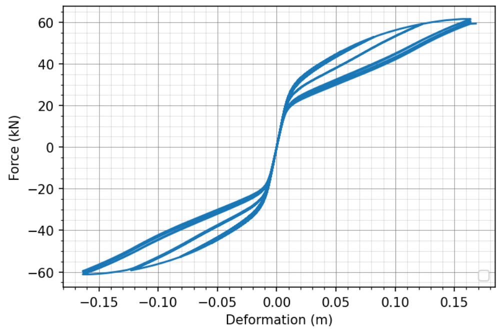
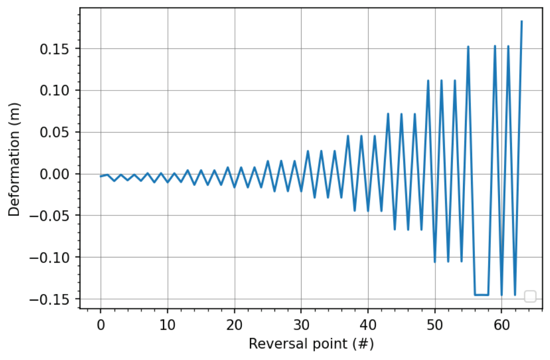


Figure 10: *The raw experimental hysteresis (top), and analysis hysteresis (bottom).*



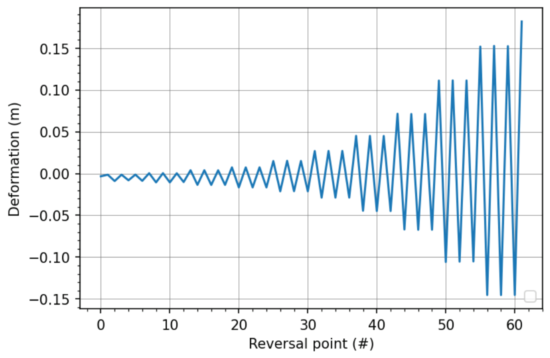


Figure 11: *A plot of detected peaks before (top) and after (bottom) filtering.*

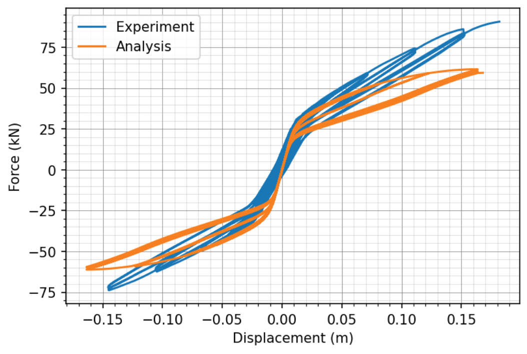


Figure 12: *A typical comparison of experiment/numerical data.*

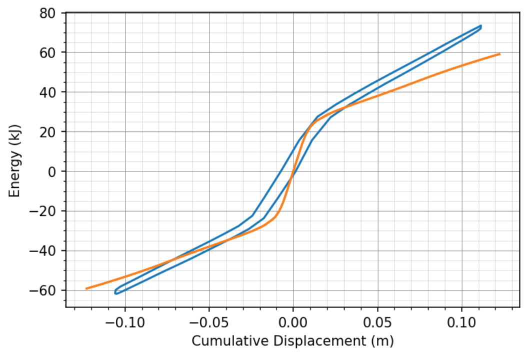


Figure 13: *Experiment and analysis results compared for a single cycle in the hysteresis.*

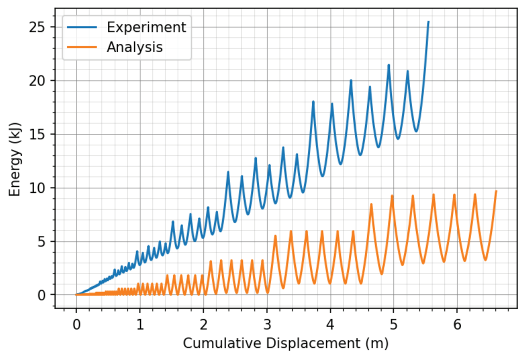


Figure 14: *The energy absorbed by each cycle in the hysteresis.*

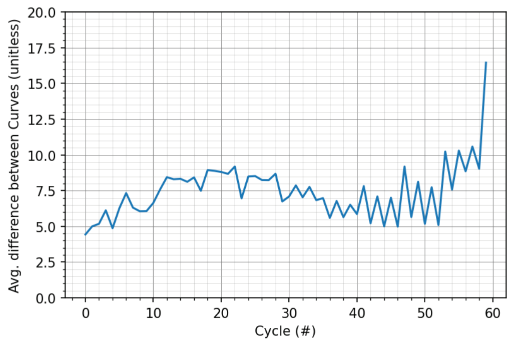


Figure 15: *The average difference between the experiment and numerical analysis.*

3.2 PARAMETER FITTING USING GENETIC ALGORITHMS

A second case study is presented where a nonlinear numerical models are calibrated to match experimental data using a genetic algorithm and the Hysteresis package. As part of a testing program on self-centring shear walls, Ganey tested a number of U-shaped Flexural Plate (UFP) steel dampers. These dampers showed a stable response with a full hysteresis until failure. It's desired to make numerical models of this damper, which can be applied to model of the wall system.

The numerical models of the damper were developed in with the Python implementation of OpenSees, OpenSeesPy [6-7]. A zero-length element was used to model the damper, with a steel02, a Giuffre-Menegotto-Pinto steel material, applied to the element. Force was applied to one of the zero length element's nodes, and the load protocol used matched the testing from Ganey's experiment. To match the damper to experimental results, the OpenSees analysis was first parameterized to take in set of values about the zero-length damper. The significant parameters chosen to be varied were: F_y, the yield strength; E_0, the slope of the elastic tangent; b, the strain hardening ratio between the elastic and plastic tangent lines; and R_0, C_{R1}, and C_{R2}, which are parameters influence how the material transfers from it's elastic to plastic tangent lines.

Once the function was parameterized, the genetic algorithm was set up to iterate through possible values until an optimum was reached. The genetic algorithm used a scheme where a pool of 30 genes was tested over 100 generations. Each gene consisted of the six parameters used to . Initially parameters were randomly generated, with the bounds for the parameters defined in Table 1.

During each generation, all genes in the population were tested by running a OpenSeesPy with those values, then using the Hysteresis Compare function to quantify how similar the experimental and analysis hysteresis was. In each generation, 8 pairs of genes are chosen to pass on their genes to the next round, and results that closely matched experiment were more likely to be chosen. The remaining 14 genes were filled up by randomly generated values within the range of Table 1. Both crossover, where information is swapped between pairs of genes, and mutation, where genes are randomly changed were considered. After 100 generations the analysis was ended, and the optimal value was taken, and is presented in Table 1.

Table 1: *Genetic algorithm parameters and solution.*

Parameter	Minimum	Maximum	Solution
F_y (kN)	0	32	23
E_0 (kN/m)	0	$4x10^6$	$2.9 x10^6$
b	0	0.023	0.002
R_0	0	29	4
C_{R1}	0	1.4	0.57
C_{R2}	0	.23	0.15

Figure 16 compares results from the experimentally tested damper to the solution chosen by the genetic algorithm, showing a close fit between the two hysteresis curves. The Hysteresis package is small part of this process, but it provides a crucial comparison between experiment and numerical data, needed to tell which parameters are good, and which are bad.

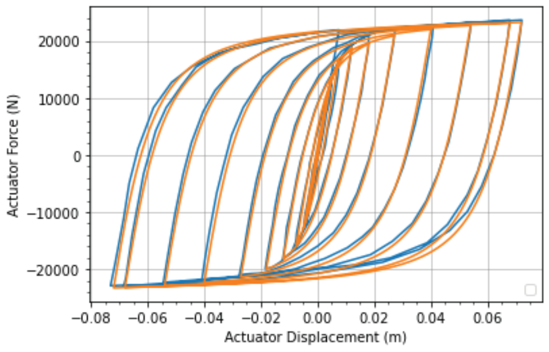


Figure 16: *Experiment (blue) vs. Optimal solution (orange) for the chosen genetic algorithm.*

4 CONCLUSIONS

The Hysteresis Python package can be used to quickly post process structural data, with an emphasis on hysteretic data. The main structure and algorithms used in the package was overviewed, highlighting the three key types of curves (Monotonic, SimpleCycle, and Hysteresis) used to represent structural data. These curve classes all have built in functions for plotting and doing numerical analysis on their data. These classes were designed with customization in mind, and the user can modify these methods by changing the underlying functions in the environment. In addition to the basic functionality, there also exists a number of functions that can be used to extract parameters from an envelop of a Hysteresis object, with some key functions being "getBackbone", and "compare".

Two case studies are then presented, showing how the hysteresis python package can be used to speed up data processing, and enable different types of analyses. The Hystersis package can be used to quickly compare numerical and experimental data in a number of ways. Also, by enabling two hysteresis curves to be compared, the Hysteresis package allows for advanced algorithms or machine learning models to work with Hysteresis dat.

ACKNOWLEDGEMENT

The Author would like to acknowledge data provided by Fast+Epp and Professor Thomas Tannert from the University of Northern British Columbia.

REFERENCES

[1] Krawinkler H, Parisi F., Ibarra L., Ayoub A., Medina R, Development of a Testing Protocol for Woodframe Structures, 2001

[2] ASTM E2126, Standard Test Methods for Cyclic (Reversed) Load Test for Shear Resistance of Vertical Elements of the Lateral Force Resisting

Systems for Buildings, ASTM International, West Conshohocken, PA, 2018

[3] Lowes L, Mitra, N, A Beam-Column Joint Model for Simulating the Earthquake Response of Reinforced Concrete Frames, Pacific Earthquake Engineering Research Center, Report 2003/10, University of Washington, 2003

[4] Filippou, F. C., Popov, E. P., Bertero, V. V., Effects of Bond Deterioration on Hysteretic Behavior of Reinforced Concrete Joints, Report EERC 83-19, Earthquake Engineering Research Center., University of California, Berkeley. 1983

[5] Mckenna F, OpenSees: A Framework for Earthquake Engineering Simulation, Computing in Science and Engineering, 2011

[6] Zhu, Minjie, McKenna, Frank, Scott, Michael H. OpenSeesPy: Python library for the OpenSees finite element framework, 2018

[7] Drexler M, Dires S, Tannert T, Internal perforated-steel-plate connections for CLT shear walls., In proceedings of World Conference for Timber Engineering, Santiago de Chile., 2021

[8] Virtanen P, Gommers R, Oliphant T/, Haberland M., Reddy T., Cournapeau D., Burovski E., Peterson P., Weckesser W., Bright J., van der Walt S., Brett M., Wilson J., Millman K., Mayorov N., Nelson R, Jones E., Kern R., Larson E., Carey C., Polat I, Feng Y., Moore E., VanderPlas J., Laxalde D., Perktold J., Cimrman R., Henriksen I., Quintero E., Harris C., Archibald A., Ribeiro A., Pedregosa F., Mulbregt P., and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python**.** *Nature Methods*, 17(3), 261-272.

[9] Hunter, J. D., Matplotlib: A 2D graphics environment, computing in Science & Engineering, 9(3) 2007

[10] Ganey, R., Seismic Design and Testing of Rocking Cross Lamiated Timber Walls, MASC. thesis, 2015

[11] Slotboom, C., Numerical Analysis of Self-centering Cross-laminated Timber Walls, MASC thesis, 2020