LoadAnalysis Class Documentation

Objectives

1. Develop a predictive model for estimating bending loads in structural systems.
2. Assess the performance and accuracy of the predictive model across various scenarios.
3. Investigate the sensitivity of the model to noise and its impact on prediction accuracy.
4. Examine the influence of point load placement (L_f) on load sensitivity.
5. Extend the model's capabilities to predict temperature changes based on thermal strain inputs.

Expected Outcomes

Our research is expected to yield the following outcomes:

1. A predictive model capable of accurately estimating bending loads within structural systems.
2. Insights into the model's performance across various scenarios, highlighting its sensitivity to noise and load placement.
3. A better understanding of the importance of sensor quality and its impact on load prediction.
4. A versatile model capable of predicting temperature changes based on thermal strain inputs.

Background and Methodology

Skip to the section "Example: Initial Concept 3x (IC3x)" if you want to review the instructions for running the code.

Background

In structural testing, precision in identifying applied loads is a pivotal factor in defining structural limits. Traditionally, load cells are the go-to instruments for measuring these loads. However, situations may arise where direct load measurement is impractical. In such cases, adopting an indirect method becomes essential for load reconstruction. This indirect approach involves measuring a parameter that sheds light on structural deformation induced by the applied load. Utilizing strain gauges, which gauge strain resulting from the load, becomes instrumental in this process. The detection of deformation signifies the presence of a distinctive strain pattern. Capitalizing on this insight, a machine learning algorithm can be deployed, leveraging strain measurements as input to reconstruct the applied load effectively. The code emphasizes load reconstruction using the pseudo-inverse matrix, with detailed information provided in the subsequent subsection. Moreover, this prediction model is not limited to mechanical load but also is employed to predict uniform temperature differences.

Methodology

Generating Training Data using Euler-Bernoulli Beam Model

In this specific context, adopting a simpler model presents distinct advantages, providing comparable results to more complex models while maintaining a more intuitive and straightforward structure. The

key concept involves utilizing the local flexural stiffness of sensor placement points to construct a beam model.

This process is initiated by computing the area moment of inertia at specific sensor locations, followed by multiplying it by Young's modulus. This results in a beam with varying thickness, where each segment corresponds to the flexural stiffness at the respective strain locations.

Another critical parameter derived from sensor placement is the distance between the neutral axis and the surface of each subsection of the beam, denoted as y_i. Three parameters remain consistent with the actual physical structure: the location of the fixed boundary condition, the position where various loads are applied (l_f for loads 1, 2, 3, and 4), and the axial distance between the fixed boundary condition and the sensor positions (l_i for sensors numbered 1 to 32).

$$\{\boldsymbol{\epsilon}\} = \sum \frac{f_{n\times t} \cdot (l_f - l_i) \cdot y_i}{EI_i} \quad \text{for} \quad l_i < l_f \tag{1}$$

These parameters play a pivotal role in generating training data. A simplified beam model is employed, facilitating the straightforward calculation of strain at each sensor using the Euler-Bernoulli beam equation, as indicated by Equation (1). This equation relates moments, flexural stiffness, and the vertical distance between the neutral axis and the sensor location to determine the strain. Consequently, the flexibility to generate a substantial amount of training data is achieved by applying various loads in different combinations and computing the corresponding strains.

Bending Load Prediction Model

The predictive model employed for forecasting bending loads is based on the Moore–Penrose inverse matrix, commonly referred to as the pseudo-inverse matrix model. By utilizing an output vector and an input vector, this model seeks an optimal solution to minimize the error norm. Equation 2 illustrates the structure of the pseudo-inverse matrix, with matrix A columns corresponding to the size of the output vector and rows corresponding to the input vector. A^+ denotes the pseudo-inverse matrix.

$$[A^+] = (A^T A)^{-1} A^T \tag{2}$$

This study focuses on predicting bending loads using bending strain as the input. The Flexibility Influence Coefficients matrix, denoted as $[C]$, serves as one such parameter connecting applied load and strain. In Eq. 3, $[C]$ is the Flexibility Influence Coefficients matrix, ϵ is the bending strain vector with i denoting the number of inputs, and f is the force vector (output vector) with a representing the number of outputs.

$$\left\{\epsilon_{i\times n}\right\} = \left[C_{i\times a}\right] \left\{f_{a\times n}\right\} \tag{3}$$

The training data consists of applied loads and resulting strain, forming the basis for deriving the Flexibility Influence Coefficients matrix, denoted as $[C]$. This process is rooted in matrix algebra, and the result of this computational procedure is succinctly presented in Equation 4.

$$[C_{i\times a}] = \left\{\epsilon_{i\times n}\right\} \left\{f_{a\times n}\right\}^T \left(f_{a\times n}\ f_{a\times n}^T\right)^{-1} \tag{4}$$

After determining the $[C]$ matrix, the subsequent step involves formulating a predictive model that employs strains as input variables to predict applied loads as output values. This is achieved by substituting the matrix A in Equation 2 with the obtained $[C]$ matrix. The resulting prediction model, utilized throughout this study, is presented in Equation 5, where f_p is the predicted load and ϵ is the strain. Notably, the dimensions of the $[C]$ matrix consistently conform to the dimensions $i \times a$.

$$\left\{f_p\right\} = (C^T C)^{-1} C^T \left\{\epsilon\right\} = [C^+] \left\{\epsilon\right\} \tag{5}$$

Thermal Strain Incorporation in the Prediction Model

The predictive model's versatility extends beyond point load prediction to include the anticipation of temperature fluctuations. When a structure undergoes temperature variations, it experiences either expansion or contraction, inducing strain even in the absence of external loads—referred to as thermal strain. The equation below illustrates how to calculate thermal strain based on temperature change, ΔT, and the coefficient of thermal expansion denoted as α. Assuming uniform temperature change throughout the structure, thermal strain can be added to the strain due to mechanical load.

$$\epsilon_{\text{thermal}} = \alpha \Delta T \tag{6}$$

The process of constructing a prediction model remains consistent with the procedure outlined in Section . However, a notable change involves expanding the output vector dimensions from $4 \times n$ to $5 \times n$ to accommodate the thermal strain.

Regarding the training data, the essential requirement for generating C^+ has evolved from a set of 4 to a set of 5. In the updated Equation 7, there is an additional term representing the thermal strain

$$\{\boldsymbol{\epsilon}\} = \sum \frac{f_{n\times t} \cdot (l_f - l_i) \cdot y_i}{EI_i} + \epsilon_{\text{thermal}} \quad \text{for} \quad l_i < l_f \tag{7}$$

Incorporating thermal strain in the prediction model enhances its capability to account for temperature-induced effects, providing a more comprehensive understanding of the structure's behavior under varying conditions.

Algorithm

GitHub Link

The GitHub repository for the Load Predictor project is available at:
`https://github.com/brandoncruz122/Load-Predictor`

Packages

NumPy:

- **Description:** Numpy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.
- **Installation:** Install NumPy using the following command:

```
pip install numpy
```

Pandas:

- **Description:** Pandas is a data manipulation and analysis library for Python. It provides data structures like Series and DataFrame, which are designed for efficient and intuitive handling of structured data.
- **Installation:** Install Pandas using the following command:

```
pip install pandas
```

PyQt:

- **Description:** PyQt is a set of Python bindings for the Qt application framework. It's commonly used for creating desktop applications with graphical user interfaces (GUIs).
- **Installation:** Install PyQt using the following command:

```
pip install PyQt5
```

Seaborn:

- **Description:** Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Installation:** Install Seaborn using the following command:

```
pip install seaborn
```

Matplotlib:

- **Description:** Matplotlib is a 2D plotting library for Python. It produces high-quality graphs, charts, and figures and is often used in combination with other libraries like NumPy.
- **Installation:** Install Matplotlib using the following command:

```
pip install matplotlib
```

Description

The `LoadAnalysis` class is designed to perform load analysis on a structure given various input parameters. The analysis includes the generation of master forces, calculation of strains, validation, thermal effects, and noise studies.

Class Initialization

LoadAnalysis(E, y_loc_master, I_master, F_l_1, F_l_2, F_l_3, F_l_4, max_training_load, min_training_load, increment_training, max_testing_load, min_testing_load, increment_testing, noise_level, BL, sen_locs, alpha, max_training_temp, min_training_temp, max_testing_temp, min_testing_temp)

Parameters:

- `E`: Young's modulus of the material .
- `y_loc_master`: List or array containing the y-locations of sensors.
- I_{master}: List or array containing the moments of inertia at corresponding sensor locations.
- `F_l_1`, `F_l_2`, `F_l_3`, `F_l_4`: Applied loads on the structure.
- `max_training_load`, `min_training_load`, `increment_training`: Parameters for generating master forces for training.
- `max_testing_load`, `min_testing_load`, `increment_testing`: Parameters for generating master forces for testing.
- `noise_level`: Standard deviation of noise in the system.
- `BL`: Boundary condition's location of the structure.
- `sen_locs`: Sensor locations on the structure.
- `alpha`: Coefficient of thermal expansion.
- `max_training_tem`, `min_training_temp`, `max_testing_temp`, `min_testing_temp`: Parameters for temperature variation during training and testing.

Methods

The methods mentioned below are listed in a specific manner. The numbers in front of each method show the step number. This particular order allows the user to have a clear idea as to how an output from a method is used as an input for another method.

1. `generate_master_force(min_training_load, max_training_load, increment_training)`

Generates a master force matrix for training based on specified parameters.

2. `calculate_strain(master_force)`

Calculates the strains in the structure based on the master force matrix.

3. `generate_C(master_training_strain, master_force)`

Generates the C matrix for the training dataset and visualizes it using a heatmap.

4. `Validation(master_force, master_training_strain, C)`

Performs validation on the training dataset and visualizes the results.

5. `generate_master_force_test(min_testing_load, max_testing_load, increment_testing)`

Generates a master force matrix for testing based on specified parameters.

6. `calculate_strain_test(master_force_test)`

Calculates the strains in the structure for testing.

7. `Validation_test(master_force_test, master_testing_strain, C)`

Performs validation on the testing dataset and visualizes the results.

8. `Validation_test_noise(noise, master_force_test, master_testing_strain, C)`

Performs validation on the testing dataset with added noise and visualizes the results.

9. `Thermal_C(calculate_strain, master_force,alpha,max_training_temp,min_training_temp)`

Calculates the C matrix considering strain due to uniform temperature change during training.

10. `Thermal_test(C_thermal, master_force_test, calculate_strain_test, alpha, max_testing_temp, min_testing_temp)`

Predicts uniform temperature change.

11. `noise_study(C)`

Studies the effect of noise on the system by plotting the variance of loads against different noise levels.

Example : Initial Concept 3x (IC3x)

The purpose of this section is to provide an example in which the above-mentioned class and its methods are applied to solve a research problem. Moreover, this section also offers insight into the output provided by the algorithm and how to interpret it. The structure under investigation is a hypersonic missile: Initial Concept 3x (IC3x).

Accessing the Program

The program has been uploaded to GitHub and is accessible to anyone. Download all the content of the **Load-Predictor-main** repository. **Please ensure that all the code and data are contained within the same file.** Once downloaded, the next step is to open **guicalling.py**. Run the program, and if everything proceeds as expected, the user will be presented with a GUI, as depicted in the figure below. If the parameters listed on the GUI are available to the user, simply fill in the appropriate inputs. In this example, click on **Use Preset**. This action will automatically populate all the sections with the parameters of the IC3x. Then click on **Submit**, and the relevant plots and results will be generated. As mentioned earlier, I_master, y_loc_master, and sen_loc are input variables stored in a .csv file and are imported into the program.

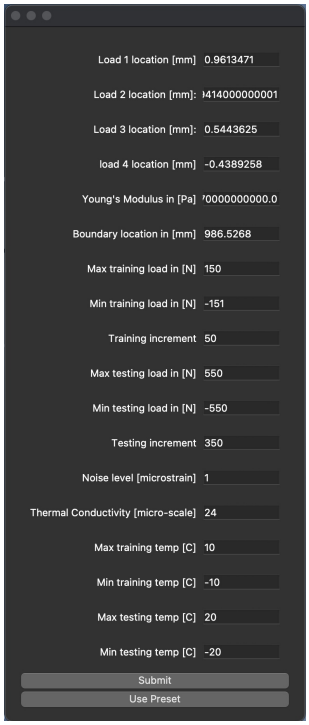

Figure 1: GUI

Geometry and load application set-up

The Initial Concept 3x (IC3x) vehicle, born from an AFRL Munitions Directory sizing study, is a rocket-scramjet two-stage vehicle designed for post-booster jettison operations. Its length is 3.56 meters, transitioning from a pointed nose to a cylindrical fuselage for optimized atmospheric ascent. The IC3x features diamond airfoil fins with a 1 mm Thermal Protection System (TPS) layer, providing stability and maneuverability for high-speed, high-altitude aerospace research and exploration.

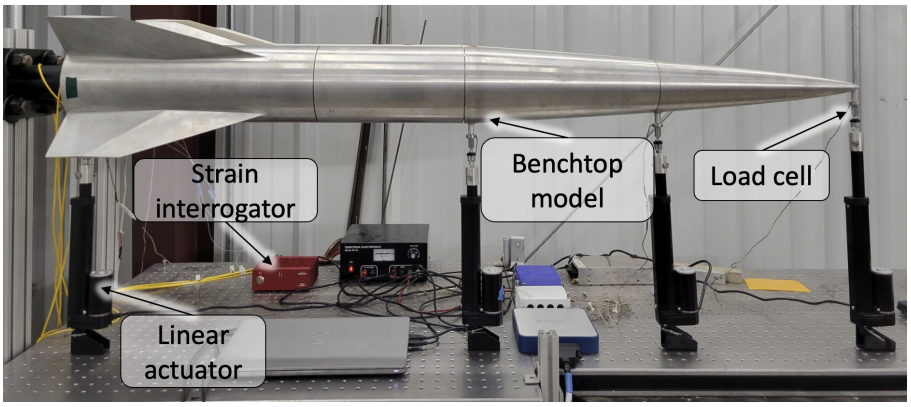

Figure 2: IC3x Benchtop Model Experimental Setup

For experimentation purposes, a scaled-down version of the IC3x, named the IC3x Benchtop Model (IC3x BM), was created by reducing the length to 40 % of the full-scale model. This version facilitated static load experiments, as illustrated in Figure 2. The IC3x BM was equipped with flexures at strategic points, allowing for the attachment of sensors using a fiber Bragg grating system. Figure 3 presents a cross-section of the model, highlighting the varied thickness at the flexures. Aluminum ($E = 70\,GPa$) was the chosen material for the benchtop model fabrication.

Sensor No	$l_i\,(mm)$
1, 9, 17, 25	-299.6022
2, 10, 18, 26	-154.8811
3, 11, 19, 27	129.96
4, 12, 20, 28	212.36
5, 13, 21, 29	380.169
6, 14, 22, 30	509.7552
7, 15, 23, 31	676.0128
8, 16, 24, 32	786.0395

Table 1: Nominal Sensor Longitudinal Distance from Fixed Boundary Condition - Positive Towards the Nose

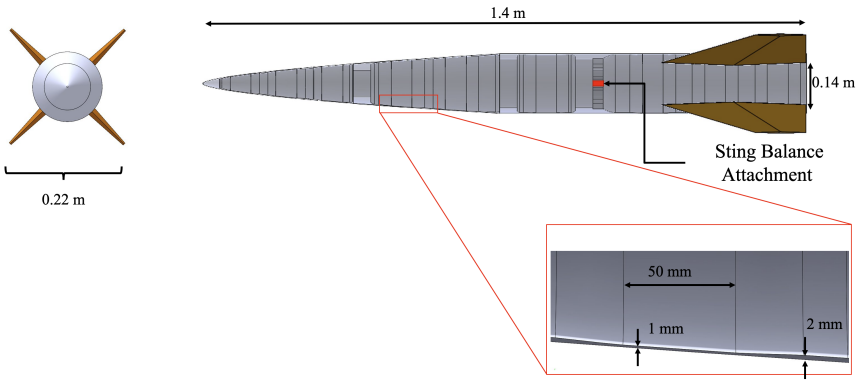

Figure 3: Cross Section of IC3x Nose Section, Highlighting Flexures

Figures 4 and 5, along with Tables 1 and 2, detail the sensor attachment locations and channels in the IC3x BM.

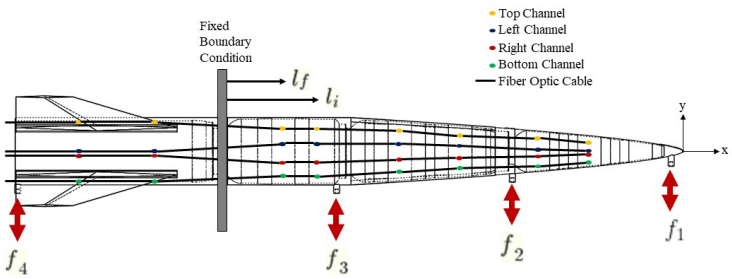

Figure 4: Graphic Representation of Sensor Longitudinal Location with Arrows Indicating Load Positions

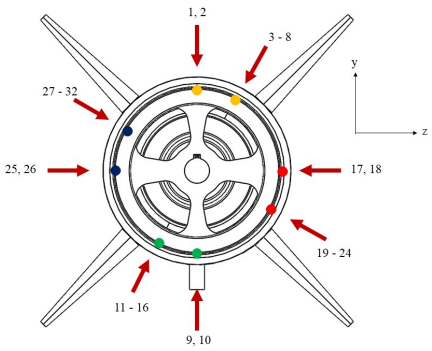

Figure 5: Graphic Representation of Sensor Angular Locations

Sensor No	θ_{sensor} (deg)	**Channel**
1, 2	0	top
3 - 8	30	top
9, 10	180	bottom
11 - 16	210	bottom
17, 18	90	right
19 - 24	120	right
25, 26	270	left
27 - 32	300	left

Table 2: Nominal Sensor Angular Displacement from the Neutral Axis in the Direction of the Positive Y-Axis, with Positive Angular Displacements Clockwise, and Sensor Channel

Load No	$l_f\,(mm)$
f_1	961.3471
f_2	620.9414
f_3	544.3625
f_4	-438.9258

Table 3: Nominal Sensor Longitudinal Distance from Fixed Boundary Condition - Positive Towards the Nose

In this study, each load varies from -150 N to 150 N in increments of 50 N, indicating seven loads for each loading configuration (f_n). With four point loads, a total of 7^4 combinations are available for data generation. This data is utilized in Equation 4 to evaluate the $[C]$ matrix. The matrix is then processed through Equation 5 to create a prediction model that forecasts point loads when supplied with strain data.

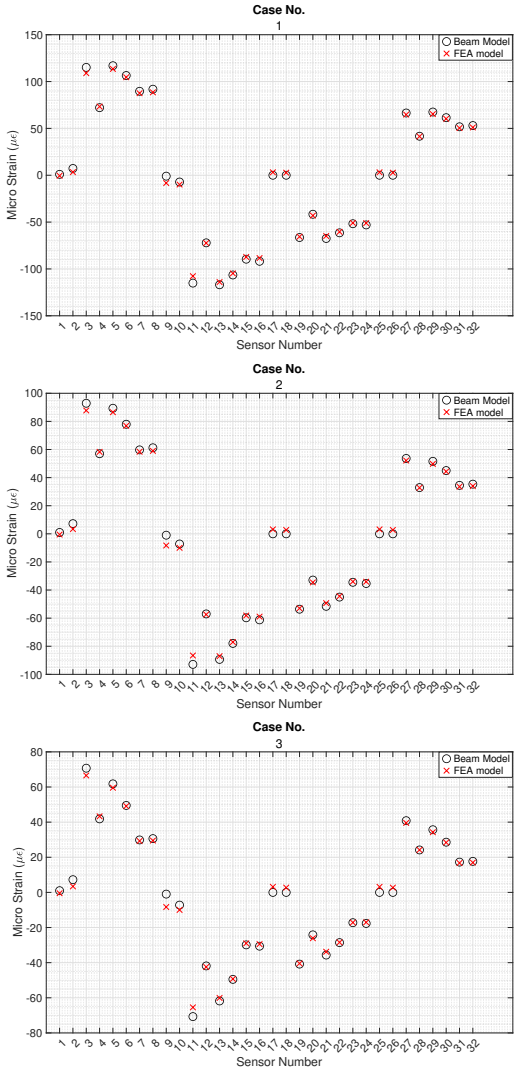

Figure 6: Comparison of Strain Values between Beam Model and FEA Model

To assess the accuracy of the beam model in representing the IC3x model, specific load cases were applied to a Finite Element Analysis (FEA) model of the IC3x. The resulting strain from the FEA analysis was compared to the strain computed using Equation 1. As depicted in Figure 6, a strong agreement is evident between the calculated strain in the beam model and that in the FEA model. Consequently, it can be confidently asserted that the beam model effectively represents the IC3x model.

Results

The goal of this section of the documentation is to review the outcomes produced by the above-mentioned algorithm when applied to the IC3x example. In summary, the algorithm generates a heat map of the prediction model, as well as box plots and direct comparison plots for validation, testing, and testing with noise. Additionally, it provides a direct comparison plot for temperature prediction. As a bonus, the algorithm includes a noise study that offers insight into the sensitivity of load prediction to noise.As a reminder, the algorithm's objective is to predict applied loads based on the input strain. NOTE: For this example, only the sensors installed at the top channel are examined; i.e., only 8 sensors are considered, as opposed to 32 sensors.

Heat Map of C^+

The heat map of the pseudo-inverse matrix C or C^+ provides a graphical representation of the "filter" through which the strain input must pass to predict the loads. This enables the user to assess which load is more susceptible to noise. Higher values for a particular load prediction indicate increased sensitivity

to noise. In the case of IC3x, as illustrated in Fig. 7, load 2, load 3, and load 4 exhibit higher sensitivity compared to load 1; thus, they are more prone to noise. Apart from just examining which load is susceptible to noise, the heat map provides an idea of which sensors have a higher importance for load prediction. For example, in the case of load 4, sensor 2 has a greater influence in predicting load 4 compared to the other sensors.

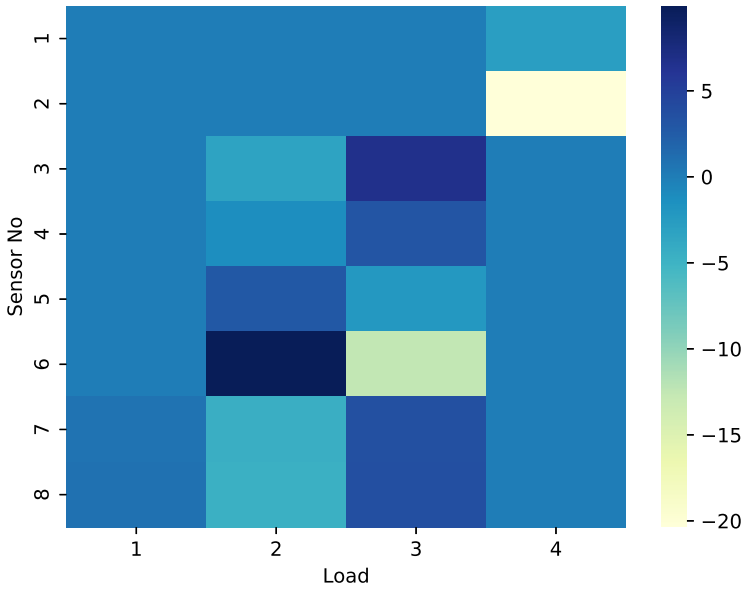

Figure 7: Heat Map of C^+

Table 4: Methods Used

generate_master_force
calculate_strain
generate_c

Validation with Training Data

The validation test is designed to assess if the prediction model works when tested against the training data. The idea is to see if any error occurred during the formulation of either C or the C^+ matrix. This step allows the user to debug any issues before running the model using test cases (explained in the section). As mentioned earlier, the algorithm provides two plots: a box plot and a direct comparison. The box plots summarized the performance of the model. The y-axis shows the normalized difference between predicted and actual load. For each load, the box plots provide information on the average differences, minimum differences, and maximum differences. The direct comparison plot, as the name suggests, compares the predicted and actual load for all training conditions.

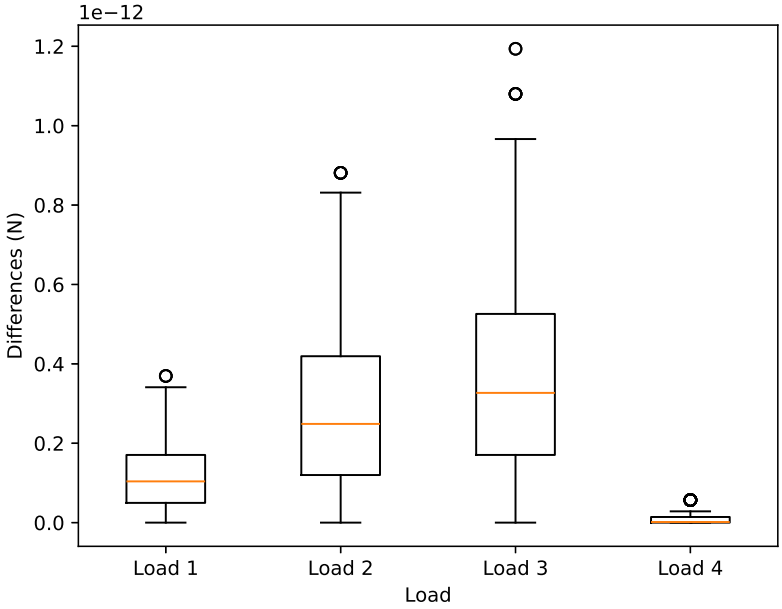

Figure 8: Box Plot of validation

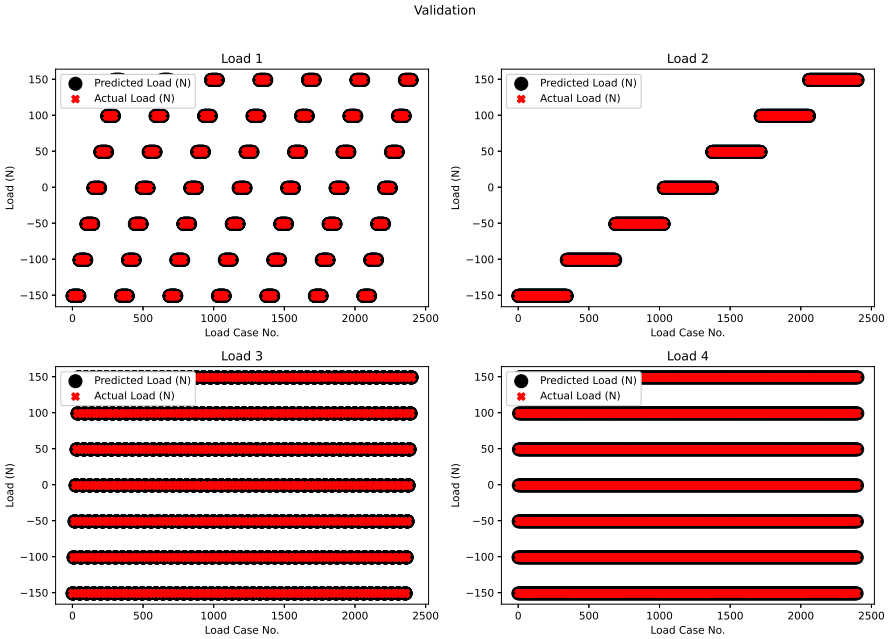

Figure 9: Direct Comparison

Table 5: Methods Used

Validation

In Figures 16 and 9, it can be seen that the loads predicted by the model agree well with the training data. This provides a certain level of confidence that the generated model has the ability to predict loads based on strain input. However, to assess the model's ability to predict loads accurately, it must be tested against conditions different from those encountered during training.

Validation with Testing Data (No-Noise)

One of the features of the algorithm is the development of a vast amount of test data based on the range of loads. Test data refers to load conditions not included in the training module. This is important because the prediction model created using training data is tested against the different loads. Therefore, this tests the robustness of the prediction model. Like the validation with training data, there are two sets of output: box plot and direct comparison. This allows the user to evaluate whether the prediction model can be used in a general case where no noise is introduced. From the figure, the loads predicted using the prediction model produce results that align with the true values.

Table 6: Methods Used

generate_master_force_test
calculate_strain_test
Validation_test

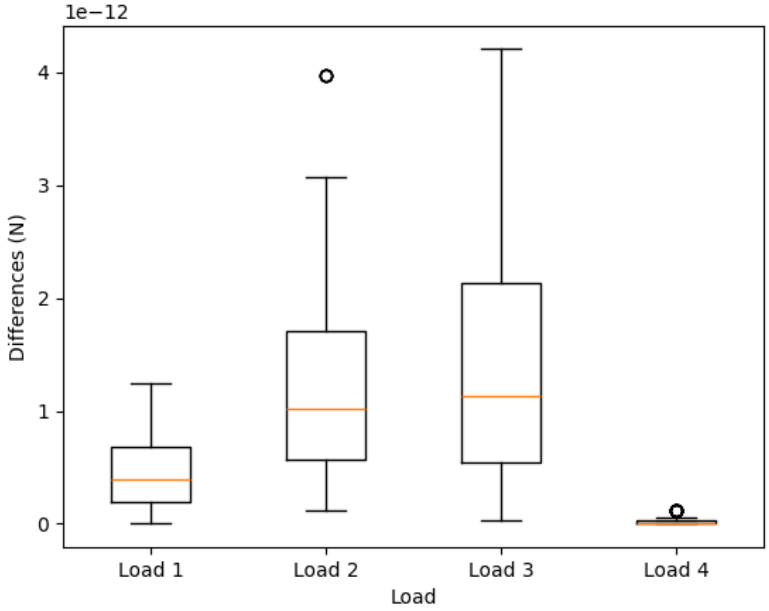

Figure 10: Box Plot of validation using Testing Data (No-Noise)

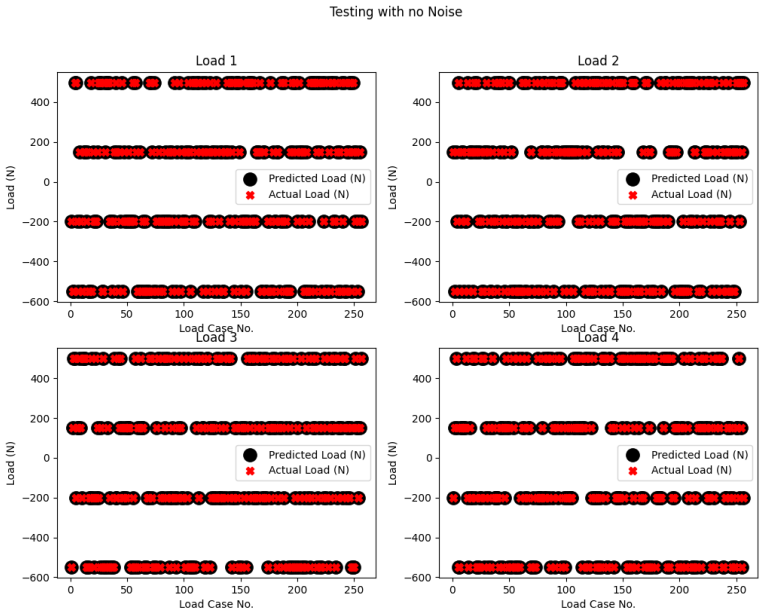

Figure 11: Direct Comparison of Testing Data (No-Noise)

Validation with Testing Data (Noise)

In the real world, data collected using physical sensors—specifically, the FBG system—contains some level of noise. Since the input strain is a calculated value using a simplified model, synthetic data is introduced to simulate real-world conditions. In this algorithm, Gaussian noise is added to the test data, and the noise level is controlled by specifying a standard deviation value in microstrain. Similar to earlier stages, two outcomes are generated: a box plot and a direct comparison.

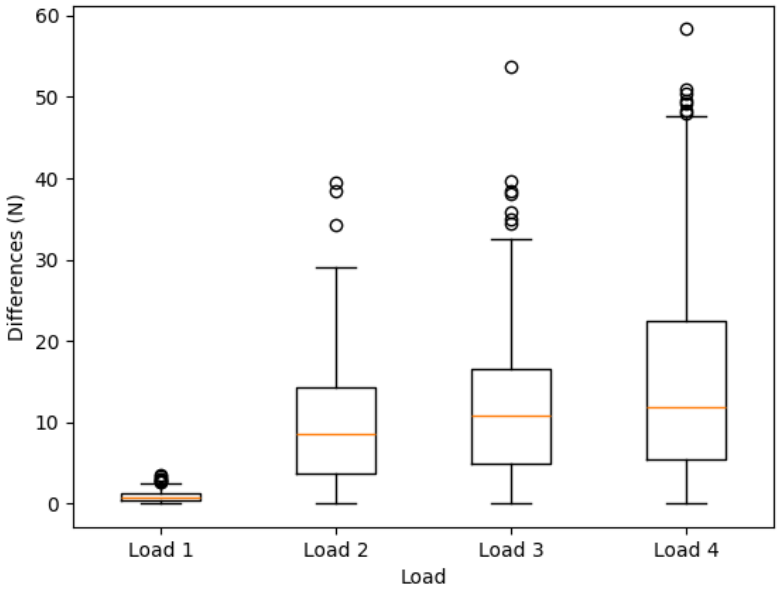

Figure 12: Box Plot of validation using Testing Data (No-Noise)

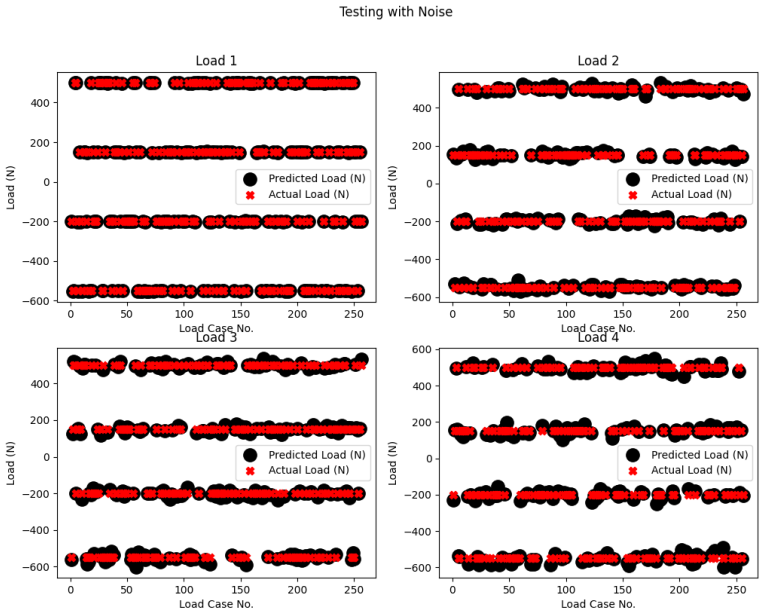

Figure 13: Direct Comparison of Testing Data (No-Noise)

This is where the heat map, shown in Fig. 7, comes in handy. In the previous section, it was noted that loads 2, 3, and 4 would be susceptible to noise, and that is exactly what is depicted in the figure above. Therefore, it can be concluded that C^+ is essentially a measure of the system's sensitivity. In general, sensitivity can be reduced by adding more sensors, but sometimes, it might not be feasible to do so.

Table 7: Methods Used

Validation_test_noise

Temperature Prediction

The method mentioned above is not limited to load prediction; it can also be extended to predict uniform temperature differences. This implies that an update is made to C^+. The mathematical computation is described in the section 'Thermal Strain Incorporation in the Prediction Model'. In this section, only direct comparison is presented, and it can be observed that the model predicts temperature changes quite accurately.

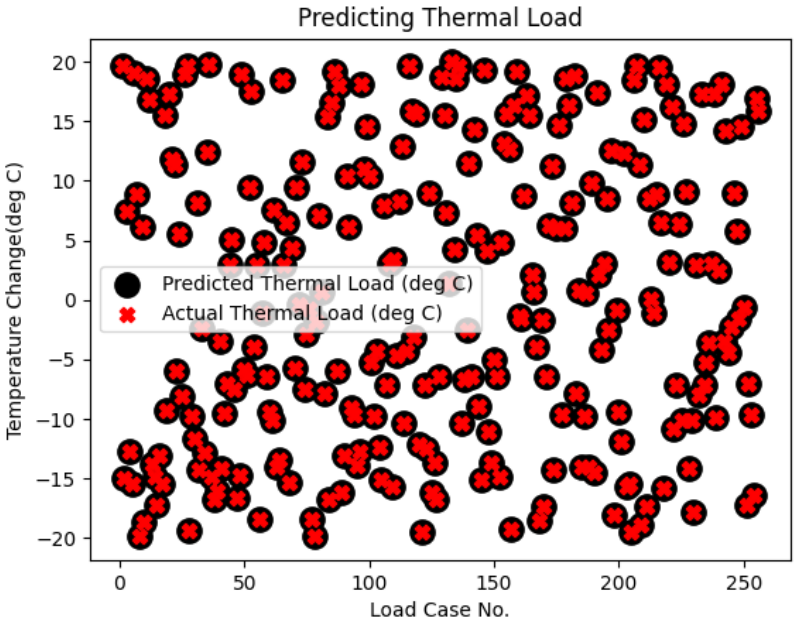

Figure 14: Temperature Change Prediction

Table 8: Methods Used

Thermal_C
Thermal_test

Noise Study

This section is an extension of the discussion on noisy data and sensitivity. Sometimes, users want to evaluate the sensitivity of the prediction model not just for a single noise level but for a range of noise levels, usually by plotting variance versus noise level. From the plot shown below, one can observe that the variance of loads 2, 3, and 4 increases rapidly. This indicates that as the noise level increases, the prediction model produces results that do not match the actual values. A note should be made that this is done only for the load prediction and not for the temperature prediction.

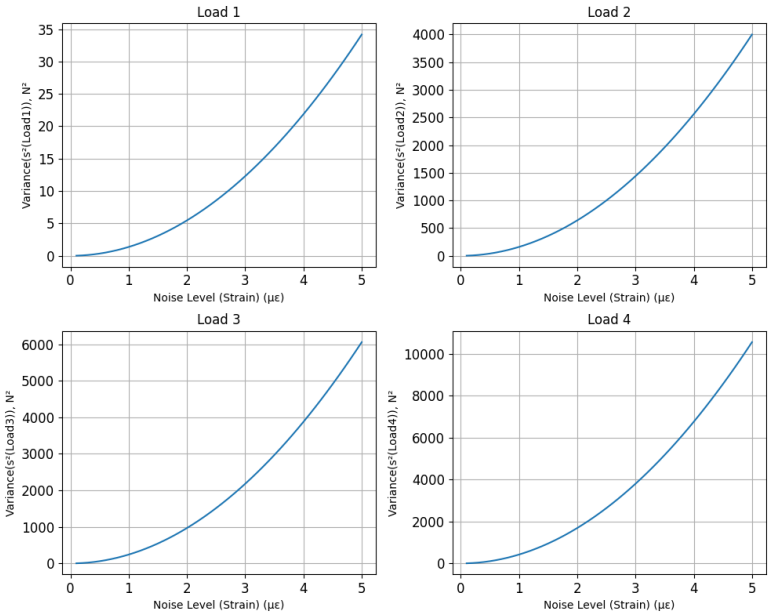

Figure 15: Variance

Table 9: Methods Used

noise_study

Extension of this Study

This section of the documentation demonstrates the practical application of this program in research. In the case of IC3x, it is evident that sensitivity plays a key role in the performance of the prediction model. Therefore, a part of the study should investigate the factors influencing this sensitivity.

Based on the Euler-Bernoulli equation, if all parameters connected to the sensor location are fixed, then only the moment arm becomes an independent variable. After running the program with preset values, the same GUI can be used to change the location of the loads. In this extended study, let's examine the effect of moving load 3 by 30 cm towards the boundary condition. The user only needs to change the value of the load 3 location from 544 mm to 244 mm, then run the program, and the same plots will be generated.

In this section, variation plots will be examined. Clearly, by moving load 3 closer to the boundary condition, noise has a more pronounced effect on the load 3 prediction. The key point here is that the user only has to change the parameter in the GUI, and with a click of a button, they can derive important results.

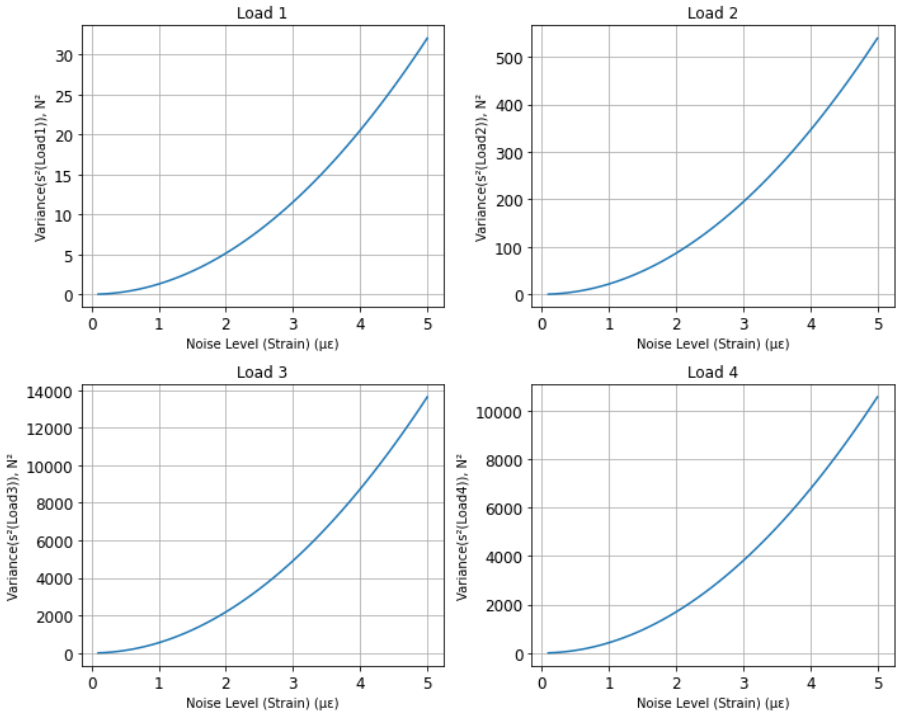

Figure 16: Variance with updated load 3 location

Limitation

This program is not perfect. There are limitations that exists and are listed below.

- The prediction model is only valid if the strain is within the linear regime. In short, the young's modulus does not change.
- The program is able to handle up to 4-point loads.
- There needs to be at least 1 load to the left of the boundary location.
- This is only for the static case. Therefore, no mass or inertial effect is considered.

References

[1] Mathieu Aucejo and Olivier De Smet. A multiplicative regularization for force reconstruction. *Mechanical Systems and Signal Processing*, 85:730–745, 2017.

[2] Z Boukria, P Perrotin, and A Bennani. Experimental impact force location and identification using inverse problems: application for a circular plate. *International journal of Mechanics*, 5(1):48–55, 2011.

[3] Steven L Brunton, J Nathan Kutz, Krithika Manohar, Aleksandr Y Aravkin, Kristi Morgansen, Jennifer Klemisch, Nicholas Goebel, James Buttrick, Jeffrey Poskin, Adriana W Blom-Schieber, et al. Data-driven aerospace engineering: reframing the industry with machine learning. *AIAA Journal*, 59(8):2820–2847, 2021.

[4] Satish Chinchanikar and Avez A Shaikh. A review on machine learning, big data analytics, and design for additive manufacturing for aerospace applications. *Journal of Materials Engineering and Performance*, 31(8):6112–6130, 2022.

[5] HG Choi, AN Thite, and DJ Thompson. A threshold for the use of tikhonov regularization in inverse force determination. *Applied Acoustics*, 67(7):700–719, 2006.

[6] Wei Gao and Kaiping Yu. A new method for determining the tikhonov regularization parameter of load identification. In *Ninth International Symposium on Precision Engineering Measurement and Instrumentation*, volume 9446, pages 285–290. SPIE, 2015.

[7] E Jacquelin, A Bennani, and P Hamelin. Force reconstruction: analysis and regularization of a deconvolution problem. *Journal of sound and vibration*, 265(1):81–107, 2003.

[8] You Jia, Zhichun Yang, and Qiaozhi Song. Experimental study of random dynamic loads identification based on weighted regularization method. *Journal of Sound and Vibration*, 342:113–123, 2015.

[9] Ryan J. Klock. *Efficient Numerical Simulation of Aerothermoelastic Hypersonic Vehicle*. PhD thesis, University of Michigan, 2017.

[10] Xingsheng Sun, Jie Liu, Xu Han, Chao Jiang, and Rui Chen. A new improved regularization method for dynamic load identification. *Inverse Problems in Science and Engineering*, 22(7):1062–1076, 2014.

[11] Linjun Wang, Xu Han, Jie Liu, and Jiujiu Chen. An improved iteration regularization method and application to reconstruction of dynamic loads on a plate. *Journal of computational and applied mathematics*, 235(14):4083–4094, 2011.

[12] Xu Yuan-Zhi, An Bing-Bing, Zhang Dong-Sheng, and Wang Rao-Rao. Novel load identification method based on the combination of tikhonov regularization and singular value decomposition. 2014.

[13] Tao Zhang, Qing Li, Chang-shui Zhang, Hua-wei Liang, Ping Li, Tian-miao Wang, Shuo Li, Yunlong Zhu, and Cheng Wu. Current trends in the development of intelligent unmanned autonomous systems. *Frontiers of information technology & electronic engineering*, 18:68–85, 2017.