AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-2528

Utilizing Machine Learning to Predict Limit Cycle Oscillation Characteristics in Aeroelastic Wing

Daniel A. Hall[1], Ryan W. Lynch[2], Michael T. Hughes[3], Matthew Bryant[4], and Ashok Gopalarathnam[5]

North Carolina State University, Raleigh, North Carolina, 27606, United States

Aeroelastic flutter is the oscillatory response of a structure to specific aerodynamic forces and speeds, that often results in a boundless increase in amplitude. One specific type of flutter, known as limit cycle oscillations (LCO), is instead bounded by a maximum possible amplitude. This bounded oscillation can have a multitude of effects, ranging from positives like energy harvesting, to negatives like structural damage to aircraft. As such, the ability to control this type of response is incredibly useful. To address this, an upstream vortex production system has been employed to impart specific aerodynamic forces onto a downstream wing in a wind tunnel test setting. With the wing's structural parameters known, the frequency of oscillation while in LCO can be calculated. Using both a static bluff body tailored to produce a certain vortex shedding frequency, and a variable frequency disturbance generator that can oscillate at prescribed frequency, LCO excitation and annihilation has been observed. Specifically, using the variable frequency disturbance generator has shown that a window of frequencies surrounding the LCO frequency exists, in which LCO can be excited by upstream vortices. However, the wing response to even the variable frequency disturbances is difficult to predict, so machine learning is employed to better model the system. In particular, a recurrent neural network (RNN) is trained on the wing motion data to be able to accurately predict upcoming states of the wing, which will allow for better control of the wing response.

I. Nomenclature

σ_g	=	Sigmoid Activation Function
f_t	=	forget gate value
W_f	=	hidden layer weights
U_f	=	recurrent connections between the RNN nodes
b_f	=	forget gate biases
x	=	data input to the neural network
y	=	neural network output/predictions
h	=	hidden layer values

[1] Graduate Researcher, Department of Mechanical and Aerospace Engineering.
[2] Graduate Researcher, Department of Mechanical and Aerospace Engineering.
[3] Graduate Researcher, Department of Mechanical and Aerospace Engineering.
[4] Associate Professor, Department of Mechanical and Aerospace Engineering.
[5] Professor, Department of Mechanical and Aerospace Engineering.

II. Introduction

Though it is still a relatively unexplored field, aeroelasticity has been a concern since the dawn of the aerospace industry, even as early as the development of the first craft capable of heavier-than-air flight. For instance, Samuel Langley, a major competitor of the Wright Brothers, created a prototype which experienced aeroelastic flutter and failed structurally[6].

In the coming years, the phenomenon was explored more extensively within the aerospace community, leading to the development of a general study of flutter, performed by Theodorsen for the National Advisory Committee on Aeronautics in 1935[7]. This led to the combination of using vibrational analysis with an aerodynamic forcing function, which could be used to describe a basic aeroelastic system. Using these methods, the flight speed at which flutter occurs could then be calculated, and the design of the aircraft tailored such that the flutter speed lies beyond its expected envelope, thus avoiding flutter altogether. However, both internal and external factors play a large role in the occurrence of flutter, which may cause the analytical predictions to be obsolete. In particular, use of external stores on aircraft effectively alters the stiffness and damping of the vibrational system, which can induce flutter below the calculated flutter speed of the base aircraft[8]. To account for this nonlinearity, variations in control surfaces and other systems to remove energy from the structure have been explored.

Flutter can cause structural damage very quickly due to the rapid amplitude growth from the energy gained from the airflow. However, aeroelasticity is not always a negative phenomenon as its oscillatory motion can be used to produce kinetic energy to be harvested and converted to electrical energy. One form of aeroelasticity, known as limit cycle oscillation (LCO), does not exhibit the monotonically increasing amplitude of oscillation found in traditional flutter and instead grows until plateauing at a certain amplitude. This constant amplitude motion is able to exist through the coupling of the heave or plunge and pitch degrees of freedom, which exchange potential and kinetic energy back and forth. In the past, utilizing an aeroelastic system designed to experience LCO, Bryant and Garcia[9] successfully demonstrated that an airfoil could generate power by converting kinetic energy into electrical energy via piezoelectric devices. Their system utilized the wing's oscillation in the flow alone to produce power; however, others have experimented with the effects of upstream flow disturbances on the aeroelastic system. One such experiment performed by Kirschmeier and Bryant showed that wings operating in tandem could produce greater energy generation by tailoring the motion of the upstream wing to produce greater oscillations in the downstream wing[10].

In a similar vein, Gianikos and Kirschmeier[11,12] utilized a bluff body upstream of the aeroelastic wing to modulate LCO. The disturbance generator was initially developed as a stationary rectangular bluff body that was tailored to shed vortices at 12 Hz. The bluff body then evolved into a rotating cylindrical bluff body[13] with central splitter plate that would shed vortices at its frequency of oscillation. Using this system, control of the downstream wing was achieved, and LCO was able to be both excited and annihilated accordingly. Through oscillating the cylinder at various frequencies around the LCO frequency, it was found that a window exists in which LCO can be excited. By using the rotating bluff body, the wing response could be controlled more consistently; however, there were still some inconsistencies at the upper and lower portions of the LCO window. To account for this, a more robust controller for the system is sought, and one method to supply this is through training a machine learning model to predict the motion's next steps.

Machine learning is a broad and diverse field, ranging from iterative statistical modeling to artificial intelligence. One of the most prominent forms of machine learning is the artificial neural network (NN), which is modeled after the human brain's structure and learning capabilities. Neural networks range greatly in form and function, but most

[6] Garrick, I. E., and Reed, W. H., "Historical Development of Aircraft Flutter," *Journal of Aircraft*, Vol. 18, No. 11, pp. 897-912, 1981.

[7] Theodorsen, T., "General Theory of Aerodynamic Instability and the Mechanism of Flutter," Tech. Rep. 496, NACA, 1935.

[8] Goodman, C., Hood, M., Reichenbach, E., and Yurkovich, R., "An Analysis of the F/A-18C/D Limit Cycle Oscillation Solution," 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2003.

[9] Bryant, M., and Garcia E., "Development of an Aeroelastic Vibration Power Harvester," Proc. SPIE 7288, Active and Passive Smart Structures and Integrated Systems, April 2009.

[10] Kirschmeier, B., and Bryant, M., "Experimental Investigation of Wake-Induced Aeroelastic Limit Cycle Oscillations in Tandem Wings," *Journal of Fluids and Structures*, Vol. 81, pp. 309-324, 2018.

[11] Gianikos, Z. N., Kirschmeier, B. A., Gopalarathnam, A., and Bryant, M., "Limit Cycle Characterization of an Aeroleastic Wing in a Bluff Body Wake," *Journal of Fluids and Structures*, Vol. 95, 2020, p. 102986.

[12] Kirschmeier, B. A., Gianikos, Z., Gopalarathnam, A., and Bryant, M., "Amplitude Annihilation in Wake-Influenced Aeroelastic Limit-Cycle Oscillations," *AIAA Journal*, Vol. 58, No. 9, 2020, pp. 4117–4127.

[13] Rockwood, M., and Medina, A., "Controlled Generation of Periodic Vortical Gusts by the Rotational Oscillation of a Circular Cylinder and Attached Plate," *Experiments in Fluids*, Vol. 61, No. 3, 2020.

operate via the same general principles. Typically, a network is given a paired set of inputs and corresponding outputs, or labels, and asked to create an equation that maps the inputs to the outputs. The goal of this mapping is to produce a function capable of predicting the outputs of a new set of inputs it has not yet encountered[14]. The shapes of these inputs and outputs each can vary from vectors to tensors, and do not need to be the same. In the field of natural language processing, the inputs of many networks are sentences fed into the model in the shape of a vector. The output could be an integer value labeling the sentence with the sentence's language, a vector of the sentence translated to a new language, or any other shape the problem might require. In the discipline of computer vision, networks are fed images in the shapes of tensors, two dimensions for the x and y positions of the pixels, and a third dimension to denote the RGB color values for each pixel. The outputs again could be whatever shape the problem statement would benefit from most.

One of the most common forms of neural network is the fully connected (FC) neural network. FC neural networks are made up of an input layer, n number of hidden layers, and an output layer. These layers each contain a number of "nodes," with each assigned a weight and bias during the training of the neural network (often initialized with random values). These weights and biases are then iteratively changed in order to minimize the error calculated when comparing the model's output and the true output originally given to the model.

For time-dependent data, the type of model typically employed is known as a recurrent neural network (RNN), which differs from traditional neural networks by retaining the hidden layer between each timestep[15]. In doing so, the RNN carries over information from the one step in the past to be used in evaluating the current step. A basic graphic of a typical RNN's architecture is shown in Fig.1. However, typical RNNs tend to experience an issue known as vanishing gradient problem. For a standard neural network, the values of the weights in the hidden layers are determined by updating the previous weights in proportion to the gradient of the error[16]. When the gradient becomes too small, as can happen with recurrent neural networks working on small timescale problems, the model does not update its weights accordingly, thus preventing it from fully training.

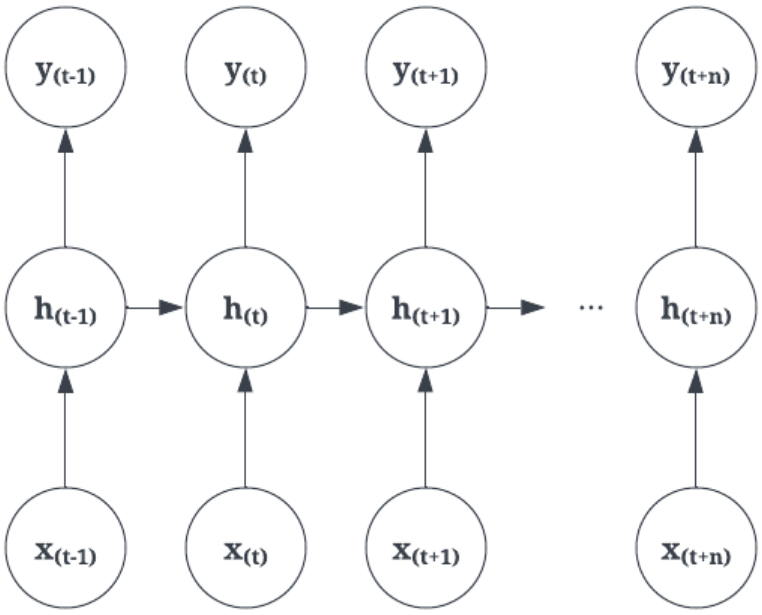

Figure 1. Basic recurrent neural network architecture

[14] Schwing, A. G., & Urtasun, R. "Fully connected deep structured networks". arXiv preprint arXiv:1503.02351, 2015.

[15] Sherstinsky, Alex. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," Physica D: Nonlinear Phenomena, Volume 404, 2020, 132306, ISSN 0167-2789, https://doi.org/10.1016/j.physd.2019.132306.

[16] Sherstinsky, Alex. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," Physica D: Nonlinear Phenomena, Volume 404, 2020, 132306, ISSN 0167-2789, https://doi.org/10.1016/j.physd.2019.132306.

To avoid the vanishing gradient problem in RNNs, several variations on the model architecture have been developed. One specific type is the Long Short-Term Memory (LSTM) unit. This type of model "remembers" data from previous timesteps just like a typical RNN, but this model can choose to forget that data[17]. It does so by adding to the RNN architecture three gates: the "input" gate, the "output" gate, and the "forget" gate. Just like with the basic RNN, the inputs of the current timestep consist of the current timestep's data and the hidden layer of the previous timestep. The forget gate is then employed by using the hidden layers saved from the previous timestep, performing a sigmoid function, shown in Eq. (1), on the data, and then multiplying that value element wise by the values in the cell. The sigmoid activation function scales all values on an infinite domain to have a range between 0 and 1, which essentially allows lower weights to be forgotten. The overall equation for the forget gate can be seen in Eq. (2) [18].

$$\sigma_g = \frac{e^x}{e^x+1} \tag{1}$$

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

Using this forget gate, the LSTM is able to avoid the disappearing gradient problem by ignoring long repeating data trends that do not vary from timestep to timestep. This is useful for the LCO data due to the extended lengths of time in which the wing undergoes either buffeting or LCO with little to no state changes. As such, the LSTM was tested as the baseline model for the aeroelastic dataset and was trained accordingly.

III. Methodology

A. Data Preprocessing

Using the experimental results of the wind tunnel aeroelasticity tests performed by the ISSRL, a dataset was constructed, consisting of fifty randomly ordered tests. Each test sample consisted of a full time-history of aeroelastic pitch and heave data, recorded via a linear encoder (heave) and two rotary encoders (pitch). As was previously stated, modulating the frequency of the upstream disturbance generator produced an interference pattern in the wing response, leading to amplitude growth and decay cycles, and sometimes limit cycle oscillation. A full time-history of the wing's pitch can be seen in Fig. 2.

Before the data could be fed to the model for training, it had to first be preprocessed. The preprocessing consisted of a twofold process. First, the data had to be interpolated to equalize the number of timesteps, as each test case consisted of different length timescales, while all retained the same sampling frequency of 500 Hz. Second, the data had to be labeled with appropriate LCO cycle states. In order to accurately describe the system, four distinct states were described within the data: 1.) simple buffeting, 2.) amplitude growth, 3.) amplitude decay, and occasionally 4.) LCO. As such, each data point had a value appended corresponding to the label for its state, which allowed for a full description of the time series, shown in Table 1.

Table 1: Characteristics of the four LCO states

State	Amplitude, at current point (A)	Amplitude difference (ΔA)
Buffeting	low	low
Growth	low	high
Decay	high	high negative
LCO	high	low

[17] Chen, Y., Zhang, S., Zhang, W., Peng, J., Cai, Y. "Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting", *Energy Conversion and Management*, Vol. 185, pp. 783-799, 2019.

[18] Chen, Y., Zhang, S., Zhang, W., Peng, J., Cai, Y. "Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting", *Energy Conversion and Management*, Vol. 185, pp. 783-799, 2019.

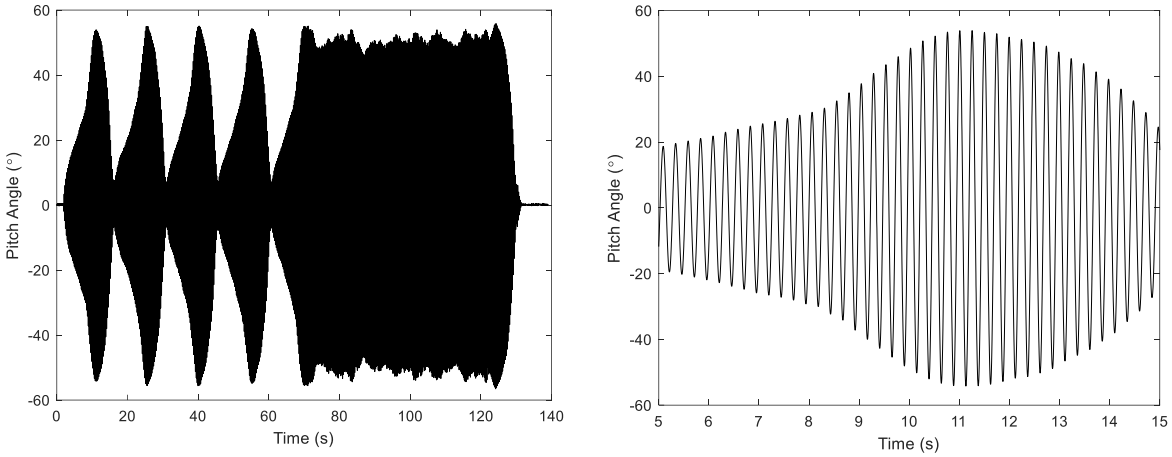

Fig. 2 Full wind tunnel time-history (a.) and 10 second time-history (b.) for disturbance generator frequency of 3.99 Hz

For LCO cases, four distinct regions appeared in the data, so a more consistent automated approach could be employed. In order to identify the states of the wing response, first the critical points of the pitch amplitude curve were found. These amplitude values at the current critical point, amplitude difference from the current critical point to the next, and the time difference from the current critical point to the next, were calculated. Using these three descriptions, the data could be separated using a statistical method known as k-means clustering[19], which generates a given number of randomized centroids. With the built-in MATLAB function, "kmeans," the centroids are then iteratively repositioned to minimize the average distance between them and their respective data clusters, until the data has distinct centroids for each region. For the LCO cases, this clustering resulted in four distinct regions in the data,

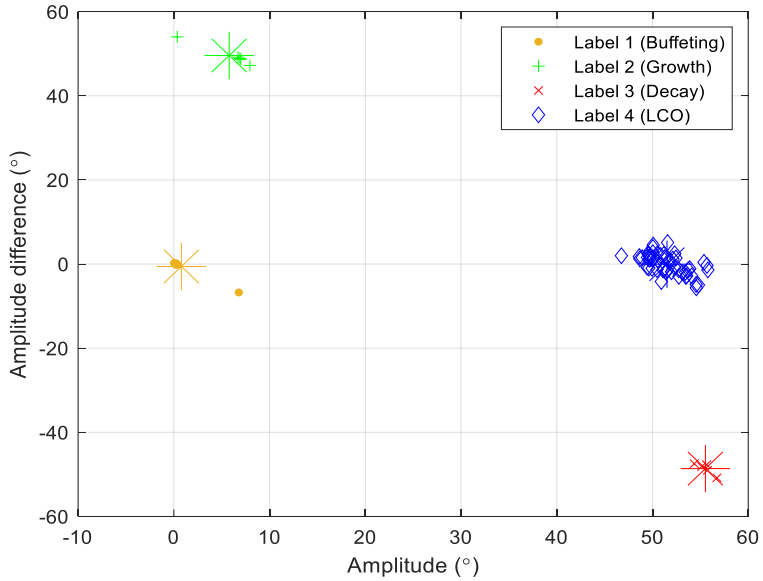

Figure 3. K-means clustered critical points of pitch amplitude for disturbance generator frequency of 3.99 Hz

[19] Mathworks. (n.d.). K-means Clustering. MATLAB Documentation: K-means Clustering. Retrieved December 13, 022, from https://www.mathworks.com/help/stats/kmeans.html

corresponding to the buffeting, growth, decay, and LCO data labels. There were some cases where the clustered data was not discretely separated, as they showed a similarity between the upper amplitude growth regions and LCO. This similarity is confirmed by the physical tests, in that at these amplitudes of the growth cycle, deactivating the disturbance generator most consistently leads into self-sustaining LCO. One such clustered dataset can be seen below in Fig. 3. Using the clustered data, the critical points were labeled accordingly, and the points in between were given the previous critical point's label. Once labeled, the data is ready to be fed into the model for training.

Labeling the non-LCO cases proved to be more difficult as the range of amplitudes between the cases was extremely inconsistent. This was due to the fact that as the frequency of the disturbance generator moves away from the LCO frequency, less constructive interference is seen. In effect, the further away from that desired frequency the disturbance generator oscillated, the lower the amplitude and amplitude range. As such some non-LCO cases seemed to follow the same trends as LCO cases, experiencing growth, decay, and buffeting regions that resembles a scaled down LCO. However, once the disturbance generator was deactivated for these cases, the motion would completely die out, showing that this region was simple buffeting. On the other hand, some non-LCO cases simply ramped up to oscillate at the frequency of the bluff body oscillation at a low amplitude, with no distinct growth and decay cycles. Because of this behavior, kmeans would often misidentify data points in non-LCO cases as flutter. As such, the method

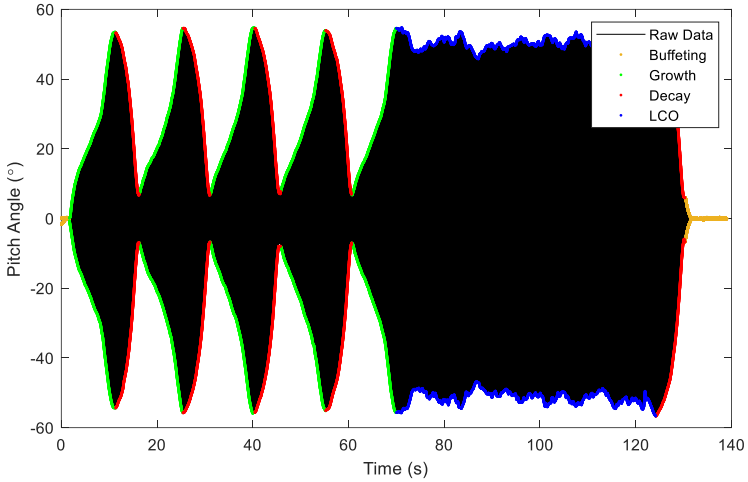

Figure 4. Labeled test data for disturbance generator frequency 3.99 Hz (LCO case)

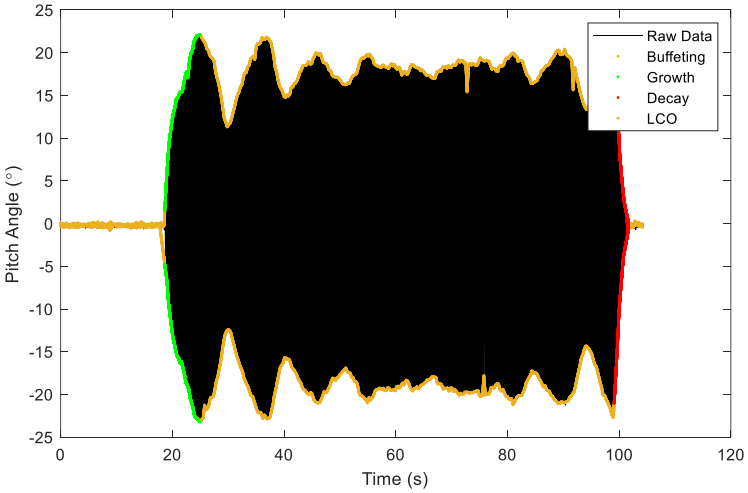

Figure 5. Labeled test data for disturbance generator frequency 3.84 Hz (non-LCO case)

of using k-means to cluster the data similarly to LCO cases would not work for non-LCO cases. Using a manual labeling method, this could be overcome, as the states of growth, decay, and buffeting can be easily seen by eye. A

fully labeled LCO and non-LCO case can be seen in Figs. 4 and 5. As is shown, a similar growth and decay region can be seen due to the interference between the disturbance generator and the wing. However, due to their being outside the LCO window, non-LCO cases fail to reach the amplitudes of oscillation that allows for self-sustaining LCO.

B. Preparing the Labeled Data for Training the Model

The training data used was selected to contain both cases that experienced LCO and cases that experienced simple buffeting in an even ratio, so that the model could learn how to account for those differences. The data was separated into training, validation, and testing sets on an 80%-10%-10% split, with the testing data saved for prediction post training the model. The order of the individual datasets was then randomized once to keep the same training and testing splits across multiple runs. Due to runtime constraints, the full dataset was downsampled from its original frequency of 500 Hz to 50 Hz. In doing so, smaller batch sizes could span more of the time-history, allowing for better learning of the datasets' trends. Once the downsampled data is separated, the training, validation, and test sets were concatenated into separate matrices of wind tunnel data and a corresponding vector of labels.

The primary goal of the model was to predict the LCO state over a specified time input. In order to do so, three different models were trained to compare the effects of varying the prediction distance. The first model made predictions on the current time step. The second made predictions based on the labels 5,000 timesteps beyond the current state, which roughly translates to 10 seconds based on the 500 Hz sampling frequency. The third model made predictions based on the labels 10,000 timesteps beyond the current timestep, roughly 20 seconds into the future. To train the model based on the forecasted flutter behavior, the data was altered so that the input $\boldsymbol{x}$ data was given labels corresponding to those from the given offset ahead of the current point. For the third model, this left the first ten thousand $\boldsymbol{y}$ labels and last ten thousand $\boldsymbol{x}$ values without pairs, and they were thus removed from the dataset before once again splitting the data into training, validation, and test sets. The model was then fed the newly labeled data and its performance was compared to the original model's performance on the unaltered data.

One important thing to note is that by shifting the data in this manner, ten thousand datapoints were removed from each experiment's collection of data. Since the data from fifty experiments was evaluated to create the original dataset, shifting the labels removed a total of 250,000 and 500,000 datapoints from the total dataset used to train, evaluate, and test the 5,000 and 10,000 step prediction models respectively. However, shifting the data's labels and removing the thousands of datapoints from the total dataset was only a loss of about 3.33 and 6.667 percent of the original dataset, which is small enough that it would not noticeably hinder the performance of the model. In further experimentation, allowing the wind tunnel setup to collect data for an additional twenty to thirty second buffer period at the end of testing will avoid this issue entirely. This additional buffer would allow for less of the valuable data to be lost due to the dataset shift for prediction.

C. Model Architecture and Training

As was previously discussed, standard RNNs suffer from the vanishing gradient problem when dealing with time-histories with long periods of the same behavior. Because of this, an LSTM model was used as the baseline, which modifies the traditional RNN with a function known as a forget gate. This function acts to remove data trends that repeat for extended numbers of timesteps, similar to the LCO state from the wind tunnel experiments. A smaller, simpler neural network was utilized for these tests to verify the efficacy of machine learning to accurately predict LCO states. The model's architecture is shown in Table 2. The input size corresponds to the 4 label states of buffeting, growth, decay, and LCO. The 128 hidden layer nodes was chosen as it was a standard size seen in other RNN and LSTM applications. The number of hidden layers was set to 2, as expanding beyond that number of layers caused the model to not train properly. And the final output size corresponds to the label states once more. In the future, these parameters can be adjusted to produce a more efficient model for this data.

Table 2: Model Architecture

Model type	LSTM
Input Layer Size	4
Hidden Layer Size	128
Number of Hidden Layers	2
Output Size	4

Once the model was initialized with these parameters, the data needed to be split into batches for training. First, the data for the no prediction, 5,000 step prediction, and 10,000 step prediction consisted of different sized datasets, so each had to be split into different-sized batches. For the same-step model, each of the 50 datasets consisted of

15,000 datapoints after downsampling, so the data was divided into batches of 3,000 points. For the 5,000 timestep prediction model, each dataset consisted of 14,500 points after downsampling and offsetting, so the data was divided into batches of 500 points. And for the 10,000 timestep prediction model, each dataset consisted of 14,000 points after downsampling and offsetting, so the data was divided into batches of 2,000 points.

After dividing the data into its batches, the training of the model was the next step. The training process consisted of several iterations or epochs of fitting the weights and biases of the initialized model to the training data and its corresponding labels. The batches are fed to the model in a randomized order so that it could learn trends existing in smaller slices of the time-history, rather than between longer time-trends between the batches. Once the weights of the neural network are established, the validation data is passed to the model without its labels so that the model can generate predictions of each datapoint's state. These predictions are then compared to the true labels to find the system's accuracy for that iteration. Each time the model's accuracy increases, the weights and biases of the network are saved for the start of the next epoch. The adjustment of the model weights is driven by the ADAM optimizer, which is well-known for its implementational and computational efficiency[20]. This process is continued for 50 epochs, and the model with the highest accuracy is saved for use in generating predictions. The loss of the network is recorded at each epoch to verify the training of the model. The training and validation loss curves of the 10,000 timestep prediction network can be seen in Fig. 6. As can be seen, both losses are lowered over the course of training, until the curve reaches a plateau near the 50th epoch. Using more epochs did not serve to improve the loss substantially, as further training had little effect on improving the validation accuracy, so the maximum remained at 50 epochs.

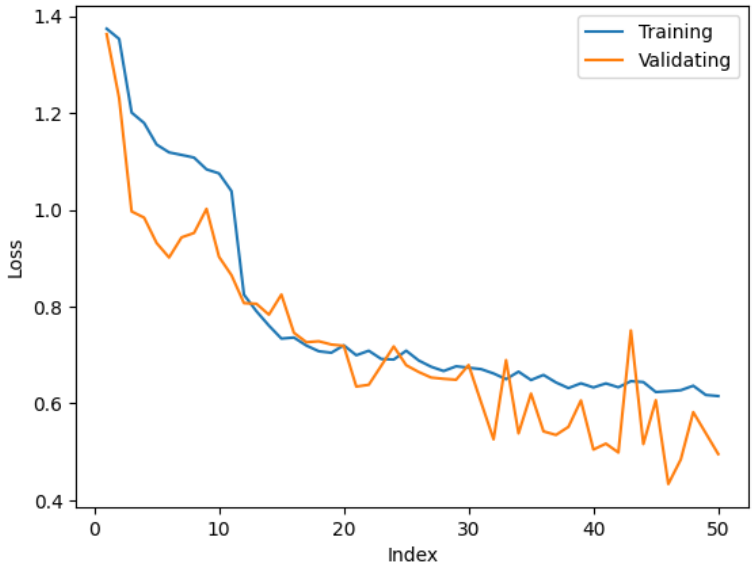

Figure 6. Model Predictions with no timestep offset

[20] Kingma, Diederik P., Ba, Jimmy. "Adam: A Method for Stochastic Optimization," arXiv preprint, arXiv:1412.6980, 2017.

IV. Results

Once trained and validated, the models were next given the unlabeled testing data, which consisted of 2 non-LCO and 3 LCO test-cases. Just like with the validation portion of training, the testing consisted of the model reading the data and generating predictions for the label states at each timestep. For the same timestep model, this label is what the model calculates is currently happening; and for the timestep offset prediction models, label is what the model calculates should happen after the prescribed time offset. These predicted labels were then compared to the true labels to produce a model accuracy for the against the test data. This performance can be seen in Figs. 7-9, which display the accuracies of the model predictions (x-axis) against the true label (y-axis). The values along the diagonal of the matrices represent the accuracy of correctly identifying the true labels, with the values 0-3 corresponding to buffeting, growth, decay, and LCO, respectively. The off-diagonal values represent labels that the model incorrectly identified. As can be seen, the same time-step model excels at identifying buffeting and growth states, with 97% accuracy for each; however, it only correctly identifies decay with 64% accuracy, as it misidentifies decay states as LCO, growth, and buffeting, with guess rates of 17%, 10% and 8% respectively. It does, however, perform well at identifying LCO data points as well, with 88% accuracy. The 5,000 timestep prediction model performs similarly for buffeting and LCO, with 94% and 92% respective accuracies, while it mostly mislabels growth as decay. Similarly, the 5,000-step predictor correctly identifies decay 51% of the time but misidentifies decay regions as LCO 35% of the time. The 10,000-step predictor performs similarly well for buffeting and LCO, with 91% and 84% respective accuracies. However, it performs better than the 5,000-step predictor for growth, accurately predicting that proper state with 83% accuracy, with incorrect predictions evenly dispersed between the other three states. This improvement is thought to be due to the increased batch size from 500 to 2,000, which allows the model's weights to be based on a larger slice of time-history data at once while training. It performed significantly worse than either of the previous models in decay, though, only correctly identifying that state 32% of the time, with an even dispersion of mislabeling between growth and LCO. Across all three models, the decay region was most inaccurately identified, but the accuracy was also negatively correlated with prediction distance. Their total accuracies also decreased with increased prediction distance, with total test accuracy of 91%, 84%, and 82% for the no timestep, 5,000 timestep, and 10,000 timestep models, respectively. Overall, though, the model shows good accuracy at both the current timestep and a set number of timesteps into the future.

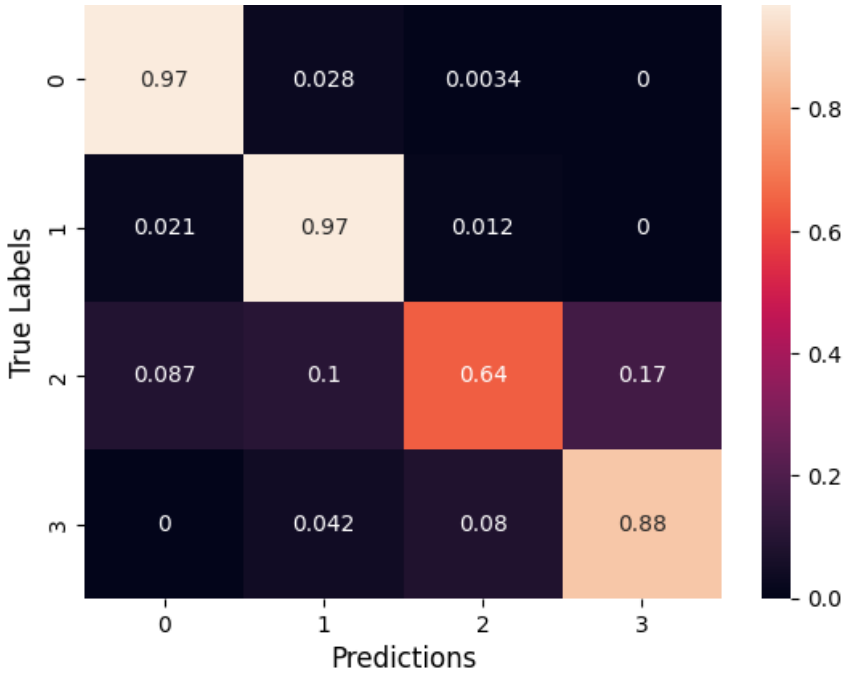

Figure 7. Model Accuracy with no timestep offset

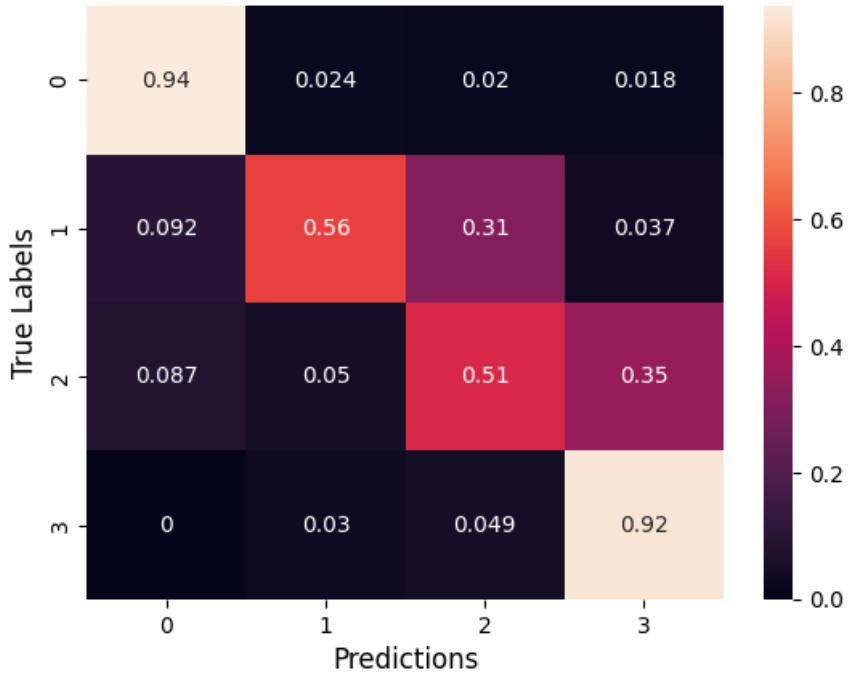

Figure 8. Model Predictions with 5,000 timestep prediction offset

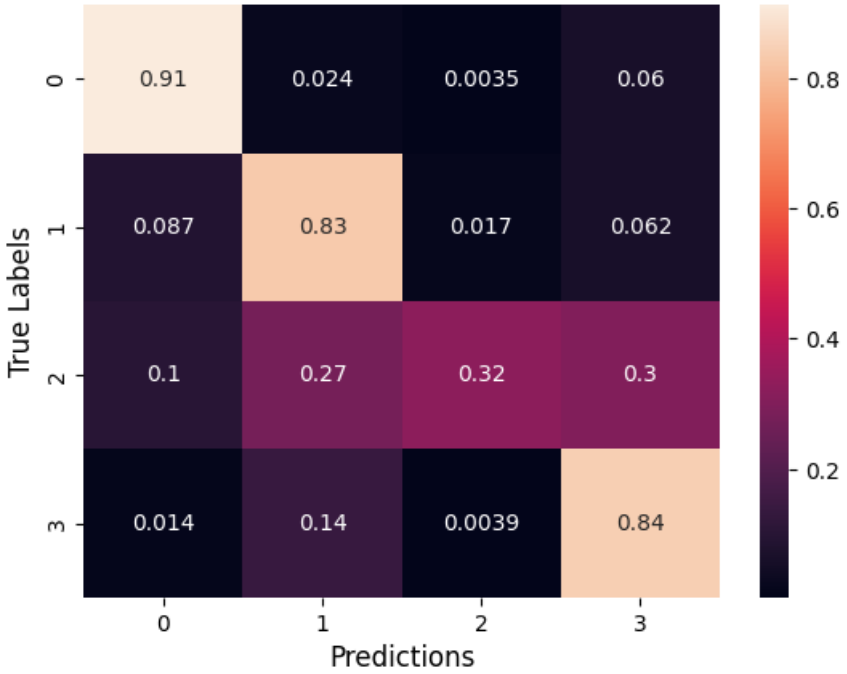

Figure 9. Model Predictions with 10,000 timestep prediction offset

Along with the general accuracies of the model at predicting the system's state, the accuracy of the model across the time-history was of interest. To visual this, the testing predictions were plotted versus the full time-history of each case from the test data in a similar manner to the labeled training data from before. In Figs. 10-12, the model predictions for the three different time offsets are shown for the disturbance generator frequency of 3.94 Hz. As can be seen, all three models perform accurately over the regions of buffeting and LCO, due to the sustained high and low amplitudes being easily learned by the LSTM. However, the 5,000 timestep model is inaccurate at distinguishing between growth and decay, which corresponds to its confusion matrix results. Once again, this is due to the batch size difference between the 5,000-step model and the other two models as this change cause the model to train on only smaller portions of the growth-decay cycles. The 10,000 timestep and same timestep models perform relatively well in growth by comparison, which agrees with their confusion matrices. All three models also identify an LCO region immediately following the peaks of the growth region, which corresponds to the amplitude at which self-sustaining LCO is usually triggered.

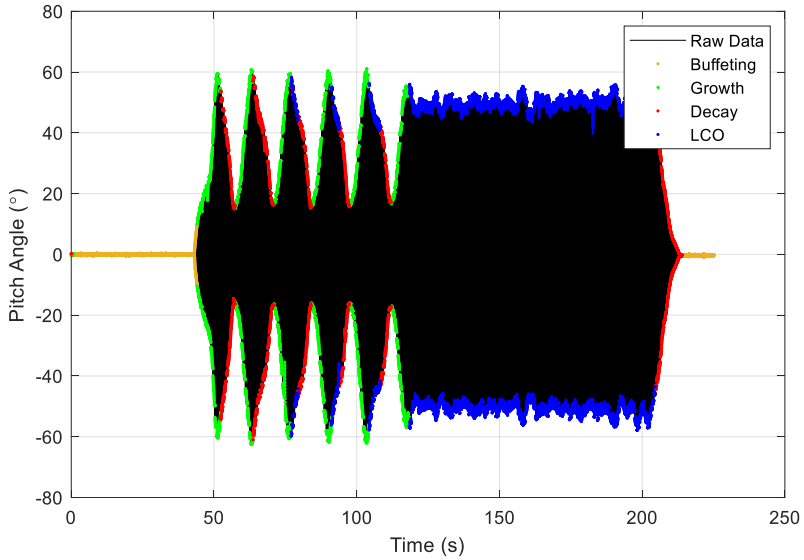

Figure 10. Model Predictions with no timestep offset

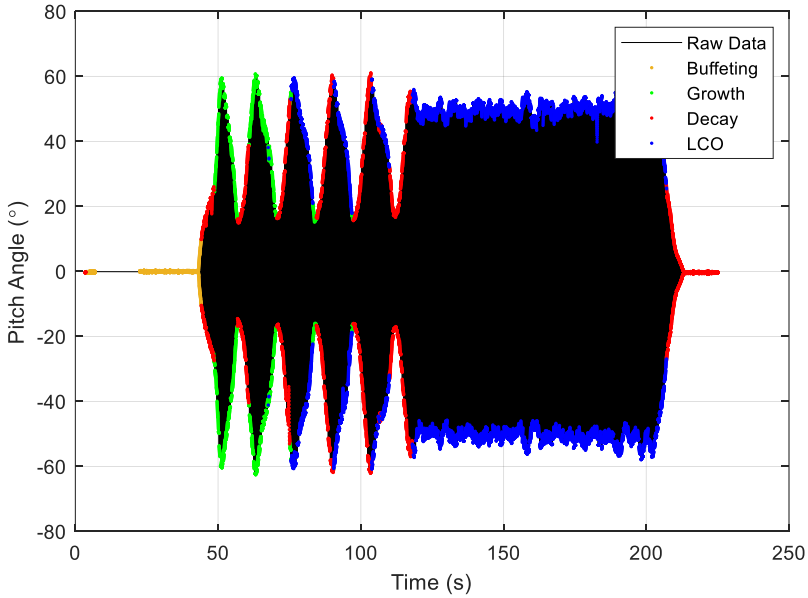

Figure 11. Model Predictions with 5,000 timestep offset

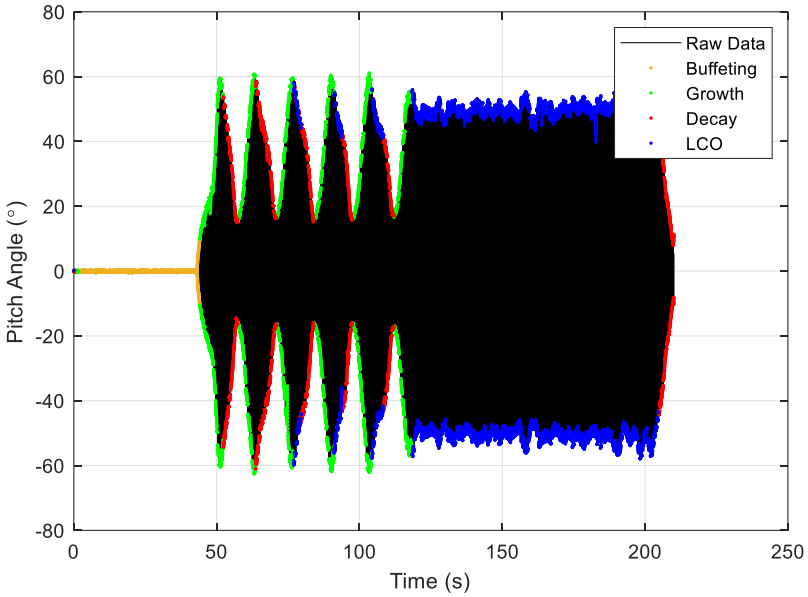

Fig 12. Model Predictions with 10,000 timestep offset

V. Conclusion

In conclusion, the models were able to correctly label the occurrence of limit cycle oscillation very accurately, with accuracies between 84% and 92%. This seems to suggest that the interference behavior exhibited by the disturbance generator onto the wing can be predicted with the use of machine learning. The hyperparameters, specifically batch sizing, play a large role in the accuracy of the model in certain regions, especially the growth region. As such, it seems that batch sizing can be tuned to allow for the model better learn trends over longer portions of the time series, while also allowing for different prediction distances. Future work would consist of deploying a variable batching window, which could call random start points of the batch, and then produce random batches with label distributions similar to the full training set. In testing the model's ability to learn at several timesteps into the future, it was found that increasing the time offset lowers the accuracy of the system. This decrease in accuracy is especially apparent in the decay region, as the further away the model predicts, the more likely it seems to mislabel decay as LCO. In general, all three models showed lower accuracies for their predictions of the growth and decay states than their accuracies for their predictions of the LCO state. As the primary goal of this study is the accurate prediction of LCO behavior in the wing, the models' performances are better understood by comparing their accuracies on the LCO labels specifically. In particular, the accuracy with which the model identifies the first several seconds of the limit cycle oscillations is of chief interest. By improving the ability to predict early LCO behavior accurately, the model could one day be utilized in a real-time control scheme to excite and extinguish LCO in a system.

References

[1] Garrick, I. E., and Reed, W. H., "Historical Development of Aircraft Flutter," *Journal of Aircraft*, Vol. 18, No. 11, pp. 897-912, 1981.

[2] Theodorsen, T., "General Theory of Aerodynamic Instability and the Mechanism of Flutter," Tech. Rep. 496, NACA, 1935.

[3] Goodman, C., Hood, M., Reichenbach, E., and Yurkovich, R., "An Analysis of the F/A-18C/D Limit Cycle Oscillation Solution," 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2003.

[4] Bryant, M., and Garcia E., "Development of an Aeroelastic Vibration Power Harvester," Proc. SPIE 7288, Active and Passive Smart Structures and Integrated Systems, April 2009.

[5] Kirschmeier, B., and Bryant, M., "Experimental Investigation of Wake-Induced Aeroelastic Limit Cycle Oscillations in Tandem Wings," *Journal of Fluids and Structures*, Vol. 81, pp. 309-324, 2018.

[6] Gianikos, Z. N., Kirschmeier, B. A., Gopalarathnam, A., and Bryant, M., "Limit Cycle Characterization of an Aeroleastic Wing in a Bluff Body Wake," *Journal of Fluids and Structures*, Vol. 95, 2020, p. 102986.

[7] Kirschmeier, B. A., Gianikos, Z., Gopalarathnam, A., and Bryant, M., "Amplitude Annihilation in Wake-Influenced Aeroelastic Limit-Cycle Oscillations," *AIAA Journal*, Vol. 58, No. 9, 2020, pp. 4117–4127.

[8] Rockwood, M., and Medina, A., "Controlled Generation of Periodic Vortical Gusts by the Rotational Oscillation of a Circular Cylinder and Attached Plate," *Experiments in Fluids*, Vol. 61, No. 3, 2020.

[9] Schwing, A. G., & Urtasun, R. "Fully connected deep structured networks". arXiv preprint arXiv:1503.02351, 2015.

[10] Alex Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," Physica D: Nonlinear Phenomena, Volume 404, 2020, 132306, ISSN 0167-2789, https://doi.org/10.1016/j.physd.2019.132306.

[11] Chen, Y., Zhang, S., Zhang, W., Peng, J., Cai, Y. "Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting", *Energy Conversion and Management*, Vol. 185, pp. 783-799, 2019.

[12] Mathworks. (n.d.). K-means Clustering. MATLAB Documentation: K-means Clustering. Retrieved December 13, 022, from https://www.mathworks.com/help/stats/kmeans.html

[13] Kingma, Diederik P., Ba, Jimmy. "Adam: A Method for Stochastic Optimization," arXiv preprint, arXiv:1412.6980, 2017.