29th Annual INCOSE international symposium
Orlando, FL, USA
July 20 - 25, 2019

# An Approach for Formal Verification of Machine Learning based Complex Systems

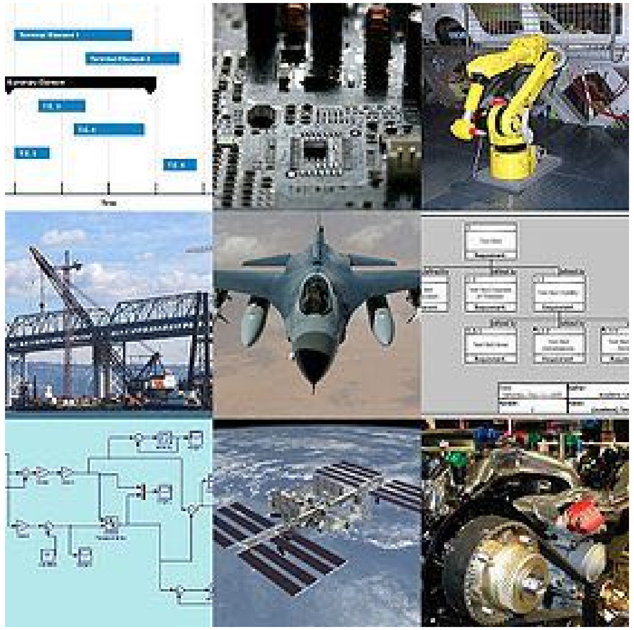
**Ramakrishnan Raman & Yogananda Jeppu**
**Honeywell Technology Solutions**

# Presentation Outline

- Introduction
  - Complex Systems, MOEs/MOPs, Machine Learning, Formal Methods
- Proposed Approach
  - ML Classifier + Formal Verification
  - Aircraft Pitch Control System – Case Study
- Conclusions & Future Work

# Complex Systems

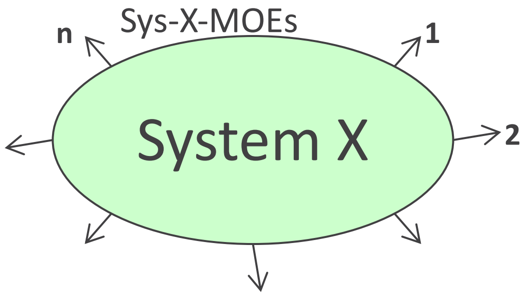
- Multiplex of relationships/ forces/ interactions between subsystems & constituent systems
- Difficulties in establishing cause-and-effect chain
- Very difficult to anticipate the behavior from the knowledge of the constituents
- Characteristics: Emergence, hierarchical organization, numerosity….



- The perspective of complexity used in this paper is with respect to the degree of difficulty in accurately *predicting* the *future behavior*
- This complexity is determined by the system being observed, the capabilities of the observer, and the behavior that the observer is attempting to predict

# MOEs: Measures of Effectiveness

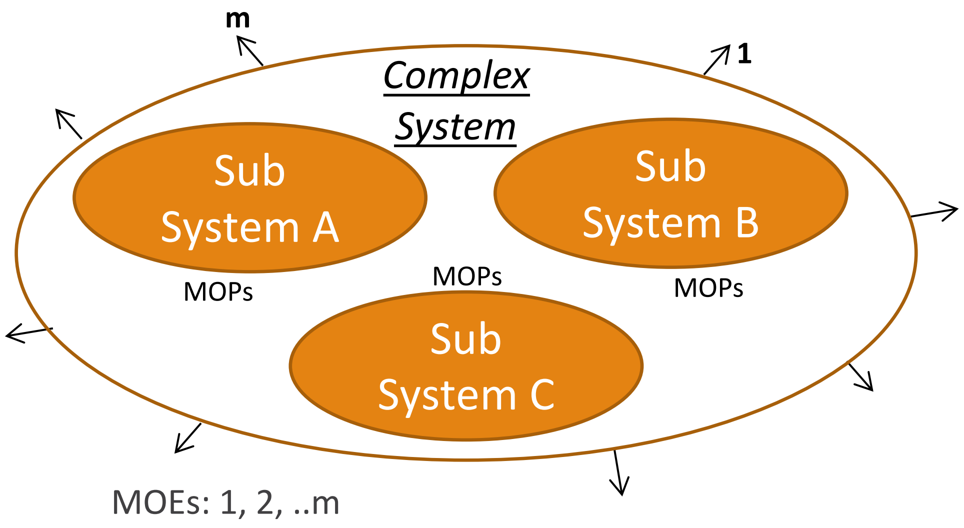
- MOEs: Operational measures of success that are closely related to the achievement of the objective of the system of interest [INCOSE Systems Engineering Handbook v4]
- MOEs represent the overall operational success criteria, and they manifest at the boundary of the system
- MOEs are independent of the particular solution




Examples of MOEs:
- Response time to a user action
- Time to Alert
- Availability of the system

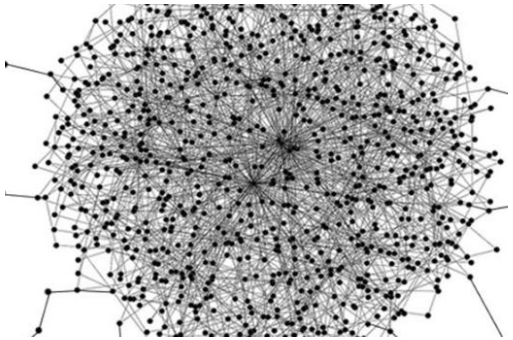
# MOPs: Measures of Performance

- MOPs: measures that characterize physical or functional attributes relating to the system operation, measured or estimated under specified test and/or operational environmental conditions [INCOSE Systems Engineering Handbook v4]
- MOPs define the key performance characteristics the system should have when fielded and operated in its intended operating environment, to achieve the desired MOEs of the system
- MOPs are dependent of the particular solution



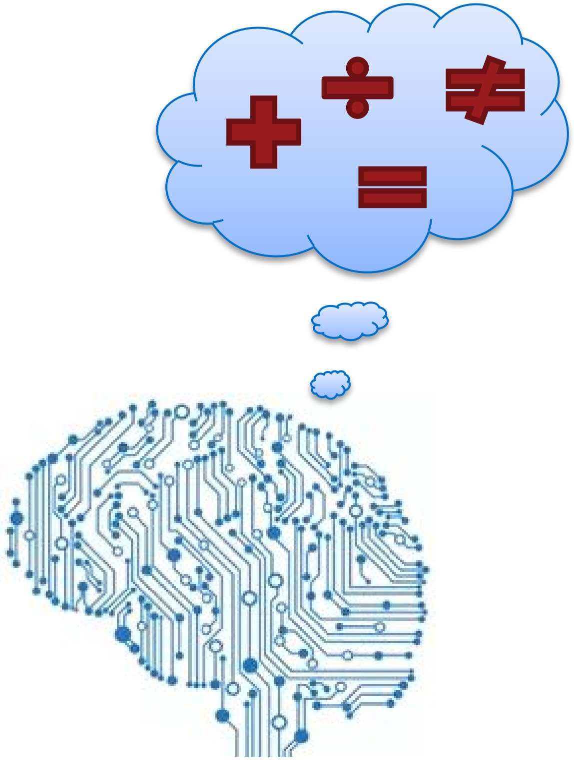
# Emergent Behavior

- Emergence refers to the ability of a system to produce a highly-structured collective behavior over time, from the interaction of individual subsystems
  - Examples: flock of birds flying in a V-formation, and ants forming societies of different classes of individual ants
- For a system, emergent behavior refers to all that arises from the set of interactions among its subsystems and components.
- Complex systems are expressed by the emergence of global properties
  - It is difficult, if not impossible, to anticipate emergence just from a complete knowledge of component or subsystem behaviors



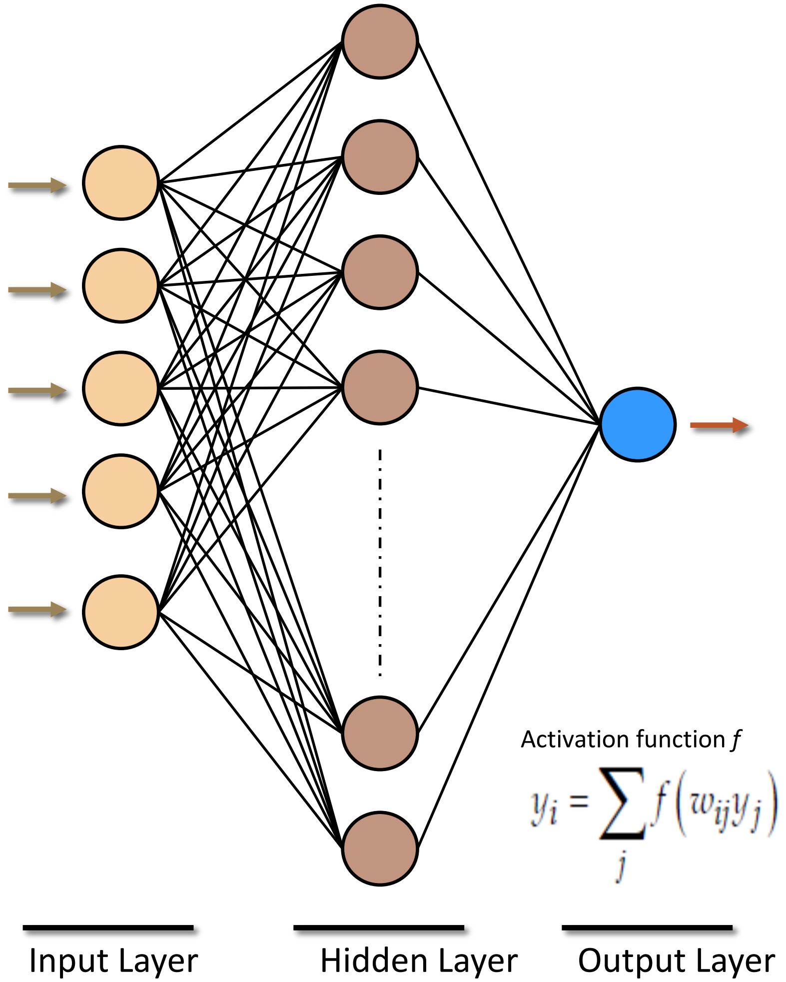
# Machine Learning

- Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions
- Machine Learning represents the field of study that allows computer programs to learn without being explicitly programmed



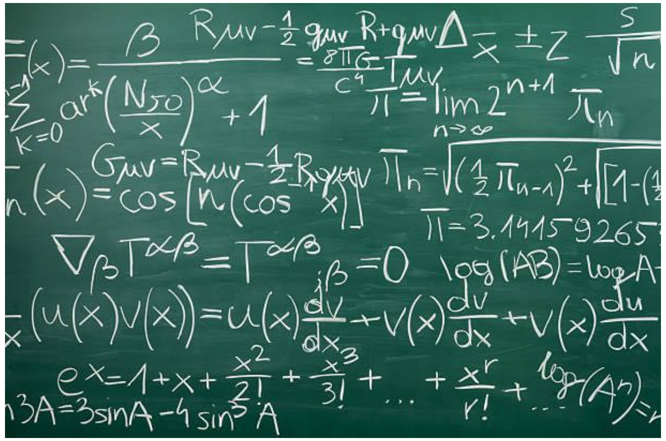
# Neural Networks

- Artificial Neural Networks (NN), comprise a collection (organized in layers) of interconnected units (nodes), with each node having the capability to receive a signal, process the signal, and transmit the processed signal to other units linked to it
- Recently, there is an explosion in the adoption of neural network based machine learning techniques and models in various systems, and are increasingly being used to control many physical systems, such as cars and drones.



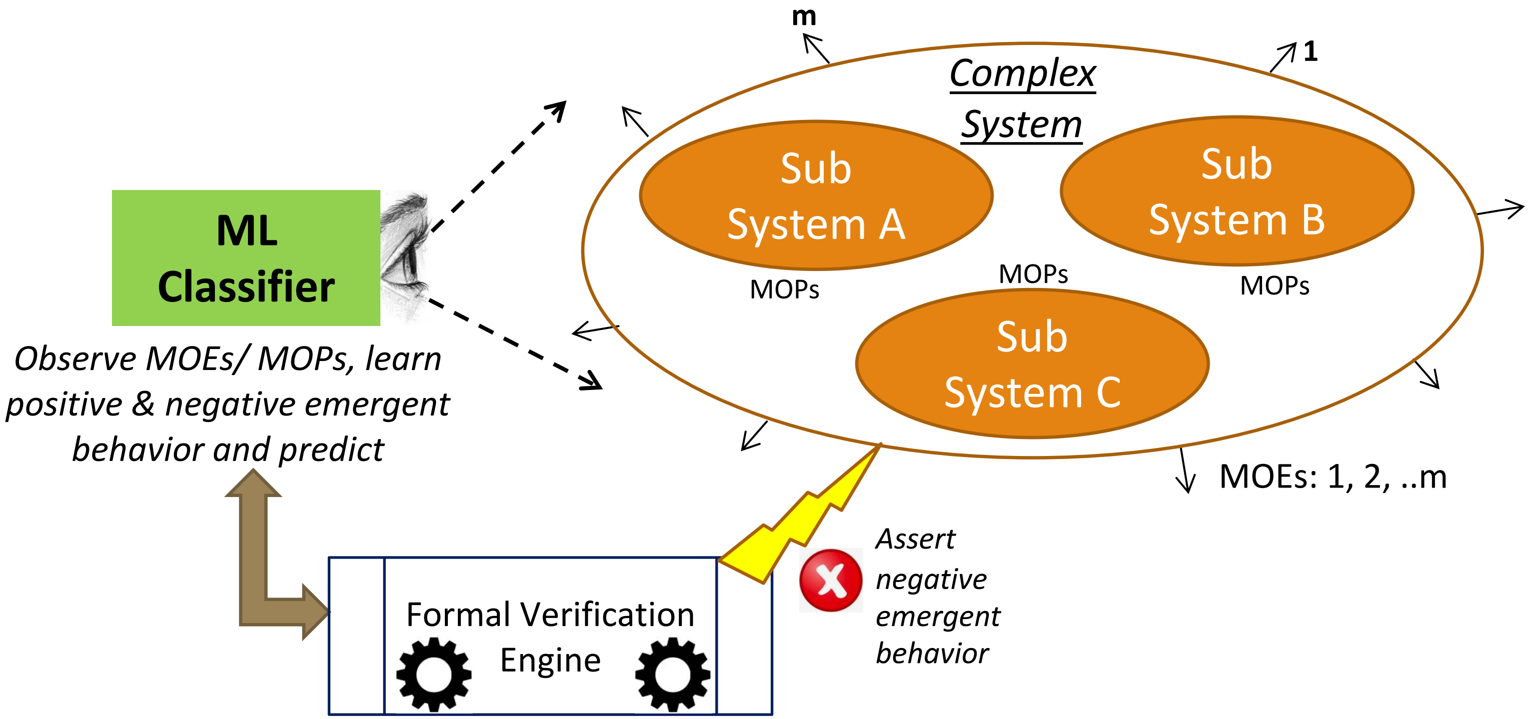
# Formal Methods: Model Checking

- Mathematics based techniques for the specification, development, and verification of digital systems
  - Formal methods can be used to model complex systems as mathematical entities
  - The complex system behavior is broken down into smaller units and each one of these is defined as mathematical equations



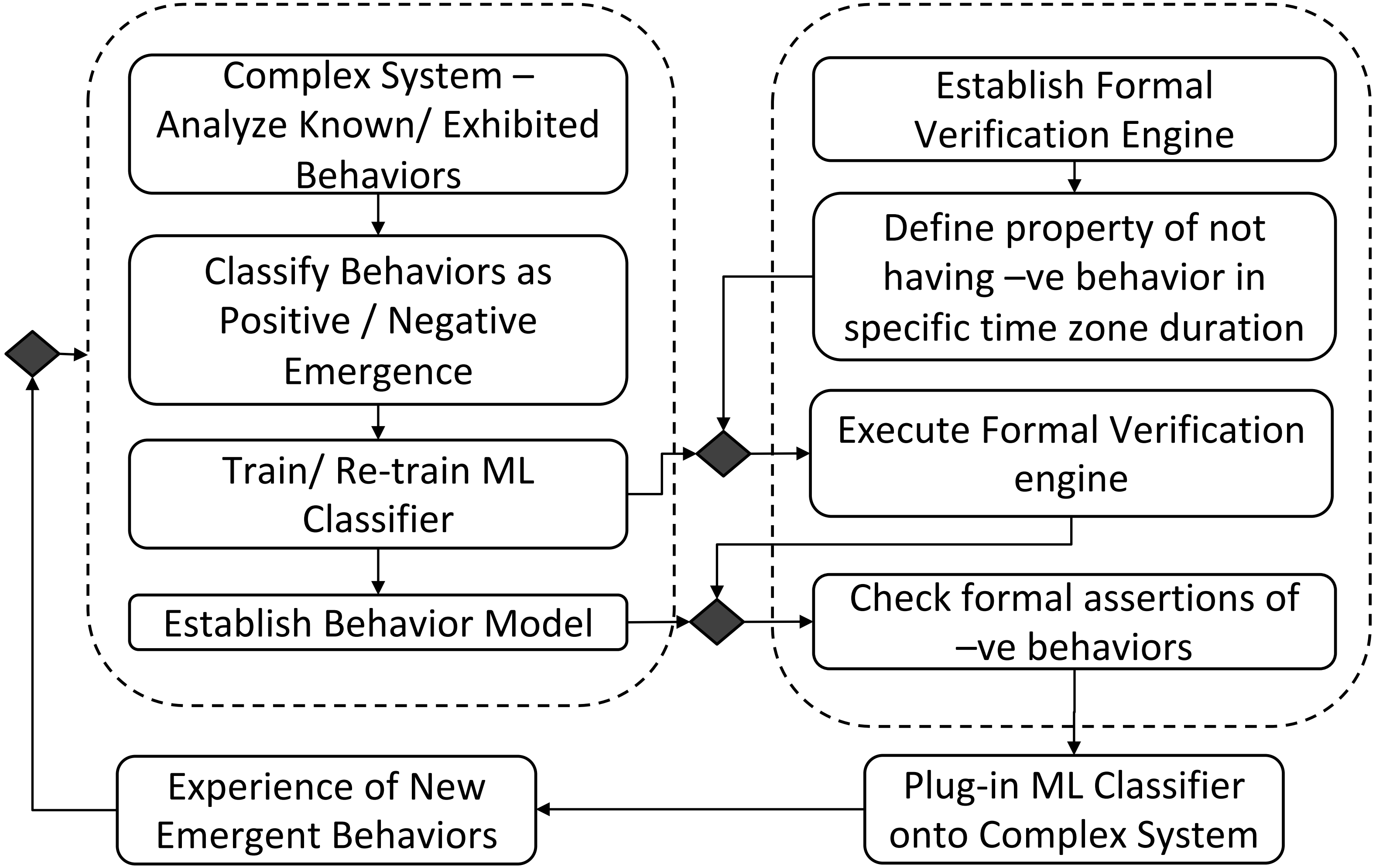
- Model Checking: A model of the system and a way to define the property of the system
  - The model checking tool then explores the possible states the model can be in, and checks for violations of the property

# Proposed Approach Overview

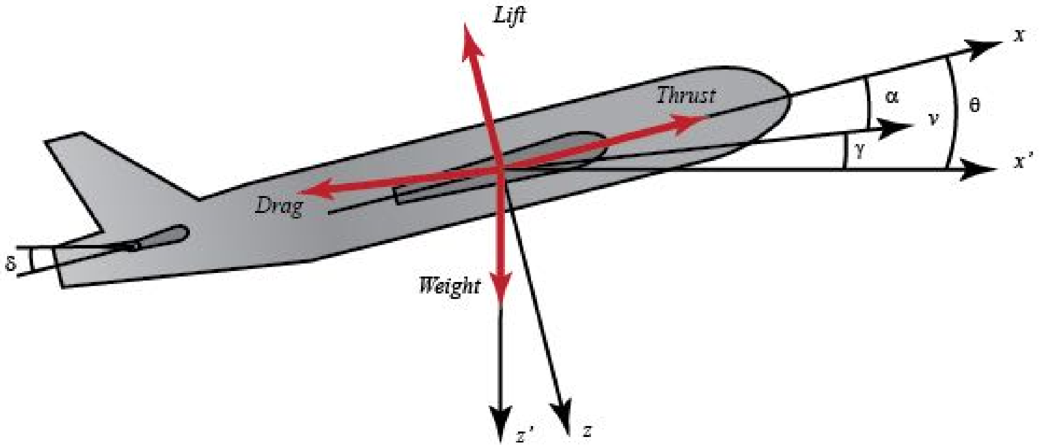



- The system comprises one or more subsystems, and corresponding performance characteristics manifesting as MOPs
- The proposed approach involves building a Machine Learning (ML) Classifier that observes the various MOPs and MOEs, and learns the emergent behavior.
- The classifier is then used by the Formal Verification Engine to assert the occurrence of negative emergent behavior.

# Overview of Proposed Approach



# Aircraft Pitch Control System Case Study

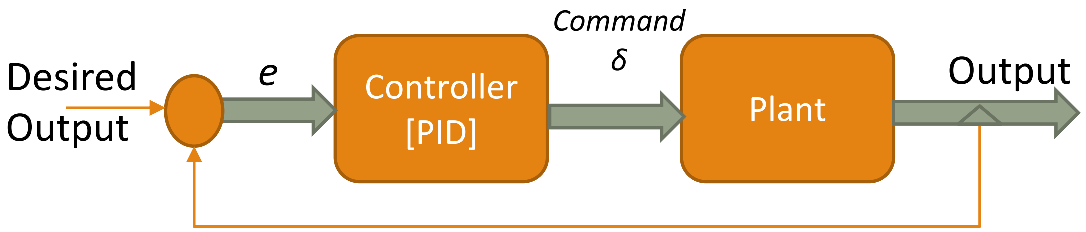



$$\dot{\alpha} = \mu\Omega\sigma\left[-(C_L + C_D)\alpha + \frac{1}{(\mu - C_L)}q - (C_W \sin\gamma)\theta + C_L\right]$$

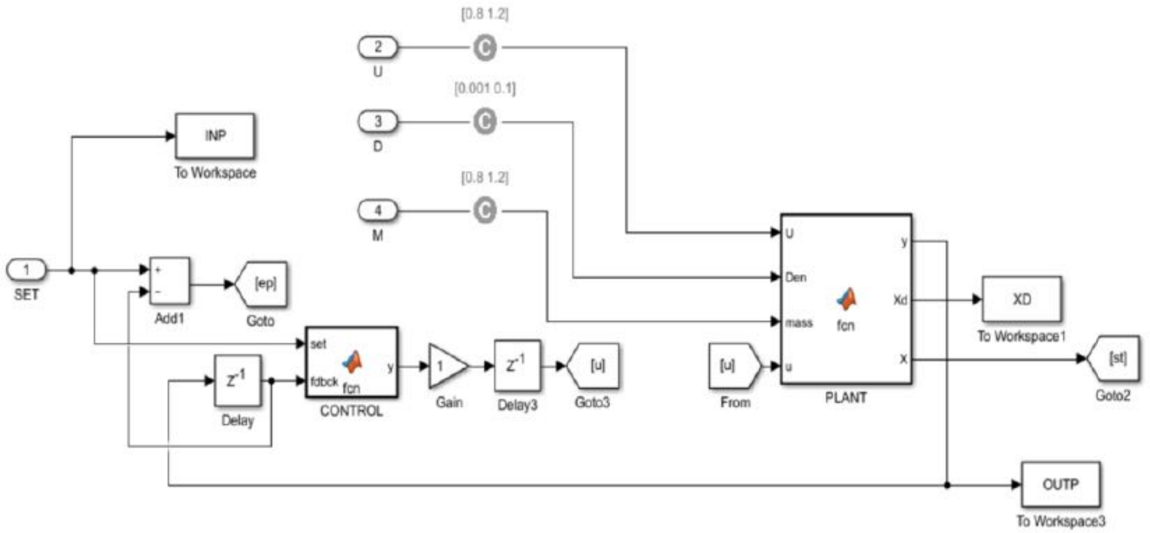
$$\dot{q} = \frac{\mu\Omega}{2i_{yy}}\left[\left[C_M - \eta(C_L + C_D)\right]\alpha + \left[C_M + \sigma C_M(1 - \mu C_L)\right]q + (\eta C_W \sin\gamma)\delta\right]$$

$$\dot{\theta} = \Omega q$$

- Complex System: Aircraft pitch controller
- MOEs for this system pertain to the comfort level of the flight
  - Comfortable pitch
  - Marginal Discomfort: marginal issues, the flight will be felt like a roller coaster with lower amplitude, finally settling down to a stable flight path
  - Significant Discomfort: will feel like a roller coaster ride, not divergent but an oscillatory unsettled behavior



# Behavior Analysis Experiments

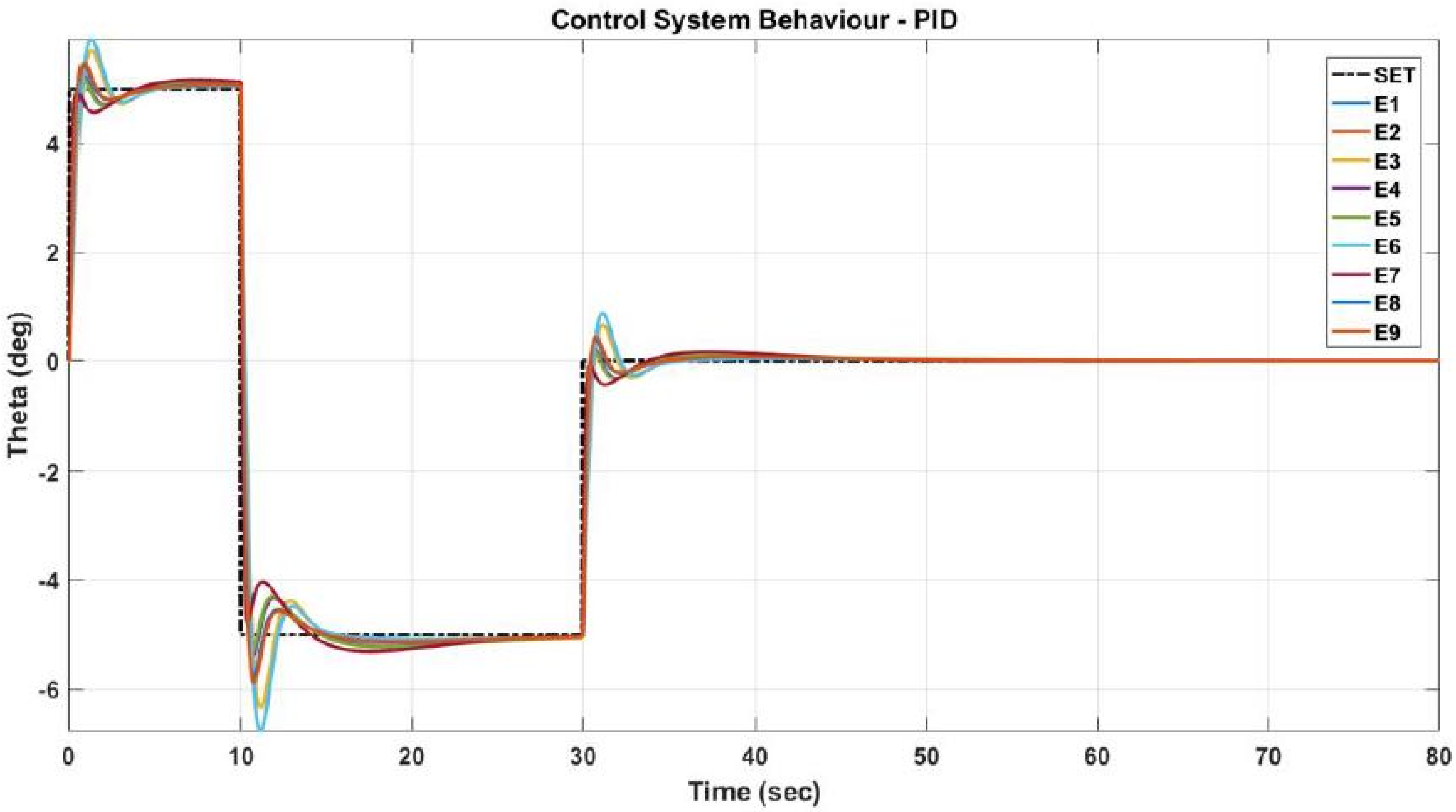



Model of the Control System

| Exp | Speed | Mass | Density |
|---|---|---|---|
| E1 | 0.8 | 1.2 | 1.2 |
| E2 | 0.8 | 1.0 | 1.0 |
| E3 | 0.8 | 0.8 | 0.8 |
| E4 | 1.0 | 1.2 | 1.0 |
| E5 | 1.0 | 1.0 | 0.8 |
| E6 | 1.0 | 0.8 | 1.2 |
| E7 | 1.2 | 1.2 | 0.8 |
| E8 | 1.2 | 1.0 | 1.2 |
| E9 | 1.2 | 0.8 | 1.0 |

Orthogonal Array of Experiments

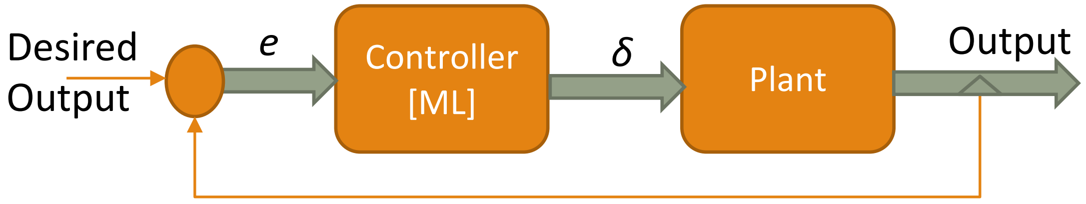
# System Behavior

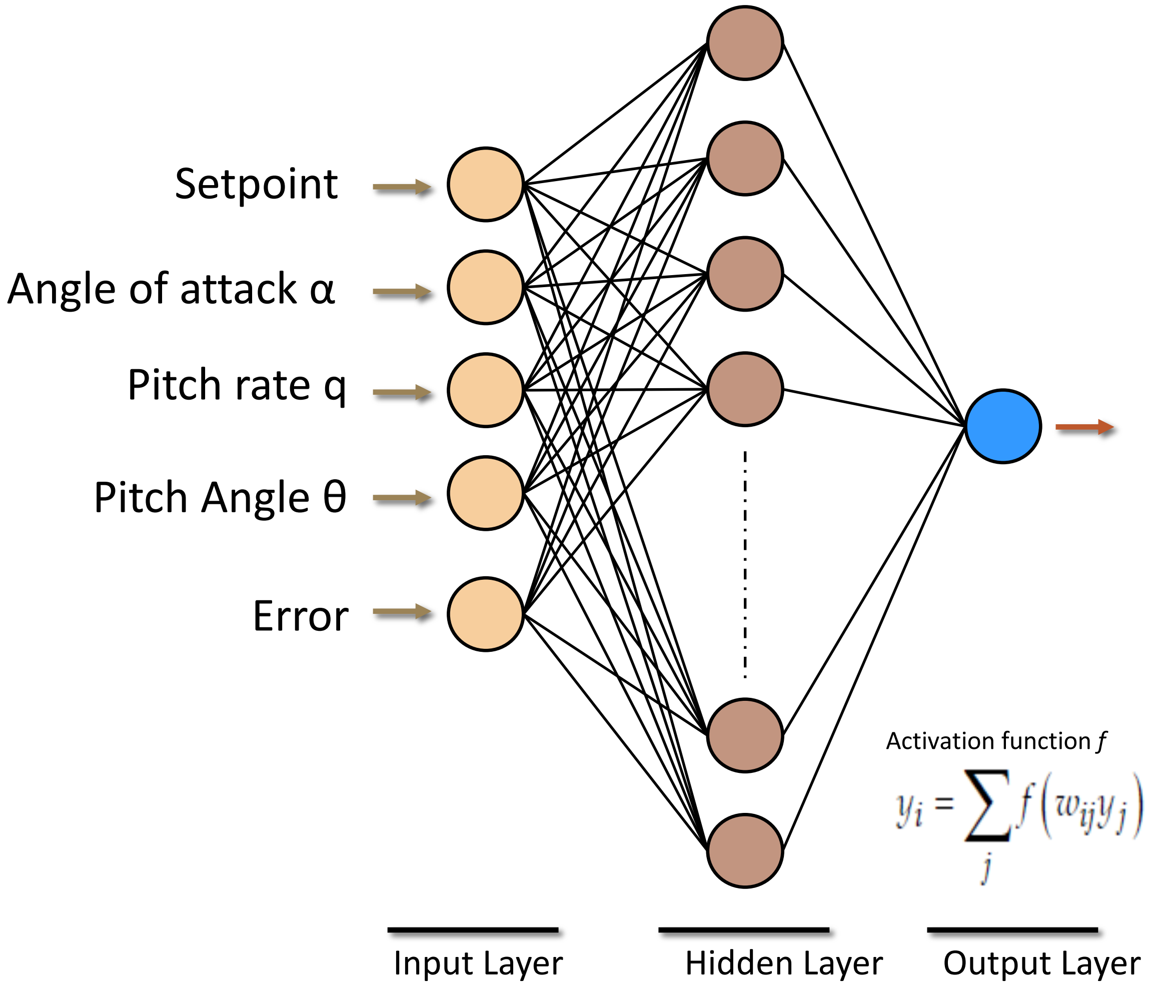



- Figure depicts stable behavior for variation in the response to a doublet set point of 5 degrees.
- This stable behavior is classified as a positive emergent behavior

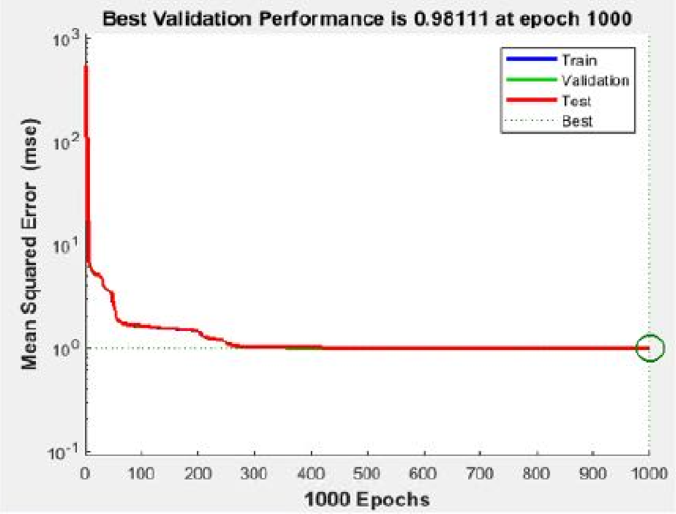
# Controller [Machine Learning]

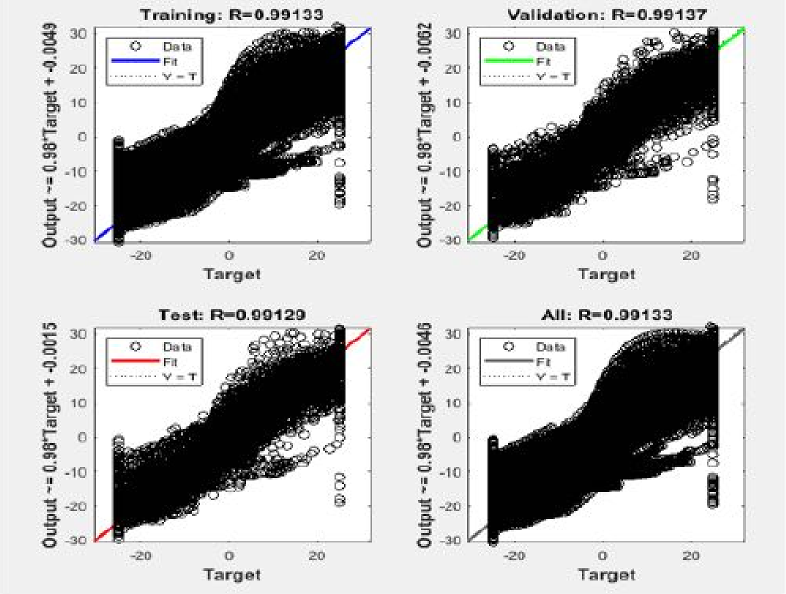
- The training data set for the ML model to learn the controller is built from simulation runs of the PID controller
- The data set (of about 3.7 million records) is split with 75% of samples used as training set, and 15% each for validation and test sets

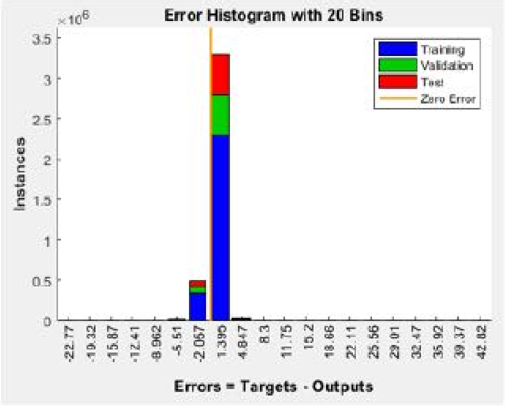


# ML Controller – Training



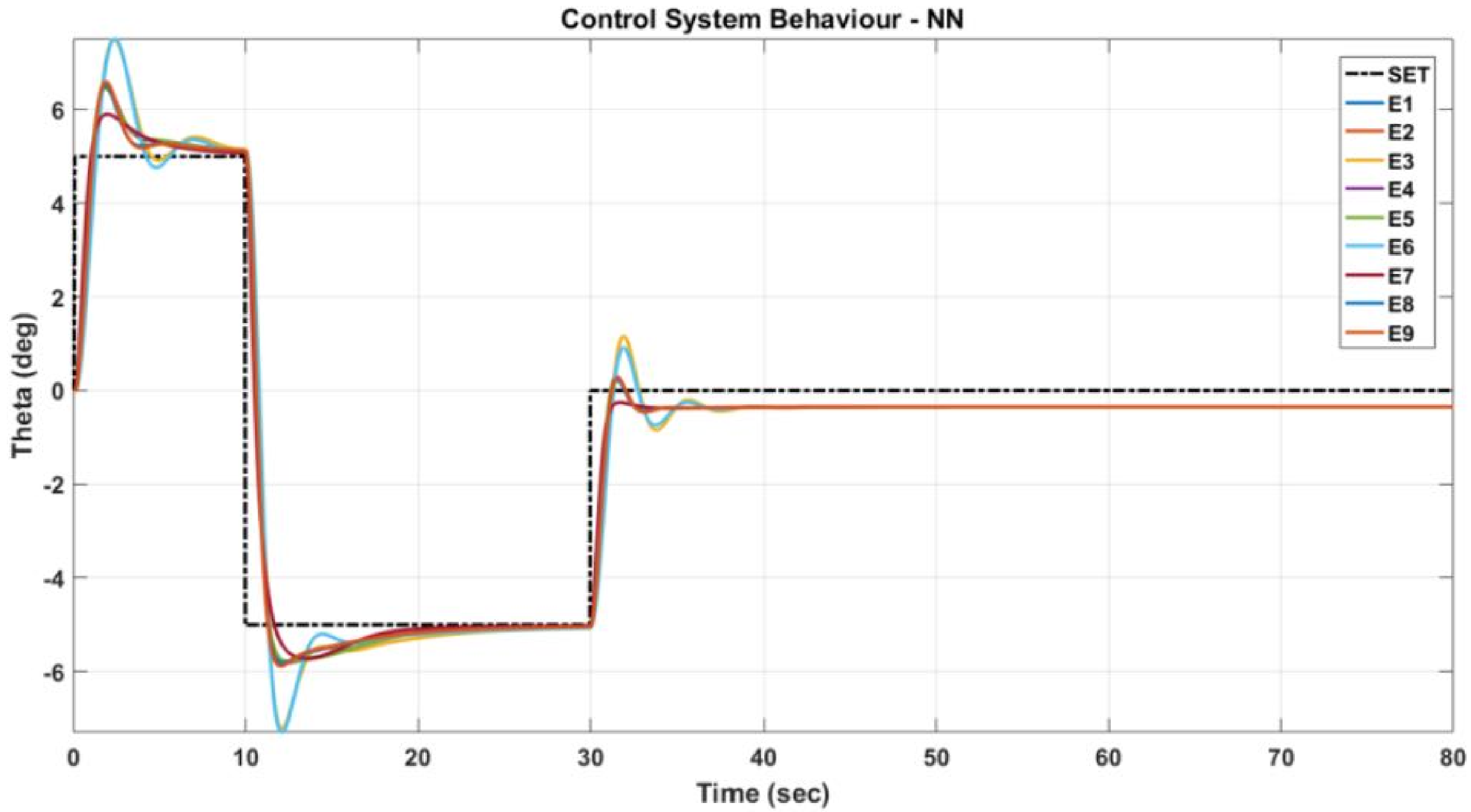




**Results**

| | Samples | MSE | R |
|---|---|---|---|
| Training: | 2695269 | 9.81089e-1 | 9.91325e-1 |
| Validation: | 577558 | 9.81105e-1 | 9.91372e-1 |
| Testing: | 577558 | 9.87390e-1 | 9.91285e-1 |

- ML model training
  - Mean Square Error (MSE): average squared difference between the outputs and targets (i.e. lower values of MSE are better)
  - R values: measures the correlation between outputs and targets (i.e. R value closer to 1 is better)
- Learning is stopped at epoch 1000, to serve the purpose of dealing with a system that can exhibit negative emergent behavior at times.

# System Behavior [ML Controller]

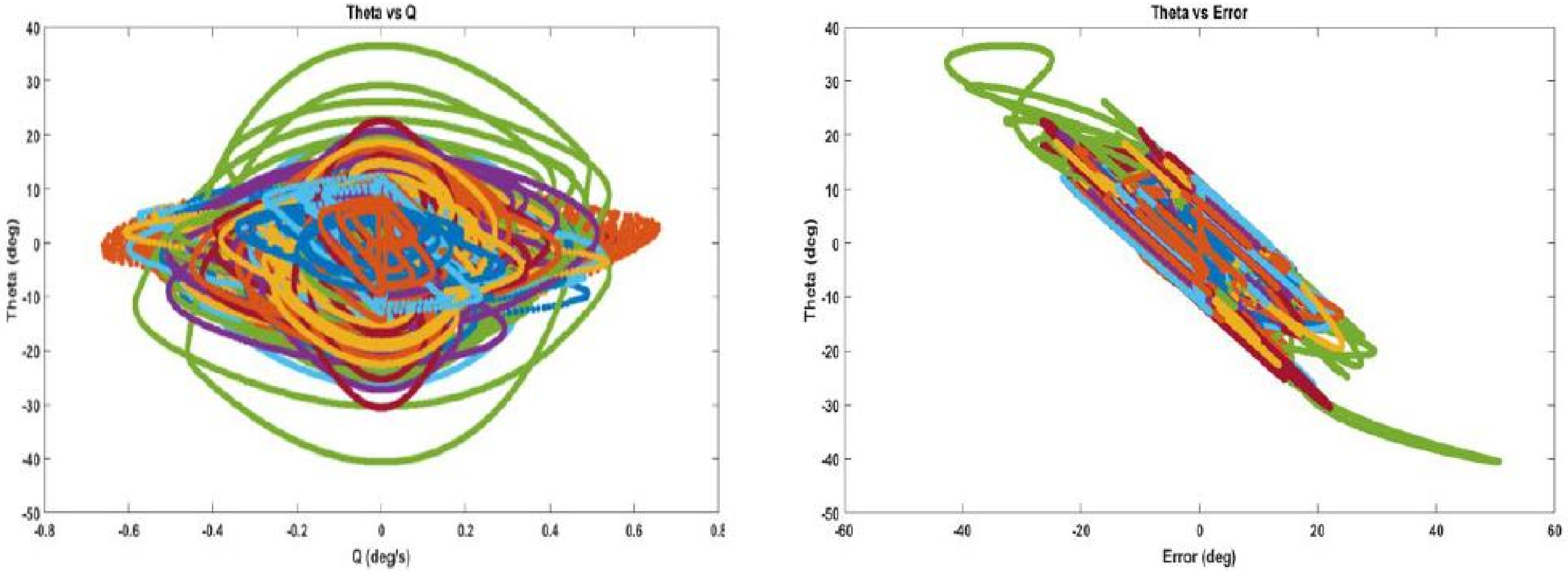



- Behavior is not as stable as traditional PID controller [intentional]
  - Response time and offset to target

# Behavior Characterization

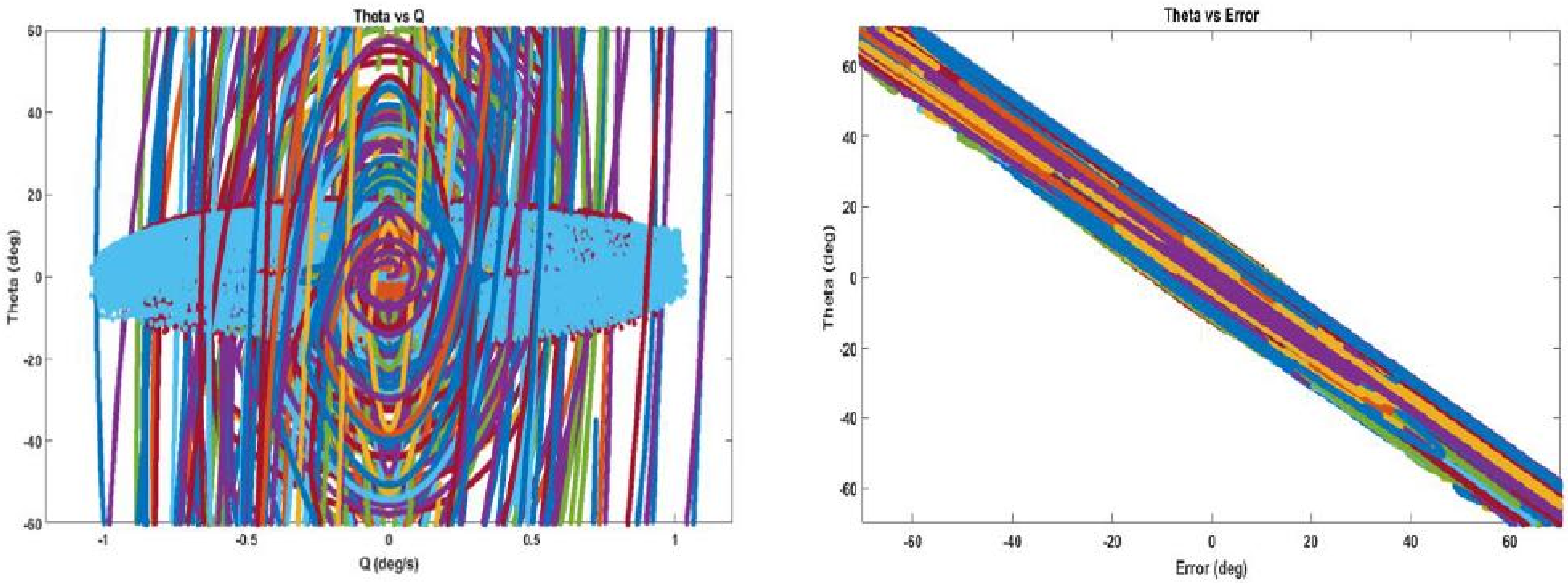
- Behavior of the system: Based on the states and the error between the set point and the achieved
- States are plotted to come up with a zone of behavior that signifies a normal stable behavior of the system (~1000 random test cases)
- The closed system is made unstable by changing the factors on speed, density and mass beyond the 20% percent bound
  - This is expected as the design is done for a linear zone and in flight controls a lookup table is used to provide gains as a factor of speed and altitude to ensure a stable performance across the flight envelope

# Behavior Characterization: +ve Emergence




Each color indicates a different test case. The angle theta and pitch rate q plots shows the spiraling movement as the system settles down to the desired setpoint. The theta and error shows a well correlated behavior as the initial theta for a setpoint of 0 is the initial error. As theta reduces, the error also reduces

# Behavior Characterization: –ve Emergence




Each color indicates a different test case. The angle theta and pitch rate q plots is distinctly different from the earlier case of positive emergence

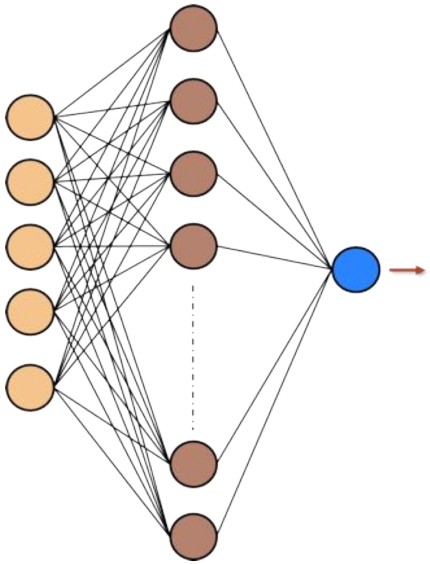
# Behavior Stability Classifier

## ZONE BASED CLASSIFIER

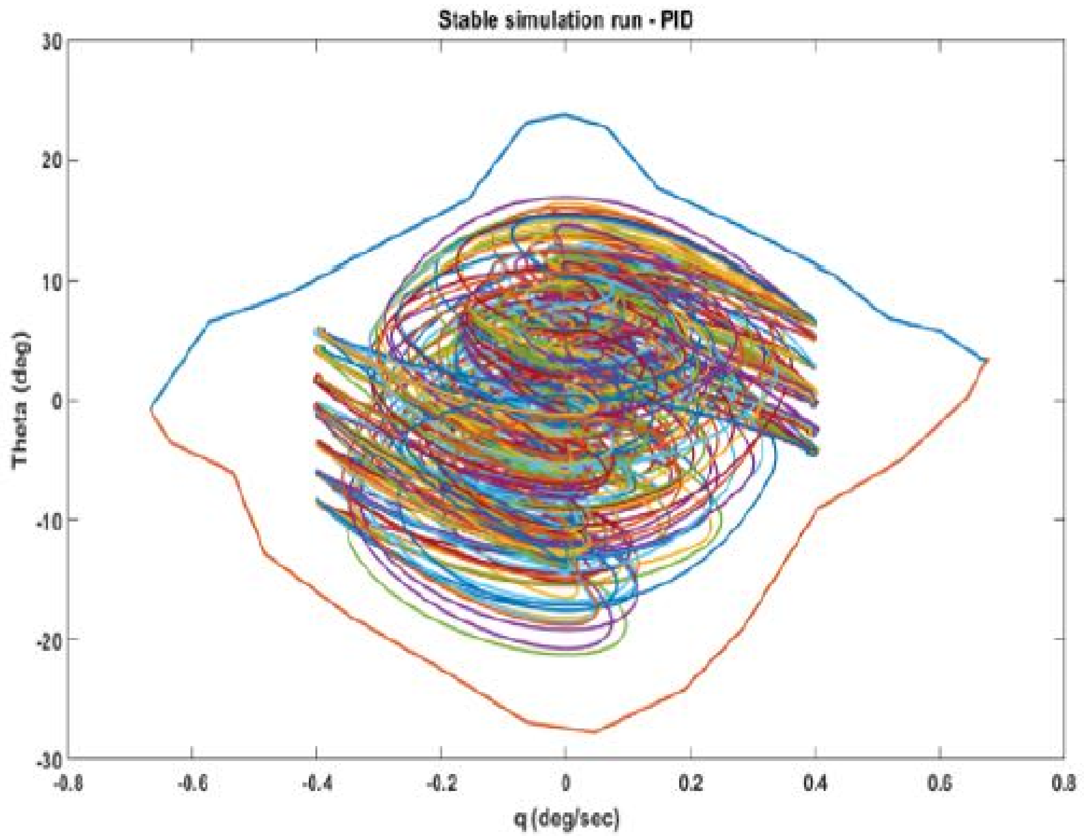
| | Coordinates | | |
|---|---|---|---|
| Error Top | 14.4051 | 26.8815 | -13.3832 |
| Theta Top | -26.7362 | -16.0381 | 24.5468 |
| Error Bottom | 14.4051 | -30.2830 | -13.3832 |
| Theta Bottom | -26.7362 | 18.7733 | 24.5468 |

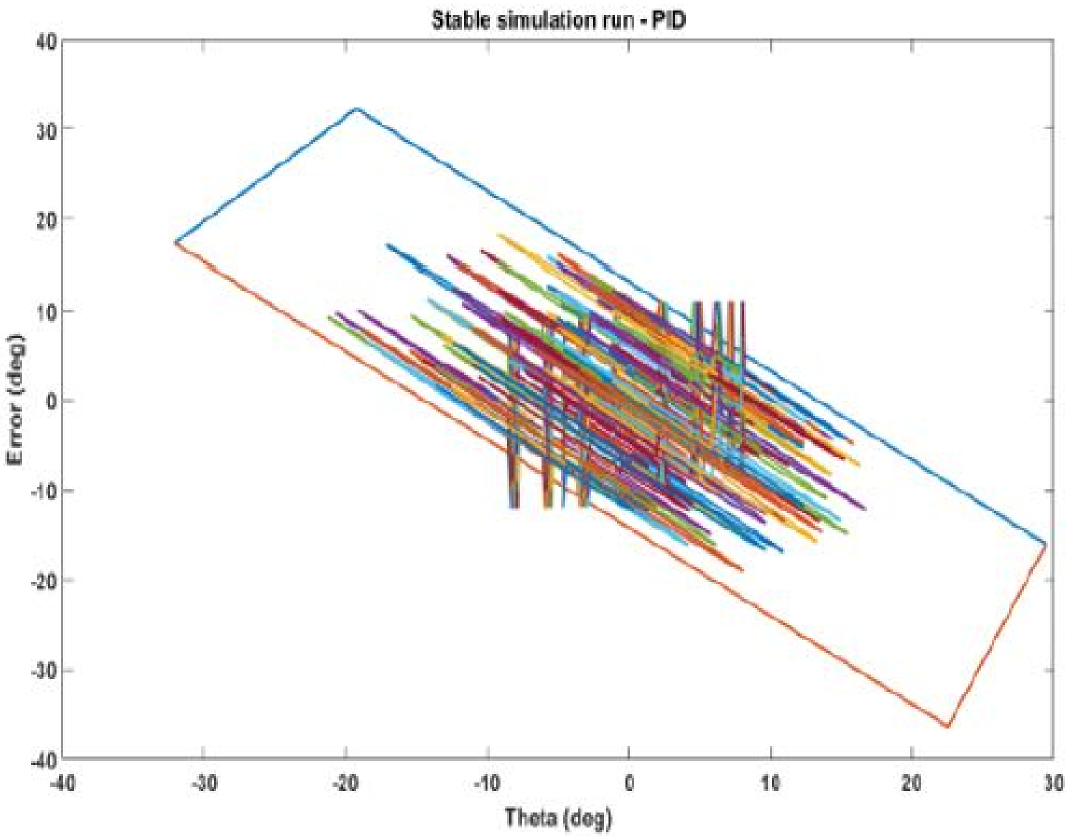
| | Coordinates | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Theta Top | -0.755 | 6.547 | 13.170 | 16.566 | 23.019 | 22.679 | 17.585 | 10.453 | 6.887 | 3.490 | 3.321 |
| q Top | -0.668 | -0.573 | -0.287 | -0.159 | -0.065 | 0.064 | 0.147 | 0.430 | 0.516 | 0.665 | 0.678 |
| Theta Bottom | -0.755 | -3.472 | -12.812 | -18.076 | -22.830 | -27.755 | -24.189 | -9.076 | -5.000 | 3.490 | 3.321 |
| q Bottom | -0.668 | -0.637 | -0.484 | -0.346 | -0.189 | 0.045 | 0.189 | 0.404 | 0.537 | 0.676 | 0.678 |

## ML BASED CLASSIFIER



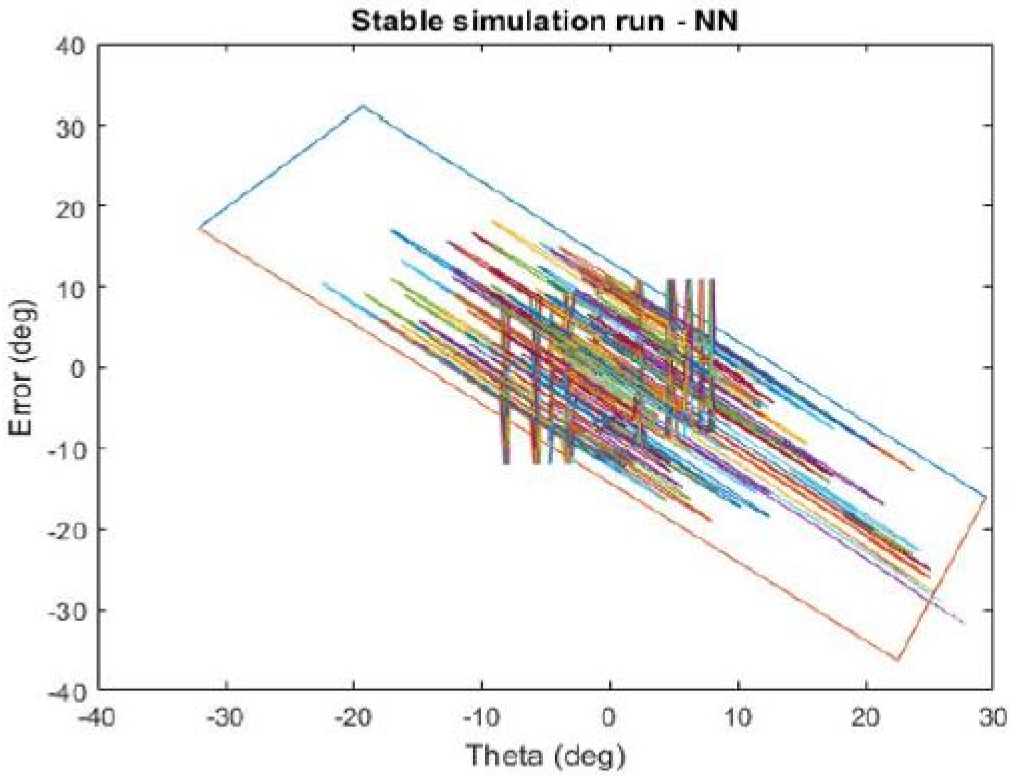
# Zone Based Classifier – PID Controller

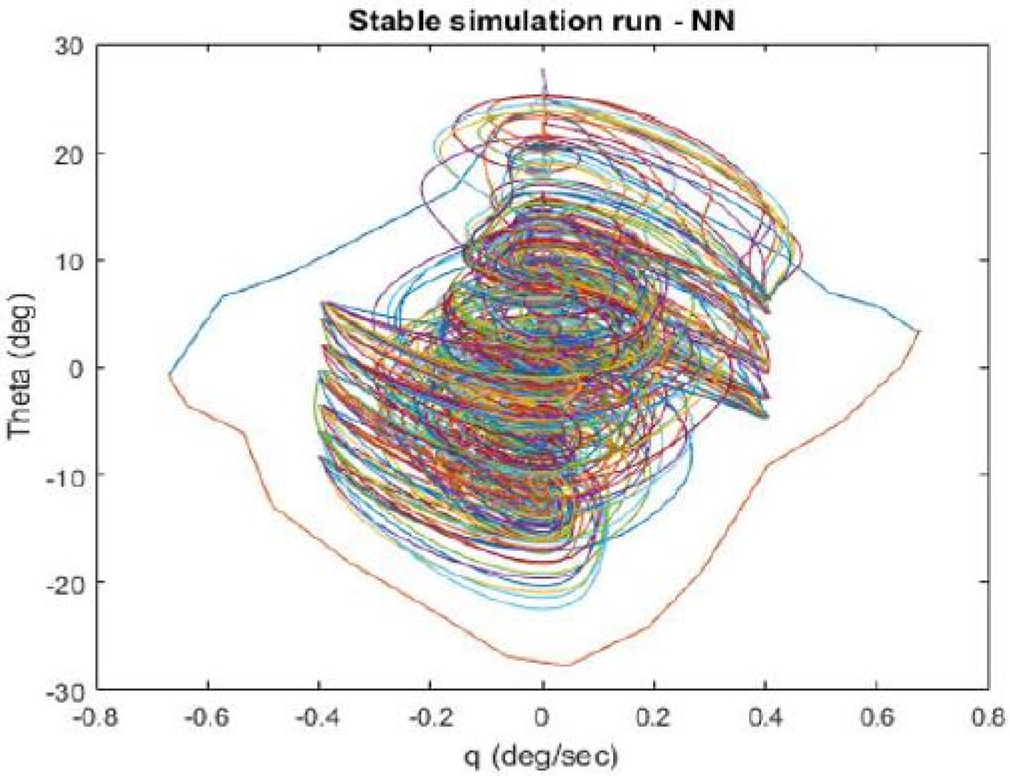



- The PID control is simulated with an initial value of state selected from the stable zone. A 20 second simulation is done for each test case.
- The test cases are defined using an orthogonal array. The behavior exhibited is well within the stable zone

# Zone Based Classifier – ML Controller

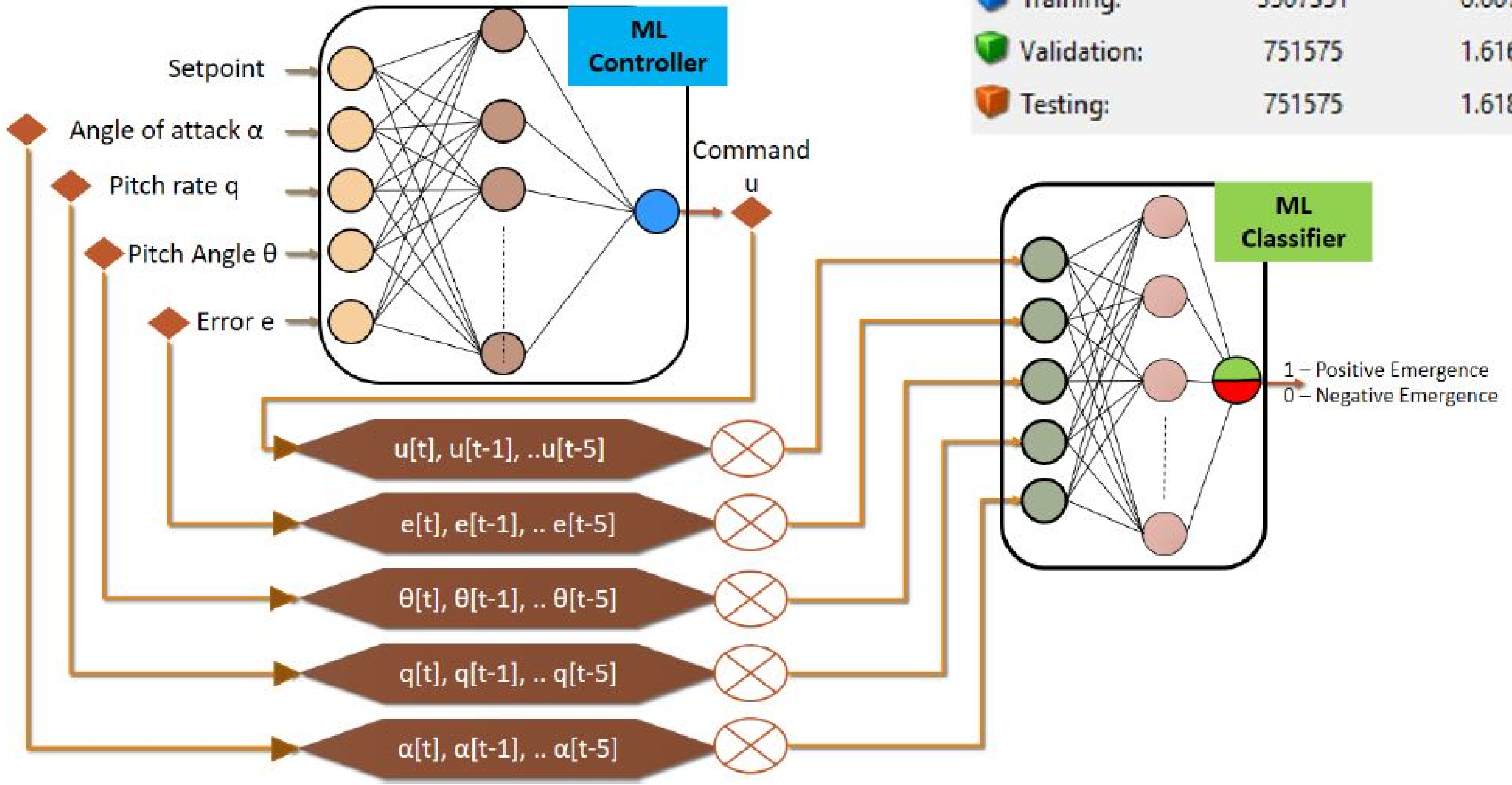



- As expected, the system based on ML Controller has some scenarios of exhibiting negative behaviors. This is seen by the excursions outside the bounds.
- The behavior is not unstable and divergent but perhaps more oscillatory. The behavior gets near the zone of negative emergence, with potential to impact MOEs if it prolongs the trend.

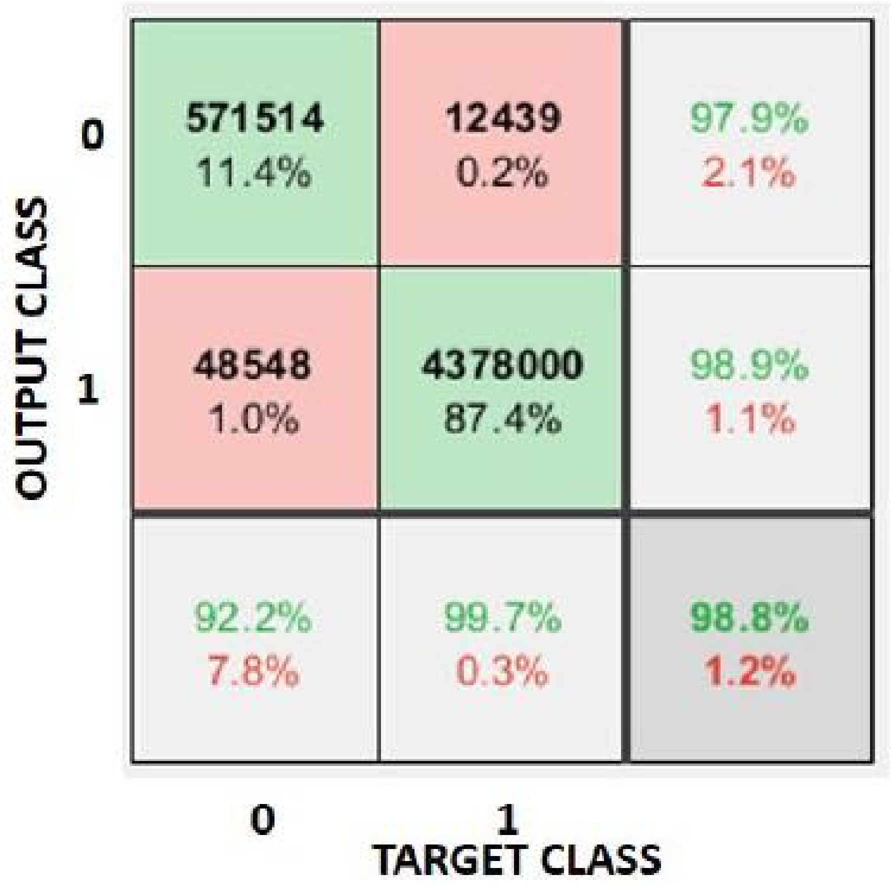
# ML Classifier

| | Samples | CE | %E |
|---|---|---|---|
| Training: | 3507351 | 6.00732e-1 | 1.21208e-0 |
| Validation: | 751575 | 1.61671e-0 | 1.24245e-0 |
| Testing: | 751575 | 1.61881e-0 | 1.21571e-0 |



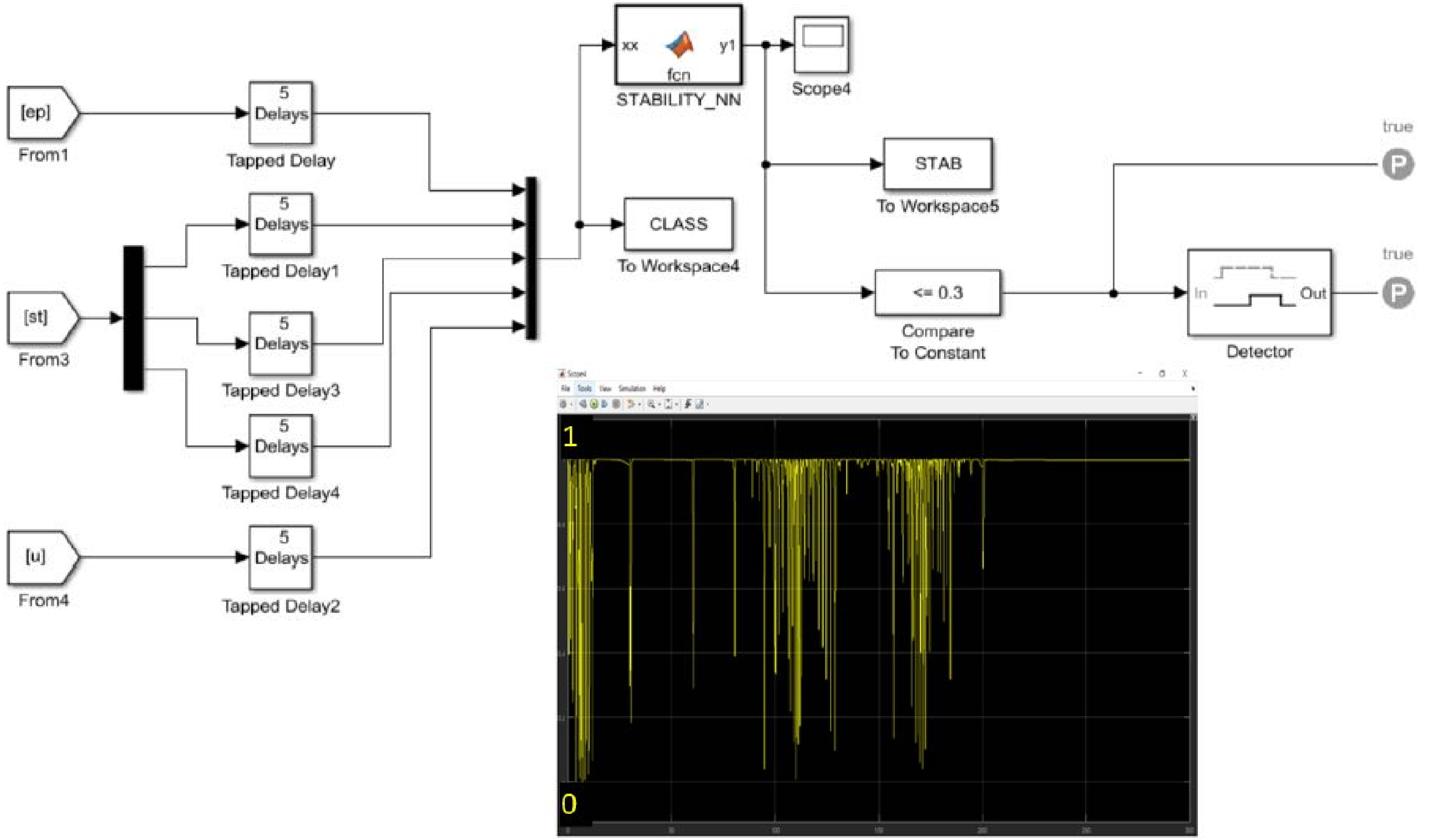

The ML Classifier is built by tapping in the same inputs used for the ML Controller, and feeding a set of six values, corresponding to current time t, and previous five instance (data set ~5 million records)

# ML Classifier - Performance



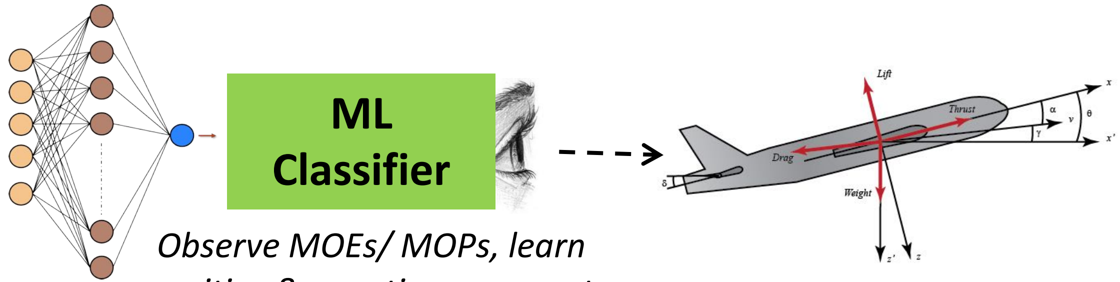

- The  column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified.
    - These metrics are the precision (or positive predictive value) and false discovery rate, respectively.
- The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified.
    - These metrics are often called the recall (or true positive rate) and false negative rate, respectively.
- The cell in the bottom right of the plot shows the overall accuracy

# ML Classifier Plug-in to Controller




- The ML Classifier is plugged on to the control model
- Monitoring is through scope monitor
- Values closer to 1 indicates positive emergent behavior
- Values closer to 0 indicates negative emergent behavior

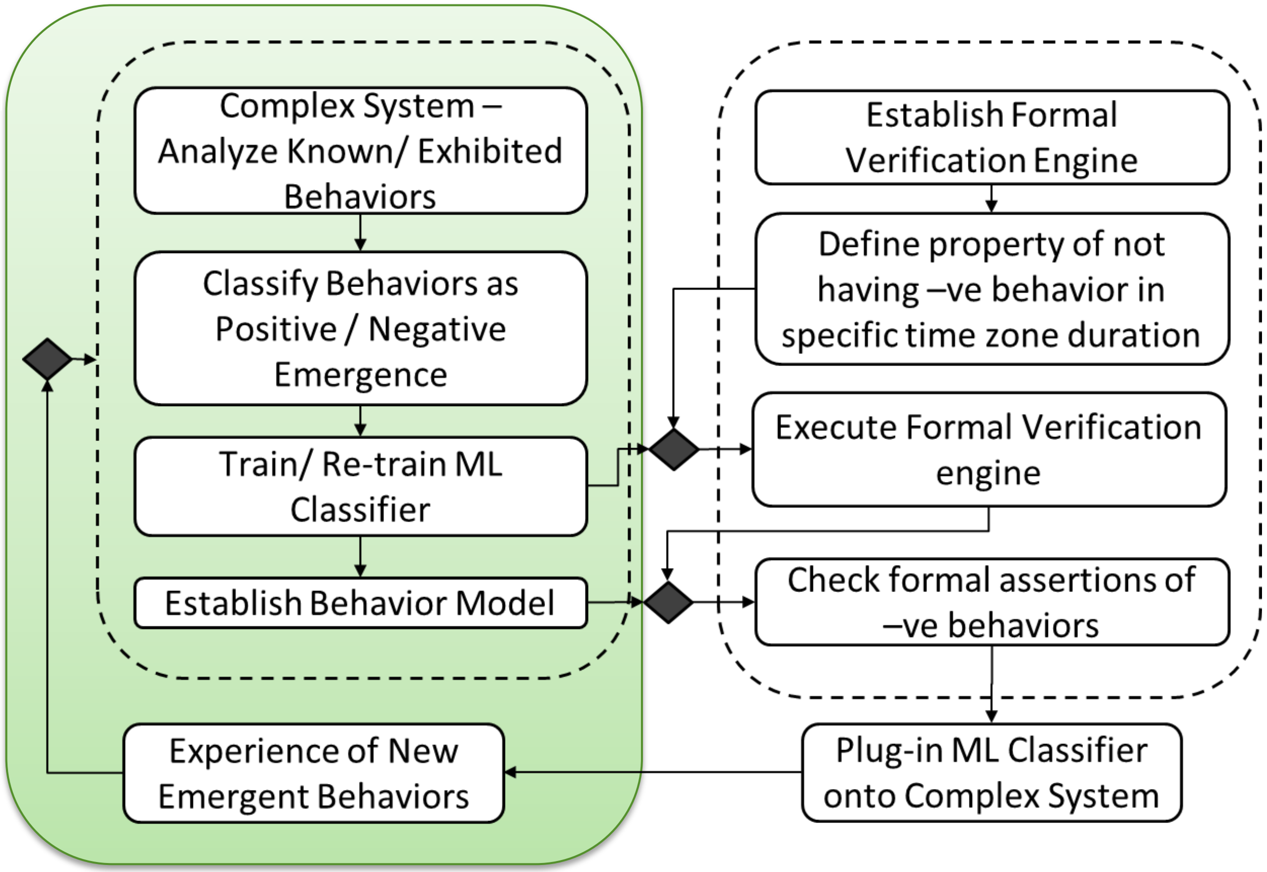
# Reflecting ….




*Observe MOEs/ MOPs, learn positive & negative emergent behavior and predict*

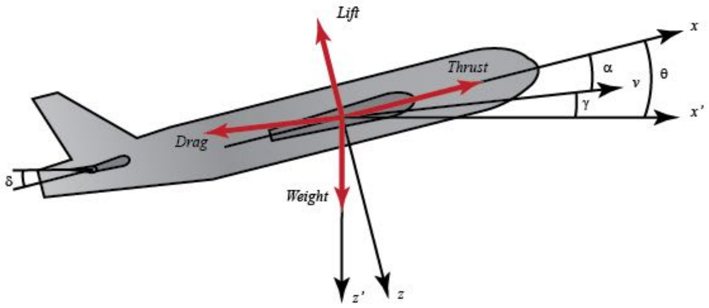
- We now have a ML Classifier that learns on potential positive and negative emergence behavior of a complex system
- The ML Classifier uses the same inputs as the complex system ML Controller uses

*Now, can we leverage the classifier in a formal verification model checking environment to assert negative emergent behavior…?*



# Applying Formal Methods

- Initial states $s_0$ is in a region $R_p$ of positive emergent behavior.
- Process C(t, S) acts on the states $s_t$ and transforms it as a function of time.
- Bounded disturbance to the system $\mu_{t0}$ at a time $t_0$ such that the system states are perturbed
  - This could be a step change to the theta demand or set point or a disturbance as a wind gust to the aircraft.
- There is a time $T_r$ called the response time of the system in which the system or process C should get back to equilibrium or a region of positive emergent behavior.
  - $T_r$ could be the measure "time to double" for the system. If this does not happen, it would imply a negative emergent behavior.




# Formal Verification Engine

MathWorks Simulink Design Verifier is used to assert that in a 2 second ($T_r$ =200 frames) simulation, the system will be stable or not

**ZONE BASED CLASSIFIER**

- *If for a given value of q the theta is greater than the upper limits (as defined by the zone) or if theta is less than the lower limit (as defined by the zone) then the system is unstable.*
- *The system should be indicated unstable for a persistent time of 5 frames to avoid spurious toggle.*

**ML BASED CLASSIFIER**

- *When the input demand (set point) transitions from 0 to 5 start a timer.*
- *If timer > 200 the classifier output is greater than 0.3.*

# ML Classifier Performance

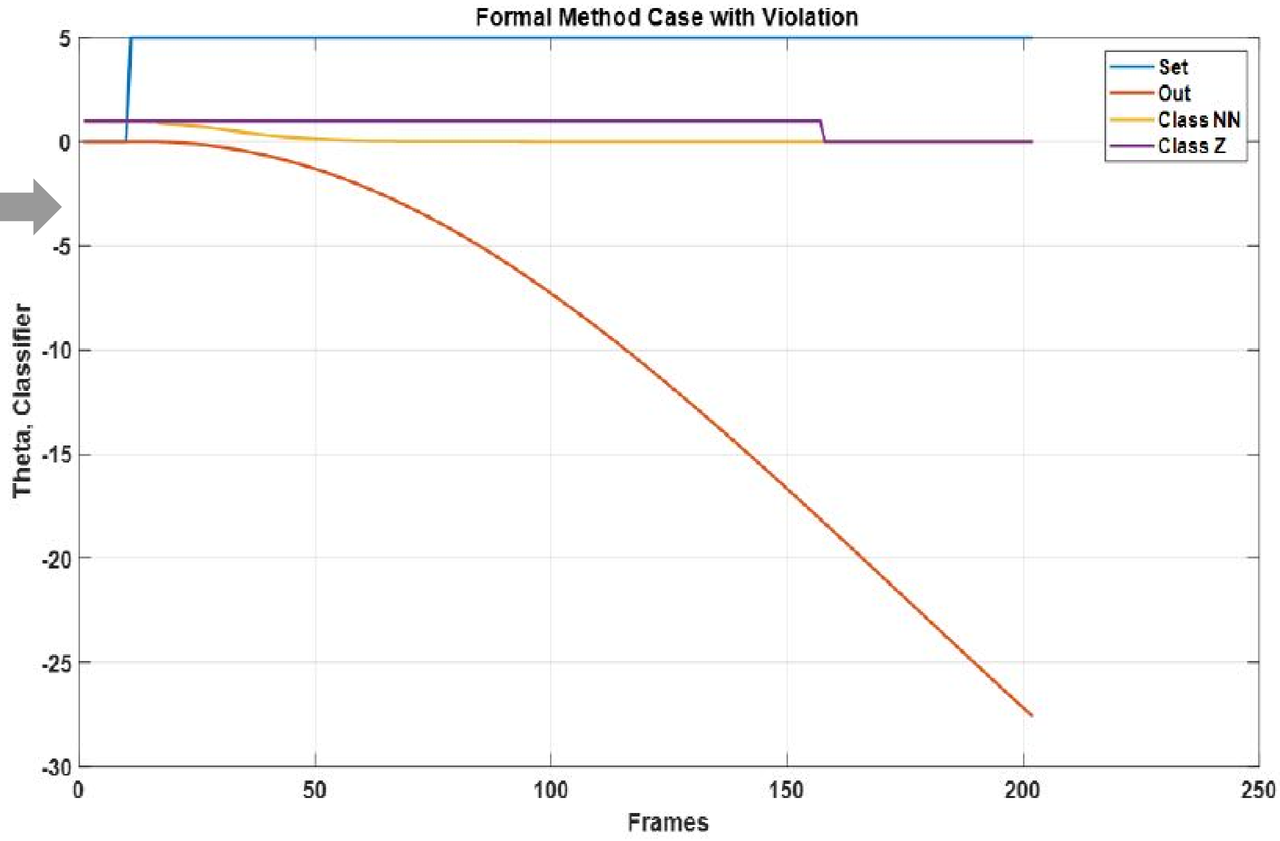
ML Classifier

Zone Classifier

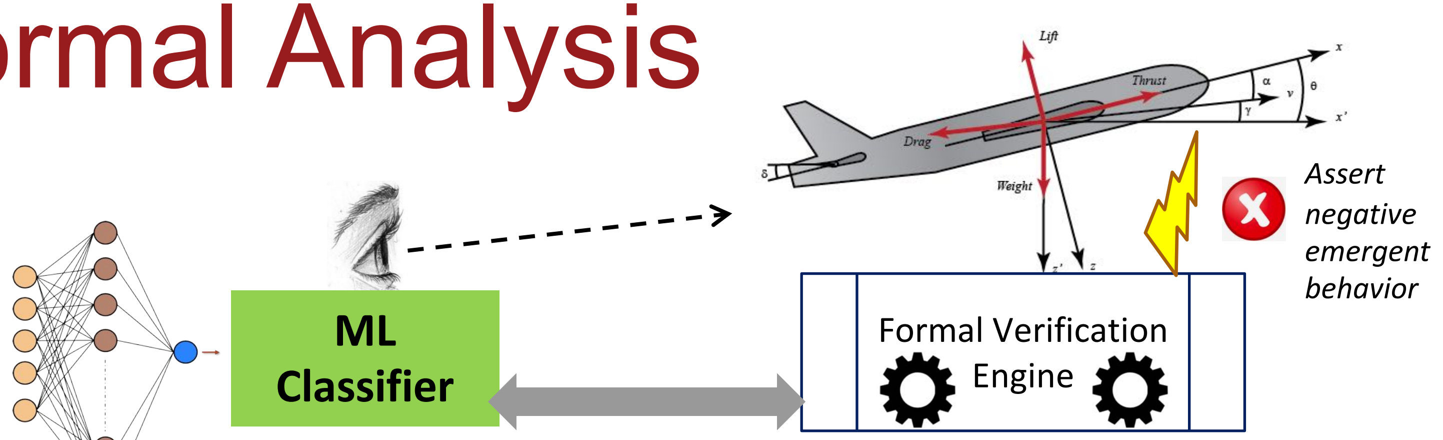
The ML Classifier identifies earlier (at about 50 frames, with value nearing zero, indicating negative emergence), as compared to Zone Based Classifier that indicates a negative emergent property at around 160 frames

PID Controller

ML Controller

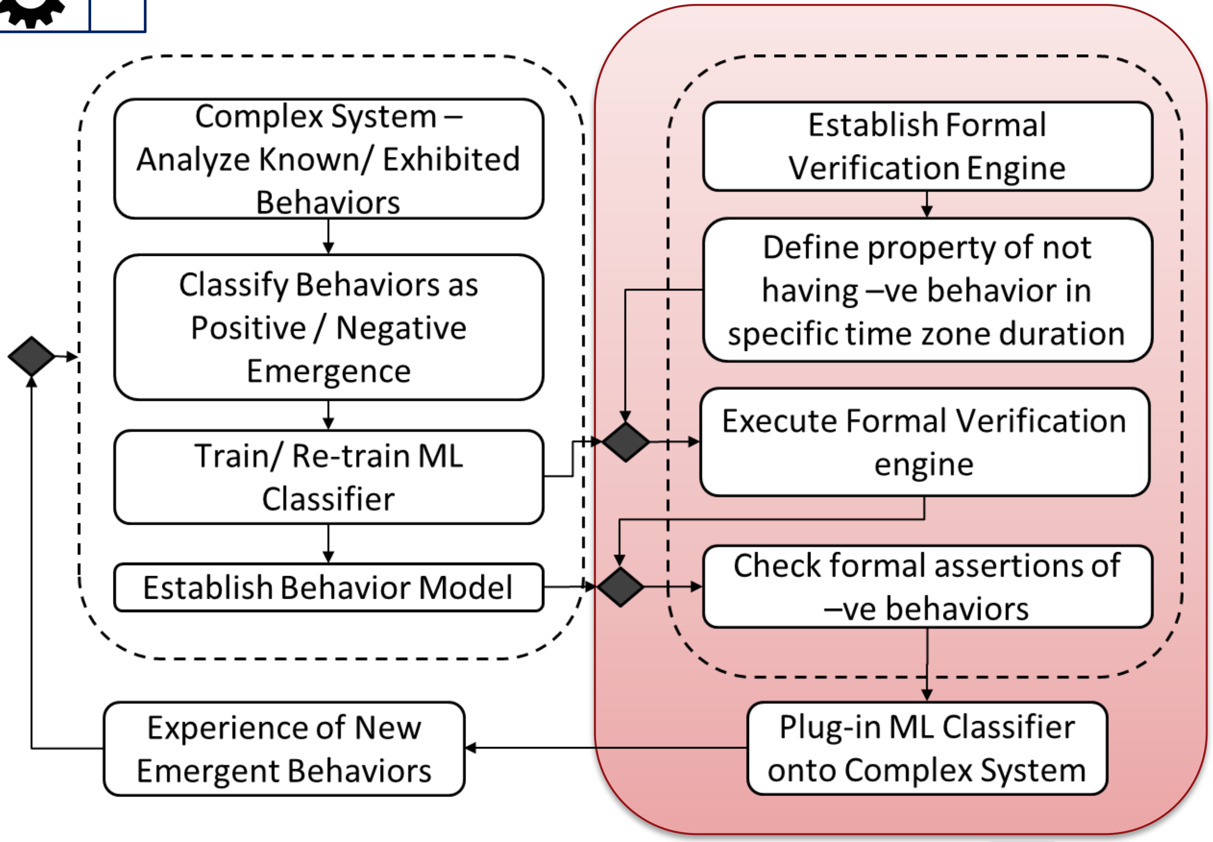


# Formal Analysis




*Observe MOEs/ MOPs, learn positive & negative emergent behavior and predict*

- ✓ We have explored an arbitrary stable state as an input criteria and carried out the formal analysis.
- ✓ The states are selected from the stable zone. Based on the initial error the set point is computed and used as constant throughout.
- ✓ We define that there shall not be any divergence (negative emergent property) in a fixed duration of 200 samples.
- ✓ We deliberately make the system unstable and the formal engine is able to provide a violation.
- ✓ The converse, that the system is stable takes significant computing resources and time, and hence we propose future directions to exhaustively explore these scenarios



# Conclusions

- Presented a novel approach for applying formal methods towards identification and assertion of positive and negative emergence behavior
- The ML Classifier and the Zone Based Classifier can predict the negative emergent behavior of the system.
  - The ML Classifier is specifically applicable for machine learning based complex systems, since it can be based on tapping the same inputs as that used for the machine learning subsystem that implements the functionality
  - The functioning of the classifier is independent of the control algorithm or the process.

# Future Work

- Counter Example provided by Formal Verification Engine - identify the root cause of this behavior and correct the system for counter acting this behavior
- Asserting system does not violate all applicable properties – requires significant computational resources
- Enhance the proposed approach for complex system-of-systems that have mix of machine learning based systems and traditional / manual system

29th Annual INCOSE international symposium

Orlando, FL, USA

July 20 - 25, 2019

www.incose.org/symp2019