Data-Driven Prognostics and Diagnostics of Industrial Machinery — A Turbofan Engine Case Study

Russell Graves[1], Peeyush Pankaj[2], Rachel Johnson[1], Michio Inoue[3] and Vineet Jacob Kuruvilla[4]

[1]*MathWorks USA, 3 Apple Hill Drive, Natick, MA, USA 01760-2098*
russellg@mathworks.com
rachelj@mathworks.com

[2] *MathWorks India, Trillium Building, Blocks I & J, Embassy Tech Village, Bangalore, India 560103*
ppankaj@mathworks.com

[3]*MathWorks Japan, 7/F Akasaka Garden City, 4-15-1 Akasaka Minato-ku, Tokyo 107-0052 Japan*
minoue@mathworks.com

[4] *MathWorks Singapore, 10C, #06-49, Ubi Techpark, Singapore 408564*
vkuruvil@mathworks.com

ABSTRACT

A machine's Remaining Useful Life (RUL) is the expected life or usage time remaining before the machine requires repair or replacement. In data-driven methods, typical RUL estimation is performed using models trained with health condition indicator values derived from measured system data. A significant challenge in developing an RUL estimation model is transforming large, multivariate, noisy sensor datasets into useful format(s) that make the data analysis and processing pipeline efficient and extract valuable condition indicators from the data. This work uses the N-CMAPSS dataset to explore options and implications for efficiently organizing and storing large time-series datasets to support prognostics and diagnostics applications. We extend the work to demonstrate a predictive maintenance workflow and solution to (1) detect and classify faults in a turbofan engine and (2) estimate the RUL once we detect performance degradation.

Under data engineering, we investigate the impact of various file formats and file types on memory and execution time when dealing with large datasets like N-CMAPSS. We analyze, pre-process, and extract/engineer critical features from the transformed dataset by leveraging our understanding of gas turbines' operation (e.g., Brayton Cycle). We also analyze the performance of various engine submodules for different flight phases (climb, cruise, and descent). This work also explains an approach to down-sample the time series data without losing information relevant to our goals. Using the health condition indicators derived and synthesized in the data engineering stage, we train machine learning models for diagnostics (differentiate between healthy operation and seven different types of faults in the turbofan engine) and prognostics (RUL estimation).

1. INTRODUCTION

Predictive maintenance can be considered the holy grail of industrial machinery equipment manufacturers and operators. It helps monitor the health of equipment to estimate its Remaining Useful Life (RUL). These techniques will help transition from reactive maintenance to a preventive and optimized maintenance strategy. There is immense value to gain from having a proactive maintenance strategy, such as cost savings [1], productivity increase for the maintenance crew, and even opening new service/revenue streams [2].

This paper focuses on a data-driven approach to aircraft engine prognostics and diagnostics. We used the N-CMAPSS dataset [3] to demonstrate a predictive maintenance development workflow, and we answered the three main questions for any predictive maintenance application: 1. Is our aircraft engine or engine components' health degrading at an abnormal rate? 2. Which subsystem(s) is failing? and 3. How many flight cycles remain before the engine fails?

Figure 1 depicts a typical data-driven predictive maintenance development workflow. In the work, we will delve into key aspects of each stage in the workflow. Reliable data pipelines ensure the availability of high-quality data, enabling accurate predictive models. We will focus on the data engineering portion of the workflow. We touch upon transforming data into usable formats and comparison of these formats' impact on memory footprint and computation time. We will briefly describe the N-CMAPSS dataset, explain each step of the workflow and our implementation details, and conclude with potential extensions to this work. We will not focus on the deployment stage in this paper.

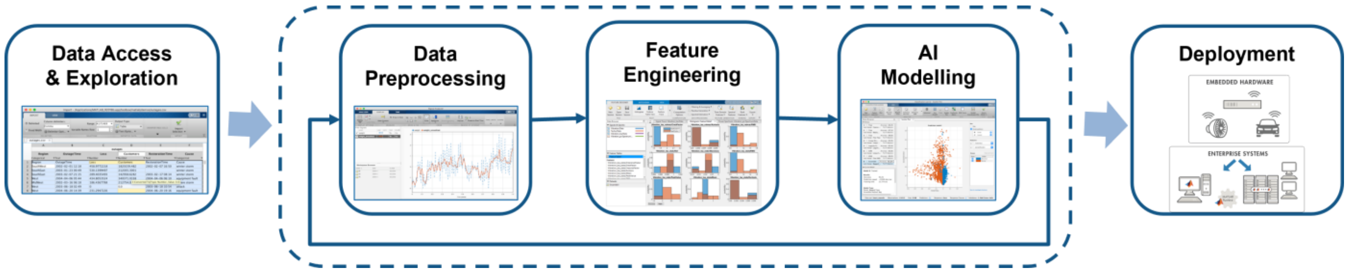

Figure 1 A typical data-driven Predictive Maintenance development workflow

2. DESCRIPTION OF DATASET

N-CMAPSS refers to a new and improved version of the CMAPSS dataset [4]. CMAPSS stands for Commercial Modular Aero-Propulsion System Simulation, the high-fidelity system model developed at NASA used to generate the dataset. The dataset contains eight run-to-failure trajectories for a fleet of 128 aircraft engines under different flight conditions. Failures can occur in either the flow (F) or efficiency (E) of different subsystems: fan, low-pressure compressor (LPC), high-pressure compressor (HPC), high-pressure turbine (HPT), and low-pressure turbine (LPT), as indicated in Table 1.

Table 1 Overview of N-CMAPSS datasets [3]

Name	# Units	Flight Classes	Failure Modes	Fan		LPC		HPC		HPT		LPT		Size
				E	F	E	F	E	F	E	F	E	F	
DS01	10	1, 2, 3	1							✓				7.6 M
DS02	9	1, 2, 3	2							✓		✓	✓	6.5 M
DS03	15	1, 2, 3	1							✓		✓	✓	9.8 M
DS04	10	2, 3	1	✓	✓									10.0 M
DS05	10	1, 2, 3	1					✓	✓					6.9 M
DS06	10	1, 2, 3	1			✓	✓	✓	✓					6.8 M
DS07	10	1, 2, 3	1									✓	✓	7.2 M
DS08	54	1, 2, 3	1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	35.6 M

Each file contains the simulated results of aircraft engines as second-by-second flight data up to 100 flights or engine failure, whichever comes first. Each unit experiences flights of a specific duration, indicated by flight class, and enters an "abnormal degradation state" randomly according to the file number and specified failure type.
In the dataset, we have access to:

- Generic airflow cycle measurements across the engine length, such as total temperature, pressure, and flow.
- Two rotor speeds, compressor stall margins, and some operational parameters (e.g., Mach number, altitude, throttle resolver angle, current cycle count, and flight class).
- A binary health state indicator and RUL label.
- A passenger/commercial aircraft goes through a well-defined mission: ground idle, take off, climb, cruise/mini-cruise, and descend. Only the climb, cruise, flight idle, and descend information in this dataset is present.

3. WORKFLOW, IMPLEMENTATION & RESULTS

As described in the introduction, we followed the workflow depicted in Figure 1 with the iteration of some stages to improve performance based on our observations at each stage. The workflow and analysis described in this paper are implemented using MATLAB R2023a [5]. We will highlight the salient aspects of each step of the workflow in this section.

3.1. Data Access & Restructuring

It is well known that schema and storage format can impact processing performance, drive footprint, portability, readability, and ease of access to data. The initial dataset was provided in a set of HDF5 files. This storage solution may be preferred if the analysis was performed with Hadoop or Spark. However, in this case, the compute environment was MathWorks Cloud Center using 24 parallel workers on an AWS instance with a 2.5Ghz Intel Xeon Platinum 8259CL CPU. A trade study was performed to determine if the data should be refactored into a new schema or file format for the best performance in our compute environment.

MATLAB® datastore enables us to point to the location of the data, ingest it using a built-in h5read function, and establish a transformation pipeline with the datastore methods `readall` and `writeall`. We explore several data configurations by refactoring the original dataset with help from built-in write functions: `save`, `writetable`, and `parquetwrite`. The configurations are described in Table 2 [6], along with their disk footprint and the time it took to write them to the disk.

The selected data configurations were then exercised with two tasks: separate flight phases (Figure 5b) and report full dataset mean temperature difference across high-pressure turbines (Figure 2a). The datastore construct in MATLAB provides many tools for working with data that will not fit into memory or exists across many files. Common large data analysis methods like transform and tall are used to address the two tasks. The transform function exercises a function on each element of a datastore to produce a `TransformedDatastore`, which can be read into memory or written back to disk. The transformation is only executed at read or write time and is optimized by MATLAB. In the case of this flight data, `tall` can produce a single monolithic table where all flight data has been vertically concatenated. The `tall` table object is not loaded into memory, and MATLAB can still interrogate this object as if it were a table in memory.

Table 2 Data configuration trade study. Disk size of final dataset reported in Gb, write time for the dataset in Minutes, ranking for access and readability from 1 (best) to n (worst), and whether the file format is portable (Y/N).

	Wide		Narrow		Nested			
	Disk (Gb)	Write (min)	Disk (Gb)	Write (min)	Disk (Gb)	Write (min)	Access	Portable
HDF5	27.2	0	-	-	-	-	4	Yes
CSV	27.3	N/A	50.5	32.6	-	-	3	Yes
MAT	12.0	4.41	9.43	6.15	8.26	6.41	2	No
Parquet	12.0	4.57	11.3	6.05	12.0	6.30	1	Yes
Readability	1		3		2		-	-
Access	1		3		2		-	-

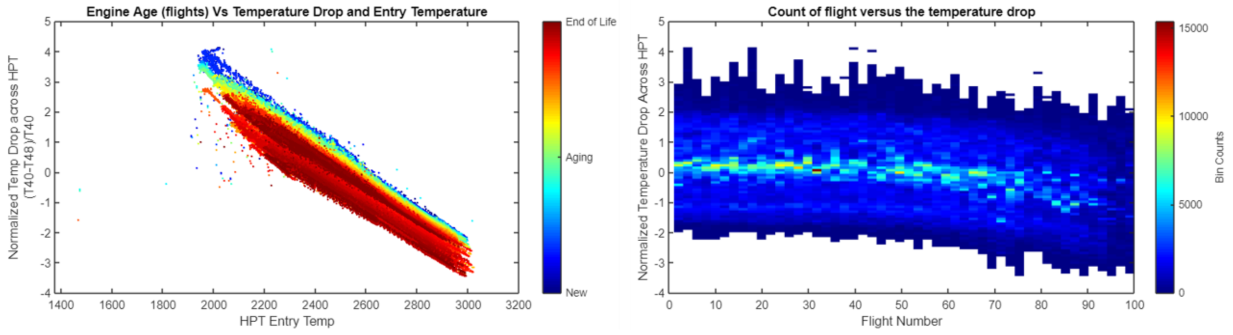

Figure 2 (a) Scatter plot of Engine age vs. Temperature and HPT Entry temperature, (b) Histogram of count of flights vs. the temperature drop across HPT

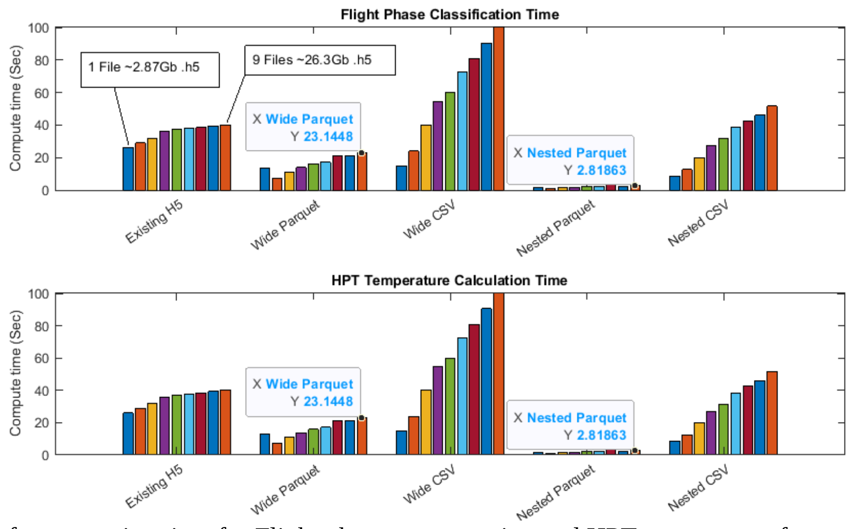

Figure 3 Comparison of computation time for Flight phase segmentation and HPT temperature for various data formats

The flight phase separation task used the datastore transform workflow. The designed transform function contains two steps for each flight:

1. Smooth data. Each flight contains climb, cruise, and descend phases, impacting the engine operation differently. We investigated whether data from some flight phases are more useful for identifying faults. Smoothing the differences in altitude will give us a cleaner way to apply a threshold to determine when the aircraft is in each flight phase, as shown in Figure 5.

2. Segment flight data. In smoothed plot, it becomes clear from the smoothed data when the aircraft is climbing, descending, or cruising. We can now apply a threshold to identify the flight phases and color-code them for easy analysis. The threshold value was specified after experimentation with single-flight data and scaling it up for the whole dataset.

The transform function produces an array of categorical labels indicating the flight phase for each sample in the time series data. The total time to read this information back into memory was recorded.

The second task was informed by domain expertise. The temperature drop across the HPT is a good measure of its overall health [7]. In this case, a tall table is created from the dataset, and the temperature drop is calculated. Functions such as `gscatter` [6] help to visualize these relationships, as shown in Figure 2a. The figure shows a strong relationship between the age of the engine and the temperature drop across HPT.

Tasks 1 and 2 were executed on the contents of the first HDF5 file, the first two files, the first three files, and so on until all nine files were evaluated to understand the impact of increasing dataset size. The resulting execution times (Figure 3), in addition to the performance metrics table (Table 2), gave us the motivation to refactor the dataset into a wide-schema set of parquet files for further processing and feature engineering.

3.2. Data Pre-processing and Feature Engineering

As with most sensor data, you need to clean and transform the raw data to create/identify the right set of condition indicators for any given asset. This is true even in the N-CMAPSS dataset. Figure 4 depicts the simplified data pre-processing steps we used in our work.

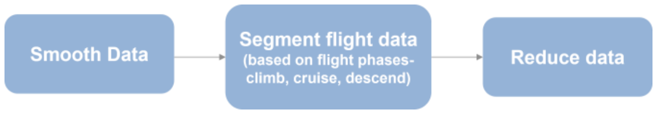

Figure 4 Data pre-processing steps

The flight phase extraction performed in task 1 of the performance evaluation for the data schema and format comprised the first portion of the pre-processing workflow for the dataset. A third and final step and data reduction were carried out as follows:
Reduce data: As we build this data by recording more flight data, the storage costs of retaining the complete dataset and processing time to train our algorithms will increase. Instead of simply downsizing, we extract change points from each sensor trajectory to retain its shape to reduce the data while maintaining enough useful information. We easily achieved this data reduction using the `findchangepoints` algorithm in MATLAB [6]. We prototype the algorithm with one sensor data and scale it up to apply it to the entire dataset (Figure 5). We were able to reduce the memory footprint from 18 GB to 7GB with this technique. A word of caution: this approach may not be a good technique if there are features of interest in the frequency domain.
Engine behavior is different in each flight phase. Therefore, we explore features from each flight phase individually. We focused on time-domain statistics such as the mean, standard error of the mean, standard deviation, skewness, variance, minimum, maximum, and range. We extracted a total of 361 features from the dataset, and based on the ANOVA algorithm, we selected the top 25 ranked features. We also engineered some features like temperature difference and pressure ratio across various subsystems and estimated their trendability. All of this feature engineering was semi-automated using MATLAB's Diagnostic Feature Designer app [6].

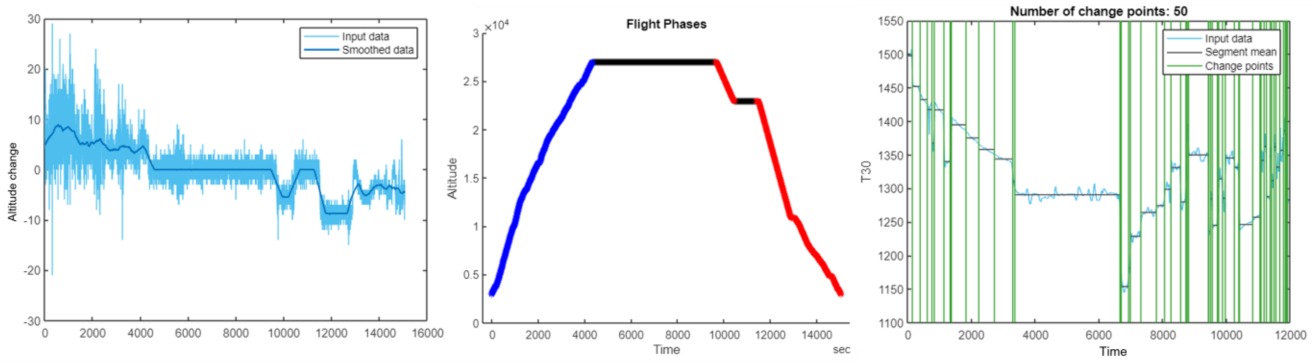

Figure 5 (a) Smooth data, b) Flight phase segmentation based on threshold, c) Reduction of data using change points

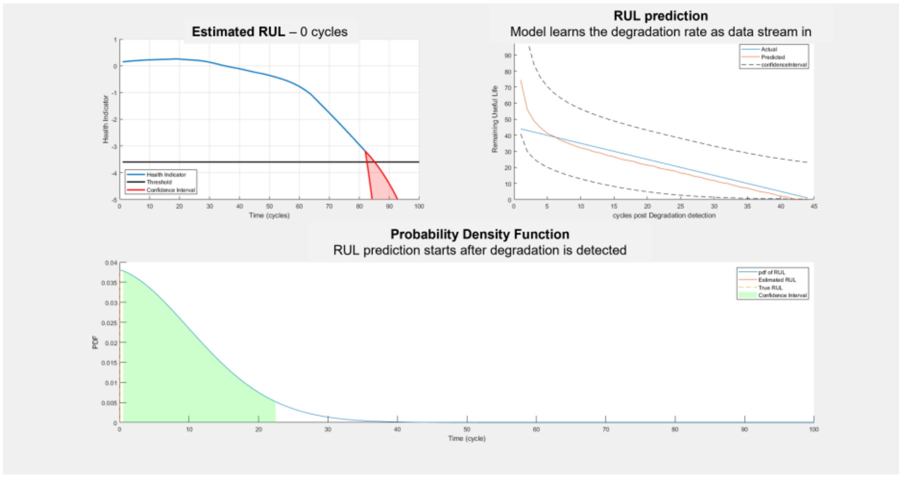

Figure 6 Visualization of health indicator trends, RUL estimation and Probability Density Function of RUL as new data streams into the data processing pipeline

4. AI Modeling

There are two critical questions for any predictive maintenance application in the AI modeling stage. First, which subsystem of the turbofan engine is failing? We treat this question as a fault classification problem. Second, what is the remaining useful life of the turbofan engine? We estimate the number of flight cycles the engine can operate before it needs to be scheduled for maintenance.

Fault classification: We train a set of machine learning models using the selected features and corresponding health labels. Using the Classification Learner app [8], we train multiple machine learning models [8] in parallel and compare and evaluate their performance with the test data using a confusion matrix and looking at the prediction accuracy. In our tests, narrow neural network models using cruise phase data were the best-performing model for turbofan engine fault classification with a prediction accuracy of 86.9%.

RUL Estimation: Estimating the RUL is a crucial part of the predictive maintenance solution. Together with fault classification, we will be able to give a complete picture of the health of the turbofan engine – which part is failing and how much time remains before requiring maintenance action. Our analysis showed temperature drop across HPT and the pressure ratio across LPT are good health indicators for HPT and LPT failure, respectively.

As multiple failure mode events are simulated in parallel, we use an Exponential Degradation model [9] as the RUL estimator model with a pre-defined threshold value (based on historical evidence). We can now use the selected health indicators to fit RUL models for each failure mode. We can also provide a complete classification and RUL estimation workflow combined with the fault classification model. We build a simple engine health monitoring dashboard (See Figure 6) that visualizes the evolution of the health indicator, estimated RUL, and the probability density function of RUL as new data from various engine unit streams into the data processing pipeline.

5. Discussion & Conclusion

This work demonstrates a streamlined workflow for developing a predictive maintenance application that gives two important pieces of information: the failing subsystem(s) and the RUL estimation. We touched upon all the key stages in the development workflow.

Using the N-CMAPSS dataset, the paper delved into the importance of data engineering and how seemingly simple decisions like selecting data formats significantly impact the computation time and memory footprint along with easing or worsening the data readability and access. For the N-CMAPSS dataset, we found that the parquet wide format gave the best performance and ease of use. We also showcased an approach to reduce the dataset size without losing useful dynamics and trends in the sensor data.

We described our workflow in a linear fashion. However, developing a robust and reliable analytics pipeline requires iterating over each stage and improving the prediction performance, whether it is through enhancing data pre-processing, featuring engineering and selection, or picking the right AI model. Though we demonstrated the workflow using a turbofan engine example, we believe this approach can be generalized for any industrial machinery.

An important limitation of the method described is that only one type of fault is identified in the fault classification stage. For example, if there is an HPT and LPT failure, it is treated as just an HPT failure. This is primarily due to the challenge in creating an feature extraction algorithm that can delineate patterns in sensor data due to different faults. One of the approach could be to model the subsystem and various faults to understand the faults and its effect on sensor data [10]

There is potential to extend this work by exploring various deployment options, whether to a cloud computing platform or as a desktop application for offline data analysis. Deploying the feature extraction module to edge devices to reduce memory storage and data transmission cost is also worth exploring. Using Deep Learning techniques for RUL estimation could also be another method to explore.

REFERENCES

1. MathWorks User Stories, *'Mondi Implements Statistics-Based Health Monitoring and Predictive Maintenance for Manufacturing Processes with Machine Learning'*, https://www.mathworks.com/company/user_stories/mondi-implements-statistics-based-health-monitoring-and-predictive-maintenance-for-manufacturing-processes-with-machine-learning.html (Accessed on Oct 27th, 2022).
2. MathWorks Stories, *'Baker Hughes Develops Predictive Maintenance Software for Gas and Oil Extraction Equipment Using Data Analytics and Machine Learning'*,https://www.mathworks.com/company/user_stories/baker-hughes-develops-predictive-maintenance-software-for-gas-and-oil-extraction-equipment-using-data-analytics-and-machine-learning.html (Accessed on Oct 27th, 2022).
3. Manuel Arias Chao, Chetan Kulkarni, Kai Goebel, and Olga Fink. (2021) Aircraft Engine Run-to-Failure Dataset under Real Flight Conditions for Prognostics and Diagnostics. Data, 6(1):5, 2021.
4. Saxena, A.; Goebel, K.; Simon, D.; Eklund, N.(2008), Damage propagation modeling for aircraft engine run-to-failure simulation. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008 pp. 1–9.
5. MATLAB. version 9.14.0 (R2023a). Natick, Massachusetts: The MathWorks Inc.; 2023
6. MATLAB R2023a Documentation, https://www.mathworks.com/help/releases/R2023a/index.html (*Accessed on 07 May 2023*)
7. Ideal Brayton Cycle, https://www.grc.nasa.gov/www/k-12/airplane/brayton.html, NASA (*Accessed on 05 April 2022*)
8. MathWorks, Classification Learner – Choose Classifier Options,https://www.mathworks.com/help/stats/choose-a-classifier.html (*Accessed on 07 May 2023*)
9. MathWorks, RUL Estimation using RUL Estimator Models,https://www.mathworks.com/help/predmaint/ug/rul-estimation-using-rul-estimator-models.html (*Accessed on 07 May 2023*)
10. MathWorks, Multi-Class Detection Using Simulated Data,https://www.mathworks.com/help/predmaint/ug/multi-class-fault-detection-using-simulated-data.html (*Accessed on 15 July 2023*)

Russell Graves is an Application Engineer based at MathWorks, Natick, USA. He specializes in machine learning and systems engineering. Before joining MathWorks, Russell worked with the University of Tennessee and Oak Ridge National Laboratory in intelligent transportation systems research with a focus on multi-agent machine learning and complex systems controls. Russell holds a B.S. and M.S. in Mechanical Engineering from The University of Tennessee.

Peeyush Pankaj is a Senior Application Engineer based at MathWorks, Bangalore, India. He has deep experience in aircraft engine designs, testing and certification. He has filed multiple patents on Advanced Jet Engine technologies and Prognostic Health Monitoring of aircraft engines. He holds a master's degree in advanced mechanical engineering from the University of Sussex, UK.

Rachel Johnson is the Product Manager for Predictive Maintenance Toolbox based at MathWorks, Natick, USA. Previously, she was a Senior Application Engineer supporting the Aerospace and Defense Industry. She holds a B.S.E. in Aerospace Engineering from Princeton University, an M.S. in Aerospace Engineering from the University of Maryland, and an M.A.T. in Mathematics Education from Tufts University.

Michio Inoue is a Senior Team Lead, Application Engineering at MathWorks Japan. His team focuses on data science and predictive analytics applications. Prior to MathWorks, he was a postdoctoral scholar at NASA/JPL engaged in Computation Fluid Dynamics (turbulence modeling) research. He received his Bachelor's degree from University of Tokyo and Ph.D. from California Institute of Technology in aeronautics and applied & computational mathematics.

Vineet Jacob Kuruvilla is the PHM Segment Manager (AeroDef) based at MathWorks, Singapore. Prior to joining MathWorks, he was a Chief Researcher at Nidec Corporation. He has extensive experience in robotics and has filed multiple patents on robotics technologies for factory automation. He received his Bachelor's degree from Cochin University of Science and Technology and, Ph.D. from Nanyang Technological University in the field of robotics system control and computer vision.