CIMAC CONGRESS 25
ZURICH, MAY 19 – 23, 2025

2025 | 411

Integrating physics-based models and ML for condition monitoring of large-bore medium-speed engines

Controls, Automation, Measurement, Monitoring & Predictive Maintenance

Rik De Graeve, Anglo Belgian Corporation

Roel Verschaeren, Anglo Belgian Corporation
Ward Suijs, Anglo Belgian Corporation
Koen Christianen, Anglo Belgian Corporation

ABSTRACT

Building on the foundation of our previous work, this paper delves deeper into the integration of physics-based models and machine learning algorithms to develop a condition-based monitoring platform for large-bore medium-speed engines. Predicting and planning maintenance tasks intelligently and efficiently remains crucial for engines where availability is paramount, providing substantial economic benefits by reducing downtime, preventing critical failures, and extending maintenance intervals based on engine condition.

Our prior research demonstrated the efficacy of a condition-based monitoring system that utilizes a physics-based digital twin, machine learning algorithms, and big data analytics. This system continuously monitors the engine using a network of sensors strategically placed across all primary engine circuits: combustion, fuel, lubrication, coolant, air/exhaust, and bearings. While physics-based models effectively detect early warnings of failures and deviations in circuits that can be accurately modeled, machine learning approaches proved advantageous for circuits like the lubrication system and the bearings, which are challenging to capture with traditional physics-based models.

In this follow-up study, we focus on creating a hybrid model that combines the strengths of both physics-based and machine learning models to enhance prediction accuracy and reliability. Specifically, we target the air/exhaust path, oil pressure, and bearings to develop a more comprehensive understanding of engine health.

The practical deployment of this system on numerous vessels introduces new challenges, such as data management, storage, accessibility for calculations, model training, validation, and active retraining. We address these challenges by implementing robust data management solutions and ensuring the hybrid model adapts to real-world operational conditions.

Our results indicate that both physics-based and machine learning models can accurately capture the engine's condition. However, translating these results into actionable health metrics remains essential. This paper presents detailed case studies and results from the deployed system.

1 INTRODUCTION

Building on the foundation of our previous work, this paper delves deeper into the integration of physics-based models and machine learning algorithms to develop a condition-based monitoring (CBM) platform for large bore medium speed engines. Such a platform can deliver significant economic benefits for engines where availability is paramount by reducing downtime, preventing critical failures and extending maintenance intervals based on engine condition rather than time.

Our previous research has demonstrated the effectiveness of a CBM-system using a physics-based digital twin, machine learning (ML) algorithms and big data analytics. This system continuously monitors the engine using a network of sensors strategically placed in all primary engine circuits: Combustion, fuel, lubrication, coolant, air/exhaust and bearings. While physics-based models are effective at detecting early warnings of failures and deviations in circuits that can be accurately modelled, machine learning approaches proved advantageous for circuits such as the lubrication system and the bearings, which are difficult to capture with traditional physics-based models.

In this follow-up study, we improve upon the existing models and work towards a hybrid model that combines the strengths of both physics-based and machine learning models. This fusion is particularly useful for applications where certain behaviours follow well-defined causal relationships, while others emerge from complex, hidden dependencies within large data sets. By leveraging both approaches, the model improves predictive accuracy and adaptability in scenarios where purely analytical or data-driven methods fall short.

In addition, we'll focus on the practical implementation of this system on multiple vessels, which presents new challenges such as data management, storage, accessibility for calculations, model training, validation and active retraining. We address these challenges by implementing robust data management solutions and ensuring the hybrid model adapts to real-world operational conditions.

2 PHYSICS BASED MODELS

2.1 Introduction

Physical model-based approaches typically employ mathematical models that are directly linked to physical processes that contribute to the health of a component or system. Physical models are developed by domain experts, and their parameters are validated by large data sets.

The initial development of the physics-based digital twin model in use here has been described in a previous work [1]. A mean value model was chosen rather than a crank-angle resolved model. Although the latter, exemplified by the fill-and-empty model, offers greater accuracy in simulating engine performance, these models fall short of meeting real-time operational requirements [2].

Over time, important shortcomings of the original mean value model were identified. Key areas requiring improvement included the need for parametrization, optimization of computational efficiency, enhanced calibration and validation approaches and the enhancement of model fidelity. To address these challenges, the following changes were made:

1. ***Expanded parametrization***:

   The initial model was developed for a specific engine type, an ABC 6-cylinder D36 PLN engine. This resulted in little flexibility and a lot of effort when trying to port the model to other engine types with different cylinder counts, fuel injection systems, etc. Therefore, the model was refactored into a much more flexible and parametrizable framework, enabling users to define key engine characteristics within a single structured configuration file. This eliminates the need for multiple engine-specific codes. Given the vast number of possible engine configurations, this enhancement was essential for ensuring scalability and efficient model management.

2. ***Improved computational efficiency***:

   To meet the increasing simulation demands, optimizations in the solver were necessary. As additional engines and engine configurations were integrated into the framework, the computational load grew significantly, requiring targeted improvements to enhance efficiency. These optimizations ensured that the model remained scalable and capable of handling complex simulations without excessive computational overhead.

3. ***Enhanced Calibration Mechanisms***:

   With the increasing number of engine configurations and engines that were integrated into the framework, manual calibration and visual validation became unsustainable. To address this, the process was systematically automated by integrating both the calibration and validation step into the

work orchestration tool, which is discussed in Chapter 5. This automation enables real-time adjustments based on empirical data while providing a structured output that clearly indicates the success or failure of the calibration and validation process.

4. ***Future proofing alternative fuels***

To ensure the longevity and adaptability of the model, it has been designed to accommodate future advancements in alternative fuels. The framework supports the inclusion of a wide range of fuel types and corresponding combustion models, including regular diesel, spark-ignited hydrogen mono-fuel, dual-fuel methanol port fuel injection, dual-fuel gas, and more. By maintaining a flexible and modular structure, the model can easily integrate new fuel technologies as they emerge, enabling seamless adaptation to evolving industry standards and regulatory requirements. This future-proofing approach ensures that the digital twin remains a valuable tool for analyzing and optimizing next-generation engines.

2.2 Model Description

The new model framework has been refactored to be fully parameterizable. All parameters from Table 1 can now be freely configured.

Table 1 - Parametrization Simulation Model

Engine Component	Options
Cycle	4 stroke
Cylinders	4 / 6 / 8 / 12 / 16 / 20 / …
Bore	256 / 365 / 230 / …
Stroke	310 / 420 / …
Compression Ratio	12.5 / 15.5 / 17.5 / …
Volumetric Efficiency	ABC map based on 1D sim
Low Pressure TC	KBB 4D map
High Pressure TC	KBB 4D map
Injection system	PLN / CR / DF PFI
Fuel	Diesel / Hydrogen / Methanol
Waste Gate	Discharge Coefficient

A schematic of the model components can be seen in Figure 1. The mean value model assumes that the airflow at the inlet and exhaust is nearly homogeneous and steady. Air is modelled as a compressible fluid. Air and combustion gasses are modelled with different heat capacities. Additionally, as the air flows from one component to another, it can be thought of as travelling through a stream tube. Inside a stream tube, the air is assumed to act as an ideal fluid. An ideal fluid is one in which there is no viscous stress and no heat conduction. Within the momentum equation the effects of potential energy are neglected. With this assumption the total enthalpy is constant in a stream tube.

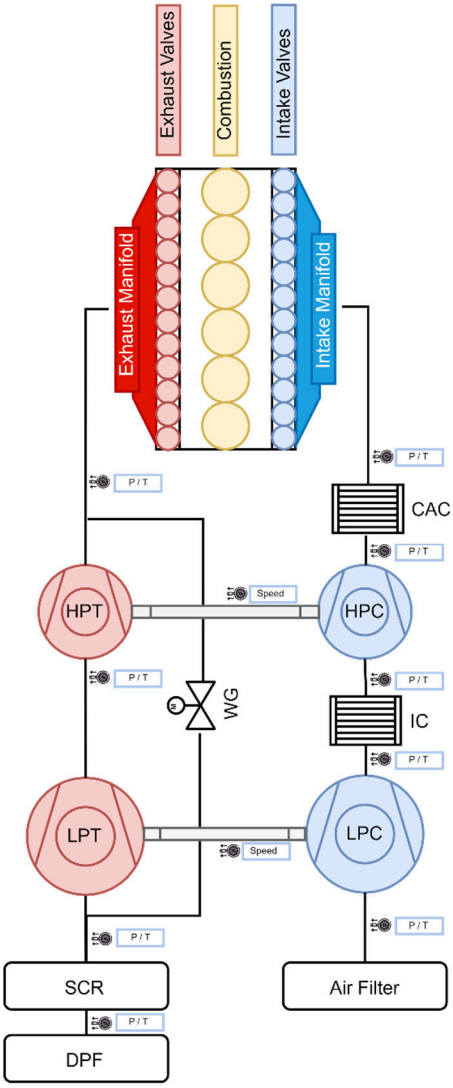

Figure 1. 0D/1D engine model schematic. P – pressure; T – temperature; LPC – low pressure compressor; HPC – high pressure compressor; IC – intercooler; CAC – charge air cooler; HPT – high pressure turbine; LPT – low pressure turbine; SCR – selective catalytic reduction; DPF – diesel particulate filter.

2.3 Model Validation

The simulation model was calibrated and validated extensively based on test bench measurements. After validation on test bench measurements, it was deployed using real world transient engine data, of which results are discussed in the following sections.

To thoroughly assess the model's performance, three distinct validation periods were conducted, each designed to evaluate its ability to accurately

capture engine behaviour over different timeframes. The goal is to prove that the model is robust and accurate across different time scales.

One parameter, the pressure after the low-pressure compressor, was chosen for the discussion of the validation. Other parameters will not be discussed in detail, but are equally well captured by the model.

2.3.1 Short length timeframe validation

A first validation of the engine model on field data was carried out in a shorter timeframe to reduce the computational cost and improve the interpretability of the results. A timeframe of 4 days was opted for to ensure enough variability in the engine operation while still allowing the data to be represented in a time series graph for validation. This validation provides an initial indication of the model reliability in predicting engine behaviour. Results are presented in four distinct types of graphs, each serving a unique purpose in order to get a good understanding of how the model performs.

Figure 2 shows a time series plot of both the modelled and measured data. It is clear from the measured data that the engine's operation is not steady state and load is varying constantly. Based on an initial visual interpretation of the graph, the modelled value clearly matches the measured value well. Note however, that Figure 2 does not give much insight into how well the model is performing for specific operating conditions of the engine, nor does it provide more than a qualitative comparison of the modelled and measured values. Therefore, three additional views were designed to gain a better understanding of the model performance.

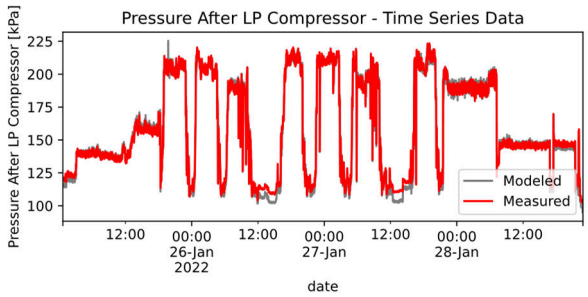

Figure 2 - Time Series Data of the modelled and the measured Pressure After the LP Compressor

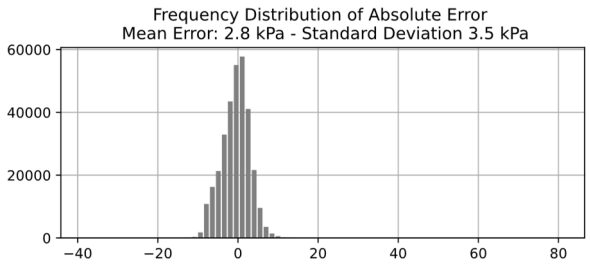

Figure 3 – Frequency distribution of the Absolute Error between the modelled and measured Pressure after the LP Compressor.

Figure 3 shows the frequency distribution, mean error and standard deviation of the absolute error between the measured and modelled data. The distribution of errors is unimodal but slightly negatively skewed, indicating that the model tends to underestimate the modelled values more often this it overestimates them. The low mean error indicates that, on average, the model's predictions closely align with the measured values, suggesting minimal systematic bias. Additionally, the low standard deviation implies that the errors are relatively consistent and do not vary significantly across the dataset. This proves that the model provides stable and reliable predictions with minimal fluctuations in accuracy.

Figure 4 shows a contour plot of the absolute difference between the measured and modelled values mapped to the engine speed and load. This allows us to clearly determine whether the engine model works well for the entire engine operating range. A few conclusions from this graph show that:

- The engine speed varies between three distinct setpoints: 600 / 685 / 720 rpm.
- The engine load varies continuously over these distinct speed setpoints ranging from 20% - 100% load.
- The model captures the engine operation well but shows that at low load the model slightly underestimates the pressure, while at high load this is not an issue. This explains the slight negative skew in the distribution as seen in Figure 3

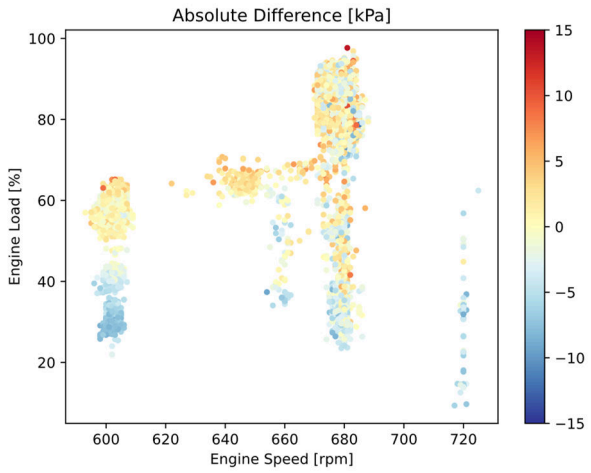

Figure 4 – Contour plot of the absolute difference between the modelled and measured Pressure after the LP Compressor.

Figure 5 presents a linearity plot, a scatter plot that compares measured values on the x-axis and modelled values on the y-axis. This plot is often used to evaluate a model's predictive accuracy, particularly for large datasets. The linearity plot also includes a density representation, where lighter-colored points indicate areas with a higher concentration of data. This visualisation helps identify where most of the measured and modelled values align, highlighting the most common error patterns. A strong clustering of high-density points around the y=x line suggests that the model performs well for the majority of cases. While deviations in lower-density regions may indicate outliers or specific conditions where the model is less accurate.

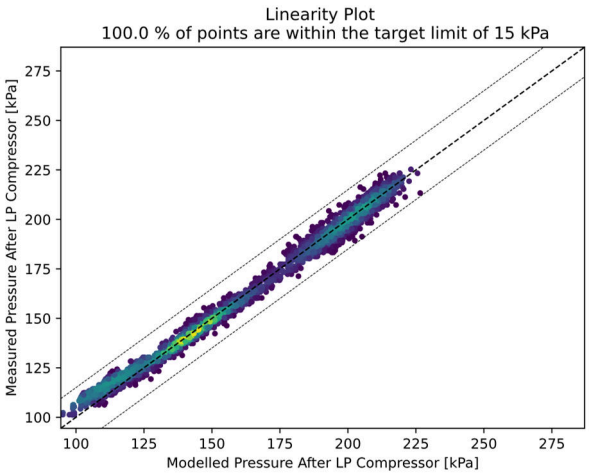

Figure 5 – Linearity plot of the modelled and measured Pressure after the LP Compressor.

2.3.2 Medium length timeframe validation

After the initial validation on a limited timeframe, more extensive validation was done for an extended timeframe of 3 months. This allows an assessment over a more extended period, capturing potential seasonal variations and longer time trends, while still being relatively short enough to retain a high degree of granularity. Results are represented in the same graphs as discussed for the short timeframe validation in Figure 6, Figure 7, Figure 8, Figure 9. The conclusions are summarised below:

- A qualitative validation based on Figure 6 does not make sense anymore due to the fine-grained nature of the data resulting in an overcrowded graph. Making it impossible to discern meaningful patterns or trends.
- Figure 7, Figure 8, Figure 9 present results that are consistent with those observed during the short-term validation, indicating that the model is capable of accurately capturing the engine's behaviour over extended periods of time.

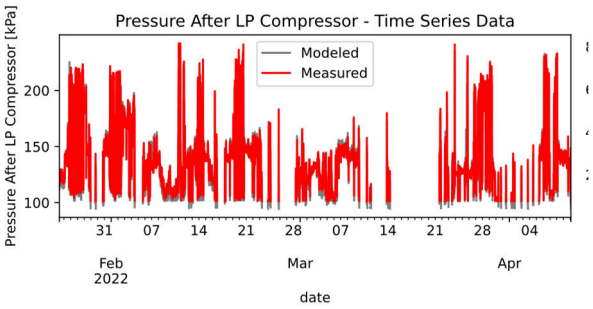

Figure 6 - Time Series Data of the modelled and the measured Pressure After the LP Compressor.

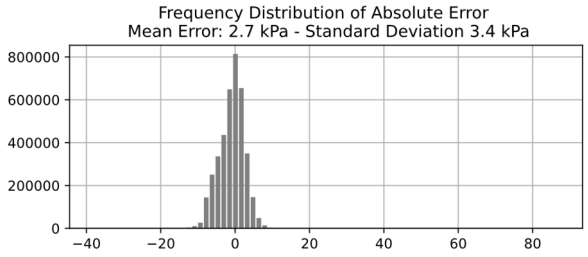

Figure 7 – Frequency distribution of the Absolute Error between the modelled and measured Pressure after the LP Compressor.

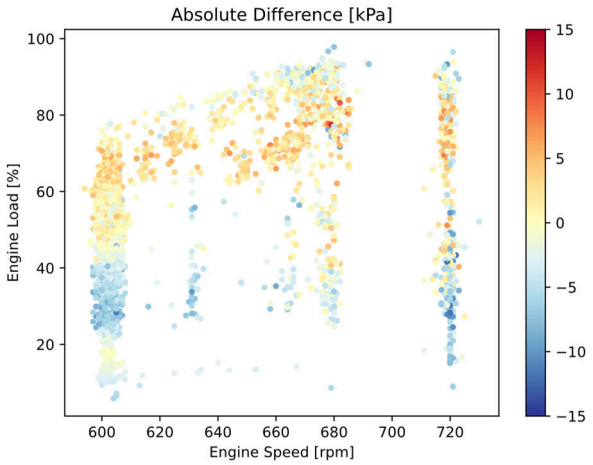

Figure 8 - Contour plot of the absolute difference between the modelled and measured Pressure after the LP Compressor.

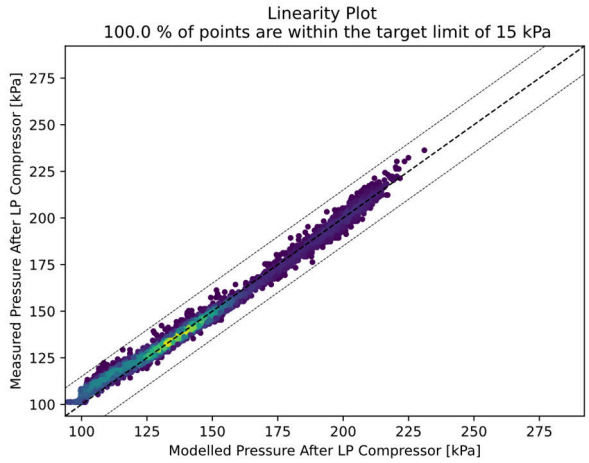

Figure 9 – Linearity plot of the modelled and measured Pressure after the LP Compressor.

2.3.3 Long length timeframe validation

The final validation involves a comprehensive three-year period. This long-term validation provides a thorough test of the model's ability to maintain accuracy over a significant period, accounting for long-term trends and potential changes in the engine's behaviour over time.

Results are bundled in a different representation, more suited for long term validation of model. Figure 10 shows a validation plot of the pressure after the low-pressure compressor. The main plot presents a time series representation of the relative error between the measured and modeled values. Within this main plot, two subplots are included: one illustrating the distribution of the absolute error between the measured and modeled values, and the other displaying a linearity plot.

In conclusion, the relative error, the distribution of absolute error, and the linearity plot collectively demonstrate that the model performs well, even when applied to a very large dataset. The relative error remains within acceptable bounds, the distribution of absolute errors shows no significant biases, and the linearity plot confirm a strong correlation between the measured and modeled values.

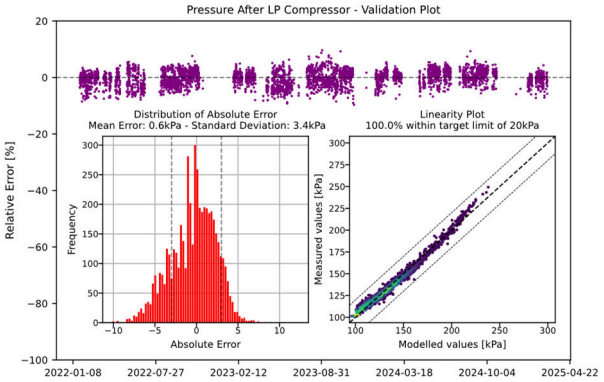

Figure 10 - Validation plot of the Pressure After LP Compressor spanning over a 3-year period.

2.4 Limitations

The accuracy of physics-based engine models heavily depends on the foregoing calibration process. Accurate engine modelling requires precise calibration of numerous parameters, such as volumetric efficiency, fuel injection strategies, combustion characteristics, turbocharging behavior, heat exchanger characteristics, etc. Traditional calibration approaches rely on controlled test bench experiments and physics-based 0D/1D simulation to construct these mappings.

However, real-world engine behavior frequently differs from controlled test conditions due to factors such as ambient variations, component aging, differences in engine integration within the installation, and dynamic operating conditions. Additionally, engine testing is often limited to standardized operating points mandated by emissions regulations, which may not accurately represent real-world usage scenarios.

Chapter 4 proposes a solution to this problem using a hybrid modelling approach, incorporating real engine telemetry and machine learning algorithms into the physics-based model. First, however, the next chapter will discuss the different machine learning algorithms that exist, and which ones are best suited to our needs.

3 MACHINE LEARNING ALGORITHMS

3.1 Introduction

Machine learning techniques, such as neural networks, support vector machines (SVM), and decision trees, can analyze vast amounts of sensor data, identify patterns, and predict failures before they occur. In our previous work [1], we used these techniques solely for parameters which were too complex for physics-based approaches. For example, ABC developed a machine learning model in collaboration with PolySense [3] for monitoring the oil pressure inside the engine's oil circuit. Other parameters, such as pressures, temperatures, rotational speeds were modeled with the mean value model.

However, machine learning can provide an alternative approach that can capture intricate patterns and relationships within this data as well that might be difficult to explicitly define through physics-based models alone. Therefore, in this work, we have expanded our methodology to apply machine learning across a broader range of engine parameters. Rather than focusing on a single parameter, our approach now aims to model the entire engine's behavior. This is made possible by an automated and standardized pipeline that enables the development of predictive models for multiple engine parameters in a consistent and scalable manner (see Chapter 5). By utilizing machine learning techniques, we can explore correlations between parameters and better understand how they interact under different operation conditions, improving predictive accuracy and adaptability across various engine configurations.

The following sections explore the selection, training, application, and validation of these models using extensive datasets derived from multiple engines, with a particular focus on field data generated under real-world operational conditions.

3.2 Model Selection: Production vs Academia

To find the right ML algorithm to work with, the distinction between machine learning research and production ML needs to be addressed first, as they are characterized by significant differences in objectives, constraints and operational challenges [4]. Academic ML typically emphasizes optimizing model performance within controlled, idealized environments, where the focus is on pushing the boundaries of algorithmic performance and achieving high accuracy under carefully curated conditions.

These specialized ML techniques, which may be tailored to specific use cases or optimized for particular tasks, are often seen as a means to achieve state-of-the-art performance. However, in a production environment, such highly specialized methods may not always be the best choice. The complexity can introduce issues such as overfitting, difficulties in generalization, and increased computational and maintenance costs.

In contrast, more generalized and flexible machine learning approaches, when carefully designed for scalability and robustness, tend to perform better in production environments. These models are easier to deploy, update and maintain while ensuring consistent performance across a variety of scenarios.

The fundamental difference lies in the shift from model-centric to system-centric thinking. Overlooking this distinction can lead to failures in ML deployment. We therefore deliberately chose to focus our efforts on well-established, rather than specialized models. An overview of the different models investigated can be found in Table 2.

Table 2. Overview of the different MLA depending on their applications.

Machine Learning Algorithms		
Engine diagnostics		
	General	Regression
	Fault detection	Decision Trees
		Random Forests
		Support Vector Machines
	Classification	Neural Networks
	Anomaly detection	K-Means
		DBSCAN
		Autoencoders
		PCA
Predictive maintenance		
	Remaining useful life (RUL)	Linear Regression
		Support Vector Regression
		XGBoost
	Degradation	Long Short-Term Memory
		GRUs
		ARIMA

3.3 Case Studies

It was observed that the approach used for developing the machine learning models could be effectively generalized, provided that variability in the feature selection process was incorporated. Specifically, by allowing flexibility in the selection of features, the model can be adapted to different datasets, different engines, and a wide range of parameters.

The pipeline (generalized approach) is designed to create models for various engine parameters, such

as pressures, temperatures, speeds, and others. While the approach is generalized, it is crucial to recognize that each parameter requires a unique set of features for optimal model performance. Therefore, a robust and adaptable feature selection method is essential to ensure that the most relevant and informative features are identified for each specific model.

To further support the required flexibility, a promising solution is through the use of linear regression enhanced with polynomial features. Polynomial regression [5], which extends linear regression by including higher-degree polynomial terms of the original features, allows the model to capture non-linear relationships between the input features and the target parameters. This approach strikes a balance by providing the necessary flexibility to model complex patterns while maintaining the simplicity and interpretability of linear regression.

The following paragraphs present a discussion of two models: one for predicting the main bearing temperature of the engine, and the other for estimating the charge air pressure of the engine.

3.3.1 Bearing Model

The first model to be discussed is designed to predict the main bearing temperatures of the engine. Typically, bearing health is monitored using accelerometers, which detect changes in the frequency spectrum that can indicate wear or failure. However, due to the cost of the high-speed data capturing necessary for these accelerometers, these are not available for our analysis.

The pipeline described in the previous section was employed for both the feature selection and model training. The model was trained and validated across three different timeframes: short-term (a few hours), medium-term (three months), and long-term (three years). This multi-timeframe approach, as previously employed for the physics-based models, allowed for a comprehensive evaluation of the model's performance and its ability to capture engine behavior across various operational periods.

Results are bundled in similar representation as introduced in a Chapter 2.3.3, Figure 10. With the addition that the data that is used for training the model is highlighted in blue, and the data on which the model is validated, is highlighted in red. This is done because in time series forecasting, traditional random splitting of data into training and testing data is not appropriate, as it can result in data leakage, where information from the future is used to predict the past, and in overfitting the model. Instead, the data should be split chronologically, ensuring that the model is trained on past data and tested on future data. This best simulates real-world forecasting where future values are predicted based on past observations.

Figure 11 shows the validation plot for a short-term period. The relative error, mean error and standard deviation are small, indicating the model captures short term periods really well. It is clear from the linearity plot that the bearing temperature does not fluctuate much, only 5°C with varying load. Even these small fluctuations are very well captured by the model.

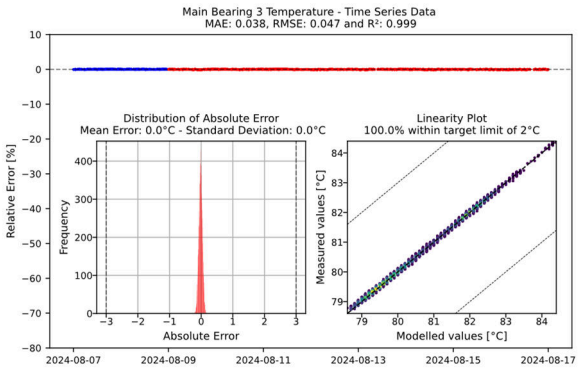

Figure 11 - Validation plot of the bearing temperature model for the main bearing temperature over a short-term period.

Figure 12 and Figure 13 show the validation plot for the medium-term and long-term timeframes. The relative error, mean error and standard deviation are higher than the short-term validation, but they are still well within bounds of a very good model. Upon closer examination of the relative error, it is observed that over time, the error transitions from a slightly positive value to a slightly negative value. This trend suggests that the actual bearing temperatures are gradually increasing under the given operating conditions.

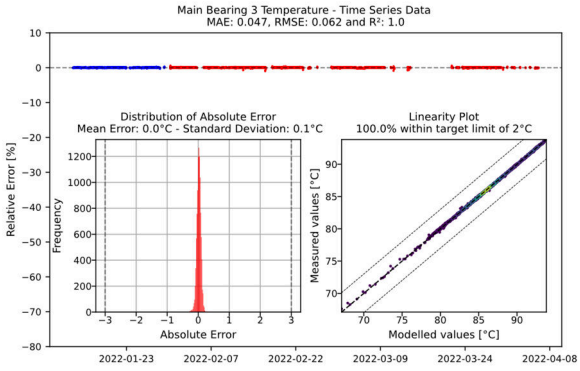

Figure 12 - Validation plot of the bearing temperature model for the main bearing temperature over a medium -term period.

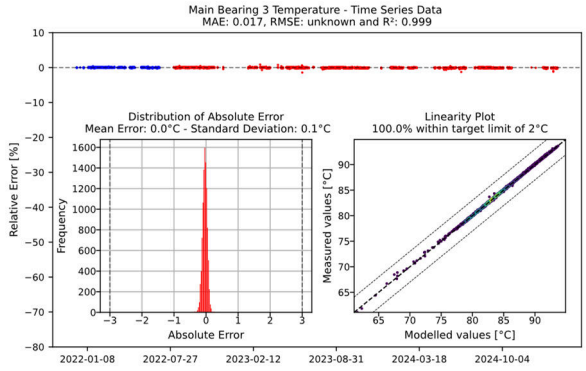

Figure 13 - Validation plot of the bearing temperature model for the main bearing temperature over a long-term period.

3.3.2 Charge Pressure Model

The second model focuses on predicting the charge air pressure after the high-pressure compressor, another critical parameter of the engine, also captured by the digital twin as previously mentioned in Chapter 2.2.

Figure 14, Figure 15 and Figure 16 show that the relative error, mean error, and standard deviation are higher in the medium- and long-term validations compared to the short-term validation; however, they remain well within the bounds of a very good model, indicating that the model's performance is stable over time. However, in the long-term validation, it is clearly observable that the charge air pressure decreases over time for a fixed engine load. This pattern suggests a gradual reduction in pressure under constant load conditions, which could point to factors such as compressor degradation, turbine degradation, fouling, or other long-term engine wear mechanisms.

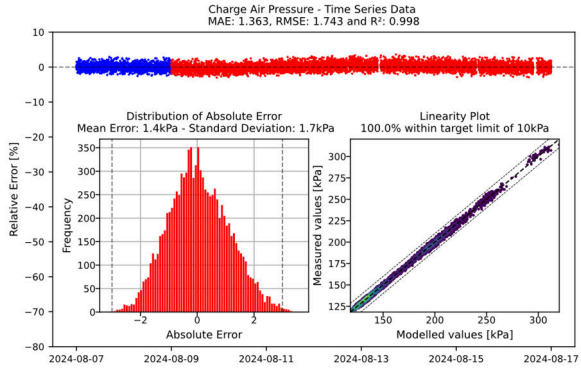

Figure 14 - Validation plot of the charge air pressure model over a short-term period.

Despite this trend, the model continues to capture the overall behavior of the engine effectively, and the observed variations in the charge air pressure provide valuable insights for further investigation into the long-term performance of the engine.

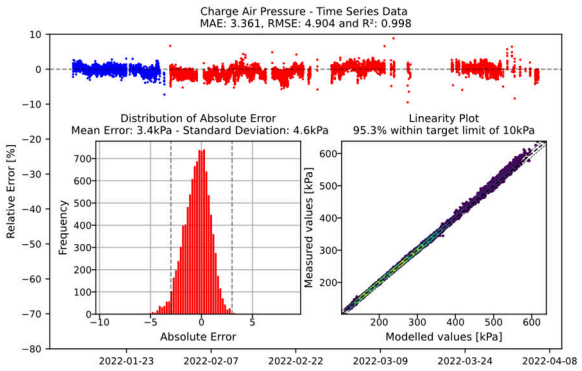

Figure 15 - Validation plot of the charge air pressure model over a medium-term period.

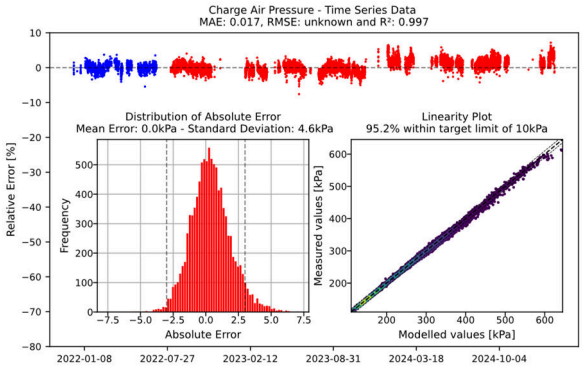

Figure 16 - Validation plot of the charge air pressure model over a long-term period.

3.4 Challenges and limitations

Data scarcity, sensor failures, and noisy data are key challenges in building accurate models. Limited or missing data can lead to overfitting or poor generalization, while sensor issues introduce inaccuracies that degrade model performance. Noice further complicates the learning process, requiring effective data preprocessing to maintain accuracy.

Model interpretability is another challenge, as many machine learning models operate as "black boxes." Lack of transparency makes it difficult to understand predictions, limiting trust and complicating troubleshooting. Ensuring model explainability is crucial for validation and regulatory compliance. This is the main reason why a regression approach is favourable.

Lastly, computational limitations pose barriers for real-time deployment. Complex models can require significant processing power, which is often unavailable in real-time systems. Optimizing model efficiency while maintaining performance is

necessary for practical deployment in time-sensitive applications.

4 HYBRID APPROACH

4.1 Introduction

Machine learning and simulation intersect in three key subfields as portrayed in Figure 17. Simulation-assisted machine learning (SAML) refers to the incorporation of simulation techniques to support and enhance machine learning models. This approach is particularly important where real-world data collection is impractical, costly, or infeasible. Simulations can generate synthetic datasets for training machine learning algorithms and enable the exploration of hypothetical scenarios beyond the limitations of empirical data. Collecting engine data is expensive, so to get the models up and running, this is a reasonable approach.

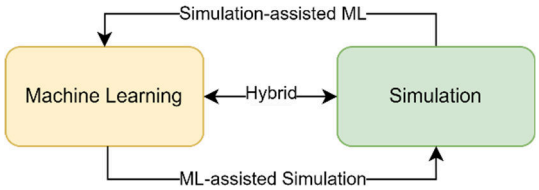

Figure 17. Combining machine learning and simulation [6].

Machine learning-assisted simulation (MLAS) involves leveraging machine learning methodologies to improve the accuracy, efficiency, and scalability of simulations. ML can serve as a surrogate model to approximate computationally expensive simulations (ID, flame speed, Combustion Event, …), optimize parameters to enhance the simulation fidelity, or reduce dimensionality to streamline complex models.

Finally, hybrid approaches represent bidirectional integration of machine learning and simulation, where both domains contribute to and refine each other in a mutually reinforcing manner.

The challenges identified in sections 2.4 and 3.4 can be addressed through either of these sub-areas. In this work specifically, we are going to explore MLAS and how it can be useful for calibration of various parameters and engine maps used in the simulation model.

4.2 MLAS for engine mapping

To address the challenges posed by diverse engine variations and real-world operating conditions, we propose a MLAS calibration approach that integrates test bench data, physics-based simulations, real-world engine telemetry, and machine learning techniques. By leveraging real-world data, we refine engine maps to better reflect operational conditions while maintaining physically interpretable constraints. This methodology combines traditional modeling techniques with data-driven corrections, ensuring enhanced model accuracy, adaptability, and predictive reliability. Figure 18 outlines a structured workflow for integrating real-world data into the calibration process, using machine learning to fine-tune maps within predefined physical limits, making the framework robust enough to accommodate a wide range of engine variations.

The first step is to create all the engine maps needed by the physics-based simulation model (i.e. the digital twin model) to process the data. The simplest maps are those of the turbines, compressors, heat exchangers, etc. These can be supplied by the manufacturer or produced from test bench data. On every engine, 160 sensors are placed strategically among the principal circuits of the engine. When the engine is factory approved according to standardised guidelines, an initial set of key performance data is created. Furthermore, 0D/1D simulations are used to fill the gaps where test data is sparse or impractical to measure: volumetric efficiency, trapping ratio, EGR, …

In a second step, once the engine is commissioned, sensor data from real-world operation conditions can be collected. This data can be used to identify deviations between the originally constructed engine maps and maps that would be developed based on the actual working regime.

This real-world data, together with the output of the simulation model and the initial engine maps will be used to train a machine learning model. The model will attempt to reduce the simulation error using engine map correction factors. To ensure safety and interpretability, adaptive models are needed to adjust the maps within predefined physical boundaries. Additionally, reinforcement learning or Bayesian optimisation techniques can be used to dynamically tune key parameters for optimal performance of the ML model.

Finally, by implementing an iterative calibration loop, updated maps can be retested and validated against any new real-world data. Automatic periodic recalibration using field data can also be added to improve the model robustness over time.

The output of the optimized simulation model will form the hearth of the CBM system. By continuously comparing the simulated results for specific operating conditions with the actual measured results, prognostics and health management algorithms will be able to detect any faults or anomalies in the system and indicate the

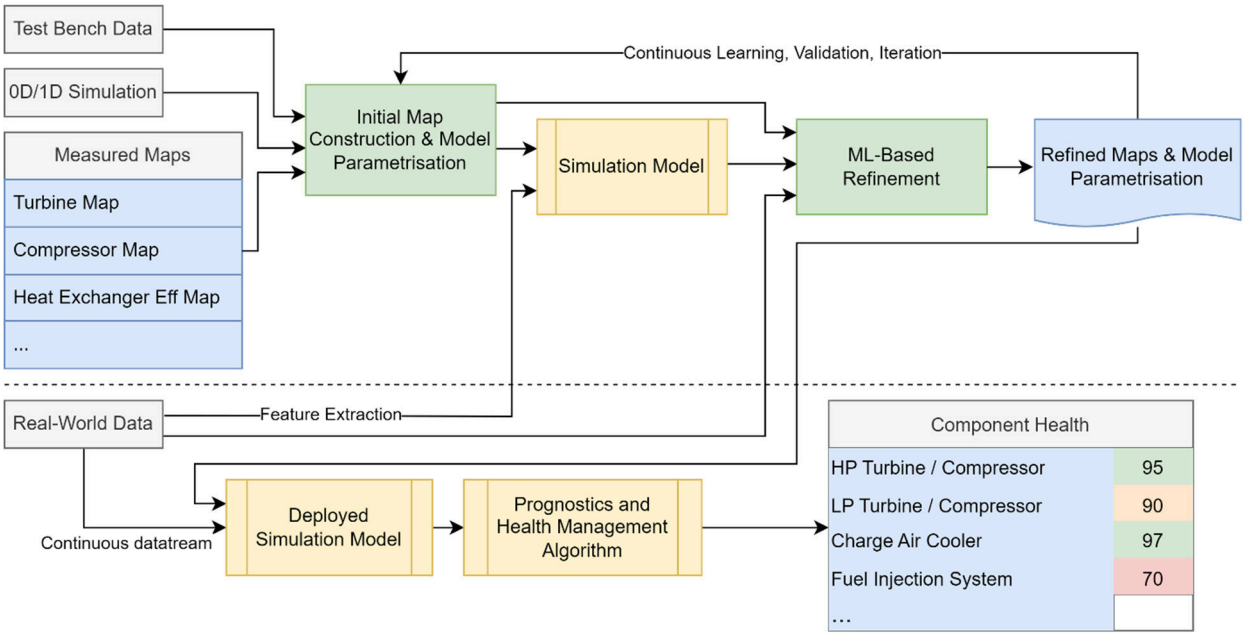

Figure 18. Flowchart of Hybrid Engine Calibration Mapping.

life expectancy of critical components based on specific component health metrics.

4.3 Results

The approach discussed above ensures that engine maps remain physically meaningful while continuously adapting to real-world conditions. It provides a structured methodology to enhance performance, efficiency, and robustness without compromising safety or interpretability.

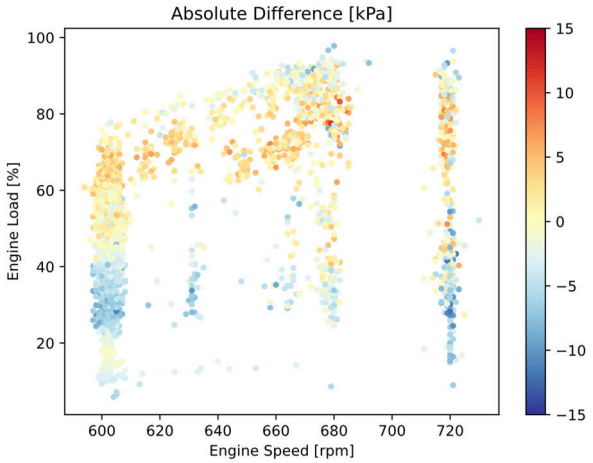

Figure 19 - Contour plot of the absolute difference between the modelled and measured Pressure after the LP Compressor.

An example of how MLAS can enhance engine mapping in the physics-based model is demonstrated through the results of the previously discussed validation of the pressure after the low-pressure compressor shown in Figure 19. The contour plot indicates that the model performs well across the entire engine operating range, remaining within an acceptable fault margin. However, a clear trend emerges: at lower loads, the model tends to underpredict the pressure, whereas at higher loads, it slightly overpredicts.

This data can be leveraged to refine the modeling maps, correcting these localized errors and improving overall model accuracy. Essentially, this process involves recalibrating the existing maps used in the physics-based simulation to better match the specific engine. This is a logical adjustment, considering that the initial simulation maps are theoretical, derived from controlled test setups under ideal conditions. In reality, manufacturing tolerances introduce slight variations between engines, making it necessary to fine-tune the maps within the bounds of physical feasibility to better represent the actual engine behavior.

4.4 Challenges & Future directions

By incorporating machine learning techniques into the physics-based model, MLAS reduces the sensitivity of the digital twin to the calibration process. However, a hybrid modelling approach also presents key challenges that need to be addressed. Effective data management is essential, as it involves integrating large streams from multiple sources, such as sensor and experimental data, while dealing with noise, missing values or limited data sets. Computational efficiency must also be balanced with model complexity, particularly when combining physics-based and data-driven approaches. Ensuring interpretability will be necessary, especially when using deep learning, with the aim of achieving

Explainable AI (XAI) in industrial applications. Additionally, maintaining and updating models is important to keep them accurate and reliable as new data or system changes emerge. By addressing these challenges, hybrid modeling can greatly enhance engine diagnostics and predictive maintenance.

As this system is further deployed, future advances in hybrid modelling for combustion engine health monitoring will use cutting-edge machine learning techniques such as deep reinforcement learning, transfer learning and federated learning to improve real-time diagnostics. A key focus will be integrating these models into real-time systems, enabling predictive maintenance and fault detection directly in operational engines. Additionally, AI-driven optimization will allow hybrid models to dynamically adjust engine parameters based on health predictions, improving efficiency and longevity. The adoption of edge computing and IoT will further enhance real-time capabilities, enabling distributed, onboard health monitoring systems that provide accurate and timely insights for engine performance management.

5 DEPLOYMENT

5.1 Motivation

Despite the challenges discussed in section 4.4, the deployment of a particular system is not generally perceived as constituting a primary challenge in the design of a platform for condition monitoring of engines. However, when the number of engines incorporated into the system exceeds a certain threshold, this rapidly becomes a bottleneck, which complicates scalability and system management.

To illustrate the scale of data generation, a typical engine is equipped with 50 - 150 sensors, each recording parameters such as temperature, pressure, and speed at a sampling frequency of 1 Hz. This results in 50 - 150 datapoints per second per engine. Over the course of a single day, this equates to 4.3 - 12.9 million data points, and over the span of a year, an individual engine generates approximately 1.58 - 4.73 billion data points. This calculation does not account for sensors with sampling frequencies of 50 kHz or higher, which generate significantly larger volumes of data independently.

As a result, the majority of development time is spent managing complex data structures and data pipelines rather than actually developing the system. The following section explains these key components that make up the system and provides a rationale for the importance of each component in ensuring an effective and scalable system. Figure 20 shows a high-level overview of the platform architecture that is currently deployed, of which the subcomponents are elaborated upon in the following sections.

5.2 Work Orchestration & Observability

In modern machine learning and data engineering pipelines, workflows often involve multiple interdependent tasks, such as data extraction, preprocessing, model training, and deployment. Managing these tasks manually or through ad hoc scripts can quickly become unmanageable, leading to inefficiencies, failures, and a lack of visibility into pipeline execution. This is where a work orchestrator becomes essential.

A work orchestrator automates, schedules, and monitors complex workflows, ensuring that tasks execute in the correct order, recover from failures, and scale efficiently. It provides dependency management, error handling, logging, and retries, reducing operational overhead and improving system reliability. Additionally, an orchestrator enables observability, offering insights into workflow performance, execution history, and data lineage, which are critical for debugging and compliance.

The work orchestrator that is chosen for this work is Dagster*,* a modern work orchestrator designed specifically for data and ML workflows [7]. Unlike traditional workflow schedulers like APACHE Airflow [8]*,* Dagster treats data assets as first-class entities, allowing for declarative pipeline definitions that promote modularity and reusability. By leveraging Dagster*,* one can achieve scalability, reproducibility, and transparency in all the ML and data pipelines.

The work orchestrator is a central component of the platform architecture, as illustrated in Figure 20 and is responsible for executing critical tasks. These include data ingestion from the cloud object store into the database, automated runs for the machine learning algorithms and digital twin, and scheduled reporting on key system metrics of the model status, database status, model calibration status. Additionally, system observability is facilitated through services like Grafana [9] and Prometheus [10], enabling real-time monitoring, alerting, and visualization of performance metrics.

5.3 Continuous integration and continuous deployment

A condition monitoring platform requires high reliability, scalability, and rapid updates to ensure accurate diagnostics and predictive maintenance. Since engines operate in dynamic environments and generate vast amounts of real-time data, the

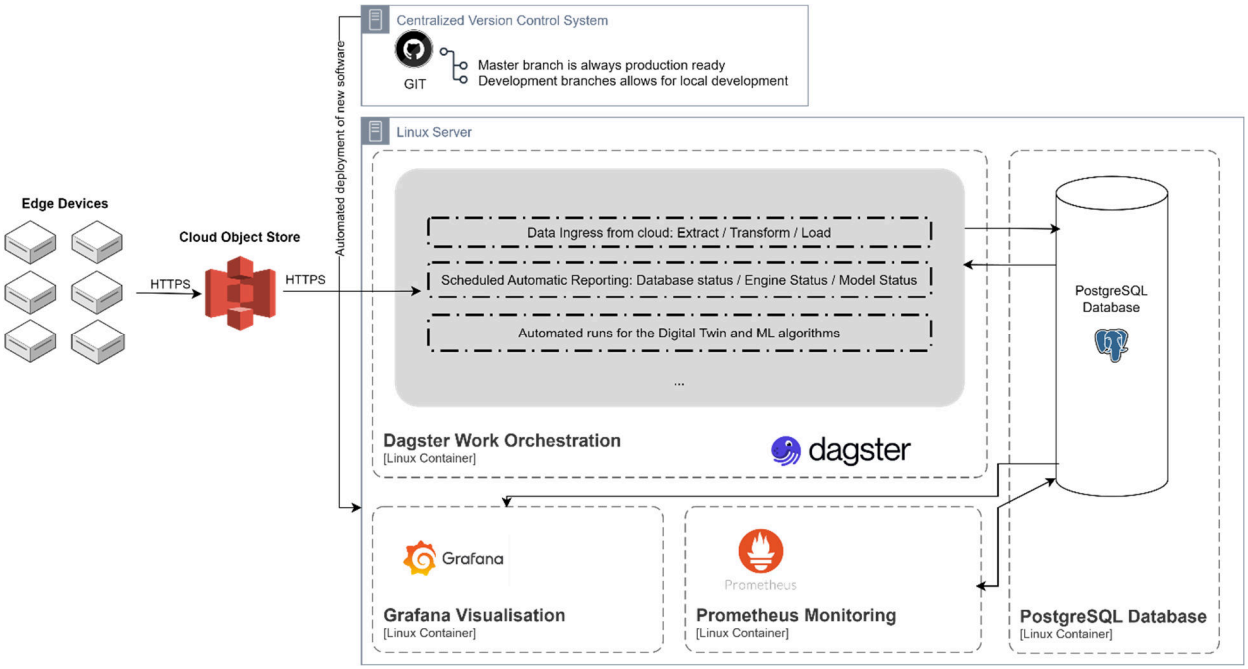

Figure 20 - Platform Architecture

platform must be able to process, analyze, and respond quickly to any anomalies.

To address this, containerization plays a critical role in ensuring the system remains efficient, scalable, and easy to manage as the number of engines grows. By encapsulating the application and its dependencies into containers, the system becomes highly portable and consistent across different environments. This enables faster deployments, easier updates, and better resource utilization, while reducing the complexity of scaling the platform to accommodate additional engines. This allows engineers to focus more on decision-making, model development, and optimization, rather than being bogged down by deployment and infrastructure concerns.

As shown in Figure 20, the entire platform architecture is designed around the use of containerized applications. This approach enables a highly dynamic environment, allowing for faster and more efficient development. By supporting seamless scaling as more engines are added, it ensures consistent performance across distributed environments.

The platform is tightly integrated with continuous integration and delivery (CI/CD) pipelines, facilitating automated updates to machine learning models, analytics algorithms, and system components without downtime. Additionally, the use of Git [11] and a local development environment significantly enhances development workflows. Git enables version control, collaboration, and code traceability, ensuring that changes can be efficiently managed, reviewed, and deployed. Combined with containerized environments, this setup allows us to test, refine, and deploy updates locally before pushing them to production, reducing errors and improving system reliability.

Additionally, containers provide a standardized runtime, eliminating deployment inconsistencies and streamlining updates across diverse hardware configurations. With this technology, we can innovate at an unprecedented pace, delivering a more robust, efficient, and intelligent condition monitoring system.

5.4 Data Management & Storage

The effective management and storage of data are of critical importance for the success of production systems. For academic purposes, data can be structured in folders and stored in CSV/text files. This approach was adopted prior to the emergence of a focus on the deployment of systems. However, as the number of vessels increased, it became evident that a structural solution was required.

Prior to the implementation of a particular technology or approach, a comprehensive delineation of the system's requirements was undertaken. The data produced by an engine is principally time-series data comprising multiple parameters that are retrieved from various systems, including the engine itself and auxiliaries. A database solution is typically required for such a dataset, and this solution must meet several key requirements:

1. High-Frequency Data Ingestion – The system must handle continuous, high-volume data streams from multiple engines and systems with minimal latency.

2. Efficient Query Performance – Fast retrieval of historical and real-time data is crucial for both the machine learning models and digital twin simulations.

3. Scalability – The storage system should efficiently scale with increasing data volume, ensuring long-term retention without performance degradation

4. Time-Series Optimization – Support for automatic partitioning, compression, and indexing to enhance read and write performance.

5. Data Integrity and Consistency – ACID compliance to ensure reliability and accuracy of the stored information

6. Integration with ML and Digital Twin Pipelines – The ability to seamlessly interact with analytics and prediction workflows.

PostgreSQL, an open source, robust, and scalable relational database system [12], is widely used for efficient data storage, retrieval, and manipulation. Combined with the TimescaleDB [13] extension, it is well-suited to meet these requirements. TimescaleDB enhances PostgreSQL by providing automatic time-series partitioning, optimized compression, and efficient indexing, allowing for scalable and performant storage of telemetry data. PostgreSQL's support for structured and semi-structured data, including JSONB, facilitates seamless integration with machine learning models and digital twin simulations. With advanced query optimization and real-time analytics capabilities, PostgreSQL and TimescaleDB ensure that vast amounts of engine data can be efficiently stored, queried, and utilized for predictive maintenance and operational insights.

Proper data management with PostgreSQL enhances model performance, reproducibility, and scalability, ultimately contributing to a robust and maintainable system.

6 CONCLUSIONS AND FUTURE WORK

- The paper describes the continuation of the development of a condition-based monitoring system that is capable of data collection, processing, aggregation, connection to the cloud, executing a Digital twin and machine learning algorithms locally on the edge device.
- Results show significant improvements in both the physics-based models and machine learning algorithms, along with extensive validation using real world data.
- Results showed that a hybrid approach towards calibration and validation of the simulation models is viable and has the potential to greatly improve the model accuracy and to decrease the effect of engine-to-engine variability on the model accuracy.
- The goal of the simulation models and machine learning algorithms is to optimise maintenance intervals and tasks, have early anomaly detection and perform remote support on (efficiency) improvements. This drives important decisions thus the accuracy of such a Digital Twin is of utmost importance in order not to draw wrong conclusions. The accuracy of the ABC Digital Twin even equals the typical sensor chain accuracy and is therefore a well-suited advisor, outperforming humans. This will be the topic of a follow-up paper.

7 DEFINITIONS, ACRONYMS, ABBREVIATIONS

CBM: Condition Based Monitoring

DT: Digital Twin

ML: Machine Learning

ABC: Anglo Belgian Corporation

PLN: Pump Line Nozzle (Fuel injection system)

CR: Common Rail

DF: Dual Fuel

PFI: Port Fuel Injection

LPC: Low Pressure Compressor

HPC: Hight Pressure Compressor

IC: Intercooler

CAC: Charge Air Cooler

P: Pressure

T: Temperature

HPT: High Pressure Turbine

LPT: Low Pressure Turbin

WG: Waste Gate

SCR: Selective Catalytic Reduction

DPF: Diesel Particulate Filter

SVM: Support Vector Machines

PCA: Principal component analysis

MLAS: Machine Learning Assisted Simulation

ID: Ignition Delay

EGR: Exhaust Gas Recirculation

XAI: Explainable Artificial Intelligence

ACID: Atomicity, Consistency, Isolation, and Durability

JSONB: JSON Binary

CI/CD: continuous integration and continuous delivery/deployment

8 BIBLIOGRAPHY

[1] R. De Graeve, K. Christianen and R. Verschaeren, “Condition-Based Monitoring for Large Bore Medium Speed Engines Using a Digital Twin, ML and Big Data,” in *CIMAC Congress*, Busan, 2023.

[2] H. Chen, K. Zhang, K. Deng and Y. Cui, “Real-time engine model development based on time complexity analysis,” *International Journal of Engine Research,* vol. 23, no. 12, pp. 2094--2104, 2022.

[3] Y. De Munck and T. Van Hauwermeiren, “PolySense,” [Online]. Available: https://www.polysense.ai/. [Accessed 04 02 2023].

[4] C. Huyen, Designing Machine Learning Systems: An Iterative Process for Production-ready Applications, O'Reilly Media, Incorporated, 2022, p. 367.

[5] Scikit-learn developers, “Linear models — Scikit-learn 1.6.1 documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html. [Accessed 26 02 2025].

[6] v. R. Laura, S. Mayer, R. Sifa, C. Bauckhage and J. Garcke, “Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions,” *Advances in Intelligent Data Analysis XVIII,* vol. 12080, 2020.

[7] Elementl, “Dagster: Data Orchestration for Machine Learning, Analytics, and ETL,” [Online]. Available: https://dagster.io/. [Accessed 26 02 2025].

[8] A. S. Foundation, “Apache Airflow: Workflow Automation Platform,” [Online]. Available: https://airflow.apache.org/. [Accessed 26 02 2025].

[9] Grafana Labs, “Grafana,” [Online]. Available: https://grafana.com/. [Accessed 27 02 2025].

[10] J. Turnbull, Monitoring with Prometheus, Turnbull Press, 2018.

[11] The Git Developer Community, “Git - Distributed Version Control System,” [Online]. Available: https://git-scm.com. [Accessed 26 02 2025].

[12] Global Development Group PostgreSQL, “PostgreSQL,” [Online]. Available: https://www.postgresql.org/. [Accessed 26 02 2025].

[13] Timescale Inc., “TimescaleDB: Scalable Time-Series Database,” [Online]. Available: https://www.timescale.com/. [Accessed 26 02 2025].

9 CONTACT

Corresponding Author:

Rik De Graeve, Anglo Belgian Corporation, Wiedauwkaai 43, 9000 Gent, Belgium.

Email: rdg@abc-engines.com