Automation of Energy Management Analysis in Automotive Engineering with Python

Larissa Pestana
UFPE
Recife, Pernambuco
0009-0009-3163-2450

Thais Cohen Costa
UFRPE
Recife, Pernambuco
0000-0003-4212-1719

Rayan Carvalho
UPE
Recife, Pernambuco
0009-0006-6872-8454

***Abstract*—This paper presents the development and implementation of an automated application for the analysis of Energy Management tests in the automotive industry, using Python as the primary tool. The application automates the processing of data acquired from various test types, including Coast Down, Parasitic Losses, Full Load, Acceleration Metrics, and Fuel Consumption. By transforming data into dataframes and utilizing Python functions, the application achieves a significant reduction in the time required for data analysis, surpassing traditional methods reliant on Excel spreadsheets. The automation improves efficiency and enhances the precision of engineering calculations, making the process more dynamic and less prone to human errors. The results demonstrate a reduction of over 70% in data analysis time, with the potential for further improvements as all engineering calculations are fully transitioned to Python. This approach offers a competitive advantage for automotive companies, optimizing their processes and accelerating product development.**

***Keywords*—Automotive Engineering, Data Analysis Automation, Test Automation.**

I. Introduction

The automation of processes in automotive engineering analysis is the key for optimizing time and reducing costs. Manual analysis can be time-consuming and prone to human error, leading to project delays and increased expenses. In a competitive industry, any delay can hinder the introduction of new products to the market and result in wasted resources.

One solution to the high cost of vehicle testing is the adoption of virtual tests. Virtual simulations provide accurate and efficient analysis, allowing engineers to test a wide variety of performance scenarios without incurring significant costs or delays. While virtual simulations can substantially reduce time and costs, it is essential to validate the virtually obtained results with physical vehicle tests after defining the engineering objectives. This ensures that different components and systems work together efficiently in the real-world context of the vehicle.

The synergy between physical and virtual tests has been a key topic of discussion at automotive testing and validation events. Regardless of the direction of these discussions, it is crucial to make physical tests more agile to accelerate vehicle development and reduce costs while ensuring the quality of the final models. To this end, we have developed technologies, automations, and software to facilitate the process. The development of this application addresses this problem within product engineering, focusing on energy management. Its proposal aligns with some previously developed and published software, which served as references in its conceptualization.

For instance, "A Python Package for Real-time CAN Data Logging, Analysis, and Visualization to Work with USB-CAN Interface" [1] and "A Customized Python Interface for Windows OS for a Low Budget USB-to-CAN-Adapter" [2] present similar proposals for developing a framework for CAN (Controller Area Network) data analysis, reception, and processing integrated with proprietary hardware structures. However, it should be noted that the application developed in this article aims at data acquisition in track tests for energy management studies. Therefore, external sensors are added to the CAN data, and known hardware solutions are utilized as cited in the Test Data Acquisition topic.

Another relevant software is the "Development of an Automotive Data Acquisition Platform for Analysis of Driving Behavior" [3]. Unlike the previous examples that focus solely on received data, this one uses the data for product development in various subareas. This reinforces the need for the explanation of engineering calculations that will be highlighted throughout the article. The main technology adopted for the development of this application is Python. The extensive set of data science libraries such as NumPy and Pandas enables efficient management of large datasets.

This work is organized as follows: Section II presents the technical Background in Automotive Energy Management. Following Section III, it starts the platform development details, in Test Data Acquisition. Section IV approaches Engineering Calculations. Section V exhibits the Interface Application. Section VI presents the Results, Section VII proposes the Discussion. Section VIII grants Artifact Availability with code availability in GitHub website. Following Section IX with Proposals for Future Research. And Section X, the Conclusion, who concludes the work, highlighting the main contributions.

II. Background in Automotive Energy Management

Energy management is a central area in automotive development, as it involves a detailed assessment of vehicle energy efficiency and performance critical aspects for meeting regulatory requirements and achieving sustainability goals. Key Energy Management tests

contributing to these evaluations include: Coast Down, which measures the vehicle's rolling resistance and aerodynamic drag; Parasitic Losses, which identifies the contribution of the transmission and brakes to rolling resistance; Full Load, which evaluates vehicle performance under full load conditions; and Acceleration Metrics, which examines vehicle dynamics across various acceleration scenarios.

In an environment where resource optimization and meeting fuel consumption targets are crucial, automating these tests not only accelerates the validation process but also reduces the risk of human error, ensuring more accurate and reliable results. Given the complexity and importance of these tests, the use of advanced technologies and automated solutions is essential for maximizing development process efficiency. In this context, this work presents a practical application of automation in the analysis of Energy Management tests, aiming to enhance the efficiency and precision of evaluations while significantly reducing the time required to obtain engineering results.

III. Test Data Acquisition

In the case study of data acquisition for track and laboratory tests, tested in real vehicles, we used three different solutions for CAN network and sensor data acquisition: National Instruments, exporting data in TDMS (Technical Data Management Streaming) format [4]; ETAS INCA, exporting data in DAT format [5]; Hottinger Baldwin Messtechnik Catman, exporting data in BIN format [6]

A. Conventional data processing

For this data to be processed by the engineering team, it must be converted into formats compatible with conventional analysis platforms, such as MATLAB and Excel. This type of analysis involves complex calculations and data manipulation.

Excel has long been a practical choice for engineering, particularly due to its lower licensing costs. However, the Excel program has limitations to handle large volumes of data which can lead to performance issues. Also the difficulty in guarantee reproducibility and version format control of analyses can compromise the efficiency and reliability of results, especially in large-scale projects or those with advanced data analysis requirements.

B. Data process by application

We developed code to handle data according to its specific format. In some tests, such as Full Load and Acceleration Metrics, acquisitions vary based on factors like accelerator pedal position, gear changes, fuel variation, and other relevant parameters. To expedite the process in our graphical user interface (GUI) platform, we implemented a strategy of selecting the directory where the acquisitions are stored. The application then lists and converts all acquisitions present in any subfolder of the directory in a single execution, simplifying and accelerating the analysis process. After conversion, these converted files should go to their destination subdirectories, which will reorganize these acquisitions depending on the type of test.

To further ensure the accuracy and consistency of the analyses, it is essential to account for variations in the time frequencies of the received data, whether from CAN signals or external sources. To maintain the reliability of the results, a unique time reference is necessary to avoid the omission of any experimental value. When data is missing at specific time points, we use linear interpolation to fill these gaps. The tests studied in this article operate with a time precision of two decimal places, which is sufficient to ensure reliable results, though the engineering team can adjust the precision as needed.

Continuing the optimization of the data analysis process, we also developed specific scripts to automate the reading and conversion of the three types of acquired data mentioned earlier. Previously, converting these acquisitions to formats like Excel involved using native tools that took, on average, 3 to 5 minutes per acquisition. With the new scripts, this conversion can now be performed in batches, depending on the type of acquisition, reducing the time to approximately 10 seconds per acquisition. These scripts were designed to deliver test results in both dataframe format and Excel files. This approach was intended to support engineers in transitioning to the new application, offering flexibility and efficiency in data processing and analysis.

1) TDMS format: The NpTDMS library, designed for reading and writing TDMS files, created and published by Adam Reeve [7], was used.

2) DAT format: The mdfreader library was used. Developed by Aymeric Rateau [8], it is a tool for manipulating MDF (Measured Data Format) files [9].

3) BIN format: CApread (Catman AP Reader) is a proprietary library for working with Catman data, created and published by Leon Bohmann [10]. The received messages are treated as channels by time frequency.

The library has internal functions that convert your data directly to various desired formats: CSV, XLSX, Pandas, MATLAB, NetCDF, HDF5 or Parquet. To ensure uniformity in data analysis processes and better integration with other parts of the project, we use the Pandas library.

IV. Engineering Calculations

Engineering calculations are tailored to each area and project, based on specific literature in domains such as aerodynamics, energy management, and losses, among others. Traditionally, this data has been handled in Excel spreadsheets, often manually. While the use of macros has streamlined parts of the process, this approach faces significant limitations, such as difficulties in handling large data volumes, a lack of reproducibility in analyses, and the complexity of maintaining and updating macros.

With the assistance of this application and the data acquisitions mentioned in the previous section organized in dataframes, calculations can be performed more quickly and

adaptively, utilizing complex mathematical equations, native functions, and efficient manipulation of data series. The application allows for precise and rapid adjustments, easily adapting to the specific needs of each project, surpassing the limitations of traditional spreadsheets and offering greater dynamism and efficiency.

V. Interface Application

To provide a good user experience, particularly for the test engineer, we developed interfaces that avoid direct interaction with the code, using JavaScript, HTML, and CSS instead of graphical tools like tkinter, which, although available in Python, has quality limitations. Figure 1 shows the interface model used in the application. The application was made available as desktop software using Flask [11], following the approach described by Bonney et al. [12] in the Data-Centric Engineering journal. Flask allows for the definition of routes to handle application pages, returning HTML files or executing operations such as data processing. The application was packaged as an executable using 'pyinstaller'.

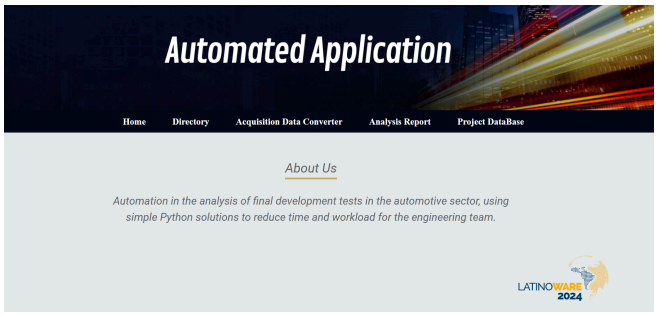

Fig. 1. Example Interface based on Application

VI. Results

To evaluate the application's efficiency, we conducted a time comparison between analyses performed with the new application and those using the traditional method. The performance analysis was divided by test type: Coast Down, Parasitic Losses, Full Load, Acceleration Metrics, and Fuel Consumption. All data processing, as outlined in the Data Process by Application section, was executed using the new application. To align with the engineering team's existing workflow, engineering calculations were integrated with Excel Macros for all tests, except for Fuel Consumption, where the analysis was fully handled by the application. This demonstrated the application's potential for even greater efficiency when fully integrated with engineering calculations across all tests.

The application produced the same engineering results as the traditional method, with identical values for CAN signals and other sensors, confirming its accuracy and reliability. The key difference between the two methods was the time required to generate the results. The application significantly reduced the time needed, making the process over 70% faster. All tests were conducted under consistent conditions, using the same computer, test engineer, and database, ensuring that the only variable was the data processing method. The table and graph below compare the time spent on each test, clearly illustrating the application's superior efficiency. The reported time values are point estimates, with a potential variability range of ±2.5 minutes and the time measure system is in 24-hour time format (hh:mm:ss).

TABEL I
TIME COMPARISON BETWEEN CONVENTIONAL ANALYSIS AND ANALYSIS BY APPLICATION

Test Type	Conventional	Application
Coast Down	01:45:00	00:40:00
Parasitic Losses	01:10:00	00:20:00
Full Load	02:30:00	00:35:00
Acceleration Metrics	02:30:00	00:40:00
Fuel Consumption	02:00:00	00:30:00

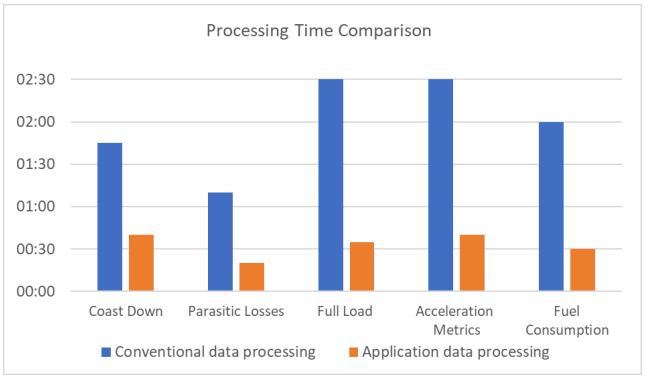

Fig. 2. Time Comparison Char - Comparative graph between Conventional data processing and Application data processing analyzing the time spent for each type of test.

VII. Discussion

The application developed in this work has proven effective in automating data analysis for Energy Management tests in the automotive sector, with enhanced efficiency achieved through the use of dataframes and Python functions, which surpass the limitations of traditional Excel spreadsheets. While engineering calculations have traditionally been performed manually or with Excel macros, facing challenges with large data volumes, the application provides a more robust and adaptable solution, allowing for quick adjustments according to project needs. The analysis of the Fuel Consumption test, conducted entirely in Python, demonstrated greater efficiency, suggesting that a complete migration of calculations to Python could yield additional benefits. The transition to this new automated tool requires adaptation and training of the engineering team, ensuring continuous validation of results compared to traditional methods, to maintain the accuracy and reliability of the analyses.

Finally, the application, initially focused on the automation of Energy Management tests, has a modular structure that allows for expansion into other testing areas within the automotive sector. Future improvements may include the automation of engineering decisions based on the generated reports, further deepening the integration into development processes.

VIII. Artifact Availability

The code from Application including *Data process by application,* interface example, and framework to export are available at https://anonymous.4open.science/r/AutomatedApplication-CEF9/README.txt

IX. Proposals for Future Research

The solution provided by this application encompasses the entire process, from test execution to report generation; however, it currently lacks automation for making engineering decisions based on the analysis of these reports. A proposed enhancement is to integrate this software with the "Software Integration Test Report Analysis Automation Using Python" [13], developed by Paigude, Gajul, Mishra, and Katkar, and presented at the 2021 ASIANCON. Due to the structural similarities and shared programming language, combining these solutions could further reduce the workload for engineers managing Energy Management tests.

X. Conclusions

The results demonstrate the benefits of automation in automotive engineering analysis, highlighting a significant reduction in the time required to deliver engineering results. The developed application was able to produce the same results as the traditional methodology, but with a more than 70% decrease in data analysis time. Furthermore, this reduction in analysis time is expected to be even greater when all engineering calculations are performed directly in Python, without the need for integrations, as was the case with the Fuel Consumption test.

The consistent operation of the platform, following standardized procedures at all stages of analysis, improves efficiency and accelerates the development process. This represents a competitive advantage for automotive companies seeking to optimize their processes and stay ahead in the market. The application provides an efficient and flexible tool for automotive engineers, contributing to cost reduction and faster development of high-quality products.

1 Acknowledgments

This work was conducted by engineers from a Private Automotive Company, and we would like to extend our acknowledgment for their contributions.

2 References

[1] R. Bhadani et al., "Strym: A Python Package for Real-time CAN Data Logging, Analysis and Visualization to Work with USB-CAN Interface," 2022 *2nd Workshop on Data-Driven and Intelligent Cyber-Physical Systems for Smart Cities* Workshop (DI-CPS), Milan, Italy, 2022, pp. 14-23, doi: 10.1109/DI-CPS56137.2022.00009.

[2] D. Wetzel, A. Reindl, H. Meier, M. Niemetz and M. Farmbauer, "A Customized Python Interface for Windows OS for a Low Budget USB-to-CAN-Adapter," 2022 *International Conference on Electrical*, Computer and Energy Technologies (ICECET), Prague, Czech Republic, 2022, pp. 1-5, doi: 10.1109/ICECET55527.2022.9872574.

[3] G. Andria, F. Attivissimo, A. Di Nisio, A. M. L. Lanzolla, and A. Pellegrino, "Development of an automotive data acquisition platform for analysis of driving behavior," Measurement, vol. 93, pp. 278-287, 2016. doi: 10.1016/j.measurement.2016.07.035

[4] "TDMS File Format Internal Structure" National Instruments, 21 de setembro de 2022. Available: https://www.ni.com/en/support/documentation/supplemental/07/tdms-file-format-internal-structure.html.

[5] INCA V7.5 | Getting Started R01 EN | 03.2024," ETAS GmbH, 2024.

[6] DATA SHEET catman: Universal data acquisition and analysis software," Hottinger Brüel & Kjaer GmbH, Darmstadt, Germany, Jul. 7, 2023

[7] A. Reeve, npTDMS, stable version. Available: https://github.com/adamreeve/npTDMS.

[8] A. Rateau, "mdfreader: A library for reading MDF (Measurement Data Format) files, version 0.1.8," PyPI, Sep. 16, 2017. Available: https://github.com/ratal/mdfreader.

[9] MDF Big Data Support," ETAS GmbH, versão V0.5 R01 EN, Stuttgart, Germany, Nov. 2020.

[10] L. Bohmann and H. Line, APReader, versão 1.1.2, Zenodo, 2024. doi: 10.5281/zenodo.8369804. Available: https://github.com/leonbohmann/APReader/compare/v1.1.1...v1.1.2.

[11] A. Ronacher, Flask, versão 2.0.1, 21 de maio de 2021. Available: https://github.com/pallets/flask.

[12] M. S. Bonney, M. de Angelis, M. Dal Borgo, L. Andrade, S. Beregi, N. Jamia, and D. J. Wagg, "Development of a digital twin operational platform using Python Flask" Data-Centric Engineering, vol. 3, p. e1, Jan. 2022. doi: 10.1017/dce.2022.1.

[13] P. Paigude, V. Gajul, J. Mishra, e S. Katkar, "Software Integration Test Report Analysis Automation Using Python" in Proceedings of the 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 27-29 August 2021, ISBN:978-1-7281-8402-9,doi: 10.1109/ASIANCON51346.2021.9544984.