Optimization of Thermal Analysis Postprocessing Through Workflow Automation

Author: Jan D. Landrón Camacho
Advisor: Héctor J. Cruzado, Ph.D.
Master in Engineering Management
Graduate Project EXPO, May 2026

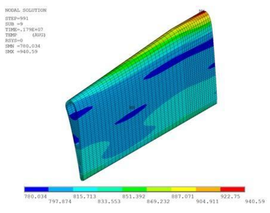
Abstract

Thermal analysis is widely used in aerospace engineering to evaluate the thermal performance of propulsion system components under different operating conditions. However, postprocessing simulation results often requires repetitive manual effort involving file transfers, data extraction, plot generation, file organization, and report preparation. This project evaluated the existing postprocessing workflow using a representative 40-mission case study to identify workflow bottlenecks and estimate processing inefficiencies. An automated workflow using Linux batch processing, Python scripting, and VBA report automation was developed. The proposed workflow reduced repetitive manual interaction, improved report consistency, and reduced the estimated processing time from approximately 21 days to about 5 days.

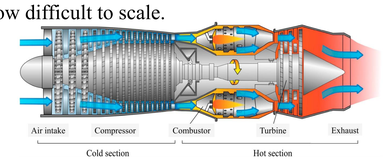
Introduction

Modern thermal analysis workflows generate large mission datasets that require extensive postprocessing activities after simulations are completed.

However, postprocessing simulation results often requires repetitive manual activities such as data extraction, file organization, plot generation, and report preparation.



When multiple missions are processed, these repeated tasks increase turnaround time and make the workflow difficult to scale.



The objective of this project was to develop an automated workflow framework to reduce postprocessing time, improve consistency, and minimize repetitive manual effort through workflow automation.

Literature Review

Thermal analysis supports engineering decision-making by converting simulation results into meaningful engineering data. However, large simulation datasets can make postprocessing activities time-consuming when extraction, visualization, and reporting tasks require repeated user interaction. Programming and automation tools can improve workflow efficiency by reducing repetitive manual activities and standardizing results across multiple missions. Python supports engineering data processing and workflow automation, Linux environments support scripting and batch processing, and VBA automation can improve consistency in report generation activities [1][2][3].

Methodology

The project methodology focused on evaluating the existing thermal analysis postprocessing workflow, identifying repetitive manual activities, and developing an automated workflow framework using a representative 40-mission case study.

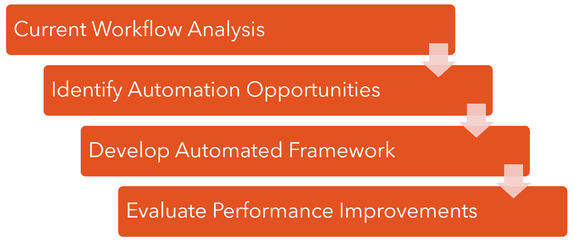


Figure 1: Project Methodology

The representative 40-mission case study was used to estimate processing time distribution across the major postprocessing activities and compare the manual workflow against the proposed automated workflow.

Results

➢ Current Manual Workflow

The existing workflow required repetitive manual interaction for file transfer, data extraction, file organization, plot generation, and report preparation activities. Many activities were repeated individually for each mission package, increasing processing time and reducing workflow consistency.

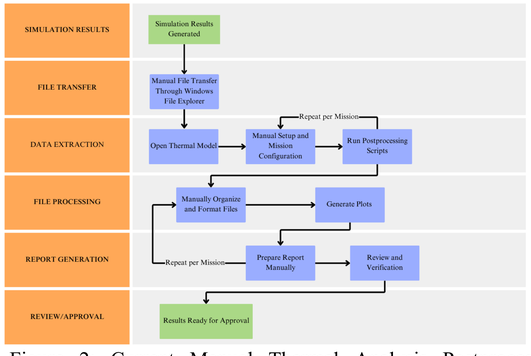


Figure 2: Current Manual Thermal Analysis Postprocessing Workflow

➢ Time Distribution Analysis

Table 1 summarizes the primary workflow tasks and the bottlenecks identified within the manual postprocessing workflow. Figure 3 presents the estimated time distribution for the representative 40-mission case study, highlighting that data extraction and file formatting represented the largest portions of the total processing time.

Table 1: Tasks Targeted for Automation in the Postprocessing Workflow

Task	Task Description	Bottleneck Identified
File Transfer	Transfer simulation results to HPC environment	Manual Windows file transfer slows large datasets
Data Extraction	Execute postprocessing extraction tools	Manual setup and execution required per mission
File Formatting and Organization	Organize and rename output files	Manual formatting creates inconsistencies
Plot Generation	Generate transient temperature plots	Repetitive plot generation across missions
Report Generation	Create standardized PowerPoint reports	Manual formatting and report compilation

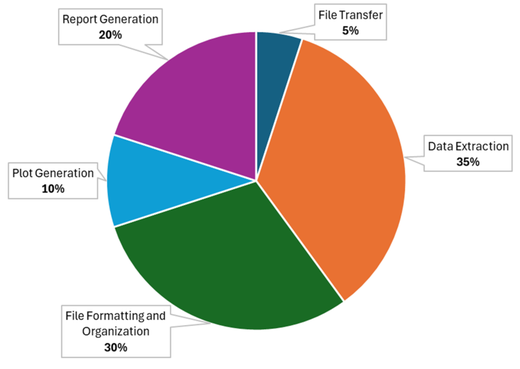


Figure 3: Time Distribution of Manual Postprocessing Tasks for a 40-Mission Package

➢ Proposed Automated Workflow

Figure 4 presents the proposed automated workflow developed to reduce repetitive manual interaction and improve workflow efficiency. The automated workflow integrated Linux batch processing, Python scripting, and VBA automation into a standardized process for data extraction, file organization, plot generation, and report preparation activities.

The automated workflow introduced a one-time template generation and setup activity that replaced many repetitive formatting and plotting tasks previously repeated throughout the workflow.

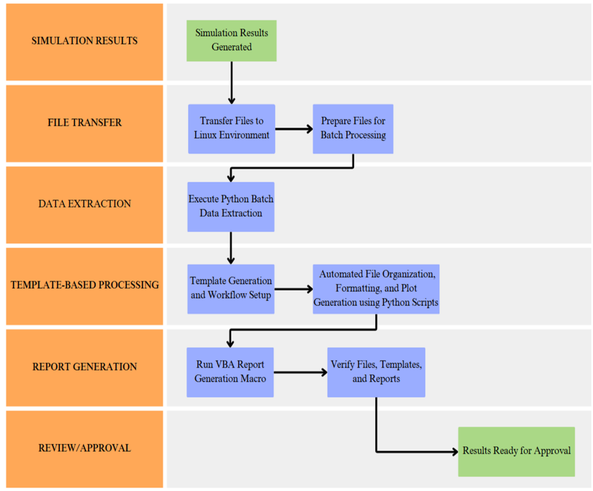


Figure 4: Proposed Automated Thermal Analysis Postprocessing Workflow

➢ Workflow Performance Improvements

- 21 Days → 5 Days Estimated Processing Time
- ~76% Reduction Postprocessing Time
- ~90% Reduction Manual Data Extraction Effort

The proposed automated workflow reduced repetitive manual interaction by standardizing extraction, file organization, plot generation, and report preparation activities across multiple mission packages.

Conclusions

This project evaluated the existing thermal analysis postprocessing workflow and identified repetitive tasks that contributed to workflow inefficiencies. The analysis showed that much of the total processing time was concentrated in manual data extraction, file organization, plot generation, and report preparation activities. To address these issues, an automated workflow using Linux batch processing, Python scripting, and VBA automation was developed to standardize postprocessing activities and reduce repetitive manual interaction. The proposed workflow reduced estimated processing time from approximately 21 days to about 5 days while improving workflow consistency and scalability.

References

[1] Oliphant, T. E. (2007). Python for scientific computing. Computing in Science & Engineering, 9(3), 10–20. Available at: https://www.researchgate.net/profile/Travis-Oliphant/publication/3422935_Python_for_Scientific_Computing/links/548f0b680cf225bf66a7f9c3/Python-for-Scientific-Computing.pdf

[2] Nemeth, E., Snyder, G., Hein, T. R., & Whaley, B. (2017). UNIX and Linux System Administration Handbook (5th ed.). Addison-Wesley.

[3] Walkenbach, J. (2015). Excel VBA Programming for Dummies (4th ed.). Wiley.