Development of a FLUENT-Based Thermal-Hydraulic Analysis Code Framework for Steam Generators using pyAnsys

Bumjin Cho[a], Jinsu Kim[a], Giwon Bae[b], Hyoungkyu Cho[b] and Minseop Song[a]*
[a] *Department of Nuclear Engineering, Hanyang University, 222 Wangsimri-ro, Seongdong-gu, Seoul, Korea*
[b] *Department of Nuclear Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, Korea*
**Corresponding author: hysms@hanyang.ac.kr*

***Keywords: Steam Generator, pyAnsys, Thermal-Fluid Analysis, Code Framework, ANSYS FLUENT**

1. Introduction

Currently, the nuclear industry still relies on legacy FORTRAN-based thermal–hydraulic analysis codes developed in the 20th century to evaluate steam generators and various reactor components. Codes such as RELAP5 and MARS-KS adopt lumped-parameter or channel-based models to represent steam generator components and simulate two-phase flow phenomena. In these models, the governing equations for mass, momentum, and energy are solved numerically to predict key parameters including pressure, temperature, and void fraction [1, 2].

However, the lumped-parameter approach inherently limits the ability to resolve detailed local flow structures and complex interactions within the steam generator. This reduced spatial resolution makes it difficult to assess local phenomena such as tube degradation and flow-induced vibration accurately. In addition, the legacy FORTRAN implementations hinder accessibility and automation for repetitive analyses. These limitations highlight the need for modern thermal–hydraulic codes that can perform high-fidelity analyses of local variables like temperature, flow rate, and void fraction, ultimately enhancing plant safety and performance.

To address these demands, there is growing interest in adopting modern Computational Fluid Dynamics (CFD) techniques. A CFD-based framework offers advanced modeling capabilities, high computational efficiency, scalability, and flexible post-processing. These features enable more precise and rapid simulations, and CFD is expected to overcome the limitations of legacy technical codes by effectively solving complex thermal–hydraulic problems.

This study proposes a FLUENT-based thermal–hydraulic simulation code framework that addresses the shortcomings of legacy technical codes. A key library used to build the framework is pyAnsys. PyAnsys is a Python-based library developed for the seamless integration and automation of ANSYS software [3]. With pyAnsys, various simulation tools can be integrated and controlled through Python scripts, enhancing the versatility and scalability of the analyses. It also offers considerable advantages in automated parameter optimization and repetitive analyses. Moreover, as a FLUENT-based code, it incorporates the reliability of the existing FLUENT software in the analysis. In particular, the focus on steam generator analysis is motivated by their role as critical components in nuclear power plants, where detailed evaluation of complex two-phase flow and heat transfer phenomena is essential. Furthermore, the steam generator serves as an example to verify the applicability of this code framework. Future research will focus on implementing and integrating all components. A dedicated framework tailored for steam generators helps avoid the potential confusion caused by FLUENT's extensive features that are not specifically required for this application. For example, the U.S. NRC has conducted nuclear reactor thermal–hydraulic analyses using FLUENT. The FLUENT-based thermal–hydraulic simulation code for steam generators, currently under development using pyAnsys, aims to maintain the macroscopic trends observed in conventional FORTRAN-based codes while incorporating the unique microscopic insights and reliability inherent in modern CFD techniques.

In various fields, including biomedical engineering, a semi-automated parametric workflow has been developed using pyAnsys for personalized, patient-specific modeling and simulation [4]. Additionally, in the field of engineering education, an educational approach has been implemented to enhance the skills of undergraduate students by increasing accessibility to ANSYS software through Python scripts [5].

Based on diverse pyAnsys applications, it is anticipated that its use in nuclear engineering, particularly for the parametric analysis of steam generators, will enhance the design process by improving both accessibility and reliability through overall ANSYS analysis. In this study, development has progressed through the implementation and initial validation of various models, including test, case, GUI, and conceptual problem models, thereby demonstrating the potential of the proposed code framework and establishing a foundation for future full system integration. Additionally, the primary objective is to complete the PyAnsys library analysis framework and integrate it with a Python-based machine learning library to create a comprehensive process for data generation, model training, and analysis, which is a critical step in developing ML models as alternatives to 3D CFD simulations.

2. Methods

In this section, we describe the methodology and essential requirements for developing a FLUENT-based thermal–hydraulic analysis code framework using pyAnsys. As summarized in Figure 1, the proposed framework comprises sequential steps of steam generator geometry generation, meshing, solver setup, calculation, and verification of result data. Each step is automated through the integration of ANSYS software and Python scripts. In addition, the suitability of the framework for developing thermal–hydraulic analysis code is verified in terms of integration, consistency, scalability, User-Defined Function (UDF) applicability, and reliability.

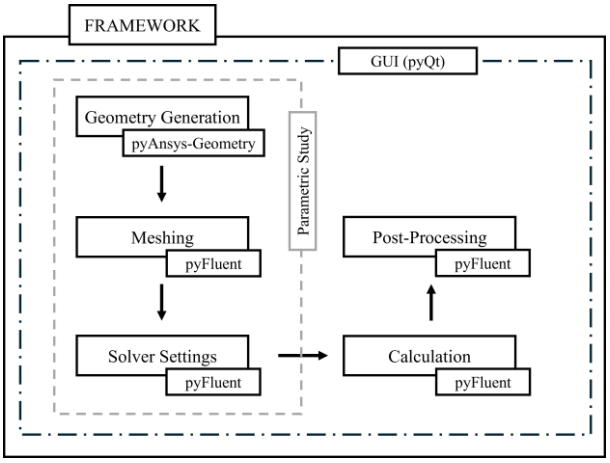

Fig. 1. Code Framework Diagram

2.1. Workflow for Framework

Thermal–hydraulic analysis codes follow the processes of steam generator geometry generation, meshing, solver setup, calculation, and verification of result data, as demonstrated by the legacy codes. In contrast to legacy codes, the present study leverages the high readability and user–friendliness of Python to interface with ANSYS software in the form of libraries. By employing the pyAnsys library package provided by ANSYS, the proven computational capabilities of FLUENT are directly incorporated to ensure the reliability of the analysis results.

The steam generator geometry is generated using the pyAnsys–Geometry library within the ANSYS SpaceClaim environment. Creating the steam generator geometry requires ANSYS version 24R2. It follows a conventional bottom-up approach that starts with a work plane setup and then defines points, lines, surfaces, and bodies. In this process, naming operations are performed for each face and body to enable effective identification of specific regions within the Python scripts. The generated geometry is saved in the SpaceClaim file format and passed to the subsequent meshing step.

In the meshing step, the pyFluent-meshing mode is utilized to generate a mesh based on the watertight geometry type. At each step, an Application Programming Interface (API) call is used to convert FLUENT's Text User Interface (TUI) command into Python script format, thereby ensuring that the meshed domain is seamlessly switched into solver mode.

The solver setup, calculation, and verification of result data steps are performed using pyFluent-solver mode. Rather than relying on FLUENT's Graphic User Interface (GUI) – based procedures, TUI commands are converted into Python scripts to implement the optimization and automation of parameters such as initial and boundary conditions. By converting the symbolic representations in FLUENT's journal files as summarized in Table I into Python scripts, a consistent script–based analysis environment is established.

Table I: Mapping of TUI Commands to Python Scripts

TUI	Python	TUI	Python
/	.	Text	'Text'
-	_	?	None

2.2. Verification

As this study represents the first development of a FLUENT–based thermal–hydraulic analysis code framework using pyAnsys within a Python environment, the suitability of pyAnsys for thermal–hydraulic code development was verified based on the following criteria.

First, with respect to integration, the framework was evaluated to determine whether the entire steam generator analysis process can be executed sequentially within a single workflow. The approach of performing individual processes separately and subsequently integrating them was deemed undesirable due to increased complexity and reduced usability.

Second, in terms of consistency, we verified that the entire analysis setup and execution can be performed solely using Python scripts without the need for manual software operations or specific command interventions.

Third, the scalability of the framework was assessed by evaluating its potential for integration with other Python libraries and systems, such as pyQt and pyvista, as well as its suitability for advanced research through integration with deep learning libraries (e.g., tensorflow, scikit–learn).

In addition, because the empirical correlations used in this study are derived from well-established and widely validated formulations in the literature, we placed particular emphasis on verifying the applicability of UDFs for implementing these correlations within the FLUENT environment [6]. The processes for creating, compiling, and applying the UDFs were systematically evaluated to verify whether FLUENT can be successfully controlled via Python scripts.

Finally, by comparing the results obtained from the script-based automation with those from the GUI approach under identical analysis conditions, the impact of minor discrepancies or input errors arising during automation on the final results was analyzed, thereby ensuring the overall reliability of the methodology.

3. Results

This section presents the outcomes of the validation process for developing a thermal-hydraulic analysis code framework using pyAnsys with FLUENT. Four evaluation models were constructed to assess the framework's integration, consistency, scalability, UDF applicability, and reliability. The evaluation models include the test model, case model, GUI model, and conceptual problem model.

3.1. Test Model

To verify the feasibility of the basic framework, we developed a test model using the simplest flow case, namely the internal flow analysis of a pipe, as a basic example. This model uses a single Python script to control the entire process from geometry generation to result verification. It also simulates the internal flow in a simple pipe domain. Integration was confirmed by executing the sequential workflow as described in Figure 1. In this model, users specify geometry parameters, such as diameter and total length, via a command line interface. These inputs are extruded to form the pipe domain, which is then visualized in real time using the pyvista library. The geometry is subsequently saved in scdocx format and exported to pyFluent. Figure 2 illustrates both the parametric input process and the rendered pipe geometry. Throughout these steps, no performance bottlenecks or development issues were encountered. This model verified that the framework can successfully perform a complete flow analysis on a simple domain, thereby laying the groundwork for its application to steam generator geometries.

3.2. Case Model

The case model extends the test model by incorporating a steam generator geometry into the simulation domain. This model is significant because it tests the framework's ability to handle complex geometries and a wider range of simulation conditions. Creation of the steam generator geometry requires ANSYS version 24R2 or later. This is because the necessary sweep functions are available only from this version onward. Integration, consistency, and scalability were verified in this model, which employs parameterized geometry generation. Figure 3 shows the rendered shroud of the steam generator geometry, produced via the pyvista library. In addition, the activation of porous media features within FLUENT was confirmed, ensuring that the model maintains consistency with established thermal-hydraulic analyses.

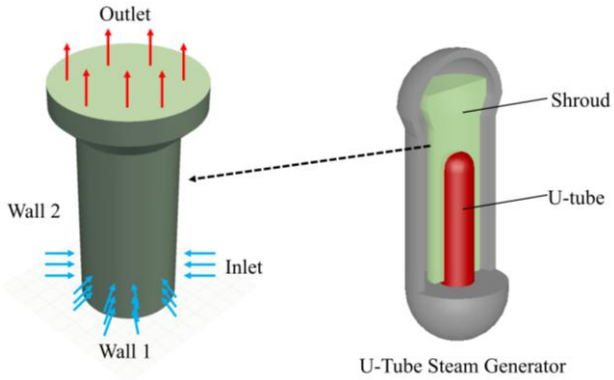

Fig. 3. U-Tube Steam Generator Geometry and feedwater flow in the shroud

3.3. GUI Model

The independently developed GUI model demonstrates that the entire simulation process can be controlled using a GUI built with pyQt and pyvista. This interface covers tasks from steam generator parameter input to meshing and solver settings. In this model, input values from the pyQt-based GUI are

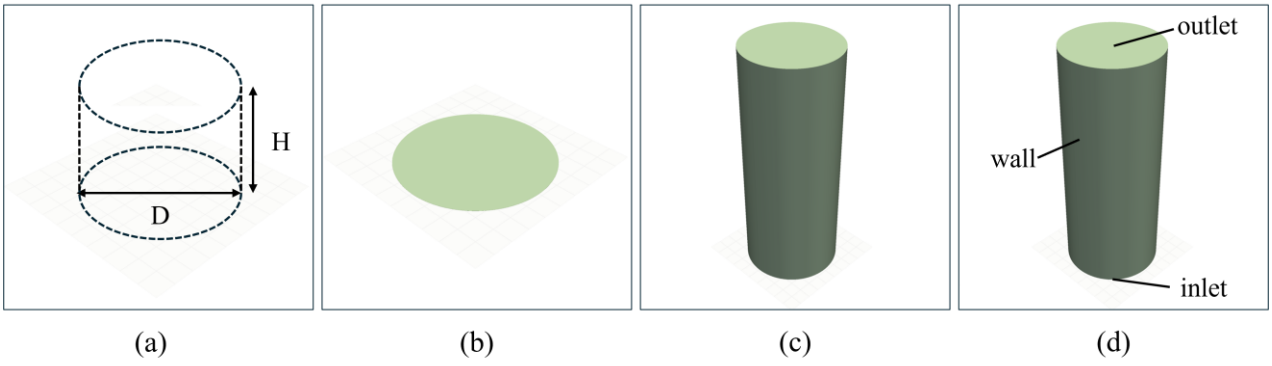

Fig. 2. Process of Test-Geometry Generation : (a) Definition of Geometry Parameter, (b) Creation of Bottom Circle, (c) Extrusion of Bottom Circle and (d) Boundary Assignment

passed as parameters to the Python script, and the simulation progress is visualized in real time using the pyvista library. Figure 4 presents the prototype GUI. Prototype tests confirmed that each step of the process was executed correctly and that output files were generated and stored as expected. The GUI represents an advancement over legacy systems by enhancing user friendliness and providing real time visualization of the analysis.

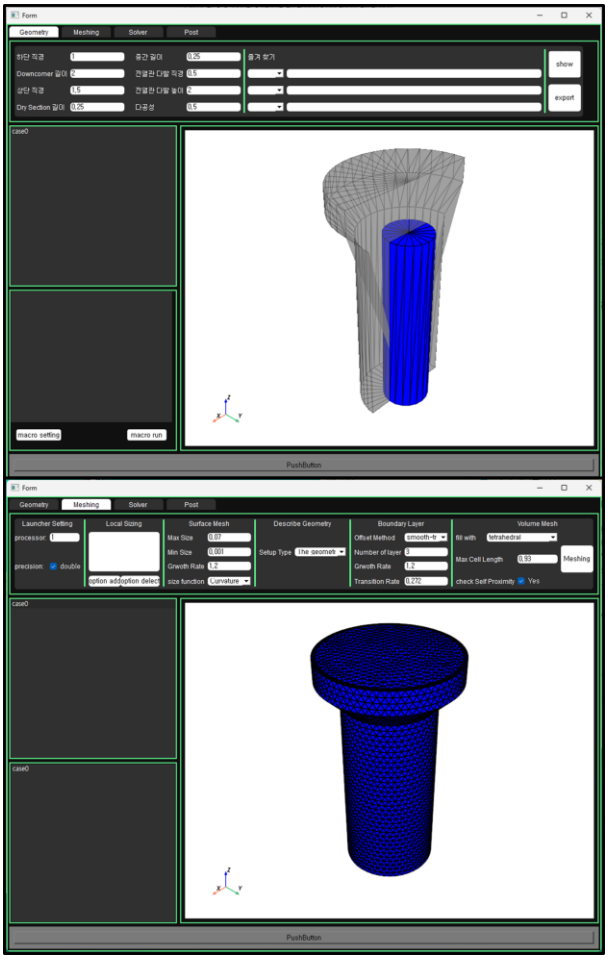

Fig. 4. pyQt and pyvista-based Prototype GUI

3.4. Conceptual Problem Model

The conceptual problem model serves as the final evaluation case, focusing on the applicability of UDFs and the overall reliability of the framework. Although the underlying algorithms remain the same, minor discrepancies in settings or input errors can occur during automated, script-based execution. To address this issue, identical simulation conditions were applied to both the script-based approach and the conventional GUI-based approach. In this model, correlations based on widely validated formulas were implemented as UDFs, compiled, and applied at the solver setup step. By successfully porting the required UDFs to FLUENT using pyAnsys, we confirmed Python's potential to provide a highly scalable correlation database.

Two independent researchers performed the calculations separately, one using Python scripts and the other using the FLUENT GUI. Table Ⅱ summarizes the turbulence model, initial conditions, boundary conditions, and multiphase model used in the calculations. The identical UDF, compiled and applied in each case, implements various source terms and a drift flux model to simulate multiphase flow and heat transfer. Figure 5 illustrates the computational domain in (a) and the corresponding contour data from the pyAnsys environment (b) and the FLUENT GUI approach (c).

Table II: CFD Setup Parameters for Conceptual Problem

Parameter	Value [Unit]
Inlet Velocity	Laminar
Inlet Velocity	0.5 [m/s]
Temperature	530 [K]
Outlet Gauge Pressure	6.8948 [MPa]
Multiphase Model	Mixture

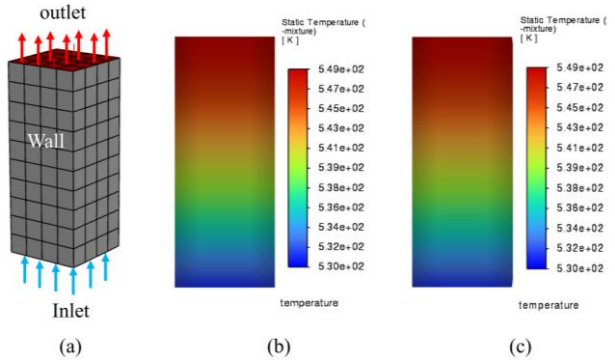

Fig. 5. (a) Computational domain for the conceptual problem model, and the resulting temperature contours obtained using (b) pyAnsys and (c) the FLUENT GUI

Figure 6 presents the average temperature distribution with respect to domain height. The identical results from both methods indicate that FLUENT's UDF implementation compiles successfully within the Python script environment and that the computational setup and performance are consistently reproduced. This confirms the reliability of the automated framework.

4. Conclusion

In this study, we proposed a FLUENT-based thermal–hydraulic analysis code framework designed to overcome the limitations of conventional technical code used for simulating internal flow phenomena within steam generators of nuclear power plants. By leveraging pyAnsys, the proposed framework integrates ANSYS software with Python scripts to automate the entire process from geometry generation and meshing to solver setup, calculation, and verification of result data.

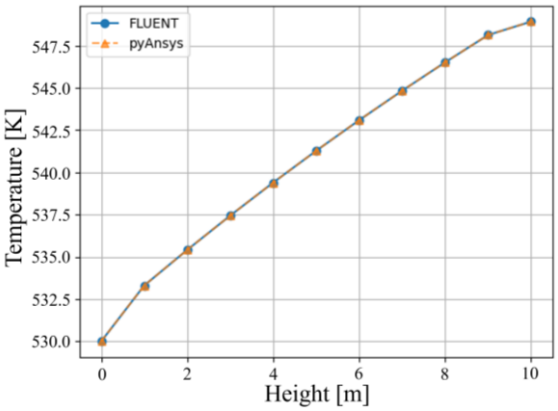

Fig. 6. Comparison between FLUENT GUI and pyAnsys

The results indicate that the framework successfully replicates the macroscopic trends of legacy technical code while providing enhanced microscopic insights and improved reliability through CFD-based methods. It is noteworthy that, although FLUENT can handle a wide range of nuclear system applications, its extensive feature set can sometimes introduce unnecessary complexity in focused steam generator analysis; the proposed Python-based framework streamlines simulation setup and computation while facilitating advanced automation and optimization.

Notably, the implementation of a Python script-based workflow and a user-friendly GUI improves the accessibility and efficiency of the simulation process. Validation through multiple evaluation models, namely the test, case, GUI, and conceptual problem models, confirmed the framework's integration, consistency, scalability, and reliability. Furthermore, the successful implementation of correlations as UDFs within the FLUENT environment demonstrates that the proposed framework can maintain the key features of existing codes while enabling automated parameter optimization and real-time visualization. In addition, this study explores whether the pyAnsys-based framework can function as an integrated model, offering a unified approach to thermal–hydraulic analysis with considerable promise for design optimization of heat exchangers and various other devices.

In the future, our primary objective is to develop a prototype code that encompasses all the analysis functionalities provided by legacy technical codes such as RELAP5 and MARS-KS. The entire process will be continuously verified against the conventional code, and through ongoing collaboration among research groups, we plan to regularly review the performance and execution to accelerate the development pace.

Acknowledgements

This research was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. GTL24031-000).

REFERENCES

[1] The Thermal Hydraulics Group, "RELAP5/MOD3 Code Manual Volume I: Code Structure, System Models, and Solution Methods", RELAP5/MOD3.2.2 Beta, NUREG/CR-5535, Scientech, Inc., Idaho Falls, Idaho, March 1998.
[2] Korea Atomic Energy Research Institute (KAERI), "MARS Code Manual," KAERI/TR-3872/2009, Daejeon, Korea, 2009.
[3] PyAnsys Documentation, "User Guide," [Online]. Available:https://docs.pyansys.com/version/dev/user_guide.html., Accessed: Mar. 8, 2025.
[4] J. Gosez, K. El Houari, S. Collin, G. Harika-Germaneau, A. Germaneau, and N. Jaafari, "Brainstim, a tool for automated personalized tDCS computational modeling based on PyAnsys," *Brain Stimulation*, vol. 16, no. 1, p. 269, Jan.–Feb. 2023.
[5] S. Cooke, S. Coleman, and J. Derrick, "Exploring the potential for scripting with simulation in engineering education – Practical examples using Python and Ansys," in *Proceedings of the SEFI Annual Conference 2023*, TU Dublin, Ireland, Sep. 11–14, 2023.
[6] R. M. C. So and C. G. Speziale, "A review of turbulent heat transfer modeling," *Annual Review of Heat Transfer*, vol. 10, no. 1, pp. 70–101, Jan. 1999.