Fine-Tuned Large Language Model for MELCOR No-Coding: the FLAMENCO Project

Dosu Park[a], Kiho Kim[b], Sooyoung Lee[b], Joongoo Jeon[c]*
[a]Department of Quantum System Engineering, Jeonbuk National University, 567 Baekje-daero, Jeonju, 54896, Republic of Korea
[b]Department of Mechanical Engineering, Chung-Ang University, 84 Heukseok-ro, Seoul, 06974, Republic of Korea
[c]Division of Advanced Nuclear Engineering, Pohang University of Science and Technology (POSTECH), 77 Cheongam-ro, Nam-gu, Pohang, 37673, Republic of Korea
**Corresponding author: jgjeon41@postech.ac.kr*

***Keywords :** MELCOR, reactor system code, artificial intelligence, large-language model, automated input generator

1. Introduction

Severe accident (SA) analyses have become increasingly important since the Fukushima Daiichi accident in 2011, as the need to prepare for beyond-design-basis accidents (BDBAs) has been emphasized worldwide. In Korea, the 2015 revision of the Nuclear Safety Act institutionalized the requirement that nuclear power plant operators submit an Accident Management Plan (AMP) including severe accident measures, which has further increased the demand for systematic analyses and verification of diverse severe accident scenarios.

Severe accident analyses rely on system codes such as MELCOR and MAAP, and therefore require accurate preparation of input files that reflect the target plant system and accident scenario. However, input development is a complex, rule-based task that must simultaneously satisfy package-specific card requirements, numbering and cross-referencing consistency, and physical constraints on key variables. As a result, manually prepared inputs are prone to user-induced errors ("User Effect"), including missing cards, incorrect parameter values, and inconsistent numbering and references [1].

To mitigate such user-induced errors in input preparation and to improve the efficiency of repetitive input development, LLM-based approaches for automatic generation of simulation codes and input files have recently attracted significant attention in various engineering fields. In general, LLM-based methods can be formulated as a conditional generation problem, where executable code or structured text is generated from user-provided natural language specifications, and prior studies have reported improvements in coding efficiency as well as the consistency of generated outputs [2].

However, generating simulation input files for nuclear applications is more constrained than general-purpose code generation, because it must simultaneously satisfy strict formatting rules, structural validity, physical constraints on key variables, and ordering/reference consistency. These requirements become increasingly critical as the problem is extended to more complex, coupled scenarios, making reliability a central challenge. Accordingly, for domains with strong rule-based constraints such as nuclear system-code inputs, prompt engineering or retrieval-augmented generation (RAG) alone often cannot guarantee consistent quality; instead, supervised fine-tuning on domain data is required to explicitly learn the formatting rules and physical constraints[3].

In conventional large-reactor applications, extensive experience has been accumulated in input development, and manual input preparation and verification procedures have been routinely practiced. In contrast, for emerging next-generation reactors such as small modular reactors (SMRs), regulatory application and modeling requirements are now being established, and corresponding input preparation and verification workflows are expected to be increasingly required.

Accordingly, this study proposes an LLM-based automatic input generation approach as a foundation for improving the efficiency of input preparation, rather than directly applying the conventional manual workflow used for large reactors to SMR applications.

2. Materials and Methods

MELCOR is a system code that simulates severe accident phenomena by coupling multiple physics packages, and its input files are organized according to package-specific cards and rules. The major packages considered in this study are as follows.

Thermal-hydraulics packages

Control Volume Hydrodynamics (CVH): calculates thermal-hydraulic states within control volumes, including pressure, temperature, and liquid level.

Flow Path (FL): models flow paths responsible for mass and energy transfer between control volumes.

Heat Structure (HS): models heat transfer between structures and fluids as well as the thermal response of structures.

Core and structural packages

Core Degradation (COR): analyzes core heat-up, oxidation, fuel melting, and relocation processes.

Cavity (CAV): analyzes thermal-hydraulic behavior in the lower containment region and interactions with relocated melt.

Radionuclide and auxiliary calculation packages

Radionuclide (RN): calculates the release, transport, and deposition of fission products.

Decay Heat (DCH): calculates decay heat from radioactive nuclide decay.

Noncondensible Gas Equation of State (NCG): computes thermophysical properties of noncondensible gas mixtures based on an equation of state.

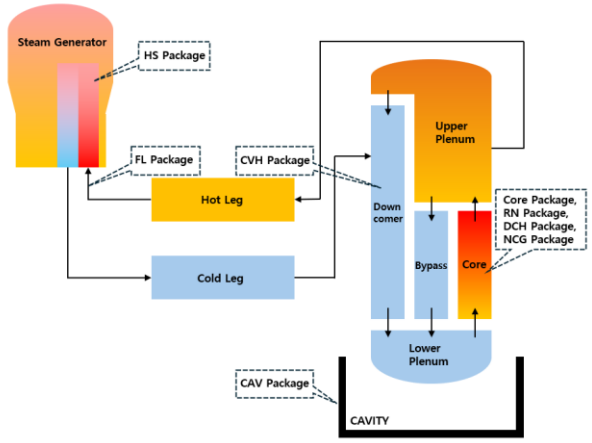

Fig. 1 Conceptual OPR1000 schematic for illustrating major MELCOR packages

Fig. 1 presents a simplified OPR1000 nodalization to illustrate where major MELCOR packages are typically applied: COR, RN, DCH, and NCG Packages are mainly associated with core modeling, CAV Package with cavity modeling, HS Package with heat-transfer components (e.g., the steam generator), while CVH and FL are used broadly to represent control volumes and flow paths throughout the system.

Each package has its own input rules, and the input file must be constructed to satisfy these requirements. To generate inputs that reflect package-level rules, this study builds a domain-example-based dataset and automatically generates MELCOR input files by fine-tuning an LLM using LoRA-based parameter-efficient fine-tuning (PEFT). The following subsections describe the LLM generation mechanism and the fine-tuning method in detail.

2.1 Principle of Large Language Models (LLMs)

A Large Language Model (LLM) is a neural network–based generative model trained on large-scale text corpora to learn linguistic patterns and to predict the probability of the next token given an input context. Here, a token is the basic unit of text used by the model and may correspond to a subword, symbol, or number rather than a full word. During training, the model is optimized to maximize the conditional likelihood of token sequences, i.e., the conditional probability of each token given its preceding context.

LLM outputs are typically generated in an auto-regressive manner. That is, the model predicts the next token one at a time based on the input prompt and the tokens generated so far, and repeats this process until the complete sequence is produced. In this study, the model takes user-provided natural language specifications as input and generates MELCOR input files; therefore, the task can be formulated as a conditional generation problem, where MELCOR inputs are generated conditioned on the user's specifications. This formulation is well suited to simulation input generation, because changes in user conditions should be consistently reflected in the generated outputs, including both the card structure and parameter values. However, unlike free-form narrative text, MELCOR input files consist of highly structured, rule-based elements such as card identifiers, numerical parameters, and symbols. To ensure executability, the generated inputs must satisfy both formatting consistency and value consistency. Accordingly, we construct an instruction–input paired dataset so that the LLM can learn the domain-specific input format, and we improve input generation performance via supervised fine-tuning, as described in the next subsection.

2.2 LoRA-based Parameter-Efficient Fine-Tuning (PEFT) Strategy

The training data are organized as one-to-one instruction–input pairs, where each instruction describes the user-specified conditions and the corresponding target is the MELCOR input file. These pairs are tokenized and used for model training. Through this process, the model learns the mapping between natural language conditions and the card-level structure of the input file, thereby acquiring generation patterns that appropriately adapt the input content to changes in the specified conditions.

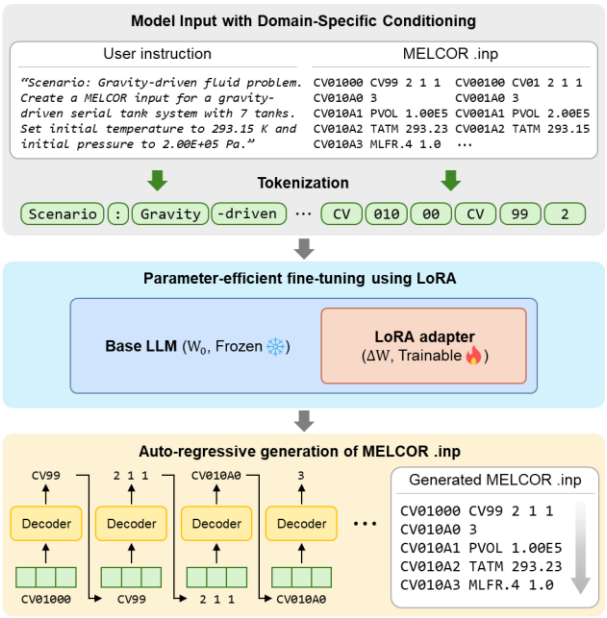

Fig. 2. Low-Rank Adaptation(LoRA) based Parameter-Efficient Fine-Tuning (PEFT) Strategy

Figure 2 summarizes the proposed domain-conditioned MELCOR input generation workflow, from instruction–input pairing and tokenization to LoRA-based PEFT and auto-regressive generation of MELCOR input files. To reduce computational cost and memory requirements, we adopt a PEFT strategy and apply LoRA as the representative method. Rather than updating all weights of the pre-trained model, LoRA performs domain adaptation by learning only a small set of additional low-rank parameters. Conceptually, the LoRA update is defined by Eqs. (1)–(2), where the pre-trained weight matrix W_0 is augmented by a low-rank adaptation term ΔW.

$$W = W_0 + \Delta W, \qquad \Delta W = BA, \tag{1}$$

$$B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, r \ll min(d,k) \tag{2}$$

Automatic generation of MELCOR input files is not a mere repetition of a single template; rather, both the card structure and parameter values must change consistently with user-specified conditions such as the number of control volumes, initial conditions, and other scenario parameters. Therefore, to ensure that the model can robustly handle diverse condition variations, a large collection of instruction–input pairs covering different settings is required. The fine-tuning dataset should be designed not only to expose the model to package-level rules, but also to enable verification that the generated inputs are structurally complete (e.g., required cards, valid formats, and internal consistency) and physically consistent (e.g., agreement with specified initial conditions and key variables).

Accordingly, we selected a gravity-driven scenario as a representative benchmark for evaluating MELCOR input generation performance. Using CVH–FL packages, this scenario examines whether the generated inputs can reproduce gravity-driven flow behavior while maintaining consistent definitions of flow paths and control-volume configurations. The next subsection describes the input conditions and input-file construction procedure for this scenario.

2.3 Gravity driven Scenario

The gravity-driven scenario constructed in this study is a simplified thermal-hydraulic model in which control volumes (CVs) are connected in series, designed to analyze transient flow behavior induced by gravitational head differences. The model consists of n tanks connected in series ($n = 2$–21), where adjacent control volumes are linked through flow paths (FLs). In addition, an atmospheric control volume (CV99) is connected to all control volumes and flow paths to represent the external ambient boundary condition.

The initial condition was set such that liquid exists only in the top control volume (CV01), while the downstream control volumes (CV02–CVn) are defined to contain no liquid initially. This configuration creates an initial liquid-level difference and associated head difference between CV01 and the downstream volumes. Elevation differences were assigned between control volumes so that gravity-driven natural flow develops from upstream to downstream. Each control volume is represented by MELCOR's control-volume-based thermal-hydraulic model with separate liquid and gas regions, and the temporal evolution of state variables is computed by integrating the mass and energy conservation equations over time.

The training and validation input cases for this scenario were constructed by combining the number of tanks with the initial thermal-hydraulic conditions. Specifically, the number of tanks was varied over $n = 2-21$, the initial temperature was set to $T_0 = \{283.15, 288.15, 293.15, 298.15, 303.15, 308.15\}$ K, and the initial pressure was set to $P_0 = \{1.01\times 10^5, 1.50\times 10^5, 2.00\times 10^5, 5.00\times 10^5, 8.00\times 10^5\}$ Pa.

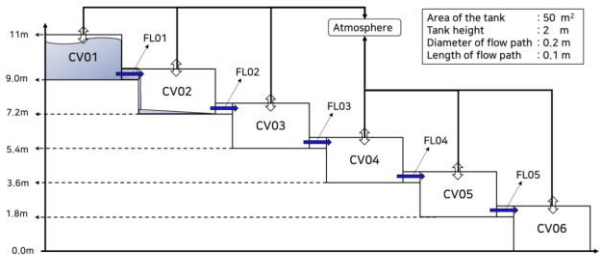

Fig. 3. Schematic of the six-tanks gravity-driven system.

Fig. 3 illustrates a representative configuration used in this study. As an example, six control volumes (CV01–CV06) are arranged at different elevations and adjacent volumes are connected by five flow paths (FL01–FL05).

Initially, the liquid in CV01 moves toward CV02 due to the gravitational head difference. As liquid accumulates in CV02, a new head difference develops

between CV02 and CV03, and this process repeats as the liquid progresses downstream. Over time, the liquid-level differences between adjacent control volumes diminish; as the head differences become negligible, the flow rate decreases and approaches zero.

3. Results and discussion

This section quantitatively compares the quality of MELCOR input files generated by the fine-tuned LLMs and evaluates their executability and representative physical behaviors by running MELCOR with the generated inputs. Because simulation input generation is governed primarily by structural correctness (e.g., card formats, missing cards, ordering, and identifier schemes) and executability (i.e., passing MELGEN preprocessing and successful MELCOR execution), rather than by sentence-level similarity, we perform both error-counting-based quantitative evaluation and code-execution-based verification.

3.1 Structural correctness analysis

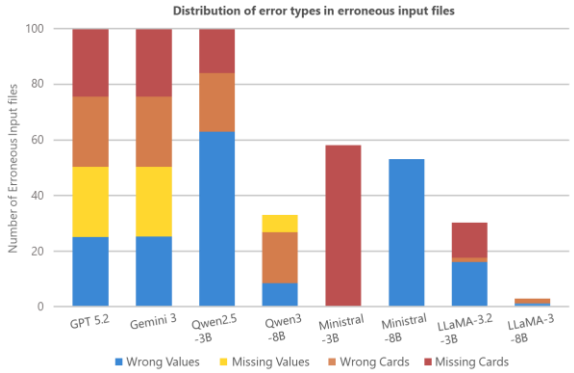

Fig. 4. Distribution of error types in erroneous input files

For quantitative comparison, we performed error counting on the generated input files. In this study, errors were defined primarily as structural defects with respect to MELCOR input rules, including missing cards, invalid card formats, inconsistent identifiers (numbering), and incorrect card placement/ordering. By aggregating the frequency and types of errors for each model under the same conditions, we found that LLaMA-3-8B Model achieved the lowest error level and most stably maintained the input structure (Fig. 4). This suggests that the model can reproduce MELCOR's rule-based input format more consistently than the others.

3.2 Code-generated execution

To assess executability, we ran MELGEN, the MELCOR preprocessing step, and confirmed that the LLM-generated input files were processed successfully without syntax errors. In particular, the formats and identifier schemes of the control volume (CV) and flow path (FL) definition cards satisfied MELCOR input rules, and the card ordering also met execution requirements. These results indicate that the LLM can consistently construct input files in an executable format, rather than merely generating plausible text.

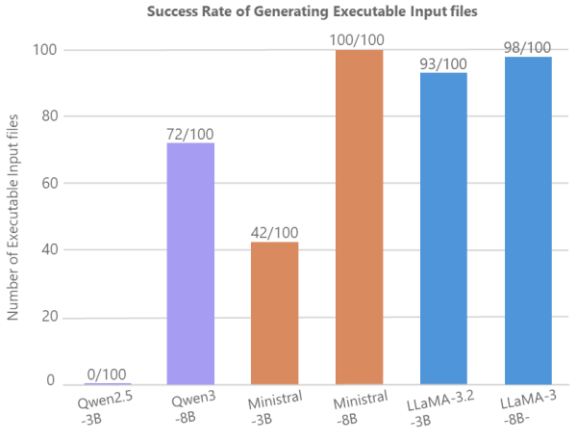

Fig. 5. Success Rate of Generating Executable input files

However, structural correctness measured by error counting does not always translate directly into run-level outcomes. As shown in Fig. 4, the Ministral-8B model achieved a high success rate in generating executable inputs (i.e., input files that passed MELGEN and completed MELCOR runs), even though its error counts in Fig. 5 are higher than those of the LLaMA-3-8B model. This indicates that executability alone is not sufficient to ensure the reliability of generated inputs; beyond confirming successful runs, it is necessary to carefully examine whether key parameter values and physical conditions are correctly reflected in the generated input files.

3.3 Advantages and future work

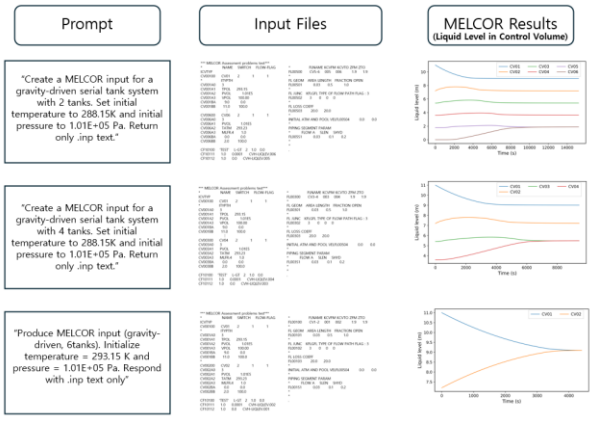

Fig. 6. Example LLM-generated input

This study demonstrated the feasibility of an LLM-based approach for reducing user-induced errors ("User

Effect") [1] in preparing MELCOR input files that require strict, rule-based formats, while improving the efficiency of input generation and verification.

Fig. 6 presents representative prompt–input–result examples, illustrating an end-to-end workflow from natural language specifications to executable MELCOR inputs and simulation outputs. These examples support the practicality of the proposed approach in reducing manual input editing and associated verification overhead. Future work will leverage available SMR input data for further training and will investigate whether the proposed approach can reliably maintain structural consistency and executability for complex, full-system input files that involve multiple interdependent packages.

4. Conclusions

As the importance of severe accident analysis continues to grow, the demand for scenario-based evaluations using severe accident system codes such as MELCOR and MAAP is increasing. However, these codes require complex, rule-intensive input files, and manual input preparation is therefore prone to User Effects, including missing cards, numbering errors, and format inconsistencies.[1] Such errors can lead to preprocessing failures or reduced reliability of the analysis results.

To mitigate the inefficiency and User Effects in input preparation, this study proposes an LLM-based framework for automatic generation of MELCOR input files. We constructed an instruction–input paired dataset and performed supervised fine-tuning with LoRA-based parameter-efficient fine-tuning (PEFT) so that the model could learn MELCOR input formats and rules. For quantitative comparison, we conducted error counting and confirmed that LlaMA-3-8B Model achieved the lowest error level, demonstrating superior structural correctness. In addition, the LLM-generated input files passed MELGEN preprocessing without syntax errors, and MELCOR simulations for representative cases completed successfully without premature termination, confirming practical executability.

Future work will extend this approach using real SMR input data to verify whether the model can reliably generate input files that satisfy structural consistency and physical constraints even for complex, full-system models with multiple interdependent packages. Ultimately, this effort aims to reduce the time and error burden of input preparation in SMR analysis workflows and to improve the efficiency and consistency of the input generation stage.

Acknowledgement

This work was supported by the Nuclear Safety Research Program through the Regulatory Research Management Agency for SMRs (RMAS) and the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. RS-2024-00509653) and by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety (KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (no. RS-2024-00403364).

REFERENCES

[1] Humphries, L., and Gauntt, R. O., *MELCOR 2.2 Severe Accident Analysis Code – Current Status and Plans for Future*, Sandia National Laboratories, SAND2018-0310C, 2018.

[2] Verduzco, J. C., Holbrook, E. W., and Strachan, A., "GPT-4 as an interface between researchers and computational software: improving usability and reproducibility," arXiv preprint, 2024.

[3] Jacobs, P. F., and Police, R., "Developing large language models for quantum chemistry simulation input generation," Digital Discovery, 2025.