Development Strategy for a Hybrid Load-Following Simulation System: Combining Deep Learning-based Prediction and Optimization-Driven Control

Jungseok Kwon [a], Tongkyu Park [a*], Sung-kyun Zee [a]
[a]*FNC Technology, Headquarters, 13 Heungdeok 1-ro, Giheung-gu, Yongin-si, Gyeonggi-do, 16954, Republic of Korea*
[*]*Corresponding author: tongkyu@fnctech.com*

***Keywords :** Load Following Operation, Machine Learning, Surrogate Model, Model Prediction Control, Optimization

1. Introduction

In Korea, nuclear energy has been utilized as a baseload source. However, to be compatible with renewable sources and support the rising electricity demand from AI and EVs, Nuclear Power Plants (NPPs) require operational flexibility.

Unlike standard base-load operation, load-following scenarios involve varying power levels, typically on a daily basis. Therefore, evaluating load-following feasibility requires analyzing the core state across a large number of time steps compared to those in base-load analysis. However, verification using conventional design codes is impractical due to the significant computation time. Consequently, there is a growing demand for faster tools to support engineering decisions.

In response to this demand, we are developing a Deep Learning (DL)-based simulation framework. However, relying solely on DL for control logic makes it difficult to enforce that control variables strictly comply with operational parameters, such as physical bounds and control rod overlap. It also faces challenges in solving inverse problems reliably. To address these limitations, we propose a hybrid system integrating our DL-based surrogate model with a Model Predictive Control (MPC)[1] framework. In this architecture, the surrogate model serves as a rapid predictive engine, while the optimization algorithm-based operator manages control variables to ensure precise and compliant load-following.

2. Methodology

In this section, we outline the overall framework and its sub-modules. We begin with a structural overview and the operational flow of the entire system, followed by a detailed description of each component. Regarding the AI models, we specify the functional roles within the system, anticipated data inputs/outputs. Furthermore, for the operator component designed to determine the optimal control rod maneuvering strategy for daily load following scenarios, we define the necessary constraints and operational mechanics, independent of a specific algorithmic selection.

2.1 Overview of the system

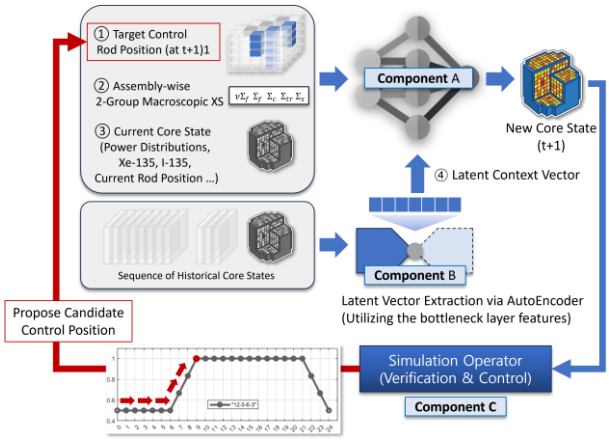

Fig. 1. Overview of the hybrid simulation framework.

As illustrated in Fig. 1, the proposed system comprises three primary components. Components A and B are designed as deep learning-based models, while the third component acts as an optimization algorithm-based simulation operator.

Specifically, Component A functions as a surrogate model for core analysis[2]. It takes the current core state, macroscopic cross-sections (XS), the displacement of control rod position, and a latent vector representing the historical context of the core state (provided by Component B) as inputs to predict the core state at the subsequent time step (t+1). The predicted state includes the thermal power distribution, which is subsequently converted into the total power load metric via a specific calculation function. Component B serves as a context encoder; it processes a sequence of past states, including parameters such as Xenon-135 and Iodine-135 concentrations (e.g., a 60-point window), and extracts compressed temporal information to feed into Component A.

The Simulation Operator executes the load-following scenario sequentially. Ideally, a comprehensive optimization strategy would necessitate predicting the system state across the entire sequence of future time steps to determine the optimal maneuvering plan.

However, we adopt a phased development strategy. Therefore, we initially employ a simplified rule-based iterative logic to validate the system framework before integrating full-scale optimization. The rule-based logic specifically focuses on the immediate next step (t+1). Its primary role is to verify whether control rod adjustments

maintain criticality and keep power load fluctuations within permissible bounds. The operational loop proceeds as follows: First, the historical context vector, current core state, and the new control rod position (calculated as a displacement) are fed into Component A. Subsequently, the Operator verifies the predicted output. Once these constraints are satisfied, the system advances to the next target load defined in the scenario. The primary objective of this phase is to validate the deep learning models (Components A and B), after which a full-scale optimization algorithm will be integrated.

2.2 Component A: Core Analysis Surrogate Model

Component A functions as the primary predictive engine within the proposed framework, serving as a deep learning-based surrogate for conventional nodal diffusion codes. Its specific objective is to predict the reactor core state at the subsequent time step (t+1) in response to a target control rod position proposed by the Operator.

To ensure high-fidelity predictions during load-following maneuvers, the model integrates the current physical state (e.g., macroscopic cross-sections and power distribution) with a latent context vector provided by Component B. This integration allows Component A to effectively capture historical trends, particularly the complex dynamics of Xenon-135 oscillations. Consequently, the model rapidly generates the new 3D core state, enabling the Simulation Operator to verify compliance with safety constraints—such as criticality and peaking factors—before executing any control action.

Component A receives a composite input vector consisting of four distinct data groups:

(1) Target Control Rod Position (at t+1): The candidate control rod configuration proposed by the Simulation Operator (Component C) for the next time step.

(2) Assembly-wise Macroscopic Cross-sections (XS): A tensor representing the core's material properties, designed to embed physics parameters into the model. Fig. 2 illustrates the data construction process, where text-based assembly identifiers are mapped to numerical cross-section values via a library lookup.

(3) Current Core State (at t): An instantaneous snapshot of the reactor status serving as the initial condition. This includes 3D spatial distributions of key variables such as thermal power, Xenon-135 and Iodine-135 concentrations, and the current control rod positions.

(4) Latent Context Vector: A compressed feature vector generated by Component B. It encodes the temporal context extracted from the sequence of historical core states, allowing Component A to account for time-dependent dynamics like Xenon oscillation

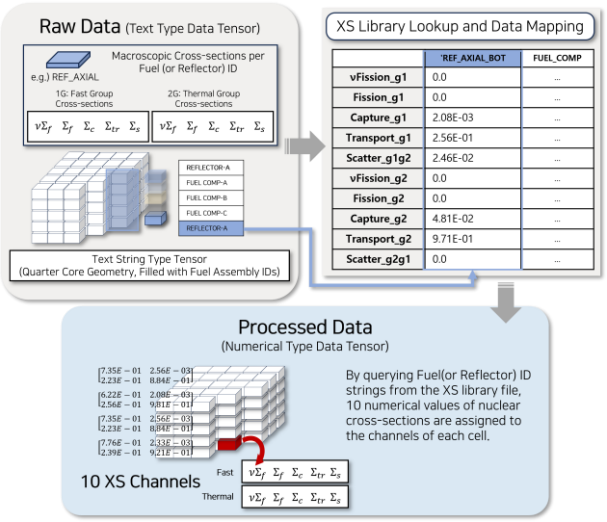

Fig. 2. The construction process of input macroscopic XS

2.3 Component B: Historical Context Extraction Model

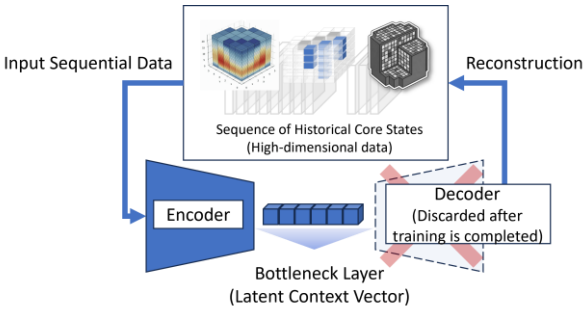

Fig. 3. Structure of the AutoEncoder-based Component B.

Component B is designed to address the non-Markovian nature of nuclear reactor dynamics, where the future state depends not only on the current snapshot but also on the cumulative history of operation (e.g., Xenon-135 build-up and decay). Instead of feeding raw high-dimensional historical data directly into the Component A, Component B functions as a feature extractor that compresses the temporal context of core states into a compact vector representation.

The model architecture is based on an AutoEncoder[3] framework (see Fig.3). It takes a Sequence of Historical Core States (e.g., a sliding window of the past N time steps) as input. During the training phase, the network learns to compress this high-dimensional input into a bottleneck layer and subsequently reconstructs the original data to ensure essential feature retention. For the operational phase, the decoder part (layers following the bottleneck) is discarded. Consequently, the activation output of the bottleneck layer is directly extracted and utilized as the Latent Context Vector to be fed into Component A.

2.4 Component C: Simulation Operator

Component C functions as the central Simulation Operator, responsible for maneuvering the reactivity control mechanisms in accordance with a prescribed load-following scenario. This scenario is defined by a specific "load profile" (or power trajectory), which specifies the target power level at discrete time intervals (e.g., every 10 minutes). For each time step, the Operator identifies the necessary control rod positions to transition the reactor from its current state (e.g., 100% power) to the target level (e.g., 98%).

In the initial phase of development, the system employs a deterministic, iterative rule-based logic. Utilizing the core state parameters predicted by Component A—such as criticality, power level, peaking factors, and axial offsets—the operator decides the displacement of control rod banks (insertion or withdrawal). Once all safety constraints are satisfied, the control rod position is confirmed, and the simulation advances to the next target point in the load profile. Figure 4 illustrates the flow of the system, which is integrated with the rule-based logic.

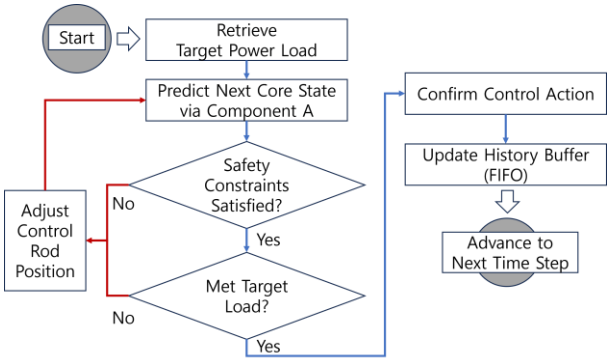

Fig. 4. Flowchart of the system integrated with rule-based control logic.

Following the verification of the initial phase, future work will focus on integrating advanced optimization solvers, including Genetic Algorithms (GA) and path-finding methods like Monte Carlo Tree Search (MCTS). With these enhancements, Component C extends its capabilities from immediate single-step corrections to long-term trajectory optimization. This allows the system to identify a control rod path that strictly adheres to the global constraints of the entire scenario, thereby validating the feasibility and safety of the load-following operation.

2.5 Training Dataset

Training data for Components A and B were generated using the MASTER nodal diffusion code under fixed soluble boron condition. Each data record corresponds to one loading pattern (LP) paired with one load-following scenario, yielding a time series of 575 steps at 10-minute intervals. At each step, 30 branch calculations are performed by perturbing the control rod position (CRS), producing a comprehensive set of cross-section and core-state variables. Table 1 summarizes the overall dataset structure, Table 2 enumerates the physical quantities used as model inputs or outputs, and Table 3 describes the normalization methods applied prior to training.

Table 1. Training Dataset Overview

Item	Value
Number of LPs	100
Scenarios per LP	1 (round-robin, 1 of 20 types)
Unique scenarios	20
Time steps per scenario	575
Branches per step	30 (CRS reference + 29 rod offsets)
Spatial resolution	20 axial planes × 5×5 quarter-core (19 fuel nodes)
Data type	float32 (float64 for normalization statistics)
Normalization basis	Training set statistics only (no data leakage)

Table 2. Physical Quantities in the Training Dataset

Physical Quantity	Sym.	Shape	Unit
Thermal power distribution	Q_abs	(20,5,5)	MW
Coolant temperature	T_cool	(20,5,5)	K
Fuel temperature	T_fuel	(20,5,5)	K
Coolant density	rho_c	(20,5,5)	g/
k_eff	k_eff	scalar	—
Axial offset	AO	scalar	—
Max pin power	P_max	scalar	—
Xe-135 number density	N_Xe	(20,5,5)	cm³
I-135 number density	N_I	(20,5,5)	cm³
Sm-149 number density	N_Sm	(20,5,5)	cm³
Macro. XS 2-grp (LP-fixed)	xs_f	(20,5,5,10)	cm^{-1}
Xe-135 abs. XS	sa_Xe	(20,5,5,2)	barn
Xe-135 fis. yield	gXe	(20,5,5)	—
I-135 fis. yield	gI	(20,5,5)	—
2-grp flux	Phi	(20,5,5,2)	n/cm^2s
Control rod map (3D)	rod_m	(20,5,5)	—
Power load	P_ld	scalar	[0,1]

2.5.1 Scenario Split

The 20 unique scenarios are partitioned into three non-overlapping subsets at the scenario level to prevent data leakage. The training set (16 scenarios, 80 LP-scenarios) covers t12_363_p50, t12_8_p50, t14_262_p50, and t14_6_p20, each in four variants: power_lower, power_upper, ramp_down, and ramp_up. The validation set (2 scenarios, 10 LP-scenarios) consists of t14_6_p50_power_lower and t14_6_p50_power_upper.

The test set (2 scenarios, 10 LP-scenarios) consists of t14_6_p50_ramp_down and t14_6_p50_ramp_up. The entire t14_6_p50 group is reserved as the holdout set, ensuring the model is evaluated on operational conditions entirely unseen during training.

2.5.2 Normalization

All features are normalized using training set statistics only, prior to train/validation/test split, to prevent data leakage. Four normalization methods are applied according to the statistical characteristics of each feature, as described in Table 3.

Table 3. Normalization Methods Applied to Training Features

Method	Characteristics	Applied to
Log Z-score	Log transform before Z-score standardization. Effective for heavily right-skewed distributions spanning multiple orders of magnitude.	Xe-135, Sm-149, I-135 densities; 2-grp flux; thermal power
Z-score	Standard mean-zero, unit-variance normalization. Suitable for approximately symmetric (normal) distributions.	Macro. XS (sa_Xe, Sf, gXe/I); T_cool, T_fuel; rho_c; AO; P_max
Min-Max	Linear scaling to [0, 1]. Appropriate for bounded quantities with well-defined physical minima and maxima.	Control rod map (3D); fractional load
PCM Z-score	Converts k_eff to reactivity in pcm ($\times 10^5$) before Z-score. Resolves the near-unity value clustering of raw k_eff.	k_eff

3. Discussion

In this section, we briefly discuss the rationale behind the architectural design and the operational constraints of the proposed hybrid framework. We specifically address the design considerations that led to combining deep learning with rule-based logic, focusing on the enforcement of hard constraints and the avoidance of inverse problems. Additionally, we describe the simplification of simulation conditions, such as the exclusion of soluble boron, and explain the necessity of utilizing latent vector compression for handling historical data.

3.1. Design Considerations for the Hybrid Framework

3.1.1. Necessity of Enforcing Hard Constraints

Our initial design strategy aimed to implement the entire framework using a fully modularized deep learning architecture, where each module would autonomously handle a specific function. However, we identified that relying exclusively on deep learning for system control poses a fundamental limitation regarding the enforcement of hard constraints. Due to the nature of deep learning, these models typically operate under soft constraints managed by loss functions, which cannot guarantee strict adherence to absolute boundaries.

For instance, consider a scenario where a deep learning model determines the target control rod position for the next time step. In such a case, it is inherently difficult to strictly mandate complex operational rules, such as the overlap ratio between control rod banks, or to guarantee that the output values fall precisely within the physical maneuvering bounds (e.g., 0 to 100).

3.1.2. Control Stability and Avoidance of Inverse Problems

Another critical factor in our design choice involves the stability of the iterative control process and the mathematical nature of the prediction task. relying solely on a self-regressive deep learning model for the entire control loop introduces significant challenges regarding error propagation and convergence. In such a setup, the model often struggles to autonomously identify the precise termination point (i.e., when to stop the iteration) once the target state is reached.

More importantly, tasking a deep learning model with directly determining the next control rod position constitutes an inverse problem, which is inherently difficult to solve reliably. In terms of system dynamics, control rod maneuvering serves as the input, while the resulting core state is the output. Attempting to predict the required input (control rod position) to achieve a specific target state (e.g., a 5% load reduction) requires mapping the output back to the input. This inverse mapping is often ill-posed for neural networks, leading to non-unique or unstable solutions.

To circumvent these issues, we adopted a hybrid architecture that strictly separates these roles. The task of determining the optimal control actions (inputs) is assigned to traditional optimization or path-finding algorithms, which are well-suited for decision-making. Meanwhile, the deep learning model is utilized exclusively as a forward simulator, predicting the resultant core state (output) based on the provided inputs. This approach effectively converts a challenging inverse problem into a manageable forward prediction task.

3.2. Simplification of the simulation condition

The current simulation framework is explicitly limited to soluble boron-free conditions. This simplification was adopted to reduce the complexity of the initial modeling phase (Component A). By holding the boron concentration constant, we significantly reduced the dimensionality of the input data required for the deep learning model. Furthermore, this assumption simplifies the decision-making process for the Simulation Operator

by constraining the search space for reactivity control solely to control rod maneuvering.

However, in actual operation, the target reactor (e.g., SMART) utilizes soluble boron as an auxiliary mechanism for reactivity control. Acknowledging this discrepancy, we plan to extend the framework in future study. The optimization algorithm will be upgraded to include boron concentration and coolant temperature as controllable variables, thereby enhancing the fidelity of the simulation to match real-world operational strategies.

3.3. Rationale for Latent Vector Compression of Historical Reactor States Data

Incorporating historical context requires processing sequences that cover an extended range of past time steps. For instance, a 10-hour history recorded at 10-minute intervals results in a sequence of 60 distinct data points. Directly feeding such high-dimensional time-series data into the primary predictive model (Component A) creates a significant challenge.

Specifically, there is a risk of architectural imbalance within the neural network layers of Component A. The sub-modules responsible for processing this voluminous input could dominate those handling the instantaneous state inputs (e.g., current power or control rod position). To address this issue, we designed the framework to first pass the raw time-series data through a dedicated deep learning module (Component B). This module acts as a feature extractor, compressing the extensive temporal information into a compact latent vector representation, which is then efficiently integrated into Component A.

4. Conclusion

In this study, we proposed a hybrid simulation framework for efficiently evaluating the feasibility of load-following operations in Nuclear Power Plants (NPPs). To address the computational burdens of conventional high-fidelity design codes and the stability limitations of end-to-end deep learning approaches, we established a decoupled architecture integrating a deep learning (DL)-based surrogate model with an algorithm-based Simulation Operator.

Through an analysis of potential operational risks, we demonstrated that relying solely on deep learning for control logic presents critical challenges in enforcing hard physical constraints and handling inverse problems. Accordingly, the proposed framework strictly separates state prediction from decision-making logic, ensuring both physical consistency and algorithmic flexibility.

Furthermore, to account for the inherently non-Markovian characteristics of reactor physics, we established the necessity of a context extraction module (Component B), which compresses high-dimensional historical information into a latent representation, thereby structuring the surrogate model input in a computationally efficient manner.

Future work will proceed in two phases. First, we will generate high-fidelity training data and validate the predictive performance of the surrogate components (Components A and B) under soluble boron-free conditions. Subsequently, the rule-based Simulation Operator will be extended to advanced optimization strategies, such as Genetic Algorithms (GA)[4] and Monte Carlo Tree Search (MCTS)[5], enabling global trajectory optimization for complex daily load-following scenarios.

Acknowledgement

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

REFERENCES

[1] J. Richalet, A. Rault, J. L. Testud, and J. Papon, “Model predictive heuristic control: Applications to industrial processes,” Automatica, vol. 14, no. 5, pp. 413–428, 1978.
[2] J. Kwon, T. Park, and SK. Zee, “AI-Based Prediction Module of Key Neutronic Characteristics to Optimize Loading Pattern for i-SMR with Flexible Operation,” Korean Journal of Chemical Engineering, vol. 41, no. 10, pp. 2741–2759, 2024.
[3] M A Kramer, “Nonlinear Principal Component Analysis Using Autoassociative Neural Networks,” AIChE Journal, vol. 37, no. 2, pp. 233–243, 1991
[4] J. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, 1992.
[5] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. 5th International Conference on Computer and Games, May 2006, Turin, Italy.