Journal of
Nuclear Engineering

MDPI

Article

Reinforcement Learning-Based Control Sequence Optimization for Advanced Reactors

Khang H. N. Nguyen, Andy Rivas, Gregory Kyriakos Delipei and Jason Hou *

Department of Nuclear Engineering, North Carolina State University, 2500 Stinson Dr, Raleigh, NC 27695, USA; khnguy22@ncsu.edu (K.H.N.N.); arivas2@ncsu.edu (A.R.); gkdelipe@ncsu.edu (G.K.D.)
* Correspondence: jhou8@ncsu.edu

Abstract: The last decade has seen the development and application of data-driven methods taking off in nuclear engineering research, aiming to improve the safety and reliability of nuclear power. This work focuses on developing a reinforcement learning-based control sequence optimization framework for advanced nuclear systems, which not only aims to enhance flexible operations, promoting the economics of advanced nuclear technology, but also prioritizing safety during normal operation. At its core, the framework allows the sequence of operational actions to be learned and optimized by an agent to facilitate smooth transitions between the modes of operations (i.e., load-following), while ensuring that all safety significant system parameters remain within their respective limits. To generate dynamic system responses, facilitate control strategy development, and demonstrate the effectiveness of the framework, a simulation environment of a pebble-bed high-temperature gas-cooled reactor was utilized. The soft actor-critic algorithm was adopted to train a reinforcement learning agent, which can generate control sequences to maneuver plant power output in the range between 100% and 50% of the nameplate power through sufficient training. It was shown in the performance validation that the agent successfully generated control actions that maintained electrical output within a tight tolerance of 0.5% from the demand while satisfying all safety constraints. During the mode transition, the agent can maintain the reactor outlet temperature within ±1.5 °C and steam pressure within 0.1 MPa of their setpoints, respectively, by dynamically adjusting control rod positions, control valve openings, and pump speeds. The results demonstrate the effectiveness of the optimization framework and the feasibility of reinforcement learning in designing control strategies for advanced reactor systems.

Keywords: advanced reactors; small modular reactors; machine learning; reinforcement learning; control sequence optimization

Citation: Nguyen, K.H.N.; Rivas, A.; Delipei, G.K.; Hou, J. Reinforcement Learning-Based Control Sequence Optimization for Advanced Reactors. *J. Nucl. Eng.* **2024**, *5*, 209–225. https://doi.org/10.3390/jne5030015

Academic Editors: Dan Gabriel Cacuci and Guglielmo Lomonaco

Received: 25 March 2024
Revised: 13 June 2024
Accepted: 15 June 2024
Published: 1 July 2024

1. Introduction

Reliable and sustainable energy is a cornerstone for numerous aspects of modern society. Nuclear energy emerges as a convincing solution to both aspects due to its ability to offer a consistent and uninterrupted delivery of energy (reliability) without compromising environmental integrity (sustainability). As the nuclear industry is motivated to innovate advanced reactor systems to address projected energy needs, reducing costs associated with construction, operational, and maintenance while adhering to safety protocols becomes crucial [1]. To this end, many advanced reactor designs envision their deployment on the multi-unit configuration of small modular reactors (SMRs) to enhance economic competitiveness, and the SMR plant operation must be simplified and optimized to reduce the cost without deteriorating the system's safety. One way to achieve this goal is to increase the overall effectiveness of operational staff by implementing a comprehensive approach that combines a higher degree of reliability and autonomy in the control system and a well-informed decision-making process. In this pursuit, data-driven techniques have gained prominence, offering innovative solutions to complex challenges across various domains within the nuclear industry [2].

Typically, operating a nuclear power plant (NPPs) necessitates a great amount of complicated control processes due to the intrinsic complexity of the system. Historically, conventional methods such as proportional–integral–differential (PID) controllers, programmable logic controllers (PLCs), and field programmable gate arrays (FPGAs) have been utilized to automate these tasks [3–5]. However, these approaches have their respective limitations. PID controllers, originally designed for simpler systems with single control variables, often struggle to cope with the complexities of NPPs, leading to issues such as overshooting and requiring manual tuning for optimal performance in different operating conditions. It was shown in a previous study that, in an NPP where safety parameters are sensitive to control actions, conventional single-variable control approaches, like the PID, struggle to manage intricate relationships between variables [6]. PLCs are designed for discrete control applications, emphasizing reliability and ease of use, but this limits their processing power. As a result, they may struggle with highly complex control tasks or those requiring real-time decision-making with fast response times [4]. Meanwhile, offering a high order of complexity, FPGAs can complete complicated tasks with ease, but this comes with higher costs and requirements in terms of having an in-depth understanding of hardware design and digital logic [7].

In recent years, model-free deep reinforcement learning (RL), inspired by human trial-and-error learning, has been studied across industries [8]. With the use of multiple deep neural networks as universal approximators [9], RL shows promising potential as an alternative approach for automating diverse decision-making and control tasks [10,11]. Additionally, with contemporary advancements in computing power, data, and deep learning research, RL-based controllers surpass traditional approaches in tasks such as devising operational strategies, real-time decision-making, and optimization systems. RL-based controllers have found applications in different domains, including robotics [12], autonomous vehicles [13,14], power management [15], and transportation [16].

RL control-based approaches offer the potential to significantly reduce the operational and maintenance costs of NPPs through several key advantages. These include optimized decision-making capabilities, allowing RL algorithms to learn and adapt control strategies that maximize system performance. By automating standard tasks traditionally conducted by humans, RL-based controllers reduce labor costs and mitigate the risk of human errors. Moreover, RL algorithms can be trained for predictive maintenance, enabling the system to anticipate failures and plan maintenance activities proactively, thereby avoiding costly unplanned downtime. RL-based controller offers adaptive control, which is superior to conventional controllers using fixed-term control. They can adjust plant operations in real time based on fluctuations in power demand. This helps maintain stable reactor conditions and prevents unsafe situations that might arise from abrupt load change. In Li et al. [17], the reactor-coordinated control systems were modeled and optimized with the deep deterministic policy gradient (DDPG) algorithm to enhance control effectiveness, showing a superior performance compared with traditional methods. Furthermore, Lee et al. [18] proposed an RL-based controller for a start-up operation. The controller was validated using a compact nuclear simulator (CNS), successfully controlled parameters within limits, and identified an acceptable operation path for increasing reactor power from 2% to 100%.

Inspired by the previous achievement in the application of RL in controlling NPPs, this work explores the application of a control sequence optimization (CSO) framework based on RL with the soft actor–critic (SAC) [19] to support operations and reduce the associated operational costs for next-generation reactors. The CSO framework aims to generate and optimize a series of operational actions to transition the reactor from an initial state to a desired state, while ensuring all system variables remain within predefined thresholds. More specifically, this work concentrates on the development and demonstration of load-following operations, which are regarded as one of the key features distinguishing SMRs from conventional large light water reactors (LWRs).

This article is structured as follows. The first section provides an overview of the background and outlines the main objectives of the study. Section 2 begins with the motivation

and core principles behind the CSO framework, followed by the description of a general PB-HTGR plant model with various key system components. Then, the operational goals, control variables, and the inputs and outputs of the RL training environments are discussed. Additionally, the RL-based controller development approach is presented with details of the customization of the SAC algorithm and the training process. Section 3 presents the validation and analysis of the results obtained from the RL-based controller. Finally, Section 4 summarizes the findings and discusses potential directions for future research.

2. Methodology

2.1. Overview of Control Sequence Optimization Framework

This section is dedicated to the development of a framework that optimizes the control sequence for the load-following operations of a general PB-HTGR. A control sequence is defined as a series of operational actions to bring the reactor from an initial state to an arbitrarily desired state. In the attempt to find the optimal control sequence, the "combinatorial optimization" (CO) problem is solved based on the operator's experience and judgment. Since the search space for CO is finite by definition, an optimal solution always exists [20].

Many traditional algorithms for solving CO problems require hand-crafted heuristics to find a solution. Such heuristics are designed by domain experts and may often be sub-optimal due to the hard nature of the problems. Genetic algorithm (GA) [21] and simulated annealing (SA) [22] represent two widely used approaches to solve static optimization problems where they find the optimal solution based on a fixed set of conditions. They efficiently explore the solution space and converge to the global optimum. However, these techniques are prone to be inferior when dealing with sequential CO problems, where the best decision at each step correlates with the outcome of the previous steps. GA and SA might become stuck in suboptimal solutions that worked well initially but become obsolete as the sequence progresses due to their inability to adapt dynamically. The Markov decision process (MDP) [23–25]) is specifically designed for such situations. It considers the dynamic nature of the problem, where each decision influences the future state. Mathematically, it can be defined by a tuple (S, A, P, R), explained as follows:

- S is the set of states representing all possible situations the system can be in.
- A is the set of actions the decision maker can take.
- P is the state transition probability function, defined as $P(s' \mid s, a) = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$. It represents the probability of transitioning to state s' given that action a is taken in state s.
- R is the reward function, defined by $R(s, a, s')$, which assigns a real value to each state–action–state triplet, representing the immediate reward received after transitioning from state s to state s' by taking action a.

RL proposes a promising approach to tackle MDP in finding the optimal solution by training an agent in a supervised or self-supervised manner [26,27]. To apply RL to solve CO problems, the problem is modeled as a sequential decision-making process, where the agent interacts with the environment by performing a sequence of actions to find a solution. Particularly, within the scope of this paper, an agent interacts with the PB-HTGR environment over a series of discrete time steps. At each time step t, the agent observes a state of the plant s_t and then selects an action a_t, which controls signals based on some policy π. The plant transitions to a new state s_{t+1} according to a transition function $P(s_{t+1} \mid s_t, a_t)$, and the agent receives a reward r_{t+1} according to the action it takes, which is based on this transition. The agent's goal is to learn an optimal policy π^* that maximizes the expected cumulative reward over time R, which, is mathematically expressed as

$$R = \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_t \right], \tag{1}$$

where π is a policy mapping states to actions; $\mathbb{E}_\pi$ denotes the expectation under policy π; R_t is the reward received at time step t, and γ is the discount factor.

In this work, the dynamic responses of the reactor system are modeled as the interactive environment, utilizing the system "digital twin" described in the following section. Each state encompasses both control variables and the target power output. The agent is trained to choose actions from the action space and evaluate their effectiveness based on a pre-defined reward function with imposed safety constraints. The subsequent subsections provide the development of the digital twin with a detailed description of the training environment, the RL algorithm employed, and the training process in this study.

2.2. PB-HTGR Training Environment

To provide a training environment for the RL algorithm within the CSO framework, data are generated from a coupled PB-HTGR core/system model using the system analysis module (SAM) and Simulink. SAM is a modern system analysis tool being developed at Argonne National Laboratory (ANL) for advanced non-LWR safety analysis, being used to model the core dynamics of a PB-HTGR [28]. The reactor core is the systems constant heat source that is controlled by the amount of positive or negative reactivity introduced to the system [29], mimicking the movement of control rods. Simulink is a MATLAB library that models the systems of ordinary differential equations (ODEs) [30], which is used to model the balance of plant (BOP) for electricity generation, the once-through steam generator (OTSG) that transfers heat from the primary to the secondary, and the control system [31]. The variables of interest are classified as manipulated variables (MVs) or controlled variables (CVs). MVs are the subset of variables that the operators can directly control and only change with operator intervention, including the control rod position, turbine control valve, circulator pump speed, feedwater pump speed, and condenser pump speed. CVs are the subset of variables that have safety-related implications if they exceed a pre-defined threshold, including the reactor power, outlet reactor temperature, inlet reactor pressure, primary mass flow rate, steam temperature, steam pressure, secondary mass flow rate, condenser pressure, and electrical power. Table 1 lists the one-to-one pairing of MVs and CVs. For load-following operations, PID controllers are introduced to control the plant during state transitions using MVs and maintain all CVs within safety limits. The CSO framework aims to replace these PID controllers with improved RL-based controllers.

Table 1. Simulink proposed control scheme mapping that has the MV directly controlling the CV or controlling the CV through an IV.

Manipulated Variable	Intermediate Variable	Controlled Variable
Control rod position	Reactor power	Outlet reactor temperature
Circulator pump speed	Primary mass flow rate	Main steam pressure
Feedwater pump speed	Secondary mass flow rate	Main steam temperature
Turbine steam valve	-	Electrical load
Condenser pump speed	-	Condenser pressure

2.2.1. Reactor Model

The generic 200 MW PB-HTGR SAM model openly available on the Idaho National Laboratory (INL) virtual test bed is adopted to capture the dynamics of the reactor during state transitions [32]. A PB-HTGR is specifically chosen over the other generation IV designs due to the core having low excess reactivity, allowing for online refueling, high efficiency, inherent safety, and operating at high temperature for process heat. The fuel within the PB-HTGR core comprises billiard ball-sized graphite pebbles that contain about 19,000 specially coated tristructural isotropic (TRISO) uranium fuel particles enriched to 15.5%. This PB-HTGR design is cooled with pressurized helium gas at 6 MPa, the inlet and outlet core temperatures are 260 °C and 750 °C, respectively, the nominal helium mass flow rate at full power is 78.6 kg/s, and the reactor power is calculated using the point

kinetics equation (PKE) with temperature reactivity feedback from the fuel, moderator, and reflector regions.

Due to the high demand for training data in the data-driven approach, a feedforward neural network (FFNN) surrogate of the SAM PB-HTGR core model is developed to replace the computationally expensive SAM core model and accelerate the training process. The surrogate was built using Tensorflow and imported to Simulink directly. FFNNs are deep neural networks that propagate information in a single direction from input to output with tuned weights and biases learned from training data generated with SAM [33]. The training database contained 5632 total samples ranging from 25% to 100% reactor power split into 60% for training, 20% for validation, and 20% for testing following 5-fold cross-validation. Specifically, the dataset is randomly split into five subsets, where we train on four and evaluate the trained model on one. We then iterate five times with a different subset reserved for evaluation to complete the model validation. The advantages of this training approach are overcoming overfitting by better estimating the model's performance on unseen data and is data-efficient since all available data are used during model training and hyperparameter tuning. The surrogate is trained to predict the outlet reactor temperature, pressure, mass flow rate, and thermal power given inlet temperature, pressure, mass flow rate, and control rod reactivity. After optimizing the hyperparameters to minimize the mean squared error following a grid search approach [34], the model was successful in explaining over 99.8% of the variance seen in the data with respective errors being less than 2% for each output variable.

2.2.2. Once-Through Steam Generator

To transfer the heat generated within the PB-HTGR core to the secondary, an OTSG is modeled using a nonlinear lumped parameter modeling approach that reduces the OTSG into discrete nodal volumes with average homogeneous physical parameters. An OTSG includes economizer, evaporator, and superheater regions into a single module shell-and-tube type heat exchanger that converts feedwater into steam using heat from the primary. On the tube side, the feedwater on the secondary enters the OTSG in the economizer region subcooled and begins to be heated as it flows upwards until exiting the top of the OTSG as superheated steam. To calculate the outlet pressure, the inlet pressure is decreased by the pressure drop due to friction, elevation, acceleration, and inlet/outlet form losses. Thus, the OTSG Simulink model outputs steam temperature, pressure, and mass flow rate with inputs of reactor outlet temperature, pressure, and mass flow rate.

2.2.3. Balance of Plant

Next, the BOP is modeled using Simulink to simulate the thermal energy conversion into electricity. The BOP model from [35] is used and includes a nozzle chest, high-pressure turbine (HPT), moisture separator, steam reheater, low-pressure turbine (LPT), condenser, condenser pump, feedwater pump, and high/low-pressure feedwater heaters. Using the steam valve, the steam coming from the OTSG is split between the nozzle chest and the reheater. The nozzle chest guides the incoming steam into separate turbine inlet nozzles of the HPT to rotate the main shaft and perform the work. After the HPT, the steam is dried using the moisture separator and reheater before entering the LPT to again rotate the main shaft. The work performed on the main shaft by the HPT and LPT is converted into electricity in the generator. Exiting the LPT, the condenser cools the exhaust steam, where cooling water from an external source is used to condense the steam. The condensate is then pumped into the low-pressure heater using a condenser pump. Next, the flow proceeds into the feedwater pump before entering the high-pressure heater to be heated once again before returning to the OTSG to repeat the process. Thus, the BOP Simulink model outputs electrical power with inputs of steam temperature, pressure, and mass flow rate.

2.2.4. Plant Control System

The PB-HTGR control system is tasked to control various system parameters to specific setpoints using PID controllers. These controllers combine one or multiple optimized gains to achieve stable and quick response to perturbations [3]. After being tuned, PID controllers use the difference between the current CV value and its setpoint to adjust actuators to maintain the CVs at their target value. The PB-HTGR control system utilizes single-level PID controllers to control a CV directly and cascaded PID controllers to control a CV through an intermediate variable (IV). An open-loop test was performed in a previous study to identify the optimal MV/CV pairings, as shown in Table 1 [36]. In the current settings, the control rod position directly controls the reactor thermal power, which influences the outlet reactor temperature. The circulator pump speed directly changes the mass flow rate in the primary loop and the OTSG heat transfer, which in turn impacts the steam pressure through the water/steam thermodynamic properties. The feedwater pump speed directly controls the mass flow rate in the secondary loop and impacts the OTSG heat transfer, which controls the steam temperature. Finally, the turbine steam valve and condenser pump speed controllers directly control the system electrical power and condenser pressure, respectively.

To evaluate the control system, a 100%–25%–100% load-following scenario is performed at the maximum rate of 5%/min [37]. Specifically, the load-following scenario is performed with 1250 s at 100% electrical power, down-ramp from 100% to 25% electrical power over 900 s, 1500 s at 25% electrical power, up-ramp from 25% to 100% electrical power over 900 s, and finally 1250 s at 100% electrical power. After performing the simulation, the electrical power is shown in Figure 1 with the actual variable value shown in green and the desired variable value shown in red. To accomplish the load-following operation, the control system changes each MV in the same direction as the electrical power. When beginning the down-ramp portion of the transient from 100% to 25%, the turbine steam valve begins to close, redirecting a larger proportion of steam towards the reheater. With less steam entering the turbine, the electrical power begins to fall and the feedwater temperature increases. Increasing the feedwater temperature decreases feedwater density, which decreases the mass flow rate on the secondary. The decreased secondary mass flow rate causes the heat transfer and pressure drop across the OTSG to decrease. Decreasing the heat transfer across the OTSG causes the steam temperature to decrease and the cold helium temperature to increase. Increasing the cold helium temperature decreases helium density and decreases the mass flow rate in the primary. A decrease in the primary mass flow rate reduces the heat removal from the fuel which causes the overall core temperature to increase and the reactor power to decrease due to reactivity feedback. To maintain the CVs at their respective setpoints, the circulator pump speed decreases to combat the increase in steam pressure and the feedwater pump speed decreases to combat the decrease in steam temperature by manipulating the heat transfer across the OTSG. To control the reactor outlet temperature, the control rods begin to be inserted into the core to maintain its setpoint.

To have more confidence in the modeled physics within the Simulink PB-HTGR model, the Simulink load-following results are compared to those presented in [36]. The work in Brits [36] modeled their PB-HTGR system using SimuPACT to conduct the same load-following scenario and the Simulink response is shown to produce the same MV and CV trends. SimuPACT is an integrated software platform developed by SimGenics that enables the development of simulators for power and process plants. SimuPACT utilizes Flownex as its flow solver to accurately simulate the transients and the corresponding system response. Flownex is an NQA1 quality assurance flow solver with nuclear accreditation that utilizes point kinetic equations coupled with thermal-hydraulic models to capture reactivity feedback during a transient. More detail concerning the performance of the Simulink model and comparisons with the SimuPACT model can be found in [38]. Since the Simulink model represents the modeled physics well throughout the load-following scenario when compared to the SimuPACT results, then one can conclude that the PB-HTGR Simulink model is an appropriate training environment for the CSO framework.

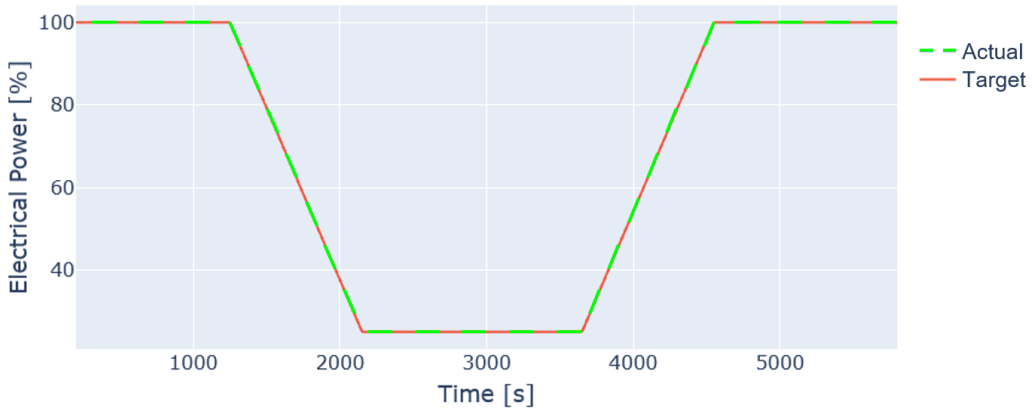

Figure 1. PB-HTGR electrical power during a 100%–25%–100% load-following scenario. The red curve is the desired electrical power and the green curve is the actual output.

2.3. Interactive Environment for RL Agent

The CSO framework in this study is proposed to leverage an RL agent to substitute traditional PID controllers. The main focus is on the load-following scenarios where the objective is to regulate the power output from 100% to a random target level between 50% and 100% within a 500 s timeframe. This characteristic stems from our decision to prioritize a fixed time frame over a constant rate of change, which results in the maximum rate of change occurring when the end level reaches 50%. The RL agent will be trained on the data generated from simulating this scenario, enabling it to learn an optimal control strategy that dynamically adapts to various operating conditions.

The PB-HTGR environment used in this problem is based on the FFNN-based SAM/Simulink surrogate model outlined previously. The framework utilized for creating the environment is the OpenAI Gym package [39]. A single operation will be simulated for 2500 s with a 1.0 s time interval. The average computational time required to complete a full operation is 750 s when using a single core on the Dual AMD EPYC 7452 processor. In addition to the electric power load-following, setpoints are configured for CVs during operations, as displayed in Table 2.

Table 2. Reference setpoints during load-following operations.

Variable	Setpoint
Outlet reactor temperature (°C)	750
Inlet reactor pressure (MPa)	6.0
Steam temperature (°C)	566
Steam pressure (MPa)	16.7

The PB-HTGR environment follows the standard RL scheme where the agent interacts with the environment by providing actions and receiving the corresponding state updates and rewards. Mathematically, the agent's action is represented as a five-dimensional vector $\mathbf{a}$ normalized within the range $[-1.0, 1.0]$, denoted by $\mathbf{a} = [a_1, a_2, a_3, a_4, a_5]$, where each element corresponds to a specific control signal for the MVs detailed in Table 3. Instead of directly using the control rod position, the system employs its corresponding reactivity value as input. This conversion, along with normalization, ensures that the signals remain within a suitable range and stabilize the performance.

Table 3. Actions for manipulated variables in the interactive PB-HTGR environment.

Manipulated Variable	Change per Action	Upper Limit	Lower Limit
Reactivity insertion	$[-4\times 10^{-6}, 4\times 10^{-6}]$	1.340×10^{-3}	-0.720×10^{-3}
Circulator pump speed (rpm)	$[-1.0, 1.0]$	3948	1002
Feedwater pump speed (rpm)	$[-0.6, 0.6]$	3571	3202
Condenser pump speed (rpm)	$[-0.6, 0.6]$	3447	3290
Turbine control valve	$[-5.0\times 10^{-4}, 5.0\times 10^{-4}]$	0.77466	0.51249

The PB-HTGR environment defines its state as a comprehensive vector, $\mathbf{s} = [\mathbf{s}_{\text{CV}}, \mathbf{s}_{\text{actions}}, \mathbf{s}_{\text{errors}}, \mathbf{s}_{\text{cumulative}}]$, encompassing several key elements:

- Controlled variables: $\mathbf{s}_{\text{CV}} = [CV_1, CV_2, \ldots, CV_n]$, where CV_i denotes the i-th controlled variable. These variables serve as the primary outputs of the underlying SAM/Simulink surrogate model and provide insights into the system's current state.
- Recent actions taken: The environment incorporates the agent's most recent control actions into the state vector, allowing the agent to learn the impact of its past decisions. Mathematically, it can be expressed as $\mathbf{s}_{\text{actions}} = [a_1, a_2, \ldots, a_m]$, representing the agent's most recent control actions.
- Absolute errors and their cumulative values: To capture the deviation from desired operating conditions, the state vector includes the absolute errors between the CVs and their reference points. $\mathbf{s}_{\text{errors}} = [e_1, e_2, \ldots, e_n]$. along with the cumulative sum of these errors over time, $\mathbf{s}_{\text{cumulative}} = [c_1, c_2, \ldots, c_n]$. This information helps the agent gauge the performance of its control strategy and identify potential deviations.

The "reset" function of the PB-HTGR environment initiates each episode anew, resetting all internal states and randomly selecting a power level between 100% and 50%. This variability in initial conditions challenges the RL agent to adapt and develop effective control strategies that are versatile across diverse operating demands. Finally, to ensure efficiency during RL agent training, the PB-HTGR environment enforces two termination conditions:

- Manipulated and controlled variable limit violation: If any of the MVs listed in Table 3 or CVs listed in Table 4 surpass their predefined upper or lower limits, the agent terminates. This limit prevents potentially unsafe operating conditions and protects the system from damage.
- Excessive relative error: If the relative error between the actual and setpoint values of the desired CV exceeds 20%, the agent terminates. This criterion ensures that the agent maintains acceptable performance and prevents significant deviations from optimal operating points. The choice of 20% as the threshold is deliberate; it strikes a balance between allowing the agent to learn before termination and preventing excessively long training periods due to overly strict termination criteria.

Table 4. Identified controlled variables with respective threshold from the IEC 45-1:1991 Standard [40].

Controlled Variables	Threshold	Weighting Factor ω
Reactor power (MW)	202	-
Outlet reactor temperature (°C)	770	0.025
Inlet reactor pressure (MPa)	6.3	0.01
Inlet reactor mass flow (kg/s)	86	-
SSSG [1] Outlet temperature (°C)	570	0.05
SSSG [1] Outlet pressure (MPa)	17.3	0.01
SSSG [1] Inlet mass flow (kg/s)	8.4	-
Electrical power (%)	-	0.5
Condenser pressure (MPa)	-	-

[1] SSSG: Secondary side steam generator.

Reward Function

The reward function within the framework of RL acts as guidance for the agent, directing its learning process and influencing its actions to reach desired goals. Reward algorithms in this framework aim to minimize the gap between the current state and the desired state, akin to minimizing the difference between the surrogate model's output and the setpoint values. Additionally, the agent strives to achieve goals with minimal action taken, leading to a penalty term that discourages excessive action across all control variables, promoting efficient and goal-oriented learning. Moreover, the reward system encourages the agent to prolong its survival, ensuring a non-zero reward at each time step. The termination condition marks the worst-case scenario, while successful survival until the end yields a substantial reward. Combining with the termination conditions previously mentioned, the reward function for state $\mathbf{s_t}$ can be formulated as follows:

$$R_t = \exp\left(-\sum_i \omega_i e_i^2\right) - 0.2\sum_t a_t^2 - 100F + 1000M, \quad (2)$$

where:

- e_i: error between the actual value and the desired value for controlled variables i, time-dependent t needs to be added;
- ω_i: weighting factor for each CV as shown in Table 4;
- $\mathbf{a_t}$: action taken at time step t, in the range $[-1.0, 1.0]$.

The F and M terms in Equation (2) are step functions defined as

$$F = \begin{cases} 1 & \text{if termination condition is satisfied} \\ 0 & \text{otherwise} \end{cases}, \text{ and}$$

$$M = \begin{cases} 1 & \text{if the agent reaches the end of the simulation (i.e., 2500 s)} \\ 0 & \text{otherwise} \end{cases}.$$

The weights in Equation (2) and Table 4 are determined through a process of trial and error, considering both the significance and sensitivity of the variables. These weights are iterated over numerous initial runs before converging to optimal values.

2.4. Soft Actor–Critic Agent

As part of the nature of the CO problem, the most tedious and challenging issue of RL is to deduce an optimized policy function that ensures that maximum rewards are achievable for all given states. Introduced in recent studies [41], the SAC algorithm was utilized in this study to find the optimal policy function. Unlike traditional RL algorithms, such as Q-learning and policy gradient methods, SAC offers several distinct features that make it well suited for various applications in complex environments. Notably, SAC excels in achieving a balance between exploration and exploitation, and it efficiently handles continuous action spaces. Unlike algorithms such as deep Q-networks (DQN) and proximal policy optimization (PPO), which frequently encounter challenges with high-dimensional and continuous action spaces [42]. The structure of the SAC agent used in this study is described in Figure 2. It is noteworthy that the SAC architecture utilizes two critic networks to improve stability and mitigate the overestimation of the critic value. Moreover, during actor updates, multiple critics provide additional feedback on the quality of actions, allowing the actor to explore a wider range of actions and learn more effectively. Additionally, a target value network is employed to prevent the value network from being solely affected by the policy updates. As a result, a stable reference for the value updates will make the training process smoother.

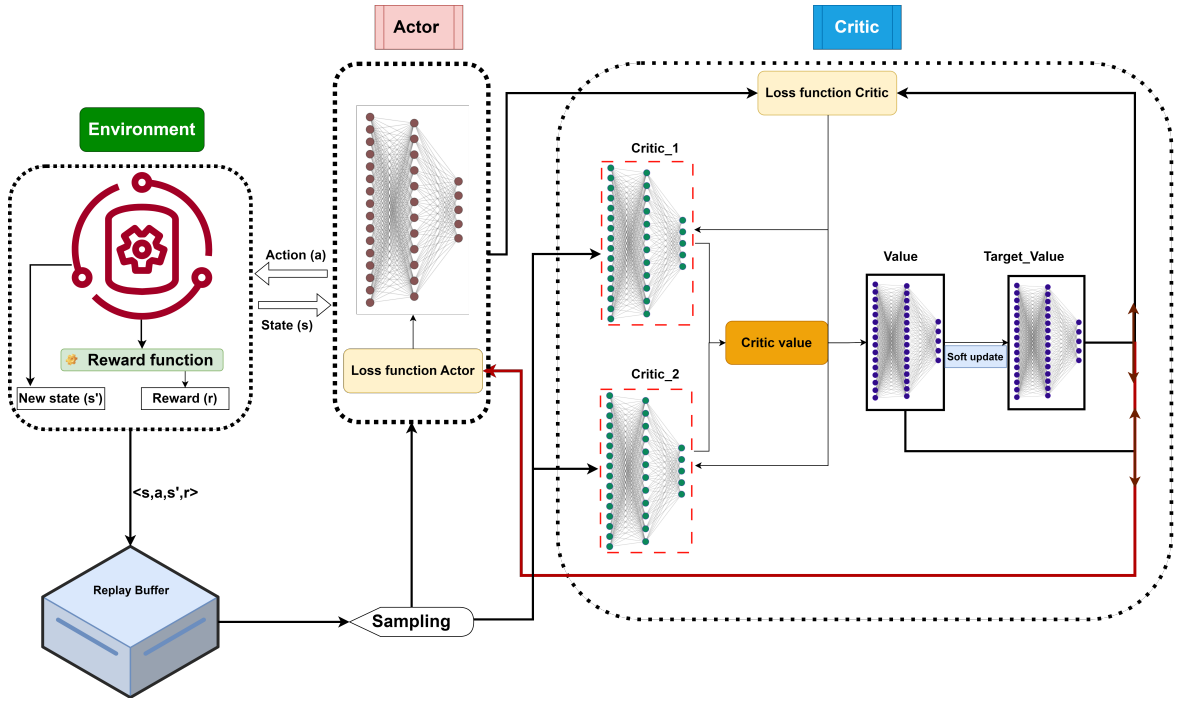

Figure 2. SAC agent architecture used in this study.

Finally, the hyperparameter settings (model complexity, learning rate, action noise, etc.) for the SAC agent are determined through a comprehensive grid search, aimed at identifying optimal values within the hyperparameter space. This process systematically explores various combinations of hyperparameters to assess their impact on the agent's performance. The resulting hyperparameter configuration is given in Table 5.

Table 5. SAC agent training parameters.

Parameter	Value
Actor–network structure	$256 \rightarrow 256 \rightarrow 5$
Learning rate	1×10^{-4}
Critics network structure	$256 \rightarrow 256 \rightarrow 5$
Total episode	10^4
Batch size	256
Discount factor γ	0.99
Action noise	0.2
Buffer memory	10^6

SAC Agent Training

The RL-based CSO framework employs a single-agent training scheme, which implies that there is only one agent that interacts with the environment. Utilizing this approach simplifies the implementation and debugging process. Moreover, this approach also reduces the computational budget, making it attractive for an investigatory study like this one. While this scheme can be simpler to implement and is effective for straightforward tasks, multi-agent training offers significant advantages in terms of scalability and computational efficiency. Therefore, this approach will be adopted in future work.

Figure 3 shows the cumulative rewards during the training process. The training concluded after approximately 2500 episodes due to an early stopping condition being met. This condition terminates the training once the agent achieves a cumulative reward (R in Equation (1)) of 3000 or more for 10 consecutive episodes, indicating successful convergence towards the desired performance. Furthermore, the average survival time of the agent throughout the operation (in the unit of operation time) is depicted in Figure 4. It can

be seen that, initially, the training was terminated almost immediately; however, after 100 episodes, the agent began to accumulate knowledge from previous failures. By the end of 500 episodes, the mean survival time of the agent stabilized between 1500 s and 2500 s.

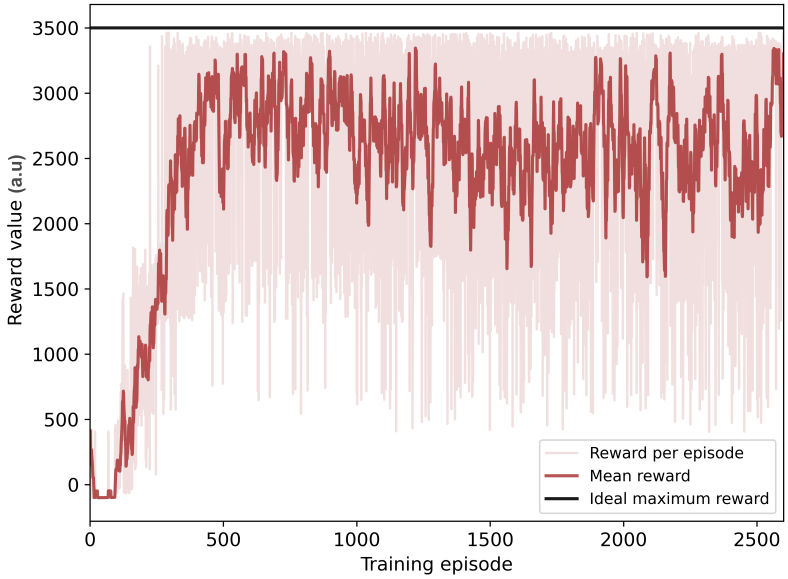

Figure 3. Learning curve of the SAC agent in the CSO framework for PB-HTGR load-following operations.

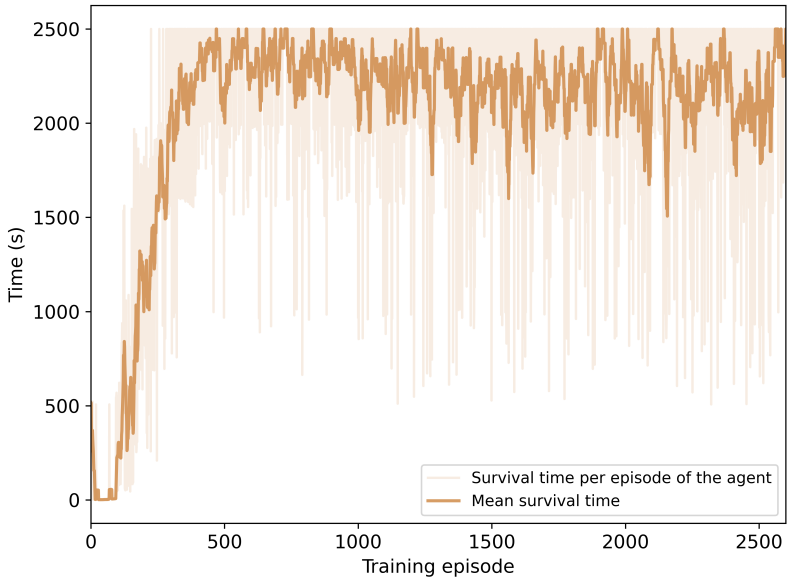

Figure 4. Number of steps that the SAC agent survived during training.

3. Results and Analysis

To assess the effectiveness of the trained RL agent, it is tested on a load-following operation similar to but different from the training scenario, as shown in Figure 5. The transient begins with the reactor operating at full power, followed by a 500 s ramp change to 80% of full power. After a stable period over 500 s, the power demand increases linearly back to full power again. The success of the operation conducted by the agent is measured by the level of agreement of the actual electric output compared with the demand curve throughout the 2500 s of operation.

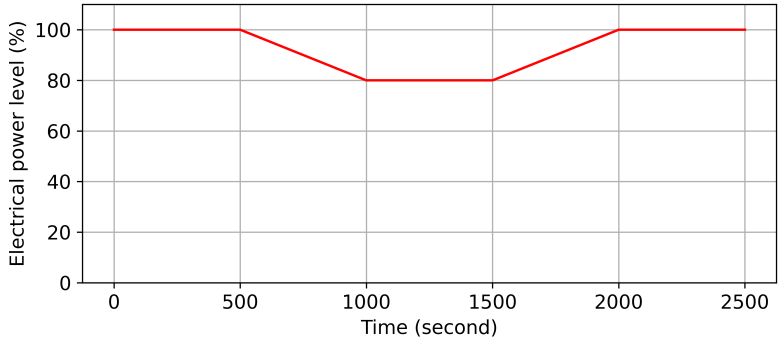

Figure 5. Test scenario of the load-following operation.

Figures 6 and 7 offer insights into the agent's control strategy during the operation. Figure 6 depicts the adjustments made to the reactor's reactivity (black line), which plays a crucial role in regulating power levels. The initial non-zero reactivity value reflects the external reactivity with a value of 0.00131 (i.e., from control rod) required to maintain 100% power in the beginning. It remains constant to ensure operation within the standard range until a decline in target power level is noticed at 500 s. The agent responds to these changes by adjusting reactivity. As the power decreases, the reactivity becomes more negative, reaching a value of 0.00101 after 1000 s. At this moment, with the power level sustained at 80%, the reactivity exhibits initial oscillations before stabilizing at 0.00101. Upon the power's increase towards full capacity, less reactivity is introduced, leading to a gradual shift towards a more positive value. Eventually, the reactivity stabilizes at 0.00130. This observation aligns with the trend displayed in electrical power load change, as reactivity has a direct impact on both the outlet reactor temperature and reactor power, as outlined in Table 1.

Additionally, the dash-line in Figure 6 illustrates the corresponding changes in the control valve opening fraction, which dictates steam consumption. The trend closely follows that of the inserted reactivity, initiating at 0.75557, declining to 0.72694 to reduce the power level to 80%, and eventually reaching 0.75891 upon achieving full power. The adjustments made by the SAC agent to the control valve opening ratio demonstrate its ability to effectively manipulate this critical variable to achieve the desired power profile, given its direct influence on the electrical load through controlling the steam output.

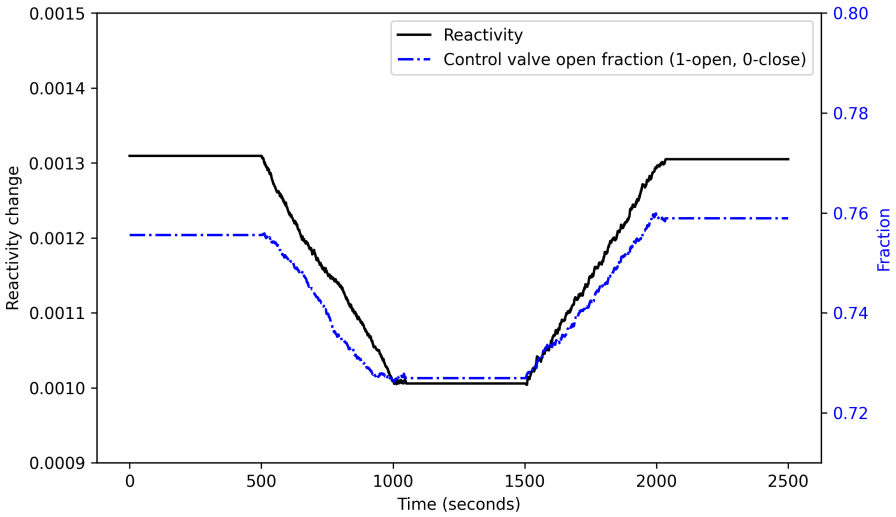

Figure 6. Temporal changes in reactivity and control valve.

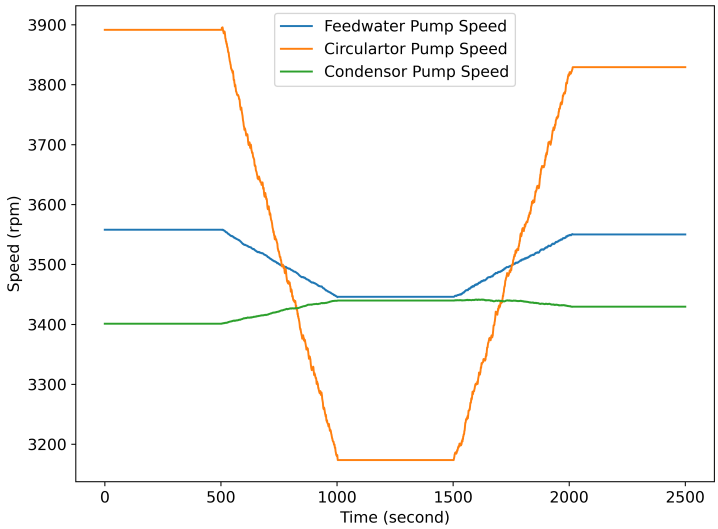

Figure 7. Pump speed changing during test operation.

Figure 7 illustrates the pump speed-controlling signal generated by the agent throughout the test. The adjustments made by the agent to the feedwater pump and circulator pump exhibit a trend similar to that of the electrical load, which results from their influence on steam temperature and pressure, respectively. Notably, the pump speed of the feedwater pump experiences a larger change compared to the circulator pump because of the greater sensitivity of steam temperature relative to pressure. Initially set at 3896 rpm, the feedwater pump's speed undergoes a linear decrease over 500 s at a rate of 0.868 rpm/s, eventually stabilizing at 3875 rpm upon reaching full power level. On the other hand, the speed of the circulator pump begins at 3558 rpm, gradually decreasing to 3493 rpm as the power diminishes and stabilizes at 80%, before ultimately returning to 3556 rpm by the conclusion of the operation. In contrast, the speed of the condenser pump starts at 3401 rpm and experiences a slight increase to 3435 rpm as the power decreases to 80%. It then decreases and stabilizes at 3425 rpm as the power returns to full capacity. This suggests that the condenser pressure may not play a significant role in controlling the output steam, which directly impacts the electrical load. Instead, its effect is more noticeable on the inlet pressure of the steam. Overall, the agent effectively generates control signals for pump speeds to ensure sufficient coolant circulation, thereby contributing to operational stability while maintaining safety constraints within acceptable ranges. One may notice that while slight discrepancies exist between the initial and final values for the 100% power transition, it indicates that there are several ways to achieve a steady state, where the absolute difference between electric power and the set-point value is below 0.05% in this context. With these control signals in place, the desired MVs with their respective setpoints are analyzed.

As depicted in Figure 8, the electric power closely followed the desired trend throughout the test, with a maximum deviation of less than 0.5%, demonstrating the agent's effectiveness in accurately tracking the intended power profile. The reactor outlet temperature remained tightly controlled within a ± 1.5 °C band around the setpoint, showcasing the agent's ability to precisely manage the reactor's thermal behavior, as shown in Figure 9.

Furthermore, the steam pressure was effectively regulated, maintaining values close to their respective setpoints with absolute errors of 0.1 MPa, as depicted in Figure 10, highlighting its capability to keep the monitored variables closed to the setpoint without violating safety limits. Finally, the RL framework empowers the agent to maintain the temperature within a ± 1.5 °C range of the setpoint as shown in Figure 11. This indicates the agent's success in effectively controlling the steam temperature and pressure, despite the inherent difficulty posed by their high sensitivity to changes in pump speeds and heat transfer across the OTSG.

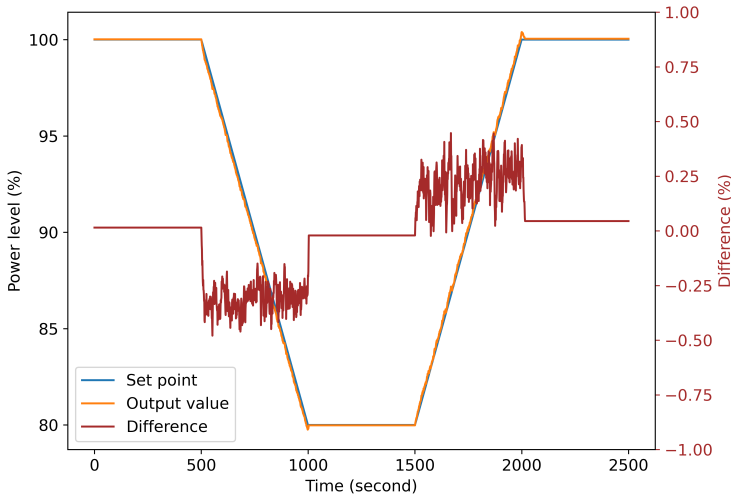

Figure 8. Comparison of the actual and target trend of electric output.

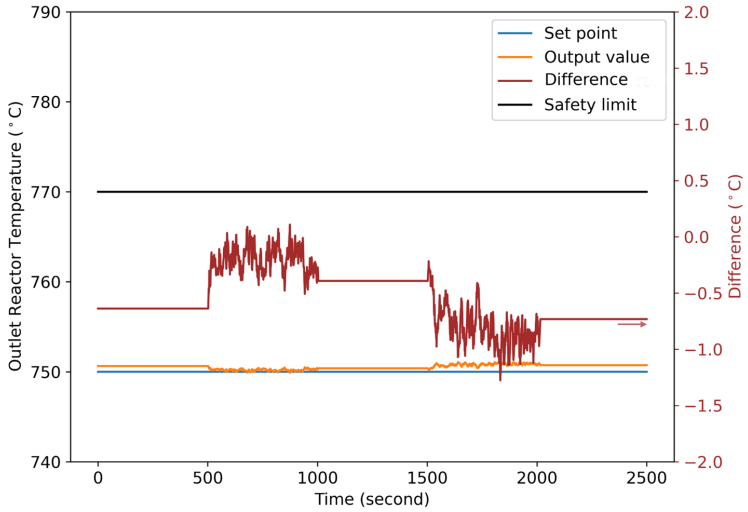

Figure 9. Outlet reactor temperature comparison during the test load-following operation.

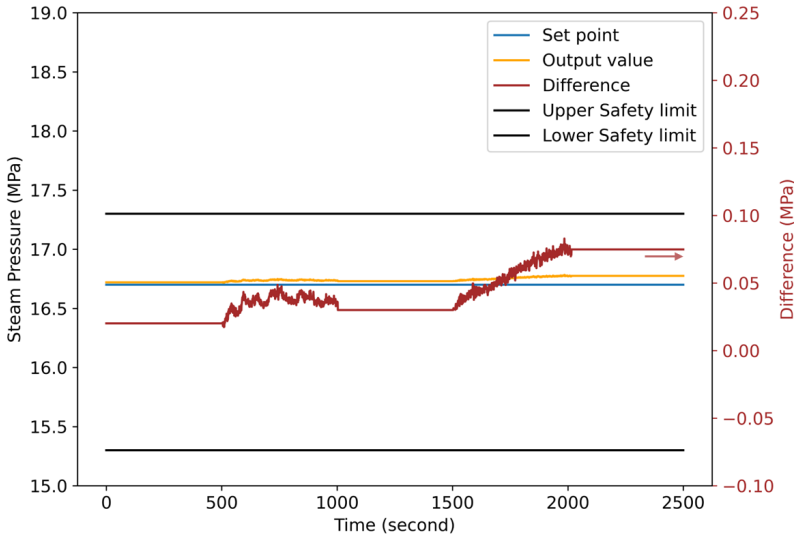

Figure 10. Steam pressure comparison during the test load-following operation.

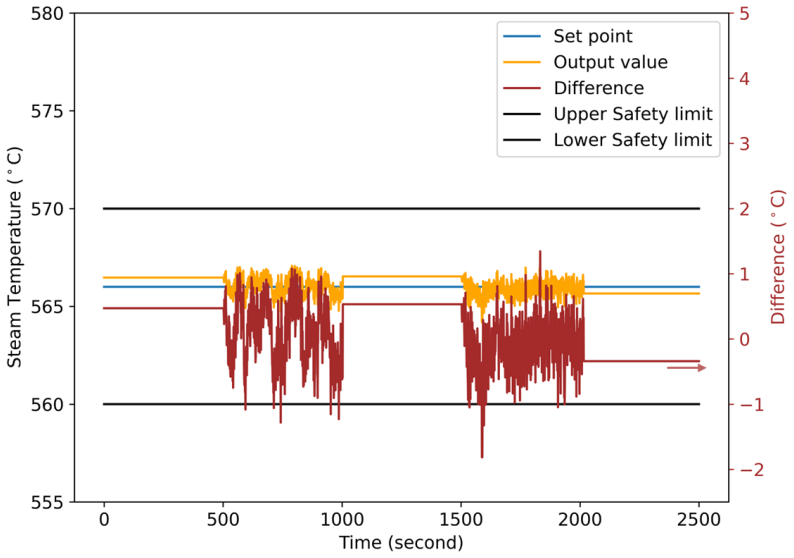

Figure 11. Steam temperature comparison during the test load-following operation.

4. Conclusions and Future Work

In this study, a prototype CSO framework that aims to improve the reliability, autonomy, safety, and economical competitiveness of advanced reactor technology was investigated. In its current state, the framework seeks to generate and execute a series of optimized operational actions to transition the PB-HTGR plant through load-following scenarios while maintaining all safety parameters within pre-defined thresholds.

The proposed framework utilizes the SAC RL algorithm and was trained on dynamic system response data from a PB-HTGR digital twin. The agent underwent training on ramp change in electric output between full power and half power, and was subsequently validated on a typical 100%–80%–100% power maneuver scenario. The control signals generated by the agent effectively maintained all MVs within a tight range of their respective setpoints, demonstrating the framework's capability for precise control during load-following operations. It was able to match electric output to the target value within a 0.5% deviation while adhering to safety thresholds. By precisely adjusting CVs as reactivity, valve positions, and pump speeds, it maintained the reactor temperature within a tight band of ± 1.5 °C and steam pressure within 0.1 MPa of their respective safe operating limits. This is indicative of the framework's ability to enhance the overall system safety by widening safety margins while maintaining stable operation. It also showcases its capability to manage intricate relationships between variables in a high-dimensional space in complex systems like HTGRs, outperforming single-variable control approach such as PIDs. Additionally, unlike traditional CO solver such as GA and SA, RL offers the potential for transfer learning—the ability to train an agent on one system and apply that knowledge to a similar system. This would significantly decrease the time required for training RL agents for a new system in comparison to the thorough fine-tuning required for PIDs. Overall, this suggests that the CSO framework is highly promising for implementing autonomous control systems, leveraging its adaptability and learning capabilities to enhance operational efficiency and effectiveness.

While the developed RL framework using the SAC agent demonstrates success in smoothly transitioning HTGR through simple load-following scenarios, additional studies are necessary before practical implementations are possible. The most desired improvement lies in expanding the controllable power range. This could be achieved by dynamically adjusting the maximum values for control variables at each timestep, allowing them to adapt to a wider operating range while adhering to safety and operational constraints. In addition, the current approach to handling the continuous/discrete action space can be enhanced by implementing a more robust scheme based on the nature of each controllable component. This could potentially improve the system's adaptability and performance in

diverse operating scenarios for broader applications in the future. Finally, the feasibility of adopting transfer learning for control system development using the RL framework requires further studies.

Author Contributions: Conceptualization, J.H. and K.H.N.N.; Methodology, A.R. and K.H.N.N.; Writing—original draft preparation, K.H.N.N.; Writing—review and editing, K.H.N.N., G.K.D., A.R. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work partially supported by the National Science Foundation (NSF) Graduate Research Fellowship under Grant No. DGE-2137100. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of NSF.

Data Availability Statement: The data are not publicly available.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lomonaco, G.; Mainardi, E.; Marková, T.; Mazzini, G. Approaching Nuclear Safety Culture in fission and fusion technology. *Appl. Sci.* **2021**, *11*, 4511. [CrossRef]
2. Gomez-Fernandez, M.; Higley, K.; Tokuhiro, A.; Welter, K.; Wong, W.; Yang, H. Status of research and development of learning-based approaches in nuclear science and engineering: A review. *Nucl. Eng. Des.* **2020**, *359*, 110479. [CrossRef]
3. Kofinas, P.; Dounis, A. Online tuning of a PID controller with a fuzzy reinforcement learning MAS for flow rate control of a desalination unit. *Electronics* **2019**, *8*, 231. [CrossRef]
4. Alphonsus, E.; Abdullah, M. A review on the applications of programmable logic controllers (PLCs). *Renew. Sustain. Energy Rev.* **2016**, *60*, 1185–1205. [CrossRef]
5. Agency, I. *Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants*; International Atomic Energy Agency: Vienna, Austria, 2016.
6. Rivas, A.; Delipei, G.; Satyan, B.; Davis, I.; Hou, J. Preliminary Investigation on Multivariate Control Scheme and Optimization for Advanced Reactors. In Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2023), Niagara Falls, ON, Canada, 13–17 August 2023.
7. Monmasson, E.; Cirstea, M. FPGA Design Methodology for Industrial Control Systems—A Review. *IEEE Trans. Ind. Electron.* **2007**, *54*, 1824–1842. [CrossRef]
8. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
9. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
10. Buşoniu, L.; Bruin, T.; Tolić, D.; Kober, J.; Palunko, I. Reinforcement learning for control: Performance, stability, and deep approximators. *Annu. Rev. Control* **2018**, *46*, 8–28. [CrossRef]
11. Littman, M. Markov Decision Processes. *Int. Encycl. Soc. Behav. Sci.* **2001**, 9240–9242.
12. Kober, J.; Bagnell, J.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
13. Campbell, M.; Egerstedt, M.; How, J.; Murray, R. Autonomous driving in urban environments: Approaches, lessons and challenges. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2010**, *368*, 4649–4672. [CrossRef] [PubMed]
14. Kiran, B.; Sobh, I.; Talpaert, V.; Mannion, P.; Sallab, A.; Yogamani, S.; Pérez, P. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 4909–4926. [CrossRef]
15. Rocchetta, R.; Bellani, L.; Compare, M.; Zio, E.; Patelli, E. A reinforcement learning framework for optimal operation and maintenance of power grids. *Appl. Energy* **2019**, *241*, 291–301. [CrossRef]
16. Parvez Farazi, N.; Zou, B.; Ahamed, T.; Barua, L. Deep reinforcement learning in transportation research: A review. *Transp. Res. Interdiscip. Perspect.* **2021**, *11*, 100425. [CrossRef]
17. Li, J.; Liu, Y.; Qing, X.; Xiao, K.; Zhang, Y.; Yang, P.; Yang, Y. The application of deep reinforcement learning in coordinated control of nuclear reactors. *J. Phys. Conf. Ser.* **2021**, *2113*, 012030. [CrossRef]
18. Lee, D.; Koo, S.; Jang, I.; Kim, J. Comparison of Deep Reinforcement Learning and PID Controllers for Automatic Cold Shutdown Operation. *Energies* **2022**, *15*, 2834. [CrossRef]
19. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.
20. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement Learning for Combinatorial Optimization: A Survey. *arXiv* **2020**, arXiv:2003.03600.
21. Sastry, K.; Goldberg, D.; Kendall, G. Genetic algorithms. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 97–125.
22. Kirkpatrick, S.; Gelatt, C.; Vecchi, M. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]

23. Sattari, F.; Lefsrud, L.; Kurian, D.; Macciotta, R. A theoretical framework for data-driven artificial intelligence decision making for enhancing the asset integrity management system in the oil & gas sector. *J. Loss Prev. Process Ind.* **2022**, *74*, 104648.
24. Bellman, R. A Markovian decision process. *Indiana Univ. Math. J.* **1957**, *6*, 679–684. [CrossRef]
25. Filipovska, M.; Hyland, M.; Bala, H. Anticipatory Fleet Repositioning for Shared-use Autonomous Mobility Services: An Optimization and Learning-Based Approach. *arXiv* **2022**, arXiv:2210.08659.
26. Hoskins, J.; Himmelblau, D. Process control via artificial neural networks and reinforcement learning. *Comput. Chem. Eng.* **1992**, *16*, 241–251. [CrossRef]
27. Amin, M.; Khan, F.; Ahmed, S.; Imtiaz, S. A data-driven Bayesian network learning method for process fault diagnosis. *Process Saf. Environ. Prot.* **2021**, *150*, 110–122. [CrossRef]
28. Hu, R.; Zou, L.; Hu, G.; Nunez, D.; Mui, T.; Fei, T. *SAM Theory Manual*; Office of Scientific: Canberra, Australia, 2021; Volume 2.
29. Lewis, E. Chapter 4—The Power Reactor Core. In *Fundamentals of Nuclear Reactor Physics*; Elsevier: Amsterdam, The Netherlands, 2008; pp. 85–113. [CrossRef]
30. Documentation, S. *Simulation and Model-Based Design*; MathWorks: Natick, MA, USA, 2020. Available online: https://www.mathworks.com/products/simulink.html (accessed on 14 June 2024).
31. Cohen, E. Nuclear energy conversion. *Nucl. Sci. Eng.* **1973**, *50*, 183. [CrossRef]
32. Ooi, Z.J.; Zou, L.; Hua, T.; Fang, J.; Hu, R. *Modeling of a Generic Pebble Bed High-Temperature Gas-Cooled Reactor (PB-HTGR) with Sam*; Argonne National Lab. (ANL): Argonne, IL, USA, 2022. [CrossRef]
33. O'Shea, K.; Nash, R. An Introduction to Convolutional Neural Networks. *arXiv* **2015**, arXiv:1511.08458.
34. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. [CrossRef]
35. Kapernick, J.R. Dynamic Modeling of a Small Modular Reactor for Control and Monitoring. Master's Thesis, University of Tennessee, Knoxville, TN, USA, 2015. Available online: https://trace.tennessee.edu/utk_gradthes/3377 (accessed on 14 June 2024).
36. Brits, Y.; Botha, F.; van Antwerpen, H.; Chi, H.W. A Control Approach Investigation of the Xe-100 Plant to Perform load-following within the Operational Range of 100–25–100%. *Nucl. Eng. Des.* **2018**, *329*, 12–19. [CrossRef]
37. OECD. *Nuclear Energy Agency Technical and Economic Aspects of Load Following with Nuclear Power Plants*; OECD: Paris, France, 2021.
38. Rivas, A. Development of the Dynamic Operation and Maintenance Optimization Framework. Ph.D. Thesis, North Carolina State University, Raleigh, NC, USA, 2024.
39. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
40. *IEC 45-1:1991 Standard*; Steam Turbines—Part 1: Specifications. European Committee for Standardization: Brussels, Belgium, 1991.
41. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2018**, arXiv:1812.05905.
42. Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; et al. Stable Baselines. *Github Repos.* **2018** . Available online: https://github.com/hill-a/stable-baselines (accessed on 12 November 2022).