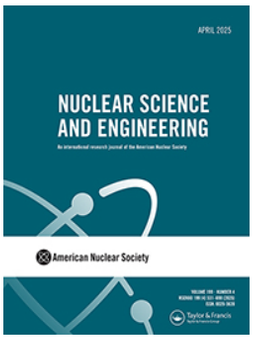

Nuclear Science and Engineering

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/unse20

Multistep Criticality Search and Power Shaping in Nuclear Microreactors with Deep Reinforcement Learning

Majdi I. Radaideh, Leo Tunkle, Dean Price, Kamal Abdulraheem, Linyu Lin & Moutaz Elias

To cite this article: Majdi I. Radaideh, Leo Tunkle, Dean Price, Kamal Abdulraheem, Linyu Lin & Moutaz Elias (12 Feb 2025): Multistep Criticality Search and Power Shaping in Nuclear Microreactors with Deep Reinforcement Learning, Nuclear Science and Engineering, DOI: 10.1080/00295639.2024.2447012

To link to this article: https://doi.org/10.1080/00295639.2024.2447012

Published online: 12 Feb 2025.

NUCLEAR SCIENCE AND ENGINEERING

DOI: https://doi.org/10.1080/00295639.2024.2447012

Multistep Criticality Search and Power Shaping in Nuclear Microreactors with Deep Reinforcement Learning

Majdi I. Radaideh,[a]* Leo Tunkle,[a] Dean Price,[a] Kamal Abdulraheem,[a] Linyu Lin,[b] and Moutaz Elias[c]

[a]*University of Michigan, Department of Nuclear Engineering and Radiological Science, Ann Arbor, Michigan 48109*
[b]*Idaho National Laboratory, Nuclear Science and Technology Division, Idaho Falls, Idaho 83415*
[c]*Lam Research Corporation, Tualatin, Oregon 97062*

Received June 29, 2024
Accepted for Publication December 19, 2024

Abstract — *Reducing operation and maintenance costs is a key objective for advanced reactors in general and microreactors in particular. To achieve this reduction, developing robust autonomous control algorithms is essential to ensure safe and autonomous reactor operation. Recently, artificial intelligence and machine learning algorithms, specifically reinforcement learning (RL) algorithms, have seen rapid increased application to control problems, such as plasma control in fusion tokamaks and building energy management. In this work, we introduce the use of RL for intelligent control in nuclear microreactors. The RL agent is trained using Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C), cutting-edge deep RL techniques, based on a high-fidelity simulation of a microreactor design inspired by the Westinghouse eVinci™ design. We utilized a Serpent model to generate data on drum positions, core criticality, and core power distribution for training a feedforward neural network surrogate model. This surrogate model was then used to guide a PPO and A2C control policies in determining the optimal drum position across various reactor burnup states, ensuring critical core conditions and symmetrical power distribution across all six core portions. The results demonstrate the excellent performance of PPO in identifying optimal drum positions, achieving a hexant power tilt ratio of approximately 1.002 (within the limit of <1.02), and maintaining criticality within a 10 pcm range. A2C did not provide as competitive of a performance as PPO in terms of performance metrics for all burnup steps considered in the cycle. Additionally, the results highlight the capability of well-trained RL control policies to quickly identify control actions, suggesting a promising approach for enabling real-time autonomous control through digital twins.*

Keywords — *Advantage Actor-Critic, criticality search, microreactors, neural networks, proximal policy optimization, reactivity control, reinforcement learning.*

Note — *Some figures may be in color only in the electronic version.*

*E-mail: radaideh@umich.edu

I. INTRODUCTION

Contemporary nuclear reactor designers are working on microreactors with versatile operating scenarios to operate in remote areas, where automated control policies will be essential to reduce operation and maintenance costs.[1] At the same time, advances over the past decade in reinforcement learning (RL) for the optimal control of

multiobjective systems have resulted in algorithms that are more efficient and stable than ever.[2] However, these algorithms may iterate through millions of time steps to find reasonable policies,[3] and in the case of future advanced reactors, the only available data are through simulations. Simulating each time step accurately enough to train precise and safe control policies must therefore be balanced with minimizing the computational cost of each of these simulations. This paper investigates implementing an accurate and efficient surrogate model to enable RL to produce control policies for microreactors.

Historically developed and tested by finding winning strategies in complex game environments, RL algorithms have seen an exponential increase in their applications to control problems, including the control of error-prone operation phases in pressurized water reactors,[4] plasma control in fusion tokamaks,[5] and building energy management,[6] among others. In nuclear reactor control, the recent study by Chen and Ray[7] developed a nonlinear reactivity controller system for a boiling water reactor using an RL algorithm called a deep deterministic policy gradient. The study found RL more robust compared to another control approach H_∞ under parameter perturbations and exogenous disturbances. Two other studies focused on using RL to control pressurized water reactors and obtained their training data from Korea Atomic Energy Research Institute's compact nuclear simulator, which represents the APR-1400 reactor within certain operating parameters.[8] In the first study, Park et al. trained the asynchronous Advantage Actor-Critic (A2C) RL algorithm to adjust two control valves during the heatup mode to achieve hard-coded pressure and temperature profiles over time.[9] In the second study, Bae et al. used a combination of the Soft Actor-Critic RL algorithm and the hindsight experience replay technique to learn a control policy for four valves and a heater to achieve pressure, volume, and temperature targets for the heatup mode while staying within safe operating limits.[4] Both of these studies were feasible only with the availability of the computationally efficient compact nuclear simulator model.

To achieve control over the nuclear steam supply system in a modular high temperature gas-cooled reactor, which cannot be directly controlled, Dong et al. successfully used a multilayer perceptron–based RL controller on several lower-level, indirect controllers.[10] In a broader multiobjective optimization study of nuclear renewable integrated systems, Yi et al. demonstrated the superiority of three deep RL algorithms over traditional control methods in maximizing revenue in a time-varying, heterogeneous energy system.[11] For nuclear reactor design optimization, which can be perceived as a static problem, several studies have demonstrated real-world applications of the Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) RL algorithms in assembly design optimization on small scales[12] and large scales[13] along with an open-source implementation framework.[14]

Certain nonnuclear power applications illustrate the potential of RL in our own field. Degrave et al. achieved tokamak feedback control to maintain various, nontrivial plasma shapes using maximum a posteriori policy optimization, another actor-critic RL algorithm.[5] Another study by Arroyo et al. combined DQN with model predictive control to learn a strategy for efficient energy use in climate control, subject to considerations of building usage, time-dependent electricity costs, and the thermal inertia of the system.[6] This hybrid controller took advantage of RL to find optimal heating strategies in a complex solution space and model predictive control to maintain constraints without requiring an unreasonable number of training episodes.

This paper addresses a steady-state optimization and control problem for a Westinghouse eVinci™–motivated heat pipe microreactor design across three different burnup levels. We compare two RL algorithms, A2C and PPO, in learning the optimal progression of control drum positions across different burnup states to achieve criticality while maintaining a balanced power distribution. To enable this comparison on a reasonable timescale, we develop a surrogate model of this system based on high-fidelity Monte Carlo simulations using Serpent.[15]

This study offers two key contributions: demonstrating a multiobjective RL achieving a multi-burnup-step criticality search and power flattening in conjunction with a surrogate model and demonstrating that this can be accurately done on short enough timescales to be usable within a digital twin control system.

For the remaining sections of this work, Sec. II presents the theoretical background behind the RL algorithms employed in this work. The methodology section, Sec. III, details how the surrogate model was trained and utilized as an RL environment as well as how hyperparameters and the reward function were selected for RL algorithms. Section IV describes the performance of the surrogate model and the RL algorithms used. Further context, limitations of this work, and avenues for further investigation are found in Secs. V and VI, respectively.

II. REINFORCEMENT LEARNING THEORY

Reinforcement learning aims to find a strategy that chooses the best action in a given situation that will

maximize a long-term reward. Within the RL field, this strategy is called a policy, and the situation is a state. An agent performs actions according to the policy within an environment. Formalism in this field begins by idealizing RL as a Markov decision process (MDP), for which an action A_t performed in state S_t results in a new state S_{t+1} with a certain probability along with a corresponding reward R_{t+1}. A trajectory of states and actions looks like $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, S_3$, etc.

A finite trajectory is called an episode. For nonfinite tasks, the trajectory must be truncated at some point to create an episode. Key to an MDP is the Markov property, which requires that each state have enough information for the rest of the trajectory to proceed without referencing previous states. For instance, the trajectory progression after A_2 is taken in S_2 should not need to account for S_0 or S_1. This property motivates and allows for several recursive relationships to be defined. The first of these is the return, commonly denoted G and representing the total rewards that will accumulate from the trajectory at a given state.

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \ldots) \\ &= R_{t+1} + \gamma G_{t+1} \ , \end{aligned} \tag{1}$$

where γ is the discount factor between 0 and 1, but is typically greater than 0.9, and is introduced to determine whether the policy should focus on immediate or long-term rewards. The return expected from a state s following policy π is called the value function:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \ . \tag{2}$$

By convention, uppercase is used to denote random variables while lowercase represents an observation of the variable. A similar action-value function describes the expected return if action a (without regard to π) is taken at state s after which policy π is followed:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \ . \tag{3}$$

While it is possible to have a deterministic policy, $\pi(a|s)$ is interpreted as the probability of taking action α in state s. For any problem, there exists a policy that is better than or equal to every other policy. This is known as an optimal policy. The details of this derivation are outside the scope of this paper, but for an optimal policy, the so-called Bellman optimality equations for the optimal value and action value are

$$\begin{aligned} v_*(s) &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \\ q_*(s, a) &= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')] \ . \end{aligned} \tag{4}$$

Considering every state, with terminal states of finite trajectories defined as having a value of zero, this becomes a system of equations with one equation and one unknown per state or state-action pair. Thus v_* and q_* can theoretically be directly solved for. Once these are known, the optimal policy is simply a matter of choosing the action that maximizes q_* in a given state. However, for all but the simplest problems, this is intractable. Most techniques in RL, therefore, make various approximations to these optimality equations to hopefully find nonoptimal but good policies. The two major categories of these techniques are value- and policy-based methods.

Value-based methods attempt to approximate the value of each state and are the most obvious extension of the Bellman optimality equations. With unlimited time, Monte Carlo methods can calculate returns in trajectory after trajectory and get successively better estimates of each value. However, complex problems tend to have state spaces that are far too large to hold in computer memory, so using a functional approximation of the state value is more practical. An artificial neural network, which acts as a universal function approximator that can improve with experience through backpropagation, serves this purpose well.[16] Deep Q-Learning, for instance, uses a convolutional neural network.[17]

Policy-based methods directly select actions without reference to the value of a state. They are parameterized such that policy parameters, denoted θ, can be updated with the gradient of some performance measure to improve the policy through experience. The canonical policy gradient method REINFORCE[18] updates θ in the direction of $\nabla_\theta \log \pi(a_t|s_t; \theta) G_t$.

A pure policy gradient method tends to be very slow to converge to a good policy because of high variance in how large these updates are. By subtracting an estimate of the state value $V(s)$ from the return G_t, this variance can be greatly reduced. Now θ gets updated in the direction of $\nabla_\theta \log \pi(a_t|s_t; \theta)(G_t - V(s_t))$. This combination of policy- and value-based techniques is known as an actor-critic method. The critic improves its ability to value states, and this value is used within the gradient descent learning process to improve the actor's policy. $G_t - V(s_t)$ is an estimator of the advantage, defined as $\mathcal{A}(a_t, s_t) = Q(a_t, s_t) - V(s_t)$.

Adding an estimate of the advantage to the policy gradient serves to increase the probability of above-average actions and decrease the probability of below-average

actions. Adding an entropy term to the policy gradient to prevent early convergence results in the A2C RL algorithm. Mnih et al. showed that running multiple actor-learners in parallel and merging their updates at regular intervals reduced the training time, stabilized the learning process, and improved the resulting policies.[19] In this paper, we take advantage of this parallelism and use the Stable Baselines3[20] implementation of A2C.

Like all machine learning methods, A2C requires careful hyperparameter tuning. Additionally, while having multiple actor-learners can reduce the frequency of overly large gradient updates through cancellation, the problem is not eliminated completely. In an early attempt to address this issue, Schulman et al. introduced the concept of a trust policy region, in which the policy update is constrained in size by ensuring that the average Kullback-Leibler divergence per state between the new and the old policies is bounded.[21] While this improved stability and reduced the need for hyperparameter tuning, the implementation was complex and incompatible with commonly used RL architectures.

Later, Schulman et al. published a simpler, more general algorithm with the same benefits called PPO.[22] Instead of using a complex trust region, this uses a simple clipping function with parameter ϵ, which prevents the probability ratio of a new policy to the old policy during gradient updates from being below $1-\epsilon$ or above $1+\epsilon$. Additionally, where A2C estimates the advantage with $G_t - V(s_t)$, PPO uses a generalized advantage estimation, which is tunable to trade off between bias and variance in the estimate.[23] Several standard RL benchmark problems demonstrated that PPO usually outperformed other algorithms, even when they had hyperparameter tuning for the specific problem and PPO did not. Since PPO is generally implemented on top of A2C, including in the Stable Baselines3 implementation that we use, we again take full advantage of parallel actor-learners.

III. METHODOLOGY

III.A. Heat Pipe Microreactor Model

In this study, Serpent[15] is used to calculate neutronic quantities associated with a heat pipe microreactor design motivated by the Westinghouse eVinci design. The particular design used in this study will be referred to as the eVinci Motivated Design (EMD) and is selected because its material and geometrical specifications are publicly available.[24] Starting at an initial enrichment of 19.75%, the EMD is designed to operate at 4 MW(thermal) with a 4-year lifetime, and it has 12 control drums, which are used to control reactivity. An illustration of the core is provided in Fig. 1, where the Serpent-rendered core geometry at the midplane is provided alongside a three-dimensional (3D) rendering of the geometry. In this figure, the 12 control drums are shown at a 90-deg orientation in the two-dimensional Serpent-rendered core geometry. These control drums can be rotated to manipulate the reactivity and power distribution of the core.

III.B. Surrogate Modeling

In this analysis, calculating neutronic core characteristics using the Serpent neutronics code would be too computationally expensive for the large number of evaluations required for the RL algorithm. Ideally, the control problem would be integrated into a digital twin that would be fed with real-time sensor data—there would be no need for any direct calculation of these quantities. Therefore, a set of surrogate models was created to predict core properties given a set of control drum positions at a particular cycle time.

The relevant core properties for the control problems considered here are the core multiplicity and hexant power fractions. The hexant power fractions are a set of six quantities that describe the fraction of total power that comes from six portions of the core. The divisions used to calculate these hexant powers are shown in Fig. 1 with red dotted lines. The control drums within each hexant are considered to move together, in opposite directions, such that the positions of the 12 total control drums can be described with a set of six control drum angles.

A nominal core depletion calculation is run in Serpent with all control drums facing 90 deg. Then, three fuel composition states can be identified: 0 YR corresponds to the beginning-of-life fuel composition when the core is loaded with fresh fuel, 2 YR corresponds to the core fuel composition after 2 years of operation, and 4 YR corresponds to the core fuel composition after 4 years of operation. Serpent is used to calculate core multiplicities and hexant powers for 250 control drum positions for each of these three fuel compositions for a total of 750 calculations. The Monte Carlo sampling statistics were selected such that the final uncertainty in core multiplicity was estimated to be about 9 pcm.

Within each fuel composition, symmetries in the core geometry can be exploited to turn any single core calculation into 12 data points. A given core configuration can be rotated five times to produce distinct configurations with the same resulting core multiplicity and hexant

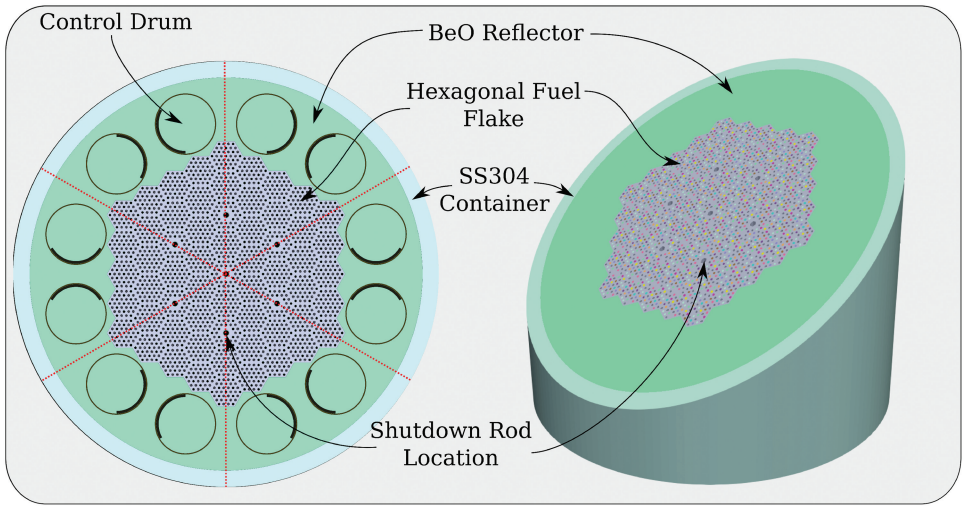

Fig. 1. EMD with control drums oriented at 90 deg, shown with the slice at the core midplane (left) and a 3D rendering with a diagonal cut (right). Hexant divisions are shown with dotted red lines. Control drums are not rendered in the 3D depiction.

power fractions. This is also true for its reflection and subsequent additional five rotations. Further symmetries repeat only these 12 configurations. Thus, the 250 calculations at each burnup are augmented into 3000 data points, which can be used to train and evaluate the surrogate models.

These 3000 data points are divided into a training data set, used to calculate weights and biases associated with the deep neural network (DNN) surrogate model; a validation set, used to select an optimal network architecture; and a testing set, used to evaluate the final performance of the DNN, with a 70%/15%/15% split, respectively. The divisions among these sets did not split data points, which may have come from the same Serpent calculation (i.e., symmetries were applied after data were divided among the three data sets).

For each of the 0 YR, 2 YR, and 4 YR, a single DNN with six inputs (control drum position) and seven outputs (k_{eff}, P_1, P_2, …, P_6) was created to simultaneously predict core multiplicity and six fractional hexant powers given a set of six control drum angles. See Fig. 2. Three parameters were optimized for each DNN: the number of hidden layers between the input and output layers, the number of nodes/neurons per layer, and the learning rate (the step size during each DNN training iteration).

Architecture optimization minimized the mean-squared error when predicting the validation data set independently for each of the three DNNs corresponding to the 0 YR, 2 YR, and 4 YR cases. A constant number of nodes per hidden layer is used to simplify the optimization process. Bayesian optimization is used for this mixed-integer optimization problem with an allowance of 30 search trials.

Details of the final models are found in Table I. The results from this training process, including performance measures of the surrogate models, are discussed in Sec. IV.A.

III.C. RL Environment

Figure 3 illustrates the RL framework used in this work. Originally developed by OpenAI, "Gym" is the most widely used Python library for defining an environment.[25] We use Gym Version 0.19.0. The details of our environment, which consults the surrogate model to compute rewards, are as follows:

1. *Action space* (a_t): The possible control actions upon the environment. While the control drum orientation in each of the six reactor hexants can physically take any value between 0 and 360 deg, we reduced the search space to only discrete integer values between 0 and 180 deg. Here, we take advantage of the symmetry of the drum reactivity worth, which is similar from 0 to 180 deg and 180 to 360 deg. The discretization of the action space will facilitate the training process and reduce training time. Following Gym's nomenclature, we use a multidiscrete object of Size 6 bounded by 0 and 181 (Index 181 is not reached by the agent).

2. *State space* (s_t): The possible information given to the agent upon which to base its next action. There are three parts shaping the state space: the current time step (0 YR, 2 YR, 4 YR), the core multiplicity (k_{eff}), and the power fraction of each hexant ($P_1, P_2, \ldots, P_6$).

3. *Reward* (r_t): Designed to encourage power symmetry in the six hexants and a critical core at

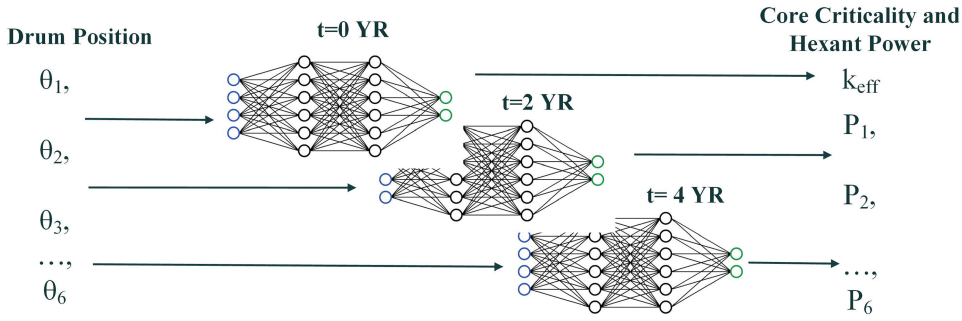

Fig. 2. The surrogate model used in this work to predict core criticality and hexant power (k_{eff} and P_i) based on the drum position (θ_i). Note that three independent neural network models are used to make predictions of the core metrics based on the drum position for each burnup step ($t = 0, 2, 4$ YR). This yielded better performance than a single model for all steps.

TABLE I

Network Architecture and Performance of Surrogate Models Evaluated on the Testing Set*

Parameter	0 YR	2 YR	4 YR
Learning rate	3×10^{-3}	2×10^{-3}	2×10^{-4}
Number of hidden layers	5	5	7
Number of nodes per layer	150	200	191
MAE_{keff}	85 pcm	83 pcm	79 pcm
R^2_{keff}	0.996	0.996	0.998
MAE_P	2.32×10^{-4}	2.33×10^{-4}	2.29×10^{-4}
R^2_P	0.999	0.999	0.999

*The MAE and coefficient of determination R^2 are shown when using the surrogate model to predict core multiplicity k and fractional hexant powers P.

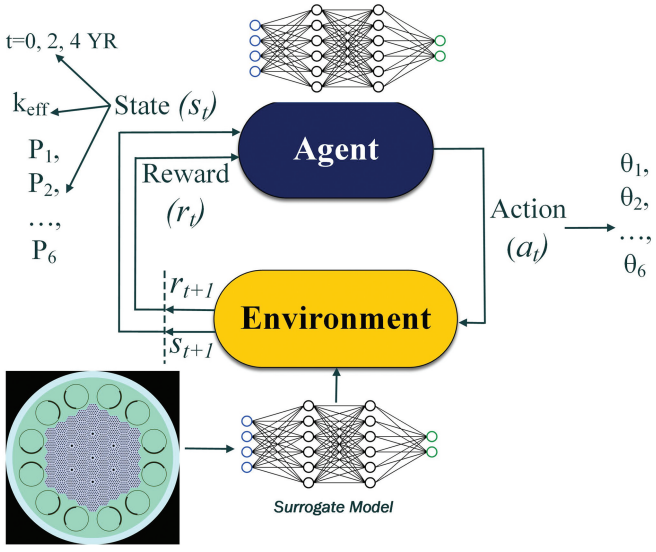

Fig. 3. The RL control framework adopted in this work. The subscripts $1, 2, ..., 6$ refer to the six hexants of the core shown in Fig. 1.

each burnup time step. The reward is calculated using k_{eff} and power fraction outputs from our surrogate model as follows:

$$r_t^0 = \frac{1}{f_1^t + f_2^t + f_3^t}, \quad t = 0, 2, 4 \text{ YR} , \tag{5}$$

where

$$f_1^t = \frac{1}{6}\sum_{i=1}^{6} |P_i - 0.166667| , \tag{6}$$

$$f_2^t = |k_{eff}^t - 1.00000| , \tag{7}$$

and

$$f_3^t = \sqrt{\frac{\sum_{i=1}^{6} (P_i - \overline{P})^2}{6}}, \tag{8}$$

where $\overline{P}$ is the average of the six hexant power fractions. The third term f_3^t was not reported in our PHYSOR paper[26] since its impact on the learning process was negligible. However, we still report that here for completeness. Since RL maximizes rewards, the reciprocal is

taken in Eq. (5) to convert a min to max problem without changing the sign.

Episodes are very short and defined as three successive burnup time steps in which the agent attempts to input the optimal angles for each one. To avoid plotting all six power fractions, we define a hexant power tilt ratio (HPTR) as follows:

$$HPTR = \frac{6 \times max(P_1, P_2, ..., P_6)}{\sum_{i=1}^{6} P_i} \ . \tag{9}$$

This expression is based on a commonly used equivalent for cores with four quadrants and is used only for plotting and monitoring purposes.

The episode always starts with the reactor at time $t = 0$ YR with random k_{eff} and P_i values, which represent the initial state s_1. The agent takes the action a_1 and calculates the reward r_1. The state is then updated to s_2, which is equivalent to $t = 2$ YR. The agent then repeats the same process to get to s_3 at $t = 4$ YR. The goal is to maximize the reward of the episode by taking the optimal action in the three time steps. After the episode terminates, a new episode starts, and the process is repeated many times until the agent learns how to play the game. Because of the sequential nature of the episode, we modify the definition of the reward function in Eq. (5) to

$$\begin{aligned} r_1 &= r_1^0 \ , \\ r_2 &= MEAN(r_1^0, r_2^0) - STD(r_1^0, r_2^0) \ , \\ r_3 &= MEAN(r_1^0, r_2^0, r_3^0) - STD(r_1^0, r_2^0, r_3^0) \ . \end{aligned} \tag{10}$$

We observed that this adjustment to the reward function enables the agent to prioritize learning across all three burnup steps simultaneously rather than becoming biased and focusing on increasing the reward for a single, potentially easier-to-optimize burnup step, which could occur if the reward function in Eq. (5) was issued as is. To achieve an optimal reward, the mean reward across the three burnup steps must be maximized while their standard deviation must be minimized.

III.D. RL Hyperparameters

As with most machine learning methods, RL algorithms require problem-specific hyperparameter tuning. For this work, we performed a grid search among the following hyperparameters described further below: the value coefficient, the entropy coefficient, N_{steps}, the maximum norm of the gradient, and the PPO clipping parameter.

While the actor and critic components of A2C and PPO can be approximated by two separate neural networks, most implementations, including that of Stable Baselines3, employ a single neural network with outputs for both the value and the policy. To improve both at once, the following objective is maximized:

$$L^{total}(\theta) = L^{policy}(\theta) - c_1 L^V(\theta) + c_2 H[\pi_\theta](s_t) \ . \tag{11}$$

The value coefficient c_1 tells the neural network how much importance to give the value portion of the output when updating parameters through backpropagation. Because of the negative sign in front of the value coefficient, the higher it is, the more it hinders the maximization of the combined objective function. A value of 0.5 would weight the policy and value components of the objective function equally, while a value of 0.75 would weight the value component more.

The entropy coefficient c_2 determines the contribution of the entropy term, which prevents the early optimization of the policy before it has fully explored the environment.

The N_{steps} value determines how many steps to learn from before updating the policy. For instance, $N_{steps} = 3$ would update the policy after every episode, which could be too soon and result in an unstable policy; $N_{steps} = 3000$ would update the policy after 1000 episodes, which could be too long to wait and waste computing resources and experience for little improvement.

The maximum gradient norm imposes a limit on how large the policy gradient can be. This prevents the issue of exploding gradients, which causes the policy update step to overshoot and result in an unstable policy. Since PPO is known to be less sensitive to hyperparameters, we chose to tune this only for A2C in favor of tuning PPO's clipping parameter ϵ.

The clipping parameter ϵ is described in Sec. III and is embedded in $L^{policy}(\theta)$. It does not consider the gradient, only the ratio of action probabilities between the current policy and the new policy proposed after a gradient update. $L^{policy}(\theta)$ is defined as

$$L^{policy}(\theta) = \mathbb{E}[\min(r_t(\theta)\mathcal{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\mathcal{A}_t)] \ ,$$

where $r_t(\theta) = \frac{\pi(a_t|s_t;\theta)}{\pi_{\text{old}}(a_t|s_t;\theta_{\text{old}})}$ is the ratio between the new policy and the old policy.

The summary of training the PPO agent is summarized in Algorithm 1, which shows that the agent makes

interactions with the reactor environment to find an optimal control policy that maximizes the reward function defined before in Eq. (10).

Algorithm 1: Training a PPO Policy

1: Initialize policy network $\pi(\theta)$ and value function network $V(\phi)$ parameters.
2: Set hyperparameters: learning rates, discount factor γ, clip range ϵ, c_1, c_2.
3: **for** each iteration **do**
4: Collect trajectories by interacting with the environment using $\pi(\theta)$ for N_{steps}
5: Compute rewards-to-go using Eq.(10).
6: Update the policy network $\pi(\theta)$ to maximize the PPO objective in Eq.(11) using gradient descent.
7: Update the value network $V(\phi)$ to minimize value prediction error
8: Log training metrics
9: **end for**

III.E. RL Simulation Details

All runs were completed with 20 processors running agent-learners in parallel. The surrogate model used Keras-2.6.0,[27] the learning environment was created with Gym-0.19.0, and the A2C and PPO implementations were from Stable Baselines3-1.1.0. The default actor-critic model was used for each implementation and consists of a feedforward neural network with two hidden layers of 64 nodes each. The training process was limited to 1.2 million time steps, which kept the overall training time around 1 h/run.

IV. RESULTS

The results of this paper are presented in Secs. IV.A through V.C. First, the surrogate model of the Serpent simulation is trained and validated. The surrogate model is then used as an environment to train the A2C and PPO RL algorithms. Last, the trained policy is tested to ensure it can identify the optimal drum positions.

IV.A. Surrogate Model Performance

In total, three different surrogate models for 0 YR, 2 YR, and 4 YR are used to supply core multiplicity and hexant powers (as fractions of total power) for a given set of control drum angles. Performance metrics for each of these optimized DNNs after evaluation on a test set of 456 data points are shown in Table I. Let us assume that $k_{\text{eff},i}^{\text{true}}$ is the true value in the test set and $k_{\text{eff},i}^{\text{pred}}$ is its corresponding predicted value using the surrogate model; then we can define the two metrics mean absolute error (MAE) and R^2 for k_{eff} as

$$\text{MAE}_{keff} = \frac{1}{n}\sum_{i=1}^{n}\left|k_{\text{eff},i}^{\text{true}} - k_{\text{eff},i}^{\text{pred}}\right| . \tag{12}$$

The R^2 metric for k_{eff} is defined as

$$R^2_{keff} = 1 - \frac{\sum_{i=1}^{n}\left(k_{\text{eff},i}^{\text{true}} - k_{\text{eff},i}^{\text{pred}}\right)^2}{\sum_{i=1}^{n}\left(k_{\text{eff},i}^{\text{true}} - \overline{k_{\text{eff}}^{\text{true}}}\right)^2} , \tag{13}$$

where

$$\overline{k_{\text{eff}}^{\text{true}}} = \frac{1}{n}\sum_{i=1}^{n} k_{\text{eff},i}^{\text{true}}$$

is the mean of the true values of k_{eff}.

For the hexant powers, the surrogate model produces six different values, each corresponding to a fractional power in the corresponding section of the core. For brevity, the performance metrics pertaining to these hexant powers have been averaged. There is relatively little variation in model performance when predicting the powers in each hexant because symmetry is used to expand the training data. Similarly, if $P_{j,i}^{\text{pred}}$ is the surrogate model prediction of the hexant power j and sample i in the test set and $P_{j,i}^{\text{true}}$ is its corresponding true value in the test set, then the MAE and R^2 metrics are as follows:

The MAE for the power variables $P_1, P_2, \ldots, P_6$ is defined as

$$\text{MAE}_P = \frac{1}{6}\sum_{j=1}^{6}\left(\frac{1}{n}\sum_{i=1}^{n}\left|P_{j,i}^{\text{true}} - P_{j,i}^{\text{pred}}\right|\right). \tag{14}$$

The R^2 metric averaged over $P_1, P_2, \ldots, P_6$ is given by

$$R^2_P = \frac{1}{6}\sum_{j=1}^{6}\left(1 - \frac{\sum_{i=1}^{n}\left(P_{j,i}^{\text{true}} - P_{j,i}^{\text{pred}}\right)^2}{\sum_{i=1}^{n}\left(P_{j,i}^{\text{true}} - \overline{P_j^{\text{true}}}\right)^2}\right) , \tag{15}$$

where

$$\overline{P_j^{\text{true}}} = \frac{1}{n}\sum_{i=1}^{n} P_{j,i}^{\text{true}}$$

is the mean of the true values for P_j.

Overall, these three surrogate models perform quickly and accurately for determining both criticality and power fraction. We leave undetailed the other model types we trained. We used neural networks because of their high performance over linear regression, which performed very poorly, and random forests, which provided $R^2 \sim 0.8$.

IV.B. RL Training Results

After a grid search for each algorithm, the hyperparameters described in Sec. III.D were selected for each algorithm. A2C performed best with a value coefficient of 0.5 ($c_1 = 0.5$), an entropy coefficient of 0.02 ($c_2 = 0.02$), 200 steps before updating the policy ($N_{steps} = 200$), and a maximum gradient norm of 5. For PPO, these were $c_1 = 0.75$, $c_2 = 0.01$, and $N_{steps} = 300$, along with a clipping parameter of 0.4 ($\epsilon = 0.4$).

While a small amount of entropy gave better results compared to the no entropy default, higher-entropy coefficients (such as $c_1 = 0.1$ or 0.2) yielded worse performance in each case, with training failing to converge to a good policy within the 1.2 million time steps. Since both algorithms were run with 20 processors, the effective number of steps before a policy update was $20 \times 200 = 4000$ time steps and $20 \times 300 = 6000$ time steps for A2C and PPO, respectively. A2C was quite sensitive to changes in N_{steps}, with values of 300 or higher resulting in rapidly worse performance.

Training progress plots for both algorithms are shown in Fig. 4. The subplots show how k_{eff}, HPTR, and the reward converge to optimal values. Each of these values was averaged over each episode and over the parallel processors, so they represent the average performance of the 20 parallel agents over the three time steps. Additionally, each point plotted represents statistics from 30 000 aggregated time steps (10 000 episodes), which we define as an epoch.

It is important to note that the reward scale for PPO in Fig. 4c differs slightly from the one originally published in our PHYSOR conference paper.[6] The authors discovered a typo in the source code regarding the use of the absolute value after the mean in Eq. (6), which should be reversed. This did not affect the training process or the performance of PPO in any way. However, the reward scale in Fig. 3 of Ref. [26] differs from Fig. 4c, even though both reward scales yield similar HPTR and k_{eff} values. This note is provided to avoid confusing the reader about having similar reward formulations between the two studies that yielded two different reward values.

For both algorithms, k_{eff} is rapidly improved over the first five epochs while power balancing takes longer to improve. PPO exhibits policy convergence within 25 epochs. Meanwhile, the high variability in rewards obtained by successive policies in later epochs, together with the lower magnitude of rewards compared to PPO, indicates that A2C is unable to converge to an optimal policy within the time step training window.

IV.C. RL Testing Results

Following RL training, we proceeded to evaluate performance with the most recent versions of each policy, asking both the A2C and PPO policies to provide optimal drum positions for each burnup time step. These are displayed in Table II. For every reactor state, the PPO policy was able to find accurate drum angles. This is evident from the excellent k_{eff} and HPTR values achieved compared to the target values. RL critical positions are close to the ideal critical positions identified in Serpent for 0 YR (90), 2 YR (113), and 4 YR (135).

On the other hand, A2C is still providing reliable results for times 0 YR and 4 YR as shown by the k_{eff} and HPTR values for these burnup steps and how close the drum angles identified by A2C compared to PPO. However, A2C has struggled to figure out drum orientation for $t =$ 2 YR as k_{eff} deviates about 500 pcm from criticality and HPTR is slightly above the constraint of 1.02. This implies that the lower reward values illustrated in Fig. 4d for A2C can be attributed to $t =$ 2 YR performance metrics, which may not be obvious because of averaging the three burnup steps. Nevertheless, unlike PPO, the larger standard deviation observed for A2C rewards and the larger gap between the max and min rewards indicate that not all burnup steps have converged to satisfactory results.

V. DISCUSSIONS AND IMPLICATIONS

V.A. Surrogate Model

Each surrogate model was trained on 2100 data points. For a six-dimensional input space and seven-dimensional output space, these are not much data, especially when considering the complexity of the DNNs used. The total number of learnable parameters for each DNN surrogate was around 70 000, 120 000, and 190 000 for the 0, 2, and 4 YR burnup models, respectively. These high numbers of

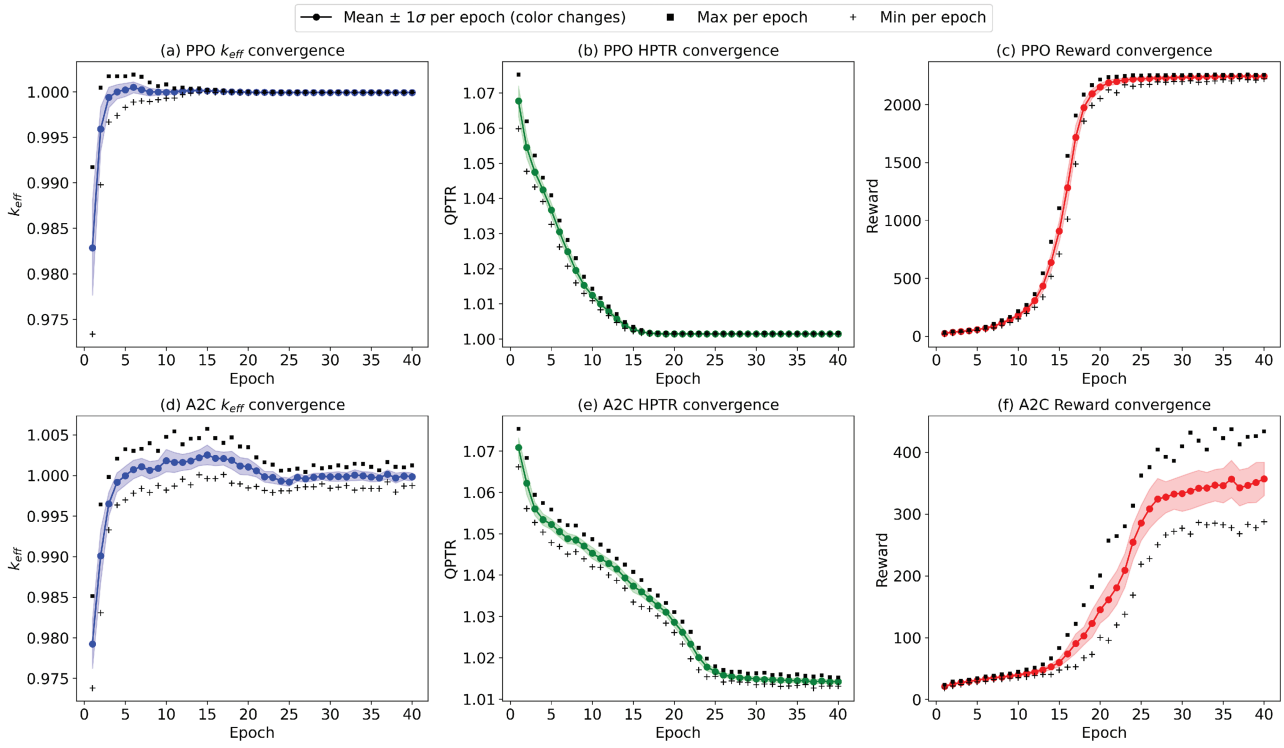

Fig. 4. Progression of PPO and A2C training as a function of the number of epochs. Each epoch comprises 30 000 time steps for which the statistics (mean, standard deviation, maximum, minimum) are derived. The top row shows the convergence of k_{eff}, HPTR, and the reward value as indicated in Eq. (5) for PPO while the bottom row shows these plots for A2C. All values shown are averaged over the three reactor states and 20 parallel agents. Also, note that the reward scale in this plot is slightly different from Fig. 3 of Ref. [24] (see the main text for explanation).

TABLE II

Testing Results of the Trained PPO and A2C Policies for Each Burnup State

Algorithm	Time (YR)	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	k_{eff}	HPTR
PPO	0	91	89	95	88	93	89	1.00002	1.002679
PPO	2	114	113	112	111	114	113	0.99993	1.001708
PPO	4	138	136	129	141	134	138	1.00000	1.002824
A2C	0	92	95	85	90	88	94	1.00008	1.005743
A2C	2	92	132	85	139	82	135	0.99502	1.027286
A2C	4	140	132	133	139	135	135	0.99998	1.003103

parameters resulted through our architecture optimization process and were necessary to keep k_{eff} errors low enough to be useful. A high number of parameters comes with the danger of overfitting to the training data; however, the test results show excellent performance, demonstrating that this did not occur. This may indicate that the underlying relationship between drum angles and k_{eff} and hexant power fractions is relatively simple.

A limitation of our method of surrogate modeling is that it requires a separate model for each burnup time. This limited our RL work to very short episode lengths of three burnup time steps. However, without this highly accurate surrogate model, running 1.2 million separate Serpent calculations would be necessary, which would make sample-hungry RL impractical. Scaling the DNN model to capture more burnup steps, and even using one model for all burnup steps, is feasible if more data can be generated from Serpent. In addition, capturing model uncertainty through models like Gaussian processes[28] might become necessary when model accuracy becomes limited.

V.B. Comparison of A2C and PPO Training

Enabled by the accurate surrogate models, both A2C and PPO learned how to take actions leading to criticality and balanced hexant powers. In each case, optimizing for criticality was easier than learning to act symmetrically. That there are far more critical states than symmetric power states explains this since each drum may move independently to achieve the criticality but must move together for symmetry to hold. In fact, with both algorithms, it took around six times longer for the HPTR to reach its lowest levels than it did for k_{eff} to get close to 1, which corresponds to the relative number of satisfactory states for each objective.

In terms of overall performance, however, PPO was far more robust than A2C within the training time frame. A2C remained unstable even as it improved its policy. In sharp contrast with PPO, which became more stable with time, A2C maintained a level of instability proportional to the mean reward and failed to converge within the training time for $t = 2$YR. Also, HPTR values achieved by PPO are still lower than A2C for all three burnup steps. These results are consistent with the Stable Baselines3 implementation of PPO as a smoother variant of A2C because of the addition of generalized advantage estimation and a clipping term to prevent overly large policy updates.

V.C. Implications for Microreactor Control

Ultimately, these algorithms learned something that any nuclear engineer already knows: A balanced power profile can be achieved through symmetric drum angles, and there should be a single, symmetric criticality position for a given material and temperature makeup. However, they did this with only a simple reward function to guide it through a 13-dimensional space (6 angle actions, 6 power fractions, and k_{eff}) within 1 h. For people in the RL field, this result is probably unsurprising, but RL applications for nuclear reactor control are sparse, with the four examples described in Sec. II by Chen and Ray,[7] Park et al.,[9] Bae et al.,[4] and Dong et al.[10] being the only ones we could find. None of these looked at microreactors.

While RL training requires about 60 min to complete, it takes only ~0.025 s for a trained policy to provide an optimal action for a given burnup time. This rapid decision capability could one day help enable real-time autonomous control. However, the black-box nature of the underlying models would mandate intense scrutiny for this purpose. Even so, while neural network policies are not interpretable, the actions they take can be studied. Especially with new microreactor designs that have not yet been built and so have no operational experience, RL has the potential to anticipate issues and opportunities in reactor control.

V.D. Future Work

The natural progression of this work would involve applying this methodology for real-time control in transient calculations while adhering to the physical limitations of drum movements (such as maximum drum speed and reactivity insertion per degree of movement). For instance, performing transient simulations in load-following scenarios using point-kinetics models will allow RL to receive feedback from each degree of rotation based on the reactivity inserted and the resulting power changes. Since load-following typically occurs over relatively long periods compared to reactor response times (e.g., 10 to 20 min), RL algorithms can adapt to system feedback while adjusting the drums to achieve the desired power level. Future efforts could also include comparisons with traditional time-series models like long short-term memory, which have been demonstrated for transient modeling in nuclear systems.[29]

VI. CONCLUSIONS

In this work, we have demonstrated an initial application of intelligent control using RL for nuclear heat pipe microreactors. The RL agent was trained using PPO and A2C, advanced deep RL techniques, based on a high-fidelity simulation of a microreactor design inspired by the Westinghouse eVinci design. A Serpent model provided high-fidelity data on drum positions, core criticality, and core power distribution for training a surrogate model. This surrogate model was then used to train a PPO and A2C control policies to determine the optimal drum position across various reactor states, ensuring critical core conditions and symmetric power distribution within all six quadrants. The results demonstrate the exceptional performance of PPO in identifying optimal drum positions, achieving an HPTR value of approximately 1.002 (within the limit of <1.02) and maintaining criticality within a range of 10 pcm. A2C policies provided lower performance overall compared to PPO where, for certain burnup steps, HPTR constraint was not satisfied. The computational cost for training the RL policy was approximately 60 min, while policy control predictions during deployment took around 0.025 s

for both algorithms. Future extensions of this work include expanding the scope to include additional control variables within the state space (e.g., temperature control), implementing advanced techniques for engineering reward functions to expedite convergence, and transitioning to real-time control for load-following scenarios using transient simulations.

Funding

The work in this paper was supported by various sources based on the authors' contributions—Majdi I. Radaideh: U.S. Department of Energy (DOE), Office of Nuclear Energy Distinguished Early Career Program (award number DE-NE0009424); Leo Tunkle: Idaho National Laboratory's (INL's) Laboratory Directed Research and Development (LDRD) Program (award number 24A1081-116FP) under DOE Idaho Operations Office contract number DE-AC07-05ID14517; Dean Price: DOE, Office of Nuclear Energy, Integrated University Program Graduate Fellowship; Kamal Abdulraheem: Michigan Institute of Data Science, Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship; Linyu Lin: INL's LDRD Program (award number 24A1081-116FP) under DOE Idaho Operations Office contract number DE-AC07-05ID14517.This research made use of INL's High Performance Computing systems located at the Collaborative Computing Center and supported by the Office of Nuclear Energy of the DOE and the Nuclear Science User Facilities under contract number DE-AC07-05ID14517.

Nomenclature

A2C: Advantage Actor-Critic
EMD: eVinci Motivated Design
DNN: deep neural network
DQN: Deep Q-Network
HPTR: hexant power tilt ratio
MAE: mean absolute error
MDP: Markov decision process
PPO: Proximal Policy Optimization
RL: reinforcement learning
3-D: three-dimensional

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Credit Author Statement

Majdi I. Radaideh: conceptualization, methodology, software, validation, formal analysis, visualization, investigation, funding acquisition, supervision, project administration, writing—original draft; Leo Tunkle: methodology, software, validation, formal analysis, visualization, investigation, writing—original draft; Dean Price: methodology, software, validation, data curation, writing—original draft; Kamal Abdulraheem: software, validation, writing—review and edit; Linyu Lin: methodology, funding acquisition, project administration, writing—review and edit; Moutaz Elias: conceptualization, supervision, writing—review and edit.

ORCID

Majdi I. Radaideh http://orcid.org/0000-0002-2743-0567

References

1. K. SHIRVAN et al., "UO_2-Fueled Microreactors: Near-Term Solutions to Emerging Markets," *Nucl. Eng. Des.*, **412**, 112470 (2023); http://doi.org/10.1016/j.nucengdes.2023.112470.
2. G. PAPOUDAKIS et al., "Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning" (2019); http://arxiv.org/abs/1906.04737.
3. V. MNIH et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature*, **518**, 529 (2015); http://doi.org/10.1038/nature14236.
4. J. BAE, J. M. KIM, and S. J. LEE, "Deep Reinforcement Learning for a Multi-Objective Operation in a Nuclear Power Plant," *Nucl. Eng. Technol.*, **55**, *9*, 3277 (2023); http://doi.org/10.1016/j.net.2023.06.009.
5. J. DEGRAVE et al., "Magnetic Control of Tokamak Plasmas Through Deep Reinforcement Learning," *Nature*, **602**, 414 (2022); http://doi.org/10.1038/s41586-021-04301-9.
6. J. ARROYO et al., "Reinforced Model Predictive Control (RL-MPC) for Building Energy Management," *Appl. Energy*, **309**, 118346 (2022); http://doi.org/10.1016/j.apenergy.2021.118346.
7. X. CHEN and A. RAY, "Deep Reinforcement Learning Control of a Boiling Water Reactor," *IEEE Trans. Nucl. Sci.*, **69**, *8*, 1820 (2022); http://doi.org/10.1109/TNS.2022.3187662.
8. K.-C. KWON et al., "Compact Nuclear Simulator and Its Upgrade Plan," *Proc. IAEA Specialists' Mtg. Training Simulators in Nuclear Power Plants: Experience, Programme Design and Assessment Methodology*, Essen,

Germany, November 17–19, 1997, International Atomic Energy Agency (1997).

9. J. PARK et al., "Control Automation in the Heat-Up Mode of a Nuclear Power Plant Using Reinforcement Learning," *Prog. Nucl. Energy*, **145**, 104107 (2022); http://doi.org/10.1016/j.pnucene.2021.104107.
10. Z. DONG et al., "Multilayer Perception Based Reinforcement Learning Supervisory Control of Energy Systems with Application to a Nuclear Steam Supply System," *Appl. Energy*, **259**, 114193 (2020).
11. Z. YI et al., "Deep Reinforcement Learning Based Optimization for a Tightly Coupled Nuclear Renewable Integrated Energy System," *Appl. Energy*, **328** (2022).
12. M. I. RADAIDEH et al., "Physics-Informed Reinforcement Learning Optimization of Nuclear Assembly Design," *Nucl. Eng. Des.*, **372**, 110966 (2021); http://doi.org/10.1016/j.nucengdes.2020.110966.
13. M. I. RADAIDEH, B. FORGET, and K. SHIRVAN, "Large-Scale Design Optimisation of Boiling Water Reactor Bundles with Neuroevolution," *Ann. Nucl. Energy*, **160**, 108355 (2021); http://doi.org/10.1016/j.anucene.2021.108355.
14. M. I. RADAIDEH et al., "NEORL: NeuroEvolution Optimization with Reinforcement Learning—Applications to Carbon-Free Energy Systems," *Nucl. Eng. Des.*, **412**, 112423 (2023); 10.1016/j.nucengdes.2023.112423.
15. J. LEPPÄNEN et al., "The Serpent Monte Carlo Code: Status, Development and Applications in 2013," *Ann. Nucl. Energy*, **82**, 142 (2015); http://doi.org/10.1016/j.anucene.2014.08.024.
16. G. TESAURO, "Td-Gammon: A Self-Teaching Backgammon Program," *Applications of Neural Networks*, p. 267, Springer, Boston, Massachusetts (1995).
17. V. MNIH et al., "Playing Atari with Deep Reinforcement Learning" (2013); https://doi.org/10.48550/arXiv.1312.5602.
18. R. WILLIAMS, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, **8**, 229 (1992); http://doi.org/10.1007/BF00992696.
19. V. MNIH et al., "Asynchronous Methods for Deep Reinforcement Learning," *Proc. 33rd Int. Conf. Machine Learning*, in *Proceedings of Machine Learning Research*, Vol. 48, pp. 1928–1937, M. F. BALCAN and K. Q. WEINBERGER, Eds. (2016); https://proceedings.mlr.press/v48/mniha16.html.
20. A. RAFFIN et al., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *J. Mach. Learn. Res.*, **22**, *268*, 1 (2021); http://jmlr.org/papers/v22/20-1364.html.
21. J. SCHULMAN et al., "Trust Region Policy Optimization," *Proc. 32nd Int. Conf. Machine Learning*, in *Proceedings of Machine Learning Research*, Vol. 37, pp. 1889–1897, F. BACH and D. BLEI, Eds. (2015); https://proceedings.mlr.press/v37/schulman15.html.
22. J. SCHULMAN et al., "Proximal Policy Optimization Algorithms" (2017); https://doi.org/10.48550/arXiv.1707.06347.
23. J. SCHULMAN et al., "High-Dimensional Continuous Control Using Generalized Advantage Estimation" (2018); https://doi.org/10.48550/arXiv.1506.02438.
24. D. PRICE et al., "Thermal Modeling of an eVinci™-Like Heat Pipe Microreactor Using OpenFOAM," *Nucl. Eng. Des.*, **415**, 112709 (2023); http://doi.org/10.1016/j.nucengdes.2023.112709.
25. G. BROCKMAN et al., "OpenAI Gym" (2016); https://doi.org/10.48550/arXiv.1606.01540.
26. M. I. RADAIDEH et al., "Demonstration of Microreactor Reactivity Control with Reinforcement Learning," *Proc. Int. Conf. Physics of Reactors (PHYSOR 2024)*, San Francisco, California, April 21–24, 2024, p. 1427, American Nuclear Society (2024).
27. F. CHOLLET et al., "Keras" (2015); https://keras.io.
28. M. I. RADAIDEH and T. KOZLOWSKI, "Surrogate Modeling of Advanced Computer Simulations Using Deep Gaussian Processes," *Reliab. Eng. Syst. Saf.*, **195**, 106731 (2020); http://doi.org/10.1016/j.ress.2019.106731.
29. M. I. RADAIDEH et al., "Neural-Based Time Series Forecasting of Loss of Coolant Accidents in Nuclear Power Plants," *Expert Syst. Appl.*, **160**, 113699 (2020); http://doi.org/10.1016/j.eswa.2020.113699.