BOARD #139: WIP: Interactive Software Platform for Undergraduate Nuclear Engineering Education

Jonathan William Ross, Drexel University

Jonathan Ross completed his studies for the BS and MS degrees at Drexel University. He is currently pursuing a PhD with the Center for Electric Power Engineering at Drexel. His current research interests include distributed generator control in power distribution systems, improving numerical methods for solving the power flow equations, and creating educational tools for engineering undergraduates.

Dr. Karen Miu Miller, Drexel University

Dr. Karen Miu Miller received her B.S., M.S. and Ph.D. from Cornell University. She is currently a Professor in the Department of Electrical and Computer Engineering at Drexel University. Her educational research and teaching interests has focused on electric power and energy delivery. systems.

Dr. Christopher Wayne Peters, Drexel University

Dr. Peters is currently a Teaching Professor in the Electrical and Computer Engineering department at Drexel University. His interests are low-cost simulations for educational use and gaming in the classroom.

WIP: Interactive Software Platform for Undergraduate Nuclear Engineering Education

Abstract

The energy sector is experiencing a revival in interest toward nuclear power as an attractive compromise between fossil fuels and renewables, particularly to address concerns about energy independence and carbon emissions. Conceptually, nuclear engineering fundamentals are challenging to teach to undergraduates as it requires a multi-disciplinary approach. Yet, it is a field that continues to power much of the world. Therefore, there is a continued need to educate engineering students in the field of nuclear power; and this paper presents a software emulation tool for nuclear power plant design. Engineering undergraduates can use the platform to explore fundamental nuclear engineering concepts and better prepare themselves for careers in and around nuclear power. A high-level controls interface allows for real-time adjustments by students, and, for detailed or long-term controls, the Python source code is also provided to the students. Modeling of the different plant subsystems and their interactions are illustrated and discussed. Sample experimental procedures are discussed and results from the simulator are presented. In summary, with the goal of increasing nuclear engineering education beyond post-graduate and on-the-job training, the interactive tool allows for hands-on experimentation at different levels of detail. Adoption of the tool into undergraduate power and energy courses seeks to accelerate undergraduate students' understanding of the integrated behavior between key disciplines of nuclear power plant design.

1 Introduction

Nuclear power plant design requires cohorts of engineers trained in various aspects of reactor theory, thermal-hydraulic analysis, and power system stability. Several fields of study must synergize to effectively harness the power of the atom; yet most undergraduates are unfamiliar with the fundamentals of atomic physics, radiation interaction, and basic reactor theory. Thus, a layered, hands-on simulation approach can provide confidence and foster proficiency in key educational topics across multiple disciplines.

To this end, one can find several institutions whose organizational goals include the advancement of nuclear education and safety with simulation tools. The Nuclear Energy Agency (NEA) maintains a catalogue of miscellaneous computer programs that are available to the public. As a forum for discussion on these programs and to champion further development, the NEA also recently started hosting the International School on Simulation of Nuclear Reactor Systems (SINUS). This new annual program seeks to develop modeling/simulation tools in conjunction with validation/verification methodologies. Participants in the program "engage in a dynamic, hands-on learning experience through self-paced project assignments that introduce them to the cutting-edge single- and multi-physics software packages" [1]. Furthermore, the International Atomic Energy Agency (IAEA) has been committed to developing and distributing its comprehensive power plant simulation software packages, such as the Advanced Two-Loop

Large PWR (Korean-OPR 1000), Russian-type PWR (VVER-1000), and Integral Pressurized Water Reactor (SMR) [2]. The IAEA programs are highly detailed and are designed to both demonstrate normal plant behavior and illustrate various fault scenarios. Similar work was conducted by Ahnert et al. in [3] by implementing a detailed graphical interface for a simulation of the Jose Cabrera Nuclear Power Plant. The interface included an interactive technical diagram of the plant, alarm panels for different circuits, and lifelike controls/instrumentation. Classroom feedback on [3] was positive from students and instructors alike, but this type of simulator is only available to specific universities.

While programs exist that are designed to handle detailed nuclear design, safety, and licensing, these tools are often expensive, closed to the public, and/or go far beyond the scope of a multi-disciplinary undergraduate reactor theory course [4]. There are several open-source options available that can simulate independent phenomena [1], [5] and others that capture some integration between subsystems, such as the research reactor simulator in [6], and the reactor kinetics package in [7].

This paper presents an interactive software package with a comprehensive commercial reactor simulator that was specifically designed for an introductory nuclear reactor theory course at Drexel University [8]. ECEP-402: Theory of Nuclear Reactors is designed for 3rd- and 4th-year electrical, mechanical, and chemical engineering students who have some background in basic quantum mechanics and differential equations. Because our institution follows an 11-week course calendar and does not have a nuclear engineering major, our reactor theory course covers many topics in a short time span; it was necessary to develop a new simulator that caters to the specific mathematical models presented in our current course materials.

Specifically, because of its widespread use, the pressurized water reactor (PWR) plant was chosen for the platform. Models for the reactor neutronics, control rods and chemical shim, fission product concentrations, coolant loops, steam generator, and steam turbine are developed and integrated in a Python environment. Each of the PWR subsystems align with corresponding lecture modules from our reactor theory course.

Students work through a Node-RED graphical user interface (GUI) to control the simulation in a real-time setting and perform experiments, like changes in demand and moving of control rods. Students may also manipulate the reactor design criteria and initial conditions in the Python case file. Programmatically interacting with the simulator allows for more detailed analyses, such as criticality and prompt criticality, coefficient of reactivity, and Xenon/Samarium precluded startup. The existing final project for our course tasks students with developing a static PWR design based on given design constraints. This simulator enables students to perform transient experiments with their designs too, expanding the range of procedures they can perform.

2 Mathematical Modeling

The interactive tool simulates a commercial PWR plant by modeling several key subsystems as groups of differential algebraic equations (DAEs) and then integrating them simultaneously. The

process begins in the reactor core with the nuclear chain reaction. Energy from fission transfers from the fuel to the coolant in the primary loop, and the primary coolant transfers energy to the secondary coolant in the steam generator. Energy from the steam is then transferred to the prime mover of an electrical generator via the steam turbine system and then flows out to the electrical grid.

The integrated DAE model is translated to a system of ordinary differential equations (ODEs) by index reduction [9]. Since the plant exhibits behaviors on a wide range of time scales, the variable step-size stiff/non-stiff ODE solver LSODA is used to determine the trajectory of the state variables at each step [10]. To foster an interactive software package, a real-time integration technique is implemented to keep the simulation running in time with the system clock [11]. Measuring the change in time since the start of the previous integration step and using this difference as the target time for the next step enables the simulation to remain synchronized. A multiplier may be applied to this duration to speed up or slow down the simulation; for instance, a student may wish to run the program in double time to observe a longer-term behavior of the plant while keeping the program interactive.

2.1 Nuclear Chain Reaction Model

The reactor core is modeled as a finite cylinder consisting of a heterogeneous lattice of uranium dioxide (UO2) fuel rods and pressurized water which serves as both the neutron moderator and primary coolant. Figure 1 contains a block diagram that visualizes the relationships between the components of the neutron chain reaction.

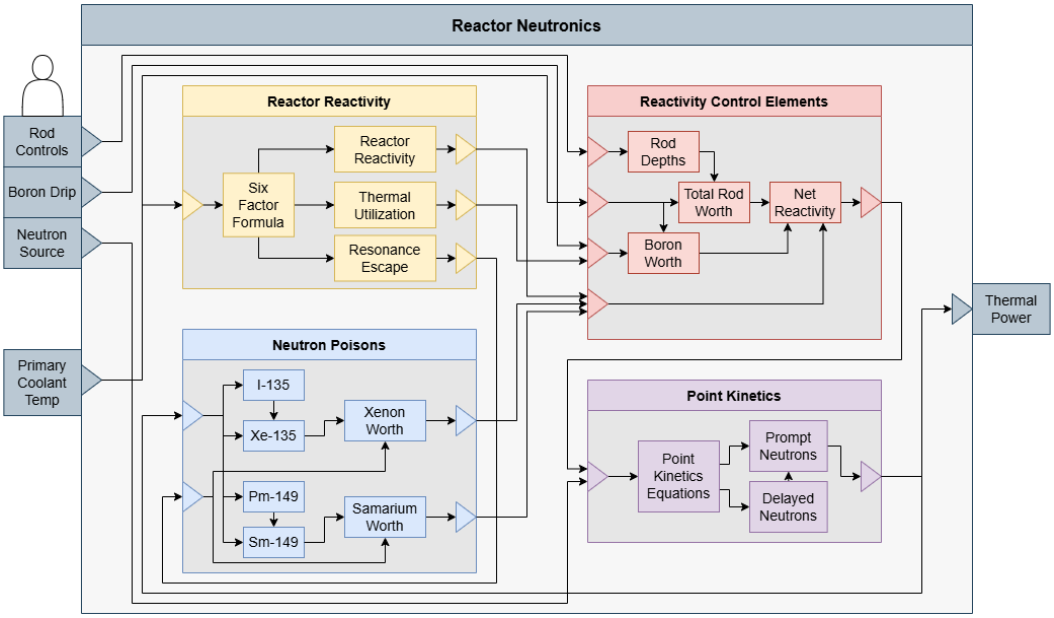

Figure 1. Nuclear Chain Rection Model Block Diagram.

The concepts of criticality, reactivity, and prompt criticality are crucial for students to understand the behavior of the neutron chain reaction. Whether the chain reaction can sustain itself and how

fast the reaction rate changes depend on the criticality k of the core. The reaction rate accelerates if $k > 1$ and decelerates if $k < 1$. However, it is often useful to use the reactivity metric $\rho = (k - 1)/k$ instead of criticality since ρ describes a fractional deviation from the critical state. This metric also makes it easier to identify the prompt criticality condition, in which the core is critical on prompt neutrons alone rather than on both prompt and delayed neutrons. Criticality is determined from Lamarsh's model of a heterogeneous cylindrical core [12] and then transformed to reactivity for use in the simulation.

To control the reactor, both control rods and chemical shim are used to absorb neutrons and slow the chain reaction. Control rods are modeled as finite, cylindrical neutron absorbers radially displaced from the center of the core, and they absorb essentially all incident neutrons (black rods). Boric acid is used as chemical shim and homogeneously mixed into the primary coolant. The worths of the rods and chemical shim are subtracted from the reactor's reactivity [13].

As the uranium U-235 nuclear fuel splits during fission it yields several daughter isotopes. Namely, iodine I-135 and promethium Pm-149 may both be produced from fissions, and they decay respectively into xenon Xe-135 and samarium Sm-149. Both decay products slow down the chain reaction as a result of very large neutron capture cross-sections [12]. The worths of these poisons are also subtracted from the reactor's reactivity to determine the net reactivity.

The six-group point kinetics equations are used to model the neutron concentrations in the core over time. The state variables include the prompt neutron concentration and the concentrations of six groups of delayed neutrons [13]. As the net reactivity of the core depends on several other state variables in the whole plant, it is considered an algebraic variable for the simulation. The neutron source term is considered a control variable. Control rod drive and boron drip rates are also control variables for the plant but are embedded within the net reactivity expression. The thermal power generated from fission is a function of the thermal neutron flux, which is in turn a function of the prompt neutron concentration.

2.2 Thermo-Mechanical Energy Transfer Model

The thermo-hydraulic loop consists of a primary loop with pressurized liquid water, and a secondary loop in which feedwater transitions to steam and proceeds to drive the steam turbine. Figure 2 shows the relationships between the lumps of the thermo-mechanical energy transfer model.

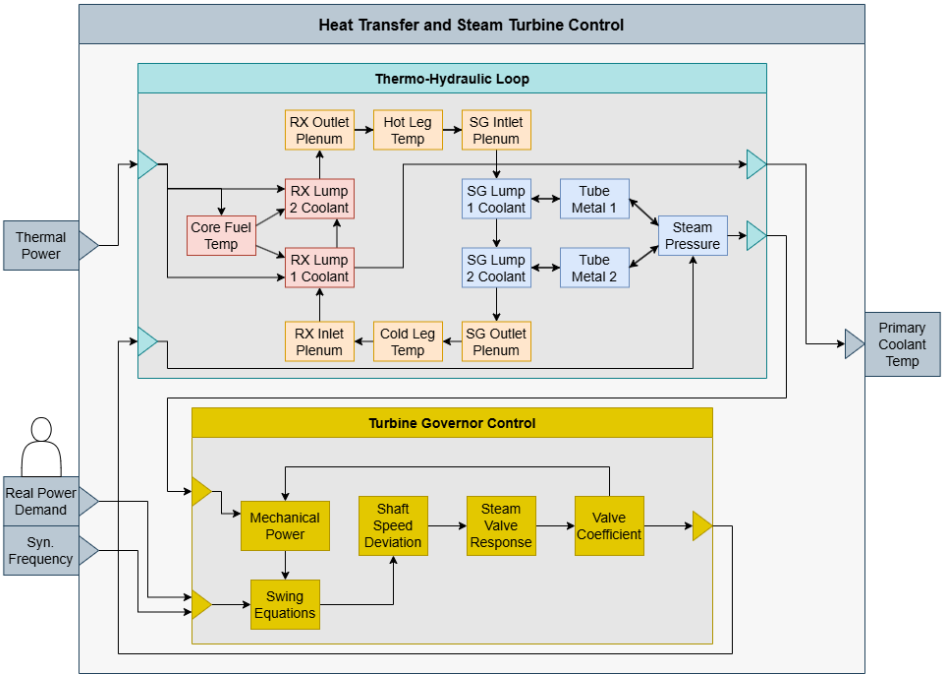

Figure 2. Thermo-Mechanical Energy Transfer Model Block Diagram.

Overall heat transfer coefficients are calculated using the physical and geometric properties of the fuel assemblies, primary coolant, steam generator tube metal, and secondary coolant. These coefficients along with the mass flow rates of the primary and secondary coolant loops determine the thermodynamic relationships between the average temperatures of the fuel and the coolant lumps [14].

The mechanical torque applied to the prime mover depends on both the steam pressure and steam valve position [15]. Any instantaneous difference between the mechanical torque and electrical demand causes the speed of the prime mover to change. To maintain synchronicity with the grid, the steam governor controls the steam valve with a simple proportional-integral (PI) controller to drive the difference between the mechanical power and electrical demand to zero [16].

Overall, the reaction controls and generator controls influence the behavior of the state and algebraic variables for each subsystem. Simulation outputs are representative of measurable phenomena, including thermal neutron flux, reactor period, average fuel and coolant lumps temperatures, steam pressure, steam valve coefficient, shaft frequency, and turbine power. The remaining states are, of course, numerically calculated over time and could be accessed by programmatically interfacing with the simulator. However, students are expected to use the mathematical relationships introduced in the lecture materials to infer these other parameters, such as the net reactivity over time. The selected inputs and outputs are displayed on an interactive GUI for students to monitor and control.

3 Software Overview

3.1 Organization of Software Components

The software package consists of a Python-based backend and a GUI that is accessible by web browser and managed by a Node-RED flow [17]. Figure 3 illustrates the relationships between the software components.

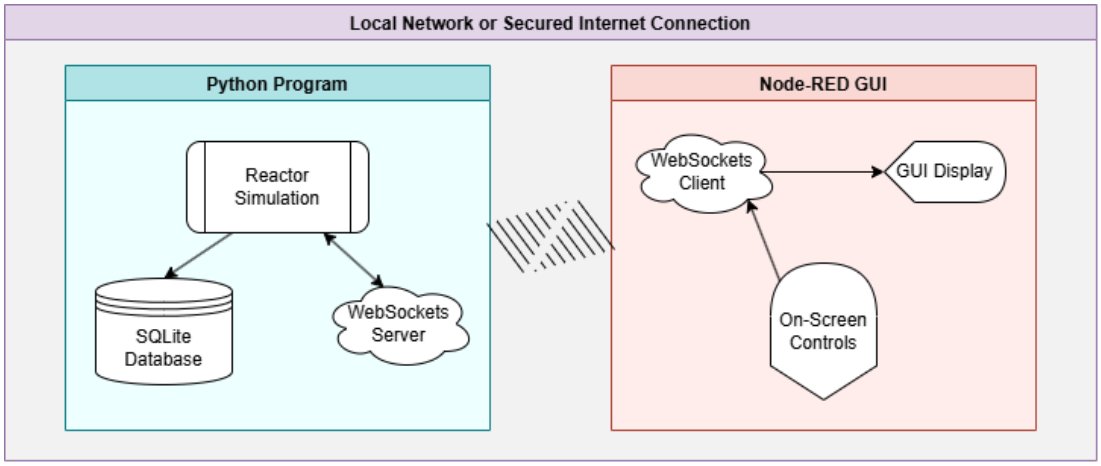

Figure 3. Illustration of Interaction Between Software Components.

Numerical simulation of the PWR plant is handled by a Python program that constantly runs in the background. Sending a pause signal causes the current state of the plant to be saved and stops the simulation loop. The simulation always begins in a paused state and waits to receive a resume signal from the GUI. While the simulation is running, the state, algebraic, and control variables are periodically recorded to an SQLite database that students can pull from the server to retrieve their data at the end of each experiment.

At the same time, the Python program manages a WebSocket server to mediate incoming requests for the system measurements, as well as convey control signals from the user to the simulation loop. This arrangement allows students to remotely access and manage their simulation(s) remotely over the Internet. Separately, Node-RED is used to both manage the GUI and make requests to the WebSocket server. Students may access the GUI through a web browser by navigating to a specified URL. The use of Node-RED also allows for rapid deployment of additional dashboards that are individually tailored for specific experiments.

3.2 Graphical User Interface Demonstration

Figure 4 shows the GUI during a sample experiment. The leftmost column of the GUI contains the simulation controls. At the top of this section are elements for managing the program, such as a pause/resume switch, a dropdown menu for the time multiplier, a button to reset the charts, and a button to reset the simulation to its full-power/steady-state initial conditions.

Figure 4. Sample Load-Following Demand Change Experiment Screenshot.

Underneath these are the actual control signals for the plant. The nuclear reaction controls include the sliders for the neutron source term, the boric acid drip rate, the control rod drive signal, and the reactor SCRAM button. The generator controls include the numeric inputs for the electric demand and synchronous frequency setpoints.

The remaining three columns provide measurements for select system state and algebraic variables.

- Nuclear Chain Reaction
    - Thermal neutron flux chart
    - Control rod insertion fraction chart
    - Boric acid concentration
    - Reactor period gauge
- Coolant Transport
    - Average fuel temperature
    - Average, hot leg, and cold leg primary coolant temperatures
    - Steam pressure in secondary loop
- Synchronous Generator
    - Steam valve coefficient
    - Turbine shaft frequency
    - Turbine mechanical torque and electrical demand

Hovering the mouse cursor over a point on a plot gives a tooltip which displays the values of each series at that point in time. This is particularly useful for performing calculations and observing relatively small changes to the states.

4 Experimental Procedures

The software package allows for a variety of experiments that help students understand the principles of reactor theory and operation. The web browser GUI best serves students performing short-term transient analyses, such as:

- Adjustments to the electrical demand
- Control rod movements and rod worth estimations
- Approach to criticality using control rods and/or chemical shim
- Short-term behavior after reactor SCRAM

Experiments that require complicated/simultaneous control adjustments or long-term transient analyses are better served by writing scripts that directly interact with the Python simulator, including but not limited to:

- Long-term fission products transients in response to demand changes
- Long-term behavior after reactor SCRAM
    - Iodine/xenon pit illustrations
    - Equilibrium samarium after shutdown
- Analyzing moderator coefficient of reactivity for different chemical shim concentrations

The default values in the design file are specified to emulate the behavior of a 3,800 MW-thermal commercial PWR plant. Students may also customize the design of the plant by editing the design file in the Python portion of the program. For instance, the reactor theory course instructs students in the calculation of thermo-hydraulic time constants as a function of both physical properties like specific heat capacity, as well as design parameters, like coolant mass flow rate, conduction heat transfer surface areas, etc. The course also instructs students to calculate reactor reactivity based on fuel enrichment, fuel rod radius and lattice pitch, core height-to-diameter ratio, etc. Students must then decide on number, size, and placement of control rods to maintain an appropriate shutdown margin, as well as an appropriate concentration of chemical shim so as to enable long-term reactivity control without inducing positive temperature feedback.

4.1 Load-Following Demand Change Experiment

Most commercial reactors are designed and operated to supply the base load for a variety of fueling, thermal, and economic reasons. However, showing an example of load-following behavior provides students with a holistic understanding of the relationships between the plant subsystems, especially those of negative reactivity feedback and conservation of energy. Figure 4 from the previous section illustrates this experiment.

For this experiment the simulated reactor begins at full-power/steady-state conditions. Students slowly ramp down the electric demand by about 5% over the course of a minute. Once the plant appears to be at its new steady-state, they then perform a step decrease in the electric demand another 5% and allow the plant to once again approach a steady-state.

Students are provided with key physical and geometric design parameters of the plant and expected to perform basic thermal calculations using their data. Students are also expected to draw their own block diagram of the plant and use it in conjunction with their data to explain/demonstrate:

- Change to the thermal power in the core,
- Thermal efficiency of the plant is not necessarily constant,
- Moderator temperature reactivity feedback working from the generator back to the core.

4.2 Control Rods Experiment

The control rods experiment illustrates the effect of control rod adjustments on the behavior of the reactor over short periods of time. For this experiment, the simulated reactor begins at full-power/steady-state conditions with the control rods all set to the same position. Figure 5 shows two subsequent control rod insertions as would be expected in the first part of this experiment.

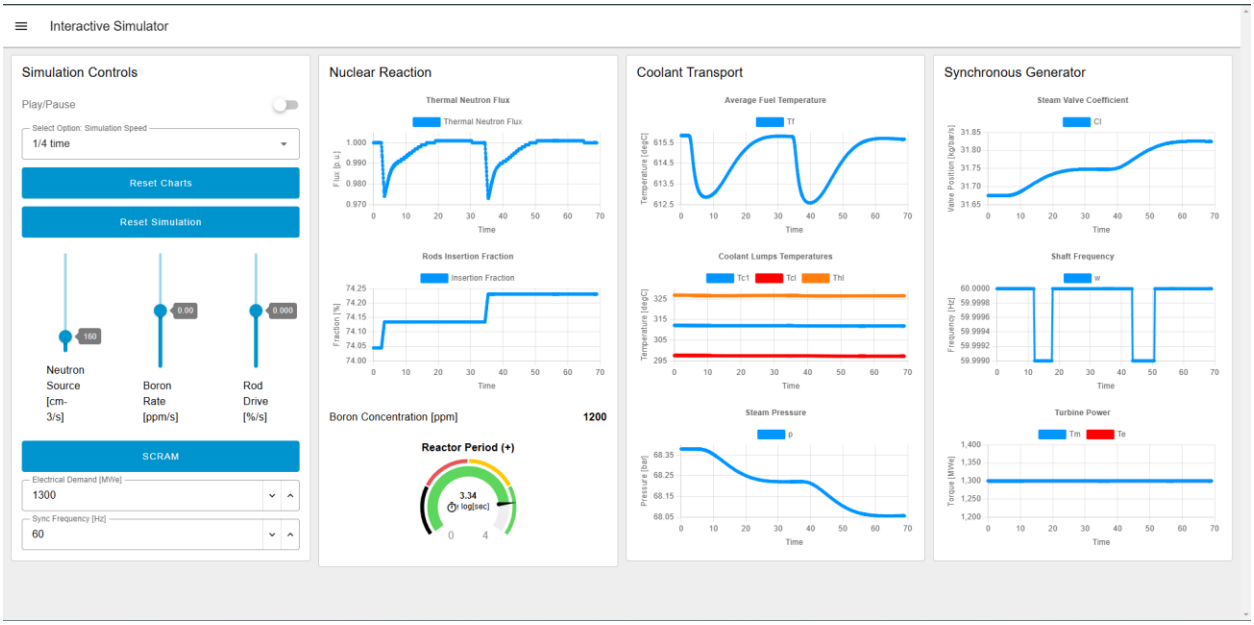

Figure 5. Sample Control Rods Experiment Part 1 Screenshot.

First the students decrease the time scale to ¼ time so that the prompt jump can be more easily seen on the neutron flux graph. Then students inject a rod drive control signal to rapidly insert the control rods by a specified small percentage of the core height. Using the prompt jump approximation [18] and the change in position of the rods, students use their data to estimate the incremental worth of all the rods about their starting positions.

The second part of the experiment returns the time scale to normal and starts after the reactor has returned to a steady-state via its own temperature feedback. Students are then tasked with reaching a target pressure in the steam generator by only adjusting the control rods.

5 Future Work and Conclusions

The most immediate need for future work is to test the software package in a classroom setting. Feedback from students and evaluations from instructors would both support the implementation of this package in the existing undergraduate curriculum, as well as inform the software developers of key aspects to improve. Plans are being made to develop and deploy student assessments for the tool within an introductory reactor theory course and an introductory power systems course at our institution.

Further, plans are in place to continue development of the PWR simulator. Perhaps the most obvious future works involve adding further subsystems to the plant, such as the pressurizer and its control, as well as a simple decay heat model to demonstrate the need for long-term cooling after shutdown. Several key modeling simplifications imposed on the physical laws driving the simulation are summarized in Table 1. These assumptions stem from both the desire to target a 3rd- and 4th-year undergraduate engineering audience, especially those without a nuclear background, as well as to align with the reactor theory course lecture materials and expected prerequisite knowledge.

Table 1. Modeling Assumptions and Suggestions for Future Work.

Subsystem	Assumptions	Notes & Suggestions
Reactor Core	• Constant fuel mass (no burnup) • Neighboring rods do not affect each other's worth • Heat transfer with average fuel and coolant temperatures	• Additional fission products may be considered for burnup and decay heat considerations • Incorporate rod utilization • Consider effects of non-uniform thermal flux distribution
Coolant Loops	• Constant primary coolant pressure and mass flow rate • Inlet liquid mass flow rate perfectly matches outlet vapor mass flow rate • Critical steam flow assumption	• Incorporate pressurizer and primary coolant pump models • Model of feedwater pump, water level in steam generator, etc.
Electrical generator	• Real power injection tied to an infinite bus	• Detailed generator model required for integration with realistic electrical network

This paper presents a physics-based interactive program that emulates a PWR plant. Students can remotely access the program and control the simulation in real-time. The mathematical models can accommodate normal plant operations and are simplified to be approachable by an undergraduate audience with a general engineering background. Students can reinforce the

fundamental nuclear engineering concepts they learn in the classroom with software laboratory assignments designed around this software package. The hands-on aspects of the program help engage students and can further inspire them to take an interest in nuclear power.

References

[1] Nuclear Energy Agency, "International school on simulation of nuclear reactor systems (SINUS)," oecd-nea.org, 2025. [Online]. Available: https://www.oecd-nea.org/jcms/pl_88778/international-school-on-simulation-of-nuclear-reactor-systems-sinus.

[2] International Atomic Energy Agency, "Nuclear reactor simulators for education and training," iaea.org, 2024. [Online]. Available: https://www.iaea.org/topics/nuclear-power-reactors/nuclear-reactor-simulators-for-education-and-training.

[3] C. Ahnert, D. Cuervo, N. Garcia-Herranz, J. M. Aragones, O. Cabellos, E. Gallego, "E. Minguez, A. Lorente, D. Piedra, L. Rebollo, and J. Blanco, "Education and training of future nuclear engineers through the use of an interactive plant simulator," in *International Journal of Engineering Education*, vol. 27, no. 4, pp. 722-732, 2011.

[4] U.S. Nuclear Regulatory Commission, "Computer codes," NRC.gov, 2024. [Online]. Available: https://www.nrc.gov/aboutnrc/regulatory/research/safetycodes.html.

[5] P. K. Romano, N. E. Horelik, B. R. Herman, A. G. Nelson, B. Forget, and K. Smith, "OpenMC: a state-of-the-art Monte Carlo code for research and development," in *Annals of Nuclear Energy*, vol. 82, Aug. 2015.

[6] J. Malec, D. Toskan, and L. Snoj, "PC-based JSI research reactor simulator," in *Annals of Nuclear Energy*, vol. 146, Oct. 2020.

[7] K. Huff, "PyRK: a Python package for nuclear reactor kinetics," in *Proceedings of the 14th Python in Science Conference*, 2015.

[8] C. W. Peters, University Course, Title: "Theory of Nuclear Reactors." ECEP-402, College of Engineering, Drexel University, Philadelphia, PA. 2022.

[9] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 1998.

[10] A. Hindmarsh, "Odepack: Fortran ode solvers," *Computing*, 2022. [Online]. Available: https://computing.llnl.gov/projects/odepack.

[11] J. W. Ross, "Remote monitoring and control of a simulated nuclear reactor," M.S. thesis, Drexel Univ., Philadelphia, PA, 2022.

[12] J. R. Lamarsh and A. J. Baratta, *Introduction to nuclear engineering*, 3rd ed., Upper Saddle River, NJ: Prentice Hall, 2009.

[13] E. E. Lewis, *Fundamentals of nuclear reactor physics*, Amsterdam: Academic Press, 2008.

[14] T. W. Kerlin and B. R. Upadhyaya, *Dynamics and control of nuclear reactors*, London: Elsevier Academic Press, 2019.

[15] S. E. Arda, “Implementing a nuclear power plant model for evaluating load-following capability on a small grid,” M.S. thesis, Arizona State Univ., 2013.

[16] J. D. Glover, T. J. Overbye, and M. S. Sarma, “Transient Stability,” in *Power System Analysis & Design*, 6th ed., Boston, MA: Cengage Learning, 2017, pp. 669–737.

[17] OpenJS Foundation & Contributors, “Node-RED,” nodered.org, 2024. [Online]. Available: https://nodered.org.

[18] G. Wakabayashi, T. Yamada, T. Endo, and C. H. Pyeon, *Introduction to Nuclear Reactor Experiments*, Singapore: Springer 2022.