INL/RPT-24-77114

Enabling a Physical Twin for Control Methods Evaluation

March 2024

Jacob Farber
Joseph Oncken
Maria Coelho
Travis Lange
Ahmad Al Rashdan (Principal Investigator)

Idaho National Laboratory

Idaho National Laboratory

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/RPT-24-77114

Enabling a Physical Twin for Control Methods Evaluation

Jacob Farber
Joseph Oncken
Maria Coelho
Travis Lange
Ahmad Al Rashdan (Principal Investigator)

Idaho National Laboratory

March 2024

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

ABSTRACT

Advanced nuclear reactors play an important role in the energy future of the United States and the rest of the world. They are designed and operated based on a different model than that of the current operating fleet, thus enabling deployment in remote locations and allowing for safe semi-autonomous or autonomous operations. Such characteristics require the development of a new reactor control paradigm.

A significant factor in the development of control technologies and methods is integration of the various technologies and methods with each other and with hardware (both reactor system hardware and control hardware). A recent workshop on control of advanced reactors identified the lack of a flexible, expandable software/hardware infrastructure to enable such integration as a key gap.

A previous phase of the current effort involved developing and demonstrating the Control and Optimization Modular Modeling Application for Nuclear Deployment (COMMAND) platform, which is capable of integrating autonomous-control-enabling technologies and methods, without the constraints imposed by existing software solutions. This platform will enable advanced reactor developers to deploy and test any developed technologies and methods by employing a common framework, and to couple them with their own models and hardware.

The present phase of this effort entails using the Microreactor Automated Control System (MACS) platform, which was developed by the U.S. Department of Energy (DOE) Microreactor Program, to serve as a control method testbed. MACS can be used by advanced reactor developers to integrate their control-related research activities with any reactor system. For the present effort, MACS was customized to mirror Idaho National Laboratory (INL)'s Microreactor Applications Research Validation and Evaluation (MARVEL) microreactor, and COMMAND was leveraged to enable MACS to emulate the physics of MARVEL, thus positioning MACS as a physical twin of MARVEL.

To create this physical twin, MARVEL models were integrated into COMMAND. Previously, the MARVEL Reactor Excursion and Leak Analysis Program (RELAP5-3D) model was modified to allow simulations to be run with interactive process control (using COMMAND), and in this effort was further customized to accurately represent asymmetric drum rotations. The MARVEL Monte Carlo N-Particle (MCNP) model was also integrated to reflect the excess reactivity generated by moving the MACS drums, as well as to capture the spatial power distribution of the MACS core. As MCNP is computationally expensive and takes a great deal of time to run, machine learning (ML) was utilized to develop a surrogate modeling approach capable of estimating the relevant simulation results in milliseconds. An ML block was integrated within COMMAND to allow for interfacing with the surrogate models, each of which captures a specific MCNP parameter of interest. To enable COMMAND to send signals to and receive feedback from MACS, the generic Remote Procedure Calls (gRPC) communication protocol was integrated within COMMAND for interfacing with MACS hardware.

Integration of MACS and the MARVEL physics models via the COMMAND software platform was successfully demonstrated. The next phase of this effort will introduce and demonstrate a control scenario utilizing MACS and COMMAND. Both COMMAND and MACS are expected to continue to expand and evolve, thus closing the gap as an integrated software/hardware infrastructure and fostering advancement toward autonomous advanced reactor operations.

ACKNOWLEDGEMENTS

This work was funded by the U.S. Department of Energy (DOE) Nuclear Energy Enabling Technologies Advanced Sensors and Instrumentation (NEET ASI) program. The authors thank NEET ASI Program Manager Daniel Nichols (DOE) and National Technical Director Pattrick Calderoni (Idaho National Laboratory [INL]) for sponsoring and guiding this effort.

The authors also thank Carlo Parisi (INL) for his input and the models he contributed for the Microreactor Applications Research Validation and Evaluation (MARVEL) component of the use case, as well as Anthony Crawford and Andrew Heim (INL) for their support in regard to the Microreactor Automated Control System (MACS) component of the use case.

Page intentionally left blank

CONTENTS

FIGURES

TABLES

ACRONYMS

COMMAND	Control and Optimization Modular Modeling Application for Nuclear Deployment
DOE	Department of Energy
FNN	feedforward neural network
INL	Idaho National Laboratory
LED	light-emitting diode
MACS	Microreactor Automated Control System
MARVEL	Microreactor Applications Research Validation and Evaluation
MCNP	Monte Carlo N-Particle
ML	machine learning
NEET ASI	Nuclear Energy Enabling Technologies Advanced Sensors and Instrumentation
RELAP5-3D	Reactor Excursion and Leak Analysis Program
RPC	remote procedure call
SSH	secure shell

Page intentionally left blank

Enabling a Physical Twin for Control Methods Evaluation

1. INTRODUCTION

Nuclear power plays a key role in energy industry decarbonization in the United States and throughout the rest of the world. However, changes made to energy requirements in recent years, including demand for more flexible and economical nuclear energy production, have raised the need for advanced nuclear technologies. Advanced nuclear reactors differ from the current operating fleet in that they are designed to be deployed in remote locations, monitored remotely, operated at lower and/or variable power ratings, compact in size, manufactured in a modular manner, and reliant on novel technologies to enhance operational safety (Figure 1). [1] These next-generation design attributes afford several advantages, including lower construction and operating costs, increased flexibility, and enhanced safety. Advanced nuclear reactors can be operated by a smaller team of operators and plant engineers, as the need for frequent manual operation configuration and maintenance monitoring will no longer be part of the reactor operational model. Ultimately, advanced reactors will need to be able to operate in an autonomous or semi-autonomous fashion, with human operators mainly playing a supervisory role. This requires the development of a new control paradigm.

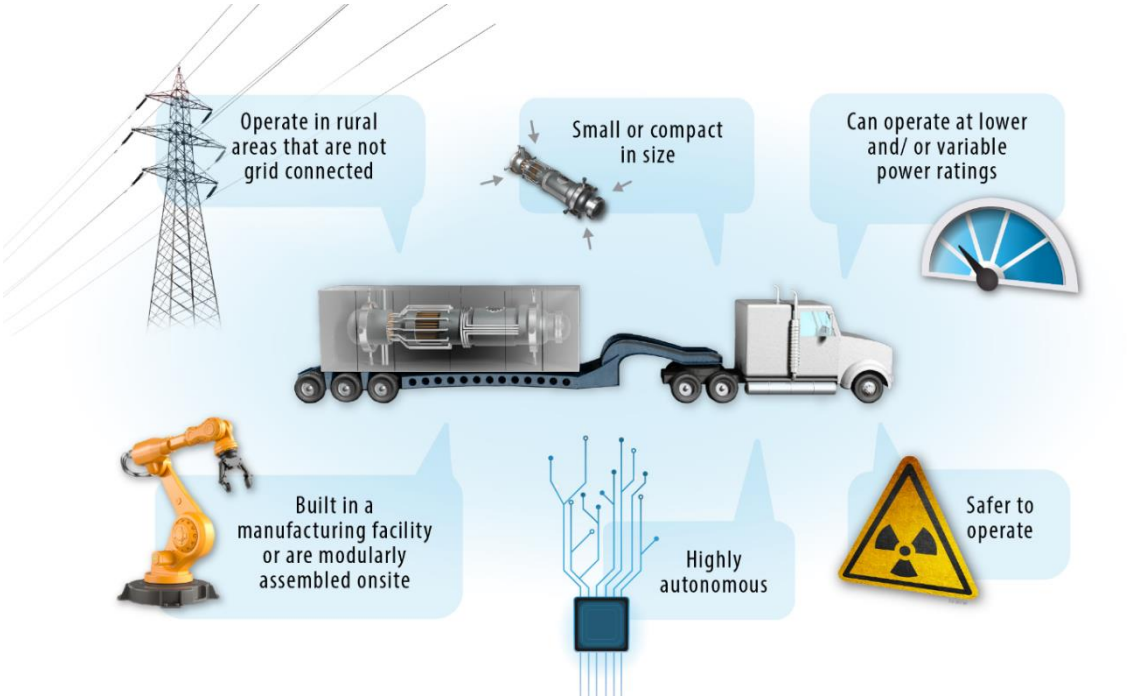


Figure 1. Design and operational features of advanced nuclear reactors.

The U.S. Department of Energy (DOE) Nuclear Energy Enabling Technologies Advanced Sensors and Instrumentation (NEET ASI) program is supporting research to enable this new control paradigm. The NEET ASI program anticipates several research needs that will have to be met in order to move advanced reactors from their current state to one in which autonomous operations can be developed, evaluated, and deployed. [2] To meet these needs, the program identified various technologies and methods that facilitate more autonomous operations when integrated into a generic framework for control (Figure 2). These technologies and methods include digital twins (a computational simulation of a process or system that can replicate the behavior of that process or system [3]), machine learning (ML) models, optimization methods, risk modeling, and various types of control methods.

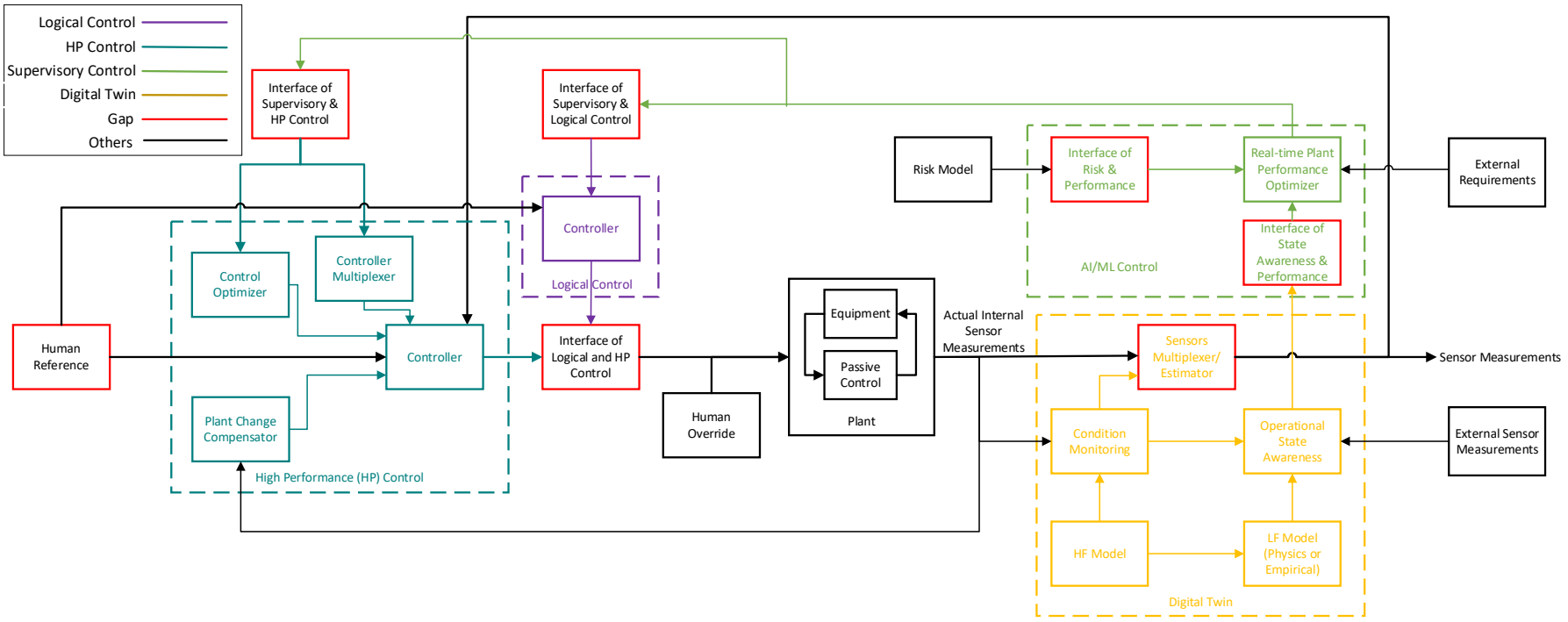


Figure 2. Framework for achieving autonomous operation of advanced nuclear reactors.

Integration is a significant factor in the development of control technologies and methods, and in evaluating their success in enabling autonomous operations of reactor systems. This includes integration of various control technologies and methods with each other and with hardware (both reactor system hardware and control hardware). At Argonne National Laboratory in July 2023, the NEET ASI program held an advanced control methods workshop that gathered together subject matter experts and stakeholders in the area of control methods/technologies. The objective of this workshop was to foster a comprehensive discussion on challenges and R&D needs in order to better focus the program research. A key finding from the workshop was that the research community lacks a platform to freely manipulate, test, and validate different approaches to the development of control methods and digital twins. Also, no existing open-source software was capable of integrating the various modules discussed in Figure 2 without introducing constraints. Among these constraints were proprietary or closed-source codes, unqualified or limited functionality, limited interfacing with other tools, and specialized knowledge requirements.

Motivated by these challenges, a previous phase of this effort developed and demonstrated a software platform called the Control and Optimization Modular Modeling Application for Nuclear Deployment (COMMAND), [1] which was capable of seamlessly integrating autonomous-control-enabling technologies and methods.

The present phase of this effort entails using the Microreactor Automated Control System (MACS) platform, which was developed by the DOE Microreactor Program, to serve as a control method testbed. [4] While MACS was not designed to mirror a specific microreactor (in order to enable it to be used by advanced reactor developers with other reactor technologies), its first embodiment was developed in concert with Idaho National Laboratory (INL)'s Microreactor Applications Research Validation and Evaluation (MARVEL) [5] microreactor. Thus, for the present effort, COMMAND was leveraged to enable MACS to mirror and emulate the physics of MARVEL, thus positioning MACS as a physical twin of MARVEL.

This report is organized as follows. Section 2 describes the existing infrastructure to be integrated into this demonstration platform, including MACS, COMMAND, and the MARVEL physics models. Section 3 describes how this integration was accomplished by employing COMMAND to create the physical twin, including COMMAND expansions and modifications. Section 4 describes scenarios used to test the integration of MACS, COMMAND, and the MARVEL physics models. The report concludes by summarizing the accomplishments made and suggesting future research directions.

2. EXISTING INFRASTRUCTURE

Demonstrating MACS as a MARVEL physical twin requires various hardware, software, and models. This section focuses on the existing components that were built upon to be able to complete this demonstration. The hardware involved in the demonstration encompassed the MACS system and its related components (Section 2.1). The software included the MACS software used for low-level control (described briefly in Section 2.1, though not the focus of this effort) and the COMMAND platform used for control development and to emulate the reactor physics (Section 2.2). The models utilized were the reactor kinetics and thermal hydraulics MARVEL models (Section 2.3).

2.1 MACS

The MACS hardware-in-the-loop testbed is a representative microreactor control system hardware simulator that was designed and constructed at INL under the DOE Microreactor Program. [4] MACS is intended to serve as a generic microreactor simulator for any microreactor, though is more representative of designs that use rotational drums to modulate reactivity. Although MACS is intended to be generic, it is currently configured to closely resemble the control system hardware reflected in the MARVEL microreactor design for this present effort. This includes the control drum motors, motor controllers, encoders, limit switches, and (to be added in the future) thermocouples.

MACS is a modular platform comprised of four main subsystems, three of which are shown and labeled in Figure 3. Subsystem 1 is the MACS control box, inside which the motor controllers, power supplies, and data acquisition equipment are mounted. Subsystem 2 is the MACS actuator cell, which includes the four control drum motors, position sensors such as encoders and limit switches, and control drum clutches. Subsystem 3 is the MACS reactor cell, which contains the control drums and the Visual Benign Reactor as Analog for Nuclear Testing (ViBRANT) Barrel light-based reactor core physics simulator and visualizer. The final system component is the MACS host computer, which hosts LabVIEW, the software that controls MACS and enables it to remotely communicate with external platforms (e.g., COMMAND running on INL's high-performance computing resources).

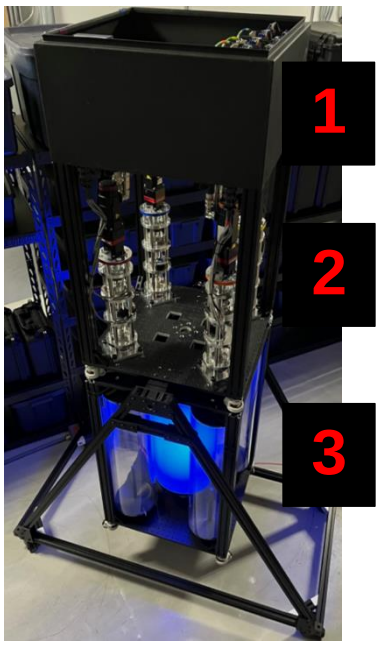


Figure 3. MACS hardware simulator.

The ViBRANT Barrel core physics simulator uses light-emitting diode (LED) arrays to represent reactor core physics (e.g., power, reactivity, or temperature). The ViBRANT Barrel core consists of 18

strips of 35 LEDs, each representing one of the outer fuel rods in a 36-rod core that is analogous to the core design used in the MARVEL reactor. The fact that each LED strip is controlled independently in terms of both light intensity and light color allows ViBRANT Barrel to represent various phenomena such as differing temperature gradients across the core. Figure 4 gives a close-up view of the illuminated ViBRANT Barrel core.

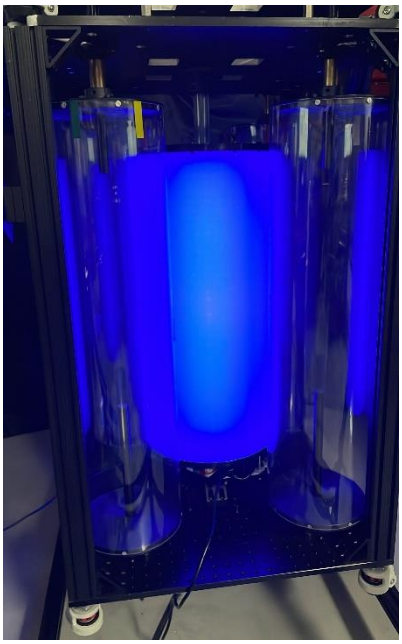


Figure 4. MACS reactor cell and ViBRANT Barrel core physics simulator.

2.2 COMMAND

COMMAND is a flexible, scalable software platform developed for use in designing, integrating, and testing advanced control technologies and methods so as to achieve autonomous reactor operations. It was designed to be accessible, with the goal of making it open-source and publicly available; modular, such that the software "pieces" can be combined and connected to create complicated use cases; and high-performing, thereby enabling simulations to take advantage of multi-core computers, servers, and nodes. COMMAND was written using the Python programming language because of Python's popularity, active community, and open-source and cross-platform compatibility.

COMMAND contains the following four primary modules, which act as adapters to integrate their respective methods and technologies together:

1. Control module: Examples include proportional integral derivative (PID) control and model predictive control (MPC).
2. Digital twin module: Examples include the MARVEL Reactor Excursion and Leak Analysis Program (RELAP5-3D) [6] and Monte Carlo N-Particle (MCNP) [7] reactor core models.
3. ML and optimization module: Examples include autoencoder-based anomaly detection or a genetic algorithm for periodically retuning controllers.
4. Hardware module: An example is the general remote procedure call (gRPC) protocol, a communication protocol for interfacing with hardware components (Section 3.2.2).

To design and run a simulation or scenario, COMMAND offers two main building blocks: variables, which are the main information blocks representing the data that are passed back and forth between systems; and systems, which are the main functional blocks that carry out numerical and algorithmic

actions on the variable data. The basic system building block was designed so as to be very general, thus accommodating a range of functionality. Example systems include basic mathematical functions (e.g., addition, subtraction, and stochastic functions) and all the modules' various functionalities (e.g., PID and MPC controllers, RELAP5-3D, anomaly detection, and gRPC communications). To connect systems to each other, each system can be assigned input and output variables. This approach ensures a highly modular design, thus enabling highly customizable and configurable tests.

To remain consistent with other simulation tools employed by the nuclear community, the COMMAND platform was designed to use text files rather than graphical interfaces in designing models and running and interacting with simulations. This means a text input file would contain the different variable and system function calls that define the respective variable/system properties and input/output variables via lines of text that adhere to a special structure and formatting style. The files are then read and interpreted, using the software's embedded instructions, to create and couple internal objects and classes, execute the simulation, and create text output files containing the generated results. Though less user-friendly, software that use text input and output files afford several benefits over software that use graphical interfaces. One such benefit is that the input files can be easily modified and manipulated via many different text editor programs, without the need for potentially proprietary software. In addition, the input files represent easily reproducible code that can be incorporated in and tracked by version control software. For COMMAND, the input files are written in native Python (a text-based language), and the output data are generated as comma separated values (CSV) files, a widely used format that can be read by text editors and spreadsheet software and postprocessed natively by most programming languages.

The COMMAND platform plays three important roles in configuring and running simulations and scenarios. First, it establishes the architecture and templates needed to support a diverse range of modules and applications. Second, it compiles a library of functions, parsers, and adapters to integrate the methods and technologies that comprise the modules. Finally, it schedules, coordinates, and synchronizes the components so as to enable them to communicate with each other and process the data. Figure 5 gives a high-level overview of this process. For a more in-depth discussion and code examples, refer to [1].

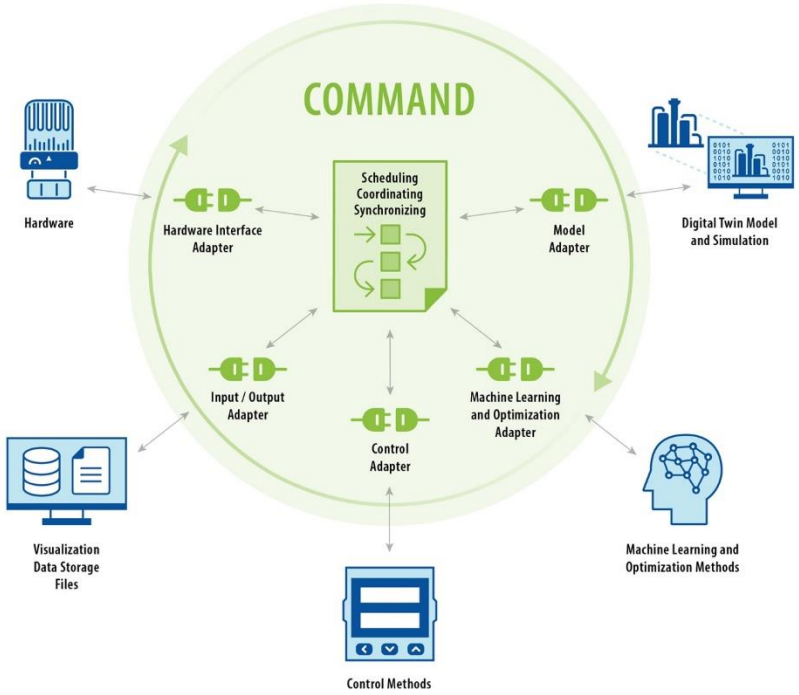


Figure 5. High-level overview of the COMMAND platform.

2.3 MARVEL

The MARVEL microreactor is an 85 kW thermal (nominal) research reactor that utilizes 36 fuel rods containing uranium-zirconium-hydride (UZrH) fuel. [8][9] The core features metallic beryllium, beryllium oxide, and graphite neutron reflectors and is controlled via four control drums containing boron carbide neutron absorbers. The fuel, purchased from TRIGA International, contains 30 wt.% uranium, with 19.75% enriched high-assay low-enriched uranium, and is clad in 304 stainless steel. MARVEL is a natural circulation system with liquid metal sodium-potassium coolant. It is designed to generate 20 kW of electricity via four Stirling engines and to produce high-grade heat for industrial processes.

The microreactor is small, with a total height of approximately 15 ft., a core diameter of less than 12 in., and a core length of approximately 3 ft. A 3D model of MARVEL is shown in Figure 6. The 36 fuel rods are arranged in three concentric hexagonal rings, with an additional central void included so as to accommodate a movable shutdown rod (depicted in Figure 7 using outputs from the MARVEL MCNP model). The relative location of the control drums and reactor core in MARVEL are shown in Figure 8.

Connecting MACS and MARVEL, the outer hexagonal ring of fuel in MARVEL consists of 18 fuel rods. In MACS' ViBRANT core, these 18 rods are depicted as lights (see Section 2.1), as if looking into the core from the outside, with the outer ring of fuel rods exposed. MACS also includes a hardware version of MARVEL's four control drums, along with their associated control hardware, and these drums can be individually manipulated using the MACS software, as is also the case with MARVEL.

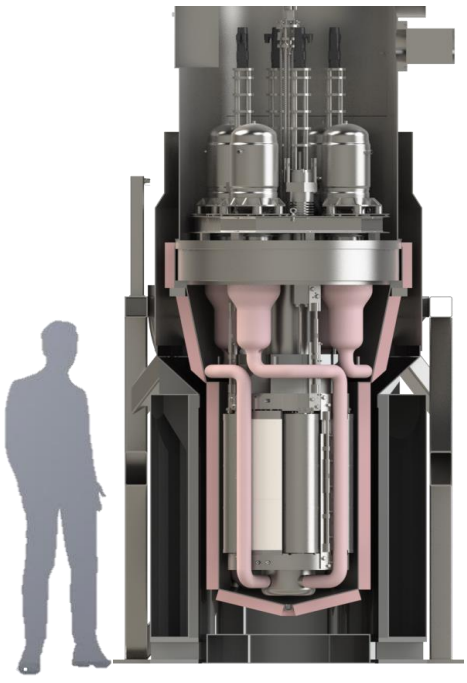


Figure 6. 3D model of the MARVEL microreactor (human silhouette included for scale). [8]

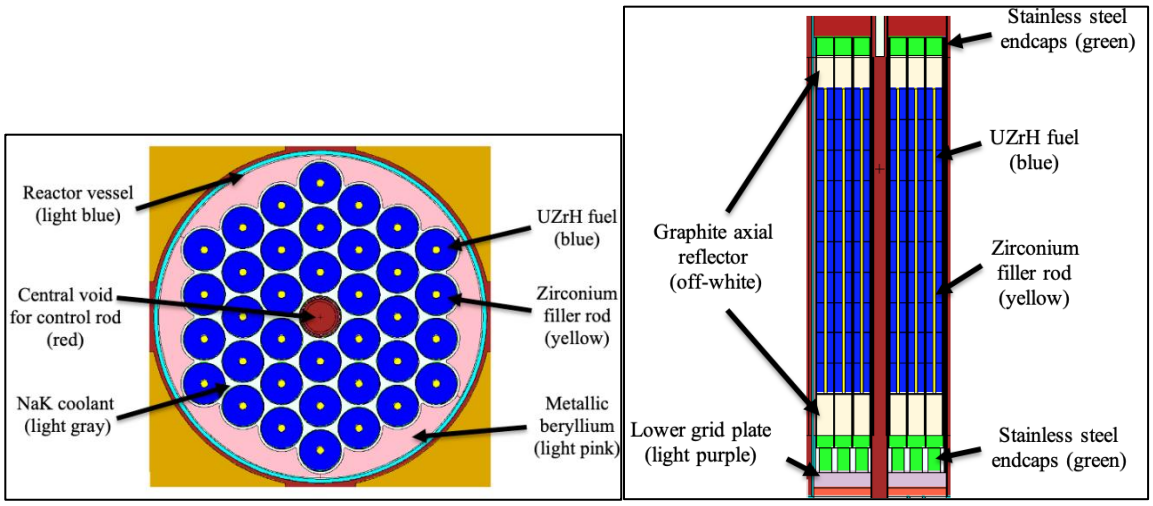


Figure 7. Radial (left) and axial (right) plots of the MCNP model of the MARVEL microreactor core.

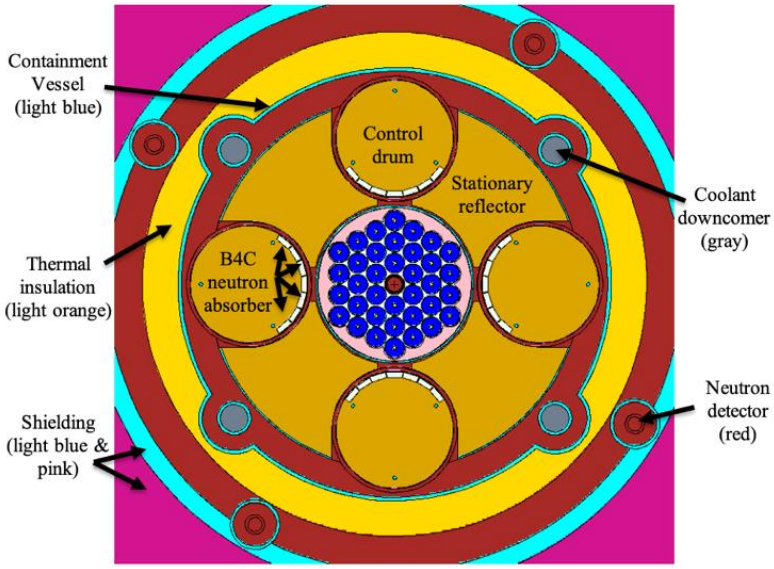


Figure 8. Radial plot of the MCNP model of the MARVEL microreactor.

2.3.1 MCNP

The MCNP continuous energy transport code (version 6.2) [7] was used to design and analyze the MARVEL reactor core. Transport simulations in MCNP are set up via a text file containing the following: the geometry, material specification and cross-section selections, location and type of particle sources, and "tallies" that are used to extract localized data from the model for analysis.

The input file is composed of text lines of code called "cards", each of which contains a maximum of 128 characters. Multiple consecutive lines of input may be linked by inserting a special character. These input cards contain a unique keyword to indicate the selected parameter being initialized, along with a series of numerical inputs or additional keyword value pairs. Among the multiple types of cards are title cards, cell cards, surface cards, and data cards. The MCNP input is divided into three sections that are separated from each other in the input by a blank line with no inputs. These sections correspond to cells, surfaces, and data (for material specification, physics model specification, and "tally" definitions). The input is terminated by the end of the file. The combination of surface, cell, and material input cards

enables users to create a 3D model of materials in which the particle transport occurs and can be analyzed. MCNP allows for 3D model generation, thanks to its native constructive solid geometry capabilities. These models are used to build general shapes and fill them with materials in order to represent the microreactor core and supporting systems.

MCNP offers two major types of simulations: criticality (called KCODE) and source-driven simulations. Most microreactor analyses are performed based on criticality calculations, as designers are most interested in nuclear reactor system characteristics when the reactor core is critical. With just a few lines of input, the models can be quickly switched between criticality and source-driven calculations, depending on the desired analysis. Different transport physics for different particles can be enabled or disabled, depending on the simulation and the analysis being performed. In simulating the microreactor system, neutron transport is used as the default. Although MCNP is capable of time-dependent simulations, the analysis performed for this work was a steady-state analysis with no dependence on time.

The default output from MCNP is a text file that summarizes the input, contains information dumps pertaining to the process calculations performed in the simulation, and presents the results. Included in the default output are text results for any user-defined tallies contained in the input file prior to the simulation being executed. Additional files may be generated as outputs containing mesh data and corresponding simulation results, depending on the types of tallies requested in the input.

For this effort, the MCNP particle transport software calculated the criticality produced in the MARVEL microreactor model and generated power distributions for the 36 fuel rods in the core. Figure 9 shows a plot of the radial rod power distribution for a single configuration of the core. The core configurations were achieved by moving the control drums so as to obtain a core power distribution for each combination of the four control drum angles of rotation. MCNP does not directly link with MACS through the COMMAND program, as it would be too slow to run in real time. Hundreds of core configurations and the resulting power distributions were calculated and postprocessed into a database for use with the MACS surrogate model, as described in Section 3.1.1.

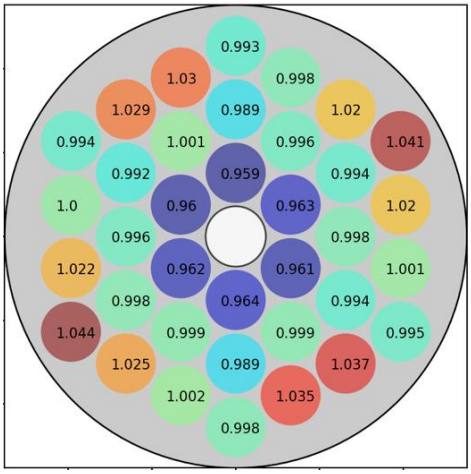


Figure 9. 2D rod power distribution of the MARVEL microreactor core.

2.3.2 RELAP

RELAP5-3D [6][10] was used to couple the behavior of a reactor's coolant system, core, and secondary coolant loop under various operational scenarios. RELAP5-3D simulations are defined via a text-based input file that can be edited by COMMAND. Also, each time a simulation run concludes, RELAP5-3D generates a printed output file readable by COMMAND. This file records the calculated simulation results. A restart file, given in uncompressed machine-dependent binary format, contains the information required to start a future simulation run from a point already calculated in the previous run.

This point can be the first record, written immediately after input processing has been conducted for the previous run; the last record, written at end of the previous run; or some record in between.

The input file is composed of text lines, or "cards," containing 96 characters each. Among the different types of cards are the title card, optional comment cards, data cards, and terminator card. Data cards are subdivided into different fields called "words." The first word of a data card is always an identification number that is determined in the user manual, in accordance with the type of data being entered (e.g., time step control, model architecture, output format). The number of fields on each data card varies, with each field representing a model or simulation parameter specification.

Problem type and problem option are two crucial parameters to be specified. The problem type specifies whether the simulation will resume from a point recorded in a restart file, or whether a new simulation will be started using the initial model conditions provided in the input file. The problem option specifies whether the simulation will follow steady-state or transient conditions. The simulation is executed according to the chosen problem type and problem option, as well as to all the other parameters specified in the input file.

The output file generated by a simulation lists the input used to start the simulation, the main variable calculation results (called "major edits") as they advance in time, and the user-specified variable calculation results (called "minor edits") as they advance in time. These printed output files can become quite large; thus, RELAP5-3D provides users with a plotting option for presenting and documenting important data in the form of time history plots. In the input file, users can specify the plotting parameters and direct the program to write the calculated simulation results to a plot file.

A previous effort developed an interface between COMMAND and the MARVEL RELAP5-3D model. [1][11] This interface interacts with RELAP5-3D input files whose problem type parameter is set to "restart" and whose problem option is set to "transient". This enables COMMAND to run RELAP5-3D in batches, and to read and/or modify key variable values (e.g. drum positions, reactivity coefficients, and desired power) in between batches, per the scenario being investigated. The interface uses plot file outputs to extract time history quantities of variables of interest at given model locations. Details on these interactions between COMMAND and RELAP5-3D are provided in Section 3.1.3.

3. ENABLING THE PHYSICAL TWIN

In the present effort, a physical twin was developed for the advanced reactor community to use as a demonstration and evaluation platform when creating new advanced control technologies and methods. The physical twin was created by interfacing the existing infrastructure presented in the various subsections of Section 2, as well as developing new capabilities described in Section 3. Integration of control technologies and methods was enabled by COMMAND, and a realistic physical testbed for evaluating these technologies and methods under various operating conditions was realized by coupling MACS with the MARVEL physics models (i.e., the MARVEL MCNP and RELAP5-3D models).

A previous phase of this effort [1] developed an initial adapter between MARVEL's RELAP5-3D model and COMMAND. However, that initial model was constrained to symmetric operational scenarios (i.e., it assumed a uniform and symmetric core power distribution). To enable advanced control technologies and methods to be evaluated under nonsymmetric operating conditions, the current phase of this effort modified the adapter between RELAP5-3D and COMMAND, and created a new adapter between MCNP and COMMAND (see Section 3.1 for details). In addition, an adapter between MACS and COMMAND was created for interfacing with the hardware (see Section 3.2 for details). The interconnections among these systems and adapters are shown in Figure 10.

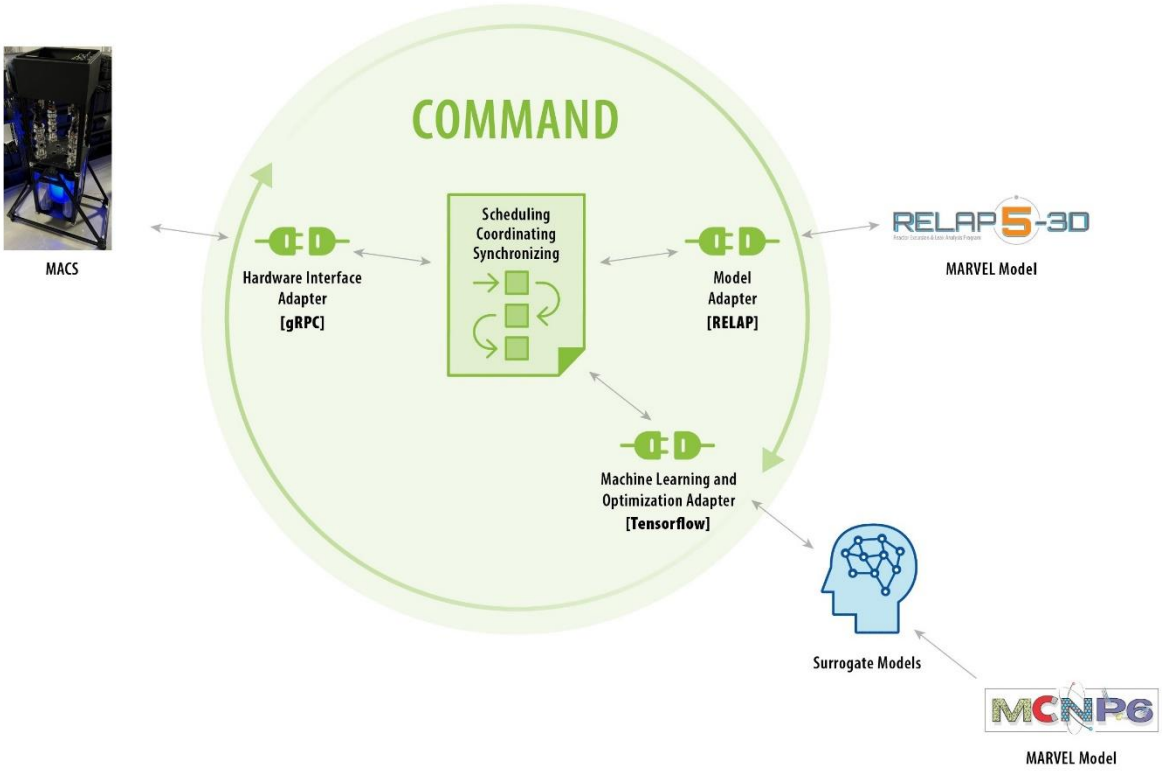


Figure 10. Overview of the systems and adapters used in the demonstration conducted as part of the present effort.

3.1 Interfacing Models

This section focuses on developing the MCNP surrogate model (Section 3.1.1), and how the MCNP (Section 3.1.2) and RELAP5-3D (Section 3.1.3) models were interfaced into COMMAND for use with the physical twin.

3.1.1 MCNP Surrogate Models

The MCNP model discussed in Section 2.3.1 generates state-of-the-art numerical simulation results for the MARVEL core. However, each run of the model (representing a different set of MARVEL operating conditions) took approximately 5 minutes when using 960 computing cores. In the present effort, the operating conditions could be changed every few seconds, meaning that direct use of the MCNP model to perform calculations would have resulted in too high a computational cost. Furthermore, it was not generally reasonable to require that users have such a significant number of cores available.

As an alternative, this effort utilized the surrogate model concept, with a limited number of MCNP simulations being run over a range of operating conditions in order to capture the desired parameter responses as a function of those operating conditions. The surrogate model could then, at a fraction of the computational cost, be used to interpolate values of those parameters in between the evaluated MCNP runs.

In this effort, the four MARVEL drum rotations were the operating conditions that varied from one MCNP simulation to the next. Each drum rotates independently between 0 and 180 degrees, and this range was discretized into six points (including both ends) to capture the full range. This resulted in $4^6 = 1{,}296$ total operating conditions. Due to accounting for symmetry in the core, only 666 total operating conditions were run, with the remainder being calculated based on their symmetric operating conditions. From these simulations, two parameters of interest were captured: the k-effective (k_{eff}) and spatial power peaking values. The MCNP software was installed on INL's high-performance computing cluster, Sawtooth. Each simulation was run for 5–6 minutes on 960 computing cores, resulting in about 80 core hours per simulation being used.

3.1.1.1 K-effective Parameter

The k_{eff} parameter, which was the first parameter of interest used from the MCNP simulations, defines the average number of neutrons from one fission reaction that cause another fission reaction, and is dependent on factors such as neutron absorbers, neutron reflectors, geometry, materials, and thermal conditions. Those factors can all be changed in the MCNP file, and the k_{eff} value is extracted for every simulation. The reactor is said to be supercritical, critical, or subcritical based on whether the k_{eff} value is more than, equal to, or less than one—indicating a neutron population that is growing, holding steady, or decreasing, respectively. By extracting this value from the MCNP model, the present effort achieved an accurate mapping between the drum positions and the reactor neutron kinetics.

3.1.1.2 Spatial Power Peaking Parameter

The second parameter of interest was the spatial power peaking factor distribution. The MARVEL core has 36 fuel rods, each modeled using a single cross-sectional node and 10 axial nodes, for a total of $36 \times 10 = 360$ nodes. The peaking factor parameter defines how the total core power (calculated using the RELAP model) is distributed among those 360 nodes, providing the spatial core power distribution. The numbers provided from MCNP are relative values (with the average of all 360 nodes being equal to one), but can be normalized to be relative powers such that their sum equals one. Extracting those values enables the modeling of reactor asymmetries caused by asymmetric drum positions (possibly as a result of drum degradation).

3.1.1.3 Modeling Approach

The results of the MCNP simulation runs were 1,296 sets of drum positions, each with an associated k_{eff} value and 360 associated peaking factor values. The next step was to develop a modeling approach to learn these values and be able to interpolate between them.

The initial approach investigated in this effort was to store the associated sets of drum positions and parameters, and to use spatial interpolation in an online manner (i.e., interpolation built into

COMMAND). Spatial interpolation, which is the extension of linear interpolation to multivariate problems, calculates new values as a weighted average of their nearest neighbors, with weights proportional to the distance from those neighbors. The challenge with this approach was that it could only predict one value per calculation, making it slower at predicting the 360 peaking values.

To overcome this challenge, the surrogate model used was a feedforward neural network (FNN), a type commonly employed in ML applications. FNNs are universal function approximators that, when given sample data pertaining to nearly any function, can learn to approximate that function, with no restrictions on output shape or size. FNNs have an input layer, which defines the input shape (for both models, the input is the four drum positions); usually one or more hidden layers, each with a specified number of neurons (which defines how complex a function they can approximate); and an output layer (for the k_{eff} model, this is a single value; for the peaking values model, this is 360 values). These layers have associated model coefficients that are learned (a process often called training the model) by using sample data to approximate the desired function.

For this effort, the number of hidden layers and the number of neurons contained in each were determined using cross validation, a process in which some of the sample data are employed to train the model, while the rest are used to test whether the model generalizes well to unseen sample data. This can be done for different combinations of numbers of hidden layers and neurons, and the simplest one that produces a small error in regard to the unseen data is selected. One challenge here was that there were only 1,296 total samples, all of which were samples on which the model should ideally be trained. To overcome this challenge, the above-described spatial interpolation approach was used to generate additional points usable for cross validation. Via this approach, the k_{eff} and peaking value models each performed well with a single hidden layer containing 500 and 10,000 neurons, respectively.

The results of this process (which was conducted outside the COMMAND framework) were two FNN surrogate models trained using the Python TensorFlow library. Each of these models takes as input the four drum positions, then output the k_{eff} and peaking values, respectively. The models were saved using TensorFlow's custom Keras file format for saving TensorFlow models, and these files could then be coupled with COMMAND by using the TensorFlowModel system class described in Section 3.1.2.

3.1.2 Interfacing of the MCNP Surrogate Models to COMMAND

To interface the MCNP surrogate models to COMMAND, a new TensorFlowModel system class was developed. As with all systems in the COMMAND platform, this class had input and output keywords to define (1) the variable information being fed into the model, and (2) variables that are assigned values by the model output, respectively. In addition, this class had a file_name keyword specifying the path to the Keras file that loads the TensorFlow neural network model.

Example: Figure 11 shows an example of the two MCNP surrogate models being used in COMMAND. As mentioned, the inputs to both models are drum angles, and the outputs are the k_{eff} and peaking values.

```
1  drum_angles = ['drum_angles1', 'drum_angles2', 'drum_angles3', 'drum_angles4']
2  keff_surrogate_path = 'keff_surrogate_model.keras'
3  peaking_surrogate_path = 'peaking_surrogate_model.keras'
4  base.Variables(*drum_angles, 'k_eff')
5  base.VariableArray('peaking_values', size=360)
6  mlo.TensorflowModel(name='tensorflow1', inputs=drum_angles, outputs='k_eff',
     file_name=keff_surrogate_path)
7  mlo.TensorflowModel(name='tensorflow2', inputs=drum_angles, outputs='peaking_values',
     file_name= peaking_surrogate_path)
```

Figure 11. Example showing the TensorFlow model class applied to the MARVEL model.

3.1.3 Interfacing of the RELAP Model to COMMAND

The interface between the RELAP5-3D model and COMMAND was developed in the previous phase of this effort, [1] via COMMAND's RELAP5-3D class. However, modifications to the RELAP5-3D class were required to accurately represent reactor states (e.g. reactor power and reactivity) under nonsymmetric drum positions. At each simulation step, the updated RELAP5-3D class modifies the input file to specify the value of certain parameters, including the excess reactivity inserted by current drum positions, the time step size, and the current simulation time. The location of these parameters in the input file (i.e. card numbers and word numbers) along with their assigned values are specified as input variables in the class definition.

The value assigned to the drum excess reactivity parameter ($\Delta\rho$, in units of dollars) is calculated in COMMAND based on the k_{eff} obtained from the MCNP surrogate models and MARVEL's delayed neutron fraction (β), per the following relationship:

$$\Delta\rho = \frac{k_{eff} - 1}{k_{eff} * \beta} \tag{1}$$

Due to the text-based nature of the input file, the class contains a parser that locates the parameters specified as input variables, then modifies the values entered in those fields according to the input variables specifications.

The simulation engine is fed both the updated input file and a restart file that contains model state data from an earlier step. It is worth noting that the first simulation is fed a restart file generated prior to the beginning of the simulation by running the MARVEL model under steady-state conditions outside of COMMAND. At the end of each simulation step, a plot output file is produced, arranging output information into blocks of parameter data for each time step. Data are provided for a very extensive number of parameters, and data blocks cannot be directly processed by most plotting programs. Thus, the RELAP5-3D class contains another parser to convert the data blocks into columns and also filter which parameters are to be converted (e.g., reactor power). The location of these parameters of interest in the output plot file (i.e. parameter name and model location) is specified as output variables in the class definition.

Input and output class variables help RELAP5-3D achieve generalizability. Although this effort was concentrated around the MARVEL RELAP5-3D model, the interface between the RELAP5-3D model and COMMAND was built to handle any RELAP5-3D code.

Example: Figure 12 shows an example of the RELAP5-3D class being applied to the MARVEL model. The input to the model is the excess reactivity inserted by the drum rotations, as calculated from the k_{eff} values obtained in Figure 11. In this example, the model output is the reactor power, though other reactor states and simulation parameters may also be output if needed.

```
1  def get_reactivity(k_eff):
2    return (k_eff – 1)/((k_eff) * 0.00751)
3
4  input_variables = [{'card_number': 20299101, 'word_number': 2, 'reference_value': 0.0}]
5  output_variables = [{'location': 0, 'variable': 'rktpow'}]
6  base.Variables('k_eff', 'reactivity', 'reactor_power')
7  base.UserDefined (name='reactivity_calculation', inputs='k_eff', outputs='reactivity',
     function=get_reactivity)
8  dt.RELAP53D(name='marvel', inputs='reactivity ', outputs='reactor_power',
     relap_file_directory='RELAP/', input_variables=input_variables,
     output_variables=output_variables)
```

Figure 12. Example showing the RELAP5-3D class being applied to the MARVEL model.

3.2 Interfacing of MACS to COMMAND

Once the models were interfaced with COMMAND, next the MACS hardware needed to be interfaced with COMMAND. This included converting the reactor power scale into a light intensity value (Section 3.2.1) and setting up a communication protocol between MACS and COMMAND (Section 3.2.2).

3.2.1 Rescaling of the Power Distribution for MACS

MACS uses strips of LED lights to represent a fuel rod placed within the MARVEL reactor core. Due to data serialization constraints, it was not yet possible to prescribe a light intensity value for every LED light within each strip. However, a peak light intensity could be prescribed for each strip of LED lights, then that peak intensity could be modulated into a cosine wave along the axial length of the strip. This enabled the LED lights to emulate the true cross-sectional distribution and approximate the axial distribution of the core power.

To prescribe the LED-strip light intensities, the total core power (calculated from the RELAP model) and peaking factors (calculated from the MCNP surrogate model) were used to calculate the individual rod powers, which were then scaled to light intensity values. First, to calculate a given rod power, the peaking factors for that rod were summed and then divided by the sum of all the peaking factors:

$$P_{rod} = \frac{\sum_{i \in rod} PF_i}{\sum_{j=1}^{360} PF_j} P_{total} \tag{2}$$

where P is the power in watts and PF is the peaking factor. Next, to calculate a given strip's light intensity, each rod power had to be converted into a percent power between 0 and 100, which required the minimum and maximum rod powers corresponding to those percentages. The minimum rod power used was 0 kW, and the maximum total core power used was the MARVEL reactor's maximum design power of 85 kW. To convert this into a maximum rod power, the total power was divided by the number of rods (36) and then multiplied by the highest ratio of maximum to average rod power (factor of 1.27), as calculated by the MCNP peaking factor data:

$$P_{max} = \frac{85000}{36} \times 1.27 \tag{3}$$

$$I_{strip} = 100 \times \frac{P_{rod}}{P_{max}} \tag{4}$$

where I is light intensity. These light intensity values were then sent to the MACS LabVIEW controller to update the corresponding LED light powers via the communication protocol described in Section 3.2.2.

3.2.2 Communication Protocol for MACS

Communication between COMMAND and MACS is structured in the form of a common computer network "client-server" relationship. In this relationship, a client initiates communication with the server and submits requests to it, while the server provides the requested service to the client. The specific protocol chosen to enable communication between MACS and COMMAND was gRPC, an open-source remote procedure call (RPC). [12] An RPC is a software communication protocol that one program can use to request a service from a program located on another computer connected to the same or different network, without having to understand the network's details. gRPC allows a client application to call a method on a server application hosted on a different machine as though it were a local object. In this case, it enables COMMAND, running on INL's high-performance computers, to communicate with MACS as though MACS and COMMAND were on the same machine. gRPC is based on the concept of defining a common service for both the client and the server. The service defines the methods—meaning the messages and data—a client can request from the server. A key highlight of gRPC is that it enables

interaction between many popular programming languages. The client and server gRPC implementations can be written in different programming languages yet still seamlessly talk with each other. Importantly, gRPC supports both Python and LabVIEW. This enables seamless communication and integration between MACS—whose control software is programmed in LabVIEW—and COMMAND, which is programmed in Python. The decision to select gRPC as the communication method between COMMAND and MACS was made jointly between this team and the MACS hardware team. This team built the gRPC adaptor for COMMAND, while the MACS hardware team built the gRPC interface in LabVIEW to the MACS hardware.

In the case of MACS and COMMAND, MACS's controller in LabVIEW acts as the server (running a gRPC-based server), while COMMAND acts as the client making requests of the server. To accomplish this, a gRPC adaptor was added to COMMAND in order to make client requests. While COMMAND and RELAP5-3D were hosted and run on INL's high-performance computers, the MACS host computer resided on INL's corporate network. These two networks can communicate with each other via a secure shell (SSH) tunnel, through which gRPC messages are sent between MACS and COMMAND. Figure 13 diagrams the basic architecture of the MACS-COMMAND interface.

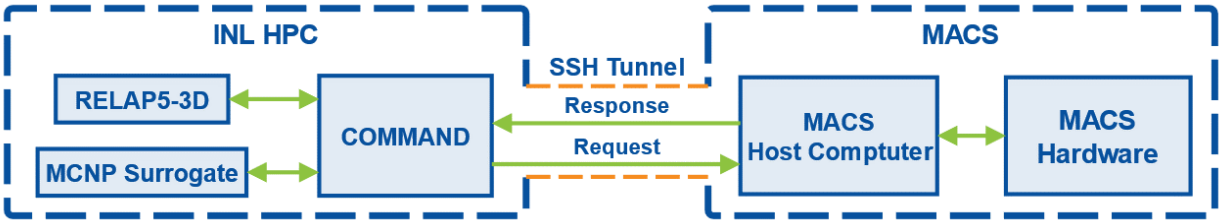


Figure 13. Diagram of the MACS-COMMAND communication interface for milestone testing.

Two request types are used in the MACS-COMMAND interface: measurement requests and command requests. In a measurement request, COMMAND sends a message containing the names of the signal values desired from MACS. Upon receipt of the message, the gRPC server running on the MACS host computer will pull a current measurement for each signal specified in the measurement request message. A message containing the requested signal name and corresponding measurement value will then be sent back from the MACS host computer to COMMAND. Figure 14 shows the signal flow for a measurement request.

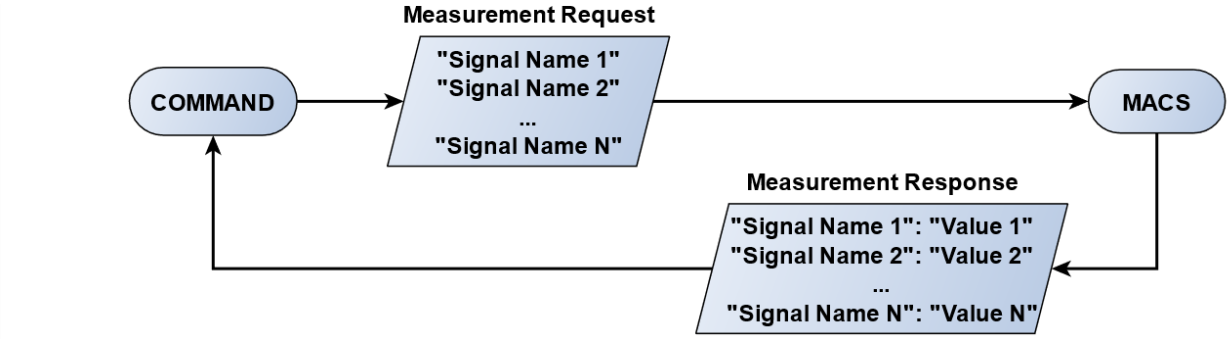


Figure 14. Measurement request signal flow.

The second request type is a command request. In a command request, COMMAND sends a message containing the signal names whose setpoint values are to be changed, along with the accompanying new values. The message is sent to the gRPC server running on the MACS host computer. Upon receiving the new value, MACS updates the setpoints to the received values, and a message is returned from MACS to

COMMAND acknowledging the changed set points. Figure 15 shows the signal flow for a command request.

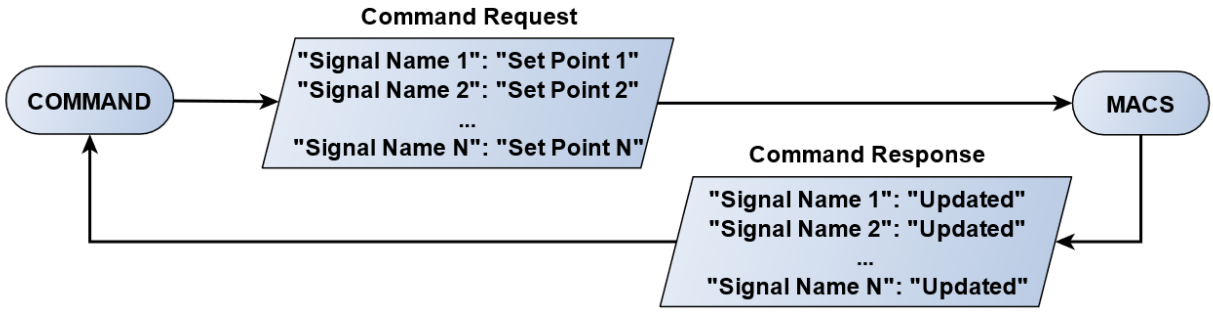


Figure 15. Command request signal flow.

Examples: Figure 16 shows an example of the gRPC communication module being used in COMMAND for making measurement requests. The inputs to the function are the IP address and port of the server, and the names of the channels requested. The outputs of the function are the received measurement.

```
1  measured_positions = ['drum_angles1', 'drum_angles2', 'drum_angles3', 'drum_angles4']
2  measured_positions_channels = ['Drum 1 Resolver Position', 'Drum 2 Resolver Position',
   'Drum 3 Resolver Position', 'Drum 4 Resolver Position']
3  grpc_port = 5901
4  base.Variables(*measured_positions)
5  hw.gRPC(name='grpc1', mode='client', ip_address='localhost', port=grpc_port,
     method_class='GetDouble', outputs=measured_positions,
     channels=measured_positions_channels)
```

Figure 16. Example showing a measurement request using gRPC in COMMAND.

Figure 17 shows an example of the gRPC communication module being used for making command requests. The inputs to the function are the IP address and port of the server, the data to be sent, and the names of the channels for which setpoint values will be updated. There are no outputs.

```
1  setpoint_light_channels = ['LED Intensity Pin 1', 'LED Intensity Pin 2', 'LED Intensity Pin 3',
     'LED Intensity Pin 4', 'LED Intensity Pin 5']
2  grpc_port = 5901
3  base.VariableArray('setpoint_light_array', size=5)
4  hw.gRPC(name='grpc2', mode='client', ip_address='localhost', port=grpc_port,
     method_class='SetDouble', inputs='setpoint_light_array', channels=setpoint_light_channels)
```

Figure 17. Example showing a command request using gRPC in COMMAND.

4. TESTING THE PHYSICAL TWIN

Two demonstration scenarios were conducted with the physical twin in order to demonstrate integration of the MACS hardware, COMMAND, and the MARVEL physics models. In Scenario 1, the initial conditions were that the reactor was in a critical state at 85 kW of thermal power (i.e., 100% power), with corresponding drum angles of 127.5°. Next, one control drum was moved, resulting in reduced reactivity and consequently power. This drum was moved twice to slowly reduce power while accounting for temperature feedback. Once the power reached approximately 13 kW, that same drum was rotated back, increasing reactivity and power back towards the 100% power level. Only one drum was moved during this scenario because the system physics are sensitive to small drum rotations, so manual control using all four drums at this stage would have been difficult.

Scenario 2 demonstrated COMMAND's ability to compute individual fuel rod powers, send those values to MACS, and have MACS display each respective power level. The scenario chosen to demonstrate this capability utilized differing drum positions on opposite sides of the core in order to induce a reactor power tilt across the core. Drums 1 and 4 were set to positions of approximately 105°, while drums 2 and 3 were set to 180°. According to neutronics calculations computed in MCNP, these drum positions should induce a 12% power difference between the rods on one side of the core and those on the other. The following sections present the results of the test scenarios.

4.1 Scenario 1: Reactor Power Range Testing

In Scenario 1, MACS and the MARVEL physics models were brought from 85 kW of thermal power down to 13 kW and back via physical rotation of the MACS control drums. For this test, the lights were set to output the color blue for all power levels, with 0% and 100% light intensity outputs corresponding to reactor powers of 0 kW and 85 kW, respectively. Figure 18 presents the position of each control drum throughout the test, while Figure 19 shows the reactor power level throughout. The test has been divided into five phases. Phase 1 occurred at time 0–4 seconds. In this phase, the control drums were at 127.5°, and the reactor power was at 85 kW. Phase 2 occurred during time 4-36 seconds, with control drum 1 moved to 117°. During this phase, power initially decreased rapidly due to the reactivity removal, but over time, the rate of decrease slowed as the fuel temperature decreased and consequently reactivity increased due to the negative temperature feedback coefficient. Phase 3 took place from 36–64 and was initiated by control drum 1 rotating to 97°. Once again, power had an initial rapid power decrease that slowed due to temperature feedback. The final power level reached during this phase was 14 kW. During Phase 4, which took place from 64–112 seconds, control drum 1 was rotated to 125°, inserting reactivity. Because the reactor was then in a super critical state, power began to rise. The final phase, Phase 5, lasted from time 112 seconds through the end of the test and began with control drum 1 rotating out by 1° to 124° in order to slow the rise of power and thus avoid an overshoot of the 100% power level (i.e., 85 kW). Figure 20 (a) and (b) show the MACS lights at minimum intensity and full intensity, respectively.

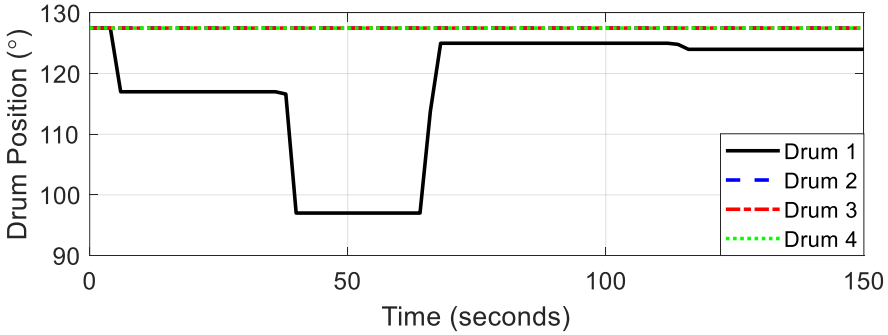


Figure 18. Drum positions for reactor power range testing.

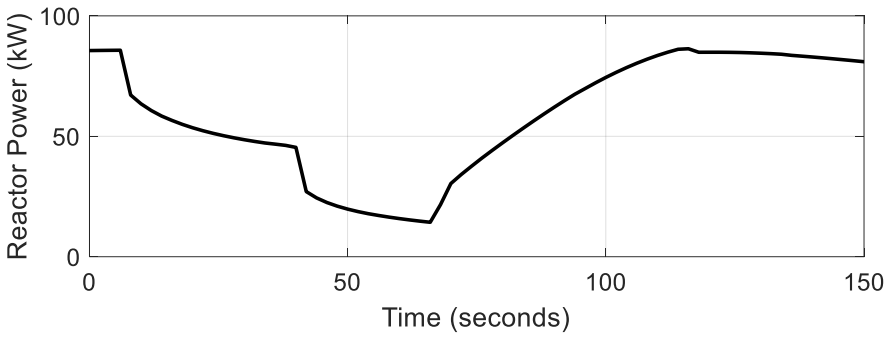
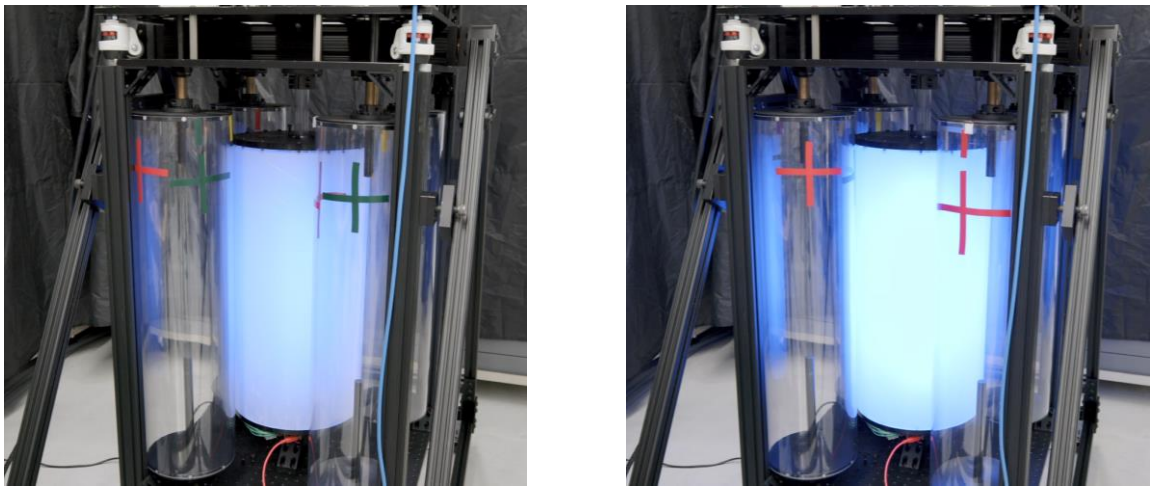


Figure 19. Reactor powers for reactor power range testing.



(a) Drums in the minimum power position, and the lights outputting at minimum intensity

(b) Drums in the full-power position, and the lights outputting at full intensity

Figure 20. MACS testbed with the drums in two different power positions and the lights outputting two different intensities, as received from COMMAND and RELAP5.

4.2 Scenario 2: Reactor Power Tilt Testing

One of the objectives of Scenario 2 was to visually demonstrate the reactor power tilt test. During the initial test, this was not as evident to the naked eye as desired, and so a second iteration was conducted with an exaggerated scale to make the tilt more visually apparent.

4.2.1 Normal-scale Testing

Integration of an MCNP surrogate model with COMMAND enables the computation and display of individual fuel rod power levels, using the LEDs in the ViBRANT Barrel core. To highlight the gradient across the core, the lights were set to output different colors at different reactor power levels (as opposed to blue light and varying light intensity as done in the previous test). For this test, blue light corresponded to 0 kW of reactor power, purple light was 42.5kW of reactor power, and red light was 85 kW reactor power. Testing of Scenario 2 demonstrated the capability to compute individual fuel rod powers, send those computed values to MACS, and have MACS display each respective power level. Scenario 2 should produce a power tilt from one side of the core to the other, due to the asymmetric positioning of the control drums. Similar to Scenario 1, all four drums began at 127.5°, and thermal power began at 85 kW. Drums 1 and 4 were moved to a position of approximately 104°, dropping reactor power to 23 kW. Then,

drums 2 and 3 were moved to a position of 180°. This move raised the power only slightly, so drums 1 and 4 were used to further insert reactivity by moving them from 104° to 105°. This brought the reactor to a super critical state, and power began to rise. As power approached 100% and reactor temperature rose, reactivity dropped towards the critical level and the power level began to stabilize. At the conclusion of the test, the power level was 81 kW. Figure 21 shows the control drum positions for Scenario 2, Figure 22 gives the average reactor power level, and Figure 23 shows the light intensity level for each of the 18 outer fuel rods. The initial symmetric drum positions generated maximum and minimum light intensity outputs of 98.1% and 93.5%, while the final asymmetric drum positions generated maximum and minimum light intensity outputs of 96.5% and 85.0%. This almost 12% difference in light intensity output when comparing the highest- and lowest-intensity fuel rod surrogates was very difficult to distinguish with the naked eye. Thus, the test was repeated with exaggerated light scaling to make the gradient across the core more obvious. Details on this follow-on test are given in Section 4.2.2.

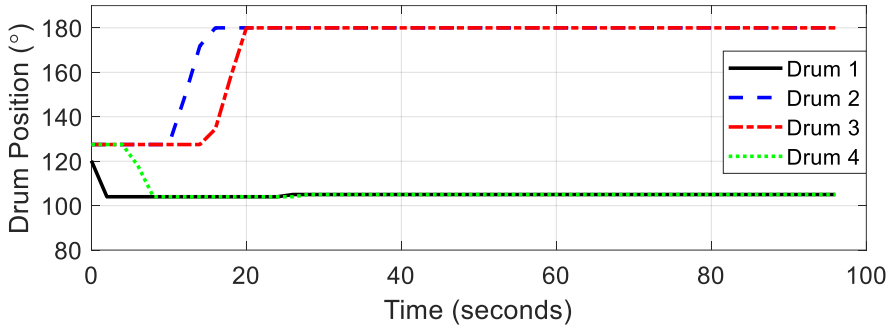


Figure 21. Drum position for the normally scaled reactor tilt testing.

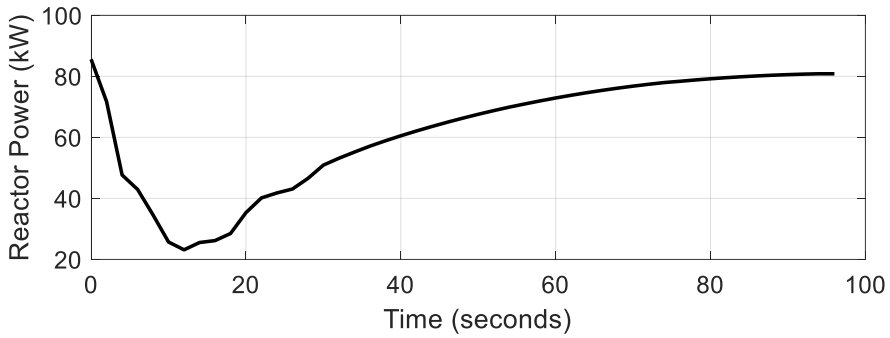


Figure 22. Reactor power for the normally scaled reactor tilt testing.

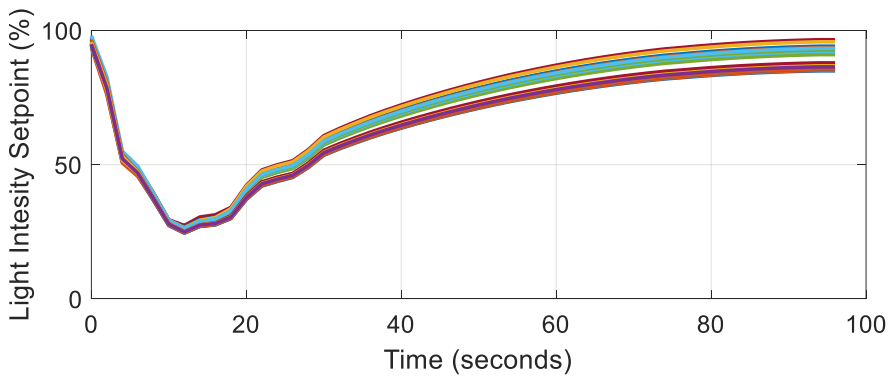


Figure 23. Light intensity values for each outer fuel rod during the normally scaled reactor tilt testing.

4.2.2 Exaggerated-scale Testing

Because the difference in power levels across the core (see Section 4.2.1) was difficult to detect with the naked eye, the tilt test was repeated with alterations made to the light settings of the core. The primary purpose of this test was to highlight for human observers that COMMAND could generate a spatial reactor core profile via different power levels for the LEDs. As such, this test was not intended to be used to draw scientific conclusions.

To highlight the spatial capability, the light intensity was scaled to emphasize the power difference across the core. In the tilt test conducted in Section 4.2.1, the maximum and minimum light intensity outputs were 96.5% and 85%. To better display this power difference across the core, the test was repeated with the lights linearly scaled such that 80% reactor power was equivalent to 0% light intensity, and 100% reactor power was equivalent to 100% light intensity. This resulted in the color mapping that blue light corresponded to 68 kW of reactor power, purple light was 76.5 kW of reactor power, and red light was 85 kW of reactor power. After the light scaling was applied and the control drums had arrived at the final position, the scaled maximum and minimum light intensity outputs were 81% and 24%, producing a difference in light that was clearly discernable to any human observer (Figure 24).

Table 1. Scaling values for highlighting the power tilt across the reactor core.

	Actual Reactor Power	**Light Intensity**
Minimum	80%	0%
Maximum	100%	100%

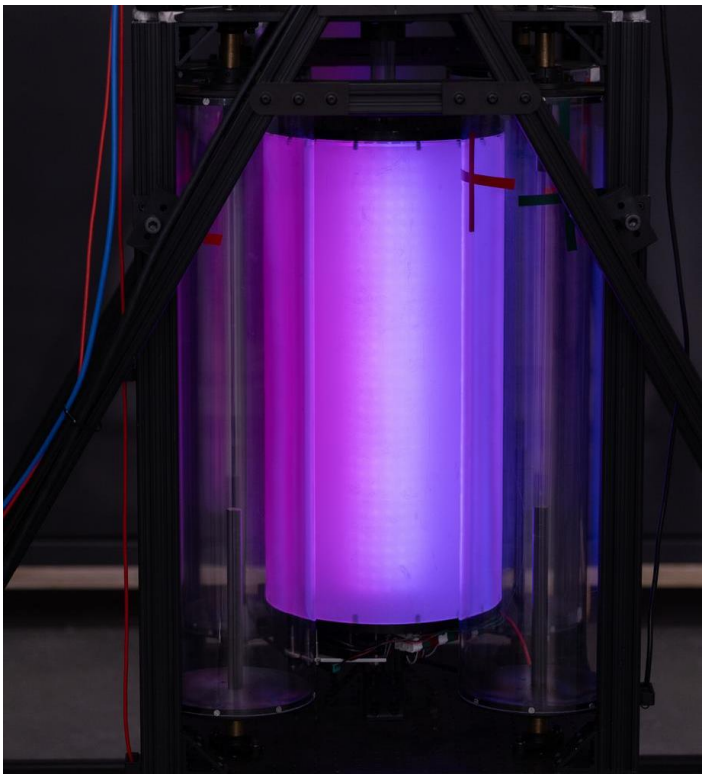


Figure 24. MACS testbed showing the reactor core tilt, shown as a color gradient, as received from COMMAND and RELAP5-3D.

5. CONCLUSIONS AND FUTURE PLANS

To accommodate the fact that advanced reactors are designed to be deployed in remote locations, monitored remotely, operated at lower and/or variable power ratings, compact in size, manufactured in a modular manner, and reliant on novel technologies to enhance operational safety, advanced reactors must ultimately operate in an autonomous or semi-autonomous fashion, thus requiring the development of new control paradigms. This will involve development of new approaches—as well as novel coupling of disparate approaches—to ensure safe operations with less human involvement. Such development will require a demonstration platform for testing and validating the approaches, and the MACS setup is proving to be a capable platform to meet these needs.

In this effort, COMMAND was updated to provide new capabilities in conjunction with the MACS demonstration platform. This included updating the RELAP model and adding MCNP surrogate models to account for spatial reactor kinetics, integrating the gRPC communication protocol to enable signals to be sent between COMMAND and the MACS LabVIEW controller, and coupling and testing these connections in order to successfully demonstrate a few simple control scenarios in which MARVEL was mirrored by MACS as a physical twin.

The next phase of this effort will introduce and demonstrate a more complex control scenario, utilizing MACS and COMMAND. Both COMMAND and MACS are expected to continue expanding and evolving to meet new requirements essential for realizing a new control paradigm, thus fostering advancement toward autonomous operations of advanced reactors.

6. REFERENCES

1. Farber, J., Al Rashdan, A., and Coelho, M. 2023. "Creating a Simulation Platform for Research and Development of Advanced Control Methods." Idaho National Laboratory. INL/RPT-23-75289.
2. Al Rashdan, A., et al. 2022. "Integration of Control Methods and Digital Twins for Advanced Nuclear Reactors." Idaho National Laboratory. INL/RPT-22-69937. https://doi.org/10.2172/1924292.
3. INL. n.d. "Glossary of Terms." Idaho National Laboratory, Digital Innovation Center of Excellence. Accessed 13 October 2023. https://dice.inl.gov/glossary-of-terms/.
4. Crawford, A., and Unruh, T. 2023. "Light as a benign reactivity analog for nuclear modeling validation and control development." Patent Application Serial No. 18/538,864, 13, December 2023.
5. US D.O.E. n.d. "Microreactor Applications Research Validation and Evaluation Project (MARVEL)." U.S. Department of Energy. https://factsheets.inl.gov/FactSheets/20-50248_MARVEL_Fact_Sheet_R15.pdf.
6. INL. 2018. "RELAP5-3D Code Manual, Volumes I-V." Idaho National Laboratory, INL/MIS-15-36723, Rev. 4.4.
7. Werner, C. J., et al. 2017. *MCNP User's Manual Code Version 6.2*. Los Alamos National Laboratory Tech. Rep. LA-UR-17-29981. Los Alamos, NM, USA. October 2017. https://mcnp.lanl.gov/pdf_files/TechReport_2017_LANL_LA-UR-17-29981_WernerArmstrongEtAl.pdf.
8. Lange, T., et al. 2022. "MARVEL Core Design and Neutronic Characteristics." *American Nuclear Society, Winter 2022.* Phoenix, AZ: ANS.
9. Parisi, C. and Arafat, Y. 2022. "MARVEL Thermal-Hydraulics: Normal and Accidental Conditions." *American Nuclear Society, Winter 2022.* Phoenix, AZ: ANS.
10. Idaho National Laboratory. n.d. "RELAP5-3D - Home." Accessed 13 October 2023. https://relap53d.inl.gov/SitePages/Home.aspx
11. Parisi, C. 2023. "RELAP5-3D Thermal Hydraulic Analysis of MARVEL Microreactor – Final Design." Idaho National Laboratory, ECAR-6332, Rev. 0.
12. gRPC. "Introduction to gRPC." Accessed 04 March 2024. https://grpc.io/docs/what-is-grpc/introduction/