ORNL/SPR-2023/2958
Revision 0

Digital Twin Development of FASTR

June | 2023

Vineet Kumar, Wesley C. Williams, Xingang Zhao, William Gurecky
Oak Ridge National Laboratory

IES
Integrated Energy Systems

ORNL/SPR-2023/2958

Integrated Energy Systems (IES) (http://www.ies.inl.gov)

Digital Twin Development of FASTR

Vineet Kumar, Wesley C. Williams, Xingang Zhao, William Gurecky
Oak Ridge National Laboratory

June 2023

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

ABSTRACT

This work package is focused on creating a digital twin (DT) of the Facility to Alleviate Salt Technology Risks (FASTR) experiment. FASTR is a high-profile facility tasked with maturing molten salt applications such as liquid salt energy storage, concentrated solar power facilities, and molten salt cooling systems supporting other advanced reactor technologies, all of which may be incorporated as components within the Integrated Energy Systems (IES) program. As part of creating a digital twin (DT), verification and validation (V&V) has been performed for the physics-based model, and gaps have been identified. Additionally, a preliminary integration of the FASTR model has been completed with the existing physics-based model of the Thermal Energy Distribution System (TEDS)/Microreactor AGile Non-nuclear Experimental Testbed (MAGNET) facility at Idaho National Laboratory (INL). A Functional Mock-up Unit (FMU) of the physics-based system model and a reduced-order model (ROM) were both used to test for a preliminary hardware in the loop operation when the physical loop was offline. Future tasks could include improving the physics-based model to better match the available experimental data, improving the hardware in the loop integration, and improving the integration with the TEDS model.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

DETAIL Dynamic Energy Transport and Integration Laboratory

DMD dynamic mode decomposition

DMDc dynamic mode decomposition with control

DOE US Department of Energy

DT digital twin

FASTR Facility to Alleviate Salt Technology Risks

FMI Functional Mock-up Interface

FMU Functional Mock-up Unit

GUI graphical user interface

IES Integrated Energy Systems

INL Idaho National Laboratory

MAGNET Microreactor AGile Non-nuclear Experimental Testbed

NE Office of Nuclear Energy

NN neural network

ORNL Oak Ridge National Laboratory

PID proportional–integral–derivative

PLC programmatic logic controller

ROM reduced-order model

TEDS Thermal Energy Distribution System

TRANSFORM Transient Simulation Framework of Reconfigurable Models

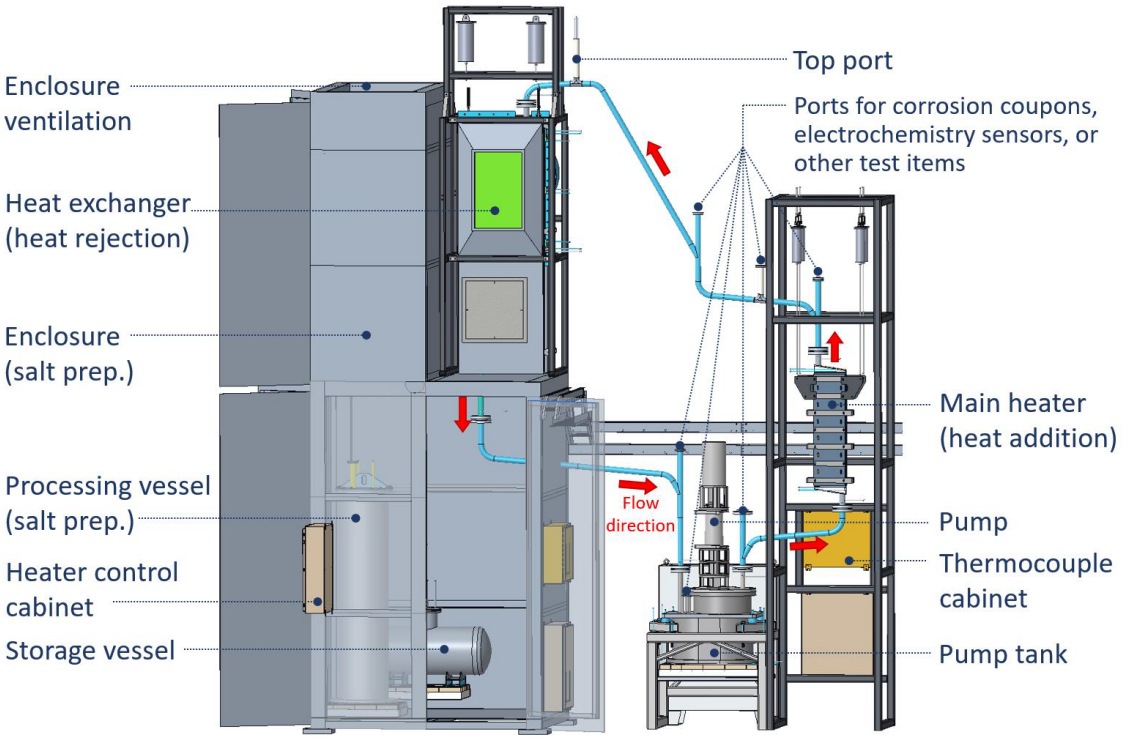
1. DIGITAL TWIN DEVELOPMENT OF FASTR

1.1 BACKGROUND

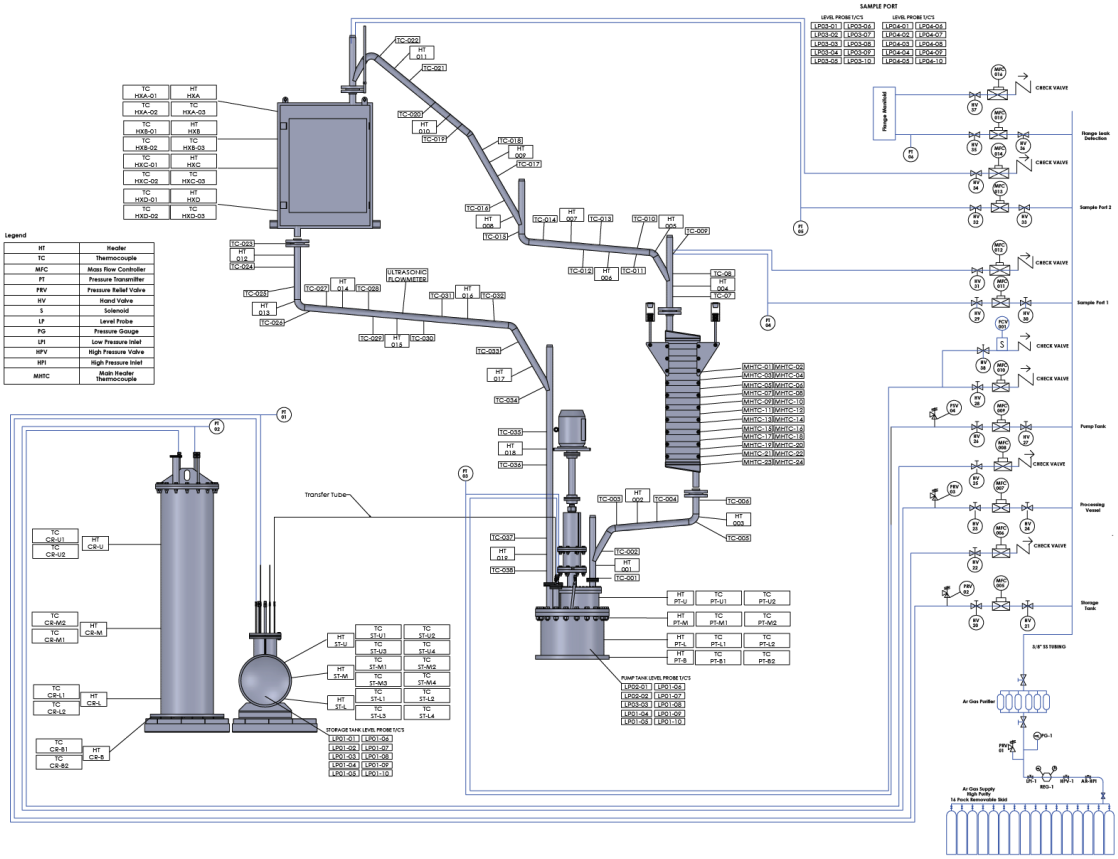
The Facility to Alleviate Salt Technology Risks (FASTR) is a versatile, high-temperature (>600 °C) molten chloride salt loop designed to advance Gen3 concentrated solar power (CSP) salt technology and is funded by the Solar Energy Technologies Office (SETO) of the US Department of Energy (DOE) Office of Energy Efficiency and Renewable Energy (EERE) [1, 3]. FASTR includes a molten chloride salt preparation system and a forced convection test loop with an instrumentation suite consisting of salt level probes, pressure transmitter and gauges, thermocouples, an ultrasonic flowmeter, mass flow meters, and multi-functional voltammetry sensors. The salt preparation system is capable of supplying large batches (~200 kg / 120 L) of clean NaCl-KCl-$MgCl_2$ ternary salt for use in the forced convection loop, which is rated to operate at 725 °C with salt flow rates of ~3–7 kg/s. The salt is a blend of two industrially sourced salts: anhydrous carnalite (AC) from Israel Chemicals Ltd. (ICL) and Silver Peak (SPK) halite from Albemarle Inc. These were chosen to arrive at a low melting point composition [4]. A custom electrochemical sensor developed at Argonne National Laboratory was used to monitor the salt potential in the C-276 lined purification vessel. Additional details regarding instrumentation and the salt purification process details can be found in Robb et al. [4]. Upon completion of the chemical purification stage, the 194.1 kg of salt (assuming a salt density of 1648 kg/m^3) was transferred at 555–565 °C with a total transfer time of ~10.3 hours; it was then allowed to freeze within the storage tank. The composition of the purified salt was analyzed using an inductively coupled plasma (ICP) mass spectrometer and an ICP optical emission spectrometer to assess the success of the purification process. It was estimated that 1.94 wt% water and 0.50 wt% HCl were released from the salt. During initial filling of the loop for pumped operation, it was discovered that there was a deficit of 20 L because of design changes and revisions while the loop was being fabricated. Therefore, an additional 59.6 kg of salt was purified and added to the storage tank, bringing the total salt mass in the system to 253.7 kg [4].

The Integrated Energy Systems (IES) program within the Office of Nuclear Energy (NE) focuses on dynamically integrating thermal and electrical energy needs in response to the power grid and associated industries. A baseload nuclear power plant (NPP) will act as the centerpiece, augmented by other renewable energy sources [5]. The digital twin (DT) development of the FASTR loop is one of the work packages under the system simulation section of the Integrated Energy Systems (IES) program, where the focus is on performing physics-based simulations of systems to capture the physical behaviors of mixed electrical and thermal energy usage and storage systems. A DT refers to an integrated multiphysics, multiscale, probabilistic simulation of the physical asset/twin, and it offers a framework to better quantify design margins, parameter uncertainty, and system performance associated with the physical object.

Creation of a DT requires a physics-based / data-driven model validated with available experimental data and integrated within the hardware of the facility to receive inputs and perform predictive simulations, which can be used by the operator and/or communicated to the hardware controller. Within the scope of this work package, validation of the system model has been performed with limited experimental data, and a workflow has been established to communicate with the facility computer, which can consequently communicate with the facility hardware. Given the export controlled nature of the facility, no outside communication is permitted. Furthermore, only one-way communication with the controller is permitted, such that a Functional Mock-up Unit (FMU) or a reduced-order model (ROM) of the system model receives inputs and generates outputs in response to the received inputs. Future work could entail better hardware integration, as well as continuous improvement of the system model.



(a) Schematic of the FASTR loop.



(b) FASTR intrumentation layout.

Figure 1. FASTR layout [1, 2].

A layout of the FASTR loop is shown in Fig. 1a [3]. The salt preparation is conducted in a ventilated enclosure with transfer lines (not shown in Figure 1a) interconnecting the processing, storage and pump vessels. The pump, which is located outside the enclosure, forces salt to flow in a counterclockwise direction around the loop. The main heater supplies most of the heat to the salt, and an air-cooled heat exchanger removes heat from the salt. Trace heaters wrapped around all the piping prevent the salt from freezing. The following is a brief description of the key component of the loop.

Argon is used as a cover gas and to pneumatically move salts from one vessel to another. The gas supply system includes a source of argon, a gas supply panel, a gas purifier, and mass flow controllers. The storage tank, which is made of C-276 with a total volume of 240 L, is used to store the salt when not in use and can accommodate a pressure of 206.8 kPa at 700 °C. The storage tank is surrounded by a heater blanket consisting of three heater zones positioned vertically and 4 in. of high-temperature insulation. A custom high-temperature cantilevered centrifugal-type molten salt pump is used to circulate salt in the loop. This type of pump does not have seals or bearings in contact with the salt. The bearings are all located in the gas space. The pump produces up to 70 gpm at 29 ft of head with a maximum temperature rating of 725 °C. A variable-frequency drive connected to the programmatic logic controller (PLC) controls the salt flow rate. The pump is inserted into a custom-designed pump tank, which stores the salt transferred from the storage tank when the loop is in operation. To reduce agitation of the salt free surface and entrainment of gases into the salt, the loop return line discharges into a flow distributor made of two stacked C-276 plates welded at the bottom of the tank.

The main heater consists of a flat plate geometry composed of a single 0.75 in. thick Inconel 600 plate measuring 46.25 in long and 13 in. wide. Fourteen holes with a diameter of 0.313 in. are gun drilled at 0.785 in. through the length of the plate and serve as vertical salt flow channels. The design goal is for the heater to provide salt up to 725 °C at the outlet and to possess a peak attainable heat flux to the salt of 1 MW/m^2. Eight IR lamps are currently available, which can provide approximately 103 kW_{th} to the heater plate, expandable to 24 lamps in the future to provide ~310 kW_{th}. For the initial isothermal salt tests, trace heating was applied to the front and back faces of the heater block using heat tapes [2].

An air-cooled crossflow-type heat exchanger is in place to reject heat from the salt. The heat exchanger core consists of two staggered rows of 8 finned vertical C-276 pipes through the salt flows. The heat exchanger core is supported by an insulated box that resides inside an enclosure with doors that can be raised to expose the heat exchanger core to forced air flow. Tubular heaters are also installed between the doors and the heat exchanger core; these expose the heat exchanger core to a blower-operated forced airflow. Auxiliary heaters with a total capacity of 24 kW were installed within the outer enclosure to preheat the enclosure and heat exchanger core. For the initial set of tests, the fan blower was not operated, as the large surface of the finned tubes provides excessive heat rejection, requiring the auxiliary heaters to be deployed. The hot air is vented to a stack located on the roof. The main piping throughout the loop is 2 in. schedule 40 seamless C-276 pipe with custom-designed flanges included on either side of the major components to facilitate swapping of components in the future. Six ports, with nominal diameters of 2.067 in., are located in the piping to allow for the introduction of corrosion coupons, sensors, and other experimental apparatuses. An additional port located at the very top of the loop is dedicated to managing the cover gas volume and pressure as the loop is filled/drained. More information about this port is given in what follows, as it plays an important role in the operation of the loop. Finally, a few additional ports are located in the pump tank.

A schematic of the instrumentation layout throughout the loop is shown in Fig. 1b. Standard N-type

thermocouples are placed externally to measure the temperature. Five Honeywell Midas T-004 gas sensors are distributed around the facility to detect the presence of hazardous gasses such as Cl_2, HCl, and H_2. An ultrasonic flow meter manufactured by Flexim, Inc. is in place to measure the salt flow rate, and it is attached to the cold side of the loop between the heat exchanger and the pump tank. For the initial set of tests, the flow meter was not used because it consistently returned a noisy signal, and future tests are planned to test other flow meters. To monitor the state of the salt—that is, the salt redox potential and concentration of certain species—two multi-functional voltammetry sensors are used, which were developed at Argonne National Laboratory. The salt level is monitored using heated thermocouple arrays located inside the storage tank (only available salt-wetted thermocouples), in the pump tank, in the top port vent, and inside one test port. The salt level can also be deduced from the multi-functional voltammetry sensors. The gas pressure is measured and controlled at several locations, as shown in Fig. 1b. The gas pressure is measured with standard pressure transducers (Honeywell model FP2000), and a set of mass flow controllers (MKS model GE50A) are used to add or remove gas to the space. It must be noted that salt-wetted pressure transducers are not currently available in the FASTR loop; therefore, the liquid salt pressure is arrived at indirectly using the gas pressure. FASTR uses an Allan Bradley-based PLC for data acquisition and control. This type of industrial control system is widely deployed and provides a representative interface that may be encountered in existing and future energy facilities. Validated system models could be useful for the operator in calculating the liquid salt pressure along various sections of the loop.

1.2 MODELICA MODEL DETAILS

The Transient Simulation Framework of Reconfigurable Models (TRANSFORM) is a Modelica-based library developed at Oak Ridge National Laboratory (ORNL) to enable rapid development of dynamic, advanced energy systems with an extensible system modeling tool [6]. TRANSFORM is organized as a series of packages, each of which has a general application. The object-oriented nature of Modelica allows users to view, extend, and/or modify any component or add new models to TRANSFORM. The TRANSFORM code, which is built upon the Modelica libraries, solves a system of time-dependent ordinary differential equations (ODEs) based on a finite-volume–based staggered grid formulation that is applicable for single- and two-phase flows. TRANSFORM can model lumped and multi-dimensional fluid dynamics, heat and mass transfer, control logic and sensors, and simple power systems. The TRANSFORM Modelica library can currently be utilized within the commercial dynamic modeling laboratory software environment, Dymola by Dassault Systèmes. Dymola has a graphical user interface (GUI) that allows for visual drag-and-drop system modeling from the various standard Modelica libraries and from the imported TRANSFORM libraries. Dymola also enables enhanced flexibility in interfacing with other codes implementing the Functional Mock-up Interface (FMI). The Functional Mock-up Interface (FMI) defines a standard interface to an exchangeable package that contains an ODE-based system model of a component, or set of coupled components, called FMUs [7]. Dymola translates the Modelica models into C-code and can compile the FMU into a binary format that fully encapsulates the model into a simulator that can be imported for use in a larger system. The FMU then behaves as a standardized black-box simulator that calculates simulated dynamic outputs driven by user-supplied inputs. These inputs and outputs can be driven and accessed by other codes through the standardized interface defined by the FMI. In this project, the FMU of a working Modelica model is run in Python with inputs fed from the Allen Bradley programmatic logic controller (PLC). The preliminary setup for hardware integration is discussed later in this section.

The system model boundary for the loop excludes the processing vessel and the storage vessel (shown in Fig. 1a). An initial report on the Modelica model, developed by Scott Nelson, and released in June 2022 [8], discusses the major loop components in detail. This model was developed before any of the benchmarking tests, and the isothermal salt tests were conducted because of unforeseen experimental delays. With the release of the initial loop commissioning report [2] earlier this year, further discussions were held with the facility PI, Dr. Kevin Robb. From the discussions, it emerged that the Modelica model required modifications, the primary modification being to change the model to a closed tank system. The previous model used a single tank as a pressurizer, sized to the pump tank. The main loop where the salt was pumped, heated and cooled, was connected to the pressurizer in the previous model using a T-section. However in the FASTR loop, two tanks control the pressure and thereby dictate the flow dynamics: the pump tank and the top port. The gas–pressure differential in the two tanks control both the filling of the loop with salt as well as the pumped operations. From a modeling perspective, this essentially requires a closed gas volume that can pressurize a closed liquid volume—that is, a two-phase gas–liquid pressurizer currently available only for steam–water flows in the TRANSFORM library. However, it was found that an open-source Modelica library, Thermopower [9, 10], has a gas–water accumulator that could be utilized for modeling the closed tanks in the FASTR loop. The accumulator model solves for the mass conservation and energy conservation equations for both the gas and liquid phases. The liquid pressure is calculated using Stevino's Law assuming static equilibrium with the gas phase at any given time, with the ideal gas equation of state used for the gas phase. There are two liquid and two gas ports: one inlet and one outlet port for each phase. For the initial set of tests, the report [2] does not specify the gas flow rates entering/exiting the two tanks. Therefore, the gas flow rates were arrived at indirectly by controlling for the liquid level or gas pressure using standard proportional–integral–derivative (PID) controllers. This is elaborated upon further in the model discussion. Finally, the model also accounts for heat transfer between the phases (apart from gas phase expansion work) based on a user-supplied heat transfer coefficient. In the current models, a default value is used for the gas–liquid heat transfer coefficient, but it could be tuned in the future based on the thermocouple array in the pump tank that is used to measure the salt level.

As part of the commissioning of the loop, the pump manufacturer ran a series of tests on the pump using water in an open pit, at room temperature and pressure. These tests were used to develop pump curves at four speeds: 500 rpm, 1050 rpm, 1200 rpm, and 1800 (100%) rpm [2]. The pump was then delivered to be installed in the FASTR loop and was tested using a mini-loop, which started at the pump discharge, raised vertically, passed over a pump motor, and then returned to the pump tank inlet. Instrumentation included two flow meters and a pressure transducer. ORNL data at various pump speeds provided independent verification for the manufacturer-reported pump curves, and data taken at 749 rpm were extrapolated to other pump speeds and are used in the current Modelica model. In the summer of 2022, forced-flow tests were conducted with the pump installed in the pump tank using water, with the primary goal being to confirm shaft seal operation at various pump speeds. The pump speeds and pressures at the pump tank and top port were recorded for different pump speeds. Testing was conducted at room temperature, and the liquid level in the top port was maintained to within ±0.8 in. The gas flow rate was adjusted to keep the liquid level constant during pumped operation and thereby the static pressure head constant. This allowed for calculating the frictional pressure drop on the return side of the loop in the following manner using Bernoulli's equation (Eq. 2) applied between the stagnant gas space in the pump tank and the top port:

$$\left(\frac{\mathbf{P}}{\rho.g} + Z\right)_{in} = \left(\frac{\mathbf{P}}{\rho.g} + Z\right)_{out} - h_{pump} + h_{flowloss}, \tag{1}$$

where P is the measured (stagnant) gas pressure, ρ is the water density, Z is the relative height, h_{pump} is the head added by the pump, and $h_{flowloss}$ is the total flow loss, which includes frictional and minor flow losses. In the return side of the loop, the flow does not pass through the pump. Therefore, the total head flow loss through the return side of the loop was determined by using the static head determined during the nonflowing operations. A similar procedure was followed to perform a one-to-one comparison with the experimental data. A more detailed analysis looking into the total flow losses in both the supply and return side using the model predictions will be conducted in the future.

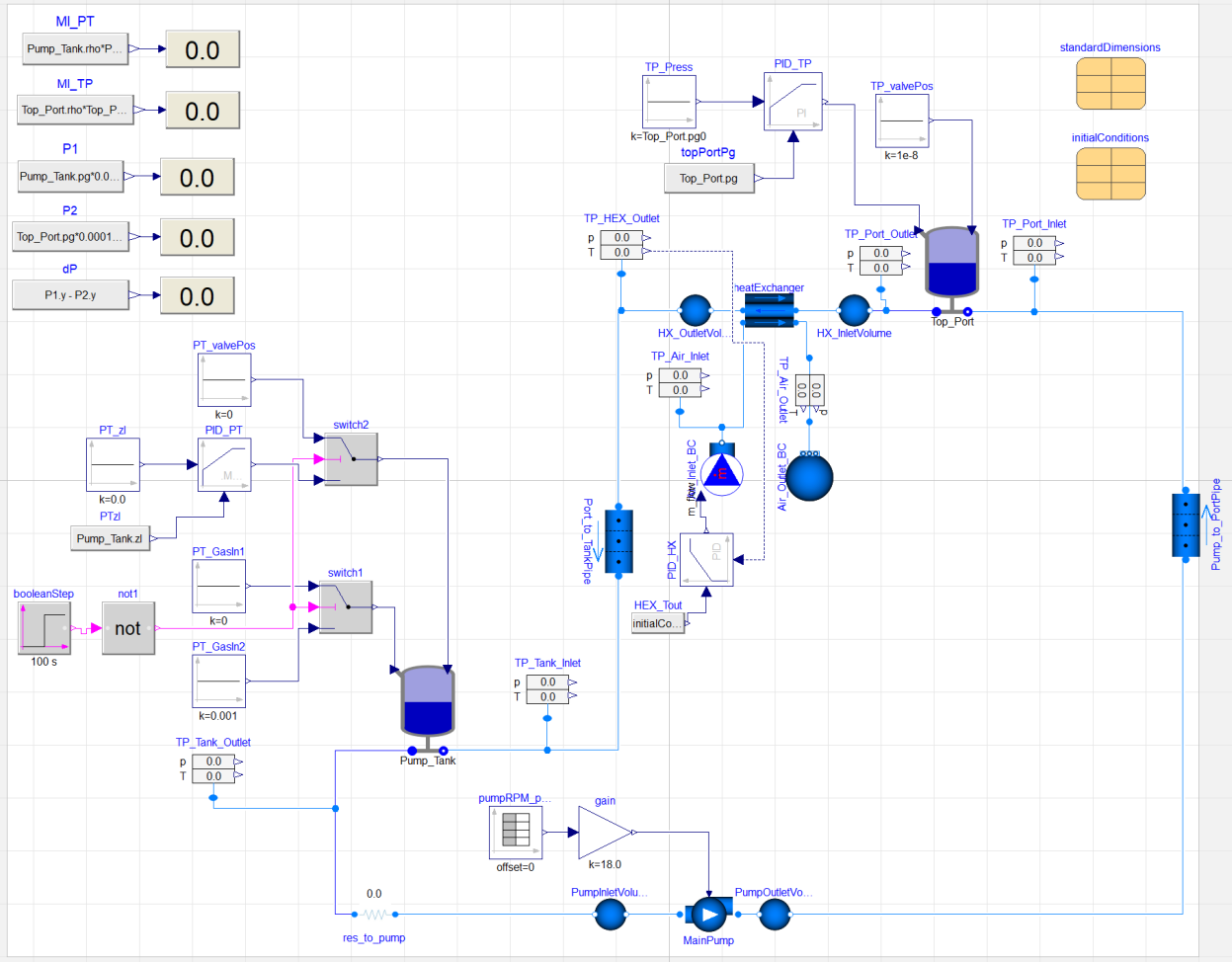


Figure 2. Simplified Modelica model to test pressure control in the top port and level control in the pump tank with water.

The first Modelica model which was developed as a closed tank system using the Thermopower library–based accumulator models for the pump tank, and the top port is shown in Fig. 2. In this model, the gas pressure in the top port is supplied as a fixed input to a PID controller that regulates the gas inflow to the top port. The gas outflow is determined by a user-supplied nominal gas flow rate in the accumulator model by a valve operating in a choked condition; the valve coefficient is determined by the working point at full opening (a coefficient of 1.0). The gas outflow rate is calculated in the following manner:

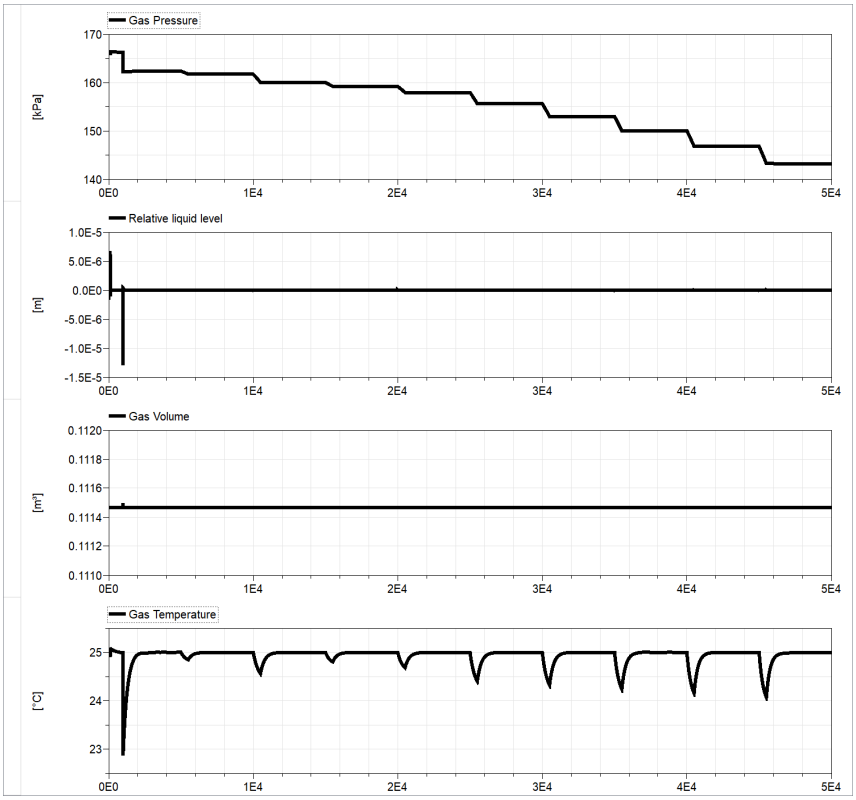
$$w_{g,out} = -\textbf{ValveCoefficient}.\frac{w_{g0}}{p_{g0}\sqrt{T_{g0}}}.\left(p_g\sqrt{T_g}\right), \tag{2}$$

where **ValveCoefficient** is the valve coefficient, w_{g0} is the nominal gas flow rate, p_{g0} is the nominal gas pressure, T_{g0} is the nominal gas temperature, p_g is the instantaneous gas pressure, and T_g is the instantaneous gas temperature.

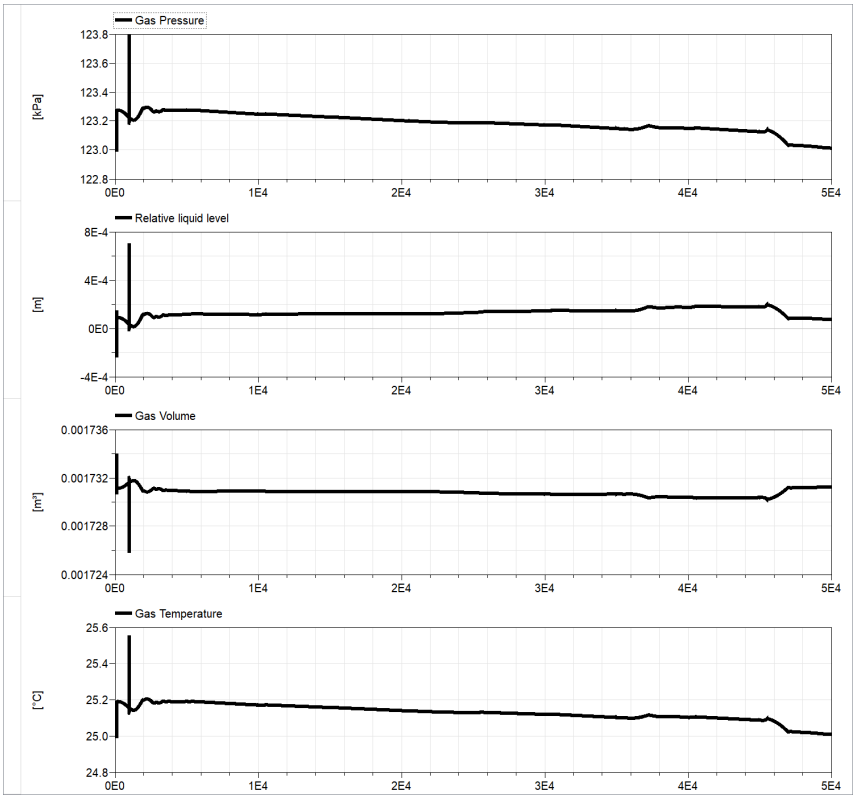
The pump tank has a PID that controls the gas outflow rate by varying the valve coefficient with a setpoint value of 0.0 m (relative) tank level. It must be noted that the absolute water levels in both the tanks are unknown and, therefore, are estimated to be at an initial vale of 0.2 m. This initial value can be adjusted to ensure that the volume of the liquid matches that used in the experiment. The gas inflow rate is initially set to zero and then set to a constant valve at 100s via a Boolean switch. The pump is turned on at 1000 s and then maintained at a given speed for ~4500 s before linearly changing the speed over 500 s. Only one upstream and one downstream pipe were modeled in this simple model. The heat exchanger blower speed is controlled such that the temperature of water exiting the heat exchanger is close to room temperature. It must be noted that the pump adds energy to the system, which increases the system temperature in the absence of heat rejection, necessitating the air blower's operation. Further discussion on heat loss is covered in the next model. The dynamics of this model are shown in Fig. 3. The predictions for the pump tank are shown in Fig. 3a, and predictions for the tank port are shown in Fig. 3b. The pump tank gas pressure drops as a consequence of the liquid pressure dropping as the pump speed is increased to accommodate the required pump head. It must be noted that the TRANSFORM pump model does not account for pump suction, but the dynamics of the model force the pump to draw down pressure. Therefore, the dynamics of this system are a complex interplay between the two tanks. The spikes in the pump tank and top port results occur at 100 s when the outlet gas port in the pump tank is switched to being controlled by the PID from a constant value of zero and are, therefore, purely numerical. The mean gas pressure predictions in the pump tank and the top port for various pump speeds are given below in Fig. 4.

Although the model captures the trends correctly, the actual levels are off. The main reason for this discrepancy is that the actual piping has not been modeled, so the flow losses are under-predicted. Moreover, the length of the simplified equivalent upstream/downstream pipe is less than what is available from the component drawings, under-predicting the static head. This model was not developed further primarily because of convergence issues. Therefore, an alternative model was developed wherein the pump tank is pressure-controlled and the top tank is level-controlled.

The pump tank pressure controlled Modelica model is shown in Fig. 5. The control logic is flipped between the pump tank and the top port. This model also allowed for the full loop to be simulated. This includes the complete piping, which has been simplified between sections—that is, between major components such as the pump to the heater. The equivalent pipe sections capture the pipe lengths and pipe orientations and include minor loss coefficients to account for pipe bends/elbows within the pipe sections. The heat exchanger is modeled as a shell and tube model, with the hot side modeled as two staggered rows of 8 tubes and the cold side shell modeled as flow across tube banks using the available Grimison correlation in the TRANSFORM library. The tube surface area accounts for the total finned area, which was independently hand-calculated. The blower flow rate is kept to a minimum value and is not PID controlled with a setpoint temperature like the simplified model. In reality, a natural circulation flow rate would have to be specified to capture the heat loss. The heater is assumed to be a parallel channel block with a double wall: a first layer of C-276 and a second layer of stainless steel, which is a proxy for the heat tape. The



(a) Pump tank predictions for gas pressure, relative liquid tank level, gas volume, and gas temperature.



(b) Top port predictions for gas pressure, relative liquid tank level, gas volume, and gas temperature.

Figure 3. Pump tank and top port predictions for the simplified model with increasing pump speeds

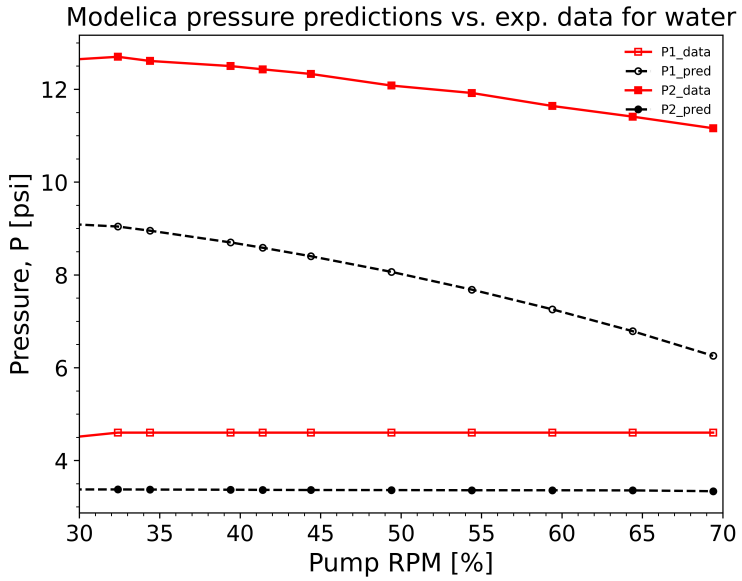


Figure 4. Modelica and experimental gas pressure comparisons for different pump speeds.

inner heater wall would be replaced by Inconel 600 in the future, and accurate equivalent properties of the heat tape would be used for the outer wall. All other pipe sections model a pipe wall made of C-276 with a thickness corresponding to schedule 40 pipe and also include a fiber glass insulation. The thickness of the fiber glass insulation and the external heat transfer coefficient would have to be optimized using the heat loss tests that were performed in the facility at different temperatures using the cover gas [2]. The heat loss tests showed that the heat exchanger dominated the trace heating requirements; and for the salt flowing operations, tubular heaters located in the inlet and outlet plenums of the heat exchanger are required to ensure that the salt temperature does not fall below the freezing margin. In the current model, a default thickness is used for the fiberglass insulation for all the pipe sections.

Predictions of some of the key variables in the pump tank and the top port are shown for different pump speeds in Fig. 6. When comparing with the previous model (see Fig. 3), it can be observed that the gas pressure in the top port increases as the pump pressurization increases with the gas pressure in the pump tank held constant. Although this is different from the manner in which the tests were conducted, this control strategy was necessary to achieve convergence with the full loop modeled. Additionally, the gas temperature increases because the pump adds energy and there is insufficient heat rejection in the current model to keep the loop temperatures fairly constant. This requires fine-tuning the heat losses once the complete raw dataset is available so as to compare the thermocouple measurements with the model predictions. The experimental and model predictions for the gas pressures in the two tanks are tabulated in Tables 1 and 2, for different pump speeds. The results are also plotted in Fig. 7. The pressure difference between the two tanks is consistently lower but perform better than the simplified model. Any modifications to the loop not captured in the drawings would need to be captured in the model to improve the static pressure drop prediction when the pump is turned off. Doing so would improve the gas pressure difference predictions between the two tanks.

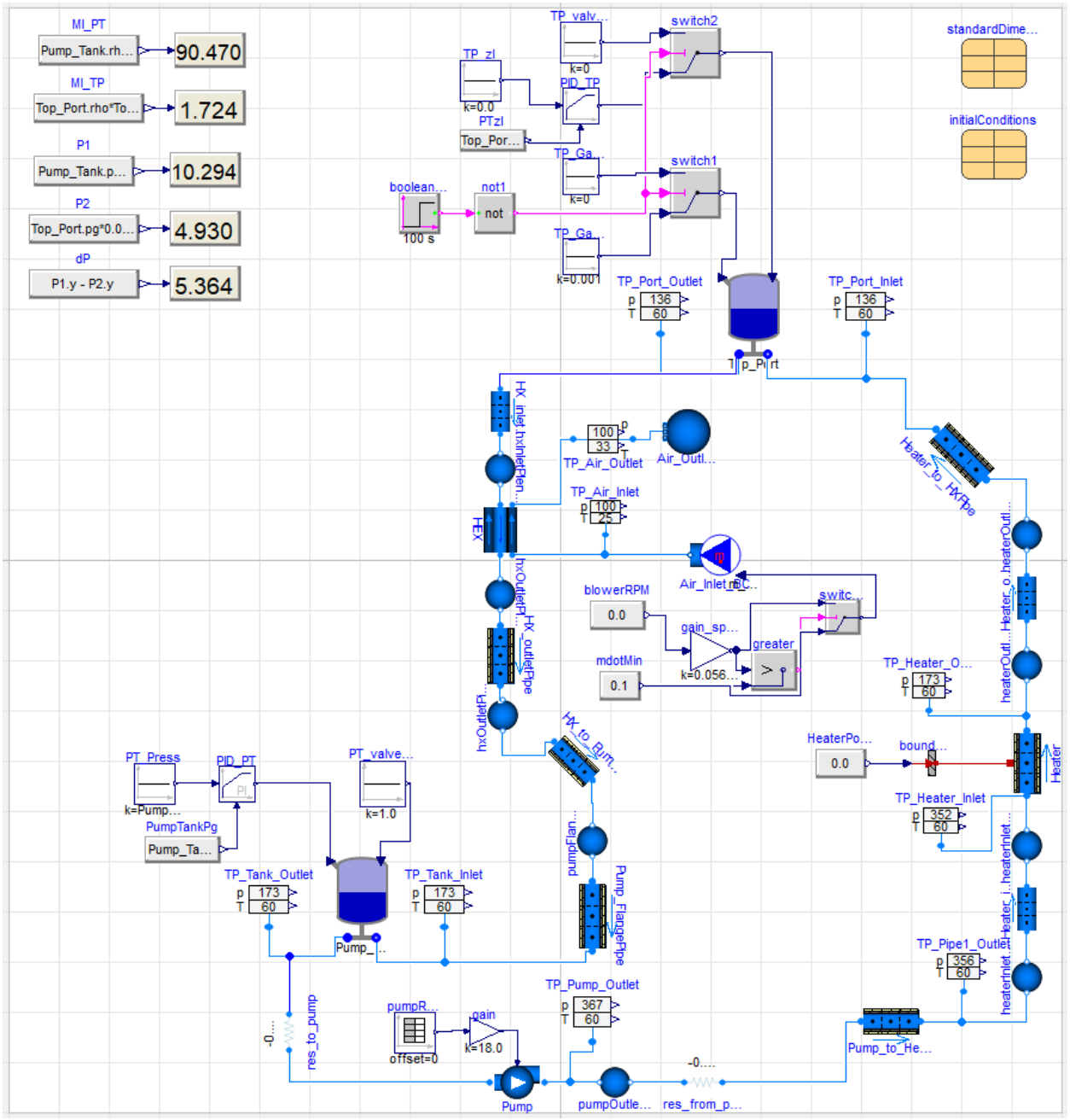
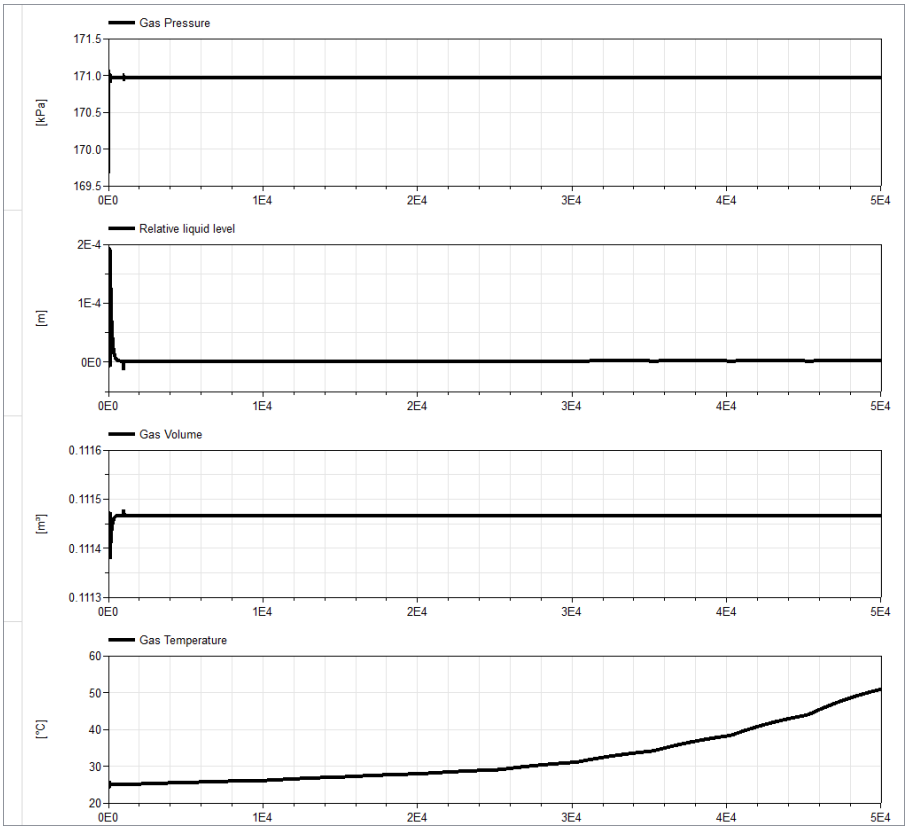
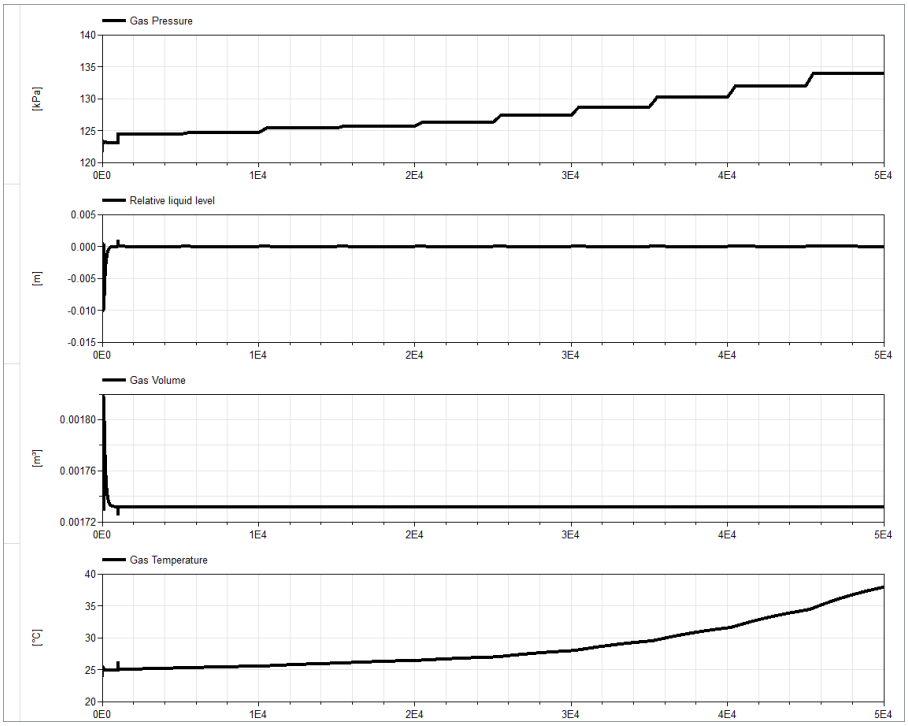


Figure 5. Modelica model to test pressure control in the pump port and level control in the top port with water.



(a) Pump tank predictions for gas pressure, relative liquid tank level, gas volume, and gas temperature.



(b) Top port predictions for gas pressure, relative liquid tank level, gas volume, and gas temperature.

Figure 6. Pump tank and top port predictions for the full model with increasing pump speeds

Table 1. Full-loop water test data [2]

Pump Speed (%)	Pump Speed (rpm)	Top Port P_g (psig)	Pump Tank P_g (psig)	Gas Pressure dP (psi)
0.0	0.0	3.45	12.01	8.56
32.4	583.4	4.60	12.70	8.10
34.4	619.4	4.60	12.61	8.01
39.4	709.5	4.60	12.50	7.89
41.4	745.5	4.60	12.43	7.82
44.4	799.5	4.60	12.33	7.73
49.4	889.6	4.60	12.08	7.48
54.4	979.7	4.60	11.92	7.32
59.4	1069.8	4.60	11.64	7.04
64.4	1159.8	4.60	11.41	6.81
69.4	1249.9	4.60	11.16	6.56

Table 2. Full-loop water test predictions

Pump Speed (%)	Pump Speed (rpm)	Top Port P_g (psig)	Pump Tank P_g (psig)	Gas Pressure dP (psi)
0.0	0.0	3.36	10.29	6.94
32.4	583.2	3.54	10.29	6.75
34.4	619.2	3.58	10.29	6.72
39.4	709.2	3.68	10.29	6.61
41.4	745.2	3.73	10.29	6.56
44.4	799.2	3.81	10.29	6.48
49.4	889.2	3.97	10.29	6.32
54.4	979.2	4.16	10.29	6.13
59.4	1069.2	4.38	10.29	5.91
64.4	1159.2	4.64	10.29	5.66
69.4	1249.2	4.93	10.29	5.36

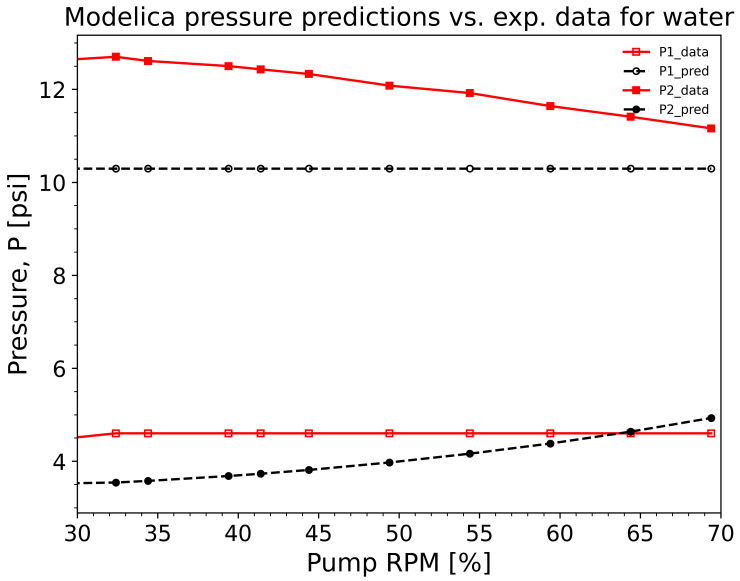


Figure 7. Modelica and experimental gas pressure comparisons for different pump speeds.

The model to capture the salt flowing operations is based on the full model for water (Fig. 5). The ternary salt properties used in the model were taken from the TRANSFORM library for a ternary salt of sodium chloride (NaCl), potassium chloride (KCl), and magnesium chloride ($MgCl_2$) with a mass composition of 30%, 20%, and 50 % respectively. This would have to be tuned in the future to accurately capture the exact composition of the purified salt mixture that is used in the FASTR loop, which contains about 19% of NaCl, 38% of KCl, and 43% of $MgCl_2$ by weight [4]. For the salt filling operations, the argon cover gas is used to pneumatically raise the salt simultaneously in both the piping legs with the displaced gas vented out of the top port. Currently, the capability to model the salt filing operations is not available and would have to be developed. The thermocouple response showed homogenized temperatures after ~ 15.5 hours, after which the pump operations began [2]. A couple of notable changes from the full model for water for the salt model are as follows: the shell and tube heat exchanger was replaced by a pipe with 16 parallel channels, and all insulation was removed. This was because the model is not conserving energy when the pump is turned off. Therefore, all the temperatures were maintained at 565 °C until the pump is turned on, after which the temperatures rise as expected as the pump adds energy. A future task will focus on remediating energy conservation issues for the model when the pump is turned off.

The pressure in the pump tank and the pump speed are the two inputs to the model. The facility operation for the salt flowing operations is as follows. The gas pressure in the pump tank is dropped as the pump speed is increased by opening/closing the valves in the pump tank. The liquid level in the top port increases, compressing the gas space when the pump speed is increased, which is mitigated by controlling the gas flow in the top port to keep the top port liquid level constant. Figure 8 shows the Modelica model predictions for the isothermal test that was conducted at 565 °C. The top port gas pressure is dictated by the gas pressure in the pump tank, which is communicated via the flowing salt. However, the drop in the gas pressure in the top port is about half of the imposed pressure drop in the pump tank. With an improved control and correcting for the static pressure drop mismatch, the top port gas pressure prediction should

match the data fairly well. The salt mass flow rate prediction is also shown and follows from the imposed pump speed. This should be useful for the operator in the absence of a flowmeter. The pump outlet pressure prediction is also shown. At the highest tested pump speed, the outlet pressure prediction exceeds the rated facility pressure, and therefore would have to be examined further. Finally, the complete Modelica model being developed to capture the salt flowing operations is shown in Fig. 9. This includes trace heating for the individual pipe sections. However, the problem of energy conservation when the pump is not in operation will have to be resolved in the complete salt model.

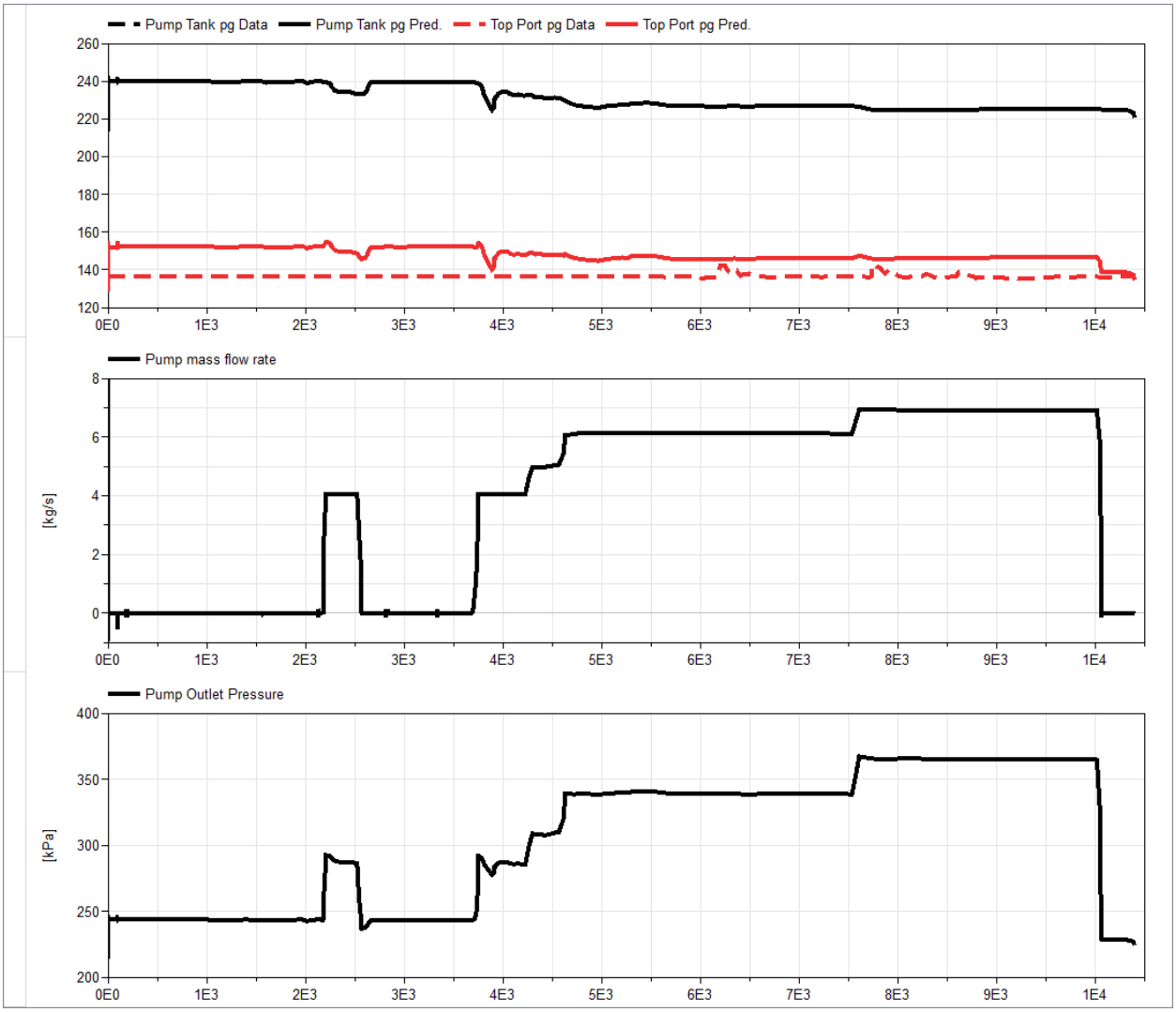


Figure 8. Modelica model predictions for the isothermal test at 565 °C with salt.

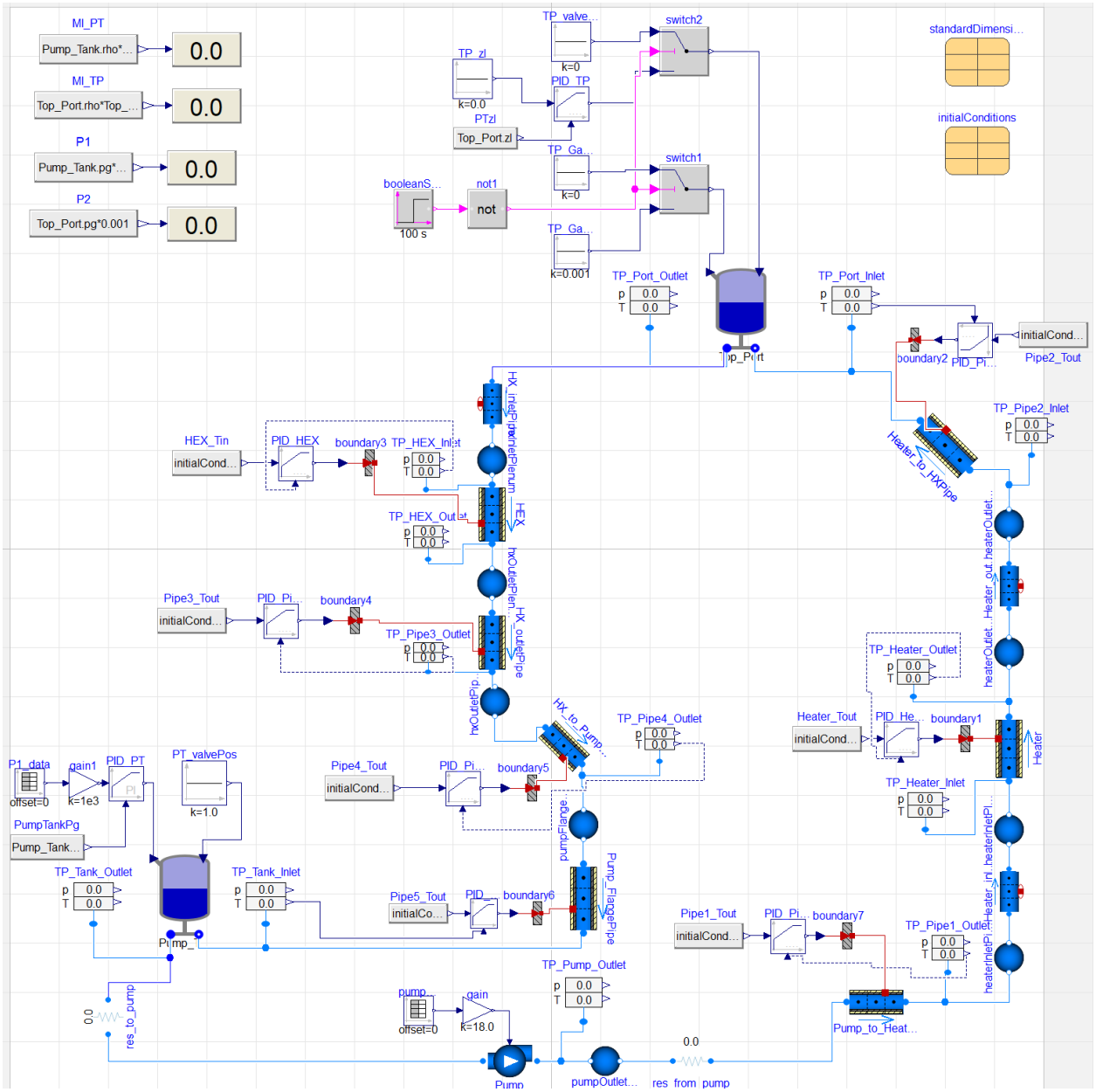


Figure 9. Modelica model being developed with trace heating and heat loss for flowing operations with salt.

Using the Modelica model for the salt flowing operation, an FMU was generated so that it could be run independently of Dymola and ported to a computer that can interface with the industrial Allen Bradley PLC hardware ([3]) used in the FASTR loop. Figure 11 shows the Modelica model used to generate the FMU with the two inputs and the various outputs. A schematic of the current setup is shown in Fig. 10. A separate system is used to mount the FMU or DT and has an Ethernet connection to the facility system that interfaces with the FASTR PLC. By allowing for all signals communicated from/to the facility computer to be made available to the DT system, a one-way data transfer has been established. Key to communicating with the Allen Bradley Logix Diver is the Python library 'pycomm3' ([11]). The DT built using this library makes it possible to perform real-time modeling based on input signals, as well as to provide real-time output. It must be noted the facility computer uses Allen Bradley's Studio 5000 Logix Designer software to communicate with the PLC. Control tags defined in the Studio 5000 software can be accessed using the 'pycomm3' library. A sample Python script using 'fmpy' ([12]) for running the FMU and 'pycomm3' to obtain real-time inputs is provided in Appendix A (2.3.5).

However, limitations to this methodology were discovered. One is the computational capacity of the PLC hardware, which would severely prevent scalable DT solutions from the full fidelity needed for this application. Secondly, the interaction of the PLC with the DT is using the serial time-based communication protocols that are standard to the I&C community, thereby treating the DT the same as any of the instruments or control signals in its list of input/output (I/O) operations. For this reason, a better approach would be to mount and process the DT as its own edge computing device. Future work could potentially look at developing the hardware architecture for full DT implementation. The hardware system could use an edge computing device (NVIDIA Jetson or similar) for running containerized compiled Modelica models/FMUs that can then be interfaced with the I/O. EtherCAT is a proposed communication protocol for connecting the model in real time to I&C hardware. The FASTR loop has existing I&C and industrial system (Allen Bradley) that can be interfaced with the NVIDIA hardware to create hardware-in-the-loop (HIL) systems.

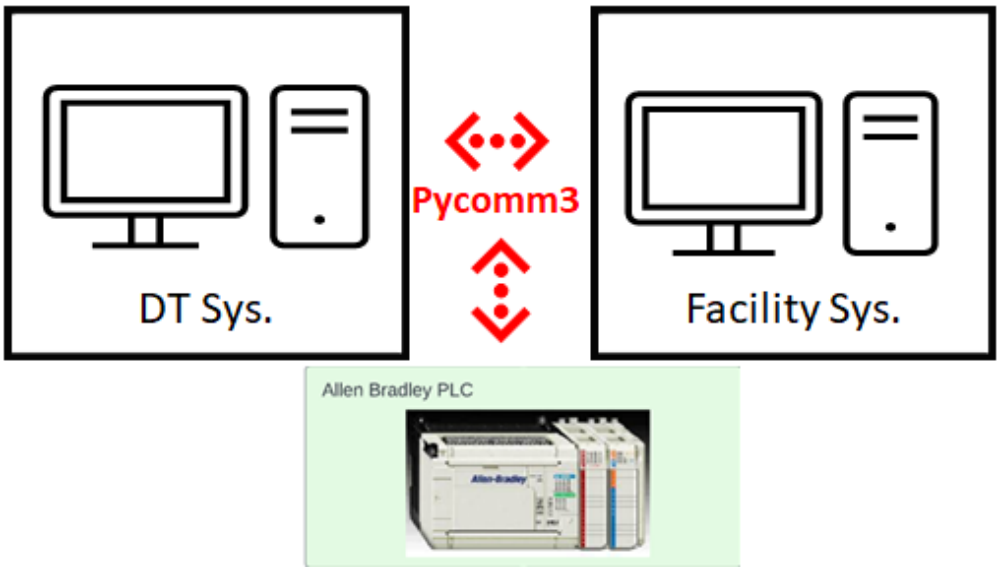


Figure 10. Modelica model to test pressure control in the pump port and level control in the top port with water.

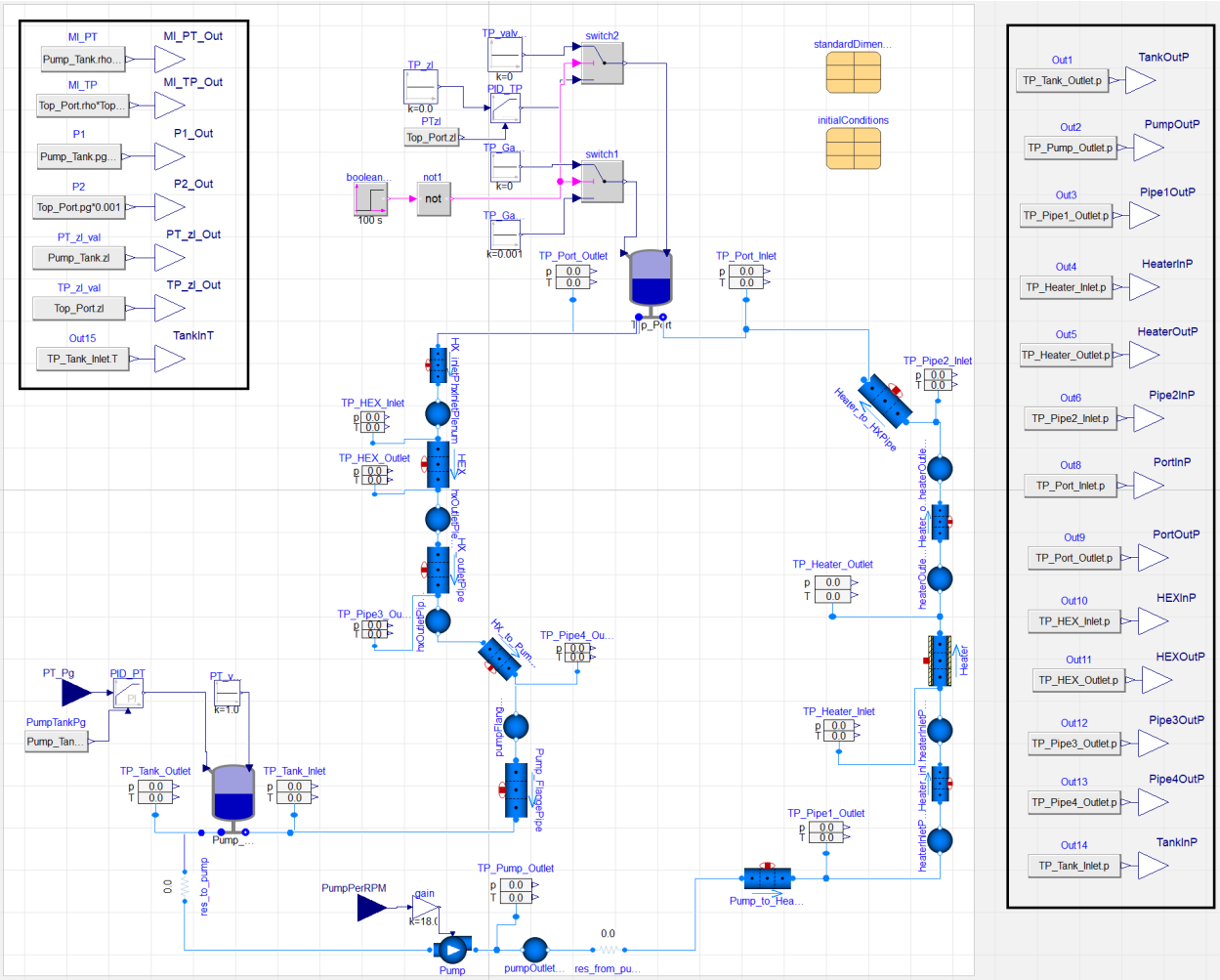


Figure 11. Modelica model predictions for the isothermal test at 565 °C with salt.

1.3 REDUCED ORDER MODEL DEVELOPMENT USING DMD

Dynamic mode decomposition (DMD) is a model reduction algorithm that is data-driven and equation-free. Originally introduced by Schmid [13], DMD has emerged as a powerful reduced-order model (ROM) technique for analyzing the dynamics of high-dimensional, complex systems. The algorithm reconstructs the underlying dynamics of the system from snapshot measurements (e.g., experimental data, numerical simulations) alone and does not make any prior assumptions about the system. It takes in time series data and computes a set of modes, each of which is associated with a complex eigenvalue. By using DMD, a nonlinear system can be described by a superposition of modes whose dynamics are governed by its eigenvalues. DMD is closely associated with the Koopman operator theory [14].

In this work, DMD was the preferred method to create a ROM of the Modelica model for the FASTR system (Fig. 11). The open-source Python library PyDMD[1] [15] was used, which implements the DMD technique and many of its variants. Simulated data of 22 time series signals were generated in Dymola to

[1]PyDMD repository: https://github.com/PyDMD/PyDMD.

provide DMD with the temporal snapshots (close to 10,000 snapshots) for it to compute the decomposition on the data. Figure 12 compares the original (Dymola) data with the ROM solutions reconstructed with DMD for all 22 signal variables. Although the DMD ROM can match the general trends and the order of magnitude of Dymola data, it fails to capture value changes and local transient patterns found in most signals. Unsurprisingly, this finding stresses the lack of spatial resolution in the snapshots, leading to temporal modes only with DMD. In fact, the application of DMD has seen substantial success in fields such as fluid dynamics [16], which have been historically difficult to analyze due to the enormous number of spatial states required for simulation.

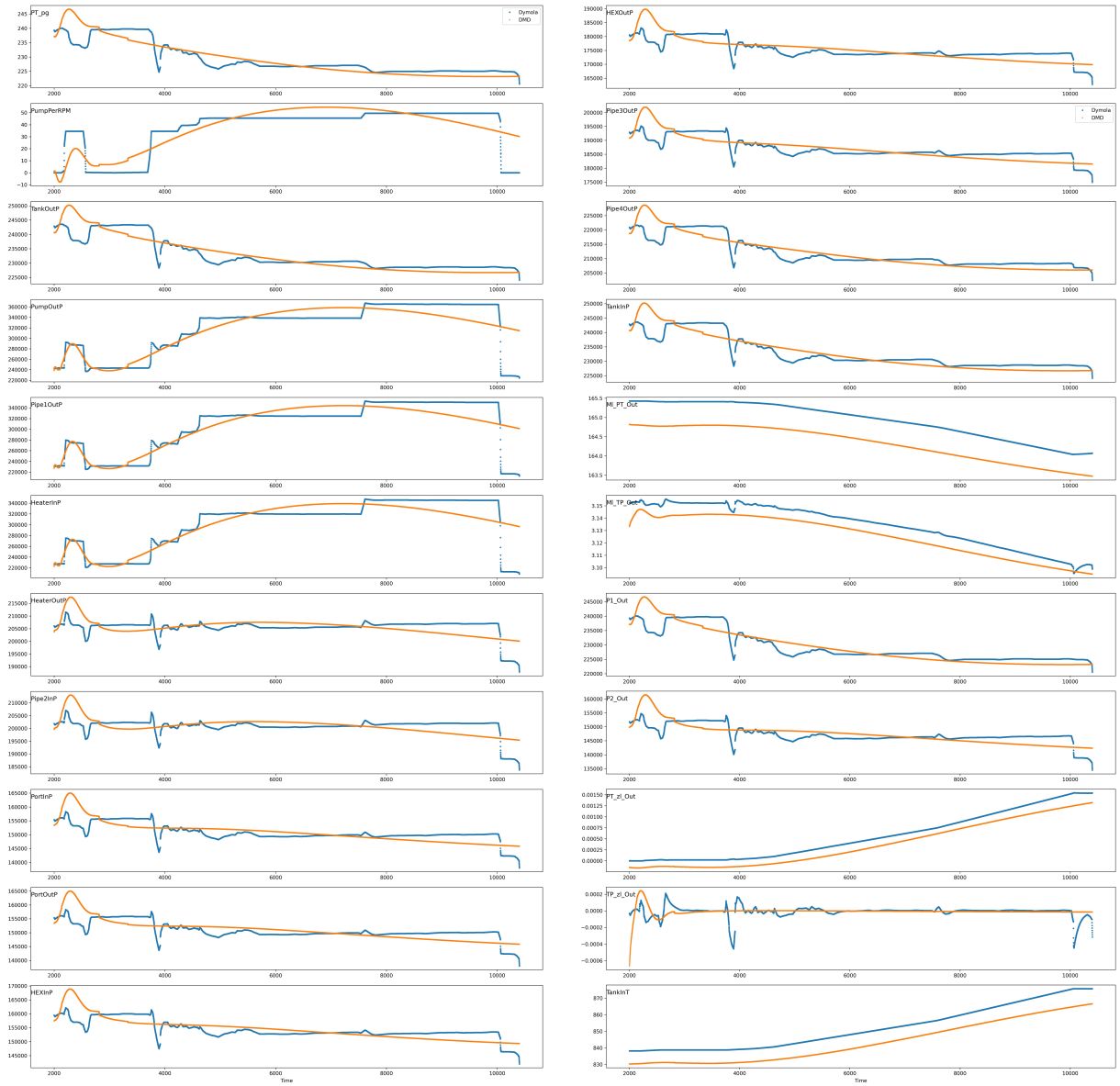
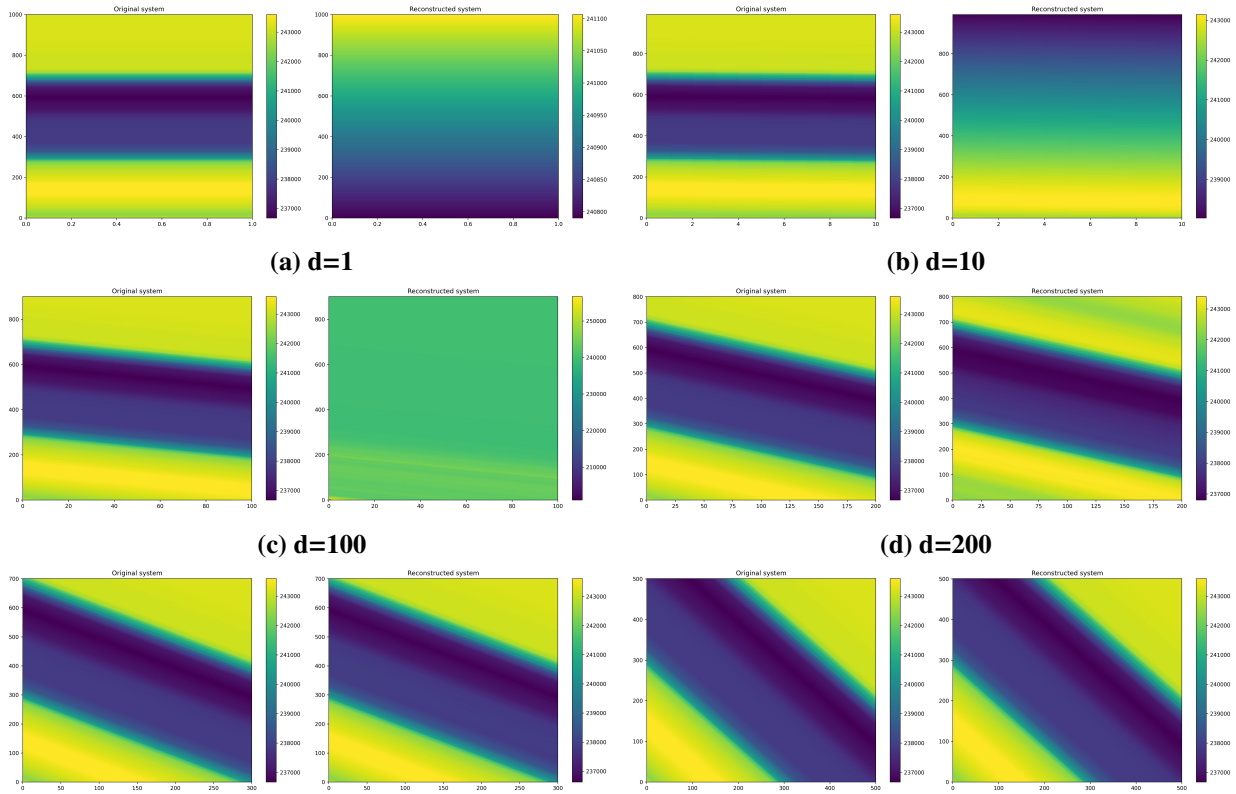


Figure 12. Dymola vs. DMD ROM solutions for the Modelica model of FASTR.

One solution to the spatial sparsity of the system is the use of Hankel DMD [17], which has been applied to

univariate time series forecasting problems [18]. The Hankel DMD approach involves a time delay embedding step that converts time series data into a multidimensional series of lagged vectors that produces a Hankel matrix. DMD is then performed on the Hankel matrix, and the average of the anti-diagonal elements forms the reconstructed time series. In this work, the Hankel DMD module in PyDMD was explored on the *TankOutP* time series. Figure 13 shows the original vs. reconstructed system with 1000 snapshots (time-steps) and varying order d for spatial dimension of the snapshots. Hankel DMD provided the identification of the meaningful structures and the complete reconstruction of the system using only the collected snapshots for high spatial dimensions (i.e., the resulting Hankel matrix is close to a square matrix).



(a) d=1 **(b) d=10**

(c) d=100 **(d) d=200**

(e) d=300 **(f) d=500**

Figure 13. Original vs. reconstructed system using Hankel DMD with 1000 snapshots and varying spatial dimension *d*.

Another shortcoming of the vanilla DMD ROM is that it does not consider the control aspect of the underlying system. The effect of control and the presence of controlled/forced variables in the 22 signals can be incorporated by a DMD variant, dynamic mode decomposition with control (DMDc) [19]. The DMDc method helps discover the underlying dynamics without the confounding effect of external control and also quantifies the effect of control inputs on the measurements of the system. In this work, the DMDc module in PyDMD was explored on the *TankOutP* time series, as shown in Fig. 14. The DMDc model included the control input *PT_pg* and made much better predictions than the vanilla DMD model. PyDMD currently does not support stacking lagged time series in DMDc (i.e., to Hankelize the data matrices for increased spatial dimension), which will be useful to improve reconstruction capability of the DMD ROM.

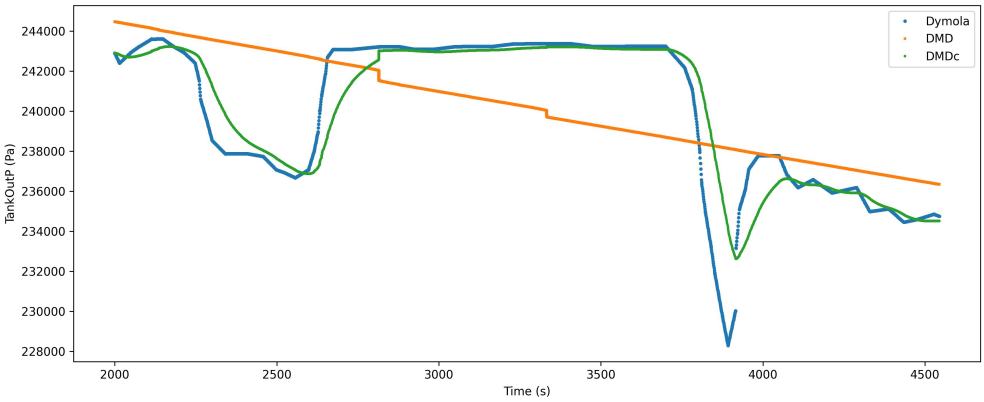


Figure 14. Reconstructed *TankOutP* time series using DMD and DMDc.

Recommended future research directions around DMD for ROM development of systems like FASTR also include: (1) showing the predictive capability of such ROMs on future unseen data, analogous to testing in machine learning, and (2) investigating physics-informed DMD variants to generate physically sound and explainable ROMs.

1.4 INTEGRATION OF ONNX MODELS INTO MODELICA

A novel capability was developed to allow for the import and use of pre-trained neural network (NN) models within a Modelica model. Within the larger system-level FMU, the inputs to the neural network model are supplied by the Modelica model, and outputs from the neural network can be fed back into the Modelica model. In practice, the NN behaves as a standard Modelica function block with inputs and outputs. This new capability enables in-memory data exchanges to be performed between neural networks and Modelica models—all within the Modelica ecosystem. This approach seamlessly integrates with the Dymola and OpenModelica GUI interfaces for constructing connections to and from the NN within a Modelica model. The resulting coupling requires no external driver scripting or slow file-based exchange between the NN and Modelica model. This allows for flexible black-box modeling of certain physics and closure relationships within a Modelica model through the use of data-driven NNs. Furthermore, this tight integration of NN into the Modelica ecosystem allows for bundling the NN model within an FMU for easy export and model sharing.

This capability is made possible by leveraging the ONNX runtime [20] and ONNX file format. ONNX provides a file format that allows one to serialize NN models from PyTorch, Tensorflow, or other machine learning frameworks into a portable binary representation. This ONNX-serialized network can then be saved to disk and shared. At inference time, the ONNX-serialized NN model can be read from disk and interacted with through the ONNX runtime library. The ONNX runtime provides a C++ API that was specifically utilized in this work to build the bridge between the Modelica model and the ONNX-serialized neural network model. Figure 15 gives a high-level overview of the software implementation of this Modelica–ONNX–model bridge. The Modelica language specification provides both the External Library and External Object capabilities to call external, user-written C functions [21]. This capability is exceptionally powerful, as a user needs only to develop a C interface to their existing (potentially highly complex) codes and models that can then be called from a Modelica model. In the context of the present work, this allowed for the creation of a thin C adaptor, which wrapped the C++ based code that interacted

with the ONNX-serialized NN model such that it conformed to the C interface specified by the Modelica language standard.

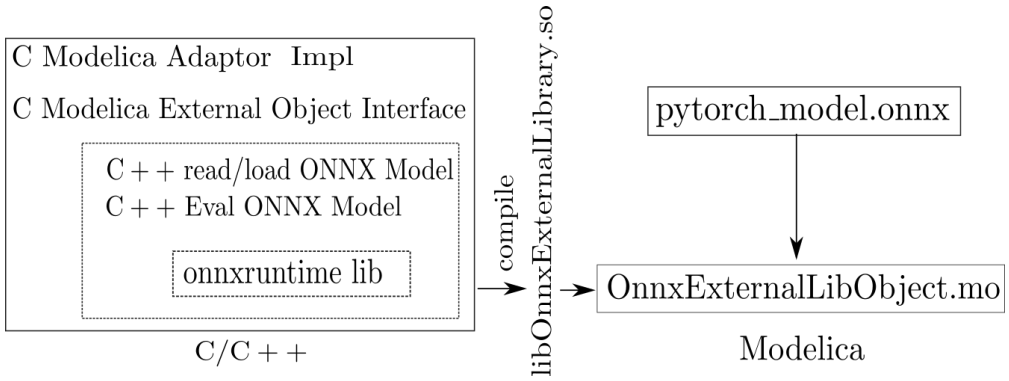


Figure 15. Outline of Modelica–ONNX bridge components and implementation.

This capability will enable data-driven models—specifically neural networks—to be effectively integrated with existing Modelica models of large systems. For instance, in the present FASTR use case, an NN model may be trained in PyTorch or Tensorflow to predict a highly nonlinear physical process such as how coefficients within a surface chemistry kinetics model depend on system state variables, or some other physical process. Another relevant potential use case for this capability is to build data-driven models for heat transfer coefficients or to couple an NN–ROM of the Microreactor AGile Non-nuclear Experimental Testbed (MAGNET) facility with the FASTR loop within Dymola. This capability could also be used to integrate the FASTR loop with a ROM trained with high-fidelity simulations of major components such as the pump tank. The data-driven model could then act as a stand-in for a mechanistic model in cases where the physics are unknown (i.e., when the functional form of the algebraic or differential equation model is unknown) or are difficult to express in the Modelica language. This new Modelica–ONNX bridge capability lays the foundation for future work avenues to incorporate data-driven NNs into the Modelica-based models of the FASTR loop.

1.4.1 Example ONNX NN Model Use from Dymola

This capability is generic and can be used in any Modelica model; it is not directly tied to the FASTR system. To the authors' knowledge, this represents the first in-memory coupling of a generic ONNX-encoded NN with a Modelica model. Furthermore, we have demonstrated the of use of the ONNX-encoded NN within the Dymola environment, where the NN can be "dragged and dropped" into an existing Modelica model and interacted with via the Dymola GUI, as any function block would, as shown in Fig. 16.

To test the Modelica–ONNX bridge, first, a simple supervised regression NN was built using PyTorch. The network was a multilayer perceptron consisting of two fully connected hidden layers of width 20 with ReLU activation functions. The input layer takes two inputs, x_1, x_2, and the network returns a single output, $\hat{y}$. Synthetic training data, y_d, were generated according a 2D quadratic function with added Gaussian noise: $f_d = x_1^2 + x_2^2 + \varepsilon$ with $\varepsilon \sim \mathcal{N}(\mu = 0, \sigma = 0.1)$ and with a limited domain of $x_1, x_2 \in [-10, 10]$. The loss function was the sum of squares error, $L = \sum_i^N (\hat{y}_i - y_{d,i})^2$, where the number of training examples, N, was set to 200, and was evaluated at random values of x_1, x_2 within the $[-10, 10]$ domain bounds. After the simple PyTorch NN was trained, it was exported in the ONNX file format.

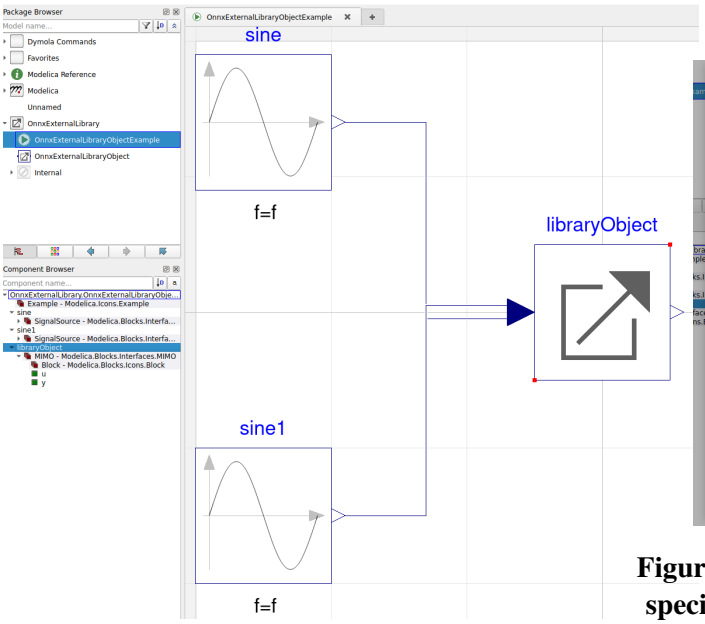


Figure 16. Dymola block diagram example of a ONNX-serialized NN model (libraryObject) connected with two sin function blocks supplying input to the NN.

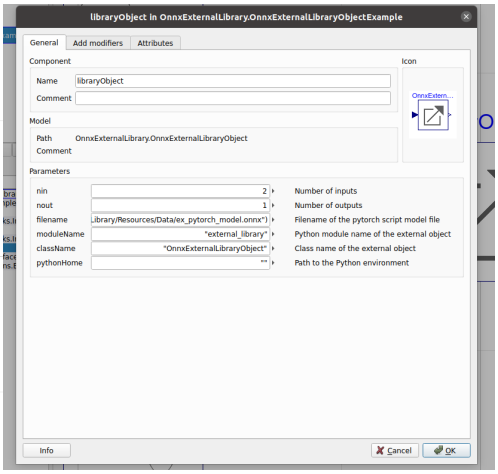


Figure 17. The ONNX NN model file path can be specified by setting the external Library Object parameters in Dymola. Additionally, the input and output dimensionality of the NN can be specified here.

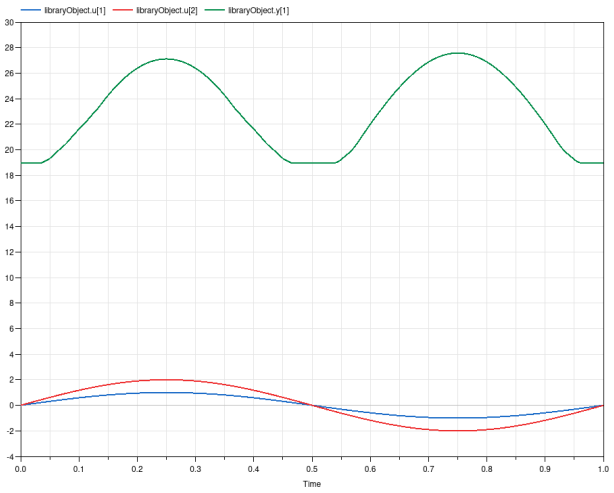


Figure 18. Output of the ONNX NN model (green) along with inputs (blue and red) visualized in Dymola's graph viewer.

2. VALUE/BENEFIT OF VIRTUALLY INTEGRATING FASTR WITH TEDS/DETAIL DIGITAL TWINS

This section describes value/benefit of virtually integrating the FASTR DT with the TEDS/Dynamic Energy Transport and Integration Laboratory (DETAIL)/MAGNET DTs. All DT models have already been developed for their respective systems. The ability to facilitate future opportunities of the TEDS/DETAIL/MAGNET systems will require the integration of a diverse array of projects. A loop similar to the FASTR loop could potentially be added as a future heat source to the TEDS loop, as an intermediate between TEDS and another system, or as the heat sink of MAGNET, i.e., a microreactor.

2.1 BRIEF DESCRIPTION OF TEDS/DETAIL/MAGNET

To address the growing need for physical experimental models that demonstrate the interconnect of integrated energy systems, Idaho National Laboratory has been developing and building the Dynamic Energy Transport and Integration Laboratory (DETAIL) facility. The DETAIL facility focuses on experimental research and validation of nuclear-renewable hybrid energy systems. The overall objective for the DETAIL facility is to demonstrate simultaneous, coordinated, and efficient transient distribution of electricity and heat for power generation, energy storage, and industrial end uses. Within DETAIL, there is a Thermal Energy Distribution System (TEDS), which is a thermal hydraulic flow loop with its own control system, the central system of the facility. TEDS integrates multiple experimental systems and includes a single-tank packed-bed thermocline, a Chromalox heater designed to simulate a thermal generator unit, an ethylene glycol heat exchanger, three materials to transfer heat around the system, and a series of insulated pipes to ensure proper redirection of flow and conservation of heat, as shown in Figure 19. A Modelica-based model has already been developed and tested as described in Frick et al. [22].

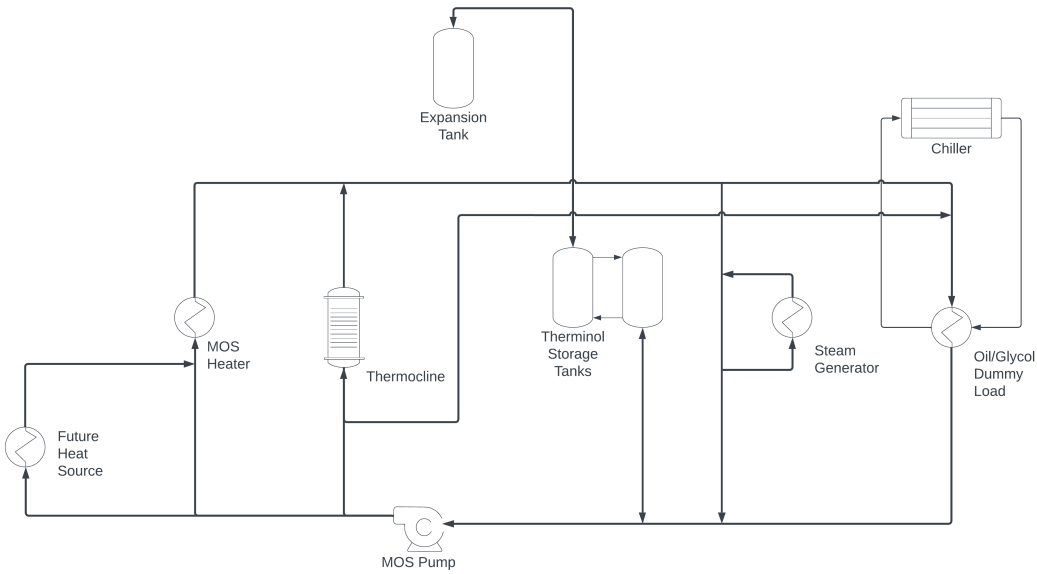


Figure 19. Simplified process flow diagram of the TEDS loop.

2.2 BRIEF DESCRIPTION OF MAGNET

The Microreactor AGile Non-nuclear Experimental Testbed (MAGNET) is an electrically heated thermal hydraulic test facility for demonstrating gas-cooled microreactor technologies. MAGNET can broadly be used to test heat transfer components, performance of core structures, heat pipe and other advanced heat removal methods, sensors and instruments and control systems, and to validate modeling and simulation tools. A simplified diagram of MAGNET is shown in Fig. 20.

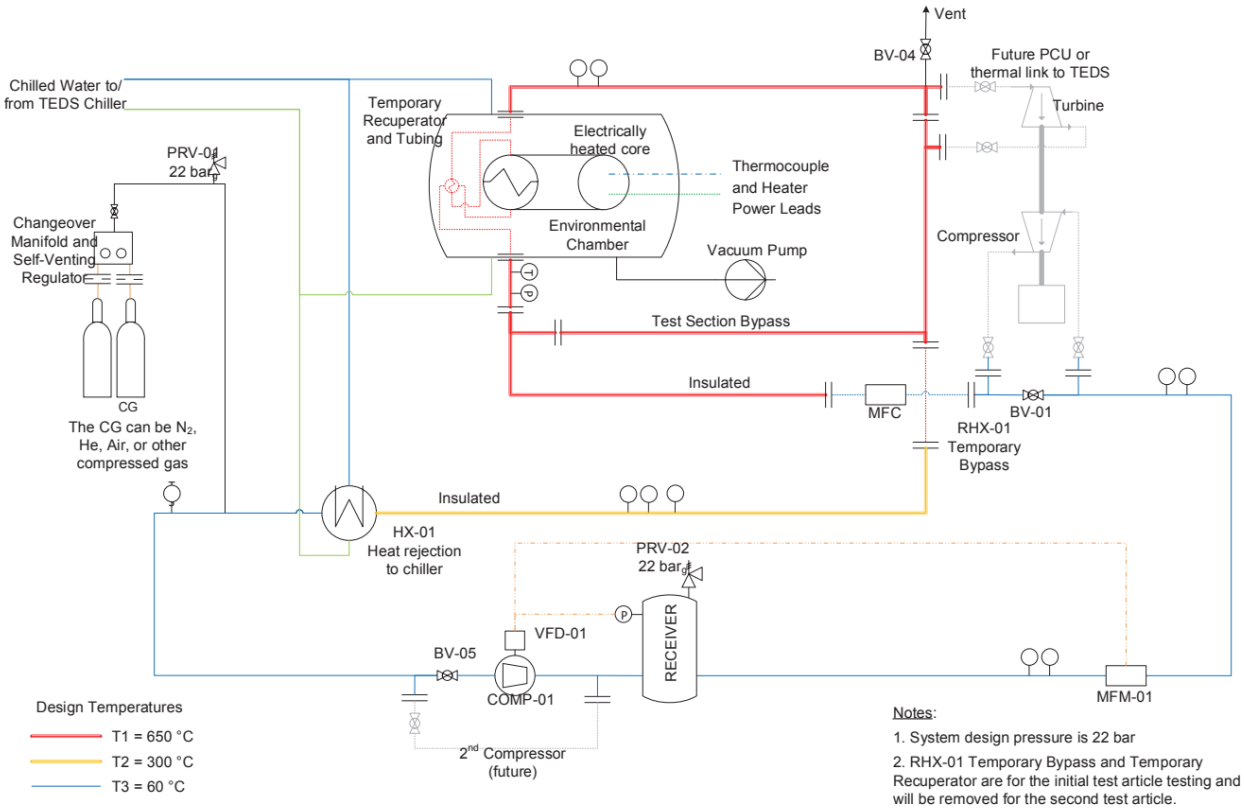


Figure 20. Simplified process flow diagram of the MAGNET system.

2.3 INTEGRATION OF TEDS, MAGNET, AND FASTR DTS

The most difficult part of creating flexible test facilities is planning for and capturing the resultant dynamic behaviors of multiple coupled systems. The ability to connect DTs remotely is a stepping stone toward integrating hardware loops virtually or integrating hardware and simulation models remotely. The first step in this investigation was to combine the TEDS loop DT and also the MAGNET DT with the FASTR DT as an initial proof of concept. Minor modifications are required to couple the models. First, the integration of FASTR with TEDS is described; the integration of FASTR with MAGNET is then described.

2.3.1 FASTR+TEDS: Integration

In Figure 21, the modified TEDS DT is shown with the MAGNET/TEDS heat exchanger hot side altered to create an interface with the FASTR DT. The inlet and outlet flows (port_a and port_b) and the mass flow

rate control signal (y) are given interface nodes. This will allow for the connection of the FASTR DT heat sink to the hot side of the TEDS heat exchanger and for TEDS to act as the FASTR loop's ultimate heat sink.

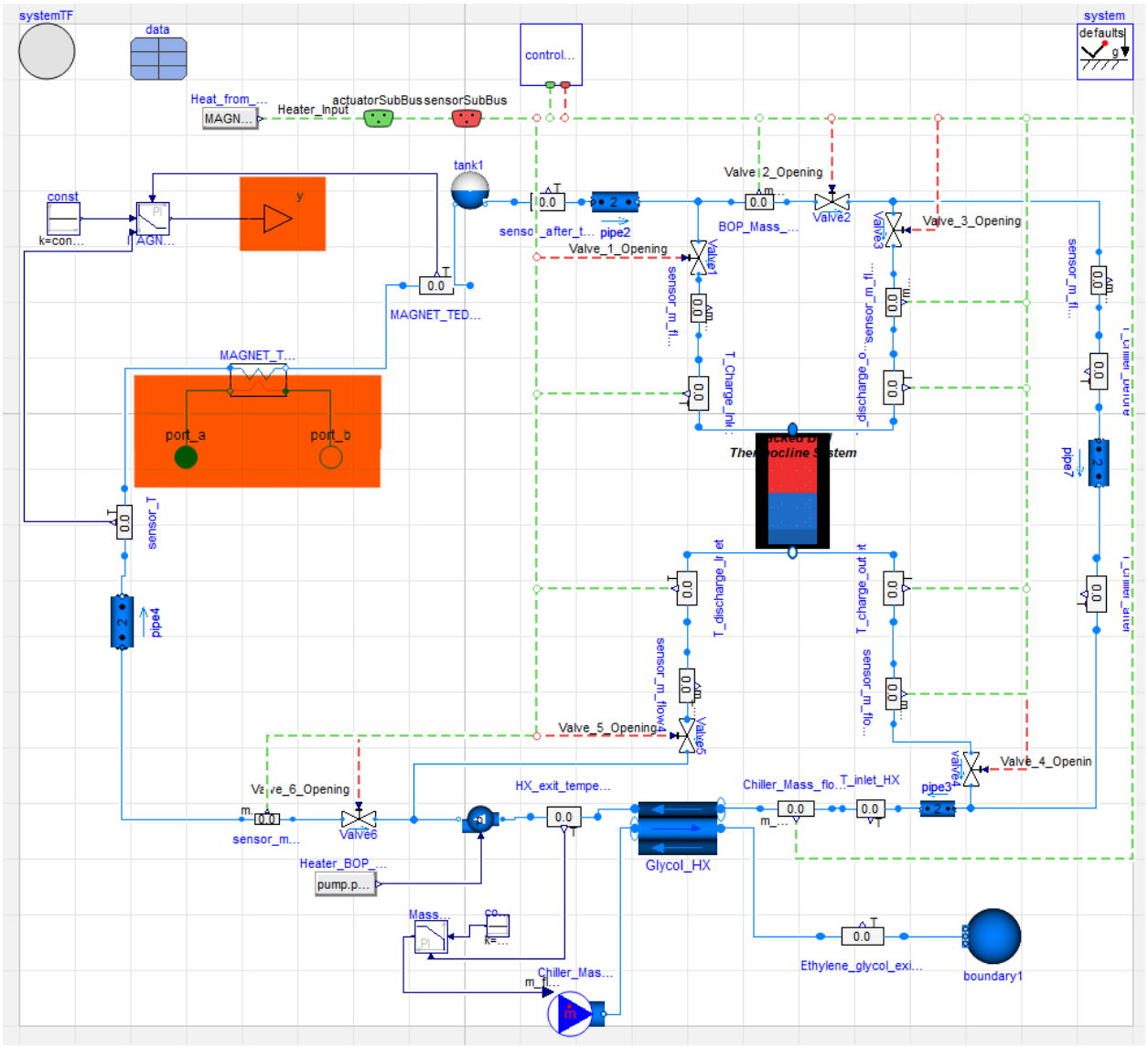


Figure 21. Modified Modelica model of the TEDS loop (modifications are highlighted in red).

A modified FASTR loop DT is shown in Fig. 22. The heat exchanger of the heat sink of the FASTR model has been removed, along with the pump control table, and replaced with the TEDS model. Connections are made to the flow ports, and the feedback for mass flow control is routed to the FASTR pump.

2.3.2 FASTR+TEDS: Testing

Attempts were made to test the behavior of the FASTR+TEDS integrated model. The functioning model was investigated using FASTR as a heat load to the TEDS system; the glycol heat exchanger was used to dump heat to the chiller system. Automatic controls maintained prescribed temperature points for the

glycol outlet temperature. The thermocline was isolated in these tests. The outlet temperature of the TEDS side of the MAGNET–TEDS heat exchanger (now the FASTR–TEDS heat exchanger) controlled the mass flow rate of the FASTR loop pump. For the first 100 s, pump tank pressure was controlled to make sure the pump is properly fed; this was done to mimic the actual FASTR loop startup. At 100 s, the top port tank control was initiated so as to push down the tank level to a desired operating value to prevent flow pulsations in the system. A heat load of 28 kW was provided through the FASTR heater to demonstrate the sinking capacity of the TEDS loop. Due to the complexity of the model, it was very slow running —24 h of compute time to generate 120 s of simulation time. This made for very limited testing.

The simulation tracked the heat exchanged in the FASTR–TEDS heat exchanger. Results can be seen in Fig. 23. It can be seen that after the initial transients, the system began to converge on the 28 kW value that is generated in the FASTR heater. The blip in heat exchange at 100 s is due to the top tank controls being engaged and pushing more flow back into the system, thus increasing heat exchange. The tank levels can be seen in Fig. 24. Key loop temperatures are shown in Fig. 25. It can be seen that the glycol temperatures in the TEDS loop are converging for the inlet to the FASTR to TEDS heat exchanger, and the overall delta T is coming up to meet the required values for removing the heat in steady state.

Due to computational speed limitations, further FASTR+TEDS testing was abandoned. There is potential for many more interesting cases to be studied. Using the thermocline of TEDS to store and unload heat from FASTR would be interesting. Studying potential dynamics between FASTR and the thermocline would also be promising. These could be pursued in future works with more computational horsepower and if the model convergence is improved.

2.3.3 FASTR+MAGNET: Integration

In Fig. 26, the modified MAGNET DT is shown with the MAGNET/TEDS heat exchanger cold side altered to create an interface with the FASTR DT. The inlet and outlet flows (port_a and port_b) are given interface nodes. This allows for the connection of the FASTR DT heat source to be connected instead of the cold side of the MAGNET to TEDS heat exchanger, as well as for MAGNET to act as the heat source of the FASTR loop. The integrated FASTR+MAGNET model can be seen in Fig. 27 where the MAGNET model is in the place of the heat source of FASTR.

2.3.4 FASTR+MAGNET: Testing

Fortunately, the FASTR+MAGNET integrated model was much less computationally intensive and was quick to generate interesting results. To achieve a functioning model, a constant wall temperature was placed as an ultimate heat sink for the FASTR loop. The temperature was set at 20 °C for the testing, and the heat exchanger was set to use the Dittus–Boelter correlation as the closure relation. The pump of FASTR model was set to the same startup plan as used in the isothermal test, where there are some interesting transients. It also contains the same tank-level control schemes as the previous FASTR+TEDS model.

The simulation takes less than one minute of compute time to generated 10,400 s of data. Interesting behavior can be seen in the balance between the FASTR heat removal capability and the MAGNET heat generation, as shown in Fig. 28. It can be seen that at initial lower flow rates, the FASTR system cannot remove heat fast enough; however, when the pump is pulsed to higher powers, it is capable of effectively removing all the heat. Up to around 2,000 s, FASTR is in fully natural convection driven mode. The pump is pulsed to 35% for about 500 s and then returned to natural convection mode until about 3,500 s. The

pump is gradually step-pulsed up to 45% at around 5,000 s and held for a while. At around 7,500 s, the pump is stepped up to 50% and settles near the steady state after around 8,000 s. Finally, the pump is shut down.

The temperature behavior of the system can be seen in Fig. 29. The MAGNET inlet and outlet temperatures match well with their respective FASTR heat exchanger inlet and outlet temperatures. The temperatures increase in the phases where the heat exchange is lagging behind the MAGNET input. The temperatures then drop at around 3,500 s as the FASTR pump is fully engaged. The system is then brought down to the approximate steady state operating values for the FASTR loop between 550 and 600 °C. The thermal expansion impacts on the system are seen in the tank levels. The pump and top port tank-level dynamics demonstrate the ability of the FASTR loop to endure the fluctuations in pump flow rate as shown in Fig. 30. It can be seen that as the system heats up, the pump tank level increases; however, once the flow rate is high enough to meet the heat removal needs, the temperatures drop, and the tank level drops. The top port is being actively controlled to maintain a steady level and functions effectively.

2.3.5 Discussion of value/benefit of virtually integrating FASTR with TEDS/DETAIL/MAGNET DTs

Initial testing has shown great potential for virtual integration of FASTR with the TEDS/DETAIL/MAGNET DTs. However, there appear to be potential difficulties in computational requirements for more complex models, most likely due to the complexity of the individual system's operating modes. The ability to couple the FASTR twin as both a heat source and a heat sink were demonstrated effectively. The ability to use FASTR as a heat sink or intermediate loop shows much more promise. The value of the DTs to demonstrate the potential for integration is generating many research ideas about how to locally and remotely couple DTs with real-world systems. The efforts to integrate the FASTR DT with the FASTR hardware demonstrate an opportunity to remotely couple real hardware with DT for testing. This could potentially alleviate many potential complications in future benchmarking of combined systems. For example, the FASTR DT could be coupled with the actual MAGNET and use TEDS as a surrogate for the FASTR loop, and the FASTR loop could use the virtual or real data from the TEDS system to act as a surrogate heat source driver in FASTR. Various levels of hardware and software in the loop are possible. It certainly warrants continued investigation.

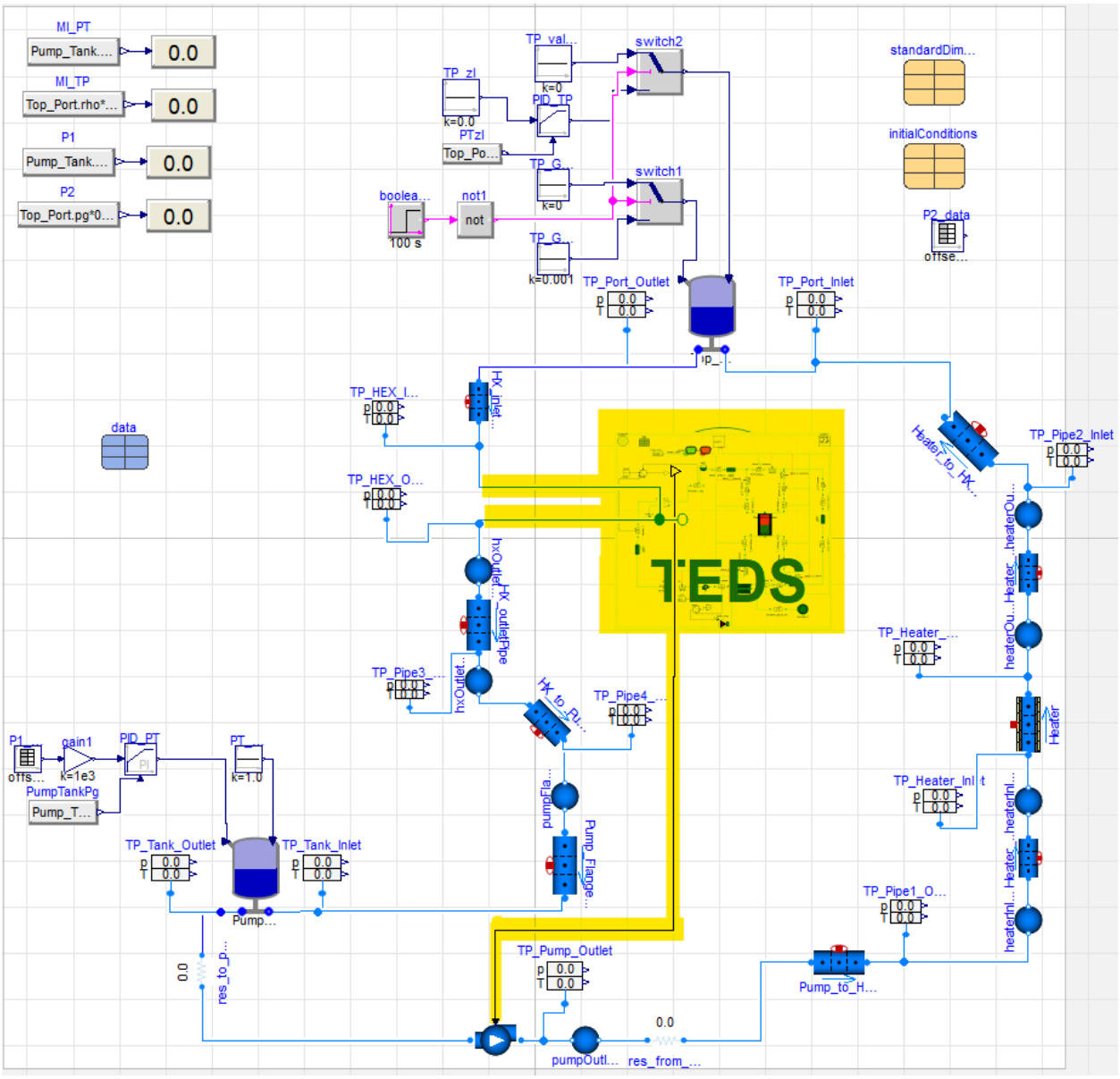


Figure 22. Modified Modelica model of the FASTR loop with the modified TEDS loop included (modifications are highlighted in yellow).

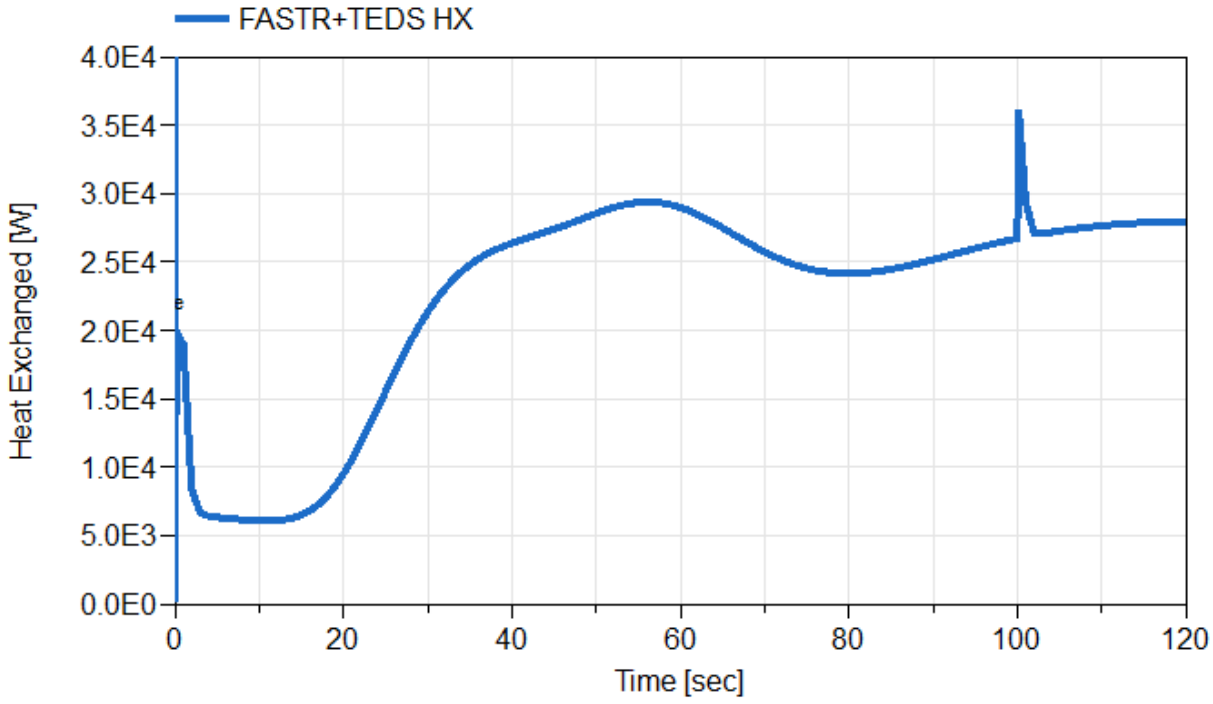


Figure 23. Heat exchanged between the FASTR loop and TEDS loops from the simulation.

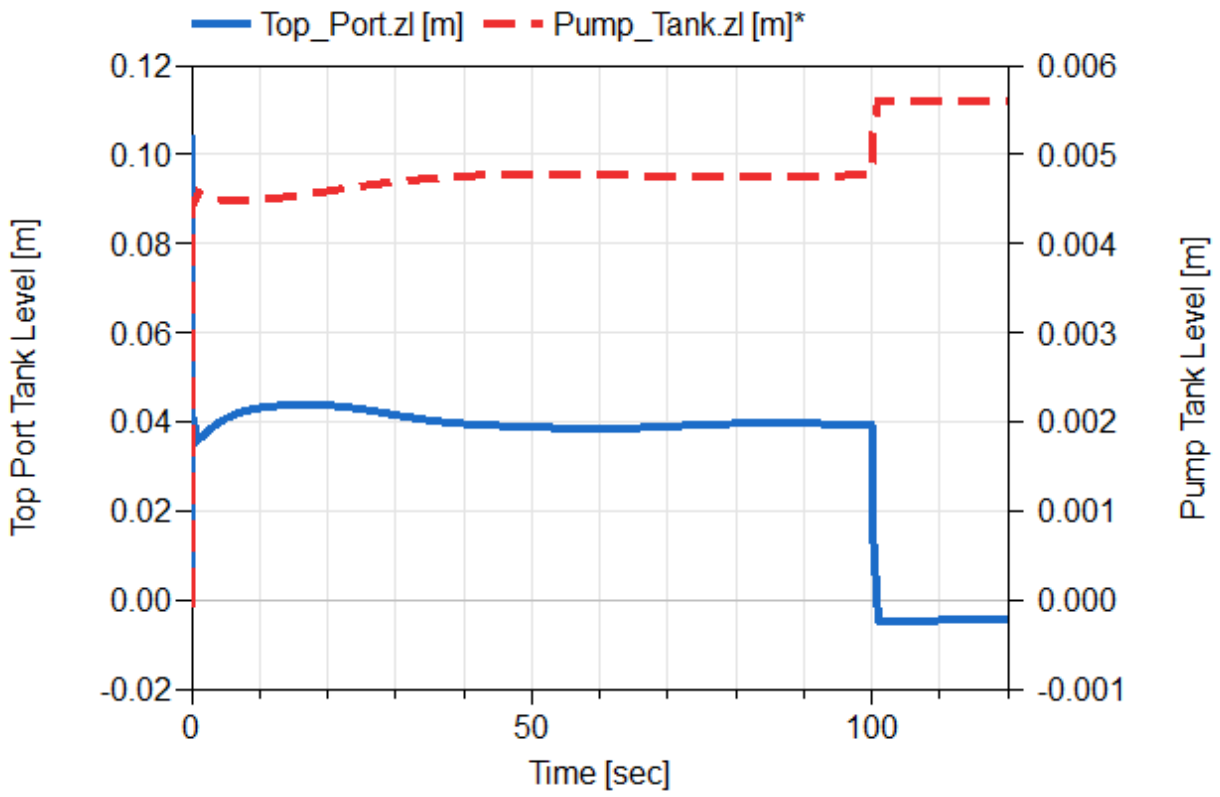


Figure 24. Balancing between the top port tank and pump tank levels during the transients (the top port level control is engaged at 100 s).

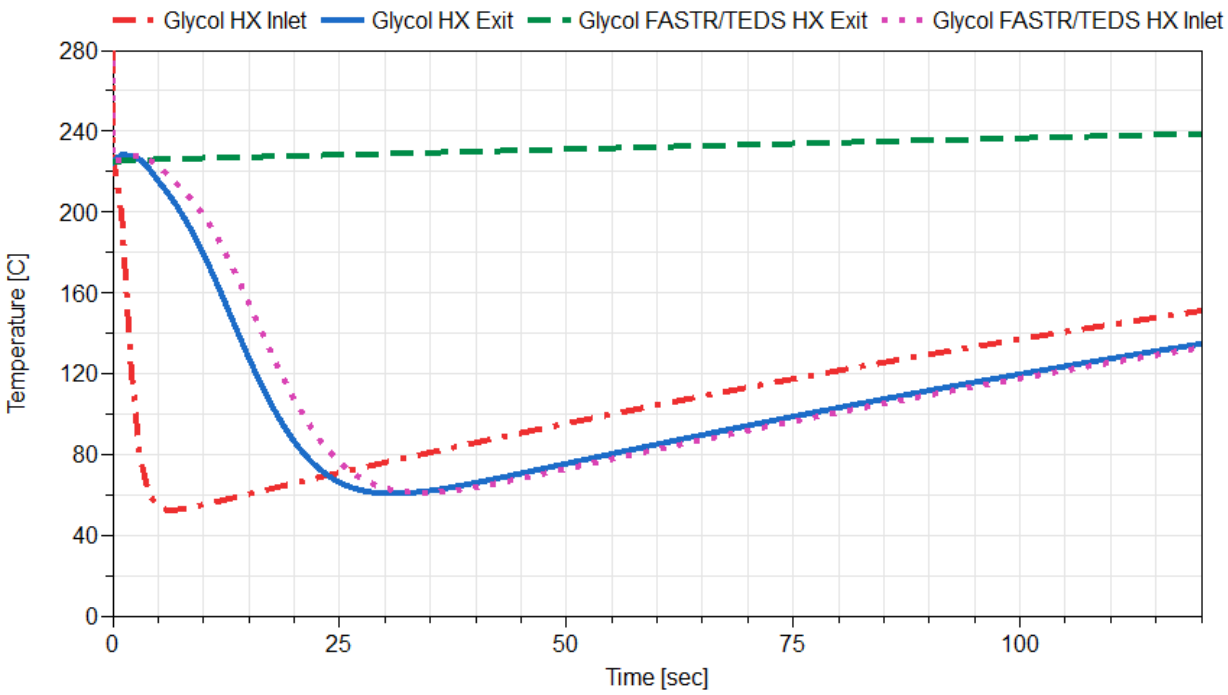


Figure 25. Key temperatures in the TEDS loop around the heat exchangers.

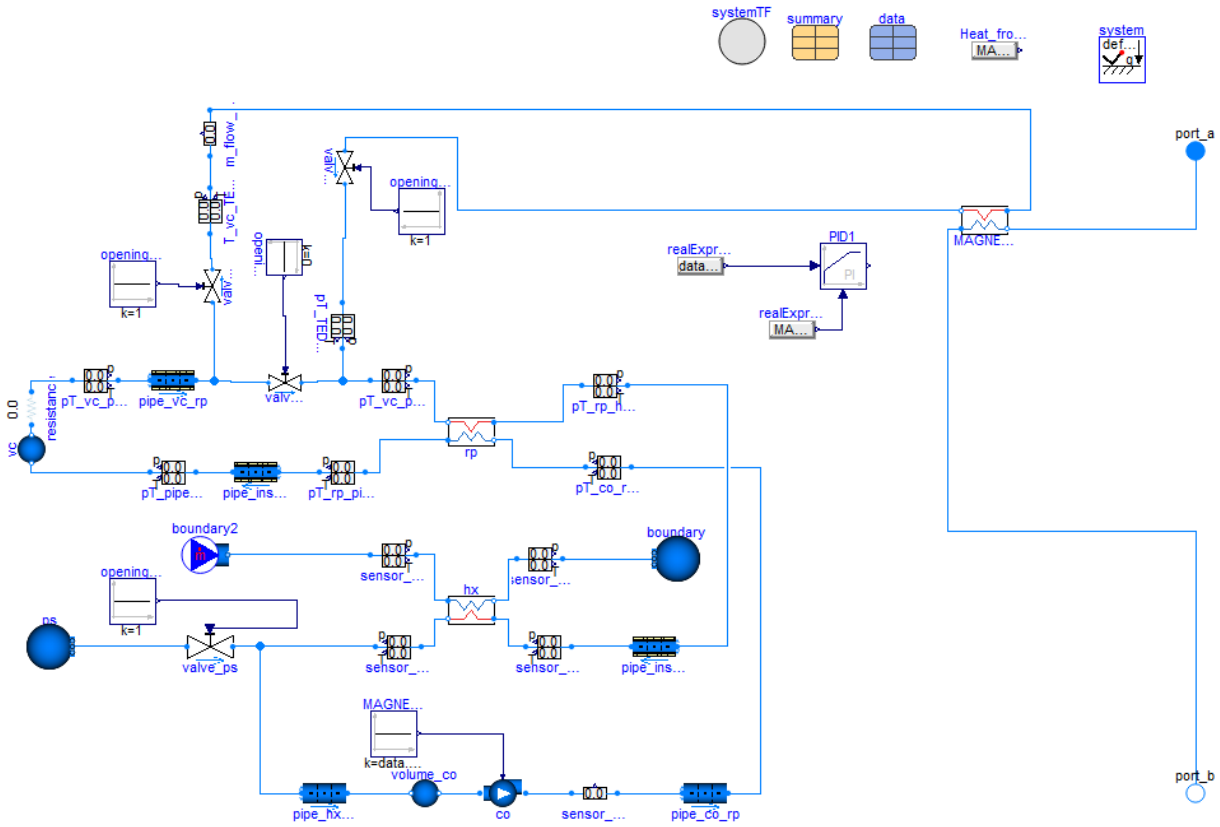


Figure 26. Modified model of the MAGNET system.

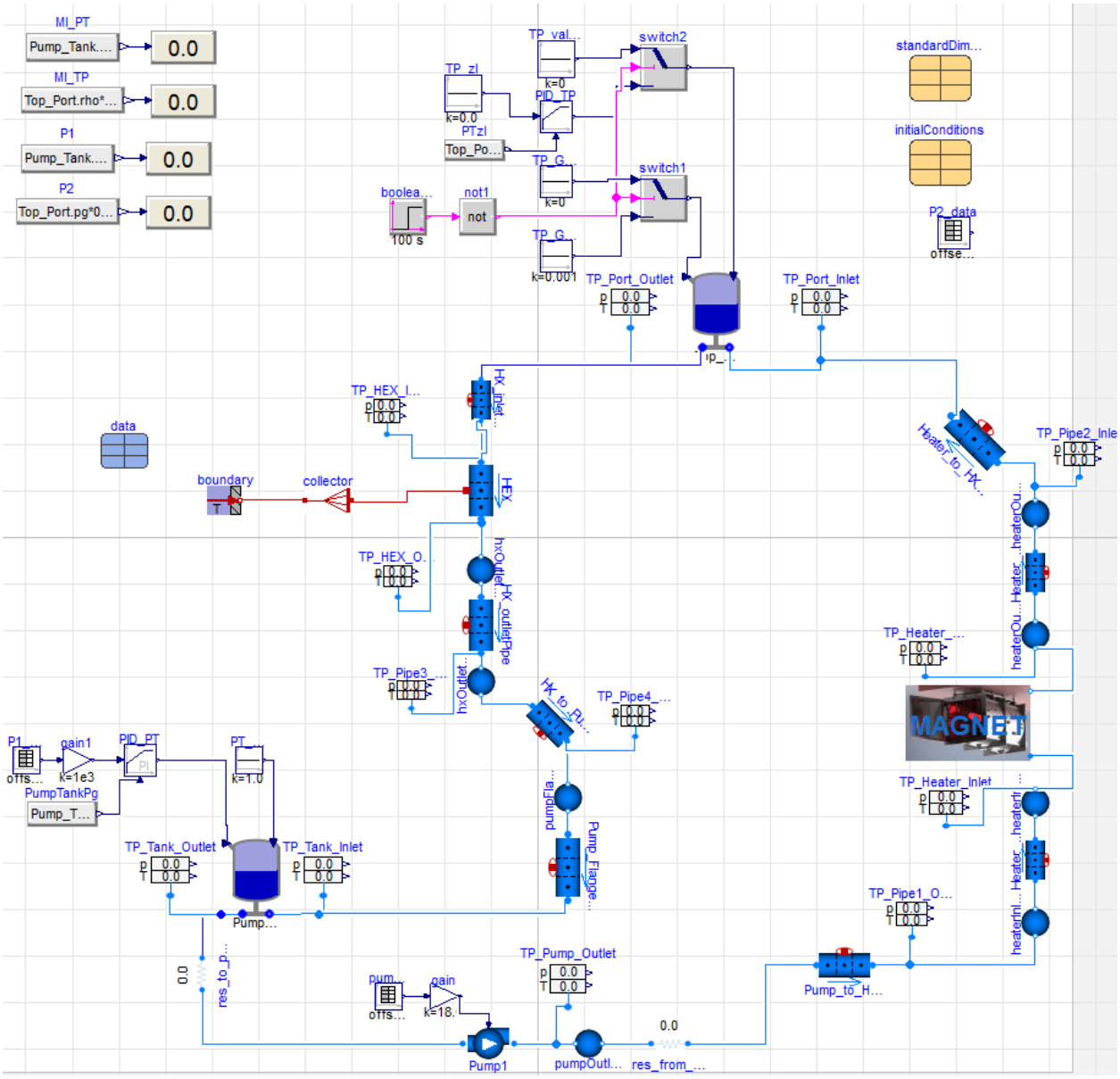


Figure 27. Integrated model of the FASTR+MAGNET system.

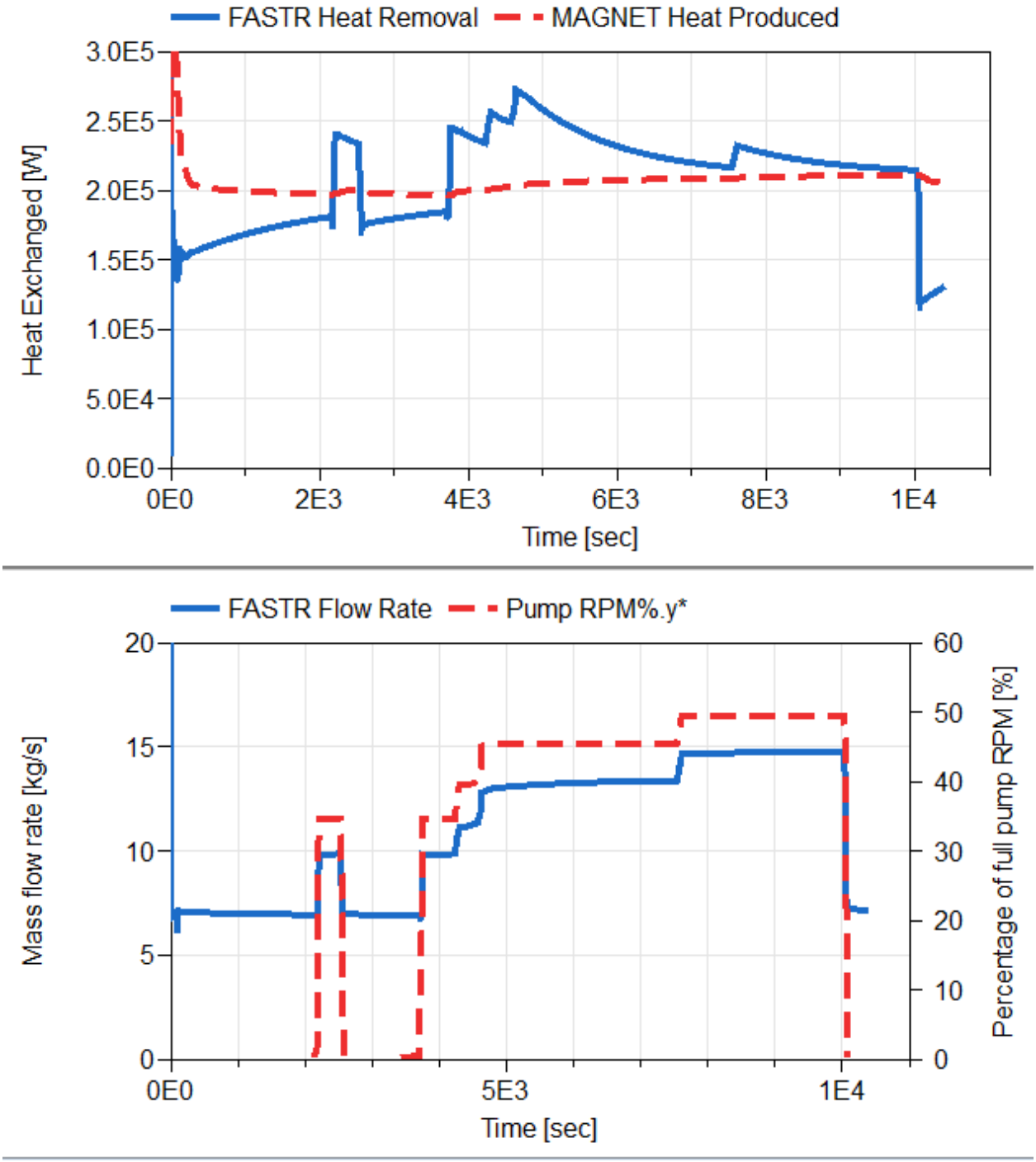


Figure 28. Heat produced by MAGNET compared to heat removed by FASTR and the effect of FASTR mass flow rate on the overall system dynamics.

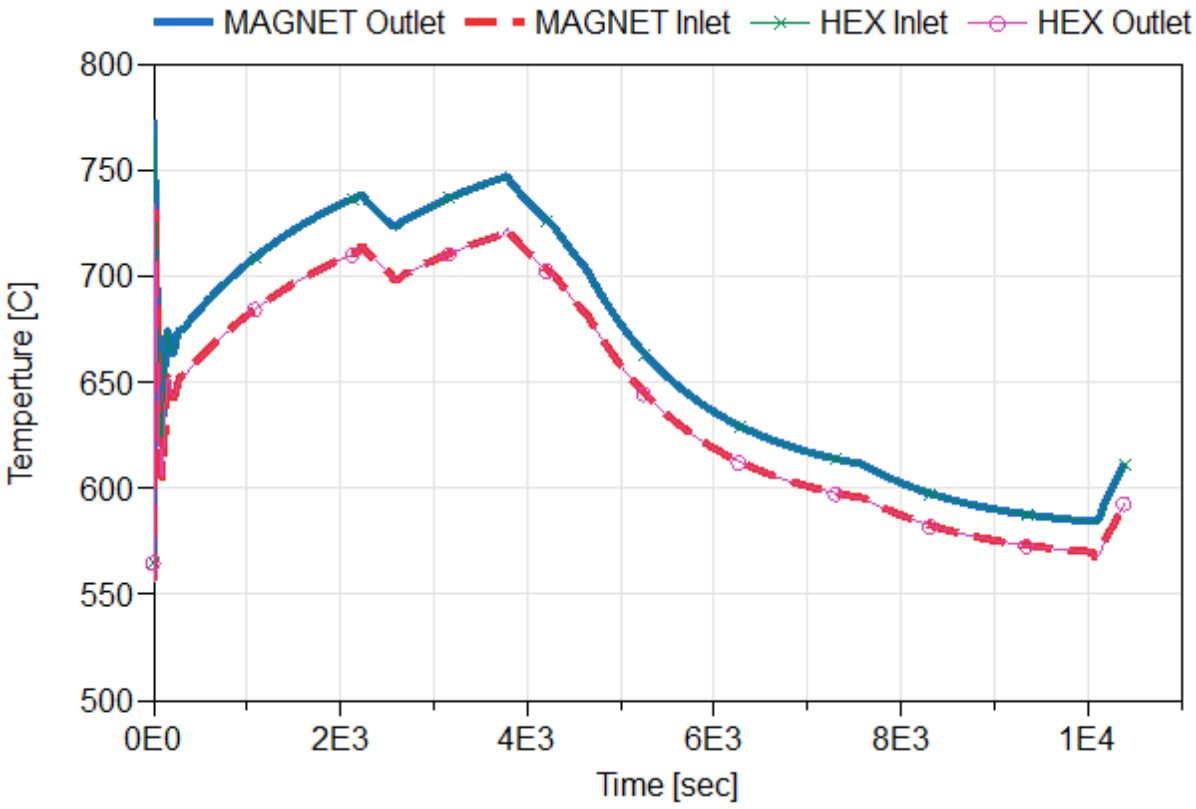


Figure 29. Dynamics of the MAGNET and FASTR HEX temperatures under various flow rates.

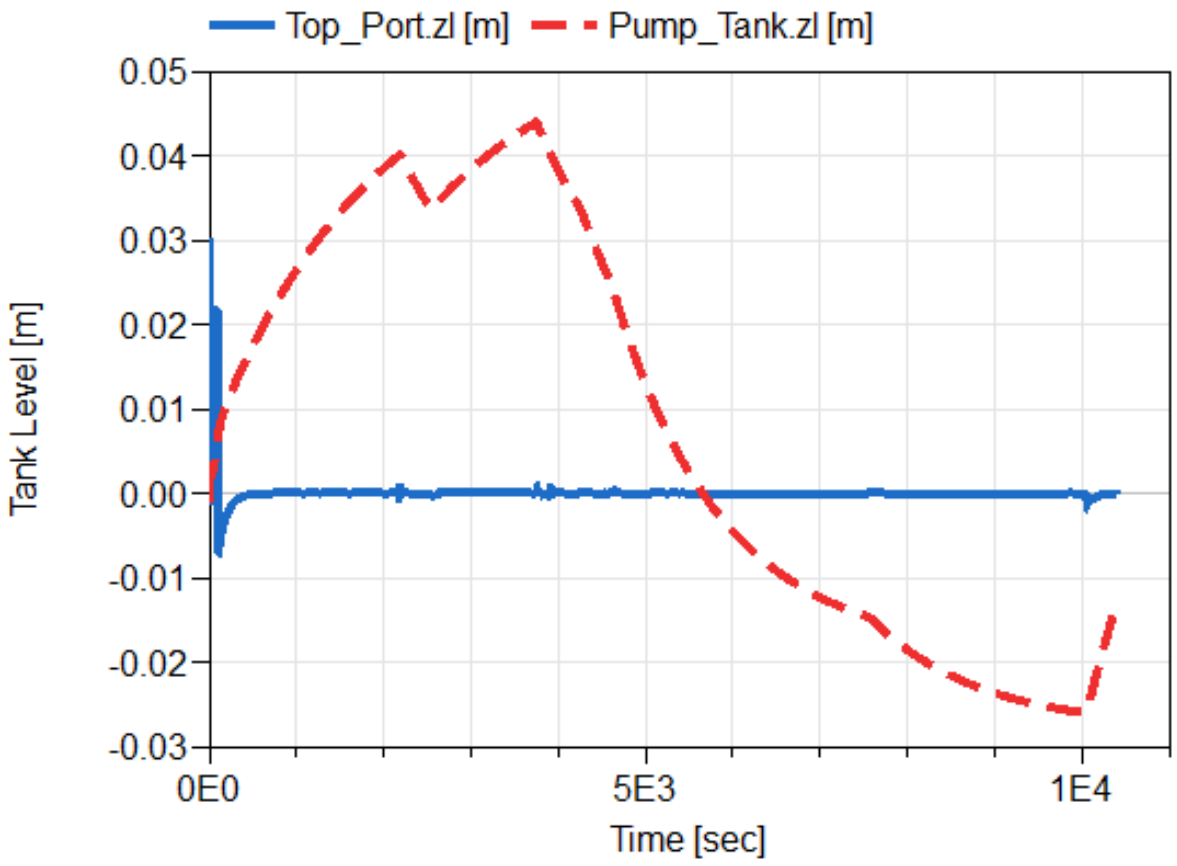


Figure 30. Dynamics of the pump and top port tank levels under various flow rates.

3. PATH FORWARD FOR FUTURE INTEGRATION OF A FASTR-TYPE LOOP WITH A MICROREACTOR AT INL

This project demonstrates the use of DTs of integrated energy systems to understand system dynamics and to discover the potential of integrated hardware tests with and without software-in-the-loop. Some major lessons are being learned from attempting to develop a DT and to integrate the DT with hardware at FASTR. For example, the communication protocols, the data acquisition systems, and programmable logic controllers are non-trivial factors in the ability of the DT to interact with the hardware. The ability to safely integrate two-way communication between DTs and the hardware is an area desperately in need of further development. Comparing the DTs with real-world data also clearly demonstrates the shortcomings of the modeling efforts. Nevertheless, the outlook is positive, with possibilities for demonstrating and moving toward semi-autonomous and even autonomous control of integrated energy systems.

One key finding of the modeling effort is the ability to generate computationally intensive models. The ability to add faster running surrogate models of the more complex parts of a system can boost performance. The ability to integrate NN or other ML models inside the Modelica environment by way of the newly developed Modelica–ONNX bridge is paving one potential pathway to develop faster performing models. The ability for these models to learn from real-time data and to produce real-time predictions of behavior is the overarching goal of DTs. Despite all the advancement, there is still a long way to go to integrate all of these technologies in a practical and useful way.

The virtual integration efforts have shown that a FASTR-type loop would be promising for integration into MAGNET and potentially actual microreactor projects like MARVEL or others. Temperature, tank level, pump flow, and heat flow dynamics can be predicted and key performance parameters can be identified with the DTs. This has the potential to lower risks of design and implementation of a FASTR-type loop at INL. Virtually, the integrated system is not overly complex in its dynamics. While the FASTR loop was successfully commissioned, the salt flow rate, for example, must be verified to enable quantified heat transfer testing and verification of the custom-designed molten salt pump [2]. Future test campaigns for FASTR will focus on gaining system operation experience and characterizing the system operation envelope, among others, and the DT could potentially play a key role in such work. With increasing confidence in the DT to predict the system dynamics of FASTR, including hardware integration with an industrial PLC, it could be used to set the stage for future DT efforts for microreactor projects like MARVEL.

4. ACKNOWLEDGMENTS

We would like to acknowledge funding from the DOE-NE Integrated Energy Systems program. We are grateful for the many fruitful discussions with the FASTR loop facility PI, Dr. Kevin Robb, and other researchers in his lab—Dr. Nolan Goth and Mr. Ethan Kappes. We are particularly grateful to them for patiently reviewing various aspects of the testing and design of the FASTR loop and for allowing access to connect a workstation and test the preliminary DT model. We also thank Mr. Scott Nelson for his help with the initial Modelica modeling effort of FASTR as well as Drs. Mohammad Abdo and Junyung Kim at INL for their insightful suggestions on ROM development in this work.

5. REFERENCES

[1] Kevin R. Robb, Padhraic L. Mulligan, Graydon L. Yoder Jr., Kurt Smith, and Jordan Massengale. Facility to alleviate salt technology risks (fastr): Preliminary design report with failure modes and effects analysis. 12 2019.

[2] Kevin Robb and Ethan Kappes. Facility to alleviate salt technology risks (fastr): Commissioning update. 3 2023.

[3] Kevin Robb, Seth Baird, Jordan Massengale, Ethan Kappes, and Padhraic L. Mulligan. Facility to alleviate salt technology risks (fastr): Design report. 12 2022.

[4] Kevin R. Robb, Seth Baird, Jordan Massengale, Nathaniel Hoyt, Jicheng Guo, and Colin Moore. Engineering-scale batch purification of ternary mgcl2-kcl-nacl salt using thermal and magnesium contact treatment. 8 2022.

[5] Onuschak Rebecca and Shannon M Bragg-Sitton. Integrated energy systems program management plan. 1 2021.

[6] Michael S Greenwood, Richard Hale, Lou Qualls, Sacit Cetiner, David Fugate, Thomas Harrison, et al. TRANSFORM-TRANsient Simulation Framework of Reconfigurable Models. Technical report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2017.

[7] Torsten Blockwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional Mockup Interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International Modelica Conference 173, September 3-5, 2012, Munich, Germany*, 2012.

[8] Scott Nelson. Transform digital twin development of the facility to alleviate salt technology risks (fastr. (ORNL/SPR-2022/2478), 6 2022.

[9] Francesco Casella, Alberto Leva, et al. Modelling of distributed thermo-hydraulic processes using modelica. In *Proc. MathMod Conference*, pages 514–521, 2003.

[10] Francesco Casella, Alberto Leva, et al. Modelica open library for power plant simulation: design and experimental validation. In *Proceeding of the 2003 Modelica conference, Linkoping, Sweden*. Citeseer, 2003.

[11] Ian Ottoway. pycomm3. https://github.com/ottowayi/pycomm3, 2023.

[12] Dassault Systems. Fmpy, 2021.

[13] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[14] B O. Koopman. Hamiltonian systems and transformations in hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–8, 1931.

[15] Nicola Demo, Marco Tezzele, and Gianluigi Rozza. PyDMD: Python Dynamic Mode Decomposition. *The Journal of Open Source Software*, 3(22):530, 2018.

[16] J. Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[17] Hassan Arbabi and Igor Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

[18] Santosh Tirunagari, Samaneh Kouchaki, Norman Poh, Miroslaw Bober, and David Windridge. Dynamic Mode Decomposition for Univariate Time Series: Analysing Trends and Forecasting. 2017.

[19] Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[20] ONNX Runtime developers. Onnx runtime. https://onnxruntime.ai/, 2021.

[21] Modelica Association. Modelica® – a unified object-oriented language for systems modeling language specification. https://specification.modelica.org/, 2023. Version: 3.7-dev.

[22] Konor L. Frick, Shannon M. Bragg-Sitton, and Cristian Rabiti. Development of the inl thermal energy distribution system (TEDS) in the modelica eco-system for validation and verification. 7 2020.

APPENDIX A. DAQ-FMU Interfacing Scripts

APPENDIX A. DAQ-FMU interfacing scripts

```
"""
Created on April 20, 2023

@author: Vineet Kumar

This script runs the FMU for an initial time and then runs it with a 1 second timestep
    with inputs from the PLC before termination
"""
from __future__ import division, print_function, unicode_literals, absolute_import
import os
from pycomm3 import LogixDriver, CIPDriver
from fmpy import simulate_fmu, read_model_description, extract, dump, instantiate_fmu,
    read_csv, write_csv
import numpy as np
import time
import warnings
mypath = os.path.dirname(os.path.abspath(__file__))

fmuInputs = []
fmuOutputs = []

# use the step_finished callback to stop the simulation at pause_time
def step_finished(time, recorder):
    """ Callback function that is called after every step """
    return time < pause_time

def runFMU(inputs, fmu_state, start_time):
    """ Run fmu every one second """
    results = simulate_fmu(filename=settings['filename'],
                           start_time=start_time,
                           stop_time=settings['stop_time'],
                           output_interval=1.0,
                           input=inputs,
                           output=settings['outputs'],
                           fmu_instance=fmu_instance,
                           fmu_state=fmu_state,
                           terminate=False,
                           step_finished=step_finished,
                           debug_logging=False)
    # retrieve the FMU state
    fmu_state = fmu_instance.getFMUState()
    return results, fmu_state

def read_multiple(tags):
    """ Read LogixDriver PLC tags """
    values = []
    with LogixDriver('192.168.0.6/1') as plc:
        plc.read(*tags)
        for typ in plc.data_types:
            values.append(plc.data_types[typ]['value'])
    return values


if __name__ == "__main__":
    # Get the FMU file name
```

```
    fmu_filename = os.path.join(mypath,'../FMUs/
    FMI_FASTR_LOOP_SALT_ISOTHERMAL_TEST_PT_PC_no_HL.fmu')

    #Read and dump the FMU file
    dump(fmu_filename)
    model_description = read_model_description(fmu_filename)

    unzipdir = extract(fmu_filename)

    # instantiate the FMU before simulating it, so we can keep it alive
    fmu_instance = instantiate_fmu(
        unzipdir=unzipdir,
        model_description=model_description,
    )

    for variable in model_description.modelVariables:
        if variable.causality == 'input':
            fmuInputs.append(variable.name)
        if variable.causality == 'output':
            fmuOutputs.append(variable.name)

    print("Correct variable input names")
    print("FMU Input variable list is ", fmuInputs)
    print("FMU Output variable list is ", fmuOutputs)

    start_time = 0.0 # Start time of the FMU in s
    stop_time = 20.0 # Stop time of the FMU in s
    pause_time = 10.0 # Initial pause time in s

    # Two time varying inputs are provided - Gas pressure and pump RPM
    dtype = [('time', np.double), ('PT_Pg', np.double), ('PumpPerRPM', np.double)]

    # Read CSV file for the two inputs
    inputs = np.array(read_csv(os.path.join(mypath, '..', 'data','fmuInputs.csv')),
    dtype=dtype)

    settings = {
                'filename':unzipdir,
                'start_time':start_time,
                'stop_time':stop_time,
                'output_interval':10.0, #initial output interval
                'outputs':fmuOutputs
                }

    # Run the FMU for an initial time until pause time
    results = simulate_fmu(filename=settings['filename'],
                           start_time=settings['start_time'],
                           output_interval=settings['output_interval'],
                           input=inputs,
                           output=settings['outputs'],
                           fmu_instance=fmu_instance,
                           terminate=False,
                           step_finished=step_finished,
                           debug_logging=False)

    # Log the results of the FMU for the initial run
```

```
    resultSummary = results

    # retrieve the FMU state
    fmu_state = fmu_instance.getFMUState()
    start_time = pause_time
    print("FMU ran for initial time, now starting 1s timestep run.")
    print("****************************************")
    print("****************************************")
    print("***         start FMU SS run          ***")
    wf = "{0:11.3E} {1:11.3E} {2:11.3E} {3:11.3E} {4:11.3E} {5:11.3E} {6:11.3E}
    {7:11.3E} {8:11.3E}"
    # While loop to run the FMU with a 1 second timestep
    while start_time <= stop_time - 2.0:
        pause_time = start_time + 1.0 # Advance the pause time
        #Hardware interfacing with the PLC
        start_time_sys = time.time()
        tags = ['PT_03_kPa','P_01_Speed'] #Tags for pump tank pressure in KPa and pump
     drive speed in %
        plcInputs = read_multiple(tags)
        fmuInputs = np.array((start_time, plcInputs[0], plcInputs[1]), dtype=dtype)
        resultsApp, fmu_state = runFMU(fmuInputs, fmu_state, start_time)
        print("   time (s)","   TankOutP (kPa)","   PumpOutP (kPa)","   HeaterInP (kPa
    )","   HeaterOutP (kPa)", "   HEXInP (kPa)","   HEXOutP (kPa)","   TankInP (kPa)","
       TankInT (degC)")
        print(wf.format(resultsApp[-1][0], resultsApp[-1][1]*1e-3, resultsApp[-1][2]*1
    e-3, resultsApp[-1][4]*1e-3,
                        resultsApp[-1][5]*1e-3, resultsApp[-1][9]*1e-3, resultsApp
    [-1][10]*1e-3, resultsApp[-1][13]*1e-3,
                       resultsApp[-1][14]-273.15))
        end_time_sys = time.time()
        #Basic synchornization used with the assumption that the FMU
        #takes less than 1.0s to retun values
        dt_sys = end_time_sys - start_time_sys
        if dt_sys > 1.0:
            warnings.warn('System time not synchronized with FMU time')
        elif dt_sys < 1.0:
            time.sleep(1.0-dt_sys)
        else:
            pass
        # Append the results every 1s
        resultSummary = np.concatenate((resultSummary, np.expand_dims(resultsApp[-1],
    axis=0)))
        start_time+=1.0 # Advance the FMU start time

    print("The FMU will terminate in 1s.")
    results = simulate_fmu(filename=settings['filename'],
                        start_time=start_time,
                        stop_time=settings['stop_time'],
                        output_interval=1.0,
                        input=inputs[int(start_time)],
                        output=settings['outputs'],
                        fmu_instance=fmu_instance,
                        fmu_state=fmu_state,
                        terminate=True,
                        debug_logging=False)

```

```
157     resultSummary = np.concatenate((resultSummary, np.expand_dims(resultsApp[-1], axis
        =0)))
158
159     # combine and output the results to a csv file
160     write_csv(os.path.join(mypath, '..', 'output','fmu_results.csv'), resultSummary,
        columns=None)
161     print("Simulation finished")
```

Listing 1. DAQ-FMU code procedure `runFMU.py`