Dynamic System Modeling & PID Controller Design for a Molten Salt Microreactor

A Thesis
Presented in Partial Fulfillment of the Requirements for the
Degree of Master of Science
with a
Major in Nuclear Engineering
in the
College of Graduate Studies
University of Idaho
by
Sam J. Root

Approved by:
Major Professor: Michael G. McKellar, Ph.D.
Committee Members: Dakota Roberson, Ph.D.; Robert A. Borrelli, Ph.D.
Department Administrator: Indrajit Charit, Ph.D.

December 2023

Abstract

The Molten Salt Nuclear Battery (MSNB) is a Generation IV microreactor concept that employs uranium-tetrafluoride fuel dissolved directly in the primary coolant. This thesis presents two dynamic multi-physics models implemented in Python 3.10 using Euler's forward method.

The first model is a zero-dimensional, object-oriented nuclide concentration code, intricately linking the mathematics of the xenon-135 decay chain to two-phase equilibrium mass transport. This model was developed to investigate the potential of fission gas stripping in reducing reactor downtime and extending fuel lifetime. Rigorous validation against simplified analytic solutions of the system of ordinary differential equations provides confidence in the accuracy of the model, while subsequent case studies demonstrate the concept to be thermodynamically favorable.

The second model couples the natural-circulation flow mode to reactor point kinetics with the uniform-state uniform-flow condition to simulate the core's response to demand transients. During the investigation, the reactor demonstrated passive stability in both steady-state operation and under moderately aggressive transient conditions. Leveraging this autonomous response, in combination with reactivity actuators characterized through Monte-Carlo neutronics with Serpent 2, a PID feedback control loop was designed. Using the Ziegler-Nichols tuning methodology, the controller reduces the settling time following step-changes to power demand by [X amount] and quantify effectively eliminates set-point overshoot following transients with a ramp-rate of 400 kW/min.

Acknowledgements

I am greatly thankful for the exceptional guidance, mentorship, and friendship of Professor McKellar. Each week, I looked forward to our meetings, where his wisdom, reassuring presence, and deep understanding of the intricate mathematics behind thermal fluid systems consistently reinvigorated my enthusiasm towards this research.

My committee members played distinct and indispensable roles in the completion of this thesis. Dr. Borrelli's guidance and mentorship over the last 2+ years have been transformative, helping me grow significantly as a researcher. His reliable availability to tackle the challenges of this project was a cornerstone of its success. Dr. Roberson exhibited great patience as I gained intuition in his area of expertise, and provided depth of understanding where mine was rudimentary.

This work and my coursework was completed under a Graduate Fellowship funded by Nuclear Regulatory Commission (NRC).

This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

Dedication

To my mother, Tammy, who planted and nurtured my love of science. To my father, coach, foreman, tech support, and #1 fan, Paul, who kindled my engineering spirit. To my cats, Babe and Bunyan, who stayed up with me all those long nights. Thank you for your endless support.

Table of Contents

List of Tables

List of Figures

List of Codes

List of Acronyms

ESS Energy Storage System.

HALEU High-Assay Low-Enriched Uranium.

HPC High Performance Computing.

IES Integrated Energy System.

INL Idaho National Laboratory.

LTI Linear Time-Invariant.

LWR Light Water Reactor.

MSNB Molten Salt Nuclear Battery.

MSR Molten Salt Reactor.

MSRE Molten Salt Reactor Experiment.

NPP Nuclear Power Plant.

NRC Nuclear Regulatory Commission.

NREL National Renewable Energy Laboratory.

ODE Ordinary Differential Equation.

ORNL Oak Ridge National Laboratory.

PID Proportional-Integral-Derivative.

Statement of Contribution

Chapter 3 is a multi-authored article that was submitted to and accepted by Nuclear Engineering and Design [1]. The author of this thesis was the primary author of the article, writing the original draft manuscript and conceptualizing the methodology. The co-authors offered the following valuable collaborative efforts in support of publication of the work:

- **Haiyan Zhao** Revisions, support and guidance in the development of the xenon stripping model.
- **R.A. Borrelli** Writing, revisions, and response to reviewer comments, assistance in conceptualization, support in development and verification of the numerical solver, case study selection
- **Michael G. McKellar** Revisions, supervision, case study selection

I am grateful for their contributions.

Chapter 1: Introduction

The world is working to move away from fossil fuel as its main energy source. National Renewable Energy Laboratory (NREL) has partnered with over 700 organizations, including large manufacturing companies, to decarbonize supply chains [2]. Nuclear power has been well established as an alternative for base-load electrical generation with 93 reactors in the United States and 440 globally which each generate on the order of 1 GWe, but there remains a need for smaller reactors to be deployed in more dynamic applications such as small remote grids, manufacturing, and power-peaking [3, 4]. These small energy utilizers could turn to microreactors to fill their needs; to make this a reality, each microreactor requires a robust control system that is capable of load following. The primary goal of this thesis is to model the behavior of the Molten Salt Nuclear Battery (MSNB), a natural circulation molten salt microreactor concept, so a preliminary controller design can be developed and tested.

1.1 Molten Salt Nuclear Battery

The MSNB is a self contained design for a liquid fueled molten salt microreactor [5, 6]. It is fueled by an inorganic form of uranium, UF_4, dissolved in a coolant salt such as FLiNaK (a eutectic mixture of three alkali fluorides) or FLiBe (a mixture of LiF and BeF_2) [7]. Heat is generated in the core by fission and is rejected in an integrated heat exchanger (Fig. 1.1). Criticality is manipulated using axial control drums, which may be rotated to aim either a neutron reflecting material or a neutron absorbing material towards the core (Fig. 1.2).

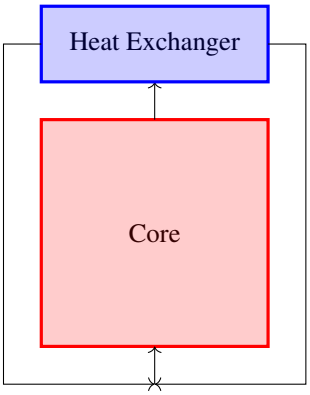


Figure 1.1: Simplified schematic drawing of an MSNB. Heat is generated in the core by fission, is transported by natural circulation of the coolant/fuel salt, and rejected to a secondary working fluid in the heat exchanger before returning to the inlet plenum through the downcomer.

Though it operates on the same physical principles, the MSNB is designed to fit a different role than what is served by our existing Nuclear Power Plant (NPP) fleet. At 10 MWth and 700 ^{o}C with a large volumetric heat capacitance, it aims to be widely deployable with a focus on manufacturing over construction. The liquid fuel also provides operators with two benefits that make the small scale practical: 1) high specific power potential; and 2) deep burn-up;

With the fuel dissolved and dispersed, fission heat is rapidly deposited directly into the primary coolant. This is in contrast to Light Water Reactors (LWRs), where the cylindrical geometry and solid state of the fuel limit the achievable specific power to a level where the fission heat can be effectively conducted to the coolant. The stationary cladded fuel is also susceptible to negative effects from spatial variation in neutron flux. The flowing and electrolytic nature of the MSNB lead to flatter temperature and power profiles and reduced concentration gradients of certain nuclides, which overall allows a larger fraction of the fuel to be fissioned.

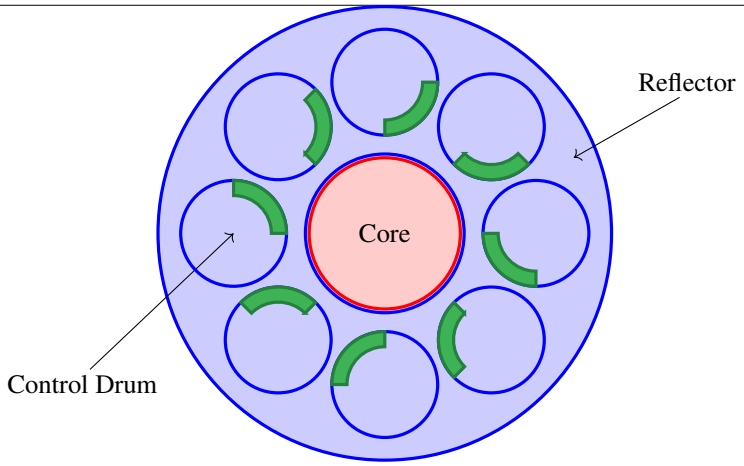


Figure 1.2: Simplified cross-section drawing of an MSNB. The core is surrounded by a reflector in which control drums are embedded. These drums are rotated to aim a neutron absorbing material either towards or away from the core to manipulate criticality.

1.2 Microreactors

Microreactors, as the name suggests, are small nuclear reactors outputting less than 50 MWth which are designed to be fully assembled when shipped, rather than constructed on site. This is a hot area of research in the private sector as companies are working to capitalize on the growing need for clean and dependable small scale electrical generation [6]. They aim not to replace the utility scale NPPs which handle base-load electrical generation, but the diesel or natural gas engines that are found at countless manufacturing facilities, peaking stations, military bases, islands, and other locations where on-site generation is the primary or only source of power.

The goal is to deliver a prefabricated microreactor to a site, integrate it to the necessary power cycles and process heat applications, and meet the needs of the site for a long period of time - up to a decade - without the need for refueling or significant maintenance. Among the biggest challenges in implementing microreactors are the types of transients that these applications often require. Engines handle these quite well, simply adjusting the flow rates of fuel and combustion air. Nuclear reactor load following is a bit more complicated, as the reactor must be made supercritical to ramp up power or subcritical to decrease power. Neutronics modeling of the reactor's criticality control & actuation system and thermal hydraulic simulation of reactivity feedback mechanisms are needed to predict the dynamic behavior of the microreactor so a 'reactor-following-turbine' controller can be tuned [8, Ch. 8].

1.3 Molten Salt Reactors

Molten salts are highly desirable heat transfer fluids in high temperature applications due to their excellent thermophysical properties [9]. Salt mixtures have been developed to have very wide liquid temperature ranges (i.e. low melting point and high vaporization point). They also have high volumetric heat capacities compared to other high temperature coolants (which tend to be gaseous), and are able to operate at or slightly above ambient pressure. These properties combine to make molten salts excellent choices in heat transfer and thermal storage applications. Furthermore, they are extremely strong electrolytes which can be useful as solvents in certain chemical reactions including a pyrometallurgical method for reprocessing spent nuclear fuel [10].

Molten Salt Reactors (MSRs) are a family of nuclear reactor in which a fuel salt (containing fissile and/or fertile nuclides) is dissolved in a coolant salt [7]. The concept was proven by the Molten Salt Reactor Experiment (MSRE) at Oak Ridge National Laboratory (ORNL) in the 1960s [11]. It has yet to take off beyond the research reactor sector, but it has re-emerged as a Gen-IV reactor concept, with a team at the Shanghai Institute of Applied Physics gaining approval to operate a now fully constructed thorium breeding MSR [12]. Some of the benefits of MSRs over more conventional LWRs include:

- Higher operating temperatures allow for use in applications requiring high-grade process heat, and yield higher thermal efficiency [7];
- Lower operating pressures contribute to inherent safety [9];
- The ability to burn minor actinides supports the goal of reducing global stockpiles of high-level waste [9];
- There is no concern of core melt-down as the reactor is designed for liquid fuel;
- The liquid state homogenizes nuclides throughout the core, which minimizes burn-up gradient to produce a flatter temperature and power profile within the core [13, Ch. 3]. The flowing nature also allows for online reprocessing, removing fission products and poisons during operation;

They also carry some demerits:

- Natural circulation of the fuel introduces an additional feedback mechanism that may introduce destabilization of autonomous load following of certain power demand transients [14];
- Molten salts are very corrosive, often requiring more expensive materials [15];
- The chemistry of the coolant (not only the fuel) is constantly changing due to fission, transmutation, and impurities from corrosion;
- Lithium is commonplace in molten salts, so tritium production is unavoidable, being formed by both the $^6Li(n,\alpha)^3H$ (Rxn. 1.1) and $^7Li(n,n\alpha)^3H$ reaction . Off-gas systems need to be robust to handle tritium as well as radionuclide noble gasses, halides, and inter-halides [16];

$$^6Li + n \rightarrow {}^3H + \alpha \qquad \text{(Rxn. 1.1)}$$

1.4 Scope

As a developing design, work has been done on neutronics [6], thermal-hydraulics and autonomous load following [5], and corrosion concerns [17] relevant to the MSNB. However, until now, little to no work has been done on the control system. First and foremost, this work details a multiphysics characterization of the MSNB required to design a feedback controller capable of matching the core power generation to the secondary power demand. In addition to the main control mode of following power transients during normal operation, specific discussion is centered around more dynamic time periods, namely: 1) initial start-up; 2) shutdown, both planned and emergency; and 3) restart;

This work is focused on the operational control system. There are a number of related systems that will also need to be considered, such as: 1) *in-situ* melting of salt immediately following delivery and installation; 2) neutron seed for initial start-up; and 3) decay heat removal for both planned and emergency shut-down; These are important systems but are out of scope for this project.

Chapter 2: Process Control Engineering

There are two main goals in process control engineering: 1) Reference tracking, where a process variable is matched to a set-point which may be changed over time; and 2) Disturbance rejection, where the process variable is held to the set-point despite outside influence upsetting it; This is usually achieved by a controller which measures the system inputs and outputs using a sensor/transmitter and controls the process variable by manipulating an actuator.

2.1 Feedback

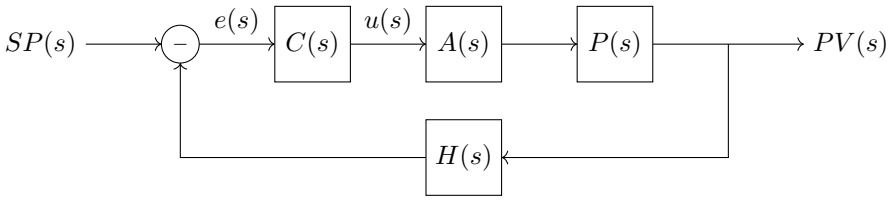


Figure 2.1: Feedback control loop. The process-variable (PV) is measured by the transducer (H) and compared to the set-point (SP). The controller (C) uses the actuator (A) to control the process (P) based on the error (e).

The most common type of controller is a feedback controller. Fig. 2.1 shows a simple feedback control loop with an output sensor/transmitter (*i.e.* transducer), controller, and actuator working together to control a process. The controller takes action (u) based on the 'error' (e) between the set-point (SP) and process-variable (PV) (Eqn. 2.1). These systems are typically modeled using transfer functions in the Laplace (or 's') domain.

$$e(t) = PV(t) - SP(t) \tag{Eqn. 2.1}$$

The action, or controller output (u) is often determined by a Proportional-Integral-Derivative (PID) equation (Eqn. 2.2), which considers the instantaneous, cumulative, and predictive error in determining the proper actuation [18, Ch. 5]. This equation has three terms:

1) Proportional control term. The control output is manipulated in proportion to the error defined by the proportional gain constant (K_P). A high gain yields an aggressive controller that is prone to overshooting the set-point, while a low gain may result in steady-state offset.

2) Integral control term, which considers the historical cumulative error (calculated by taking the time integral of the error) in an effort to eliminate steady-state offset that a P-Only controller may exhibit. As the process variable settles around the set-point, the cumulative error approaches a constant value and the effect of the integral controller diminishes.

3) Derivative control term, which estimates the time rate of change of the error to dampen overshoot. This mechanism, sometimes referred to as anticipatory control, slightly reduces the proportional response to the error when the error is changing rapidly. This results in reducing the peak overshoot. A well tuned anticipatory gain can allow a more aggressive proportional gain to be used without the large overshoot.

$$u(t) = \underbrace{K_P e(t)}_{\text{Proportional}} + \underbrace{K_I \int_0^t e(t)dt}_{\text{Integral}} + \underbrace{K_D \frac{de(t)}{dt}}_{\text{Derivative}} \tag{Eqn. 2.2}$$

Instead of using three different gain constants, it is common for controllers to be tuned in terms of a single controller gain (K_C) plus two time constants: 1) The integral time constant (τ_I); and 2) The derivative time constant (τ_D); In this case, Eqn. 2.2 is rewritten as:

$$u(t) = K_C \left(e(t) + \tau_I^{-1} \int_0^t e(t)dt + \tau_D \frac{de(t)}{dt} \right) \quad \text{(Eqn. 2.3)}$$

2.2 Feedforward

The term 'Feedforward' can be used to refer to any element in the control block diagram that exists outside of the feedback loop. In process control, feedforward controllers are almost always implemented alongside, not instead of feedback controllers because a standalone feedforward controller is not guaranteed to reach the set-point.

2.2.1 Disturbance Feedforward

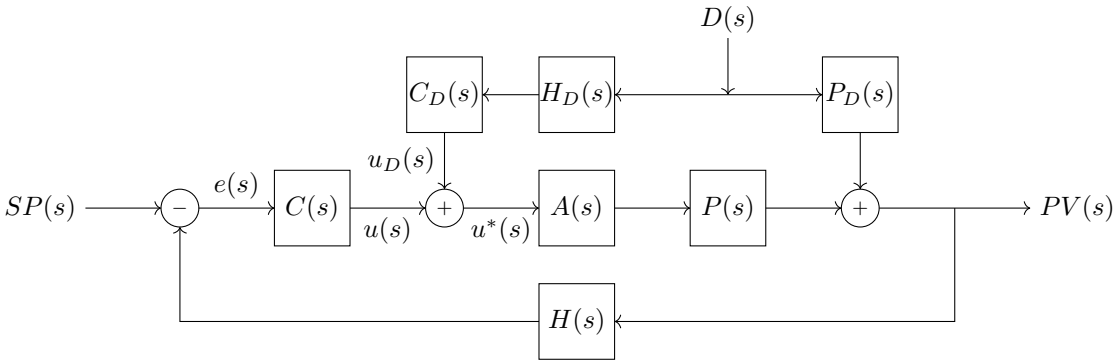


Figure 2.2: Feedback control loop with disturbance feedforward. It is identical to Fig. 2.1 with the addition of a disturbance (D) which effects the process variable (PV) according to the disturbance dynamics (P_D), and is measured by the disturbance transducer (H_D). The signal from H_D is sent to the disturbance feedforward controller (C_D) who's output (u_D) is combined with (u) to form the total control output (u^*).

In many processes, the process variable is effected by phenomena other than the actuator. These other phenomena are defined as disturbances. A well-tuned feedback controller is capable of disturbance rejection, but only after the disturbance causes error. In some cases, a disturbance feedforward controller may be added to the feedback controller to cause the actuator to counteract the effect of the disturbance before it occurs [18, Ch. 10]. Fig. 2.2 shows a feedback control loop with the addition of a disturbance feedforward controller.

The most prevalent disturbances that would effect the power output of the core of a MSNB are the temperature reactivity feedback effect common to all nuclear reactors and the flow reactivity specific to natural circulation driven liquid fueled MSRs [14].

Temperature reactivity feedback is dominated by Doppler broadening, where the radiative capture resonance peaks[1] of nuclides such as ^{238}U are depressed to cover a wider epithermal neutron spectrum [19, Ch. 2]. Higher temperature atoms by definition have larger velocity in random direction. This means that for a given neutron energy, a wider range of relative (*e.g.* center-of-mass frame of reference) energies are observed based on the velocity component of the target nucleus in the direction of neutron incident [8, Ch. 7]. Neutrons with proper energy near a resonance energy are then more likely to have relative energy that resonates. This results in a

[1] Neutrons with energy that matches a quantum excitement energy of an incident nucleus are more likely to be captured.

lower resonance escape probability (*i.e.* , a larger percentage of neutrons are absorbed while slowing down) and a negative correlation between fuel temperature and fuel reactivity[19, Ch. 3]. Liquid fuels also have a high thermal-expansion coefficient, so higher core temperatures lead to lower heavy metal density and lower macroscopic fission cross-section in the core [6]. With less fission targets in the core, neutrons must travel further to have the same number of chances to cause a subsequent fission, increasing the odds of being captured by a non-fissile nuclide and impeding the fission chain reaction.

Flow reactivity is driven by the advection of delayed neutron precursors [8, Ch. 3]. Not all fission neutrons are released promptly; sometimes an unstable nuclide which eventually decays by neutron emission produced instead. These unstable nuclides are called delayed neutron precursors and have half-lives ranging from less than a second to over a minute [20, Ch. 7]. An example is given by Rxn. 2.1

$$^{87}Br \xrightarrow[56sec]{\beta^-} {}^{87}Kr^* \rightarrow {}^{86}Kr + n \qquad \text{(Rxn. 2.1)}$$

Since the fuel in a MSNB is flowing, some delayed neutron precursors will leave the core by advection before the neutron is emitted in a much less reactive part of the reactor. When the temperature differential between the core and primary heat exchanger is increased, the natural circulation flow rate increases too. This decreases the likelihood of delayed neutrons being emitted in the core, and negatively impacts core reactivity. Helix devices meant to elongate the in-core flow path may minimize delayed neutron losses [5]. This is an important consideration in flowing fuel reactor designs, as significant losses of the longer lived precursors greatly shortens the effective neutron generation time, making the reactor more difficult to control [19, Ch. 6].

Disturbance feedforward will not be utilized in the design of the controller outlined in this work. When the outlet temperature of the heat exchanger is decreased, it takes time for the cooler salt to reach the core. The disturbance transport delay is on the order of minutes. Contrastly, Doppler Broadening has a nearly instantaneous effect, so disturbance dynamics are on the order of milliseconds, governed by the mean neutron lifetime [20, Ch. 7]. The effect of control actuation are similarly prompt. Even with a temperature sensor just at the inlet of the core it would be nearly impossible to reliably predict the exact moment that control reactivity would be need to be inserted to counteract the temperature reactivity change.

2.2.2 Pre-Filter

Pre-filters are another type of feedforward mechanism common in control systems. They are typically first-order transfer functions such as Eqn. 2.4 used to improve the performance of the associated feedback controller (as depicted by Fig. 2.4) by 'slowing down' the rate of change of the set-point. The gain for a pre-filter is always unity because they are meant to only reshape the input, not resize it. The time-constant (τ_F) describes how quickly the output equilibrates with the input.

$$F(s) = \frac{1}{\tau_F s + 1} \qquad \text{(Eqn. 2.4)}$$

By resisting instantaneous set-point changes, or step functions, the pre-filter minimizes the instantaneous error during transients and minimizes overshoot by reducing over-actuation or actuator saturation. This is particularly useful in a control system such as the one designed in this report where the set-point is coupled to some other value. In this case, the core power generation needs to match the demand of the secondary system, *e.g.* power cycle. This method allows the secondary system to immediately change to its necessary value while giving the reactor core time to safely and effectively respond.

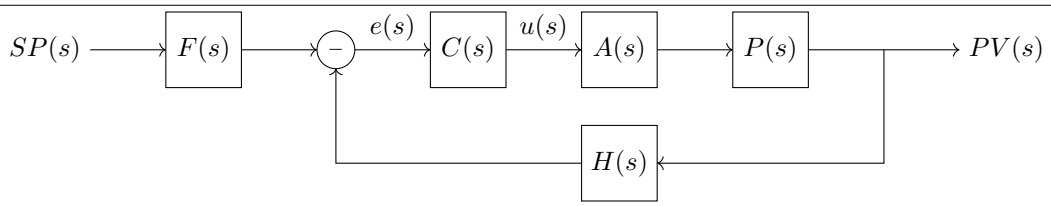
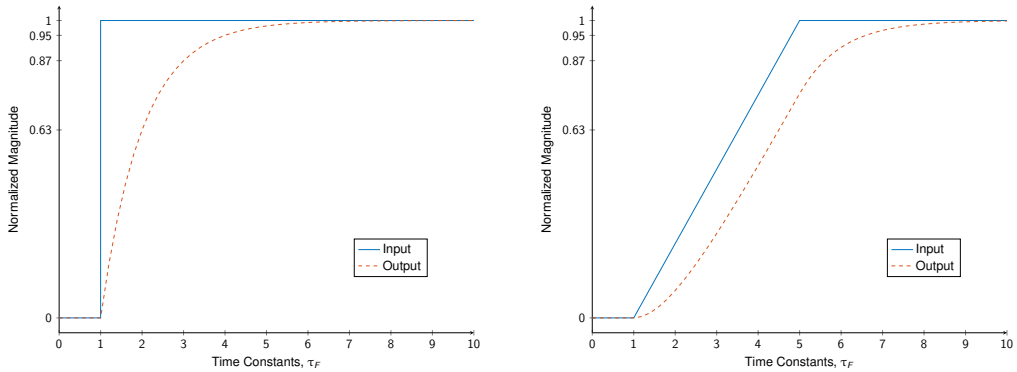


Figure 2.3: Feedback control loop with pre-filter. It is identical to Fig. 2.1 with the addition of the pre-filter (F) which reshapes changes to the set-point (SP) before calculating the error (e).

The control loop for a feedback system with a pre-filter is included as Fig. 2.3. The area between the solid blue and dashed orange curves corresponds the net amount of energy expelled from the molten salt loop following the transient. This manifests as a drop in the average salt temperature. If the reactor begins to operate near thermophysical boundaries, it may be necessary for the core to over-produce in order to bring the temperature back up. An alternative approach could be to use a slightly under-damped second-order pre-filter so the core power generation briefly overshoots the heat exchanger power consumption to balance the energy inequality.



(a) Step-function

(b) Ramp-function

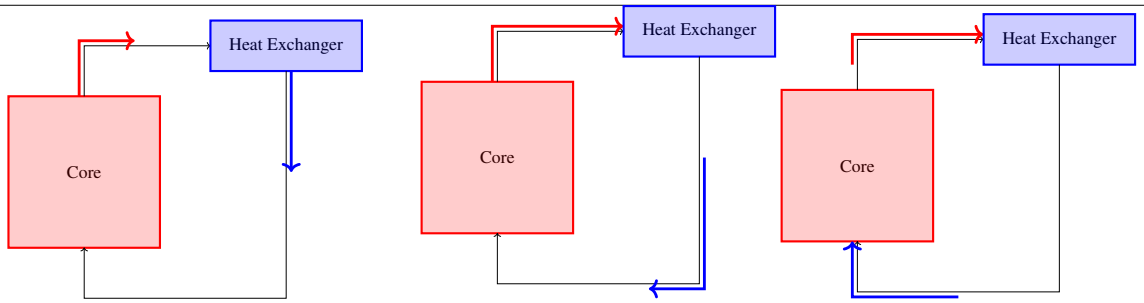
Figure 2.4: Pre-filter on step-function and ramp-function. When the pre-filter acts on a step-function, it follows an exponential curve, reaching 63.2% of the magnitude of the step in 1 time constant, 86.5% in 2 time constants, 95.0% in 3 time constants and so on. The ramp-function exhibits similar but more complicated dynamics due to the changing input.

2.3 Transport Delay Problem

A good place to start in the design of this controller is discussing the dynamics associated with anticipated transients. The prominent factors here are the natural circulation flow mode and the transport delay separating the heat exchanger and core. A common transient for the MSNB is a step-increase in power demand to a steady-state critical MSNB where the core power generation set-point is instantaneously equal to the heat exchanger power consumption. For this thought experiment, consider an ideal controller which produces rapid load following with minimal overshoot. The behavior describe below is illustrated in Fig. 2.5

2.3.1 Immediate Response

The heat-exchanger immediately rejects more thermal energy to the secondary loop and the core immediately generates more power. The core outlet temperature increases quite sharply, but since there is a transport delay between the core outlet and heat exchanger inlet (θ_{riser}), this hot salt does not instantaneously reach the heat



(a) Immediate Response (b) Heat Exchanger Perturbation (c) Core Perturbations

Figure 2.5: Simplified schematic drawings of the transport delay problem in a natural circulation MSNB. These figures illustrate how an ideal controller would create hot (shown in red) and cold (shown in blue) 'edges' in the temperature profile of the primary loop. These edges result in periodic disturbances in the form of sharp temperature reactivity insertions to the core, and are caused by three key events: a) The instantaneous power changes upsetting the outlet temperature of both the core and heat exchanger; b) The first hot salt reaching the heat exchanger θ_{riser} after the power change; and c) The first cold salt reaching the core $\theta_{downcomer}$ after the power change; It would take a very long time for these periodic disturbances to dampen with a controller that very quickly counteracts reactivity feedback.

exchanger, and the heat exchanger outlet temperature drops sharply. Thus, the heat exchanger outlet temperature also drops.

As these hot and cold regions propagate and grow, the natural circulation driving force increases which results in a negative flow reactivity insertion. This is a gradual disturbance which the ideal controller can effectively reject by a counteracting insertion of positive control reactivity.

2.3.2 Heat Exchanger Perturbation

Following the response to the initial step-increase, the first notable event occurs when the hot region in the riser reaches the heat exchanger. This produces a hot 'edge' in the downcomer temperature profile that lags the cold edge by approximately θ_{riser}, and again disturbs the core through a change in flow reactivity.

2.3.3 Core Perturbations

The next event occurs when the cold edge reaches the core inlet, $\theta_{downcomer}$ after the step-increase, causing a rapid insertion of positive temperature reactivity which must be rejected by the controller. θ_{riser} later, the hot edge inserts negative temperature reactivity. Each of these responses cause subsequent temperature edges which rise to the heat exchanger and continue through the system. It is apparent that the 'ideal' controller actually inhibits the ability for the reactor to return to steady-state following a transient, instead prolonging both flow reactivity and temperature reactivity oscillations.

A pre-filter that reshapes the core set-point will make the initial hot edge more gradual. Proper tuning of the pre-filter time-constant will allow the reactivity oscillations to decay more quickly. Previous work has shown that the passive feedback mechanisms (temperature and flow reactivity) are capable of autonomous load following for small transients, though not at the level of performance that may be required in certain applications [14]. Still, this provides the opportunity to minimize fine and rapid actuation while dampening oscillations by using a dead-band or low-pass filter; the 'ringing out' of minor perturbations could be left to the passive feedback mechanisms after the active feedback controls the bulk of the power change.

2.4 Time-Variance and Non-Linearity

In control systems, it is typically preferred to work with Linear Time-Invariant (LTI) systems. There are a number of non-linearities and time variances at play in the control of the MSNB which must be handled:

- The Control Drum Angle vs. Reactivity Curve, which describes the relationship between control actuation and system response, is not linear, but sinusoidal [6]. Over small changes to the control drum angle, the curve may be linearized using Taylor Series approximation [18, Ch. 2];
- The core reactivity decreases over the course of months and years due to the depletion of fissile nuclides [20, Ch. 7]. This means that the bias-point (or unity-point) of the control drums will drift to provide less negative control reactivity. The bias-point, controller gain, and perhaps time constants will need to change with time. These parameters will be gain-scheduled according to the core's burn-up level, similarly to how autopilot systems for high-altitude aircraft account for the different air properties and mach-number at different altitudes [21];
- The control drums manipulate the criticality of the core, making it supercritical to increase the power, and subcritical to decrease. This is a highly time dependant exponential control mechanism. The derivative control time constant will need to be carefully tuned or even excluded to minimize the likelihood of significant overshoot or instability following a power transient;

In addition to the relatively slow time variance of fissile fuel depletion during steady-state critical operation, there are specific times in a MSNB's expected operational life-cycle that exhibit a higher degree of time variance: 1) Start-up; 2) Shut-down; and 3) Re-start.

2.4.1 Start-up

Prior to start-up, it is very likely that the molten salt fuel/coolant mixture will need to be thawed. Initial start-up will also require a neutron source as a seed for the fission chain reaction [19, Ch. 2]. Each of these systems are important considerations for the design of the MSNB and warrant further investigation, but are out of scope for this work and are briefly discussed in Chapter 7.2. The key consideration regarding the start-up control system is the build-up of fission product neutron poisons in the first hours and days of operation.

Xenon Poisoning

^{135}Xe is the strongest known neutron poison, with a microscopic neutron capture cross-section of 2.65 Mb [20, Table II.2]. The concentration of ^{135}Xe is described by a system of differential equations which quantifies the generation, consumption, and decay of itself and its beta-precursor ^{135}I :

$$\frac{dI}{dt} = \underbrace{\gamma_I \Sigma_f^F \phi(t)}_{\text{Fission Yield}} - \underbrace{\lambda_I I(t)}_{\text{Beta Decay}} \tag{Eqn. 2.5}$$

$$\frac{dXe}{dt} = \underbrace{\gamma_{Xe} \Sigma_f^F \phi(t)}_{\text{Fission Yield}} + \underbrace{\lambda_I I(t)}_{\text{Precursor Decay}} - \underbrace{\lambda_{Xe} Xe(t)}_{\text{Beta Decay}} - \underbrace{Xe(t) \sigma_a^{Xe} \phi(t)}_{\text{Radiative Capture}} \tag{Eqn. 2.6}$$

The dynamics of this system are discussed in detail in Chapter 3 [1]. To summarize, each nuclide builds-up in the reactor until the formation terms equilibrate with the removal terms, beta decay and radiative capture. At equilibrium, ^{135}Xe can contribute over 2000 pcm of negative poison reactivity to the core. This causes a time variance that must be understood and accounted for in the design of the controller.

Samarium Poisoning

The second most important neutron poison to consider after ^{135}Xe is ^{149}Sm . ^{149}Pm is formed by fission and decays to ^{149}Sm , which is stable, and has a large neutron capture cross-section [20, Ch. 7]. Eqn. 2.7 looks identical to Eqn. 2.5, but Eqn. 2.8 is simpler than Eqn. 2.6 as ^{149}Sm is stable and is not a direct fission product.

$$\frac{dPm}{dt} = \underbrace{\gamma_{Pm}\Sigma_f^F\phi(t)}_{\text{Fission Yield}} - \underbrace{\lambda_{Pm}Pm(t)}_{\text{Beta Decay}} \tag{Eqn. 2.7}$$

$$\frac{dSm}{dt} = \underbrace{\lambda_{Pm}Pm(t)}_{\text{Precursor Decay}} - \underbrace{Sm(t)\sigma_a^{Sm}\phi(t)}_{\text{Radiative Capture}} \tag{Eqn. 2.8}$$

This system too reaches an equilibrium which may be found algebraically where the time derivatives are null. Interestingly Eqn. 2.10 is independent of the neutron flux in the core. Because it is stable, it is only removed by the same neutron flux that forms it, and contributes a relatively constant 442 pcm of poison reactivity.

$$Pm_\infty(\phi) = \frac{\gamma_{Pm}\Sigma_f^F}{\lambda_{Pm}}\phi \tag{Eqn. 2.9}$$

$$Sm_\infty(\phi) = \frac{\gamma_{Pm}\Sigma_f^F}{\sigma_a^{Sm}} \tag{Eqn. 2.10}$$

2.4.2 Shut-down

Planned shut-down, either for scheduled maintenance or due to a lack of demand, is quite simple. The heat exchanger is brought to very low power, and the core set-point gradually follows. Careful tuning of the pre-filter time constant can alleviate potential thermal hydraulic (*e.g.* stagnation or reverse-flow) and thermophysical (*e.g.* salt precipitation, freezing, or vaporization) concerns [14]. Whether the reactor is shut down for planned purposes or in an emergency, a decay heat removal system is needed to keep the salt from vaporizing [13]. The emergency shut-down (SCRAM) system and decay heat removal systems are briefly discussed in Chapter 7.2.

2.4.3 Re-start

Compound dynamics in the ^{135}Xe decay chain result in the generation of a large poison reactivity in the hours following a shut-down. This can make restarting the reactor difficult or impossible for a period of approximately two days.

The duration of the xenon spike may be shortened by adding an additional negative term to Eqn. 2.6. This is not physically realizable in cladded solid fuel reactors but the concept of xenon stripping to do just this is studied numerically in Chapter 3 [1]. ^{149}Sm is stable and strongly dissolved in the molten salt, so super-equilibrium samarium is always 'burnt off' during restart [20, Ch. 7].

Chapter 3: Thermodynamic Analysis on Xenon Stripping

Xenon-135 is the strongest known neutron poison; its precursor chain is prominent in the set of uranium-235 fission products. Compound dynamics in the decay chain result in a large ^{135}Xe concentration spike following shut-down, which can make restarting the reactor difficult or impossible for a period of days. This may be the single greatest hurdle inhibiting nuclear power from breaking into power peaking and remote grid applications. In conventional solid fueled reactors, operators must wait for the super-equilibrium xenon (or ^{135}Xe in concentration above the full-power equilibrium level) to decay before restarting. Molten Salt Reactors (MSRs), which have their fissile nuclides dissolved in a liquid salt, present the opportunity for advective removal of xenon. A stream of helium gas may strip dissolved and entrained xenon such that the core returns to its power dependent equilibrium ^{135}Xe concentration much more quickly than by beta-decay alone. This would allow the reactor to restart sooner. This study investigates the feasibility of restarting a MSR more quickly than traditionally viable. We derived a numerical solver and a single stage equilibrium model of a 2-phase mass transport unit operation to this end. Results show that the thermodynamics of xenon stripping from molten salt fuel are highly favorable, with a very small amount of helium being theoretically required for practical control of the ^{135}Xe decay chain after shut-down. Additionally, fuel lifetime could be extended by exchanging poison reactivity at end-of-life for deeper burn-up; this would require a relatively small operational cost of helium. Further investigation into the kinetics and unit operation design is warranted by the favorable thermodynamic results [1].

3.1 Introduction

The concentration of Xenon-135 in nuclear reactor fuel has a large influence on core reactivity. Its effect is most prevalent after a reactor is shut-down. When the neutron flux is cut-off, the poison is no longer formed from fission reactions, nor removed by neutron absorption. Iodine-135 continues to decay, however, and acts as a residual source of ^{135}Xe . For the first several hours after the shutdown, ^{135}Xe is formed by precursor decay faster than it decays. This results in a spike in the fuel ^{135}Xe concentration. Nuclear reactors are not typically designed with enough excess reactivity to overcome this 'iodine pit', and must wait for the excess xenon to decay. Even if the control system does have the ability to restart during the xenon peak, it may be hazardous to do so, and is against regulation [22, 23].

3.1.1 Motivation

At utility scale base-load power plants, shut-down often precedes maintenance, where the reactor needs to remain off for this time period regardless of the iodine pit. Microreactors may not have this luxury [24]. These are small reactors often considered to fit well into microgrids [25] and Integrated Energy Systems (IESs) [26]. Such systems tend to rely on batteries and thermal storage for fast demand response, which allows the microreactor to operate on a 'nuclear island'. These Energy Storage Systems (ESSs) come with a significant additional footprint and economic cost. Cost is already a large factor hampering the defossilization of microgrids and power peaking. Eliminating the need for an ESS by building a more dynamic nuclear reactor system would increase the number of applications where microreactors are economically viable.

The relative ease of fluid phase separation engineering gives molten salt microreactors an advantage over their solid fueled counterparts [5, 6]; negative poison reactivity may be expelled from the core rather than counteracted with a huge amount of positive control reactivity [27]. The application of transport phenomenon to manipulate a macroscopic equilibrium is the essence of the unit-operation discipline of chemical engineering. Implementing this concept on an MSR contributes to the goal of studying how a certain Generation-IV reactor design lends itself to a specific function. Many companies are competing on the frontier of small nuclear reactor development,

and several of them are using molten salts. They may be able to achieve significant market penetration in highly dynamic environments and non-traditional markets by accelerating ^{135}Xe removal. Furthermore, the ability to remove poison from the core without refueling offers the opportunity to achieve deeper burn-up and increased fuel longevity, which supports the goal of building decade scale microreactors.

3.1.2 Goals

This paper is a high level analysis investigating the thermodynamics of xenon stripping from molten salt by: (1) Deriving a model of an equilibrium xenon scrubber based on Henry's law; (2) Incorporating this model into a numerical solver of the Bateman equations for ^{135}I and ^{135}Xe in a MSR; and (3) Analyzing the reactor's response to power changes and xenon stripping;

During a transient, the ^{135}Xe concentration is determined with a low computation time, so parameters can be adjusted iteratively until the set of inputs that yield a desired result can be identified. Research goals based on this methodology are to: (1) Benchmark the numerical solver against the characteristic fixed points of known analytical solutions as well as simplified analytical solutions; and (2) Provide an order of magnitude estimate of the flow rate of helium required to strip xenon at the rate required to produce a desired outcome, such as the time since a reactor shutdown that it can be restarted.

3.1.3 Previous Work

Oak Ridge National Laboratory (ORNL) has studied a method of stripping xenon out of molten salt fuels during operation [28], in a way depicted by Figure 3.1. Helium bubbles are injected into the liquid fuel outside of the core and flow with the fuel before the phases are separated and a poison-poor stream returns to the core. This was primarily focused on the kinetics of a sparge tube stripper, and proposed an experiment to determine the mass transfer coefficients of dissolved xenon gas into fine helium bubbles. Such data is instrumental in the design of a mass transport unit operation, but unfortunately records of the experiment being conducted cannot be found. We seek to determine whether a return to the proposed experiment would be productive by conducting a thermodynamic analysis on the equilibrium system. More detailed design of such a stripper would be warranted if the thermodynamic analysis yields promising results.

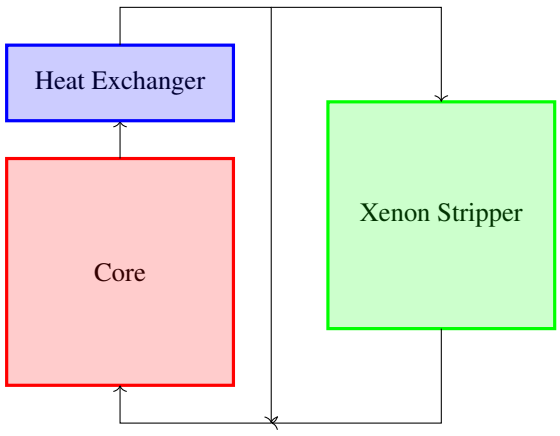


Figure 3.1: Block Flow Diagram - Molten Salt Microreactor with Xenon Stripping Module. The fuel/coolant salt mixture is heated by fission in the core, rises to the heat exchanger where it gives up heat to the secondary coolant and sinks by natural circulation.

It is worth noting the difference between the stripping operation described in this paper and off-gas management systems that all MSRs require [29]. Off-gas systems remove fission gases that naturally make their way into the head-spaces of the primary coolant loop of an MSR. Other work has concluded that a large fraction of fission gasses, including ^{135}Xe , fail to transport to the head-space, remaining dissolved due to poor liquid-gas interface, entrained due to the lack of nucleation sites, or flowing in circulating voids due to surface tension [30]. Stripping greatly improves the kinetics of fission gas removal, and is more akin to fuel reprocessing, while the off-gas management system is more akin to the cladding gap in solid fuel designs.

While the present work is centered around removing gaseous poisons from MSR fuel, previous researchers have expressed interest in doing the opposite, and dissolving gaseous poisons such as boron-trifluoride into the melt for reactivity control [31]. The transport delay of such actuation is likely to be on the order of seconds/minutes, while the effective neutron lifetime, which characterizes the neutron chain reaction, is on the order of tens/hundreds of milliseconds [19, Ch. 1]. As such, soluble boron is not a suitable means of actuation for set-point tracking and is best reserved for fuel-reactivity shimming over the life of the reactor. Similarly, ^{135}Xe stripping is not practical for operational control and is only being studied for its open loop effects.

3.2 Xenon-135 Decay Chain

One of the goals of this paper is to model xenon poison transients in MSRs. Therefore, it is important to understand the physics of the decay chain, particularly during dynamic time periods such as start-up, power transients, shut-down, and restart. The concentration of xenon-135 at a given time is described by a system of first-order Ordinary Differential Equations (ODEs) which quantify the generation, consumption, and decay of itself and iodine-135:

$$\frac{dI}{dt} = \underbrace{\gamma_I \Sigma_f^F \phi(t)}_{\text{Fission Yield}} - \underbrace{\lambda_I I(t)}_{\text{Beta Decay}} \tag{Eqn. 3.1}$$

$$\frac{dXe}{dt} = \underbrace{\gamma_{Xe} \Sigma_f^F \phi(t)}_{\text{Fission Yield}} + \underbrace{\lambda_I I(t)}_{\text{Precursor Decay}} - \underbrace{\lambda_{Xe} Xe(t)}_{\text{Beta Decay}} - \underbrace{Xe(t) \sigma_a^{Xe} \phi(t)}_{\text{Radiative Capture}} \tag{Eqn. 3.2}$$

Eqn. 3.1 and Eqn. 3.2 can be used to track the concentrations of ^{135}I and ^{135}Xe during steady-state or dynamic operation [20, Ch. 7]. Each nuclide is formed by ^{235}U fission, and removed by beta-decay. Additionally, ^{135}Xe is formed by the beta-decay of ^{135}I and removed by radiative neutron capture. In reality, the decay chain is a bit more complicated than this. ^{135}Te, the direct beta-precursor to ^{135}I is also formed by ^{235}U fission [32]. Its half-life is orders of magnitude shorter than ^{135}I , so it is common to neglect its formation and instead lump its fission yield with ^{135}I . ^{135m}Xe is also formed by ^{135}I beta-decay and is neglected for similar reasons. Neutron flux (ϕ) exists on an energy spectrum, which effects the fuel-fission cross-section (Σ_f^F) and the ^{135}Xe poison cross-section σ_a^{Xe}. The form of Eqn. 3.1 and Eqn. 3.2 in this work uses 1-group cross-sections and fluxes, which requires neutron energy-integration, usually performed using a weighted-finite sum methodology.

3.2.1 Equilibrium Poisoning

Following start-up, the nuclides builds-up in the reactor until the formation terms (i.e. fission yield and precursor decay) equilibrate with the removal terms (beta decay and radiative capture). By setting the time derivatives to zero in Eqn. 3.1 and Eqn. 3.2, the neutron flux dependent equilibrium concentrations may be derived algebraically [20, Ch. 7]:

$$I_\infty(\phi) = \frac{\gamma_I \Sigma_f^F}{\lambda_I}\phi \quad \text{(Eqn. 3.3)}$$

$$Xe_\infty(\phi) = \frac{(\gamma_I + \gamma_{Xe})\Sigma_f^F}{\lambda_{Xe} + \sigma_a^{Xe}\phi}\phi \quad \text{(Eqn. 3.4)}$$

When the power level (and therefore neutron flux) of the reactor is changed, the terms of Eqn. 3.1 and Eqn. 3.2 are no longer balanced, and the nuclides move to a new equilibrium level along a path defined by the system of ODEs. The equilibrium ^{135}Xe can be converted to poison reactivity by writing the one-group neutron balance equation for the explicit contribution of neutron capture by ^{135}Xe and assuming that the reactor is critical in the absence of ^{135}Xe . This is essentially the fraction of fission neutrons in a given generation that are captured by ^{135}Xe [33].

$$\rho_{Xe} = -\frac{\sigma_a^{Xe} Xe_\infty}{\nu \Sigma_f^F} \quad \text{(Eqn. 3.5)}$$

3.2.2 Reactor Shutdown

Table 3.1: Relevant nuclear constants [20]. The ^{135}I fission yield (γ) is the sum of direct and indirect fission. The ^{135}I microscopic neutron capture cross-section (σ) is neglected as it is so small that it is insignificant compared to its own decay rate and the ^{135}Xe cross-section.

	γ (%)	λ (hr^{-1})	σ_a (Mb)
^{135}I	6.39	0.1035	-
^{135}Xe	0.237	0.0753	2.65*

*At 0.025 eV

When the reactor is scrammed, each flux-containing term in the system of ODEs goes to zero, and the concentrations of the two nuclides are described by the Bateman equations [33], which have a readily available solution given some initial condition [20, Ch. 1]:

$$I(t) = I_\infty e^{-\lambda_I t} \quad \text{(Eqn. 3.6)}$$

$$Xe(t) = Xe_\infty e^{-\lambda_{Xe} t} + I_\infty \frac{\lambda_I}{\lambda_I - \lambda_{Xe}}(e^{-\lambda_{Xe} t} - e^{-\lambda_I t}) \quad \text{(Eqn. 3.7)}$$

In this case the initial conditions are the equilibrium levels described by Eqn. 3.3 and Eqn. 3.4. ^{135}I follows simple exponential decay. ^{135}Xe has a longer half-life than ^{135}I , as well as a lower equilibrium concentration (owing to its huge radiative capture cross-section). This results in an inverse response, where the ^{135}Xe concentration initially spikes before its decay rate exceeds its formation. Table 3.1 contains literature values for the constants relevant to the iodine-xenon system. The time between the scram and the peak ^{135}Xe concentration, can be calculated by setting the derivative of Eqn. 3.7 to zero:

$$t_{max} = \frac{ln\left(\frac{\lambda_{Xe}}{\lambda_I^2}\frac{Xe_\infty}{I_\infty}\right)(\lambda_I - \lambda_{Xe}) + \frac{\lambda_{Xe}}{\lambda_I}}{\lambda_{Xe} - \lambda_I} \quad \text{(Eqn. 3.8)}$$

Eqn. 3.8 demonstrates that the time of the xenon peak depends mostly on the decay constants of ^{135}Xe and ^{135}I , with only a weak dependence (logarithmic and buffered) on $^{Xe_\infty}/_{I_\infty}$. By inspection of Eqn. 3.3 and Eqn. 3.4, it can be observed that this equilibrium nuclide ratio is only weakly dependent on the power level of the

reactor before the scram. $\Sigma_f \phi$ cancels out, leaving the contribution of neutron capture to the effective half-life[1] $\sigma_a^{Xe}\phi$ is analogous to the decay constant [20, Ch. 7]. of ^{135}Xe as the only flux containing term that impacts t_{max}

3.2.3 Reactor Restart

When the fission chain reaction is restarted, the neutron flux increases and resumes transmuting ^{135}Xe . This shortens the effective half-life[1] of the poison, accelerating the removal rate which is already high, owing to the elevated concentration. The removal of poisons is akin to positive reactivity insertion [32]. Decreasing poison concentration increases the neutron population and therefore the rate of poison removal. This is a positive feedback system which could escalate and disturb the temperature of downstream systems such as power cycles and process heat applications if control reactivity is not removed to match. As such, regulation and standard operating procedures may restrict or ban the restart of a reactor in a super-equilibrium state, and operators typically must wait until the ^{135}Xe level returns to near the full-power equilibrium level to restart [23].

The duration of the xenon dead-time could be shortened by adding an additional negative term to Eqn. 3.2. This is not physically realizable in our current fleet of cladded solid fuel reactors. Vented solid nuclear fuel elements have been conceptualized for high temperature reactors which have high partial pressure from fission products [34, 35]. Still, this concept only allows for the venting of fission gasses due to positive pressure within the fuel element. Even if a flowing sweep gas were incorporated to further facilitate fission gas removal, a vented rod's ability to manipulate neutron poison concentration within the fuel will be limited by the diffusion of xenon through the fuel matrix [36].

It is however possible to strip a gaseous solvent from a liquid such as MSR fuel. Xenon, being a noble gas, is weakly dissolved in the molten salt, and it is more energetically favorable for it to exist as a gas. A gas bubble cannot form spontaneously and requires a nucleation site to surmount the associated activation energy barrier [37]. This often comes in the form of an impurity that has a trapped bubble attached. When a xenon atom comes into contact with the salt/gas interface, it transports into the bubble. When the bubble grows large enough, it will stop circulating and get caught in the reactor head-space where the off-gas system can remove it.

Instead of waiting for such a chance incident to occur, a bulk gas phase can be bubbled through the salt, which provides interfacial area across which and volume into which the xenon will preferentially transport through an operation called stripping [38, Ch. 10]. [39] has studied the off-gassing of nobel gas fission products from liquid fueled MSR, including the MSRE. This was modeled using flow blocks in SCALE's TRITON sequence by assuming or calculating a removal constant for fission gasses that transport into the off-gas. The present work extends upon this idea by deriving a removal constant composed from optionally time dependent parameters for a single stage equilibrium stripper, and implementing it in an open source solver. For this purpose Eqn. 3.2 is modified with the addition of the stripping term which will be derived in Section 3.3.1 (Eqn. 3.9).

3.3 Methodology

A two pronged methodology is employed to model the xenon-iodine decay chain in an MSR with the addition of xenon stripping. A unique model was derived to describe the thermodynamic equilibrium products of a unit operation that removes dissolved and entrained xenon gas from a molten salt by preferential transport into a bulk helium stream (see Section 3.3.1). This model is combined with the well understood model of the ^{135}Xe system described in Section 3.2 to form Eqn. 3.15, which is the basis for this work and is solved numerically in Section 3.3.2.

[1] Effective half-life quantifies the actual removal rate of a species by aggregating all removal modes, in this case beta-decay and neutron capture. For transmutation,

$$\frac{dXe}{dt} = \underbrace{\gamma_{Xe}\Sigma_f^F\phi(t)}_{\text{Fission Yield}} + \underbrace{\lambda_I I(t)}_{\text{Precursor Decay}} - \underbrace{\lambda_{Xe}Xe(t)}_{\text{Beta Decay}} - \underbrace{Xe(t)\sigma_a^{Xe}\phi(t)}_{\text{Radiative Capture}} - \left(\frac{dXe}{dt}\right)_{strip} \quad \text{(Eqn. 3.9)}$$

3.3.1 Two-phase Equilibrium Mass Transport

The stripping term from Eqn. 3.9 is derived in this section. Dissolved gasses follow Henry's law, at low concentration:

$$C_\ell = Hp \quad \text{(Eqn. 3.10)}$$

This describes a relationship between the concentration of the solute in the liquid phase (C_ℓ) and the partial pressure of the solute in the cover gas (p) when the two phases are in equilibrium [38, Ch. 10]. Fresh cover gas can be supplied to strip the solute into the cover gas. By considering total pressure of the cover gas and assuming an equation of state, Henry's law can be modified to be expressed in terms of gas phase concentration instead of partial pressure, as shown by:

$$C_\ell = H^* C_g \quad \text{(Eqn. 3.11)}$$

An early study conducted at ORNL predicted such a unitless Henry's Law constant (H^*) for the xenon-FLiNaK system on the order of 1×10^{-4} at MSR temperatures [40]. While [40] did not experimentally validate this specific prediction, it did show through experimentation with lighter noble gasses that the formula used to make the prediction was accurate at the order of magnitude level. This level of precision should be refined in future work but is acceptable for this high-level analysis. The solver is and will continue to be open source, allowing future researchers to refine parameters such as H^*, including writing a simple plug-in to allow for potential time-variability in H^* due to burn-up and temperature.

Consider a co-current sparge pipe stripper that uses fresh helium to remove dissolved and entrained xenon from molten salt fuel. The helium does not react and simply provides a bulk gas phase for macroscopic-scale xenon transport. If the interfacial area in the pipe is sufficient, the xenon concentration in the outlet-gas stream will be in equilibrium with that of the liquid outlet. Figure 3.2 is a schematic drawing of such a stripper and shows the xenon concentration at the inlet and outlet of the molten salt and gas streams, as well as the volumetric flow rate of each stream.

The salt enters with ^{135}Xe concentration equal to that of the xenon level in the core, and exits at some lower value. The helium enters with no xenon, and exits with a ^{135}Xe concentration in equilibrium with the salt's outlet concentration. This analysis is based on the horizontal co-current sparge pipe designed at ORNL during the Molten Salt Reactor Experiment (MSRE) [28]. A counter-current design may allow for greater stripping effectiveness by maintaining a more constant (and therefore a larger log-mean average) driving force, but it makes the separation of phases more challenging. The horizontal design also eliminates hydrostatic pressure differential across the unit operation, so a constant volumetric gas-phase flow rate constant across the module can be assumed.

A simple mass balance shows that the concentration in the liquid outlet must be equal to the inlet concentration minus the gas outlet concentration times the gas-to-liquid volumetric flow rate ratio. As shown in Figure 3.2, substituting the definition of C_{He}^{out} into the definition of $C_s^{out}alt$, we obtain:

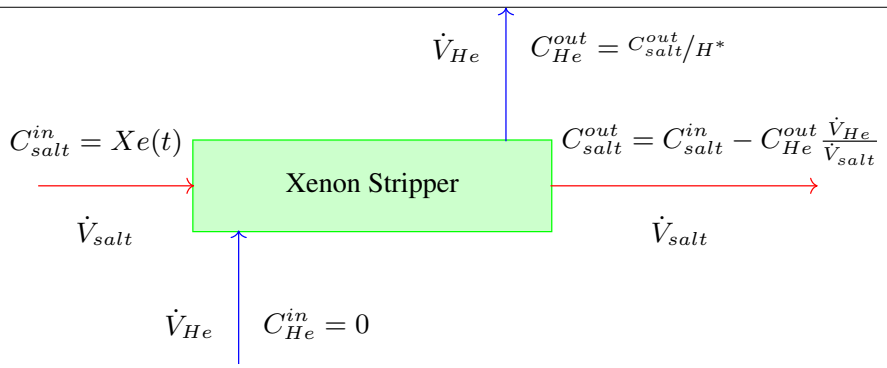


Figure 3.2: Schematic Drawing of Xenon Stripping Module. Fresh helium and molten salt with a given ^{135}Xe concentration enters on the left. Dissolved and entrained xenon preferentially transports from the liquid phase to the gaseous stream. Loaded helium and molten salt with a lower ^{135}Xe concentration exit on the right. The exiting streams are in equilibrium with each other according to Henry's Law.

$$C_{salt}^{out} = C_{salt}^{in}\left(1 + \frac{1}{H^*}\frac{\dot{V}_{He}}{\dot{V}_{salt}}\right)^{-1} \quad \text{(Eqn. 3.12)}$$

The local (e.g. inside of the xenon stripping module) change in xenon-135 concentration can be simplified by substituting the definitions of C_{He}^{out} and C_{salt}^{out} from Figure 3.2 into Eqn. 3.12:

$$dC_{salt} = C_{salt}^{in}\left(H^*\frac{\dot{V}_{salt}}{\dot{V}_{He}} + 1\right)^{-1} \quad \text{(Eqn. 3.13)}$$

Over a certain amount of time (dt), only some fraction (determined by the MSR flow period) of the molten salt will pass through the xenon stripping module. Therefore, $\left({}^{dXe}/_{dt}\right)_{strip}$ appearing in Eqn. 3.9 is given by :

$$\left(\frac{dXe}{dt}\right)_{strip} = \tau_{salt}^{-1} dC_{salt} \quad \text{(Eqn. 3.14)}$$

Eqn. 3.9 can then be re-written as below, where the inverse of $\tau_{salt}(H^*\dot{V}_{salt}/\dot{V}_{He} + 1)$ is analogous to the decay constant for stripping:

$$\frac{dXe}{dt} = \underbrace{\gamma_{Xe}\Sigma_f^F\phi(t)}_{\text{Fission Yield}} + \underbrace{\lambda_I I(t)}_{\text{Precursor Decay}} - \underbrace{\lambda_{Xe}Xe(t)}_{\text{Beta Decay}} - \underbrace{Xe(t)\sigma_a^{Xe}\phi(t)}_{\text{Radiative Capture}} - \underbrace{\frac{Xe(t)}{\tau_{salt}}\left(H^*\frac{\dot{V}_{salt}}{\dot{V}_{He}} + 1\right)^{-1}}_{\text{Stripping}} \quad \text{(Eqn. 3.15)}$$

This derivation neglects the positional variance of xenon-135 concentration resulting from the stripper. This simplification is reasonable when the salt traverses the flow loop multiple times while the xenon stripping module is active; i.e, the flow period is small compared to the restart time.

The stripping term in Eqn. 3.15 is mathematically undefined at zero helium flow rate. It can logically be inferred (and shown by taking the limit of the term as $\dot{V}_{He}$ approaches zero) that such a case corresponds to zero stripping change.

3.3.2 Numerical Solver

A numerical solver was derived in Python 3.10 to track the time dependent ^{135}I and ^{135}Xe concentrations while giving the user control over the neutron flux and helium flow rate throughout the process[2] This allows the user to investigate the change in xenon poisoning after a power change, and the effectiveness of the equilibrium stripper characterized in Section 3.3.1 in shortening the post-scram restart time in an MSR. Euler's forward method is applied, where the nuclide concentrations for the next time step are calculated using the current concentrations and the current rates of change, given by Eqn. 3.1 and Eqn. 3.15:

$$I[t+dt] = I[t] + \frac{dI}{dt}[t] \quad \text{(Eqn. 3.16)}$$

$$Xe[t+dt] = Xe[t] + \frac{dXe}{dt}[t] \quad \text{(Eqn. 3.17)}$$

The solver is object oriented, which makes it modular and allows code to be re-used for both ^{135}I and ^{135}Xe . The class 'Core' contains constants such as the macroscopic fission cross-section (Σ_f^F) and key variables, most notably the neutron flux and helium flow rate. It also contains class methods that can be used to modify each of the aforementioned variables to simulate a scram and activation of the xenon stripping module. The class 'Nuclide' serves as the framework for the simulation. An object containing all physical constants and parent-daughter relationships is instantiated for ^{135}I and ^{135}Xe . It also contains methods to calculate each term of Eqn. 3.1 and Eqn. 3.15 which are combined to track the concentrations using Eqn. 3.16 and Eqn. 3.17.

A time step length (dt) of 1 second was selected to strike a balance between precision and computational intensity. The solver is able to simulate hundreds of hours of nuclide evolution in under a minute on an HP laptop with a 2.3 GHz Intel Core i7 processor and 12 GB of memory, without long time-steps (e.g. on the order of the nuclide's half-life) which would cause local truncation error. This is long enough to allow the ^{135}Xe concentration to equilibrate, and an entire dynamic case study to be conducted.

To conduct a case study, the solver iterates over a time loop, appending the new concentration of ^{135}I and ^{135}Xe at each time step. At any time, the prompt neutron flux and helium flow rate may be independently changed by using class method 'scram' and 'strip'. These two methods are used in conjunction to conduct the experiments discussed in Section 3.4.

Class method 'scram' can optionally call a subroutine (class method delayed_neutron_precursors) that creates a list of delayed neutron flux following shutdown ($\phi^d(t)$) using the six-group approximation and the data in Table 3.2, obtained from [8]:

$$\phi^d(t) = \phi_0 \vec{\beta} \cdot e^{-t\vec{\lambda}} \quad \text{(Eqn. 3.18)}$$

The solver then checks if the list contains values, and removes the zeroth value, assigning it to be the new flux using the 'pop' method. This approach can be used to treat any parameter assumed to be constant in this paper as a time-variable.

[2]The numerical solver is available at https://github.com/sjroot97/NumericalSolverXe135. The version of the solver used to plot Figure 3.5 is included in Appendix A as an example. Users are permitted free use of the solver with the condition that any refinements, advancements, or plug-ins built-on also be shared freely.

Table 3.2: Delayed neutron fraction (β) and decay constant (λ) for ^{235}U [8]. The total delayed neutron fraction is 0.67%

Group	β (%)	λ (s^{-1})
1	0.0221	0.0124
2	0.1467	0.0305
3	0.1313	0.111
4	0.2647	0.301
5	0.0771	1.14
6	0.0281	3.01

3.4 Results and Discussion

3.4.1 Input Data

A MSNB concept was used as a case study. This design uses 19.75% enriched UF_4 derived from High-Assay Low-Enriched Uranium (HALEU) as a fuel salt, and dissolved at 18 mol% in FLiNaK, a eutectic mixture of three alkaline fluoride salts: LiF, NaF, and KF [14]. This is an epithermal (0.167 eV) reactor with microscopic fission cross-section of $\sigma_f^{235} \approx 180\ b$ and a microscopic ^{135}Xe neutron absorption cross-section of $\sigma_a^{Xe} \approx 430\ kb$ [41]. The core is a $166\ cm$ tall right cylinder $25\ cm$ in radius, and has a nominal transport time of $\tau_{salt} \approx 130\ s$. With a primary coolant loop length (which has equivalent cross-section) of $571\ cm$, this corresponds to a volumetric flow rate of $8.26\ L/s$. The reactor is operated at low flux in the interest of fuel longevity [14]. At a more typical core average epithermal neutron flux of $10^{14}\ n/cm^2 - s$, the reactor would generate around 200 MWth of power, approaching the power density limit of liquid fueled molten salt reactors [42].

The generic MSR reactor has been designed with constant flow-cross section, meaning that half of the salt is inside the core and half is outside. The overall volume-integrated average flux is therefore $5 \times 10^{13} n/cm^3 - s$ to account for the time the salt spends in the downcomer. As this is a natural circulation design, the higher power would result in a higher mass flow rate, but for the purposes of this study, it is assumed that a reactor designed to operate at this higher flux will include additional valving to add sufficient pressure drop to hold the nominal transport time at $\tau_{salt} \approx 130\ s$.

Natural circulation systems in the fully developed turbulent flow regime have a cubic relationship between the mass flow rate and the thermal power transported[43, Ch. 3]. In the hours after shutdown, decay heat accounts for 3-5% of full power [13, Ch. 3], so the hot-standby flow rate should drop to about 30-35% of the operational flow rate. As such, the class method 'scram(cls)' also increases τ_{salt} by a factor of 3 which represents managing decay heat in a roughly isothermal manner.

3.4.2 Case Studies

Benchmark Study

First, a base case study was conducted where the xenon stripping module remained off for the entire experiment. The reactor is started, operates until ^{135}I and ^{135}Xe levels equilibrate, and is scrammed. The results given by the numerical solver are compared with characteristic stationary points of the analytical solutions so that the model can be benchmarked before moving on to studies where the stripping term is required.

The ^{135}Xe and ^{135}I concentration profiles during a clean start-up are depicted by Figure 3.3. The ^{135}I level rises to an equilibrium concentration of 28.4 $^{mmol}/_{m^3}$ as a first order response over the course of two days. The ^{135}Xe level follows an over-damped second order response to the its equilibrium level (denoted by the horizontal dashed line) of 20.0 $^{mmol}/_{m^3}$. Using Eqn. 3.5, equilibrium xenon accounts for -1.38% of poison reactivity.

The ‘shoulder’ during the first few hours is formed because its primary source, precursor decay, is initially null. Similarly, the elongation of the curve is caused by the changing formation rate as the ^{135}I concentration rises.

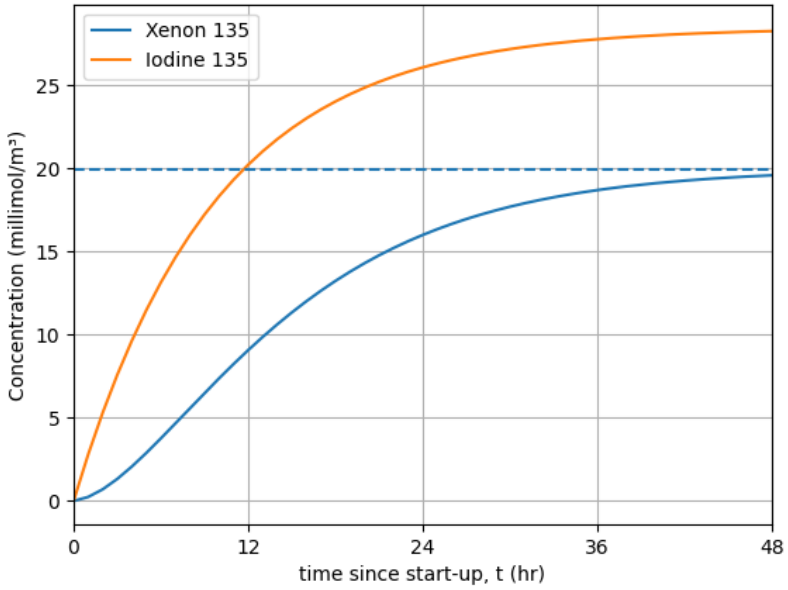


Figure 3.3: Concentration of ^{135}I and ^{135}Xe vs. time following cold/clean start-up. ^{135}I and ^{135}Xe are formed by fission as soon as the reactor starts. Soon after, ^{135}I decays to ^{135}Xe , and ^{135}Xe captures neutrons, being transmuted to ^{136}Xe . These two species reach equilibrium values when generation and consumption are equal.

After ^{135}I and ^{135}Xe equilibrate, the reactor is scrammed by setting the prompt neutron flux to zero, subsequently producing the behavior shown in Figure 3.4. The ^{135}I concentration falls by simple exponential decay, while the ^{135}Xe concentration undergoes a second order response with numerator dynamics, peaking at 23.11 $^{mmol}/_{m^3}$ 7 hours after the scram. The -1.60% xenon reactivity at the peak represents a 16% super-equilibrium. The ^{135}Xe concentration crosses its full power equilibrium 12.4 hours after the scram, which is denoted by a vertical dashed line. Delayed neutrons make a small contribution over the hours following shutdown, decaying from 0.67% to 1 ppm of the full power flux in under 8 minutes. Impact to the ^{135}Xe concentration profile was estimated to be at the part-per-billion order by inspection of results with and without the delayed neutron subroutine.

Verification of the Numerical Solver

For the given inputs, Eqn. 3.3 and Eqn. 3.4 predict an equilibrium ^{135}I concentration of 28.4 $^{mmol}/_{m^3}$, and an equilibrium ^{135}Xe concentration of 19.96 $^{mmol}/_{m^3}$. Eqn. 3.8 predicts that the peak of the inverse response occurs 5.1 hours and 3 minutes after the reactor is scrammed, with a magnitude of 23.1 $^{mmol}/_{m^3}$ obtained by inserting the peak time into Eqn. 3.7. The analytic solutions of characteristic stationary points of the system of ODEs give identical results with respect to the numerical solver, verifying the results of the model for the benchmark study.

To further verify the numerical solver, the path by which nuclide concentration approaches equilibrium can be compared to analytical solutions. By making the simplification of constant (time-independent) neutron flux, an analytical solution can be obtained for the ^{135}I concentration:

$$I(t)|_{\phi} = (I_o - I_\infty)e^{-\lambda_I t} + I_\infty \qquad \text{(Eqn. 3.19)}$$

The accuracy of the numerical solver was quantified by comparing the ^{135}I concentration array to Eqn. 3.19. An $1 - r^2$ score of 1.46×10^{-8} was obtained over the first 48 hours of start-up. This value is not sensitive to the initial condition, with approximately the same r^2 value being observed regardless of the ^{135}I level at start-up. This was tested over a range from 10% to 5 times the equilibrium value.

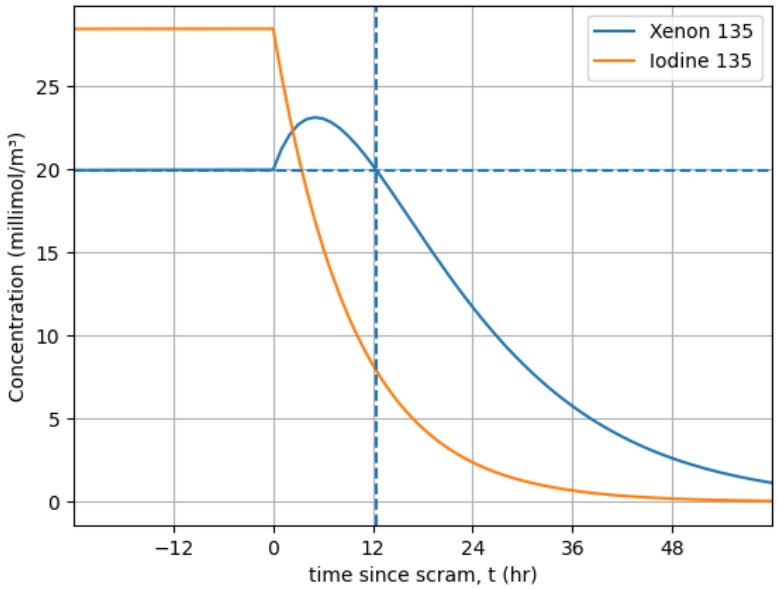


Figure 3.4: Concentration of ^{135}I and ^{135}Xe vs. time following reactor scram. When the reactor is scrammed, fission and transmutation cease. ^{135}I and ^{135}Xe each continue decaying. Initially, ^{135}I decays at a higher rate due to the shorter half-life and larger population. The ^{135}Xe concentration reaches its peak when the rate of ^{135}I decay drops below the rate of ^{135}Xe decay. After the equilibrium value is passed, the reactor may be restarted safely.

The ^{135}Xe rise to equilibrium is more difficult to analytically solve, owing to the variable generation rate caused by the changing decay rate of ^{135}I . Because I_∞ is already known, we can choose this as I_o, which makes the ^{135}I concentration and ^{135}Xe generation rate constant at $\gamma_{Xe}\Sigma_f^F\phi$ and the ODE separable. This is not a physically meaningful calculation, but it does hold mathematical meaning, and the solver is capable of computing it. The solution to this simplified ODE has the same form as Eqn. 3.19, with the effective decay constant, $\lambda_{Xe} + \sigma_a^{Xe}\phi$, in the exponent. By replicating the assumptions of the simplified ODE, the numerical solver was found to have an r^2 value of essentially 1. During normal operation of the solver, some error propagation is expected which would cause the numerical ^{135}Xe solution to be slightly less precise. However the low error calculated in the ^{135}I verification limits the effect of error propagation.

The ^{135}I verification gives assurance that the fission yield and beta decay methods work as intended, while the ^{135}Xe scenario verifies the radiative capture method. To verify the precursor decay method, Eqn. 3.6 and Eqn. 3.7 are compared to the numerical solutions, obtaining $1 - r^2$ scores of 1.05×10^{-9} and 4.59×10^{-10}. The level of precision achieved by the numerical solver in the verification studies gives confidence that it will accurately predict the behavior in more complicated studies, for which analytical solutions are difficult to obtain.

Super-Equilibrium Restart

The first experiment was repeated, but this time the reactor was restarted at the xenon peak, producing the behavior shown in Figure 3.5. 96 pcm of ^{135}Xe burned out in the first hour, and the ^{135}Xe reactivity dipped to 120 pcm below its equilibrium level 10.4 hours after the reactor was restarted.

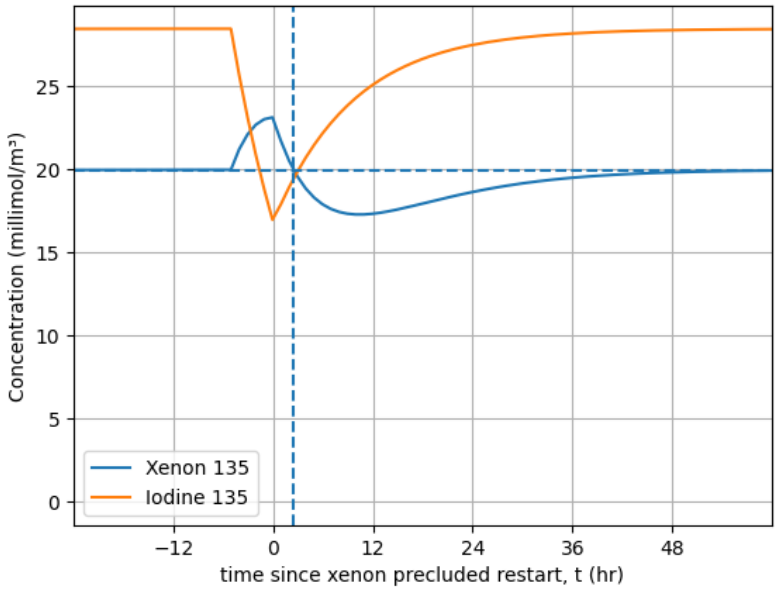


Figure 3.5: Concentration of ^{135}I and ^{135}Xe vs. time following restart at xenon peak. When the reactor is restarted, fission and transmutation resume. ^{135}I immediately builds back up, while ^{135}Xe burns out, returning to equilibrium after a substantial overshoot.

This is a significant disturbance to the poison reactivity of the reactor during operation. The magnitude of the reactivity swing during the restart is equivalent to 25% of the difference between clean start-up and equilibrium xenon poisoning, and the maximum burnout rate is more than 1.5 times the maximum ^{135}Xe build-up rate during initial start-up (63 pcm/hr). The most significant issue is that the burn-out of super-equilibrium ^{135}Xe during operation causes a positive feedback loop that the reactivity controller has to reject; if the drop in poison reactivity is not counteracted by control rod insertion, the neutron population grows, which increases the xenon burn-out rate.

With proper reactor and controller design, this disturbance can be managed. With a temperature coefficient of 3.5 pcm/K, the core temperature could swing by up to 27 degrees Celcius in the first hour, and 97 degrees overall [44]. Depending on the hot standby temperature, coolant salt volatility limits could be approached. More likely, downstream systems such as power cycles and process heat applications may be disturbed, putting stress on the plant-wide distributed control system. While this may be manageable, it would be preferable, and potentially required, to avoid this type of re-start.

Scheduled Restart

An MSR deployed in a load-following capacity on an isolated grid may at times need to be shut down (or ramped down) overnight, then be powered up in the morning. The numerical solver was used to study such a

scenario plotted in Figure 3.6. The ^{135}I concentration decays as in Figure 3.4, and the ^{135}Xe concentration rises initially. Due to the stripping term in Eqn. 3.15, the peak is lower and occurs sooner, and the poison level crosses the equilibrium value sooner. The flow rate ratio ($\frac{\dot{V}_{He}}{\dot{V}_{salt}}$) required to return the ^{135}Xe concentration to its equilibrium value after 8 hours of dead-time is 2.15×10^{-7}. This corresponds to just 2.1 $^{mL}/_{hr}$ of helium, or 16.8 mL over the entire dead-time. At approximately atmospheric pressure and 1000 K, this is such a small amount of helium that, if single-stage equilibrium is practically achievable, it may make more sense to release all of the necessary helium in a single pulse or series of pulses over the dead-time rather than continuously.

Such a pattern may fit into the energy demands of a modest island population, for example Guam, where much less power is required overnight. Conventionally this would not be feasible as it would require powering up from hot-standby or low-power mode during the xenon peak. A xenon stripping module could be coupled to the reactor to shorten the super-equilibrium time period, and make it possible to produce electricity on schedule.

A more traditional method for accelerating the effective half-life below what is achieved by beta decay alone, is low power burn-out. Maintaining the shut-down reactor at 1% power, which is on the order of the thermal power from decay heat, can reduce the effective half-life to 9.11 hours (compared to the half-life of 9.21 for pure beta decay). In comparison, the usage of the xenon stripper outlined in this experiment yields an effective half-life of 7.29 hours. This represents a 25 fold increase in the effectiveness of ^{135}Xe removal.

Demand Response

Power-peaking is another dynamic energy sector that must be decarbonized. An MSNB acting in this capacity will need to be able to black-start within minutes of a call for power. To return the ^{135}Xe concentration from its peak at 7.0 hours after shut-down to its equilibrium level in 390 seconds (3 hot-standby flow periods) requires a

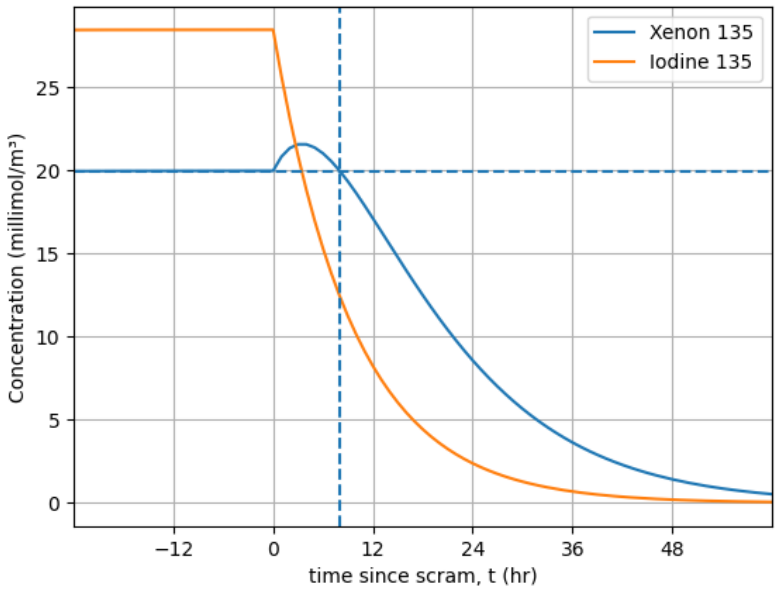


Figure 3.6: Concentration of ^{135}I and ^{135}Xe vs. time following reactor scram with Xenon Stripping Module. The same dynamics take place as in Figure 3.4, except in this case the ^{135}Xe is also being removed from the molten salt fuel in the Xenon Stripping Module, so the reactor may be restarted in 8 hours.

helium flow rate of 2.9 $^{mL}/_{min}$, totalling 19 mL over the entire start-up procedure. Figure 3.7 displays the ^{135}I and ^{135}Xe as calculated by the numerical solver with these settings. It is identical to Figure 3.4 until the peak poison level, when the xenon stripping module is activated and the ^{135}Xe level drops rapidly.

If this standby restart was conducted every day for 10 years, it would require only 1 mole of helium gas. This is a conservative estimate due to the short duration that the xenon-stripping module is active. The solver instantly distributes the effect of the stripped xenon across the entire reactor. This prematurely depresses the mass transport driving force and therefore predicts a lower separation efficacy than would be observed in practice. Expansion of the solver to a 1D+time or multi-region lumped parameter model could improve the accuracy for this case study, at the expense of significantly more computational expense.

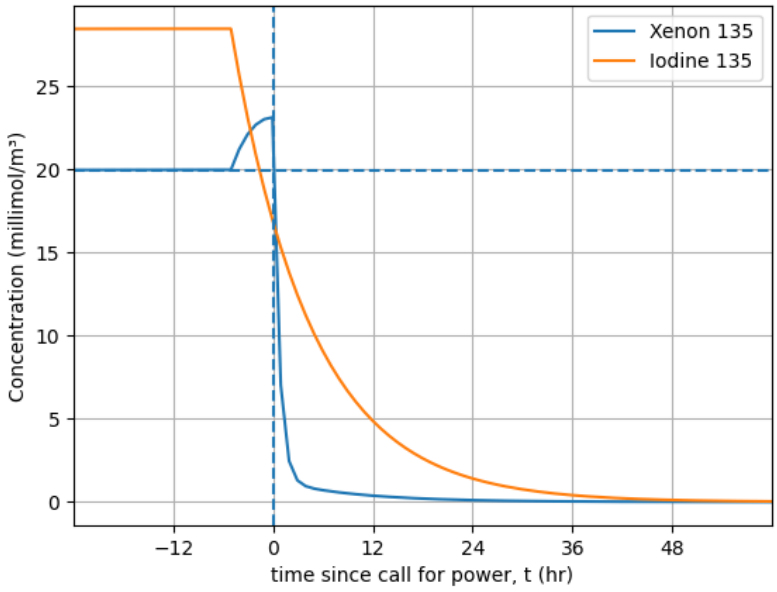


Figure 3.7: Concentration of ^{135}I and ^{135}Xe vs. time following reactor scram with Xenon Stripping Module operating only immediately after the ^{135}Xe concentration reaches its peak. The same dynamics take place as in Figure 3.4, except in this case ^{135}Xe is rapidly stripped from the molten salt fuel immediately after the ^{135}Xe concentration reaches its peak.

Operators of an MSR that is implemented in a power-peaking application will not know exactly when it needs to be restarted, but will need to have the reactor on standby. If this is the case, the xenon stripping module could remain off until the start-up signal comes. The module could then operate at a much higher helium flow rate than seen in the previous experiment for a short period of time similar to the start up time of our current natural gas peaking generators [45] before the MSNB is black-started.

End of Life Fuel

Molten salt microreactors such as the MSNB could extend their lifetime with xenon stripping. At the end of the fuel lifetime, excess reactivity could be reclaimed by constantly removing ^{135}Xe from the fuel, driving down the equilibrium concentration. A helium flow rate of 146 mL/hr reduces the equilibrium poison reactivity from 1.38% to 0.15%, a reduction of 1227 pcm. Figure 3.8 displays that the ^{135}I concentration is unchanged with respect to Figure 3.3, while the ^{135}Xe concentration is greatly depressed.

Equilibrium xenon accounts for a significant amount of poison reactivity at high power density. Because microreactors are small, they are often limited in excess reactivity. They are also often designed to last upwards of a decade without refueling. For an MSNB with approximately 5% excess reactivity at initial start-up, ^{135}Xe poisoning greatly reduces the fuel lifetime. A reduction in the equilibrium poison level such as the one described in this experiment requires 1270 L of helium per year, but could conceivably extend the overall reactor lifetime by up to 30%. This result alone, without considering the effect of shortening restart time, makes the xenon stripper an important design feature to small MSRs, which are competing for a market share in a class of reactors that are aiming for decade scale operation [6].

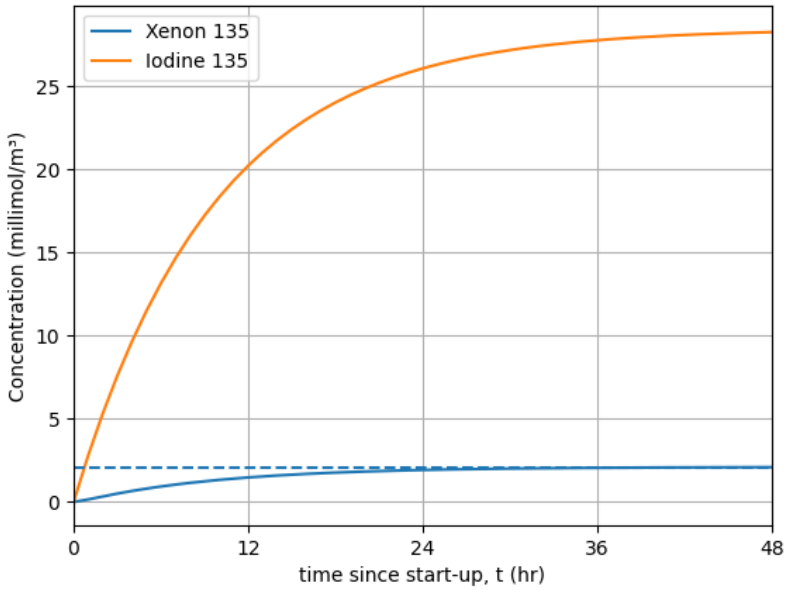


Figure 3.8: Concentration of ^{135}I and ^{135}Xe vs. time during reactor start-up with Xenon Stripping Module operating to lower the equilibrium ^{135}Xe concentration.

3.5 Future work

This work is a computational screening of xenon stripping for neutron poison management on a generic MSR, both during operation and during brief shut-down periods. Furthermore, it contains the development of a preliminary design tool that can be used to advise decisions in the specific design of MSRs. Many assumptions were made, notably a constant Henry's Law coefficient, which is subject to change during operation due to changing composition, and a constant natural circulation flow rate. These assumptions were made because of the lack of experimental data and the effect of design specifications on these parameters. By making the solver open source, we encourage other researchers to refine these assumptions as progress in experimental work supports.

3.5.1 Xenon Stripping Module

This paper works on the simplifying assumption that the xenon stripping module is an equilibrium mass transport unit operation. It has been demonstrated that the thermodynamics of ^{135}Xe stripping from molten salt fuel are highly favorable due to xenon's noble gas nature. This warrants a more detailed investigation into the kinetics of the unit operation. Literature must be consulted to design such a module [38, Ch. 10]. It is highly

likely that a higher helium flow rate will be required to prevent flooding and result in a sufficient interfacial area to volume ratio for mass transport to occur at any meaningful macroscopic rate.

This paper also neglects the effect of other fission gasses that will be produced in the core, namely tritium. Core burn-up modeling will need to be conducted, and literature reviewed to see what other gasses may be stripped [29]. These additional gasses may inhibit xenon stripping by effecting the mass transfer coefficient, but will more likely facilitate improved stripping by contributing to the formation of circulating voids which may be removed from the salt more readily [30].

3.5.2 Numerical Solver

The numerical solver is an object oriented program, making it modular. The architecture supports the instantiation of additional nuclides with a minimal amount of extra code. It could be used to study other equilibrium poison systems such as the ^{149}Sm decay chain, burnable poisons like gadolinium isotopes, and non-burnable poisons like hafnium isotopes. Outside of neutron poisoning, the solver could be modified to optimize isotope production schemes, track the gamma-ray source strength or decay heat potential of irradiated materials, or even estimate radiation damage and fission product interaction.

The solver is faster than real-time. A potential application beyond preliminary design is in control systems. A similar architecture could be employed to develop reduced-order just-in-time digital twins which would advise supervisory controllers during transient operations. This would allow, for example, the control system of a nuclear reactor to account for poison reactivity changes without having to wait for the disturbance to cause a set-point error, as would be required in a pure feedback controller.

3.6 Conclusions

A new model was derived that quantifies the advective removal rate of ^{135}Xe from a molten salt fuel in an equilibrium stripper using Henry's law constants obtained from literature. It was used to demonstrate that, given appropriate conditions, mass transport can strongly outweigh beta decay in the post shutdown ^{135}Xe decay chain dynamics. Results show that a bulk gas stream on the order of milliliters provides enough volume for nearly all of the ^{135}Xe in a liter of molten salt fuel to transport into the gas phase. The challenge now shifts from providing sufficient volume to do this, to providing sufficient interfacial area; this paper shows that this problem is worth solving. If solved, this technology opens the door to MSRs breaking into remote grids and power peaking.

Chapter 4: Reactor Characterization

To design a reactivity controller for the Molten Salt Nuclear Battery, many components of the reactor needed to be characterized. The reactor was modeled in Serpent 2 to characterize the control drums using a series of criticality models. Depletion models were also used observe how the control drum characterization changes as the fuel is burned. Finally, a 1-dimension (spatial) uniform-state uniform-flow finite element model was developed in Python. This serves as the thermohydraulic process simulation needed to approximate the autonomous dynamic operation of the reactor during transients as well as study the performance improvement gained by application of the controller.

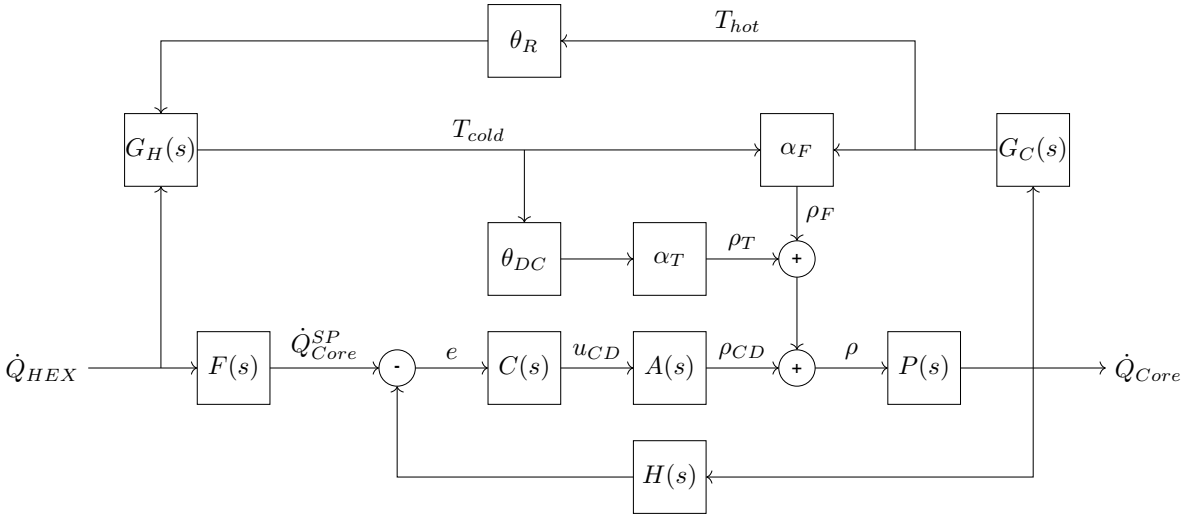


Figure 4.1: Control loop of a natural circulation MSNB. It is a normal feedback loop with a pre-filter, with the addition of the passive feedback mechanisms. The core ($\dot{Q}_{Core}$) and heat exchanger ($\dot{Q}_{HEX}$) powers go through the respective temperature dynamics (G_C and G_H) and time delays for the riser (θ_R) and downcomer (θ_{DC}) before being converted to reactivity by the temperature(α_T) and flow (α_F) feedback mechanisms. The passive reactivity feedback is combined with the control drum reactivity (ρ_{CD}) and fed into the reactor dynamics ($P(s)$).

4.1 Reactor Design

The MSNB is self contained in a 145 cm diameter, 242 cm tall cylindrical rector vessel that is buried in the ground or concrete for shielding purposes. The core is a concentric 166 cm tall cylinder 50 cm in diameter surrounded by a large reflector into which 8 equally spaced control drums are embedded. A neutron trap sits above the reflector to separate the riser, where fission caused by delayed neutrons occurs at a significant rate, from the heat exchanger. The downcomer is an annular gap between the outer part of the reflector and the outer reactor vessel. It has flow area identical to the core, and returns cold salt from the outlet of the heat exchanger to the inlet plenum.

Figure 4.2 is an axial cross-section of the MSNB. Figure 4.3 contains four radial cross-sections of the MSNB. These two figures were generated by the Serpent model described in Section 4.2. In these models, UF_4dissolved in FLiNaK is depicted as varying shades of red depending on temperature, beryllium-oxide is light-blue, boron-carbide is green, graphite is yellow, stainless steel is light-gray, Hastelloy-N is medium-gray, barite concrete is dark-gray, and air is pink. The reactor is buried in barite concrete for radiation shielding purposes, and the top is at the surface so secondary coolant systems can readily be connected.

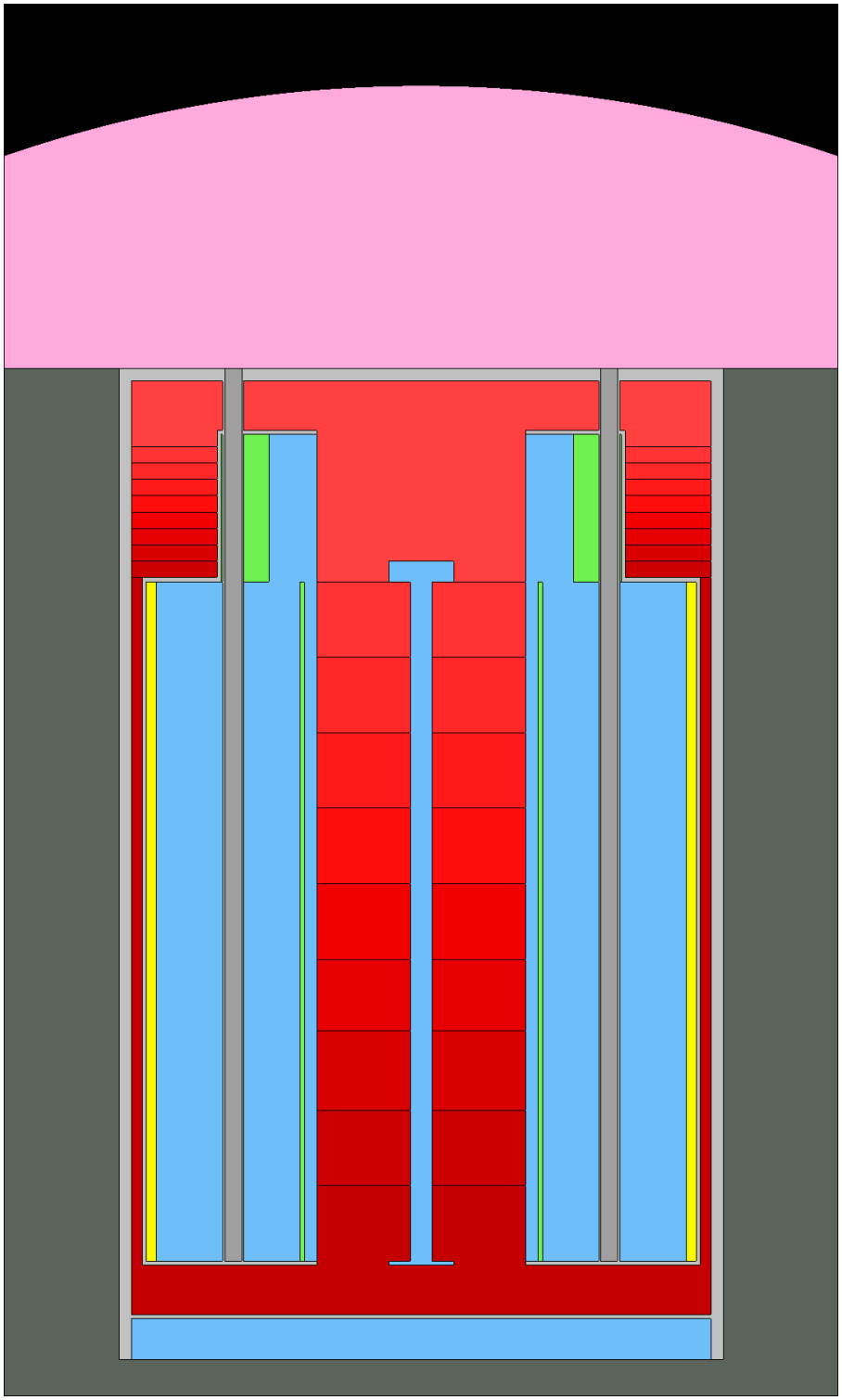
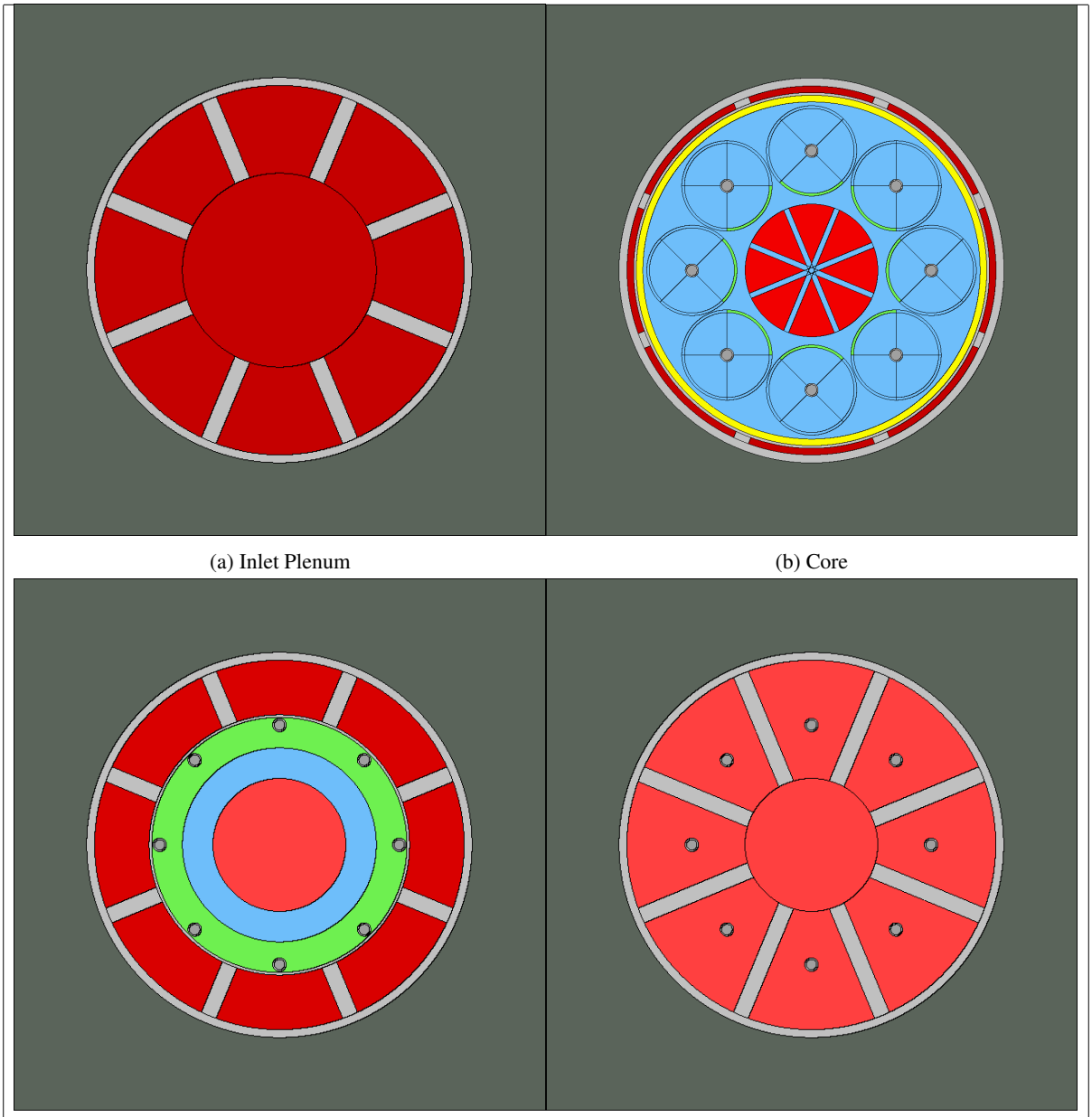


Figure 4.2: Y-Z view of MSNB. Molten salt is shown in red, with darker shades corresponding to a lower temperature and higher density. The core is surrounded by the beryllium-oxide reflector (blue) and the boron-carbide absorber plate (green). The core is separated from the riser by a a perforated reflector plate and the riser is separated from the heat exchanger by an absorbing ring. From this angle the internal moderating structure appears as only the center rod, as the fins exist in other planes.



(a) Inlet Plenum

(b) Core

(c) Heat Exchanger

(d) Spoke

Figure 4.3: X-Y Views of MSNB a) Molten salt from the downcomer travels inward below the reflector the center, where it rises to enter the core; b) The core is surrounded by the reflector and control drums, which may be adjusted to manipulate criticality. The reflector and downcomer are separated by a graphite shield; c) The molten salt in the outer ring is in the heat exchanger. As it sinks, heat is being rejected to the secondary coolant (not modeled). A ring of boron-carbide ensures that delayed neutrons emitted in the riser do not transport to the heat exchanger; d) Molten salt exiting the riser travels radially outward to the top of the heat exchanger;

4.1.1 Molten Salt

The molten salt in the MSNB serves as both the primary coolant and the fuel. It is composed of 18 mol% HALEU UF_4 (enriched to 19.75%) dissolved in eutectic FLiNaK (enriched to 99.99% ^7Li). It is composed of about 1.4 atom% ^{235}U . The remaining composition is listed in Table 4.1. This molten salt fuel system has been

studied in previous work [5]. The present work leverages intermediate calculations, such as the thermophysical property temperature functions - density[1] (ϱ) and heat capacity (c_p) - of the salt:

$$\varrho[kg/m^3] = 4682.0365 - 0.94301046 \cdot T[K] \quad \text{(Eqn. 4.1)}$$

$$c_p[kJ/kg - K] = 0.97678 + 0.0010634 \cdot T[K] \quad \text{(Eqn. 4.2)}$$

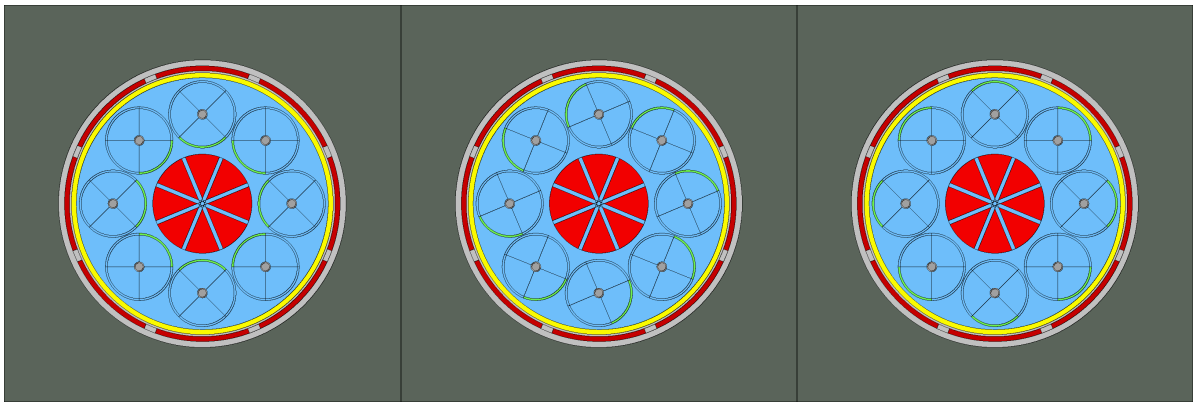
FLiNaK was selected for study due to its thermal properties and prevalence in MSR research, and UF_4 was selected as it is soluble in FLiNaK to a concentration that has a wide liquidus range, supports criticality in the reactor geometry necessary to fit into a shipping container, and provides adequate power density for thermal hydraulic design.

Table 4.1: Composition of molten salt prior to burn-up

Element	Isotope	Atom Percent	Weight Percent
Fluorine	19	60.63 %	32.40 %
Lithium	6	15 ppm	2.5 ppm
	7	15.01 %	2.96 %
Sodium	23	3.71 %	2.40 %
Potassium	39	12.61 %	13.82 %
	41	0.95 %	1.09 %
Uranium	235	1.40 %	9.25 %
	238	5.69 %	38.08 %

4.1.2 Control Drums

Control drums are cylinders of neutron reflector with a portion of the circumference replaced with a neutron absorber. As is depicted by Figure 4.4, rotating the control material inward inhibits the neutron chain reaction. This concept is used in the Advanced Test Reactor at INL, using a beryllium/hafnium design [46]. It is also a popular concept for accident tolerance in space reactors, which have the potential to crash during launch [47].



(a) Least Reactive (b) Initial Criticality (c) Most Reactive

Figure 4.4: X-Y Views of MSNB with control drums in three orientations: a) 0 degrees, which is used to test the reactor's shutdown margin; b) 112 degrees, which is the initial critical orientation; and c) 180 degrees, which is used to test the reactor's excess reactivity; Beryllium oxide is depicted in blue while boron carbide is green.

[1] note the use of ϱ to distinguish density from reactivity (ρ)

Each of the eight drums in the MSNB is the same height as the core, 17 cm in radius, and has a 1 cm thick absorber pad covering 25% of the circumference. This study uses beryllium oxide as the reflector material for cost, machinability, and radiation damage considerations, and boron carbide for the absorber, as it provides adequate shutdown margin while preserving more excess reactivity than hafnium.

In addition to the higher absorption cross-section, hafnium also is considered a non-burnable poison, as the transmutation products also have a high cross-section. In contrast, the products of ^{10}B capture are essentially transparent to neutrons. It was confirmed using a burn-up study that the boron carbide would not be significantly depleted during the life of the MSNB.

4.1.3 In-Pile Moderator

Previous work has suggested an in-pile helix made of a neutron scattering material to extend the in core flow path and simultaneously soften the neutron energy spectrum to provide more excess reactivity [5]. This work investigates a simpler version of this concept focused only on providing excess reactivity. It is composed of 8 radial fins spaced 45 degrees apart, and is made from beryllium oxide.

4.1.4 Structural Materials

The reactor vessel, along with supplementary structural materials such as reflector and moderator supports, heat exchangers, and control drum driveshaft sheaths are made from 316 stainless steel. Control drum driveshafts are made from Hastelloy-N, a nickel-chromium-molybdenum alloy that is resistant to corrosion from high temperature fluoride salts. The reactor vessel is encased in barite concrete for added radiation shielding.

4.2 Neutronics Modeling

The reactor described in Section 4.1 was modeled in Serpent 2, making use of the Sawtooth supercomputer at INL's HPC center. The fuel was stratified for an expected temperature rise using Eqn. 4.1 and redundant material cards. A python script was written to automatically generate the input cards that reflect a specific molten salt composition and control drum angle[2] This made it easy to submit batches of several criticality models sequentially to Sawtooth using a Bash submission script.

An alternating methodology of criticality and burn-up models was employed to characterize the control drums. First, the k_{eff} for the entire range of control drum angles (from 0 to 180 degrees) was obtained using a criticality model that used 1,000,000 source particles per cycle, 500 active cycles, and 100 inactive cycles. With 32 nodes parallelizing 4 threads, each control drum angle took between 2 and 5 minutes to complete. This allowed the entire range of control drum angles to be simulated with a wall-time of 2 hours.

With the entire control drum angle vs. reactivity curve defined, the unity point was calculated and a burn-up model was conducted at that angle. A stochastic volume calculation was obtained using the 'mcvol' subroutine, and the depletion module was loaded at 10 MW with a time-step of 6 hours until ^{135}Xe reached equilibrium, and 6 months following. Originally, smaller time-steps of 1-5 days were used to study the change to the control drum-reactivity curve as ^{149}Sm built up. This was forgone after finding that ^{149}Sm acts as a non-equilibrium fission product neutron poison at the relatively low power density studied.

To approximate the flowing and electrolytic nature of the MSNB that disperses fission products over time, the burn-up studies were completed without fuel stratification. This is in contrast to how solid fueled burn-up studies are often conducted, where depletion zones are employed to resolve the effects of the spatial neutron flux profile. This provided the new molten salt and boron carbide compositions for the next set of criticality models.

[2]The input file writing script is available at https://github.com/sjroot97/MSNB-Serpent2-Autodeck and is also included in Appendix B.

4.3 Process Simulation

A multiphysics transient simulation was written in Python to study the MSNB in dynamic operations by the coupling of thermal hydraulics and neutronics. First principle physics were implemented to approximate the expected behavior of the reactor. It is not meant to be a digital twin, but rather a responsive tool for the design of the power controller[3].

The simulation is built on three physics principles: 1) Thermally driven natural circulation flow mode, where the pressure differential driven by a difference in density between the hot and cold leg is equilibrated by frictional losses to calculate the flow rate; 2) Reactor point kinetics, where the compound passive dynamics are used to time-advance the reactor power based; and 3) Uniform-state uniform-flow time and spatial advancement of energy in the flow loop.;

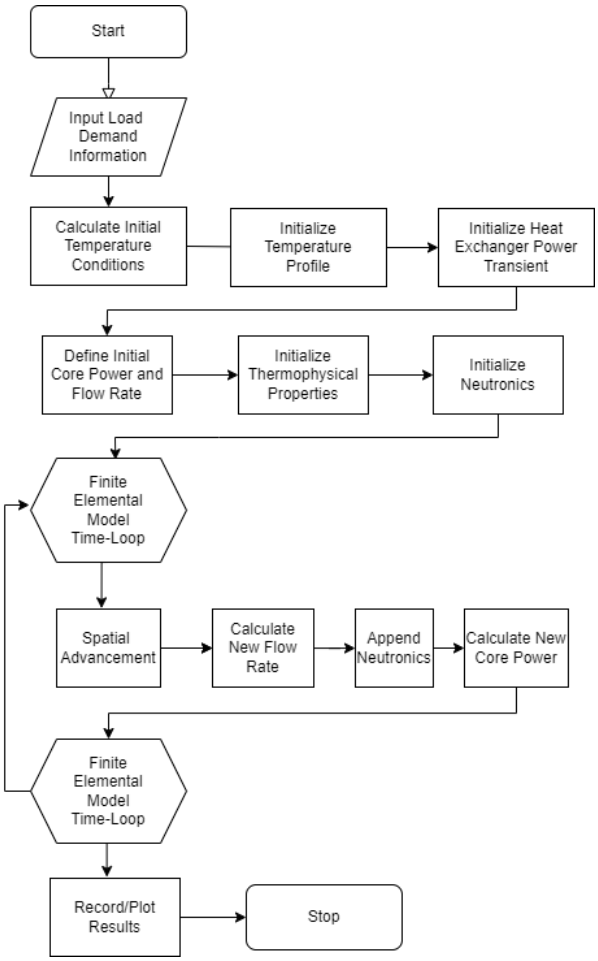


Figure 4.5: Logic flow diagram for the MSNB transient process simulator.

Figure 4.5 is a logic flow diagram that describes how the code simulates the dynamic operation of the MSNB. Similar to previous work, the flow loop of the reactor was simplified into a 1-dimensional flow loop with equal flow area throughout [14, 5, 48]. The simulator is broken up into two different sections - first the problem is set-up, then the transient is simulated in a finite element time loop with time steps of 1 second. The model is described in Sections 4.3.1 and 4.3.2, and rely on the parameters listed in Table 4.2

[3]The input file writing script is available at https://github.com/sjroot97/MsNB-Simulator/ and is also included in Appendix C.

Table 4.2: System parameters used in the simulation of the MSNB [14].

	Value	Unit	Description
α_T	-3.5	pcm/K	Temperature reactivity feedback coefficient
ℓ^*	1.63×10^{-4}	sec	Mean neutron generation time
β_{eff}	6.96×10^{-3}	-	Effective delayed neutron fraction
λ	0.1	sec^{-1}	One-group delayed neutron precursor decay constant
ξ	25	-	Pressure loss coefficient
h	1.09	m	Height between thermal centers
α_f	-0.3398	s/cm	Flow reactivity feedback coefficient
A_x	0.4	m^2	Cross-sectional flow area
g	9.81	m/s^2	Acceleration due to gravity
δt	1	sec	Time-step duration
δx	1	mm	Control volume length

4.3.1 Model Initialization

The code first takes inputs to define the heat exchanger power transient that is to be studied. The reactor is initially assumed to be steady-state and critical, so the temperature is constant in the chimney and downcomer, and heating/cooling are uniform and equal in the core and heat exchanger. A binary search algorithm is used to iteratively find the cold leg (T_{cold}) temperature that corresponds to the initial power requirement with a hot leg temperature (T_{hot}) of 700 ^{o}C. The heat exchanger power ($\dot{Q}_{HEX}$) is calculated by:

$$\dot{Q}_{HEX} = \bar{\varrho}_{HEX} A_x v \bar{c_p}_{HEX} (T_{hot} - T_{cold}) \quad \text{(Eqn. 4.3)}$$

Where $\bar{\varrho}_{HEX}$ and $\bar{c_p}_{HEX}$ are the average density and heat capacity in the heat exchanger, A_x is the cross-sectional flow area within the reactor, and v is the velocity of the salt in the loop. The key assumption for this model is that the flow velocity during a given time-step is uniform throughout the flow loop, both during steady-state and unsteady-state time periods. This assumption allows for a simple spatial advancement, and an explicit total energy balance solution, which is described in detail in Section 4.3.2. The flow velocity is calculated based on the temperature profile, assuming a natural circulation flow mode:

$$v = \sqrt{2gh(\varrho_{cold} - \varrho_{hot})/\xi\varrho} \quad \text{(Eqn. 4.4)}$$

where g is the gravitational acceleration, h is the height between thermal centers, ξ is the pressure loss coefficient (estimated to be 25.0 by STAR-CCM+ [14]), $\bar{\varrho}_{cold}$ and $\bar{\varrho}_{hot}$ are the average salt densities of the cold and hot leg, and $\bar{\varrho}$ is the average salt density of the entire reactor loop.

The criticality assumption must be satisfied between the two passive feedback mechanisms. Flow reactivity (ρ_f) is calculated explicitly, while temperature reactivity (ρ_T) is calculated relatively [8]:

$$\rho_f = -\frac{L}{L+H}\beta_{eff}\left(1 - e^{-v\alpha_f}\right) \quad \text{(Eqn. 4.5)}$$

$$\Delta\rho_T = \alpha_T \Delta T \quad \text{(Eqn. 4.6)}$$

where L and H are the out-of-core and in-core lengths, β_{eff} is the effective delayed neutron fraction, and α_f and α_T are the flow reactivity feedback coefficient and temperature reactivity feedback coefficient. Table 4.3 contains the loop coordinates for key points in the MSNB [14].

Table 4.3: MSNB loop coordinates for the transitions between equipment [14].

Location in Reactor	Loop Coordinate (mm)
Core Inlet	0, 5710
Core Outlet	1660
Top of Hot Leg	2090
Heat Exchanger Inlet	2340
Heat Exchanger outlet	2825
Bottom of Cold Leg	4975

For a critical core, the initial temperature reactivity is defined as equal and opposite the initial flow reactivity. This is satisfied by orienting the control drums at their bias point, such that control reactivity (ρ_C) is null.

4.3.2 Discrete Time Step

The process simulator uses two steps to simulate the passage of time: 1) The flow loop is advanced by the distance the molten salt travels during the discrete time step; and 2) Time dependent parameters are updated.

Spatial Advancement of Flow Loop

The spatial advancement of the flow loop is a 1D+time computational fluid dynamics problem, with each millimeter long slice (δx) of the entire cross-section of the flow loop being treated as a control volume. With the assumption of uniform flow velocity around the loop for each discrete time step, the continuity equation for the system can be expressed as:

$$\delta x A_x \frac{d\varrho}{dt} = v A_x (\varrho_{in} - \varrho_{out}) \qquad \text{(Eqn. 4.7)}$$

In solving the total energy balance for the time-step, it is useful to convert the temperature profile (T_x) to an energy profile (mu_x). This is done by using the temperature functions for density, Eqn. 4.1 and heat capacity, Eqn. 4.2 and assuming a reference temperature (T_r) where the energy is defined as null:

$$mu_x = \varrho(T_x)\delta x A_x c_p \left({}^{T_x+T_r}/_{2}\right)(T_x - T_r) \qquad \text{(Eqn. 4.8)}$$

By neglecting gravimetric, kinetic, and pressure-volume differentials and assuming the uniform-state uniform-flow condition, where the thermophysical and thermodynamic properties of the control volume are considered equal throughout, the discrete total energy balance for each control volume becomes:

$$\frac{d(mu)}{dt} = mu_{enter} - mu_{exit} + Q_{c.v.} \qquad \text{(Eqn. 4.9)}$$

In the simulator, the energy profile is stored as a numpy array, with the energy in each control volume being stored as a single element in the ordered series. During a discrete time-step, the salt described by the element corresponding to the control volume exits the control volume, the element lagging by $v\delta t$ enters the control volume, and every element in-between both enters and exits the control volume. As such, each of these interior elements can be neglected. mu_x does not need to be modified to obtain mu_{exit}, and mu_{enter} can be obtained by using the numpy 'roll' method, passing a value of $v\delta t$ for the 'shift' argument. The outlets the heat exchanger and core must be flattened to account for edge heating/cooling effects.

The control volume power ($Q_{c.v.}$) is calculated ay assuming uniform heating/cooling and preserving the convention of negative heat rejection:

$$Q_{c.v.} = \begin{cases} {}^{Q_{Core}}/_{\ell_{core}} & , \ Core \\ 0 & , \ Riser \\ -{}^{Q_{HEX}}/_{\ell_{HEX}} & , \ HeatExchanger \\ 0 & , \ Downcomer \end{cases} \quad \text{(Eqn. 4.10)}$$

After the discrete time-step, the new energy profile is calculated using Euler's forward method:

$$mu[t + \delta t] = mu[t] + \delta t \frac{d(mu)}{dt}[t] \quad \text{(Eqn. 4.11)}$$

Then, the updated temperature profile can be obtained from the updated energy profile by inverting Eqn. 4.8. This is not a readily separable function, so an expected range of temperatures was put through the forward function, and the results were fit to a 6th order polynomial to define an inverse function. With the updated temperature profile, the rest of the time variables can be updated to solve for the new core power.

Updating Time Variables

First, the updated temperature profile is used to calculate the new molten salt flow velocity using Eqn. 4.4, which is in turn used to calculate the new flow reactivity using Eqn. 4.5. Then the change in average core temperature is used to calculate the new temperature reactivity:

$$\rho_T[t + \delta t] = \rho_T[t] + \alpha_T \Delta T_{core} \quad \text{(Eqn. 4.12)}$$

Each reactivity phenomenon is summed, and reactor period (τ) is obtained using one group reactor point kinetics:

$$\tau = \frac{\ell^*}{\rho} + \frac{\beta_{eff} - \rho}{\lambda\rho + \dot{\rho}} \quad \text{(Eqn. 4.13)}$$

where λ is the delayed neutron precursor decay constant and $\dot{\rho}$ is the reactivity rate of change [19, Ch. 6]. Finally, the updated power can be obtained from the e-folding equation:

$$Q_{core}[t + \delta t] = Q_{core}[t] e^{\delta t/\tau} \quad \text{(Eqn. 4.14)}$$

With the new power, the entire sequence of calculations can be repeated for the next discrete time-step, repeating until the simulation is over. These steps result in a first principles based multiphysics model that simulates compound passive feedbacks to compute the approximate behavior of a natural circulation MSNB during dynamic operation.

Chapter 5: Controller Design

The process simulation was written in Python 3.10 and is modeled in the time-domain. One approach to the controller design would be to fit the autonomous dynamic response of the model to a second order transfer function and model the control loop in the Laplace domain using a software package such as Simulink. Despite using robust fitting heuristics, this method might sacrifice precision and potentially obscure certain features of the core power curve that arise due to the spatial dimensionality in the present work. Notably, the acceleration of outlet responses, as observed in previous lumped parameter models, is likely to lead to inaccuracies during periods of changing dynamics [49]. Instead, user-defined functions were developed to model the control loop natively.

5.1 Pre-filter Implementation

As a demand-response control loop, it is desired that the core power follows the heat-exchanger power. As discussed in Sections 2.2.2 and 2.3, there are thermal-hydraulic and kinetic benefits to the use of a pre-filter. A first-order time constant of $\ell_{downcomer}/v$ ensures that the controller allows temperature reactivity feedback to present as it matches the power output to the demand:

$$Q_{core}^{SP} = \frac{v}{\ell_{downcomer} s + v} Q_{HEX} \qquad \text{(Eqn. 5.1)}$$

where v is the initial flow velocity in mm/s and $\ell_{downcomer}$ is the length of the downcomer in mm. The heat exchanger power array (Q_{HEX}) was reshaped into the core power set-point (Q_{core}^{SP}) using the Python control system library [50]. The 'control.forced_response' method extends the functionality of scipy's 'TransferFunction' object to allow operation on user-defined input arrays.

5.2 PID Controller and Actuator Implementation

When the controller is active, the control drum feedback is included in the reactivity summation. At each time step, the instantaneous error is passed to a function that calculates the cumulative error and rate-of-change of error using global variables. These values are used along with the controller bias, controller gain, and integral and derivative time constants to calculate the desired control drum orientation. The control drum reactivity is then obtained using this angle and the actuator curve fitted in Section 6.1.3.

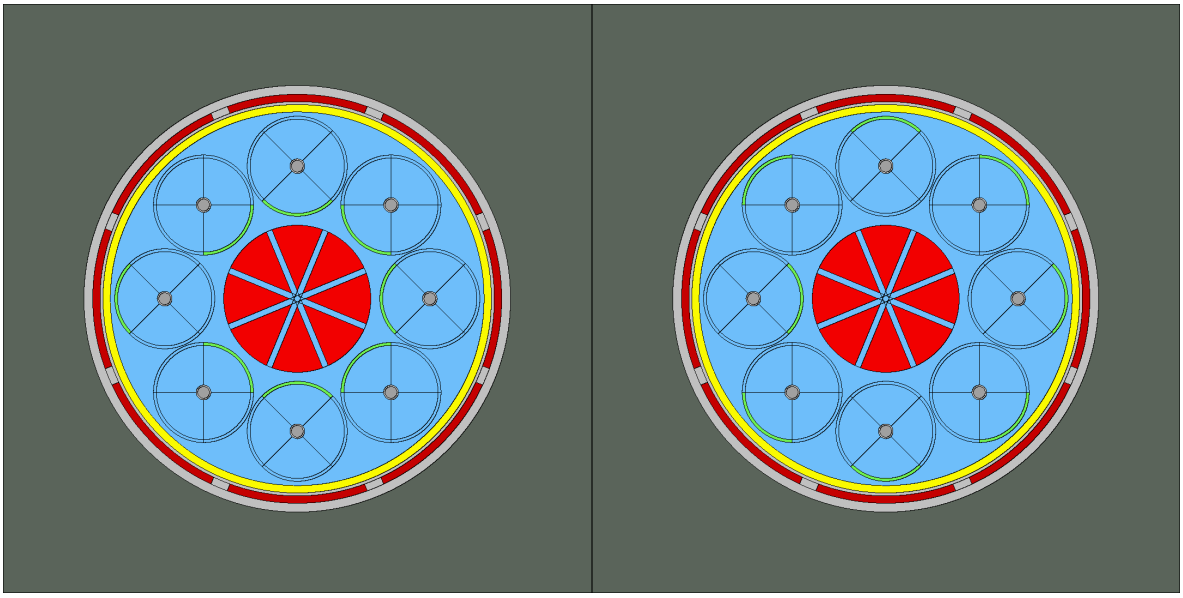
5.3 Core Power Transducer

The power sensor/transmitter is not modeled in the present work, as it is calculated from reactor point kinetics. The physical realization of the power sensor will be comprised of an array of temperature and flow sensors, likely K-type thermocouples and thermal mass flow sensors [51]. These sensors will be distributed around the core, and their data will be synthesized using the energy balance $Q = mc_p\Delta T$. This sensor is expected to be very noisy. The integral controller will help reject this noise to some extent, however some type of signal processing, *e.g.* a low-pass filter or a convolution will be needed. In future work, noise should be injected and processed. While the sensor will be noisy, the computational requirements are relatively modest; given that the simulator uses 1 second time-steps, the transducer is assumed to have fast-unity dynamics.

Chapter 6: Results and Analysis

6.1 Serpent Model

6.1.1 Excess Reactivity and Shutdown Margin



(a) Single Misfire (b) Single Success

Figure 6.1: X-Y Views of MSNB with control drums in two shut-down margin failure modes: a) Seven drums in least reactive orientation, one drum failed in most reactive orientation; and b) One drum in least reactive orientation, seven drums failed in most reactive orientation;

including burned B4C

6.1.2 Neutron Spectra

6.1.3 Actuator Curve

6.2 Multi-physics Simulation

6.2.1 Steady-State

6.2.2 Up-Step

autonomous controller tuning controlled response

6.2.3 Down-Step

6.2.4 Start-Up

6.2.5 Shut-Down

6.2.6 Demand-Response

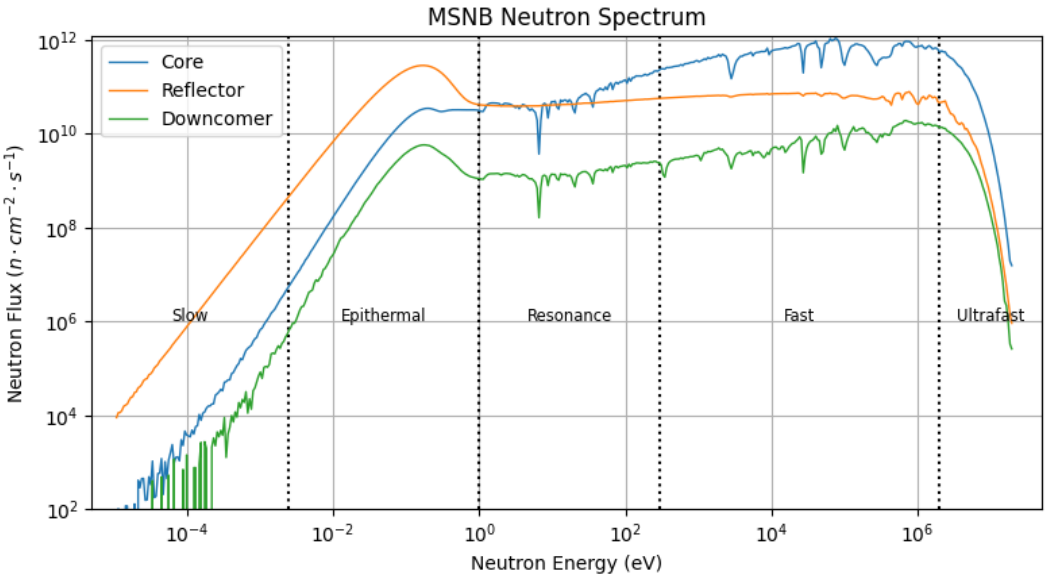


Figure 6.2: MSNB neutron energy spectrum in core, reflector, and downcomer.

Chapter 7: Conclusions

7.1 Limitations

7.2 Future Work

7.2.1 Fuel Salt Thawing

Because microreactors are meant to be delivered in a fully or mostly assembled state, it is likely that the MSNB will be shipped with the molten fuel/coolant salt mixture frozen *in-situ*. The salt will need to be melted before initial start-up, and the heat source for melting the cannot be fission, as the MSNB requires advective heat removal caused by natural circulation; this is not possible if the flow channels between the core and heat exchanger are frozen. One possible method for salt thawing involves passing low-voltage high-current electricity through the pipes in contact with the salt, similar to how frozen water pipes are thawed [52]; this would be coupled with the introduction of hot secondary coolant into the heat exchanger to provide the necessary hot reservoir.

7.2.2 Neutron Source

A neutron source will be required to start the fission chain reaction after installation, and after any long periods of inactivity. ^{235}U undergoes spontaneous fission [53, Ch. 6], and at high enrichment may be used as the only neutron seed simply by putting the control actuators in a supercritical orientation. More commonly, a dedicated source is used, such as ^{252}Ca , which undergoes spontaneous fission much more rapidly, or a composite of a strong alpha-emitter (*e.g.* ^{238}Pu , ^{241}Am , ^{210}Po , or ^{226}Ra) and ^{9}Be which emits a neutron according to Rxn. 7.1 [54, Ch. 2]. Dedicated neutron seed materials composed of these nuclides could be placed in the core through specialized mechanisms, though the introduction of the seed species as a soluble salt warrants a feasibility analysis.

$$^{9}Be + \alpha \rightarrow {}^{12}C + n + \gamma \qquad \text{(Rxn. 7.1)}$$

7.2.3 SCRAM System

The emergency shutdown (*i.e.* SCRAM) system must be passive. In LWRs, this is achieved by including large control rods which are actively held out of the core, so that a loss of power results in automatic insertion. Larger MSR designs may include a SCRAM tank into which the fuel/coolant salt drains in the event of power failure. These systems are often actuated by a freeze plug [55] and put the salt in a subcritical orientation by the inclusion of neutron control materials [56, Ch. 1] and high geometric buckling [20, Ch. 6].

In the MSNB, the control drums will be actively actuated such that loss of power results in a negative control reactivity insertion of the greatest possible magnitude. Still, a freeze plug SCRAM tank system should also be included to make the system truly fail-safe

how to emphasize that this is author opinion

7.2.4 Decay Heat Removal

Thermal power continues to be released by the decay of radio-nuclides after the fission chain reaction is stopped. For non-emergency shut-down, this heat may be removed by the same heat exchanger used for the secondary loop. There should also be a passive system that rejects decay heat in the event of total power failure, such as a direct contact system which removes heat through the vaporization of sodium [57] in the scram tank. This latent heat driven system is ideal because it minimizes the possibility of the salt freezing due to over-cooling.

7.2.5 Flow Rate Control

7.3 Summary Remarks

References

[1] Root, Sam J., Zhao, Haiyan, Borrelli, R.A., McKellar, Michael G., 2023. Thermodynamic analysis on xenon stripping to shorten restart time in molten salt microreactors. Nuclear Engineering and Design 414, 112606. ISSN 0029-5493. doi:https://doi.org/10.1016/j.nucengdes.2023.112606.
URL https://www.sciencedirect.com/science/article/pii/S0029549323004557

[2] NREL, 2021. Technology partnerships. Technical report, National Renewable Energy Laboratory (NREL).

[3] Nuclear Energy Institute, 2018. Roadmap for the deployment of micro-reactors for u.s. department of defense domestic installations. Technical report, Nuclear Energy Institute.

[4] Secretary of the Air Force, Public Affairs, 2022. Request for propasal released for eielson air force base micro-reactor pilot program. Energy Installations and Environment.

[5] Carter, John P., 2022. Multi-Physics Investigation of a Natural Circulation Molten Salt Micro-Reactor that Utilizes an Experimental In-pile Device to Improve Core Physics and System Thermal-Hydraulic Performance. Ph.D. thesis, Univesity of Idaho.

[6] Peterson, John, 2019 8. An Analysis of the Nuclear Characteristics of a Molten Salt Microreactor. Master's thesis, University of Idaho.

[7] Roper, Robin V., Sabharwall, Piyush, Christensen, Richard, 2019. Chemical overview of molten salts. ANS Annual Meeting.
URL https://www.ans.org/pubs/transactions/article-45524/

[8] Kerlin, Thomas W., Upadhyaya, Belle R., 2019. Dynamics and Control of Nuclear Reactors. Elsevier Inc., Knoxville, Tennessee.

[9] Roper, Robin, Harkema, Megan, Sabharwall, Piyush, Riddle, Catherine, Chisholm, Brandon, Day, Brandon, Marotta, Paul, 2022. Molten salt for advanced energy applications: A review. Annals of Nuclear Energy 169, 108924. ISSN 0306-4549. doi:https://doi.org/10.1016/j.anucene.2021.108924.
URL https://www.sciencedirect.com/science/article/pii/S030645492100801X

[10] Simpson, Michael F., 2012. Development of spent nuclear fuel pyroprocessing technology at Idaho National Laboratory (INL). Technical report, Idaho National Laboratory (INL).

[11] Haubenreich, P N, Engel, J R, Prince, B E, Claiborne, H C, 1964 2. Msre design and operations report. part iii. nuclear analysis. Technical report, Oak Ridge National Laboratory. doi:10.2172/4114686.
URL https://www.osti.gov/biblio/4114686

[12] World Nuclear News, 2022. Chinese molten-salt reactor cleared for start up. World Nuclear News.
URL https://www.world-nuclear-news.org/articles/chinese-molten-salt-reactor-cleared-for-start-up

[13] Todreas, Neil E., Kazimi, Mujid S., 1990. Nuclear Systems Volume I: Thermal Hydraulic Fundamentals. Taylor and Francis, USA.

[14] Carter, John P., Christensen, Richard, Yoon, Sujong, 2022. Numerical analysis of dynamic load following response in a natural circulation molten salt power reactor system. Nuclear Engineering and Design .

[15] Roper, Robin V., Christensen, Richard, 2019. Redox potential control for the molten salt reactor concept. ANS Winter Meeting.
URL https://www.ans.org/pubs/transactions/article-47571/

[16] Andrews, Hunter B., McFarlane, Joanna, Chapel, A. Shay, Ezell, N. Dianne Bull, Holcomb, David E., de Wet, Dane, Greenwood, Michael S., Myhre, Kristian G., Bryan, Samuel A., Lines, Amanda, Riley, Brian J., Felmy, Heather M., Humrickhouse, Paul W., 2021. Review of molten salt reactor off-gas management considerations. Nuclear Engineering and Design 385, 111529. ISSN 0029-5493. doi:https://doi.org/10.1016/j.nucengdes.2021.111529.
URL https://www.sciencedirect.com/science/article/pii/S0029549321004817

[17] Roper, Robin, 2022. The Effect of Impurities and Geometry on the Corrosion and Thermodynamic Behavior of Molten Salts. Ph.D. thesis, Univesity of Idaho.

[18] Bequette, B. Wayne, 2003. Process Control: Modeling Design and Simulation. Pretice Hall, Upper Sadle River, New Jersey.

[19] Duderstadt, James J., Hamilton, Louis J., 1976. Nuclear Reactor Analysis. Wiley & Sons, New York, NY, 1st edition.

[20] Lamarsh, John R., Baratta, Anthony J., 2001. Introduction to Nuclear Engineering. Pretice Hall, Upper Sadle River, New Jersey, 3rd edition.

[21] Gahinet, Pascal M., Apkarian, Pierre, 2013. Automated tuning of gain-scheduled control systems. IEEE Conference on Decision and Control.
URL https://ieeexplore.ieee.org/document/6760297

[22] NRC, 1987. Report on the Accident at the Chernobyl Nuclear Power Station.

[23] NRC, 2007. 10 CFR §50, Appendix A, Critereon 28.

[24] DOE, 2021. The u.s. doe microreactor program. Technical report, U.S. Department of Energy.

[25] Poudel, Bikash, McJunkin, Timothy R., Kang, Ning, Reilly, James T., 2021 10. Small reactors in microgrids: Technical studies guidance. INL/EXT-21-64616 doi:10.2172/1829672.
URL https://www.osti.gov/biblio/1829672

[26] Saeed, Rami M., Shigrekar, Amey, Mikkelson, Daniel Mark, George Rigby, Aidan Christopher, Yang Hui Otani, Courtney Mariko, Garrouste, Marisol, Frick, Konor L., Bragg-Sitton, Shannon M., 2022 9. Multilevel analysis, design, and modeling of coupling advanced nuclear reactors and thermal energy storage in an integrated energy system. INL/RPT-22-69214 doi:10.2172/1890160.
URL https://www.osti.gov/biblio/1890160

[27] Engel, J.R, Steffy, R.C., 1971. Xenon behavior in the molten salt reactor experiment. U.S. Atomic Energy Commission .

[28] Peebles, F.N., 1968. Removal of xenon-135 from circulating fuel salt of the msbr by mass transfer to helium bubbles. U.S. Atomic Energy Commission .

[29] Andrews, Hunter B., McFarlane, Joanna, Chapel, A. Shay, Ezell, N. Dianne Bull, Holcomb, David Eugene, de Wet, Dane, Greenwood, Michael S., Myhre, Kristian G., Bryan, Samuel A., Lines, Amanda, Riley, Brian J., Felmy, Heather M., Humrickhouse, Paul W., 2021 10. Review of molten salt reactor off-gas management considerations. Nuclear Engineering and Design 385. doi:10.1016/j.nucengdes.2021.111529.

[30] Price, Terry J., Chvala, Ondrej, Taylor, Zack, 2019. Xenon in molten salt reactors: The effects of solubility, circulating particulate, ionization, and the sensitivity of the circulating void fraction. Nuclear Engineering and Technology .

[31] Shaffer, J. H., Grimes, W. R., Watson, G. M., 1962. Boron trifluoride as a soluble poison in molten salt reactor fuels. Nuclear Science and Engineering 12(3), 337.

[32] Al Rashdan, Ahmad, Roberson, Dakota, 2019. A frequency domain control perspective on xenon resistance for load following of thermal nuclear reactors. IEEE Transactions on Nuclear Science 66(9), 2034. doi:10.1109/TNS.2019.2934171.

[33] Ding, Zechaun, 2019. Solving bateman equation for xenon transient analysis using numerical methods. Nuclear Engineering and Technology .

[34] Campbell, Ronald Hugh, 1971 Mar. Nuclear fuel venting elements for the discharge of fission gases.

[35] Campana, Robert J., Lindgren, James R., 1974. Irradiation testing of design models for the gcfr fuel pressure equalization (vent) system. Nuclear Engineering and Design 26(1), 201. ISSN 0029-5493. doi:10.1016/0029-5493(74)90056-9. Special Issue: High Temperature Gas-Cooled Power Reactors.

[36] Cooper, Michael WD, Pastore, Giovanni, Che, Yifeng, Matthews, Christopher, Forslund, Axel, Stanek, Christopher R, Shirvan, Koroush, Tverberg, Terje, Gamble, Kyle A, Mays, Brian, Andersson, David A, 2021. Fission gas diffusion and release for cr2o3-doped uo2: From the atomic to the engineering scale. Journal of Nuclear Materials 545, 152590. ISSN 0022-3115. doi:https://doi.org/10.1016/j.jnucmat.2020.152590.
URL https://www.sciencedirect.com/science/article/pii/S0022311520311983

[37] Price, Terry J., Chvala, Ondrej, Taylor, Zack, 2019. Molten salt reactor xenon analysis: Review and decomposition. Nuclear Engineering and Radiation Science .

[38] Geankoplis, Christie J., 1993. Transport Processes and Unit Operations. Pretice Hall, Englewood Cliffs, New Jersey, 3rd edition.

[39] Bostelmann, Rike, Lo, Austin, Betzler, Benjamin, Hartanto, Donny, Wieselquist, William, 2022. MSR Reactivity, Power, and Inventory Simulations with SCALE.

[40] Blander, M, Grimes, W.R., Smith, N.V., Watson, G.M., 1958. Solubility of nobel gasses in molten fluorides ii. in the lif-naf-kf eutectic mixture. Physical Chemistry .

[41] Koning, A.J., Rochman, D., Sublet, J.-Ch., Dzysiuk, N., Fleming, M., van der Marck, S., 2019. TENDL: Complete Nuclear Data Library for Innovative Nuclear Science and Technology. doi:10.1016/j.nds.2019.01.002. Special Issue on Nuclear Reaction Data.
URL https://www.sciencedirect.com/science/article/pii/S009037521930002X

[42] Chen, Shuning, Zhang, Ao, Zou, Chunyan, Chen, Jingen, 2022. Impacts of power density on the breeding performance of molten salt reactors. International Journal of Energy Research 46(13), 18609. doi:10.1002/er.8475.

[43] Todreas, Neil E., Kazimi, Mujid S., 2001. Nuclear Systems Volume II: Elements of Thermal Hydraulic Design. Taylor and Francis, USA.

[44] Carter, J., Borrelli, R.A., 2020. Integral molen salt reactor neutron physics study using monte carlo n-particle code. Nuclear Engineering and Design 365.

[45] Rapier, Robert, 2017. The Load Following Power Plant: The New Peaker. General Electric.

[46] Stanley, Clifford J., Marshall, Frances M., 2008. Advanced test reactor - a national science user facility. International Conference on Nuclear Energy.
URL https://inldigitallibrary.inl.gov/sites/sti/sti/3991950.pdf

[47] Lee, Hyun Chul, Han, Tae Young, Lim, Hong Sik, Noh, Jae Man, 2015. An accident-tolerant control drum system for a small space reactor. Annals of Nuclear Energy 79, 143. ISSN 0306-4549. doi:https://doi.org/10.1016/j.anucene.2015.02.001.
URL https://www.sciencedirect.com/science/article/pii/S0306454915000511

[48] Root, Sam J., Christensen, Richard, 2022. Feedback control of molten salt nuclear battery. ANS Student Conference.
URL https://www.ans.org/meetings/student2022/session/view-1083/

[49] Singh, Vikram, Wheeler, Alexander M., Lish, Matthew R., Chvala, Ondrej, Upadhyaya, Belle R., 2018. Nonlinear dynamic model of molten-salt reactor experiment: Validation and operational analysis. Annals of Nuclear Energy 113, 177. ISSN 0306-4549. doi:https://doi.org/10.1016/j.anucene.2017.10.047.
URL https://www.sciencedirect.com/science/article/pii/S030645491730381X

[50] Murray, Richard, 202. Python Control Systems Library.
URL https://python-control.readthedocs.io/en/0.9.4/

[51] Sabharwall, Piyush, Ebner, Matt, Sohal, Manohar, Sharpe, Phil, Anderson, Mark, Sridharan, Kumar, Ambrosek, James, Olson, Luke, Brooks, Paul, 2010. Molten salts for high temperature reactors: University of wisconsin molten salt corrosion and flow loop experiments: Issues identified and path forward. Technical report, Idaho National Laboratory (INL). INL/EXT-10-18090.

[52] Bohlander, Thomas W., 1963. Electrical method for thawing frozen pipes. Journal AWWA 55(5), 602. doi:https://doi.org/10.1002/j.1551-8833.1963.tb01056.x.
URL https://awwa.onlinelibrary.wiley.com/doi/abs/10.1002/j.1551-8833.1963.tb01056.x

[53] Shultis, J. Kenneth, Faw, Richard E., 2002. Fundamentals of Nuclear science and Engineering. Marcel Dekker Inc., New York City, New York, 1st edition.

[54] Kok, Kenneth D., 2009. Nuclear Engineering Handbook. Taylor & Francis Group, Boca Raton, Florida, 1st edition.

[55] Tiberga, Marco, Shafer, Devaja, Lathouwers, Danny, Rohde, Martin, Kloosterman, Jan Leen, 2019. Preliminary investigation on the melting behavior of a freeze-valve for the molten salt fast reactor. Annals of Nuclear Energy 132, 544. ISSN 0306-4549. doi:https://doi.org/10.1016/j.anucene.2019.06.039.
URL https://www.sciencedirect.com/science/article/pii/S0306454919303573

[56] Murty, K. Linga, Charit, Indrajit, 2013. An Introduction to Nuclear Materials: Fundamentals and Applications. Wiley-VCH, Raliegh, NC.

[57] Wang, Shisheng, Massone, Mattia, Rineiski, Andrei, Merle-Lucotte, E., Laureau, A., GÃ©rardin, D., Heuer, D., Allibert, M., 2019. A passive decay heat removal system for emergency draining tanks of molten salt reactors. Nuclear Engineering and Design 341, 423. ISSN 0029-5493. doi:https://doi.org/10.1016/j.nucengdes.2018.11.021.
URL https://www.sciencedirect.com/science/article/pii/S0029549318309567

Appendix A: Xenon-135 Numerical Solver with Fission-Gas Stripping

Code 1: Library Imports

```
import numpy as np
from matplotlib import pyplot as plt
from tqdm import tqdm as tqdm
```

Code 2: Class Core

```
class Core: #Defining constants and allowing parameter changing (e.g. scram or power transient)
    #Constants
    #tau = 130 #sec // flow period of molten salt
    sigma_F =1e-24*180 #cm-2
    gamma ={'I':0.0639,'Xe':0.00237} #dictionaries are used to assign constants to instances of class Nuclide

    lamda ={'I':2.87e-5, 'Xe':2.09e-5} #sec^-1
    sigma ={'I':0, 'Xe':1e-24*4.3e5} #cm^2
    H ={'I':None,'Xe':1e-4} #c_g/c_l

    #FUEL... 18mol% HALEU-UF4 in FLiNaK @ 19.75% enrichment
    mole =0.18
    enrichment =0.1975
    Avo =6.0221408e23
    mass_den =3.60510 #g/cm3 @915K
    MW =35.555
    N =Avo*mass_den/MW #atoms/cm3
    basis =5*mole+2*(1-mole)
    molefrac235 =mole/basis*enrichment
    print(f'{molefrac235} molefrac U235')
    Sigma_F =sigma_F*N*molefrac235 #cm-1
    print(f'Sigma_F^f{Sigma_F} cm-1')
    #Parameters (initial)
    flux =1e14 #n/cm2/s
    FissionRate =Sigma_F*flux #cm-3.s
    print(f'Fission Rate: {FissionRate} cm-3.s')
    PowerDensity =FissionRate*200 #MeV/cm3.s
    Volume =4.29116E5 #cm3
    Power =PowerDensity*Volume*1.6022e-13 #W
    print(f'{np.sum(Power*1e-6)} MegaWatts')

    #Start-up Conditions
    phi =flux
    dnfluxes =[]
    Vdot_ratio =0
    check =False #A bool used to set up plotting
```

```
41      @classmethod
42      def scram(cls):
43          cls.dnfluxes =cls.delayed_neutron_precursors(cls.phi)
44          cls.dnfluxes.append(0)
45          print(f'significant (>1ppm) contribution from delayed neutron flux for {len(cls.
                                                                dnfluxes)} seconds')
46          cls.phi= 0 #n/cm2/s
47          cls.check =True
48          cls.eqXe =Xenon.concentration[-1]
49          cls.eqI =Iodine.concentration[-1]
50          cls.tau*=3
51
52
53      @classmethod
54      def strip(cls,strip,Power):
55          cls.Vdot_ratio =strip
56          He =8.26*strip #L/s
57          if Power =='off':
58              He /=3
59          He *=1000 #mL/s
60          He *=3600 #mL/hr
61          print(f'He flow rate of {He} mL/hr with reactor {Power}')
62
63      @classmethod #This is a method used to place the vertical dashed line in the post
                                                            scram plot
64      def cross(cls):
65          cls.check =False
66
67      @classmethod
68      def restart(cls):
69          cls.phi =0.5e14
70          cls.Vdot_ratio =0
71          cls.tau=130
72          cls.dnfluxes =[]
73
74      @classmethod
75      def delayed_neutron_precursors(cls,initial_total_flux):
76          #constants from Kerlin and Upadhyaya "Dynamics and Control of Nuclear Reactors"
77          betas =np.array([0.000221, 0.001467, 0.001313, 0.002647, 0.000771, 0.000281])
78          lambdas =np.array([0.0124, 0.0305, 0.111, 0.301, 1.14, 3.01])
79          initial_delayed_fluxes =initial_total_flux*betas
80          time_dn =np.arange(0,3601,1) #sec
81          total_delayed_flux =np.sum(initial_delayed_fluxes *np.exp(-lambdas *time_dn[:, np.
                                                                newaxis]), axis=1)
82          cutoff =initial_total_flux/1e6
83          return [flux for flux in total_delayed_flux if flux >=cutoff]
```

Code 3: Class Nuclide

```
class Nuclide: #used to create iodine and xenon objects
   nuclides=[]
   def __init__(self, element,daughter):
      #dunder-init creates an instance of the Nuclide Class and assigns it initial properties.
      self.gamma, self.lamda, self.sigma, self.H =Core.gamma[element], Core.lamda[element], Core.sigma[element], Core.H[element]
      self.daughter =daughter
      self.concentration =[0] #atoms/cm3
      self.fission=[0] #atoms/cm3.s
      self.beta=[0] #atoms/cm3.s
      self.capture=[0] #atoms/cm3.s
      self.precursor=[0] #atoms/cm3.s
      self.strip=[0]
      Nuclide.nuclides.append(self)

   def ddt_fission(self): #generation term directly from fission
      ddt_multigroup =self.gamma*Core.Sigma_F*Core.phi
      ddt =np.sum(ddt_multigroup)
      self.fission.append(ddt)
      return ddt

   def ddt_beta(self): #consumption term by beta decay -- generation term for beta daughter
      ddt =self.lamda*self.concentration[-1]
      self.beta.append(ddt)
      if self.daughter !=None:
         self.daughter.precursor.append(ddt)
      return ddt

   def ddt_capture(self): #consumption term. atom species changed by radiative capture
      ddt_5group =self.concentration[-1]*self.sigma*Core.phi
      ddt =np.sum(ddt_5group)
      self.capture.append(ddt)
      return ddt

   def ddt_strip(self): #novel - my derivation of a ddt term for single stage equilibrim stripping
      try: #calculates nuclide concentration term when valid
         ddt =self.concentration[-1]/Core.tau/(self.H/Core.Vdot_ratio+1)
      except (ZeroDivisionError, TypeError): #excludes stripping for Iodine and when helium flow rate is zero for Xenon
         ddt =0
      self.strip.append(ddt)
      return ddt
```

```
126     def timestep(self): #calls each term specific method then sums them and appends it to
                                                        the object.concentration array
127         self.ddt_fission()
128         self.ddt_beta()
129         self.ddt_capture()
130         self.ddt_strip()
131         ddt =self.fission[-1]+self.precursor[-1]-self.beta[-1]-self.capture[-1]-self.strip
                                                                [-1]
132         new =self.concentration[-1] +ddt
133         self.concentration.append(new)
134         return ddt
135 
136     @classmethod
137     def numpy(cls): #converts the concentration lists to numpy arrays for ease of
                                                        plotting
138         for nuclide in cls.nuclides:
139             nuclide.concentration =N2C(np.array(nuclide.concentration))
140 
141     @classmethod
142     def reset(cls):
143         for nuclide in cls.nuclides:
144             nuclide.concentration =[0] #atoms/cm3
145             nuclide.fission=[0] #atoms/cm3.s
146             nuclide.beta=[0] #atoms/cm3.s
147             nuclide.capture=[0] #atoms/cm3.s
148             nuclide.precursor=[0] #atoms/cm3.s
149             nuclide.strip=[0]
```

Code 4: Solver Functions

```
def N2C(Ni):
    Ci =1e6*Ni/(Core.Avo/1000)
    return Ci

def N2rho(N):
    nu =2.43
    rho =(Xenon.sigma*N)/(nu*Core.Sigma_F)
    return rho

def SS(x):
    x_squared =np.square(x)
    return np.sum(x_squared)

def r2_score(analytical,numerical):
    RES =analytical-numerical
    TOT =analytical-np.mean(analytical)
    return 1-SS(RES)/SS(TOT)

def riseI(Io,Ieq,t):
    C =Io-Ieq
    I =C*np.exp(-Iodine.lamda*t)+Ieq
    return I

def riseXe(Xeo,Xeeq,t):#Only works when Io is set to Ieq from the beginning
    C =Xeo -Xeeq
    k =Xenon.lamda +Xenon.sigma*Core.phi
    Xe =C*np.exp(-k*t) +Xeeq
    return Xe

def scramI(Ieq,t):
    I =Ieq*np.exp(-Iodine.lamda*t)
    return I

def scramXe(Xeeq,Ieq,t):
    Xe1 =Xeeq*np.exp(-Xenon.lamda*t)
    ratio =Iodine.lamda/(Iodine.lamda-Xenon.lamda)
    Xe2 =Ieq*ratio*(np.exp(-Xenon.lamda*t)-np.exp(-Iodine.lamda*t))
    return Xe1+Xe2
```

Code 5: Instantiate Solver

```
Xenon =Nuclide('Xe',None)
Iodine =Nuclide('I',Xenon)
Core.restart()

infI =Iodine.gamma*Core.Sigma_F*Core.phi/Iodine.lamda
infXe =(Iodine.gamma+Xenon.gamma)*np.sum(Core.Sigma_F*Core.phi)/(Xenon.lamda+np.sum(
                                                    Xenon.sigma*Core.phi))

print(f'Equilibrium Concentrations')
print(f'{N2C(infI)} millimol/m3 I-135')
print(f'{N2C(infXe)} millimol/m3 Xe-135')
print(f'Xenon is at {infXe/infI*100}% of the Iodine level')
print(f'Poison Reactivity: {round(N2rho(infXe)*100,2)}%')

print(f'Initial Scram Rates')
print(f'{N2C(-infI*Iodine.lamda)} millimol/m3-s I-135')
print(f'{N2C(infI*Iodine.lamda-infXe*Xenon.lamda)} millimol/m3-s Xe-135')

tmax =np.log(Xenon.lamda/Iodine.lamda**2*infXe/infI*(Iodine.lamda-Xenon.lamda)+Xenon.
                                                    lamda/Iodine.lamda)/(Xenon.lamda-Iodine.
                                                    lamda)

Xe_max =(infXe)*np.exp(-Xenon.lamda*tmax)+(infI)*(Iodine.lamda)/(Iodine.lamda-Xenon.
                                                    lamda)*(np.exp(-Xenon.lamda*tmax)-np.exp(
                                                    -Iodine.lamda*tmax))


print(f'Peak time: {round(1/3600*tmax,2)} hr')
print(f'Peak level: {N2C(Xe_max)} millimol/m3 Xe-135')
print(f'Peak Poison Reactivity: {round(N2rho(Xe_max)*100,2)}%')

dt=1 #sec
begin =0 #hours
scram =100 #hours
peak =5 #hours
end =165 #hours
time =np.arange(begin*3600,end*3600+1,dt) #sec
```

Code 6: Run Solver

```
Core.restart()
Nuclide.reset()
for t in tqdm(time):
    Iodine.timestep()
    Xenon.timestep()

    if t==scram*3600:
        Core.scram()

    if t==(scram+peak)*3600:
        Core.strip(1.725e-5,'off')

    if Core.check and Xenon.concentration[-1] <Core.eqXe:
        Core.cross()
        eqt=t

#convert to numpy arrays for ease of plotting
Nuclide.numpy()
Core.eqXe =N2C(Core.eqXe)

print(f'Return to equilibrium at {round(eqt-(scram+peak)*3600,1)} seconds since call for
                                                    power')

ax =plt.gca()
plt.xlabel('time since scram, t (hr)')
plt.ylabel(f'Concentration (millimol/m\N{SUPERSCRIPT THREE})')
plt.xlim(-24,60)
plt.xticks([-12,0,12,24,36,48])
plt.plot(time[::3600]/3600-scram,Xenon.concentration[::3600],label='Xenon 135')
plt.plot(time[::3600]/3600-scram,Iodine.concentration[::3600],label='Iodine 135')
plt.axhline(Core.eqXe,linestyle='--')
plt.axvline(eqt/3600-scram,linestyle='--')
plt.grid()
plt.legend(loc='best')

plt.show()
```

Appendix B: Serpent Deck Writing Script

Code 7: Control Drum Rotating

```
#./drum.py
def card(deg):
   with open('cards/drum.txt','w') as drumcards:
      drumcards.write(f'% Control drum absorber pads, numbered clockwise from 9 o clock\
                                                        n')
      drumcards.write(f'surf 80 pad -45.05 0 16 17 {deg-45} {deg+45} % drum 1\n')
      drumcards.write(f'surf 81 pad -31.8551605 31.8551605 16 17 {deg-0} {deg+90} % drum
                                                          2\n')
      drumcards.write(f'surf 82 pad 0 45.05 16 17 {deg+45} {deg+135} % drum 3\n')
      drumcards.write(f'surf 83 pad 31.8551605 31.8551605 16 17 {deg+90} {deg+180} %
                                                        drum 4\n')
      drumcards.write(f'surf 84 pad 45.05 0 16 17 {deg+135} {deg+225} % drum 5\n')
      drumcards.write(f'surf 85 pad 31.8551605 -31.8551605 16 17 {deg+180} {deg+270} %
                                                        drum 6\n')
      drumcards.write(f'surf 86 pad 0 -45.05 16 17 {deg-135} {deg-45} % drum 7\n')
      drumcards.write(f'surf 87 pad -31.8551605 -31.8551605 16 17 {deg-90} {deg+0} %
                                                        drum 8\n')
```

Code 8: Salt Composition

```
#./salt.py
def uranium(enrich):
   total =1.40246e-2 +5.69859e-2
   u235 =total*enrich/100
   u238 =total -u235
   return '{:.5e}'.format(u235), '{:.5e}'.format(u238)

def card(enrich):
   u235, u238 =uranium(enrich)
   with open('cards/salt.txt','w') as saltcards,\
       open ('cards/salttemps.txt','r') as temps:

      saltcards.write('%_____Material Cards_____\n')
      saltcards.write('%LiF-NaF-KF-UF4\n')
      saltcards.write(f'%18 mole percent UF4 at {enrich} percent enrichment\n')

      for line in temps:
         saltcards.write(line)
         with open('cards/flinak.txt','r') as flinak:
            for l in flinak:
               saltcards.write(l)
         saltcards.write(f'92235.06c {u235}\n')
         saltcards.write(f'92238.06c {u238}\n')
```

Code 9: Deck Writing Script

```
#./makedeck.py
import datetime,os
import salt, drum
now =datetime.date.today()

deg =112#float(input('')), take input to interact with Bash
enrich =19.75 #percentage
folder =f'./results/{deg}'

if not os.path.exists(folder):
   os.makedirs(folder)

print(folder)
salt.card(enrich)
drum.card(deg)

with open(f'{folder}/MSNB', 'w') as deck,\
    open('cards/cell.txt','r') as cell,\
    open('cards/surface.txt','r') as surface,\
    open('cards/drum.txt','r') as drum,\
    open('cards/salt.txt','r') as salt,\
    open('cards/material.txt','r') as material,\
    open('cards/physics.txt','r') as physics,\
    open('cards/plot.txt','r') as plot:

   deck.write(f'set title "Root MSNB at {deg} Degrees, {enrich} percent enrichment"\n')
   deck.write(f'%Revised {now}\n%\n')

   for line in cell:
      deck.write(line)
   for line in surface:
      if not 'WRITE_PADS' in line:
         deck.write(line)
      else:
         for l in drum:
            deck.write(l)
   for line in salt:
      deck.write(line)
   for line in material:
      deck.write(line)
   for line in physics:
      deck.write(line)
   for line in plot:
      deck.write(line)
```

Code 10: Cell Cards

```
%./cards/cell.txt
% ___________________CELL CARDS_____________________
%
% Fuel in Inlet, Outlet, and Downcomer
cell 101 0 Salt0 13 -16 45 -46 101 102 %Fuel in Downcomer
cell 102 0 Salt0 12 -13 41 -46 101 102 %Fuel in Lower Spoke
cell 103 0 Salt0 12 -13 -41 % Fuel Lower Plenum
cell 104 0 Salt9 -18 31 -40 %Fuel in Riser
cell 105 0 Salt9 18 -19 -40 %Fuel in Upper Plenum
cell 106 0 Salt9 18 -19 -46 40 101 102 60 61 62 63 64 65 66 67 % Fuel in Upper Spoke
cell 107 0 Salt0 13 -14 -40 101 102 % Fuel passing thru Lower Orifice Plate
cell 108 0 Salt9 -31 15 -40 101 102 % Fuel passing thru Lower Orifice Plate
%
% Fuel in Core
cell 111 0 Salt0 14 -21 -40 103 104
cell 112 0 Salt1 21 -22 -40 103 104
cell 113 0 Salt2 22 -23 -40 103 104
cell 114 0 Salt3 23 -24 -40 103 104
cell 115 0 Salt4 24 -25 -40 103 104
cell 116 0 Salt5 25 -26 -40 103 104
cell 117 0 Salt6 26 -27 -40 103 104
cell 118 0 Salt7 27 -28 -40 103 104
cell 119 0 Salt8 28 -15 -40 103 104
%
%Fuel in HEX
cell 121 0 Salt1 16 -31 43 -46 101 102
cell 122 0 Salt2 31 -32 43 -46 101 102
cell 123 0 Salt3 32 -33 43 -46 101 102
cell 124 0 Salt4 33 -34 43 -46 101 102
cell 125 0 Salt5 34 -35 43 -46 101 102
cell 126 0 Salt6 35 -36 43 -46 101 102
cell 127 0 Salt7 36 -37 43 -46 101 102
cell 128 0 Salt8 37 -38 43 -46 101 102
cell 129 0 Salt9 38 -18 43 -46 101 102
%
% Structure
cell 200 0 SS304 11 -12 -46 % Base
cell 201 0 SS304 13 -14 -45 40 % Reflector Bottom
cell 202 0 SS304 14 -15 44 -45 % Downcomer Pipe
cell 203 0 SS304 15 -16 42 -45 % Reflector Topper
cell 204 0 SS304 17 -18 40 -43 60 61 62 63 64 65 66 67 % Chimney Absorber Topper
cell 205 0 SS304 16 -17 42 -43 % Chimney Absorber Pipe
cell 206 0 SS304 10 -19 46 -47 % Reactor Vessel Pipe
cell 207 0 SS304 19 -20 -47 60 61 62 63 64 65 66 67 %Lid
%
```

```
% Reflector/Absorber
cell 300 0 BeO 10 -11 -46 % Bottom Reflector
cell 301 0 BeO 15 -17 40 -41 % Chimney Reflector
cell 302 0 BeO 14 -15 40 -48 90 91 92 93 94 95 96 97 % Interstitial Reflector
cell 303 0 GRPH 14 -15 48 -44 % Safety Reflector Outer
cell 304 0 B4C 15 -17 -42 41 60 61 62 63 64 65 66 67 % Chimney Absorber
cell 305 0 BeO 14 -15 -40 (-103:-104) %In Core Moderator
cell 306 0 BeO 15 -31 (-101:-102) -40 %Core Top Orifice Plate
%
% Control Drum Reflectors
cell 401 0 BeO 14 -15 -90 80 60 % drum 1
cell 402 0 BeO 14 -15 -91 81 61 % drum 2
cell 403 0 BeO 14 -15 -92 82 62 % drum 3
cell 404 0 BeO 14 -15 -93 83 63 % drum 4
cell 405 0 BeO 14 -15 -94 84 64 % drum 5
cell 406 0 BeO 14 -15 -95 85 65 % drum 6
cell 407 0 BeO 14 -15 -96 86 66 % drum 7
cell 408 0 BeO 14 -15 -97 87 67 % drum 8
%
% Control Drum Absorbers
cell 501 0 B4C 14 -15 -80 60 % drum 1
cell 502 0 B4C 14 -15 -81 61 % drum 2
cell 503 0 B4C 14 -15 -82 62 % drum 3
cell 504 0 B4C 14 -15 -83 63 % drum 4
cell 505 0 B4C 14 -15 -84 64 % drum 5
cell 506 0 B4C 14 -15 -85 65 % drum 6
cell 507 0 B4C 14 -15 -86 66 % drum 7
cell 508 0 B4C 14 -15 -87 67 % drum 8
%
% Control Drum Drive Shafts
cell 601 0 SS304 -60 70 14 -20 % driveshaft tube 1
cell 602 0 SS304 -61 71 14 -20 % driveshaft tube 2
cell 603 0 SS304 -62 72 14 -20 % driveshaft tube 3
cell 604 0 SS304 -63 73 14 -20 % driveshaft tube 4
cell 605 0 SS304 -64 74 14 -20 % driveshaft tube 5
cell 606 0 SS304 -65 75 14 -20 % driveshaft tube 6
cell 607 0 SS304 -66 76 14 -20 % driveshaft tube 7
cell 608 0 SS304 -67 77 14 -20 % driveshaft tube 8
cell 701 0 NiCrMo -70 14 -20 % driveshaft tube 1 fill
cell 702 0 NiCrMo -71 14 -20 % driveshaft tube 2 fill
cell 703 0 NiCrMo -72 14 -20 % driveshaft tube 3 fill
cell 704 0 NiCrMo -73 14 -20 % driveshaft tube 4 fill
cell 705 0 NiCrMo -74 14 -20 % driveshaft tube 5 fill
cell 706 0 NiCrMo -75 14 -20 % driveshaft tube 6 fill
cell 707 0 NiCrMo -76 14 -20 % driveshaft tube 7 fill
cell 708 0 NiCrMo -77 14 -20 % driveshaft tube 8 fill
%
```

```
% Flow separators
cell 800 0 SS304 -101 12 -13 41 -46 % lower spoke dividers 1
cell 801 0 SS304 -102 12 -13 41 -46 % lower spoke dividers 2
cell 802 0 SS304 -101 13 -16 45 -46 % downcomer surround dividers 1
cell 803 0 SS304 -102 13 -16 45 -46 % downcomer surround dividers 2
cell 804 0 SS304 -101 16 -18 43 -46 % heat exchanger dividers 1
cell 805 0 SS304 -102 16 -18 43 -46 % heat exchanger dividers 2
cell 806 0 SS304 -101 18 -19 40 -46 % upper spoke dividers 1
cell 807 0 SS304 -102 18 -19 40 -46 % upper spoke dividers 2
cell 808 0 BeO (-101:-102) 13 -14 -40 % core bottom orifice plate dividers 1
%
% Outside
cell 900 0 Conc -10 -100 %  below vessel
cell 901 0 Conc 10 47 -20 -100 %  around vessel
cell 902 0 Air 20 -100 %  above vessel
cell 999 0 outside 100 %void
%
```

Code 11: Surface Cards

```
%./cards/surface.txt
% ___________________SURFACE CARDS_____________
%
% Reactor planes
surf 10 pz -11 % Bottom
surf 11 pz -1 % Reactor Vessel base bottom
surf 12 pz 0 % Reactor Vessel base top
surf 13 pz 12 % Reflector clad bottom
surf 14 pz 13 % Reflector clad top
surf 15 pz 179 % core/chimney divider
surf 16 pz 180 % step clad upper
surf 17 pz 215 % Spoke clad bottom
surf 18 pz 216 % Spoke clad top
surf 19 pz 228 % Lid top
surf 20 pz 231 % Lid top
%
% Core Stratification
surf 21 pz 31.4444 %top of slice 1
surf 22 pz 49.8889 %top of slice 2
surf 23 pz 69.3333 %top of slice 3
surf 24 pz 86.7778 %top of slice 4
surf 25 pz 105.2222 %top of slice 5
surf 26 pz 123.6667 %top of slice 6
surf 27 pz 142.1111 %top of slice 7
surf 28 pz 160.5556 %top of slice 8
%
```

```
% HEX Stratification
surf 31 pz 184 %top of slice 1
surf 32 pz 188 %top of slice 2
surf 33 pz 192 %top of slice 3
surf 34 pz 196 %top of slice 4
surf 35 pz 200 %top of slice 5
surf 36 pz 204 %top of slice 6
surf 37 pz 208 %top of slice 7
surf 38 pz 212 %top of slice 8
%
% Reactor cylinders
surf 40 cyl 0 0 25 % riser inner
surf 41 cyl 0 0 36.5 % chimney reflector outer
surf 42 cyl 0 0 48 % chimney absorber outer
surf 43 cyl 0 0 49 % absorber liner
surf 44 cyl 0 0 66 % safety absorber outer
surf 45 cyl 0 0 67 % reflector liner
surf 46 cyl 0 0 69.5 % RV inner
surf 47 cyl 0 0 72.5 % RV outer
surf 48 cyl 0 0 63.5 % reflector outer
surf 49 cyl 0 0 2.0 % AS center stem
%
% Control drum driveshafts
surf 60 cyl -45.05 0 2.5 % drum drive tube outer 1
surf 70 cyl -45.05 0 2 % drum drive tube inner 1
surf 61 cyl -31.8551605 31.8551605 2.5 % drum drive tube outer 2
surf 71 cyl -31.8551605 31.8551605 2 % drum drive tube inner 2
surf 62 cyl 0 45.05 2.5 % drum drive tube outer 3
surf 72 cyl 0 45.05 2 % drum drive tube inner 3
surf 63 cyl 31.8551605 31.8551605 2.5 % drum drive tube outer 4
surf 73 cyl 31.8551605 31.8551605 2 % drum drive tube inner 4
surf 64 cyl 45.05 0 2.5 % drum drive tube outer 5
surf 74 cyl 45.05 0 2 % drum drive tube inner 5
surf 65 cyl 31.8551605 -31.8551605 2.5 % drum drive tube outer 6
surf 75 cyl 31.8551605 -31.8551605 2 % drum drive tube inner 6
surf 66 cyl 0 -45.05 2.5 % drum drive tube outer 7
surf 76 cyl 0 -45.05 2 % drum drive tube inner 7
surf 67 cyl -31.8551605 -31.8551605 2.5 % drum drive tube outer 8
surf 77 cyl -31.8551605 -31.8551605 2 % drum drive tube inner 8
%
WRITE_PADS %This line tells the python code to write the pads
%
```

```
% Control drums, likewise
surf 90 cyl -45.05 0 17   % drum 1
surf 91 cyl -31.8551605 31.8551605 17  % drum 2
surf 92 cyl 0 45.05 17  % drum 3
surf 93 cyl 31.8551605 31.8551605 17 % drum 4
surf 94 cyl 45.05 0 17 % drum 5
surf 95 cyl 31.8551605 -31.8551605 17  % drum 6
surf 96 cyl 0 -45.05 17 % drum 7
surf 97 cyl -31.8551605 -31.8551605 17 % drum 8
%
% Flow separators
surf 101 cross 0 0 75.5 3
surf 102 cross 0 0 75.5 3
strans 101 0 0 1 0 0 22.5
strans 102 0 0 1 0 0 -22.5
%In Core Moderator
surf 103 cross 0 0 27 1
surf 104 cross 0 0 27 1
strans 103 0 0 1 0 0 22.5
strans 104 0 0 1 0 0 -22.5
%
%Extremities
surf 100 sph 0 0 0 300
%
```

Code 12: Surface Card for Drums

```
% Control drum absorber pads, numbered clockwise from 9 o clock
surf 80 pad -45.05 0 16 17 67 157 % drum 1
surf 81 pad -31.8551605 31.8551605 16 17 112 202 % drum 2
surf 82 pad 0 45.05 16 17 157 247 % drum 3
surf 83 pad 31.8551605 31.8551605 16 17 202 292 % drum 4
surf 84 pad 45.05 0 16 17 247 337 % drum 5
surf 85 pad 31.8551605 -31.8551605 16 17 292 382 % drum 6
surf 86 pad 0 -45.05 16 17 -23 67 % drum 7
surf 87 pad -31.8551605 -31.8551605 16 17 22 112 % drum 8
```

Code 13: Material Cards for Molten Salt

```
%_____Material Cards_____
%LiF-NaF-KF-UF4
%18 mole percent UF4 at 19.75 percent enrichment
mat Salt0 -3.643499 tmp 873 rgb 197 0 0
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt1 -3.628150 tmp 889.66 rgb 204 0 0
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt2 -3.620470 tmp 897.99 rgb 217 0 0
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt3 -3.612788 tmp 906.32 rgb 230 0 0
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt4 -3.605102 tmp 914.65 rgb 242 0 0
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt5 -3.597414 tmp 922.98 rgb 255 13 13
3006.06c 1.49954e-5
```

```
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt6 -3.589722 tmp 931.31 rgb 255 26 26
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt7 -3.582028 tmp 939.64 rgb 255 39 39
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt8 -3.574330 tmp 947.97 rgb 255 51 51
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
mat Salt9 -3.566629 tmp 956.3 rgb 255 64 64
3006.06c 1.49954e-5
3007.06c 1.49939e-1
9019.06c 6.06524e-1
19039.06c 1.26311e-1
19041.06c 9.11554e-3
11023.06c 3.70854e-2
92235.06c 1.40246e-02
92238.06c 5.69859e-02
```

Code 14: Material Cards

```
% ./cards/material.txt
% 304 Stainless Steel
mat SS304 -7.5983 tmp 923 rgb 192 192 192
26054.06c -0.037589 %54-Fe
26056.06c -0.611384 %56-Fe
26057.06c -0.014379 %57-Fe
26058.06c -0.001932 %58-Fe
7014.06c -0.000995 %14-N
7015.06c -0.000004 %15-N
28058.06c -0.070485 %58-Ni
28060.06c -0.028087 %60-Ni
28061.06c -0.001241 %61-Ni
28062.06c -0.004025 %62-Ni
28064.06c -0.001058 %64-Ni
24050.06c -0.008338 %50-Cr
24052.06c -0.167231 %52-Cr
24053.06c -0.019327 %53-Cr
24054.06c -0.004902 %54-Cr
14028.06c -0.006881 %28-Si
14029.06c -0.000363 %29-Si
14030.06c -0.000248 %30-Si
16032.06c -0.000284 %32-S
16033.06c -0.000002 %33-S
16034.06c -0.000014 %34-S
15031.06c -0.000020 %31-P
25055.06c -0.019980 %55-Mn
6000.06c -0.000799 %(Carbon)
%
% Graphite
mat GRPH -2.12 tmp 923 moder
grph 6000 rgb 250 250 0
6000.06c -1 %(Carbon)
therm grph 1010 gre7.00t gre7.26t
%
% Boron Carbide (10-B enriched) Absorber
mat B4C -2.52 tmp 923 rgb 110 240 80
5010.06c 0.8 %10-B
%5011.06c 0.4 %11-B
6000.06c 0.2 %(Carbon)
%
% "Air"
mat Air -0.001205 tmp 300 rgb 255 170 220
6000.50c 0.000150
7014.50c 0.784431
8016.50c 0.210748
18000.59c  0.004671
%
```

```
% Beryllium Oxide Reflector
mat BeO -3.01 tmp 923 rgb 110 190 250
4009.06c 0.5
8016.06c 0.5
%
% Hastelloy CHIMNEY REFLECTOR
mat NiCrMo -8.89 tmp 956 rgb 160 160 160
28058.06c -0.48
24052.06c -0.07
26056.06c -0.05
14028.06c -0.01
28060.06c -0.23
42092.06c -0.02344
42094.06c -0.014704
42095.06c -0.025392
42096.06c -0.026672
42097.06c -0.015328
42098.06c -0.038864
42100.06c -0.0156
%
% Barite Concrete
mat Conc -3.35 tmp 600 rgb 90 100 90
1001.06c 0.109602
8016.06c 0.600189
12000.06c 0.001515
13027.06c 0.004777
14000.06c 0.011473
16000.06c 0.103654
20000.06c 0.038593
26000.06c 0.026213
56130.06c 0.000114381
56132.06c 0.000103983
56134.06c 0.002516389
56135.06c 0.00685248
56136.06c 0.008162666
56137.06c 0.011677291
56138.06c 0.074555811
%
```

Code 15: Physics Cards

```
%./cards/physics.txt
% ____________________Physics cards_____
set power 1e7 %10MWth
set pop 1000000 500 100 1
%dep daystep 1 1 1 1 1
%set mcvol 10000000
%set nbuf 10
%set printm 1 1e-10
%set inventory "all"
%set pcc leli 10 10
set acelib "endfb71r1_p2" "endfb71r1" "jeff31u"
%set declib "sss_jeff31.dec"
%set nfylib "sss_jeff31.nfy"
det EnergyDetector dm Salt4 de EnergyGrid
ene EnergyGrid 3 500 1e-11 2e1
```

Code 16: Plotter Cards

```
%./cards/plot.txt
% ____________________Plotter_________
%
plot 22 1080 1800 [0 -100 100 -20 320] %Axial
plot 31 640 640 [6 -100 100 -100 100] %Lower Plenum/Spoke
plot 31 640 640 [12.5 -100 100 -100 100] %Lower Orifice
plot 31 1080 1080 [100 -100 100 -100 100] %Core
plot 31 640 640 [179.5 -100 100 -100 100] %HEX Bottom/Upper Orifice
plot 31 640 640 [186 -100 100 -100 100] %Heat Exchanger
plot 31 640 640 [215.5 -100 100 -100 100] %Heat Exchagner Top
plot 31 640 640 [220 -100 100 -100 100] %Upper Plenum/Spoke
```

Appendix C: MSNB Transient Simulator

Code 17: MSNB Simulator

```
#./simulator.py
import os; os.system('cls')
print('import libraries')
import initial, params, loop, plots, functions, TempProfile, controller
import numpy as np
#from scipy.signal import argrelextrema
from tqdm import tqdm as tqdm

print('refresh simulation environment')
plots.kill()

#Calculate Initial Conditions
'''
The algorithm function in the file initial.py uses binary search to identify the cold
                                                temperature that when paired with a hot
                                                temperature of 700 C gives the desired
                                                power output. It uses the natural
                                                circulation flow rate, heat capacity, and
                                                 temperature difference to calculate the
                                                power for a given cold temperature, and
                                                adjusts the cold temperature as needed
                                                based on the result.
'''
Q =8000 #kW
if Q <=0:
   print('The reactor power is calculated using an exponential. Use a positive initial
                                                    power. This code cannot simulate
                                                    bringing up to hot-standby')
   exit()
print('determine initial conditions')
T_hot,T_cold,Q0,v =initial.algorithm(Q)
print(f'T_hot: {T_hot} degC')
print(f'T_cold: {T_cold} degC')
print(f'Power: {Q0} kW')
print(f'Velocity: {round(v*100,1)} cm/s')
```

```
#Define T(x)
'''
Calls function initial from file TempProfile.py, which assumes linear heat increase in
                                                the core, constant hot temperature in the
                                                 riser/chimney, linear heat decrease in
                                                the heat exchanger, and constant cold
                                                temperature in the downcomer. It then
                                                plots the initial temperature profile
'''
print('set up temperature profile')
#T_x = TempProfile.(T_hot, T_cold)
T_x =TempProfile.initial(T_hot, T_cold)
plots.x_vs_Tx("img/animateTx_t/t-0.png",0.0,T_x,T_cold,T_hot)
#exit()
#Initialize HEX transient
'''
Q0 is calculated above, Q1 is the power after the first power change, Q2 is the power
                                                after the second power change. tlen is
                                                the total time in seconds of the
                                                simulation (3600sec = 1hr). t01 is the
                                                time over which the first power change
                                                occurs (ramp function ~ use t01 = 0 for
                                                step response). t1 is the time in seconds
                                                 for which the reactor is held at Q1. t12
                                                 is the time in seconds over which the
                                                second power change occurs. t2 is
                                                calculated and is the time between the
                                                end of the second power change and the
                                                end of the simulation
'''
Q1,Q2 =10000,8000
#Q1,Q2 = Q0,Q0

t0,t01,t1,t12,t2 =300,300,600,300,600
times =(0,t0,t01,t1,t12,t2)
tlen= t0+t01+t1+t12+t2

print('set up HEX transient')
t=np.arange(0,tlen+1,params.dt)
Q00 =Q0**np.ones(t0)
Q01 =np.linspace(Q0,Q1,num=t01)
Q11 =Q1*np.ones(t1)
Q12 =np.linspace(Q1,Q2,num=t12)
Q22 =Q2*np.ones(t2+1)
Qhex_t =np.concatenate((Q00,Q01,Q11,Q12,Q22))
pf_tau =len(loop.xdowncomer)/functions.base_to_milli(v)
Qcore_SP =list(controller.prefilter(Qhex_t,t,pf_tau))
plots.t_vs_Q(t,Qhex_t,None,Qcore_SP)
```

```
59 #FEM
60 '''
61 With the problem defined, the 1D+time finite element model is set-up. Time arrays are
                                                    initialized for the core power, flow
                                                    velocity, and temperature profile, as
                                                    well as reactivity. The flow reactivity
                                                    is calculated, while the temperature
                                                    reactivity is set to the inverse, on the
                                                    assumption that the reactor is initially
                                                    critical. The control drums will start at
                                                     the bias point, so their reactivity is
                                                    null. Then the period and reactivity rate
                                                     of change are set to zero, again on the
                                                    steady state critical assumption.
62 '''
63 print('set up FEM')
64 Qcore_t= [Q0]
65 v_t =[v]
66 T_x_t =[T_x]
67
68 Freac_t =[functions.FlowRxty(T_x)]
69 Treac_t =[-Freac_t[0]]
70
71 error =0
72 CDtheta_t =[controller.drum(error)]
73 Creac_t =[controller.angle2reac(CDtheta_t[0])]
74
75 reac_t=[Freac_t[0]+Treac_t[0]+Creac_t[0]]
76 reac_dot_t =[0]
77 exponent=[0]
78
79 print('run simulation')
```

```
'''
The time loop makes the simulator calculate the new values necessary to advance the
                                                    temperature profile. Descriptions of the
                                                    functions called are included in the
                                                    functions.py file.
'''
for step in tqdm(t[1:]):
    T_x =TempProfile.advance(T_x, v ,Qcore_t[-1], Qhex_t[step-1])
    vnew =functions.Velo(T_x)
    v_t.append(vnew)
    T_x_t.append(T_x)

    Freac_t.append(functions.FlowRxty(T_x))
    Treac_t.append(Treac_t[-1] +functions.TempRxtyChange(T_x_t[-2],T_x_t[-1]))

    CDtheta_t.append(controller.drum(Qcore_SP[step]-Qcore_t[-1]))
    Creac_t.append(controller.angle2reac(CDtheta_t[-1]))

    reac_t.append(Freac_t[-1]+Treac_t[-1]+Creac_t[-1])
    reac_dot_t.append(functions.RoC(reac_t[-2], reac_t[-1]))

    '''
    the reactor period is defined as 0 if the new overall reactivity is 0, otherwise, it
                                                        is calculated using the neutron
                                                        generation time (prompt effects) and
                                                        reactivity rate of change (delayed
                                                        neutron effects).
    '''
    if reac_t[-1]==0:
    #if 1==1:
        exponent.append(0)
    else:
        tau=params.lstar/reac_t[-1]+(params.beta_eff-reac_t[-1])/(params.lam*reac_t[-1]+
                                                            reac_dot_t[-1])
        exponent.append(params.dt/tau)

    '''
    The new power is calculated using the reactor period and the previous power.
    '''
    Q=Qcore_t[-1]*np.exp(exponent[-1])
    Qcore_t.append(Q)
```

```
113
114 print('plotting time arrays')
115 plots.t_vs_Q(t,Qhex_t,Qcore_t,Qcore_SP)
116 plots.t_vs_reac(t,Freac_t,Treac_t,reac_t)
117 plots.t_vs_exp(t,exponent)
118 plots.t_vs_velo(t,v_t)
119 plots.t_vs_angle(t,CDtheta_t)
120 plots.auto_reac_phase(Freac_t,Treac_t,times)
121 plots.contr_reac_phase(Freac_t,Treac_t,Creac_t,times)
122
123 print('plotting temperature profiles')
124 Tmin =np.min(np.array(T_x_t))
125 Tmax =np.max(np.array(T_x_t))
126
127 exit()
128
129 for step, T_x in tqdm(zip(t[1:],T_x_t),total=len(T_x_t)):
130    if step*params.dt%60 ==0:
131       path ="img/animateTx_t/t-" +str(int(round(step/60,0))) +".png"
132       plots.x_vs_Tx(path, step, T_x, Tmin, Tmax)
133 plots.gif('img/animateTx_t')
134
135 exit()#remove if you want to print local mins/maxes
136
137 maximums =argrelextrema(np.array(Qcore_t),np.greater)
138 minimums =argrelextrema(np.array(Qcore_t),np.less)
139
140 print('max', np.max(np.array(Qcore_t)))
141 print('local maximums')
142 for max in maximums:
143    for i in max:
144       print(round(t[int(i)]/60),Qcore_t[int(i)])
145
146 print('min', np.min(np.array(Qcore_t)))
147 print(' local minimums')
148 for min in minimums:
```

Code 18: MSNB Simulator Functions

```
#./functions.py
import params, loop, TempProfile
import numpy as np

#Basic Calculations
def absT(T): #input Celcius
   return T+273.15 #output Kelvin

def base_to_kilo(x): #input base
   return x/1000 #output kilo

def kilo_to_base(x): #input kilo
   return x*1000 #output base

def base_to_centi(x): #input base
   return x*100 #output centi

def base_to_milli(x): #input base
   return x*1000 #output milli

def list_ave(list): #input list
   return sum(list)/len(list) #output float

def RoC(x1,x2): #time rate of change
   return (x2-x1)/params.dt

#Physical Properties
def cp(T): #input Celcius
   T=absT(T) #Convert T to Kelvin
   return (1.0634*T+976.78)/1000 # output kJ/kg-K

def density(T): #input Celcius
   T=absT(T) #Convert T to Kelvin
   return 1000*(4.6820365-T*9.4601046e-4) #output kg/m3\

def T2mu(T):
   Tref =600
   m_x =density(T)*(params.Ax*.001) #kg
   u_x =cp((T+Tref)/2)*(T-Tref)
   E_x =m_x*u_x
   return E_x

TempArray =np.linspace(600,800,num=1000)
EnergyArray =T2mu(TempArray)
coeff =np.polyfit(EnergyArray,TempArray,6)

def mu2T(E):
   value =np.zeros(len(E))
   for c in coeff:
      value =value*E+c
   return value
```

```
#System Quantities
def DiffP(T_x): #input Celcius
    hot=density(list_ave(TempProfile.hotleg(T_x)))
    cold=density(list_ave(TempProfile.coldleg(T_x))) #calculate average density of cold
                                                     leg
    return (cold-hot)*params.h*9.81 #output Pa

def Velo(T_x): #input Celcius
    DrivingForce=DiffP(T_x) #Convert Differential Pressure
    rho=density(list_ave(T_x)) #Calculate Average Density of entire Loop
    v_squared =(2*DrivingForce)/(params.xi*rho)
    if v_squared<0.01**2: v_squared=0.01**2
    if v_squared>0.15**2: v_squared=0.15**2
    v =np.sqrt(v_squared) #output m/s
    return v# 0.05 #m/s

#Reactivity
def TempRxtyChange(previous,current): #input Celcius
    T1=list_ave(TempProfile.core(previous))
    T2=list_ave(TempProfile.core(current))
    dT=T2-T1 #convert T from Celcius to Kelvin
    return params.alphaT*dT #unitless

L =loop.Ri2-loop.Ro #calculate out of core path length
H =loop.Ro-loop.Ri1 #calculate in core path length
def FlowRxty(T_x): #input Celcius
    v=base_to_centi(Velo(T_x))
    # #calculate fluid velocity
    rho =-(L/(L+H))*params.beta_eff*(1-np.exp(-params.alphaF*v)) #unitless
    return rho
```

Code 19: MSNB Flow Loop

```
#./loop.py
import numpy as np
#Linear Path Length (mm)
'''Define the ends of all of the regimes'''
Ri1 =0
Ro =1660
top =2090
HEXi =2340
HEXo =2820
bottom =4970
Ri2 =5710

#Define Loop
'''Create arrays for each regime then concatenate into one loop array'''
xcore =np.arange(Ri1,Ro+1)
xchimney =np.arange(Ro+1,HEXi)
xhex =np.arange(HEXi,HEXo+1)
xdowncomer =np.arange(HEXo+1,Ri2)
x =np.concatenate((xcore,xchimney,xhex,xdowncomer))
```

Code 20: MSNB System Parameters

```
#./params.py
#Thermal and Neutronic Parameters
alphaT =-3.5e-5 #K-1
lstar =1.63e-4 #sec
beta_eff =6.96e-3 #
lam =0.1 #sec
h =1.09 #m
xi =25.0 #
alphaF =3.3976917e-1 #s/cm
Ax =0.4 #m2
dt =1 #sec
dx =1 #mm
```

Code 21: MSNB Temperature Profile

```
#./TempProfile.py
import loop, functions
import numpy as np

def initial(T_hot,T_cold):
    '''
    Define linear arrays in the heat transfer regimes, then converts those arrays back to
                                                        temperature. Concatenates the regime
                                                       arrays into a total temperature
                                                       profile array.
    '''
    T_xcore =np.linspace(T_cold,T_hot,num=len(loop.xcore))
    T_xchimney =T_hot*np.ones(len(loop.xchimney))
    T_xhex =np.linspace(T_hot,T_cold,num=len(loop.xhex))
    T_xdowncomer =T_cold*np.ones(len(loop.xdowncomer))
    T_x =np.concatenate((T_xcore,T_xchimney,T_xhex,T_xdowncomer))
    return T_x

#_____________________________________________________________
'''
These six functions split the full temperature array into regime arrays
'''
def core(T_x):
    T_xcore =np.split(T_x,[loop.xcore[-1]+1])[0]
    return T_xcore

def chimney(T_x):
    T_xchimney =np.split(T_x,[loop.xchimney[0],loop.xchimney[-1]+1])[1]
    return T_xchimney

def hex(T_x):
    T_xhex =np.split(T_x,[loop.xhex[0],loop.xhex[-1]+1])[1]
    return T_xhex

def downcomer(T_x):
    T_xdowncomer =np.split(T_x,[loop.xdowncomer[0]])[1]
    return T_xdowncomer

def coldleg(T_x):
    T_xcoldleg =np.split(T_x,[loop.HEXi,loop.bottom])[1]
    return T_xcoldleg

def hotleg(T_x):
    T_xhotleg =np.split(T_x,[loop.top])[0]
    return T_xhotleg

#_____________________________________________________________
```

```
45  def advance(T_x,velo,Qcore,Qhex):
46      '''
47      This function is the heart of the simulator. It rounds and converts the velocity to
                                                        an integer mm/s. It generates power
                                                        arrays where the core and heat
                                                        exchanger have linear heat rates, with
                                                         edge effects being considered to
                                                        account for the fact that not some of
                                                        the distance traversed is done in the
                                                        riser or downcomer. The temperature
                                                        profile is then converted to energy.
                                                        It follows a uniform state uniform
                                                        flow assumption where the change in
                                                        energy is equal to the fluid internal
                                                        energy entering minus the fluid
                                                        internal energy entering, plus the
                                                        heat into the control volume. It
                                                        neglects gravemetric, kinetic, and PV
                                                        differentials, as in liquids the
                                                        temperature/heat capacity effects
                                                        dominate. The entering fluid is
                                                        obtained by "rolling" back the energy
                                                        array by the velocity (times the
                                                        timestep length), again considering
                                                        edge effects. The new energy array is
                                                        computed and converted to temperature,
                                                         then returned to simulation.py
48      '''
49      velo=int(round(functions.base_to_milli(velo),0))
50
51      Q_core =Qcore/len(loop.xcore)*np.ones(len(loop.xcore))
52      Q_core[0:velo] =np.linspace(0,Q_core[velo],num=velo)
53      Q_chimney =np.zeros(len(loop.xchimney))
54      Q_hex =-Qhex/len(loop.xhex)*np.ones(len(loop.xhex))
55      Q_hex[0:velo] =np.linspace(0,Q_hex[velo],num=velo)
56      Q_downcomer =np.zeros(len(loop.xdowncomer))
57      Q_x =np.concatenate((Q_core,Q_chimney,Q_hex,Q_downcomer))
58
59      E_x =functions.T2mu(T_x)
60      E_enter =E_x.copy()
61      E_xcore=core(E_enter)
62      E_xcore[-velo:] =E_xcore[-1]
63      E_xhex=hex(E_enter)
64      E_xhex[-velo:] =E_xhex[-1]
65      E_enter =np.roll(E_enter,velo)
66
67      E_exit =E_x.copy()
68      dE_x =E_enter -E_exit +Q_x
69      E_x +=dE_x
70      Tnew =functions.mu2T(E_x)
71      return Tnew
```

Code 22: Initial Condition Set-up

```python
#./initial.py
import params, functions, TempProfile
import numpy as np

#Initialize Operating Range
Tmax =700
Tmin =600
dTmax= Tmax-Tmin


def MassFlow(T_x): #input Celcius
    '''This function is called by Power in this file.'''
    v =functions.Velo(T_x) #calculate fluid velocity
    rho=functions.density(functions.list_ave(TempProfile.hex(T_x))) #calculate average density of regime
    return rho*v*params.Ax,v #output kg/s
#_____________________________________________________________

def Power(Th,Tc): #Input Celcius
    '''This function is called by algorithm in this file.'''
    T_x =TempProfile.initial(Th, Tc)
    m,v =MassFlow(T_x)
    cp_bar =functions.cp(functions.list_ave(TempProfile.hex(T_x))) #Calculate average heat capacity of regime
    dT =Th-Tc #Calculate temperature difference across regime
    power =m*cp_bar*dT
    return power,v #Output kW

def algorithm(Q): #performs a binary search
    '''
    This function is called by the <Calculate Initial Conditions> block of simulation.py. It first checks the power from 700 to 600. Then uses binary search to change the cold temperature until the desired power is obtained. Change variables Tmax and Tmin in this file to adjust the starting cold temp or hot temp
    '''
    dT =dTmax
    T_cold =Tmax-dT
    Qcalc,_ =Power(Tmax,T_cold)
    error =Qcalc-Q

    dTs =[2*dT,dT] #setup for binary search

    i =0
    while round(error,1): #Binary search, adjust the number in the second argument to change the rounding (decimal places of kW)
        change =abs(dTs[-2]-dTs[-1])*0.5 #how much should the cold temperature be adjusted by?
        if error>0: #positive, need less power
            dT -=change #increase cold temperature
        elif error<0: #negative, need more power
            dT +=change #decrease cold temperature

        dTs.append(dT) #store temperature differential
        T_cold =Tmax-dT #calculate new cold temperature
        Qcalc,v =Power(Tmax,T_cold) #calculate new Power
        error =Qcalc-Q #calculate difference between new power and desired power
        i +=1
    print(f'{i} binary search operations')
    return Tmax, T_cold, Qcalc, v
```

Code 23: MSNB Simulator Plots

```
#./plots.py
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
from adjustText import adjust_text
import numpy as np
import glob
from PIL import Image
import os
import TempProfile, loop, controller

def x_vs_Tx(path,t,T_x,Tmin,Tmax):
    ymin=Tmin
    ymax=Tmax

    plt.figure()
    plt.ylim(ymin-1, ymax+1)
    plt.plot(loop.xcore, TempProfile.core(T_x), label='core', color ='red')
    plt.plot(loop.xchimney, TempProfile.chimney(T_x), label='chimney', color ='orange')
    plt.plot(loop.xhex, TempProfile.hex(T_x), label='heat exchanger', color ='blue')
    plt.plot(loop.xdowncomer, TempProfile.downcomer(T_x), label='downcomer', color ='
                                                    green')
    plt.xlabel('Position along Loop, x (mm)')
    plt.ylabel('Temperature at position x, T(x) (Â°C)')
    plt.title('Temperature (Â°C) along MsNB Loop, t='+ str(int(round(t/60,0)))+ 'min')
    plt.legend(loc='best')
    plt.savefig(path)
    plt.clf()
    plt.close()

def t_vs_Q(t,Qhex,Qcore,SP):
    plt.figure(figsize=(8,4.5))
    plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%.2f MW'))
    plt.plot(t/60,Qhex/1e3,label='Heat Exchanger',color='blue')
    plt.plot(t/60,np.array(SP)/1e3, label='Core Set-Point',color='orange',linestyle=':')
    if Qcore !=None:
       Qcore =np.array(Qcore)
       plt.plot(t/60,Qcore/1e3, label='Core',color='orange')
    plt.legend(loc='best')
    plt.xlabel('time, t (min)')
    plt.ylabel('Power duty and load vs. time')
    plt.savefig("img/t_vs_Qt.png")
    plt.clf()
    plt.close()

def t_vs_reac(t,Flow,Temp,Total):
    plt.figure(figsize=(8,4.5))
    plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%d pcm'))
    Flow,Temp,Total =np.array(Flow),np.array(Temp),np.array(Total)
    plt.plot(t/60,Flow*1e5,label='Flow Reactivity')
    plt.plot(t/60,Temp*1e5, label='Temperature Reactivity')
    plt.plot(t/60,Total*1e5, label='Reactivity')
    plt.xlabel('time, t (min)')
    plt.ylabel('Reactivity')
    plt.legend(loc='best')
    plt.savefig("img/t_vs_reac.png")
    plt.clf()
    plt.close()
```

```
58 def t_vs_exp(t,exp):
59    plt.figure(figsize=(8,4.5))
60    plt.plot(t/60,exp)
61    plt.xlabel('time, t (min)')
62    plt.ylabel('dT (sec) per Reactor Period (sec)')
63    plt.savefig("img/t_vs_exponent.png")
64    plt.clf()
65    plt.close()
66
67 def auto_reac_phase(Flow,Temp,Times):
68    plt.figure()
69    plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%.d pcm'))
70    plt.gca().xaxis.set_major_formatter(FormatStrFormatter('%.d pcm'))
71    Flow,Temp =np.array(Flow)*1e5,np.array(Temp)*1e5
72    max =np.max(np.array([np.max(Temp),-np.min(Flow)]))
73    min =np.min(np.array([np.min(Temp),-np.max(Flow)]))
74    pad =3
75    plt.xlim(min-pad,max+pad)
76    plt.ylim(-max-pad,-min+pad)
77
78    plt.plot(Temp,Flow,color='blue')
79    plt.plot([max,min],[-max,-min],linestyle=':',color='black',alpha=0.5)
80    timeslices =np.cumsum(np.trim_zeros(np.array(Times),'b'))
81    plt.scatter(Temp[timeslices],Flow[timeslices],color='darkorange',zorder=5)
82
83    ts,texts =[],[]
84    for t,F,T in zip(timeslices,Flow[timeslices],Temp[timeslices]):
85       text =f' {t//60} min '
86       if text not in texts:
87          texts.append(text)
88          ts.append(plt.text(T,F,text))
89
90    plt.tight_layout()
91    xavoid =np.concatenate((Temp,np.linspace(max,min,num=100)))
92    yavoid =np.concatenate((Flow,np.linspace(-max,-min,num=100)))
93    adjust_text(ts,x=xavoid,y=yavoid,force_text=0.2,force_points=0.2,arrowprops={'
                                                    arrowstyle':'->','color':'darkorange'}
                                                    )
94    plt.xlabel('Temperature Reactivity')
95    plt.ylabel('Flow Reactivity')
96
97    plt.gca().locator_params(axis='both', nbins=6)
98
99    plt.title('Passive Feedback Phase Space')
100   plt.savefig("img/auto_reac_phase.png",bbox_inches='tight')
101   plt.clf()
102   plt.close()
```

```
103
104 def contr_reac_phase(Flow,Temp,Control,Times):
105     plt.figure()
106     plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%.d pcm'))
107     plt.gca().xaxis.set_major_formatter(FormatStrFormatter('%.d pcm'))
108     Flow,Temp,Control =np.array(Flow)*1e5,np.array(Temp)*1e5,np.array(Control)*1e5
109     Passive =Flow+Temp
110     max =np.max(np.array([np.max(Passive),-np.min(Control)]))
111     min =np.min(np.array([np.min(Passive),-np.max(Control)]))
112     pad =3
113     plt.xlim(min-pad,max+pad)
114     plt.ylim(-max-pad,-min+pad)
115
116     plt.plot(Passive,Control,color='blue')
117     plt.plot([max,min],[-max,-min],linestyle=':',color='black',alpha=0.5)
118     timeslices =np.cumsum(np.trim_zeros(np.array(Times),'b'))
119     plt.scatter(Passive[timeslices],Control[timeslices],color='darkorange',zorder=5)
120
121     ts,texts =[],[]
122     for t,P,C in zip(timeslices,Passive[timeslices],Control[timeslices]):
123         text =f' {t//60} min '
124         if text not in texts:
125             texts.append(text)
126             ts.append(plt.text(P,C,text))
127
128
129     plt.tight_layout()
130     xavoid =np.concatenate((Passive,np.linspace(max,min,num=100)))
131     yavoid =np.concatenate((Control,np.linspace(-max,-min,num=100)))
132     adjust_text(ts,x=xavoid,y=yavoid,force_text=0.2,force_points=0.2,arrowprops={'
                                                    arrowstyle':'->','color':'darkorange'}
                                                    )
133     plt.xlabel('Passive Reactivity')
134     plt.ylabel('Control Drum Reactivity')
135
136     plt.gca().locator_params(axis='both', nbins=6)
137
138     plt.title('Controlled Feedback Phase Space')
139     plt.savefig("img/contr_reac_phase.png",bbox_inches='tight')
140     plt.clf()
141     plt.close()
```

```
143 def t_vs_velo(t,v):
144     plt.figure(figsize=(8,4.5))
145     plt.gca().yaxis.set_major_formatter(FormatStrFormatter('%.2f cm/s'))
146     v=np.array(v)
147     plt.plot(t/60,v*100)
148     plt.xlabel('time, t (min)')
149     plt.ylabel('velocity, v (cm/sec)')
150     plt.savefig("img/t_vs_velocity.png")
151     plt.clf()
152     plt.close()
153 
154 def t_vs_angle(t,theta):
155     offset =np.floor(controller.bias)
156     theta -=offset
157     #dtheta = np.diff(theta,append=0)
158     fig,ax =plt.subplots(figsize=(8,4.5))
159     ax.yaxis.set_major_formatter(FormatStrFormatter('%.3fÂ°'))
160     #plt.ylim(np.floor(np.min(theta)),t[-1]/60)
161     #plt.yticks(np.array([0,15,30,45,60,75,90,105,120,135,150,165,180]))
162     plt.text(0,1.01,f'+{offset}Â°',transform=ax.transAxes)
163     plt.plot(t/60,theta)
164     #plt.plot(t/60,dtheta)
165 
166     plt.xlabel('time, t (min)')
167     plt.ylabel("Control Drum Orientation")
168 
169     plt.savefig("img/t_vs_angle.png")
170     plt.clf()
171     plt.close()
172 
173 def gif(frame_folder):
174     frames =[Image.open(image) for image in sorted(glob.glob(f"{frame_folder}/*.PNG"),
                                                    key=os.path.getmtime)]
175     frame_one =frames[0]
176     frame_one.save("img/Tx_t.gif", format="GIF", append_images=frames,
177             save_all=True, duration=500, loop=0)
178 
179 def kill():
180     import os
181     dir= 'img/animateTx_t'
182     for f in os.listdir(dir):
183         os.remove(os.path.join(dir,f))
184     dir= 'img'
185     for f in os.listdir(dir):
186         if os.path.isfile(os.path.join(dir,f)):
187             pass
188             #os.remove(os.path.join(dir,f))
```

Code 24: PID Controller

```
#./controller.py
import functions
import numpy as np
import control as ct

#Prefilter
def prefilter(Qhex,time,tau):
    num =[1]
    den =[tau,1]
    pf =ct.TransferFunction(num,den)
    response_in =Qhex-Qhex[0]
    _,response_out =ct.forced_response(pf,T=time,U=response_in)
    Qcore_SP =Qhex[0] +response_out
    return Qcore_SP

#Proportional Control Loop
Control =True
#Control = False
bias =111.41092285851987
Ku,Tu =3e-4,45
KP =0.45*Ku
tau_int =0.83333*Tu
KI =KP/tau_int
tau_der =0
KD =KP*tau_der

angle_memory =bias

cumu_error =0
error_memory =0
def drum(error):
    inst_error =error
    global cumu_error #preserve cumulative error from one call to next
    cumu_error +=error
    global error_memory #preserve previous error
    roc_error =functions.RoC(error_memory, error)
    error_memory =error

    angle =bias +KP*inst_error \
         +KI*cumu_error \
         +KD*roc_error
    return angle

coeff =[-2.79662670e-09,
        1.78852923e-06,
        -4.36111940e-04,
        4.82874795e-02,
        -2.00900479e+00]

def angle2reac(angle):
    reac =0
    if not Control:
        return reac
    else:
        for c in coeff:
            reac =reac*angle+c
    return reac
```