2016

Modeling Radionuclide Diffusion Using Moose A Multiscale, Multiphysics Platform

Jonathon Gardner
University of South Carolina

MODELING RADIONUCLIDE DIFFUSION USING MOOSE A MULTISCALE, MULTIPHYSICS PLATFORM

by

Jonathon Gardner

Bachelor of Science
Georgia College & State University, 2014

Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Science in

Mechanical Engineering

College of Engineering and Computing

University of South Carolina

2016

Accepted by:

Travis W. Knight, Director of Thesis

Kyle Brinkman, Reader

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

DEDICATION

This thesis is dedicated to my wife Rachel Gardner and my family, for their constant encouragement and support.

ACKNOWLEDGEMENTS

Funding for this project is from the U.S. Department of Energy. This material is based upon work supported by the U.S. Department of Energy Office of Science, Office of Basic Energy Sciences and Office of Biological and Environmental Research under Award Number DE-SC-00012530.

Great contributions to this project were made in the form of the review and instruction of Dr. Travis W. Knight, Dr. Elwyn Roberts, Dr. Brian Powell, Dr. Lindsay Shuller-Nickles, and Dr. Kyle Brinkman. Their instruction has helped focus and direct my research. I also express deep gratitude to Dr. Travis W. Knight for providing me with the opportunity, the means and the support to conduct this research.

I would also like to thank my colleagues at the University of South Carolina; Aaren Rice, Ryan Preist, Donald Shurtliff, Kallie Metzger, Austin Freeman, and David Mason. At times, they have given me good insight and advice to improve and maintain my work.

ABSTRACT

Very durable ceramic waste forms have been proposed which offer long-term stability by binding certain radionuclides in their specific crystalline network. Moreover, multiple such ceramic phases can be tailored to contain specific radionuclides generated in the fuel cycle. Many such candidate ceramic forms are in the early stages of development with limited data. Modeling and simulation including modeling of diffusion can inform and provide direction to ongoing fabrication and experimental efforts. Material properties important in modeling can be obtained through laboratory measurement where available and atomistic simulation. Since diffusion occurs over multiple scales and follows multiple pathways, multiscale modeling is important to capture the detailed behavior from different material phases and microstructures. In light of this, a MOOSE based application (TREX) has been developed to meet these conditions. Different kernel, material, and postprocessor objects have been created in TREX to model anisotropic multiphase, multiscale, multipath diffusion, radionuclide decay , multiphase, multiscale thermal conduction and other physics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DOE . . . . . Department of Energy

FEM . . . . . Finite Element Method

INL . . . . . Idaho National Laboratory

MOOSE . . . . . Multiphysics Object-Oriented Simulation Environment

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The commercial nuclear fuel cycle produces nuclear waste in the form of numerous distinct and mobile radionuclides. To ensure safe disposal of nuclear waste, it is important to stabilize the radionuclides. To this end, certain ceramic materials and phases are proposed to immobilize and stabilize these radionuclides. Modeling these ceramics can be helpful in determining the effectiveness of these waste storage systems. A key modeling detail will be diffusion. Diffusion will occur on many scales and be affected by multiple characteristics.

1.2 OBJECTIVES

With the evolution of computers, computer modeling has become an appealing way to research real world phenomena. When modeling nuclear waste, computer modeling can become irresistible because experiments that could take years can be done in days, hours, or even possibly minutes. The objective of this research is to create a tool to model and confirm the effectiveness of nuclear waste storage systems (specifically the proposed ceramics) and to ultimately broaden computer modeling capabilities. To achieve this goal, TREX was created. TREX uses MOOSE as the base application.

CHAPTER 2

REVIEW OF LITERATURE

2.1 WASTEFORMS

Glassy wasteforms are currently being used to immobilize high-level radioactive waste [13]. The common glasses used are borosilicate and phosphate. Glass was initially used because of its ability to contain radionuclides and the fabrication process is well known. To make the wasteform, glass forming materials (like Silicon) are added to the high-level radioactive waste. In order for this to be done, the high level waste needs to be a liquid. This waste is then put in a canister, usually stainless steel, to cool and solidify. A concern with glass wasteforms though is the uncertainty of long-term stability. Glass are able to contain radionuclides because glass structures are highly disordered. Over time the glass could decompose because to become more stable. It has also been shown that glasses have to be stored in a controlled environment due to leaching [16]. To this end, ceramic wasteforms are currently being investigated because they offer the process and chemical stability of glasses while adding the durability of ceramics. They will also have higher saturation rates because the radionuclides will be bound in the crystalline structure. Synroc (Synthetic rock) has become a popular ceramic of interest for storing high-level waste. It is usually comprised of Zirconolite, Perovskite, and Hollandite. Zirconolite is added to immobilize Plutonium while Perovskite is added to immobilize Plutonium, Strontium, and Barium. Hollandite is used to immobilize Cadmium, Potassium, Rubidium, Barium, and Cesium. Since Cesium is soluble

in water, Hollandite's ability to stabilize and immobilize Cesium is extremely important. Cesium's solubility gives it a higher chance of contaminating the surroundings. The mobility of Cesium in Hollandite will depend on Hollandite's microstructure. It has been shown that Hollandite will have preferred crystallographic directional dependence diffusion.

2.2 MODELING SOFTWARE

Currently in progress is TREX, which is a MOOSE based application for modeling and simulating advanced ceramic wastes. MOOSE (Multiphysics Object-Oriented Simulation Environment) is a finite-element (FEM) multiphysics framework under development by Idaho National Laboratory [8]. It was chosen as the base application for many different reasons. It is designed to be quickly and easily adaptable; it contains a method for coupling multiple applications; and it has a wide user base. Even though MOOSE is highly developed, some *Kernel*, *Material* and *Postprocessor* objects were created and added in order to model the advanced ceramic waste. *Kernel* objects are used to describe the physics. For example, a *Kernel* would be used to describe Diffusion. *Material* objects are used to calculate the material properties, which is where a diffusion coefficient would be defined. The *Material* object can even compute material properties that depend on other variables such as temperature, concentration, etc. *Postprocessor* objects are used to find results. For instance, the flux, concentration gradient, and concentration can be computed using a *Postprocessor* object [8].

Modeling the microstructure of Hollandite is important because as stated earlier, cesium's diffusion rate through Hollandite will depend on Hollandite's microstructure. Similarly, it will be just as important to model the entire wasteform so that the effects of Cesium's release to the surrounding can be seen. The entire wasteform is on the scale of meters, while the microstructure is on the scale of micro- and nano-meters. Modeling details so precise in a simulation is computa-

tionally demanding. To represent both of these realities then, multiscale modeling will be vital. This will be done by having a main engineering scale simulation (to capture the larger wasteform package) coupled with one or more mesoscale simulations (to capture the microstructure). This can be seen in Figure 2.1.

2.3 FINITE-ELEMENT METHOD

As previously stated, MOOSE uses the finite element method (FEM) to solve multiphysics problems. FEM is a numerical method for solving real-world problems involving complicated physics, geometry, and/or boundary conditions [15]. The finite element method has three distinguished characteristics that make it preferred to other numerical methods. The first is it solves real-world problems by dividing domains into subdomains. These subdomains are referred to as finite elements. This is beneficial because it is easier to represent a complicated function by many simple polynomial functions. The second characteristic is governing equations from the problem are used to develop algebraic equations over each finite element. Third, all finite elements are assembled using certain interelement relationships. In other words, the problem is solved with the subdomains in the original position of the domain [15]. Apart from these three distinct characteristics, there are some other features of the finite element method that are beneficial to this area of research. One benefit is the FEM's ability to solve differential equations, like those used when modeling nuclear waste (i.e Diffusion, Decay, etc). While FEM can solve many types of problems, it is exceptionally good at solving partial differential equations. Another feature of FEM is the ability to couple equations, this allows for many physical phenomena to be modeled together.

The finite element method is a very broad method, with many implementations. MOOSE uses the Jacobian-free Newton-Krylov (JFNK) model [8]. While MOOSE can be used without a deep understanding of FEM, having an intimate knowledge of it is crucial in developing a MOOSE based application. To help un-

derstand the JFNK model some introduction models will be discussed that underline many of the JFNK characteristics. The advantage of the JFNK model is the ability to reduce the amount of memory needed therefore decreasing computational cost. In the finite element method, we seek approximation functions over each element Ω_e (where the ^{e} denotes the current element). The approximation functions are in the form of polynomials. The polynomial power of the function used depends on computational power, application of the problem, and the type of mesh used. A mesh is how a domain is divided up into elements. The approximation functions are defined as followed.

$$u_h^e = \sum_{j=1}^{n} u_j^e \psi_j^e(x) \tag{2.1}$$

It must meet some conditions in order that it be convergent to the actual solution, though. First, the functions should be continuous over the element and differentiable, as required by the weak form. Second, it should be a complete polynomial (i.e. include all lower-order terms up to the highesst order used). Lastly, it should be an interpolant of the primary variables at the boundary and if necessary other points [15]. It must be at the boundaries first so the continuity of the solution can be imposed across the interelement boundary. The points (including boundaries) that are interpolants of the variables are refferred to as nodes. The number of nodes depends on the polynomial power and solution type but each element must have at least one node at each boundary point.

To show and derive the finite element model a 1D differential equation with boundary condition will be considered. The eqaution is as follows and in Ω, the domian.

$$-\frac{d}{dx}\left(a\frac{du}{dx}\right) + cu - f = 0 \tag{2.2}$$

where $a = a(x)$, $c = c(x)$, and $f = f(x)$. It can be seen that when c and f equal zero, Equation 2.2 is the steady state diffusion equation (Eq. 2.21), where a would be the diffusion coefficient. It is important to remember this domain will be divided up

into smaller domains (elements in Ω_e) and then an approximation function will be found over the boundary.

The first step in FEM is deriving the weak form. The weak form is desired because it transfers part of the differential from the dependent variable. The weak form is also the format used to add physics to MOOSE. To develop the weak form, all terms need to be moved to one side, multiply by a test function $w(x)$, and integrate over the whole domain Ω. The domain is not divided up until after the equation is in the weak form. Next is to shift half of the derivatives from the dependent variable to the test function. This can be done by the Divergence Theorem (or in 1D by integration-by-parts). Last is to impose the actual boundary conditions. The steps for deriving the weak form will be carried out on Equation 2.2.
Derivation of the weak form Step 1.

$$\int_{x_a}^{x_b} \Big(-w\frac{d}{dx}\Big(a\frac{du}{dx}\Big) + wcu - wf \Big) dx = 0 \tag{2.3}$$

For the second step inegration-by-parts formula is needed:

$$\int_{x_a}^{x_b} w dv = -\int_{x_a}^{x_b} v dw + [wv]_{x_a}^{x_b} \tag{2.4}$$

Derivation of the weak form Step 2:

$$\int_{x_a}^{x_b} \Big(a\frac{dw}{dx}\frac{du}{dx} + wcu - wf \Big) dx - \Big[wa\frac{du}{dx} \Big]_{x_a}^{x_b} = 0 \tag{2.5}$$

Derivation of the weak form Step 3: Since the equation is being derived the boundary condition have not yet been defined. They are therefore defined as followed:

$$Q_1 = \Big(-a\frac{du}{dx} \Big)\Big|_{x_a}, \; Q_2 = \Big(a\frac{du}{dx} \Big)\Big|_{x_b} \tag{2.6}$$

Now that the boundaries are defined the weak form equation is.

$$0 = \int_{x_a}^{x_b} \Big(a\frac{dw}{dx}\frac{du}{dx} + wcu - wf \Big) dx - w(x_a)Q_1 - w(x_b)Q_2 \tag{2.7}$$

Now that the weak form has been derived another important characteristic of FEM is the approximate solution. The approximation functions are defined

by Equation 2.1 and follow the condition describe for an approximation function (which were described earlier). For Equation 2.7 the approximation functions are:

$$u_h^e(x) = c_1^e + c_2^e x \tag{2.8}$$

where c_1 and c_2 are constants. The approximation functions must be linear (or higher) because it has to be differentiable according to the weak form, in this case (Eq. 2.7) at least once. Linear was chosen because it is the simplest form. The second condition is met because the polynomial is complete. The third condition is satisfied if the following is met:

$$u_h^e(x_a) = c_1^e + c_2^e x_a \equiv u_1^e, \quad u_h^e(x_b) = c_1^e + c_2^e x_b \equiv u_2^e \tag{2.9}$$

With two unknowns and two equation c_1 and c_2 can be found so the third condition can be met.

$$\begin{aligned} c_1^e &= \frac{u_1^e x_b - u_2^e x_a}{x_b - x_a} \\ c_2^e &= \frac{u_2^e - u_1^e}{x_b - x_a} \end{aligned} \tag{2.10}$$

This can than be substituded back into Equation 2.8.

$$u_h^e(x) = \frac{u_1^e x_b - u_2^e x_a}{x_b - x_a} + \frac{u_2^e - u_1^e}{x_b - x_a} x \tag{2.11}$$

This equation is simplified.

$$u_h^e(x) = \psi_1^e(x) u_1^e + \psi_2^e(x) u_2^e = \sum_{j=1}^{2} \psi_j^e(x) u_j^e \tag{2.12}$$

where ψ_1^e and ψ_2^e are interpolation functions.

$$\psi_1^e(x) = \frac{x_b - x}{x_b - x_a}, \quad \psi_2^e(x) = \frac{x - x_a}{x_b - x_a} \tag{2.13}$$

In the local cordinate system, where $\bar{x}$ is the local position ($\bar{x} = x - x_a$) and h_e is the element size ($h_e = x_b - x_a$), the interpolation functions are

$$\psi_1^e(\bar{x}) = 1 - \frac{\bar{x}}{h_e}, \quad \psi_2^e(\bar{x}) = \frac{\bar{x}}{h_e} \tag{2.14}$$

The order of the polynomial approximation can be increased to improve the accuracy. If the polynomial was increased to quadratic the approximation functions would be as follows:

$$\begin{aligned} u_1^e \equiv u_h^e(x_1) &= c_1^e + c_2^e x_1 + c_3^e x_1^2 \\ u_2^e \equiv u_h^e(x_2) &= c_1^e + c_2^e x_2 + c_3^e x_2^2 \\ u_3^e \equiv u_h^e(x_3) &= c_1^e + c_2^e x_3 + c_3^e x_3^2 \end{aligned} \tag{2.15}$$

where x_1 and x_3 are the nodes at the boundary and x_2 is a node between x_1 and x_3. x_2 is commonly the midpoint but it does not have to be. To get the interpolation functions the same method as above could be carried out. When interpolation functions are only derived from interpolating the function values and not the derivatives of the function (Eq. 2.8, and Eq. 2.9), they are known as Lagrange interpolation functions [15]. By definition then, Lagrange functions derivatives are not continues between the elements. When the interpolation functions are derived from interpolating the function values and the derivatives of the function they are known as Hermite family of interpolation functions [15]. Hermite functions would be used if the derivatives needed to continue between elements. An example can be seen below.

$$\begin{aligned} w_1^e \equiv w_h^e(x_a) &= c_1^e + c_2^e x_a + c_3^e x_a^2 + c_4 x_a^3 \\ w_2^e \equiv \frac{w_h^e}{dx}\Big|_{x_a} &= c_2^e + 2c_3^e x_2 + 3c_4 x_a^2 \\ w_3^e \equiv w_h^e(x_b) &= c_1^e + c_2^e x_b + c_3^e x_b^2 + c_4 x_b^3 \\ w_4^e \equiv \frac{w_h^e}{dx}\Big|_{x_b} &= c_b^e + 2c_3^e x_b + 3c_4 x_b^2 \end{aligned} \tag{2.16}$$

The minimum power of a Hermite polynomial is cubic because if the function and first derivative are interpolated the minimal number of equations will be four, giving the need for four unknowns. The next step in FEM is to plug the approximation functions into the weak-form equation for each element. This process is very tedious and involves extremely good book keeping. The new equation can then be assembled in a matrix to be solved. This is where FEM has many permutations for solving. Different solving techniques can improve the FEM process but can be

very complicated. The JFNK method uses different algorithms and theorems to make some assumption about the matrix to solve it quicker.

2.4 DIFFUSION

In Section 2.1 diffusion was identified as a key phenomenon for possibly releasing radionuclides from the ceramic waste. Diffusion is the movement of atoms from areas of high concentration (Wasteform) to areas of lower concentration (Canister, Nature, etc.). This is most commonly done through lattice defects. Solid Materials have an organized system for configuring atoms. When an atom is missing or out of configuration, it is referred to as a lattice defect. Three important lattice defects are vacancies, grain boundaries, and interstitials. Vacancies (as one would assume) are empty atom sites. An atom next to a vacancy can move if it attains enough energy (like thermal energy) to squeeze by its neighbors [11]. This can be seen in Figure 2.2. A grain boundary is a boundary at which two grains come together. Diffusion commonly occurs here because the atoms at this boundary are not as closely packed as the grains of the material. This can be seen in Figure 2.3. Interstitials are atoms in spacing between atom sites. This can happen when an area is overpacked; it can be seen in Figure 2.4. Interstitials diffusion is most common in two (or more) element diffusions when one (or more) elements atoms are smaller than the other elements atoms. The smaller atoms allow the element to move more freely in the interstitials than larger atoms. The atoms move through these lattice defects because they are constantly oscillating. The higher the frequency of oscillation, the higher the probability of moving lattice sites. This relation can be modeled by an Arrhenius-type equation.

$$f = f_0 exp\left(\frac{-Q}{k_B T}\right) \tag{2.17}$$

where f is the number of jumps per second, f_0 is a constant that depends on the number of equivalent neighboring sites and on the frequency, Q is the activation

energy and k_b is Boltzmann's constant. These types of diffusion are all driven by concentration gradients. Fick's laws can be used to describe this. Fick's first law is:

$$J = -D\nabla C \tag{2.18}$$

where D is the diffusion coefficient (in m^2/s), which is an important material property. ∇ is the gradient operator ($\frac{\partial}{\partial x}\hat{\mathbf{i}} + \frac{\partial}{\partial y}\hat{\mathbf{j}} + \frac{\partial}{\partial z}\hat{\mathbf{k}}$), and C is the concentration. The diffusion coefficient is similar and could be derived from the jump rate. The jump rate is the 'probability' of a jump occurring per time unit, and the diffusion coefficient is the 'average' distance an atom will travel per time unit. This relation can most easily be understood when the diffusion coefficient equation is known.

$$D = D_0 exp\left(\frac{-Q}{k_B T}\right) \tag{2.19}$$

It can now be seen that $D_0 \approx f_0$ and more than that, the following is known:

$$D_0 = \frac{1}{6} f_0 a^2 \tag{2.20}$$

where a is the jump distance (the distance to the nearest neighbor). This relation helps show that the diffusion coefficient is average distance traveled per time unit. Fick's first law describes diffusion through a cross-sectional area in a given time interval. This is known as flux. Fick's second law describes how concentration moves through a host material over time. This is the nonsteady-state case (Steady state $\frac{\partial C}{\partial t} = 0$)

$$\frac{\partial C}{\partial t} = \nabla(D\nabla C) \tag{2.21}$$

If the diffusion coefficient does not depend on position, then it can be brought out of the gradient. Since the FEM process moves half of the differential to the weighting function, it can handle a position-dependent diffusion coefficient. MOOSE already had a *kernel* for modeling diffusion. The diffusion equation (Eq. 2.21) was broken into two parts to be added to MOOSE. The first *kernel* was just the time derivative and included $\frac{\partial C}{\partial t}$ the *kernels* name is 'TimeDerivative'. The second part

was $-\nabla(D\nabla C)$ the *kernels* name is 'Diffusion'. MOOSE can break up equations because the code is fully couplable. This means when two or more *kernels* are acting on one variable, the equation is the sum of the kernels. When the 'TimeDerivative' and 'Diffusion' are used on the same variable, the equation is equivalent to $\frac{\partial C}{\partial t} + -\nabla(D\nabla C) = 0$, which is Equation 2.21. Dividing the equation into two kernels is also benificial because it allow 'Diffusion' to be used alone for the steady state case and 'TimeDerivative' to be used for other equations that use $\frac{\partial v}{\partial t}$ where v is the variable the *kernel* is acting on.

An analytical solution to Fick's second law can be found, if the diffusion is one dimensional, the diffusion coefficient is constant with position, and the solvent is "infinitely long" (i.e. the solute does not reach the boundary of the solvent $10\sqrt{Dt}$). The analytical solution is:

$$\frac{C_i - C_x}{C_i - C_0} = erf\left(\frac{x}{2\sqrt{Dt}}\right) \tag{2.22}$$

where C_i is the solute concentration at the interface, C_x is the solute concentration at distance x and time t and C_0 is the initial solute concentration.

2.5 HEAT TRANSFER

Heat generation from radionuclide decay was also identified as an important factor in modeling this high-level waste. The heat generation is of interest because it can cause high temperatures which can increase release rate. In section 2.4, it was explained that atoms are oscillating and that with an increase in energy, they have a higher chance of changing lattice sites and therefore will diffuse. This relation was first seen in the jump rate equation Equation 2.17 and ultimately seen carried out in the diffusion coefficient equation (Eq. 2.19). This, along with other temperature dependent processes, make the heat generation from radionuclide decay an area of interest. The equation used to model this is the heat diffusion equation, which

is as follows:

$$\nabla k \nabla T + q = \rho c_p \frac{\partial T}{\partial t} \tag{2.23}$$

where T is temperature, k is thermal diffusivity, q is the volumetric heat generation, ρ is density and c_p is the heat capacity [3]. MOOSE again already had *kernels* for the heat equation. Again the equation was broken up. The first part was $-\nabla k \nabla T$ the *kernels* name is 'HeatConduction'. The second part is the heat generation term $-q$ the *kernels* name is 'HeatSource'. The last part was the time derivative $\rho c_p \frac{\partial T}{\partial t}$ the *kernels* name is *HeatConductionTimeDerivative*.

2.6 RADIONUCLIDE DECAY

Radionuclide decay will be another significant phenomenon. It will help validate the wasteform by showing whether or not it contains the radionuclides until it has decayed into something stable. It can be modeled by the following differential equation:

$$-\lambda N = \frac{dN}{dt} \tag{2.24}$$

where N is the concentration of radionuclides and λ is a decay constant, which is a property of the radionuclide. A more common constant used is the half-life which is related to the decay constant as followed:

$$t_{1/2} = \frac{ln2}{\lambda} \tag{2.25}$$

The solution to Equation 2.24 can be solved with some simple calculus. The analytical solution is:

$$N(t) = N_0 e^{-\lambda t} \tag{2.26}$$

MOOSE only had part of the decay equation, so a *kernel* had to be added to TREX. As usual, the equation was broken up. The time derivative $\frac{dN}{dt}$ was already in MOOSE. This kernel was already described in section 2.4. However, the other part of the equation, $-\lambda N$, had to be added to TREX. To add the equation it must be

in the weak form. The first thing in deriving the weak form is to move everything to one side so the equation is equal to zero. Since $\frac{dN}{dt}$ is already implemented as positive $-\lambda N$ will be moved to the right side making it positive λN. Next, the equation can be split into two (since $\frac{dN}{dt}$ is already implemented), multiply by a test function and integrate over the domain. The equation then becomes:

$$0 = \int_{\Omega} w\lambda N d\Omega \tag{2.27}$$

The equation is now in the form to add to TREX. The name of the *kernel* is 'RadioactiveDecay'. It is also necessary to look at the rate of generation of the daughter isotope. This can be useful if a phenomenon (like diffusion, or decay chain) depends on the concentration of the daughter isotope. For example in hollandite cesium does not like to be in the same tunnel as barium (cesium daughter isotope). This could cause cesium to diffuse quicker when in the presence of barium. If the radionuclide decay rate follows Equation 2.24, then the daughter isotope generates at the following rate:

$$\frac{m_{ad}}{m_a}\lambda N = \frac{dN_d}{dt} \tag{2.28}$$

where N is the concentration of radionuclide, λ is the decay constant of the radionuclide, m_a is the Atomic mass of the radionuclide, m_{ad} is the Atomic mass of the daughter isotope and N_d is the concentration of the daughter isotope. When the radionuclide decays to multiple daughters isotope the equation becomes:

$$p\frac{m_{ad}}{m_a}\lambda N = \frac{dN_d}{dt} \tag{2.29}$$

where p is the percent of the radionuclide that decays to the daughter isotope. Adding radionuclide generation to TREX must take similar steps as adding radionuclide decay. Again move everything to the side that makes $\frac{dN_d}{dt}$ positive (since it is already implemented in MOOSE) and split the equation into two equations. Next multiply by a test function and integrate over the domain.

$$0 = \int_{\Omega} -p\frac{m_{ad}}{m_a}\lambda N d\Omega \tag{2.30}$$

The equation is now in the format accepted by TREX. The name of the *kernel* is 'RadioactiveGeneration'.

2.7 EFFECTIVE PROPERTIES

A key desire when designing TREX was the ability to couple microscale simulations to an engineering scale. MOOSE offers the ability to run multiple simulations and share data across them. This makes MOOSE a good base for multiscale modeling because the only thing left to design is a method for applying these properties found on the micro scale to the engineering scale. In other words, the microscale simulation will be heterogeneous. It will have multiple grain boundaries, grains, phases, and other preferred paths, while the engineering scale will be homogenous. The engineering scale will be homogenous because modeling the heterogeneity of the entire waste form would require the mesh to be extremely fine and therefore be computationally demanding. This is similar to an extremely detailed picture that requires many pixels so the picture will not be blurry. It is impossible to directly apply a heterogeneous property as a homogenous property but an 'average' value of a region of the heterogeneous property could be applied as a homogenous property. It would only be a region of the heterogeneous property because again it would be too computationally demanding to model the entire waste form. To use a region, it is important that the region captures the entire heterogeneity of the property. This means that if the waste form has multiple grains and phases the 'average' of a single phase and single grain would be invalid because it does not capture the entire heterogeneity. Also, since the material property is multi-dimensional, it is not as simple as finding the 'average' of the heterogeneous property. It will require finding an effective property. Effective properties depend on the equation they are applied to. There are many methods for finding effective properties. Three methods are used and will be discussed.

The popular method used in TREX is the Asymptotic Expansion Homoge-

nization method (AEH) [9] [2]. AEH was chosen because it is a diverse and well-developed method. These other methods were mostly used for validating AEH. The effective diffusion coefficient using the AEH method is:

$$D_{ij}^{eff} = \frac{1}{|\Omega|}\int_{\Omega} D_{ik}^{*}\left(I_{kj} + \frac{\partial \chi_j^{*}}{\partial y_k} d\Omega\right) \tag{2.31}$$

D_{ij}^{eff} is the effective diffusion coefficient matrix, D_{kj} is the heterogeneous diffusion coefficient of the microstructure, Ω is the domain, I is the identity matrix and the vector field χ is the solution of following local boundary-value problem.

$$-\frac{\partial}{\partial y_i}\left[D_{ik}^{*}\left(I_{kj} + \frac{\partial \chi_j^{*}}{\partial y_k}\right)\right] = 0 \tag{2.32}$$

$$-N_i\left[D_{ik}^{*}\left(I_{kj} + \frac{\partial \chi_j^{*}}{\partial y_k}\right)\right] = 0 \; on \; \Gamma \tag{2.33}$$

where Γ is the boundary of Ω. The effective thermal diffusivity using the AEH method is similar:

$$k_{ij}^{eff} = \frac{1}{|\Omega|}\int_{\Omega} k_{ik}^{*}\left(I_{kj} + \frac{\partial \psi_j^{*}}{\partial y_k} d\Omega\right) \tag{2.34}$$

k_{ij}^{eff} is the effective thermal diffusivity matrix, k_{kj} is the heterogeneous thermal diffusivity of the microstructure and the vector field ψ is the solution of following local boundary-value problem.

$$-\frac{\partial}{\partial y_i}\left[k_{ij}^{*}\left(I_{jk} + \frac{\partial \psi_j^{*}}{\partial y_k}\right)\right] = 0 \tag{2.35}$$

$$-N_i\left[k_{ij}^{*}\left(I_{jk} + \frac{\partial \psi_j^{*}}{\partial y_k}\right)\right] = 0 \; on \; \Gamma \tag{2.36}$$

These effective equations are similar because their governing equations (Eq. 2.21, Eq. 2.23) are similar. When deriving these two equations, using AEH, the assumption is made that the homogeneous (x) to heterogeneous (y) scale is very small ($1 << \epsilon = x/y$). This will be valid on our cases because the wasteform is on the scale of meters (x=1) and the micro scale is on the scale of micro-, nano-meters (y=1e6 or 1e9) making the scale $\epsilon = 1e-6$ or $1e-9$. The effects of the scale not being small will be shown in section 3.2. To use the AEH method in TREX for effective diffusion coefficients it must first be in the weak form. To do this, multiply

Equation 2.32 by a test function and integrate over the domain (since the equation is already equal to zero).

$$0 = -\int_{\Omega} \left(w \frac{\partial}{\partial y_i} \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Omega \tag{2.37}$$

Next using integration by parts Equation 2.37 can be rearranged.

$$0 = -\int_{\Omega} \left(\frac{\partial}{\partial y_i} w \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] - \frac{\partial w}{\partial y_i} \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Omega \tag{2.38}$$

Then using the divergence theorem the equaiton becomes:

$$0 = \int_{\Omega} \left(\frac{\partial w}{\partial y_i} \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Omega - \int_{\Gamma} \left(N_i w \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Gamma \tag{2.39}$$

The equation now has two parts.

$$\int_{\Omega} \left(\frac{\partial w}{\partial y_i} \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Omega \tag{2.40}$$

$$- \int_{\Gamma} \left(N_i w \left[D^*_{ij} \left(I_{jk} + \frac{\partial \chi^*_j}{\partial y_k} \right) \right] \right) d\Gamma \tag{2.41}$$

Equation 2.40 was added as a *kernel* named 'HomogenizationDiffusionCoefficient'. Equation 2.41 was already implemented as a *Boundary Condition* named 'periodicBC'. A *postprocessor* named 'HomogenizedDiffusionCoefficient' was added based on Equation 2.31. The AEH method for effective thermal diffusivity was already implemented in MOOSE. The *kernel* based on Equation 2.35 name is 'HomogenizationHeatConduction'. The *postprocessor* based on Equation 2.34 name is 'HomogenizedThermalConductivity'.

The second method is quicker and computationally easier but it is for limiting cases. It is limited to 1D isotropic cases. The effective diffusion coefficient can be derived by finding the average of Fick's first law (Eq. 2.18) using integration.

$$\frac{\int_a^b J dx}{b-a} = \frac{\int_a^b -D^{eff} \frac{dC}{dx} dx}{b-a} \tag{2.42}$$

Since the definition of D^{eff} is that it is homogenous (does not depend on x) it can move outside of the integral and be solved.

$$D^{eff} = \int_a^b J dx \Big/ \int_a^b \frac{dC}{dx} dx \tag{2.43}$$

The third method is a common method used in electronics for resistors in parallel and series [21]. This method is restricted to materials with parallel preferred paths that are parallel or perpendicular to the flux. To derive these equations for a material with two paths (path A and path B), the following variables are defined: Flux J_T, J_A and J_B, diffusion coefficient D_e, D_A and D_B, concentration gradient $\frac{dC_T}{dx}$, $\frac{dC_A}{dx}$ and $\frac{dC_B}{dx}$, and volume V_T, V_A and V_B for the total block, path A and path B, respectively. Also, the volume fraction for path A and B is f_A and f_B where volume fraction is the volume of the path over the total volume. When the flux is parallel to the paths, the following is known:

$$\begin{aligned} \frac{dC_T}{dx} &= \frac{dC_A}{dx} = \frac{dC_B}{dx} \\ V_T J_T &= V_A J_A + V_B J_B \end{aligned} \tag{2.44}$$

Using Ficks First Law (Eq. 2.18), the following is derived:

$$D_{PRL} = D_e = f_A D_A + (1 - f_A) D_B \tag{2.45}$$

To derive this, equation $\frac{1}{\frac{dC_T}{dx}}$ was multiplied throughout, making the equation invalid when the concentration gradient is equal to zero. When the flux is perpendicular to the paths, the following is known:

$$\begin{aligned} V_T \frac{dC_T}{dx} &= V_A \frac{dC_A}{dx} + V_B \frac{dC_B}{dx} \\ J_T &= J_A = J_B \end{aligned} \tag{2.46}$$

Equation 2.47 is derived using Ficks First Law (Eq. 2.18):

$$D_{PRP} = D_e = \frac{D_A D_B}{f_A D_B + (1 - f_A) D_A} \tag{2.47}$$

To derive this equation, $\frac{1}{J_T}$ was multiplied throughout, making the equation invalid when the flux is equal to zero.

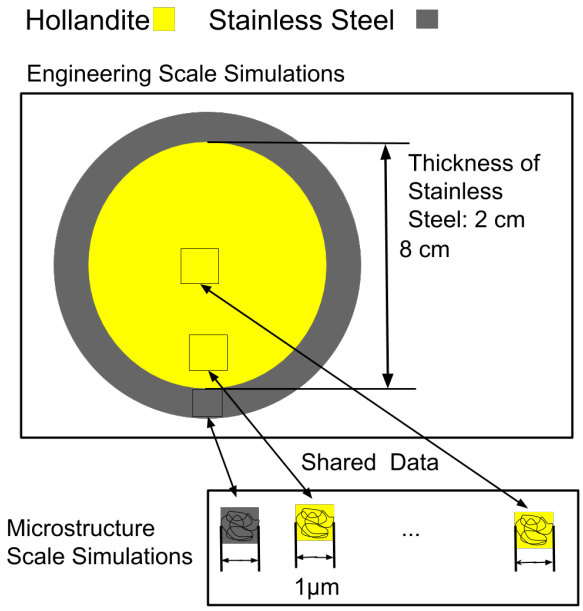

Figure 2.1 Coupled Simulation

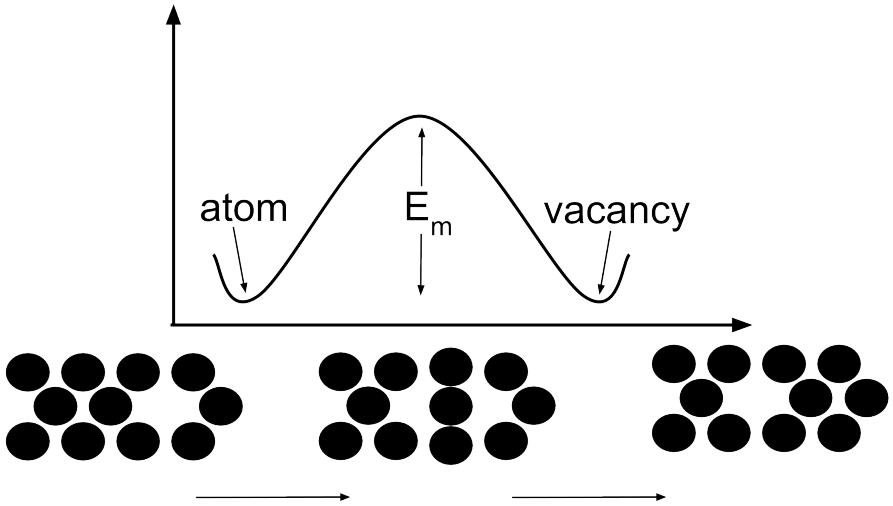

Figure 2.2 Vacancy Diffusion

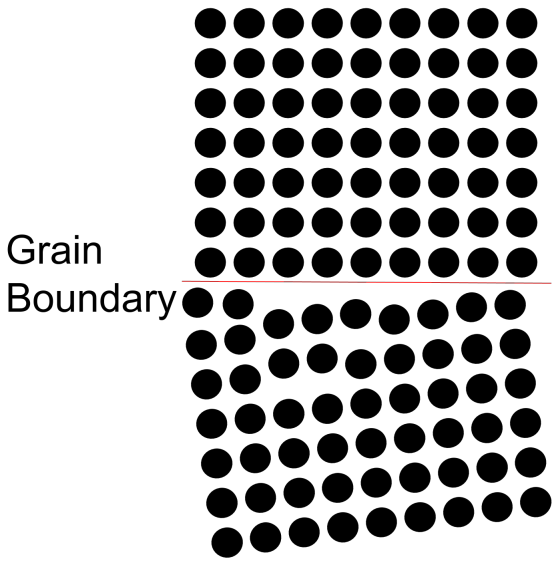

Figure 2.3 Grain Boundary Diffusion

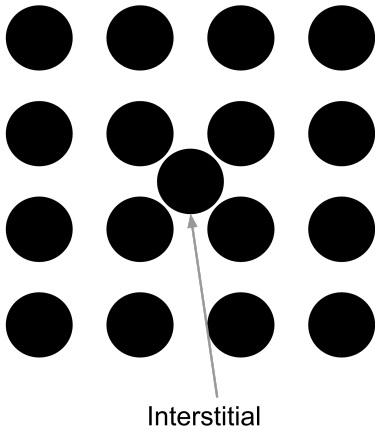

Figure 2.4 Interstitials Diffusion

CHAPTER 3

METHODOLOGY

3.1 ADDED OBJECTS

To model these ceramics many objects (*Kernels*, *Material*, etc.) were developed and created for TREX. These objects were used to model the ceramics, validate the code, and/or obtain resualts. Some of these objects, like the 'HomogenizationDiffusionCoefficient', have been mentioned. The following section will describe all of the objects created for TREX and provide some sample inputs. An object, if desired and benificial outside of modeling ceramics, could be pushed to MOOSE to help improve MOOSE's ability to model. Many of these objects have that potential. All objects are indirect, if not direct, sub-objects of the respected main object. This is important to note because certain objects will inherit different inputs. For example, to create a new kernel, the new kernel does not have to create an input for the variable it is acting on. An input for the variable it acts on will be inherited from the main object.

Kernels

The *Kernel* objects are the heart of TREX. This is were the actual physics is described. Table 3.1 shows all of the *kernel* objects added to TREX.

'CylinderHeatSource' is a sub-*kernel* of 'HeatSource' which is based on the heat diffusion equation (Eq. 2.23). As the name would reveal, 'CylindricalHeat-

Source‘ allows the user to restrict the volumetric heat generation term to a cylindrical shape. Since ’CylindricalHeatSource‘ is a sub-*kernel* of ’HeatSource‘ it accepts the same input plus whatever is added to restrict the heat generation to a cylinder. Since the parent *kernel* (’HeatSource‘) already has the necessary inputs for volumetric heat generation, the following inputs were the only ones needed to be added, radius, height, normal, normal_axis and origin. Radius is the radius of the cylinder. Height is the height of the cylinder. Normal and Normal_axis is the axis parallel to the height. Normal takes an input in vector, while normal_axis takes an input of ‘x’, ‘y’, or ‘z’. Only one of the inputs is required, not both. Lastly, origin is the midpoint of the cylinder. Since origin is the midpoint it is important to note the height only extends half way up and down from the origin. For example, the wasteform modeled later will be a cylinder with a radius in the xy plan of 0.5612 meters with a height extending from 0 to 2.95 meters in the z-axis (Figs. 3.6 - 3.8). An example code can be seen in Input 1. After the initial values are set the heart of the code computes the solution. When solving the equation the current point is tested to see if it is in the region of the defined cylinder. If it is in the cylinder the solution of HeatSource is called (the physics for heat generation). If not the solution is set to zero (no heat generation).

’HomogenizationDiffusionCoefficient‘ and ’HomogenizationAnisoDiffusionCoefficient‘ model Equation 2.40 discussed in section 2.7. The only difference with the two is, ’HomogenizationDiffusionCoefficient‘ uses an isotropic diffusion coefficient and ’HomogenizationAnisoDiffusionCoefficient‘ uses an anisotropic diffusion coefficient. The two *kernels* were created because MOOSE uses a single value for isotropic values to increase computational time. The only necessary inputs to add (since it is a sub-kernel of ’kernel‘) are those that model the specific equation, which are D_name and component. D_name is the name of the material property of the diffusion coefficient (D^*_{ik}). Component is the direction of the variable this kernel acts on (component $= k$ in Eq. 2.40).

'RadioactiveDecay' was added to model Equation 2.27. The created *kernel* takes one input half_life. Half_life is the half life of the radionuclide. The object finds the decay constant using Equation 2.25. An example input block for cesium 137 (half life of 30.1 years) can be seen in Input 2.

The last kernel added was 'RadioactiveGeneration'. This was added to model Equation 2.30. The created *kernel* takes four inputs coupled, half_life, mass_ratio, and decay_chain_ratio. Coupled is the name of the variable decaying. Half_life is the half-life of the decaying variable. Mass_ratio is the mass of the daughter isotope over the mass of the decaying isotope. Decay_chain_ratio is the percent of the radionuclide that decays to the daughter isotope. An example of the input blocks can be seen in Input 3.

Aux Kernels

Aux Kernel objects are used to find secondary values of actual physics. For example, if flux was needed in another equation a *Kernel* would be used to describe diffusion and store the concentration and a *Aux Kernel* would be used to find and store the flux. While the same calculation could be done in a postprocessor if it will be used in another equation in MOOSE common practice is to use an *Aux Kernel*. Table 3.2 shows all of the *Aux Kernel* objects added to TREX.

The 'FluxAux' and 'FluxAnisoAux' are used to create an aux variable of the diffusion flux defined by Fick's first law (Eq. 2.18). The only difference with the two 'FluxAux' used an isotropic diffusion coefficient and 'FluxAnisoAux' uses an anisotropic diffusion coefficient. Since this is an *aux kernel* it uses a variable to find the aux variable and it does not have to be in the weak form. These *kernels* can take four inputs, coupled, D_name, cross_section_axis, and normal. Coupled is the concentration variable. D_name, like before, is the name of the diffusion coefficient. Cross_section_axis and normal are axis normal to the flux. Cross_section_axis takes an input of 'x', 'y', or 'z' while normal takes the input in vector form. Again, only

of the inputs is required, not both.

Initial Conditions

Table 3.3 shows all of the *Initial Condition* objects added to TREX. *BoundingBlockIC* is used to give a block an initial value. A block is a region defined when creating the mesh. The IC takes two input, inside and outside. Inside is the initial value of the block. Outside is the rest of the mesh.

BoundingBoxIC is used to give a mesh blocks different initial values. A mesh block can be defined when creating the mesh. The IC takes two inputs block and inside. Block is the block the initial conditions is applied to and inside is the value applied to the block.

BoundingCircleIC is used to give a defined circle (or cylinder) an initial value. The IC uses radius, height, origin, inside, and outside as inputs. Radius is the radius of the circle (or cylinder). If applicable height is the height of the cylinder. Origin is the starting point of the circle (if applicable the height start at the initial point and goes up and down half the height). Inside is the value inside the cylinder and outside is the value outside.

Materials

Material objects are used to define material properties. For example, the diffusion coefficient would be defined in the *Material* objects. The *Material* object can be broad, like describing the Arrhenius type equation use for diffusion coefficients, or it can be specific describing the diffusion coefficient of Cesium in Graphite. Table 3.4 shows all of the *Material* objects added to TREX.

'AnisoArrheniusDiffusionCoefFromPostprocessor' object is used to create a diffusion coefficient material property from values in a postprocessor. This was created to be used with multiscale simulations. When simulating multiscale simulations using multiapp, data can be shared through postprocessors. This *material*

object is used for creating a position dependent diffusion coefficient from the effective diffusion coefficient found at different sub regions. Currently this object uses a linear fit between known diffusion coefficients. The inputs for this object are diffusion_coef, D_xx, D_xy,...,D_zz, radius, x, y, z, and position_from_center. Diffusion_coef is the name of the material property, it is used for other objects to reference the value. D_xx,D_xy,..., and D_zz are the names of the postprocessors the contains the sub regions diffusion coefficient. There must be a postprocessor name for each sub region and at least two regions. Radius, x, y, and z are the dimensions of the domain where the diffusion coefficient will be applied to. If only x, y, and z are supplied then the domain is a rectangle. If the radius and either x, y, or z are supplied then the domain is a cylinder with the height parallel to the axis supplied. If only the radius is supplied then it is a sphere. It is important to note that the x, y, and z are the distances from the center. If the mesh is off center use position_from_center. Lastly, position_from_center is the distance of the center of the geometry from the center of the mesh. After the initial conditions are set, If else statements are used to find which region the current position is between. Then the following equation is used.

$$D_{ik} = \frac{p_l - p_c}{p_l - p_f} D_{ik}^f + \frac{p_c - p_f}{p_l - p_f} D_{ik}^l \tag{3.1}$$

where D_{ik} is the diffusion coefficient, p_l is the position of the region furthest from the center, p_c is the current position (this value should be between p_l and p_f), p_f is the position of the region closest to the center, D_{ik}^f is the value of the diffusion coefficient closest to the center, and D_{ik}^l is the value of the diffusion coefficient furthest from the center. An example input block of a cylinder region (similar to the one modeled in the final cases) can be seen in Input 4. An example input block of a rectangular region can be seen in Input 5.

The 'ArrheniusDiffusionCoef' and 'ArrheniusAnisoDiffusionCoef' model the diffusion Arrhenius equation (Eq. 2.19). The only difference with the two are 'ArrheniusDiffusionCoef' uses an isotropic diffusion coefficient and 'ArrheniusAniso-

DiffusionCoef' uses an anisotropic diffusion coefficient. The inputs are diffusion_coef, D0, Q, R, and T. Diffusion_coef is the name of the material property, it is used for other objects to reference the value. Q is the activation energy. R is the gas constant. T is the temperature of the material, this can be a variable or a constant value.

'BoxDiffusionCoef' is used to restrict a diffusion coefficient to a box. The inputs are diffusion_coef, inside, outside, x1, y1, x2, and y2. Diffusion_coef is the same as previous *material* objects, it is name of the material property referenced by other objects. Inside is the value of the diffusion coefficient inside the box and outside is the value of the diffusion coefficient outside the box. X1 is the left position, y1 is the bottom position, x2 is the right position and y2 is the top position of the box.

'CalcDiffusionCoef' is used to create a diffusion coefficient from the flux and concentration gradient. This material object was described in section 2.7. The inputs are diffusion_coef, flux, and grad. The diffusion_coef is the same as described previously. Flux is the name of the postprocessor that computes the average flux. Grad is the name of the postprocessor that computes the average concentration gradient.

'GenericTensorConstantMaterial' creates constant generic material properties. This could be anything from thermal conductivity to diffusion coefficients to youngs modules, any constant anisotropic material property. The inputs are prop_name and prop_value. Prop_name is the name of the material property referenced by other objects. Prop_value is the tensor value of the constant material property.

'GrainMaterialDiffusion' and 'GrainAnisoMaterialDiffusion' create an isotropic or anisotropic diffusion coefficient from a 'map' of the microstructure. The inputs are diffusion_coef, D_B_value, D_GB_value, shape_var, unique_grains, and euler_angle_provider. As previously described the diffusion_coef is the name of

the diffusion coefficient referenced by other objects. D_B_value is the value of the diffusion coefficient in the grains, this will contain a value for each phase, if applicable. D_GB_value is the value of the diffusion coefficient in the grain boundary. Shape_var is a binary map of the microstructure. A value of zero is the grain boundary and a value of one is the grain. Unique_grains is a 'map' of each grain. Euler_angle_provider contains each grains angle and phase. Figure 3.1 shows an example of a map of a microstructure (left image), the unique grains (center image) and the final diffusion coefficient value (right image).

'GrainMaterialThermal' is similar to 'GrainMaterialDiffusion' and 'GrainAnisoMaterialDiffusion'. 'GrainMaterialThermal' creates the specific heat and thermal conductivity from a 'map' of the microstructure. The inputs are specific_heat_name, thermal_conductivity_name, density_name, unique_grains, and euler_angle_provider. specific_heat_name, thermal_conductivity_name, and density_name are the names of the specific heat, thermal conductivity and density referenced by other objects. Unique_grains is a 'map' of each grain. Euler_angle_provider contains each grains phase.

'LineGrains', 'SquareGrains', 'HexagonGrains', and 'CircleGrains' create microstructure 'map'. The inputs are shape_name, volume_fraction, sections, and dimensions. Shape_name is the name of the map that will be referenced by other objects, volume_fraction is the ratio of grain vs. grain boundaries, and sections is the number of grains. The four different shapes can be seen in Figure 3.2. A value of zero (blue) is the grain boundary and one (red) is a grain.

'WasteContainer' and 'WasteAnisoContainer' create the necessary material properties to model a cylinder waste container. The inputs are prop_names, inside, outside, T, radius, and height. Prop_names are the name of the material properties referenced by other objects, more than one can be defined. This could be diffusion coefficients, thermal conductivity, etc. Inside and outside are the values of the material properties. T is the temperature, radius is the radius of the cylinder and

height is the height of the cylinder.

'DiffusionProperties' are predefined diffusion coefficient of many different materials used. These material objects were created for materials that are commonly used. Rather than the user inputting the material properties every simulation certain material have predefined material properties. All of the diffusion coefficients follow the Arrhenius-type equation (Eq. 2.19). The following are implemented, cesium in Hollandite [20], barium in Hollandite [20], cesium in Borosilicate glass [19], cesium in graphite [10], iodine in graphite [5], cesium in silicon carbide [17], iodine in silicon carbide [7], cesium in Stainless Steel [12], barium in Stainless Steel, and iodine in Stainless Steel. Input 6 is an example code block.

'ThermalProperties' are predefined thermal conductivity, specific heat and density of many different materials used. The thermal conductivity and specific heat can be temperature dependent. The following were implemented: Hollandite [4], Borosilicate glass [14], graphite [3], silicon carbide[3], stainless steel [3] and uranium dioxide [6]. Input 7 is an example code block.

Postprocessors

Table 3.5 shows all of the *Postprocessor* objects added to TREX. *Postprocessors* are used to extract the date from the simulation. This can be as simple as returning a variable's value at a given point or it can involve a complex equation over the whole domain.

The first *postprocessor* is different than most, in that it is used to find a good future time step when modeling diffusion. 'AverageTimeStepForDiffusion' was designed to be used with the *timestepper* object 'PostprocessorDT'. 'PostprocessorDT' uses the value of the postprocessor as the time step. 'AverageTimeStepForDiffusion' can be used when modeling diffusion. When modeling diffusion it is important to take the right size time steps that capture change in concentration. If the time steps are too large, though, then the simulation could miss certain things, but

if the time steps are too small then computer resources can be wasted, like time and memory because the simulation will take longer and the output will be larger. While an initial time step could meet this criterion over time the concentration becomes saturated making larger time steps more beneficial because the diffusion is "slower" (a change in concentration takes longer) because the concentration gradient is smaller. The postprocessor takes the desired concentration change and calculates a time step that would achieve this. The *postprocessor* was designed for a 1D homogeneous case. It can be used in most diffusion cases, though, and yield the same results. With this in mind the inputs for the *postprocessor* are increase_by, interface_value, average_concentration, volume, and decreasing. Increase_by is the desired percent of change in the concentration per time step. Interface_value is the value dividing the solute and the solution (or equivalent value for other cases). Average_concentration is the name of the *postprocessor* that computes the average concentration of the mesh. Volume is the volume (area or length) of the mesh. Decreasing is a boolean value, true if the concentration is decreasing. To find the future time step an equation was derived using the following. Since the error function is for an 'infinitely long' solute and the mesh is finite, to find the average the error function can be integrated over infinity. This is a better measurement because it captures conservation of mass.

$$C_{ch} = \frac{C_{avg,t_1} - C_{avg,t_2}}{C_{avg,t_1}} \tag{3.2}$$

where C_{ch} is the concentration change in percent, C_{avg,t_1} is the average conentration at t_1, the current time and C_{avg,t_2} is the average concentration at some time t_2. Both C_{ch} and C_{avg,t_1} are known values. C_{avg,t_2} can then be calculated, if t_2 can be calculated than the time step size can be found knowing $dt = t_2 - t_1$. The analytical solution to Fick's second law involving the error function (Eq. 2.22) can be used to find t_2. Equation 2.22 finds the concentration at a given point though. To find the average concentration, the equation can be integrated and divided by the volume

(area or length).

$$C_{avg,t} = \int_0^\infty C_i erfc\left(\frac{x}{2 \cdot \sqrt{D \cdot t}}\right) d\Omega \Big/ V \tag{3.3}$$

Equation 3.2 and 3.3 can be combined and rewriten as:

$$C_{avg,t_1} - C_{ch} \cdot C_{avg,t_1} = \int_0^\infty C_i erfc\left(\frac{x}{2 \cdot \sqrt{D \cdot t_2}}\right) d\Omega \Big/ V \tag{3.4}$$

Everything in Equation 3.4 are known except D, and t_2. D can be solved using Equation 3.3 and C_{avg,t_1}.

$$C_{avg,t_1} = \int_0^\infty C_i erfc\left(\frac{x}{2 \cdot \sqrt{D \cdot t_1}}\right) d\Omega \Big/ V \tag{3.5}$$

The interpolation search is a common computer science methods for searching an assorted array. It can be applied, though, to solve for D in Equation 3.5 and t_2 in Equation 3.4. The interpolation search finds a value n (where n is known to be in the array) in a sorted array through a recursive search. The search starts with a min and max position, the min being the start of the array, 0 and the max being the end, l. First, the search checks the midpoint of these two points $m = \frac{l-0}{2}$. If the value at the midpoint is greater than n, then since the array is sorted the search knows n must be somewhere before m. Therefore the search keeps the min point the same and sets the max point to m. If the value at the midpoint is less than n, then since the array is sorted the search knows n must be somewhere after m. Therefore the search sets m to the min point and keeps the max point the same. It continues this process until n is found. This can be applied to find D in Equation 3.5 by having a D_{max} and a D_{min}. In most cases, the diffusion coefficient is below one so D_{max} is one and D_{min} is zero because a diffusion coefficient cannot be negative. The $D_{mid}(= \frac{D_{max}-D_{min}}{2})$ is then tested. If C_{avg,t_1} is to low then D_{mid} is set to D_{min}. If C_{avg,t_1} is to high then D_{mid} is set to D_{max}. This process is then repeated till C_{avg,t_1} is within a tolerance. Once the diffusion coefficient is found then the same method can be used to solve for t_2 in Equation 3.4. The Gaussian integration method is used to integrate both equations.

'HomogenizedDiffusionCoefficient' and 'HomogenizedAnisoDiffusionCoefficient' model Equation 2.31. The inputs for these postprocessors are D_name, conc_x, conc_y, conc_z, row, and col. D_name is the name of the material property of the diffusion coefficient referenced by other objects (D^*_{ik}). Conc_x, conc_y, and conc_z is the solution to Equation 2.32 when $k = 0, 1,$ and 2, respectively. Row and col are the row and column of the diffusion coefficient to return, i and j, respectively.

Actions

Action objects are similar to scripts, they can be used to repeat processes. Table 3.6 shows all of the *Action* objects added to TREX.

PointValueOnLineAction is used to find a value of a variable at a given number of points on a line. The object is a *Action* because rather it creates a new *postprocessor* for each point needed. The inputs are variable, point1, point2, num_out_per_unit, and base_name. Variable is the name of the variable this object will act on. Point1 and point2 are the starting and ending points of the line. Num_out_per_unit is the number of points on the line. Base_name can be specified to give the *postprocessors* a base name. This could be used if the action is used more than once. The name of *postprocessors* are x position, y position, z position appended to the base_name, if one is given. An example input can be seen in Input 8. This produces 100 points on the line from '0 0.5 0' to '10 0.5 0'.

Executioners

Executioner objects are where the properties for solving the problem are defined. This is where the problem can be defined as steady state or time dependent. If necessary start time, end time, and/or run time can be defined. Certain convergence criteria is defined here too. Table 3.7 shows all of the *Executioner* objects added to TREX.

TransientPPEnd is a transient executioner that ends the simulation when a given *postprocessor* criteria is met. The criteria can be defined to end with the *postprocessor* gets above or below a certain value. The inputs are postprocessor, criteria, and below. Postprocessor is the name of the postprocessor that is being monitored. Criteria is the value that must be met. Below is a boolean value true if the *postprocessor* value must be below the criteria, and false if it must be above. *TransientPPEnd* is a *sub-Executioner* of *Transient* object. Since its a *sub-Executioner* if an end time is defined it will end on whichever comes first, the *postprocessor* criteria or time.

Multiapps

Multiapp objects are used to couple codes together. It can be used to couple MOOSE based applications or other codes. Table 3.8 shows all of the *Multiapp* objects added to TREX.

SteadyMultiApp is a *MultiApp* object used to couple a transient main app with steady state sub apps. Rather than creating a new instance of a steady state app every time step, the sub app creates a backup during initialization. This can be done because the geometry (mesh) never changes, only initial conditions on the mesh. By using a backup the run time can be decreased noticeably. The sub app then starts at the backup every time step. The *SteadyMultiApp* is used when the AEH method is used on a microstructure and the effective values are applied to the an engineering scale simulation. The AEH method uses a transient executioner because the equation do not depend on time.

3.2 VALIDATING CASES

Diffusion Validation

A simple diffusion case was simulated and compared to the literature to validate the 'diffusion' *kernel* and ensure understanding of TREX and MOOSE. The case found in the literature is cesium diffusing through graphite. Since Hollandite is in early stages of development and known material properties are few, many cases use Cesium diffusing through graphite (additionally Silicon Carbide for the final case). Graphite and Silicon Carbide were chosen to replicate low diffusion coefficient of Cesium in Hollandite. For this experiment, the graphite was in a cylindrical shape and had a Cesium source disk on one end of the graphite. The graphite was then annealed at 700 C for 4 h and a concentration profile of the center of the disk was plotted [10]. The diffusion coefficient for these conditions was found to be $9 \cdot 10^{-5}\ \frac{mm^2}{s}$. Since the center was examined, the simulation in TREX could be treated as 2D. A representation of the simulation in TREX can be seen in Figure 3.3. The results from TREX matched the data found in the literature well. They can be seen in Figure 3.4.

To further validate the diffusion model a case was compared to the analytical solution of Fick's first law (Eq. 2.22). The diffusion coefficient for this case was the same as cesium in graphite, the boundary is a 5 mm by 1 mm block. The left boundary (x=0) has an infinite source of cesium holding the boundary to a constant concentration of 1. The flux at the top, bottom and right boundary are set to zero (cesium cannot leave the mesh). The results can be seen in Figure 3.5. Since the analytical solution to Fick's first law (Eq. 2.22) assumes an infinite boundary it can be seen when the cesium reaches the boundary the models start to differ. This is expected because once the cesium reaches the boundary the simulated case can no longer be considered infinitely long. The blue lines are 4 hours, green are 15 hours, and red are 30 hours.

Thermal Validation

To validate the thermal model in MOOSE a case found in the literature was again compared with an equivalent simulation. The case is Heat release from borosilicate glass wasteform in a stainless steel canister. The material properties of the borosilicate glass and stainless steel are in Tables 3.9 and 3.10. The heat generation versus time was in an article by Savannah River Site. The temperature at the edge of the canister will be equal to a function of the ambient temperature. The decay heat and ambient air function is created during the simulation by TREX from data found in the literature [14]. The geometry can be seen in Figures 3.6, 3.7, and 3.8. The results can be seen in Figure 3.9.

Radioactive Decay and Generation Validation

To validate 'RadioactiveDecay' a decay case will be compared to the analytical solution (Eq. 2.26) and the conservation of mass will be used yo validate 'RadioactiveGeneration'. The case will be 482 g ($1.547 \cdot 10^{15}$ Bq) of Cesium 137 with a half life of 30.1 years. This would be the typical mass of Cs-137 in a waste canister. A typical waste canister contains 1.5 fuel assemblies. It is known that $1.115 \cdot 10^5$ Ci is discharged per tonne of heavy metal [18] and $3.7 \cdot 10^{10}$ Bq per Ci. Knowing a PWR assembly contains 0.25 tonne of heavy metal [1]. This gives:

$$(0.25)(1.5)(1.115 \cdot 10^5)(3.7 \cdot 10^{10}) = 1.547 \cdot 10^{15}\ Bq \tag{3.6}$$

To convert this to mass the following equation is used:

$$M = \frac{A \cdot m_a}{N_A \lambda} \tag{3.7}$$

where A is activity in Bq (decays per second), m_a is atomic mass (137 fo Cesium 137), N_A is Avogadro number ($6.022 \cdot 10^{23}$ 1/mol) and λ is the decay constant in 1/s ($7.30 \cdot 10^{-10}$ 1/s for cesium 137). Knowing this the typical mass of Cs-137 in a waste canister is 482 g. All of the cesium will decay to barium making p equal

to one and since cesium beta decays to barium the difference in mass is negligible making m_{da}/m_d equal to one also. The resualts for this case can be seen in Figure 3.10. The dark blue line is the mass of cesium 137 and the orange line is the mass of barium 137. The red dashed line is the analytical solution for the decay of cesium 137. Since the decay and generation is a 1 to 1 ratio the conservation of mass (the sum of the two isotopes) can be used to validate the *kernel*. The sum of the two isotopes for all time is equal to the initial amount of cesium 137 (482 g) validating the *kernel*. The conservation of mass is the green line on the graph.

A second case was modeled to further validate 'RadioactiveDecay' with diffusion. The second case was the decay of radionuclide A to radionuclide B with diffusion. All of A will beta decay to B making both p and m_{da}/m_d equal to one. The necessary condition and properties for radionuclides A and B can be seen in Table 3.11. The geometry will be the canister described earlier except there will be not be a stainless steel outer layer. The geometry can be seen in Figures 3.6, 3.7, and 3.8. The boundarys of the canister have a constant concentration of zero. The resualts can be seen in Figure 3.11. The dark blue solid line is the mass of radionuclide A in the canister, the dashed line is the amount of mass released of radionuclide A, the red solid line is the mass of radionuclide B in the canister, and the dashed line is the amount of mass released of radionuclide B. Again the green line is the sum (released + contained). By the law of conservation of mass, the sum should be equal to the initial mass validating the *kernel*.

Adaptive Time Step Validation

To validate 'AverageTimeStepForDiffusion' *postprocessor* a simple diffusion simulation was set up. The solute is a 10 by 10 micrometer block, the left boundary interface has a constant solution concentration of 1. The solutions diffusion coefficient in the solute is 25 $\frac{\mu m^2}{s}$. The change in concentration per time step will be set to 0.05 percent using *AverageTimeStepForDiffusion* object. The graph shows that

when using *AverageTimeStepForDiffusion* over time change in concentration falls away from the user's input. The graph shows that it does stay close, but as the concentration starts to level out (approach equilibrium) the *postprocessor* is not as effective. The *postprocessor* did cut the computational demand down significantly, though. When using a constant time step the simulation took 350 steps to reach 90 percent of possible saturation. The run time for this was 2 minutes and 9 seconds (129 seconds). When using 'AverageTimeStepForDiffusion' the simulation took 20 steps to reach 90 percent of possible saturation. The run time for this was 12 seconds. By using this *postprocessor* the run time was cut down by 90 percent. In a simple diffusion simulation case, the difference in run time might only be minutes but in a large case the difference could be hours. The concentration profiles can be seen in Figure 3.13 (the concentration is normalized by the possible saturation). The concentration profiles agree and are within tolerance.

Effective Properties Validation

Effective properties are a key method to the design and development of TREX. This will allow TREX to model engineering scale simulations while still capturing the microstructure dependent properties. With this in mind, many tests were done to validate the used methods. Since the asymptotic expansion homogenization method is the most versatile it is implemented in TREX. The other two methods (Eq. 2.43, 2.45, and 2.47), though, will be used to validate the asymptotic expansion homogenization method (Eq. 2.31). The effective diffusion coefficients, for five different material structure, found using Equations 2.31, 2.43, 2.45, or 2.47 (when applicable) will be compared. These simulations will all be a 10 by 10 micrometer block, 1D, and steady state. To achieve 1D on a 2D surface the boundary on the left will have a constant value of one. The diffusion coefficient in the bulk is 0.1 $\frac{\mu m^2}{s}$ and in the grain boundary is 10 $\frac{\mu m^2}{s}$. Each preferred path will be tested for volume fractions of the fast path ranging from 0 to 0.5 by 0.1. The different structures can

be seen in Figure 3.14. The results can be seen in Figure 3.15. The solution for solid lines were found using Equation 2.31, 'X' markers were found using Equation 2.43, and 'O' markers were found using Equation 2.45 or 2.47.

It can be seen that these three methods strongly agree. After comparing the effective diffusion coefficient it is important to test how effective they are. For this validation, only the asymptotic expansion homogenization method will be used (again because it is the most versatile). The concentration profiles of heterogeneous and homogeneous material will be compared. The heterogeneous material will be the structures shown in Figure 3.14. The homogeneous material will be the equivalent effective diffusion coefficient found from AEH. It is important to note that one of the assumptions of the AEH method is the homogeneous (x) to heterogeneous (y) scale is very small ($1 << \epsilon = x/y$). To model such detail is computationally demanding. Because of this, these test will only be done for a scale of one ($\epsilon = 1$). It will be shown later the effects of not having a small scale. For the validation cases 1D and 2D were tested. The domain was 10 by 10 micrometer block, for both the microstructure and homogenized body domain. For 1D the diffusion coefficient in the grain boundary (fast path) was 10 $\frac{\mu m^2}{s}$, the Bulk (slow path) was 0.1 $\frac{\mu m^2}{s}$. The Initial concentration of the solute in the solution was 0 percent. The left boundary then had a constant concentration of one. For 2D isotropic and anisotropic were tested, the initial concentration of the solute in the solution was 100 percent and the four boundaries had a constant concentration of zero. This was done because the wasteform case will model concentration being released from the waste. The diffusion coefficients were the same as 1D, except for 2D anisotropic, the grain boundary in the y direction was 0.01 $\frac{\mu m^2}{s}$. The results can be seen in Figures 3.16 - 3.30. The concentration profiles for the homogeneous 1D and 2D isotropic structures lined up well with the concentration profiles of the heterogeneous 1D and 2D isotropic structures. The concentration profiles for the homogeneous 2D anisotropic structures lined up well with the concentration pro-

files of the heterogeneous 2D anisotropic structures, too, except for the structure with parallel paths parallel to the x-axis (Fg. 3.26). This error is caused by the small scale. To validate this the size of the homogenized body domain was changed. Due to computational limits, two domains were tested, one with a larger scale (scale = 2) (Fg. 3.31) and the other with a smaller scale (scale = 0.5) (Fg. 3.32). The error of the large, original and small scale were graphed. It can be seen that as the scale decreased the error decreased. This would be expected since the equation assumes the scale is very small ($1 << \epsilon = x/y$). Since this method will be used to apply effective properties found from the microstructure to the engineering scale simulation the scale will be extremly small. The ratio of the microstructure to engineering scale will roughly be $\epsilon = 1/10^6$. To have the homogenous and heterogeneous concentration profiles to agree the smallest scale was $\epsilon = 1/2$, but most concentration profiles agreed with a scale of $\epsilon = 1/1$.

Table 3.1 Added *Kernel* objects

Name	Description
CylinderHeatSource	Heat generation term, confined to a cylindrical shape.
HomogenizationDiffusionCoefficient	Asymptotic Homogenization method for an isotropic diffusion coefficient.
HomogenizationAnisoDiffusionCoefficient	Asymptotic Homogenization method for an anisotropic diffusion coefficient.
RadioactiveDecay	Radioactive decay model.
RadioactiveGeneration	Radioactive daughter generation model.

Table 3.2 Added *Aux Kernel* objects

Name	Description
FluxAux	Calculates flux from concentration and an isotropic diffusion coefficient.
FluxAnisoAux	Calculates flux from concentration and an anisotropic diffusion coefficient.

Table 3.3 Added *Initial Condition* objects

Name	Description
BoundingBlockIC	Creates an initial value on given blocks.
BoundingCircleIC	Creates an initial value in a given circle.

Table 3.4 Added *Material* objects

Name	Description
AnisoDiffusionCoefFromPostprocessor	Defines a diffusion coefficient from postprocessor values.
ArrheniusDiffusionCoef	Generic Diffusion Coefficient described by the Arrhenius equation.
ArrheniusAnisoDiffusionCoef	Generic Anisotropic Diffusion Coefficient described by the Arrhenius equation.
BoxDiffusionCoef	Restricts diffusion coefficients to a box.
CalcDiffusionCoef	Creates diffusion coefficient from flux and concentration gradient.
GenericTensorConstantMaterial	Creates an anisotropic generic material property.
GrainMaterialDiffusion	Creates a diffusion coefficient from a map of the microstructure.
GrainAnisoMaterialDiffusion	Creates an anisotropic diffusion coefficient from a map of the microstructure.
GrainMaterialThermal	Creates thermal properties from a map of the microstructure (Only needed for different phases).
LineGrains	Creates a 'microstructure' with parallel lines.
SquareGrains	Creates a 'microstructure' with square grains.
HexagonGrains	Creates a 'microstructure' with hexagonal grains.
CircleGrains	Creates a 'microstructure' with circle grains.
WasteContainer	Creates thermal properties and diffusion properties for borosilicate glass container.
WasteAnisoContainer	Creates thermal properties and anisotropic diffusion properties for borosilicate glass container.
DiffusionProperties*	Diffusion coefficient of different materials.
ThermalProperties*	Thermal properties of different materials.

Table 3.5 Added *Postprocessor* objects

Name	Description
AverageTimeStepForDiffusion	Finds a future time step for diffusion that will change the concentration by a prescribed amount.
HomogenizedDiffusionCoefficient	Asymptotic Homogenization method for an isotropic diffusion coefficient.
HomogenizedAnisoDiffusionCoefficient	Asymptotic Homogenization method for an anisotropic diffusion coefficient.

Table 3.6 Added *Action* objects

Name	Description
PointValueOnLineAction	Finds the value of certain points on a line.

Table 3.7 Added *Executioner* objects

Name	Description
TransientPPEnd	Creates a transient executioner that ends on a given postprocessor criteria.

Table 3.8 Added *MultiApp* objects

Name	Description
SteadyMultiApp	Runs a steady state TREX app.

Table 3.9 Thermal Material Properties

Properties	Borosilicate Glass	Stainless Steel
Specific Heat ($J/kg-K$)	$467.7 + 1.2057 \cdot T$	477
Thermal Conductivity ($W/m-K$)	0.95	14.9
Density (kg/m^3)	2027.4	7900

Table 3.10 Heat Output

Years	Heat Output ($W/Canister$)	Heat Output (W/m^3)
1	730	1000
5	414	567.12
10	257	352.05
100	39.2	53.699

Table 3.11 'RadioactiveGeneration' case properties

Radionuclide	Initial Mass	Half Life	Diffusion Coefficient
A	482 g	30.1 years	0.00055 $\frac{m^2}{y}$ ($1.7428 \cdot 10^{-5} \frac{mm^2}{s}$)
B	0 g	–	0.00065 $\frac{m^2}{y}$ ($2.0597 \cdot 10^{-5} \frac{mm^2}{s}$)

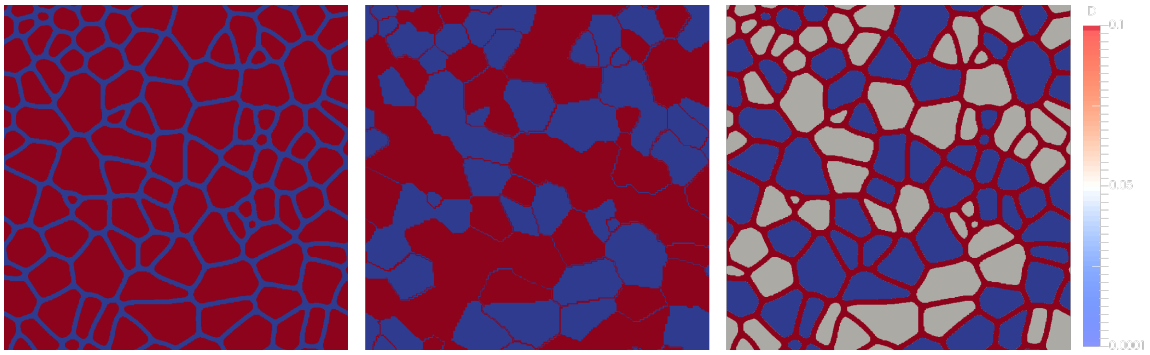

Figure 3.1 Multi Phase Diffusion Coefficient

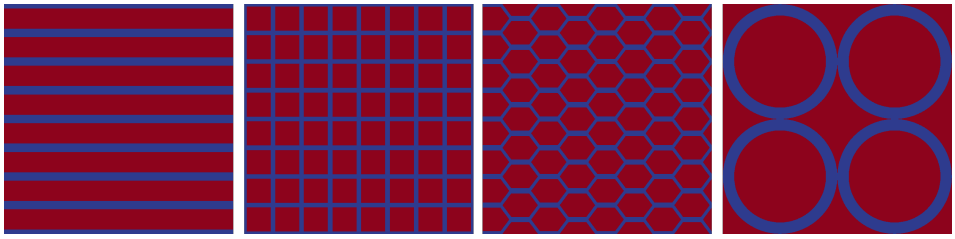

Figure 3.2 Multi Grains

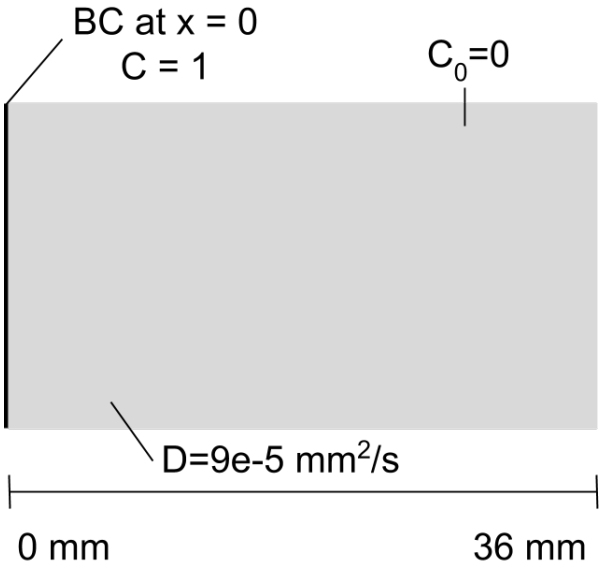

Figure 3.3 Diffusion Validation a: initial and boundary conditions

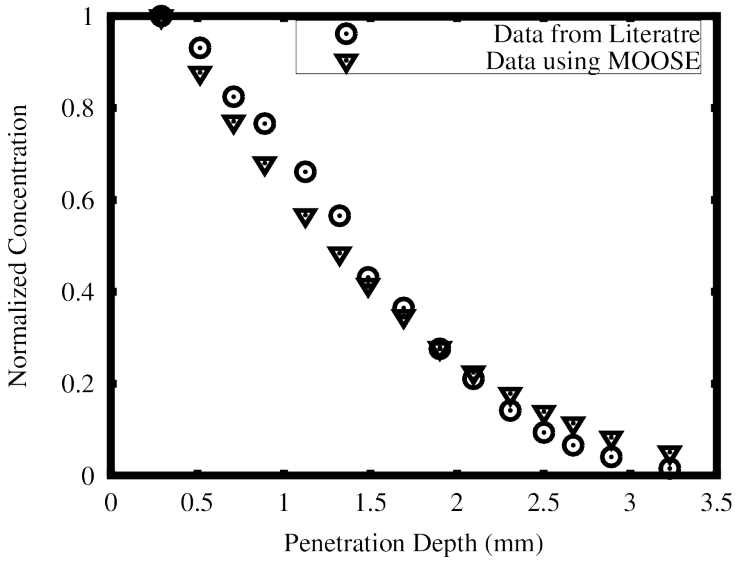

Figure 3.4 Diffusion Validation a: Results

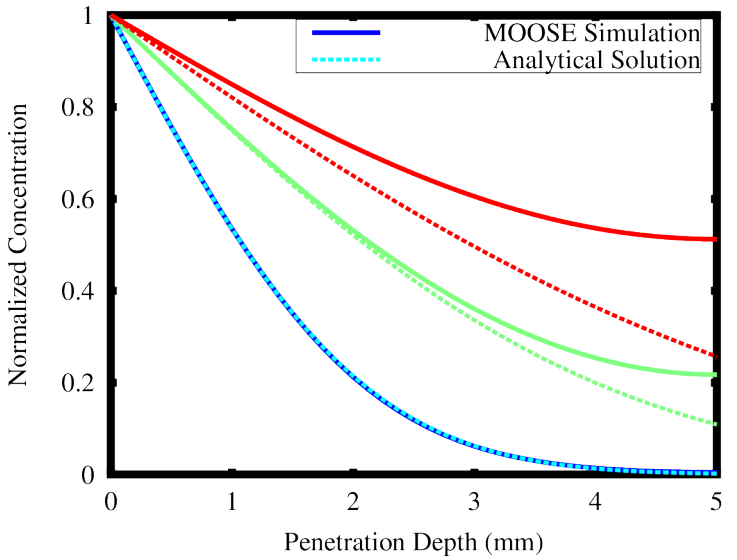

Figure 3.5 Diffusion Validation b: Results

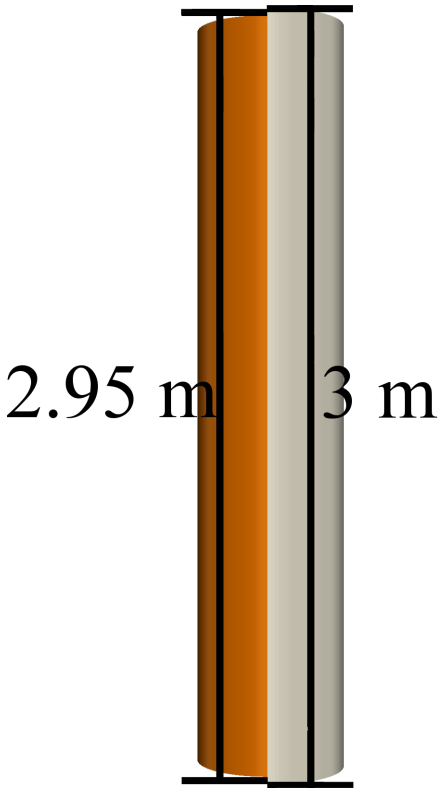

Figure 3.6 Side View of Canister

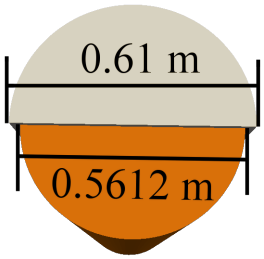

Figure 3.7 Top View of Canister

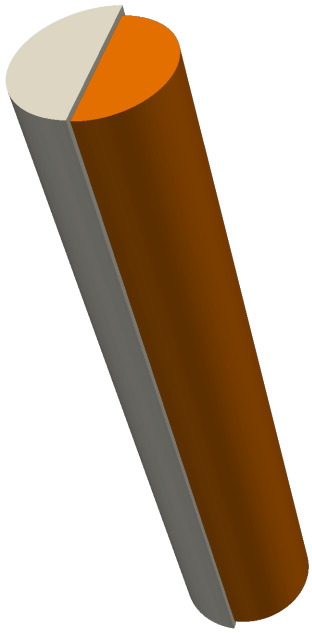

Figure 3.8 Canister

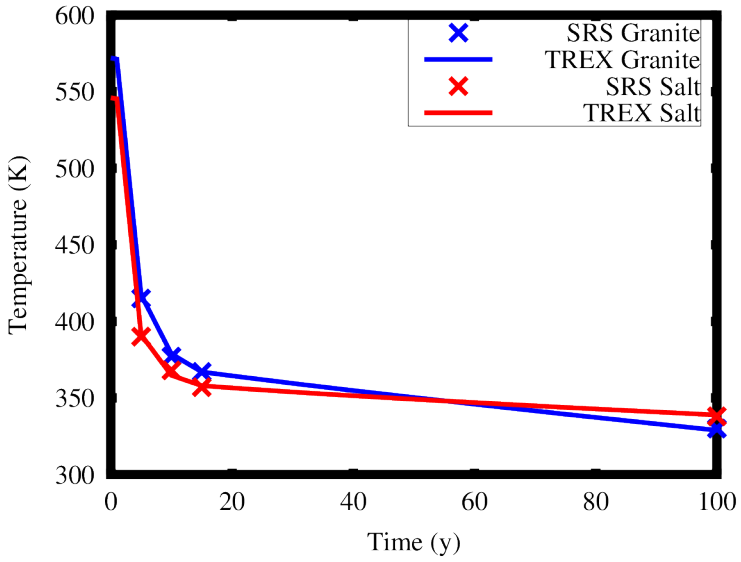

Figure 3.9 Thermal Validation: Results

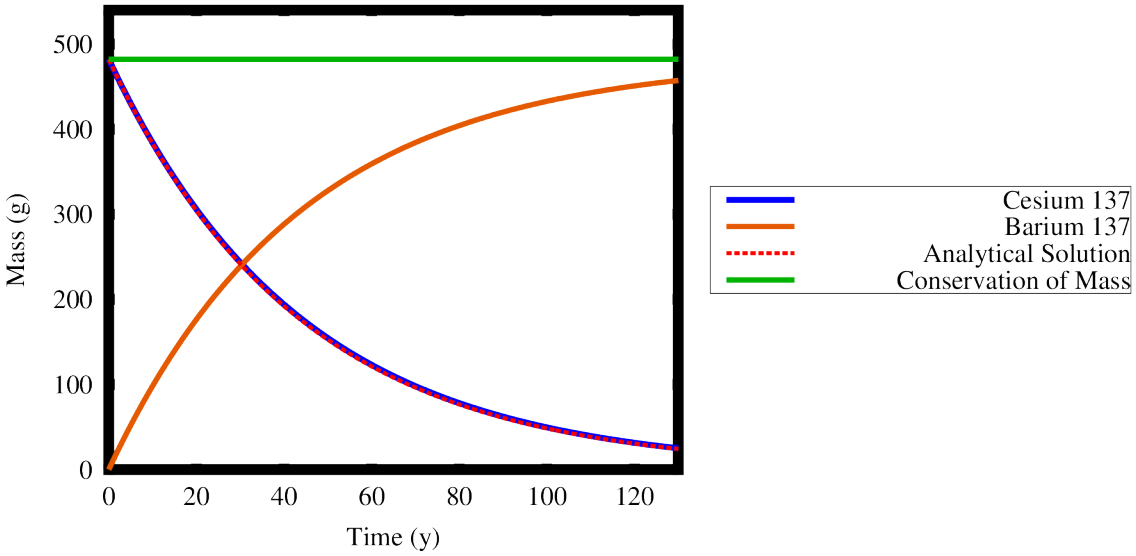

Figure 3.10 Radionuclide Decay Validation

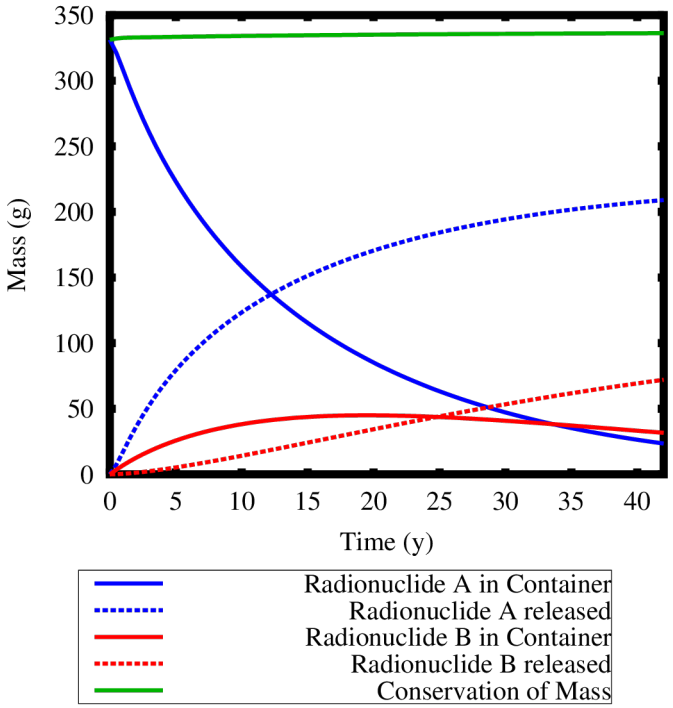

Figure 3.11 Radionuclide Decay, Generation and Diffusion Validation

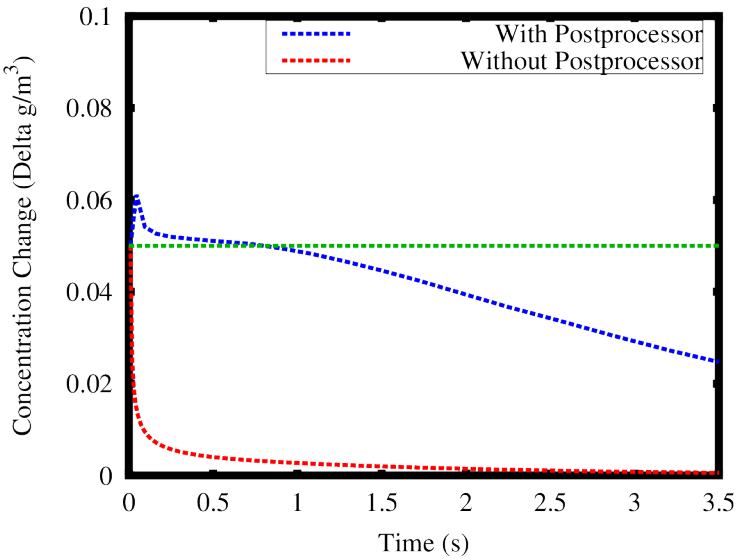

Figure 3.12 AverageTimeStepForDiffusion Validation

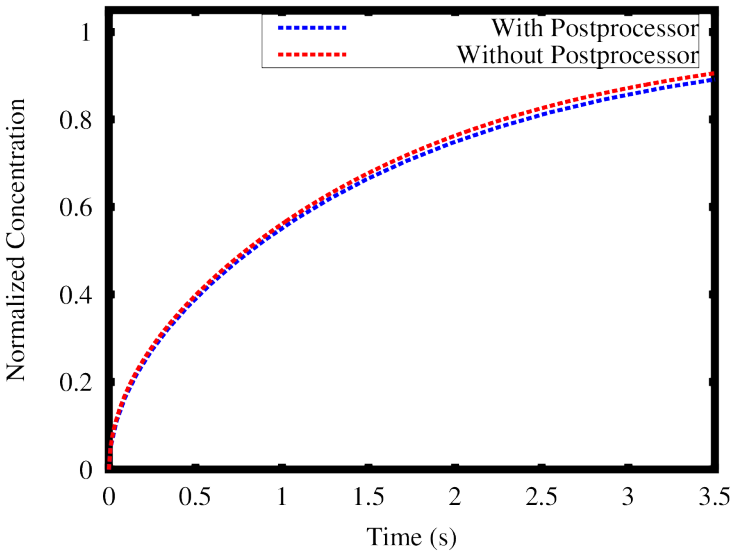

Figure 3.13 AverageTimeStepForDiffusion Concentration Validation

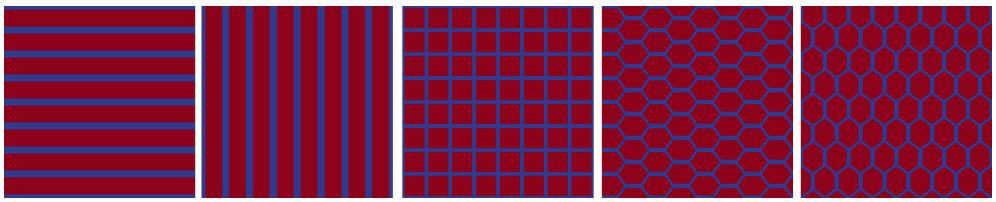

Figure 3.14 Preferred Paths Structures

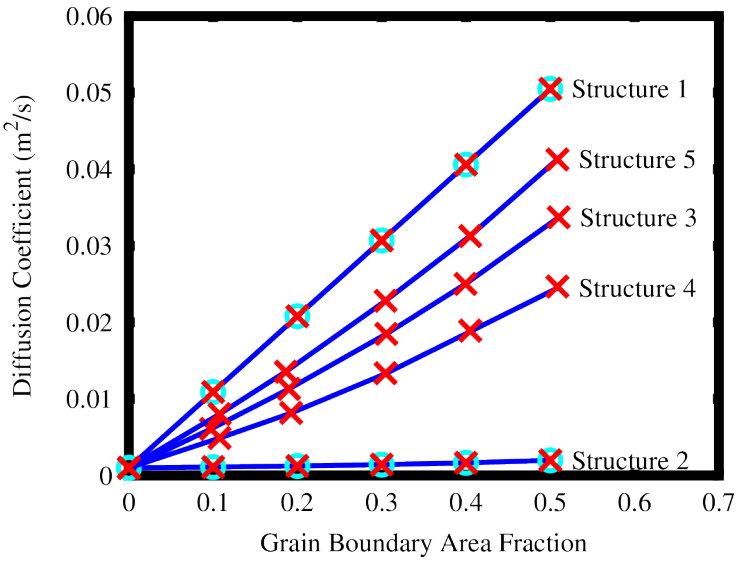

Figure 3.15 1D Validation

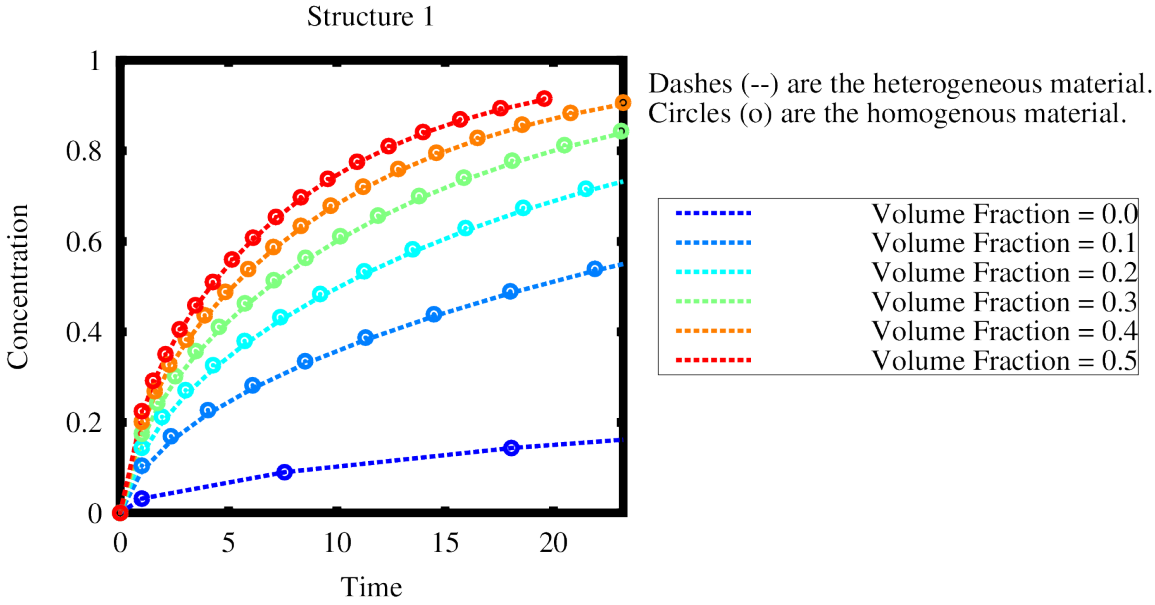

Figure 3.16 Structure One 1D Isotropic Validation

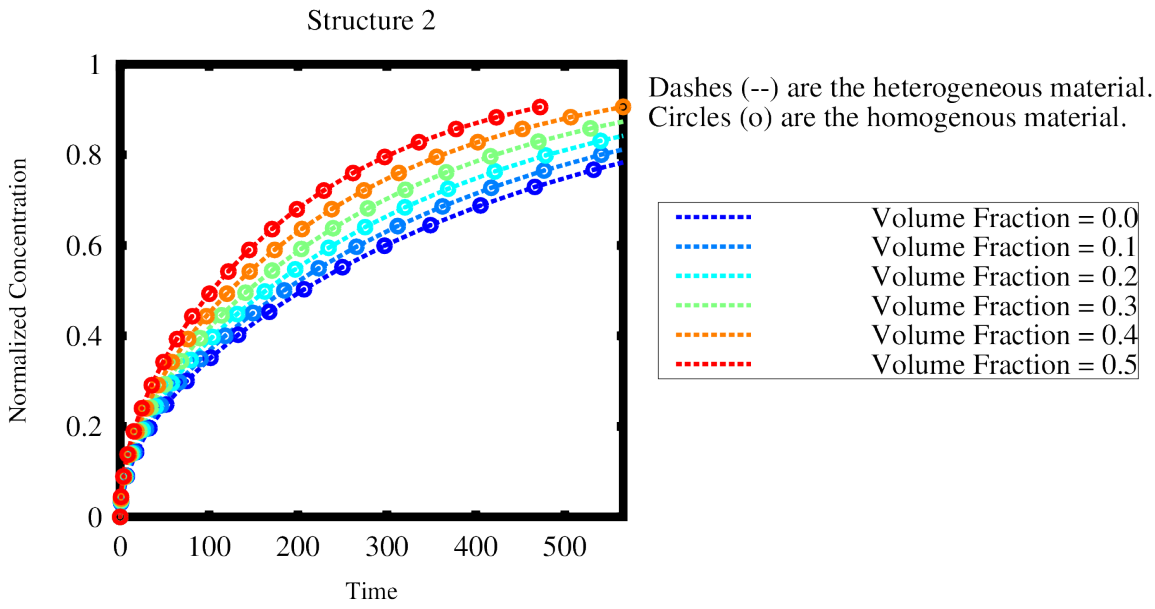

Figure 3.17 Structure Two 1D Isotropic Validation

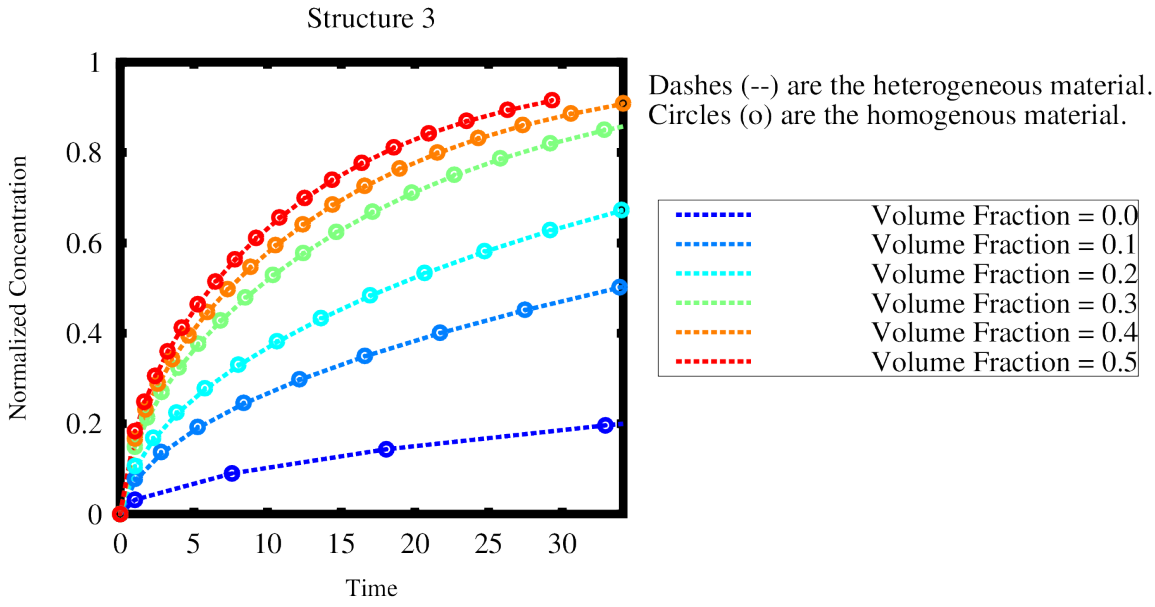

Figure 3.18 Structure Three 1D Isotropic Validation

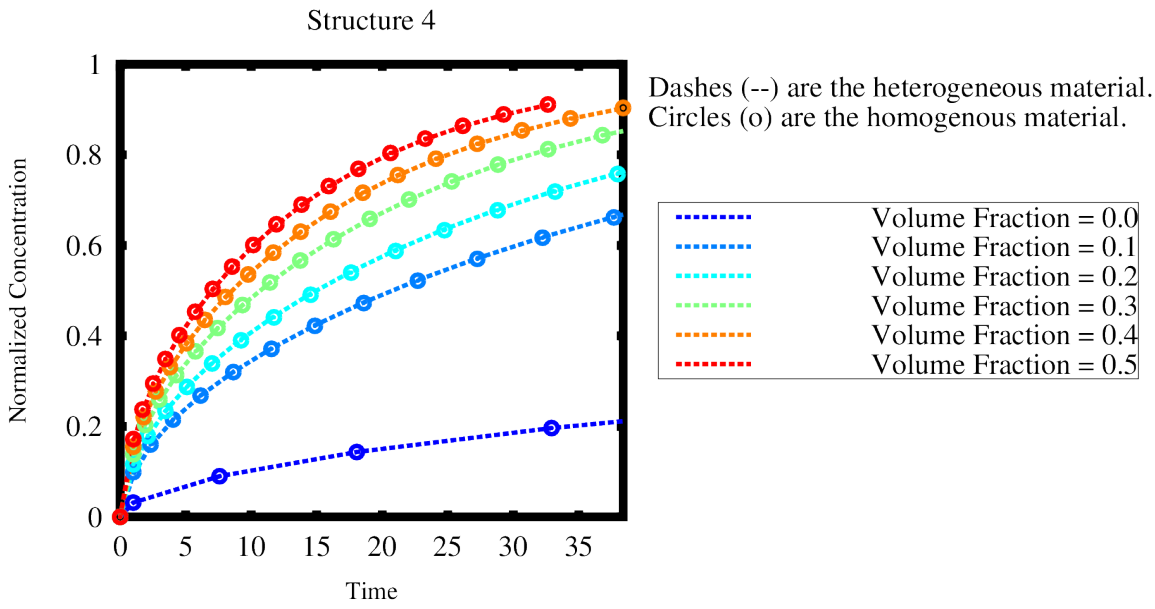

Figure 3.19 Structure Four 1D Isotropic Validation

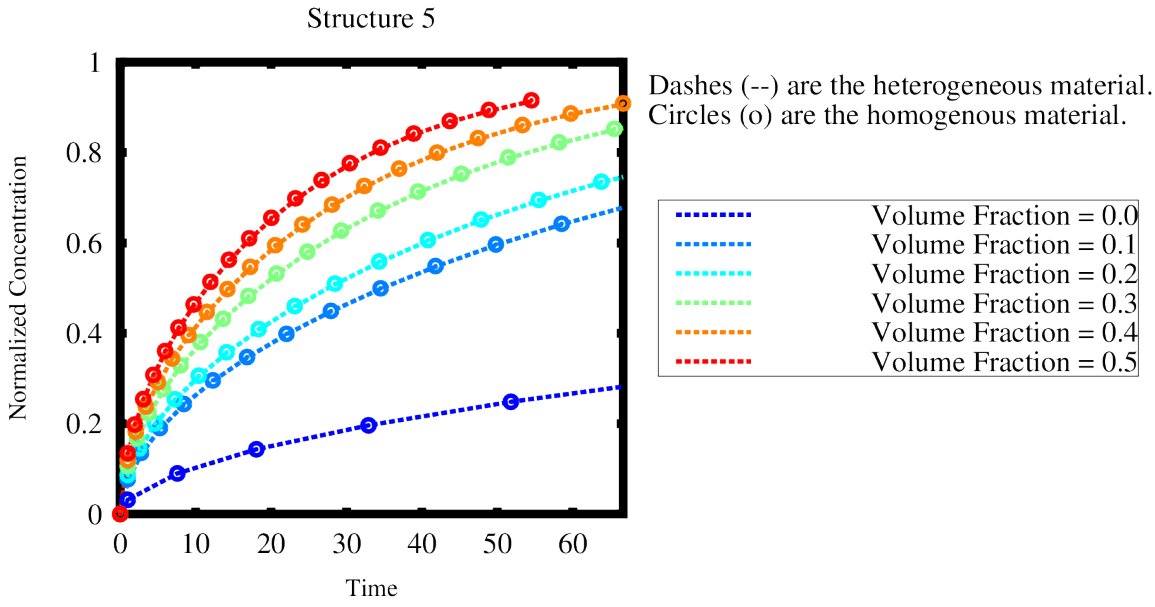

Figure 3.20 Structure Five 1D Isotropic Validation

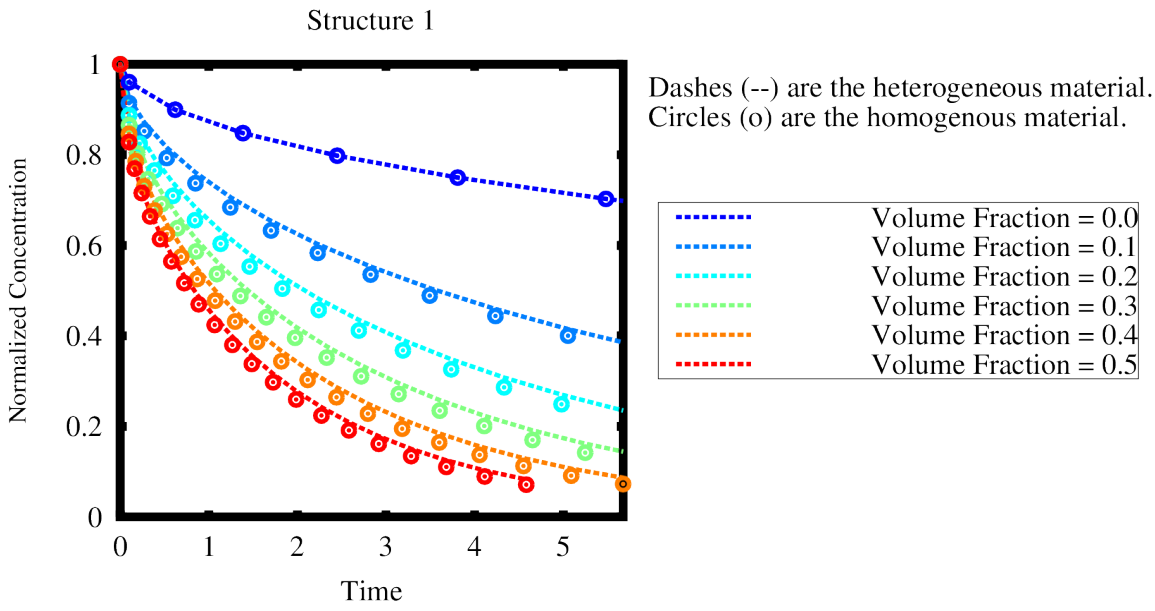

Figure 3.21 Structure One 2D Isotropic Validation

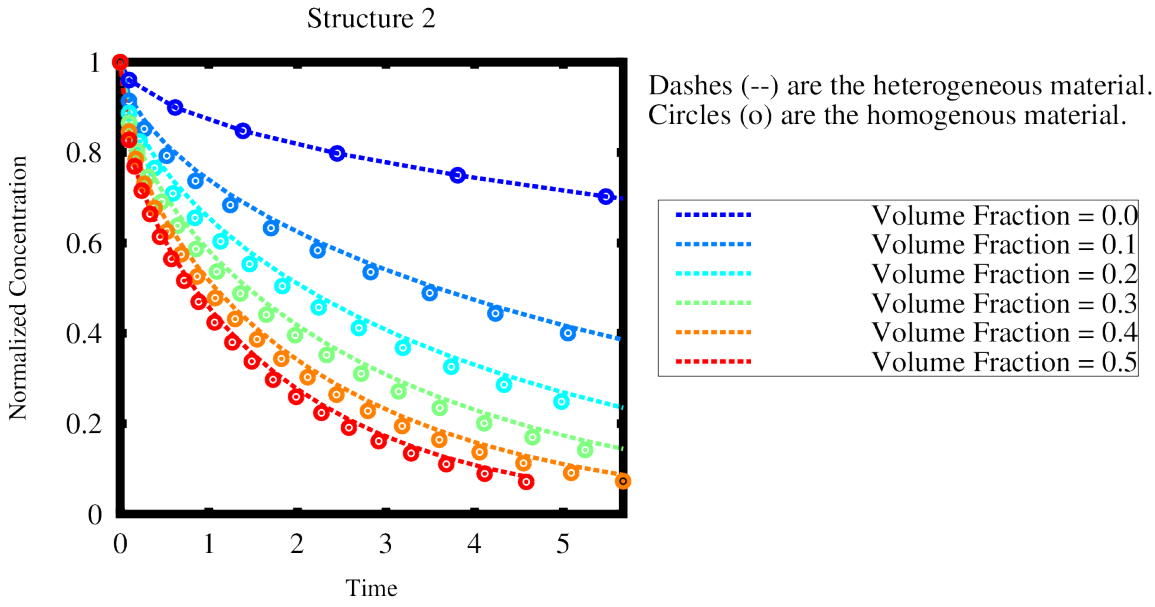

Figure 3.22 Structure Two 2D Isotropic Validation

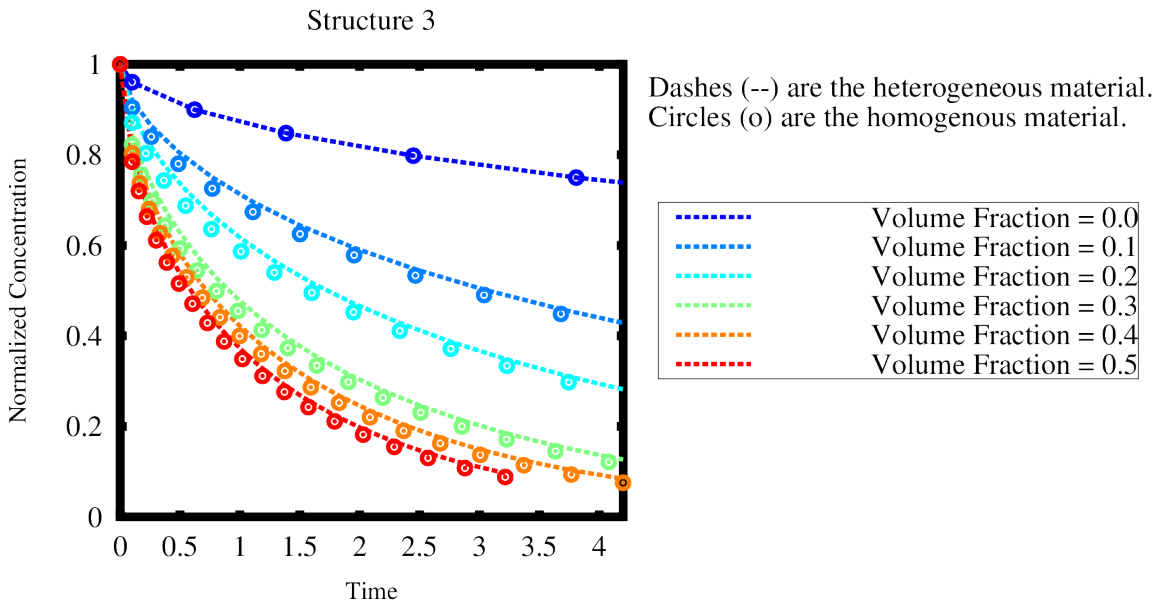

Figure 3.23 Structure Three 2D Isotropic Validation

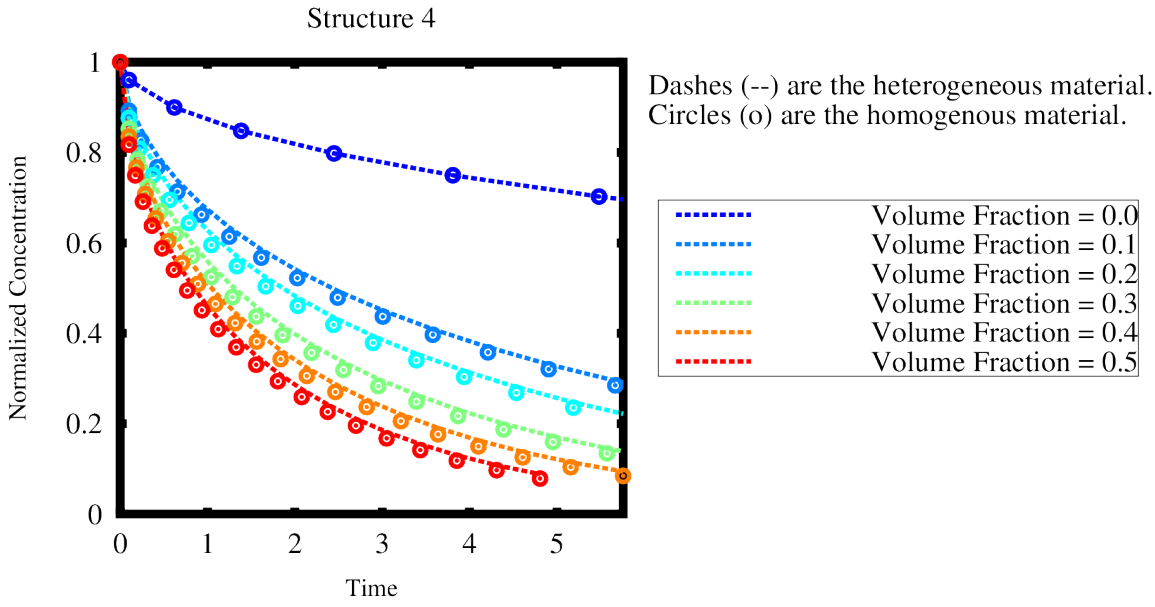

Figure 3.24 Structure Four 2D Isotropic Validation

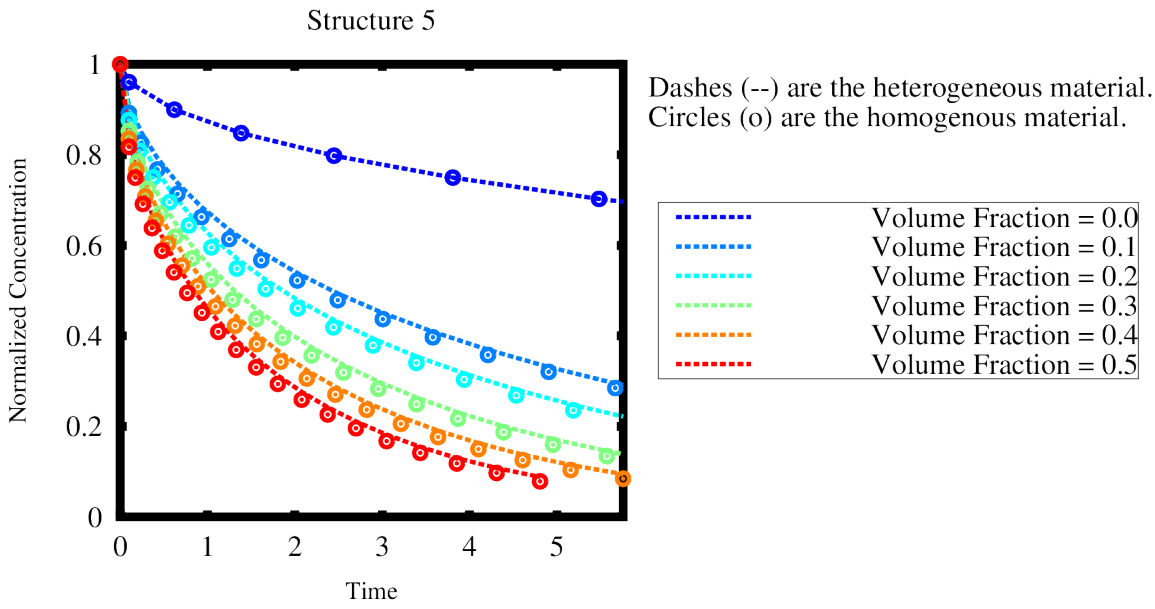

Figure 3.25 Structure Five 2D Isotropic Validation

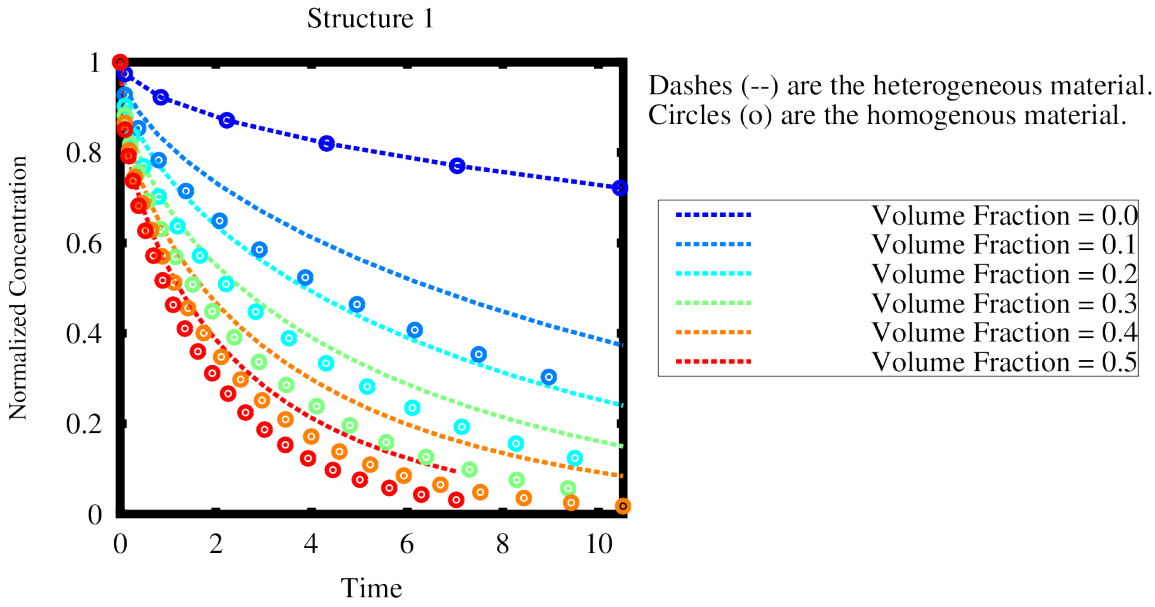

Figure 3.26 Structure One 2D Anisotropic Validation

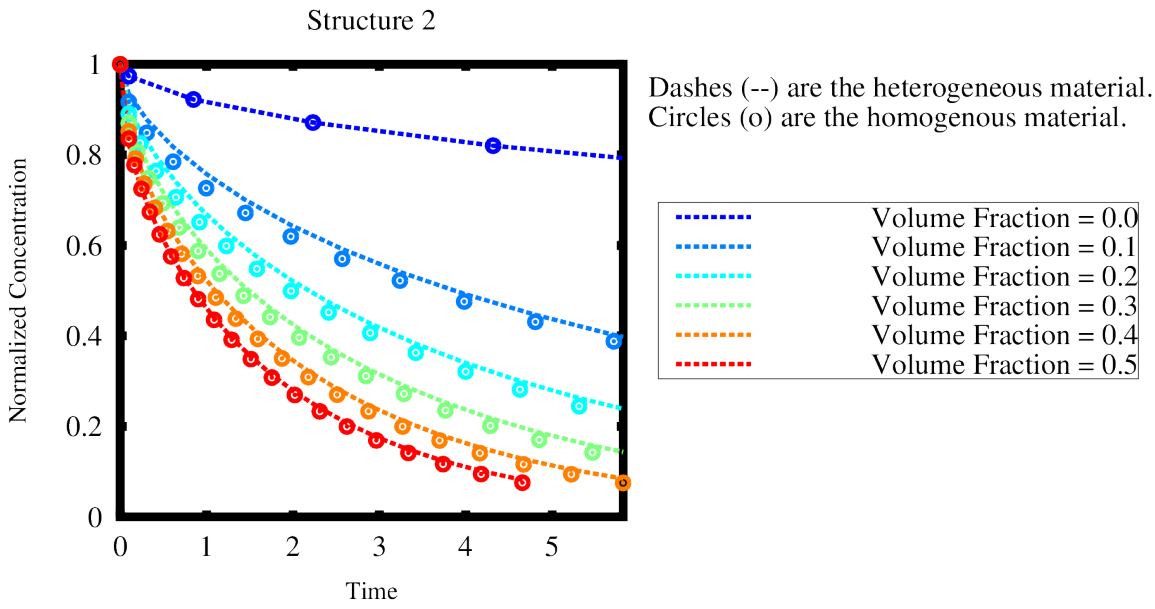

Figure 3.27 Structure Two 2D Anisotropic Validation

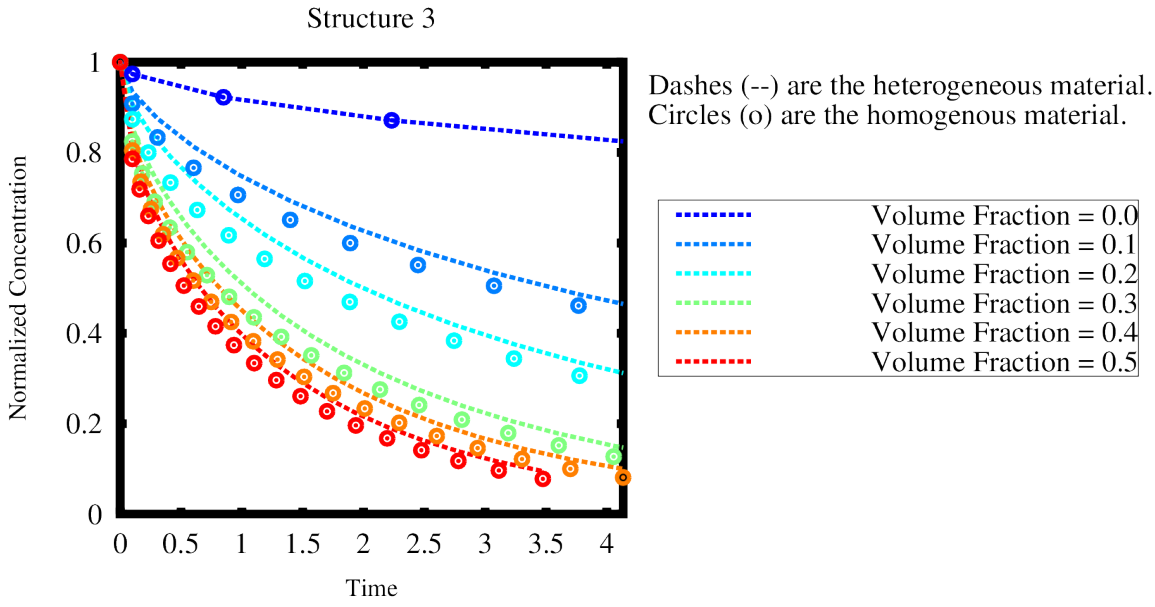

Figure 3.28 Structure Three 2D Anisotropic Validation

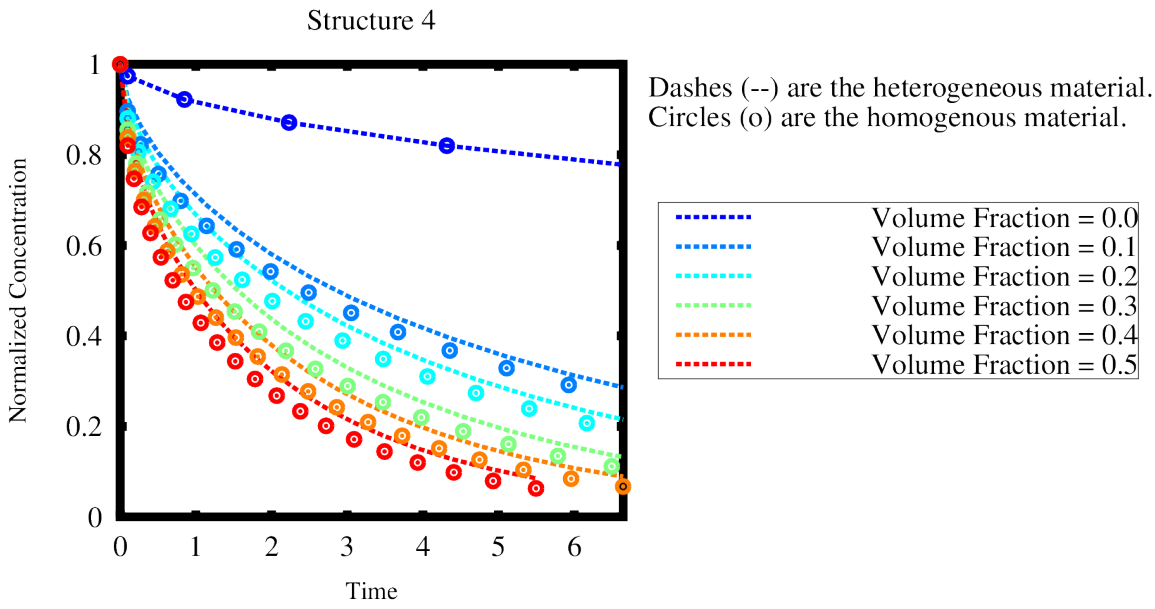

Figure 3.29 StructureFourTwoDACValidation

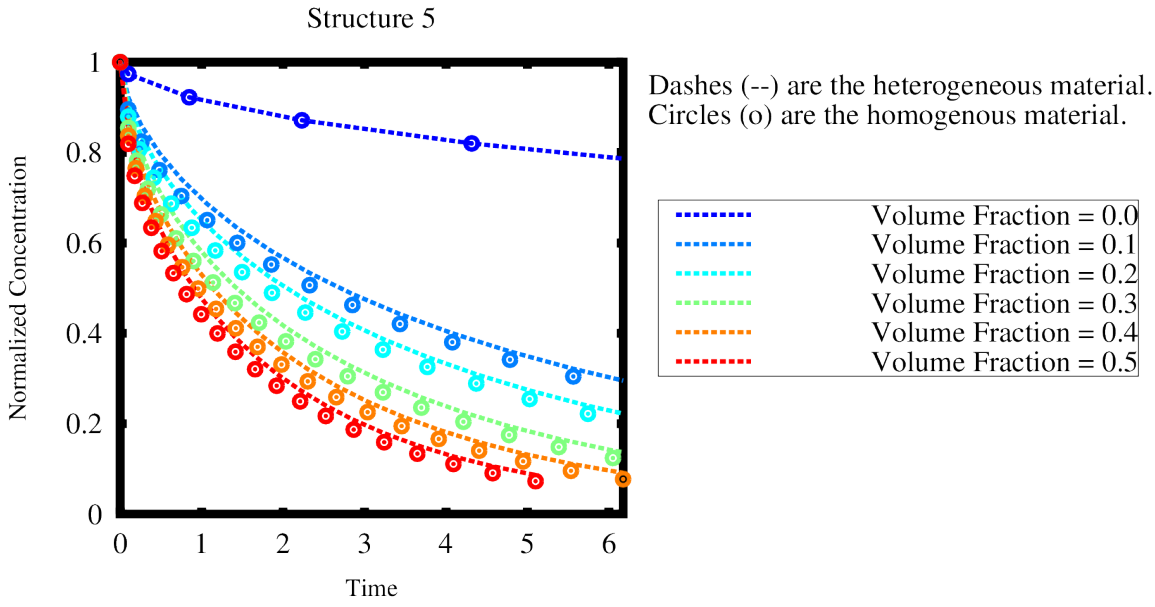

Figure 3.30 Structure Five 2D Anisotropic Validation

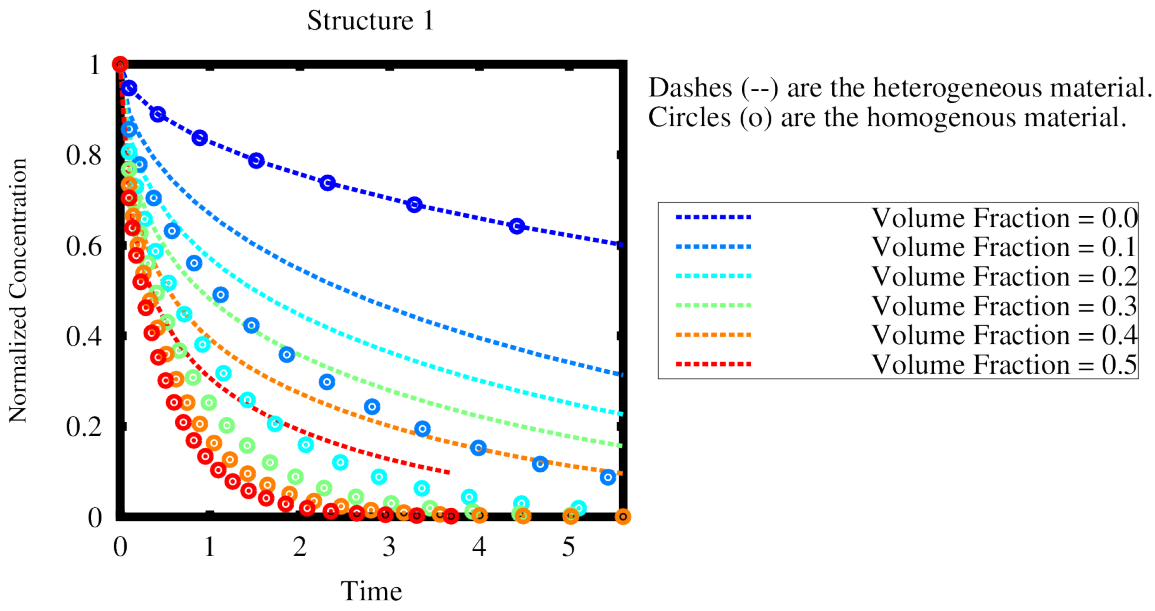

Figure 3.31 Structure One Small Domain 2D Anisotropic Validation

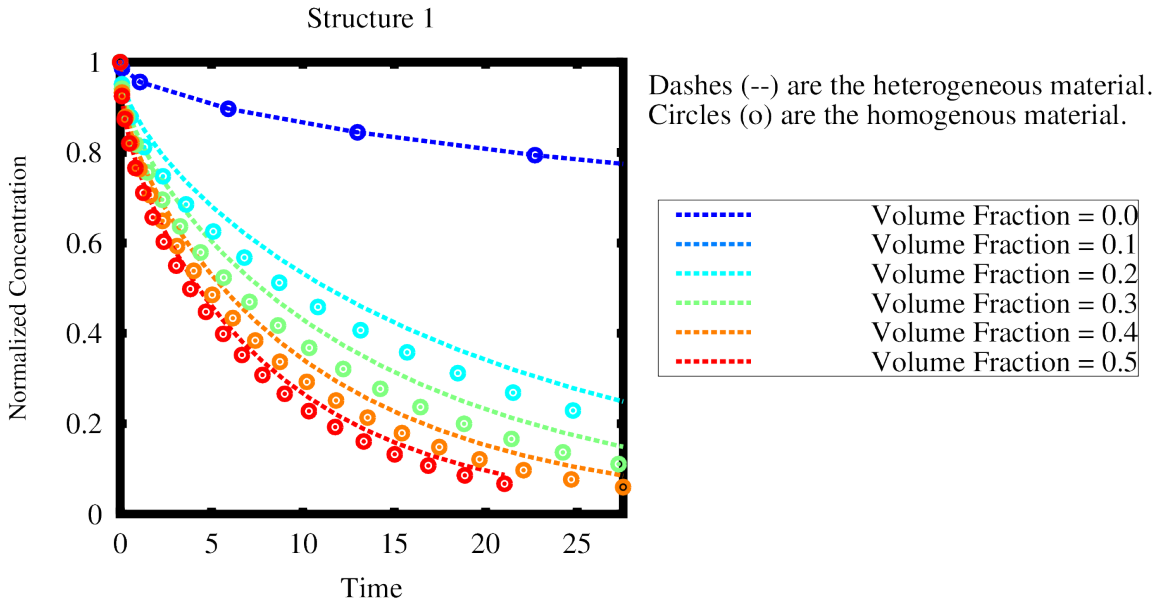

Figure 3.32 Structure One Large Domain 2D Anisotropic Validation

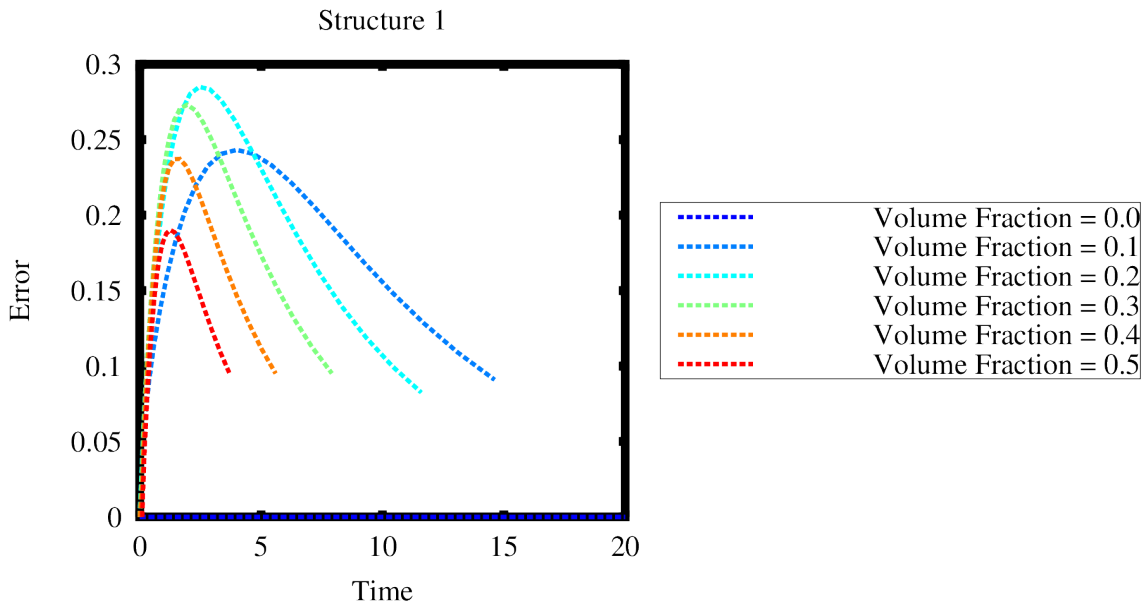

Figure 3.33 Structure One Small Domain 2D Anisotropic Validation Error

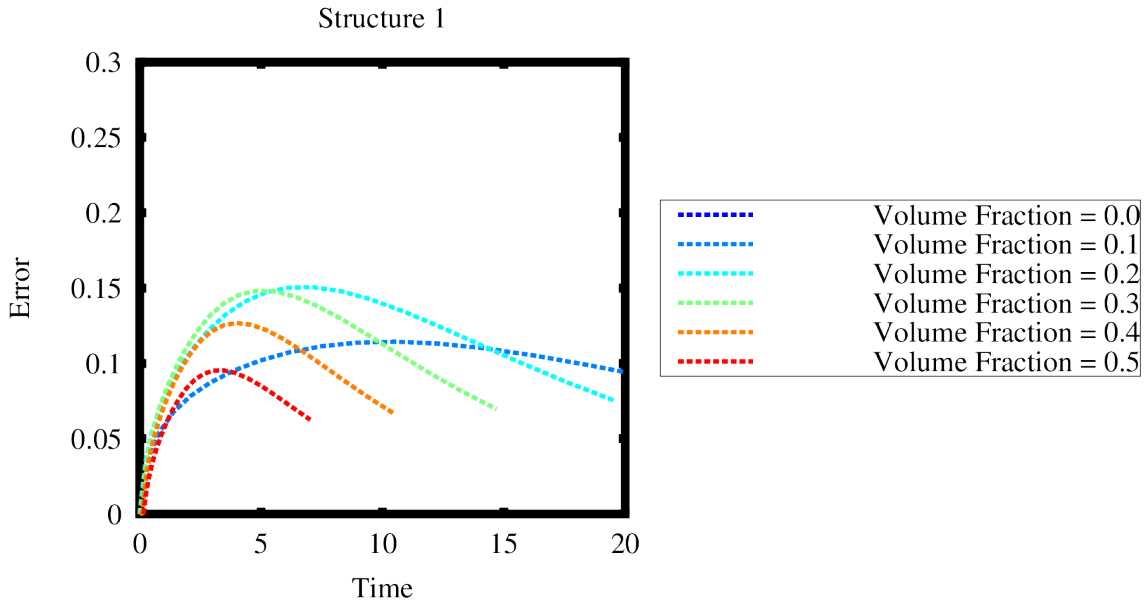

Figure 3.34 Structure One Original Domain 2D Anisotropic Validation Error

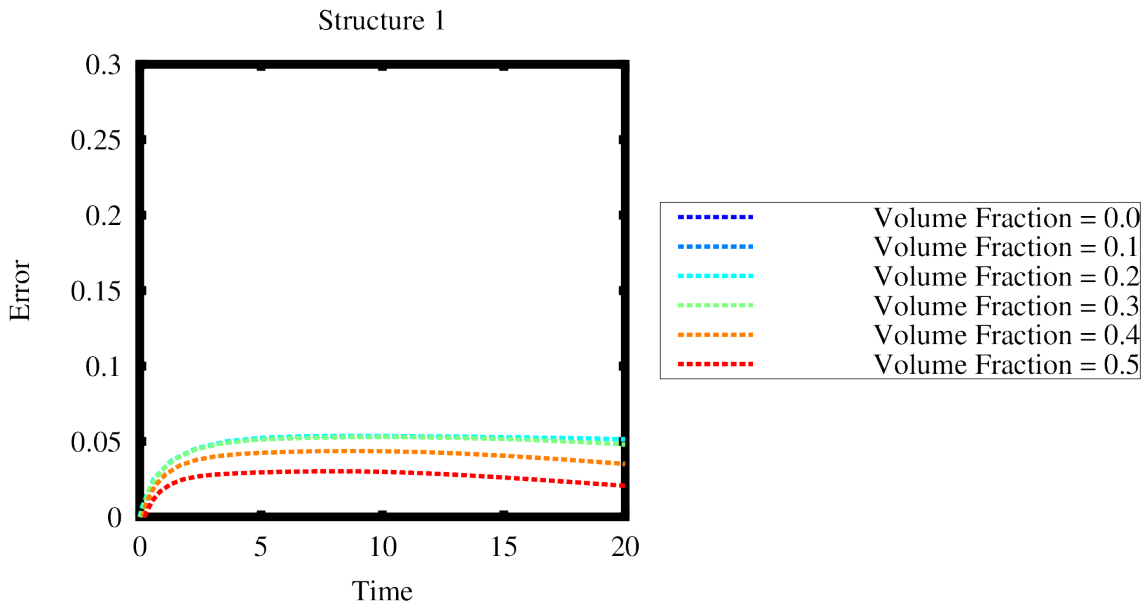

Figure 3.35 Structure One Large Domain 2D Anisotropic Validation Error

```
[./xyCylinder]                  # The reference name
  type = CylindricalHeatSource  # The name of the kernel
  variable = temp               # The name of the variable this kernel acts on
  value = 2                     # The value of the volumetric heat generation term, if applicable
  function = power_function     # The function of the volumetric heat generation term, (can be a function of time, position, etc..)
  radius = 2                    # Radius of the cylinder, units are in the same units as the mesh
  height = 2.95                 # Height of the cylinder, units are in the same units as the mesh
  normal_axis = 'z'             # The axis parallel to the height OR normal = '0 0 1'
  origin = '0 0 1.475'          # The orgin of the center of the cylinder
[../]                           # End of entry
```

Input 3.1 CylindricalHeatSource Input

```
[./cesiumDecay]                 # The reference name
  type = RadioactiveDecay       # The name of the kernel
  variable = Cs_137_conc        # The name of the variable this kernel acts on
  half_life = 65008656          # The half life of Cesium in seconds (must be in the same units as the rest of the input)
[../]                           # End of entry
```

Input 3.2 RadioactiveDecay Input

```
[./bariumGeneration]            # The reference name
  type = RadioactiveGeneration  # The name of the kernel
  variable = Ba_137_conc        # The name of the variable this kernel acts on
  coupled = Cs_137_conc         # The name of the variable this kernel acts on
  half_life = 65008656          # The half life of Cesium in seconds (must be in the same units as the rest of the input)
  mass_ratio = 1                # Since cesium beta decays to barium the mass ratio is one
  decay_chain_ratio = 1         # Since cesium fully decays to barium the decay chain ratio is one (cesium does first decay to metastable barium but the half life is small)
[../]                           # End of entry
```

Input 3.3 RadioactiveGeneration Input

```
[./CylRegionDiffusionCoef]                # The reference name
  type = AnisoArrheniusDiffusionCoefFromPostprocessor  # The name of the kernel
  diffusion_coef = 'D'                    # The name of the material property referenced by other objects
  D_xx = 'sim1_D_xx sim2_D_xx'            # The name of the postprocessors with the value of the D_xx (The first value is the center, the second is the boundary)
  D_xy = 'sim1_D_xy sim2_D_xy'            # The name of the postprocessors with the value of the D_xy
  D_xz = 'sim1_D_xz sim2_D_xz'            # The name of the postprocessors with the value of the D_xz
  D_yx = 'sim1_D_yx sim2_D_yx'            # The name of the postprocessors with the value of the D_yx
  D_yy = 'sim1_D_yy sim2_D_yy'            # The name of the postprocessors with the value of the D_yy
  D_yz = 'sim1_D_yz sim2_D_yz'            # The name of the postprocessors with the value of the D_yz
  D_zx = 'sim1_D_zx sim2_D_zx'            # The name of the postprocessors with the value of the D_zx
  D_zy = 'sim1_D_zy sim2_D_zy'            # The name of the postprocessors with the value of the D_zy
  D_zz = 'sim1_D_zz sim2_D_zz'            # The name of the postprocessors with the value of the D_zz
  radius = 0.2806                         # The value of the radius
  z = 1.5                                 # The distance of the z from the center (if cylinder is in yz plane x = height)
  position_from_center = '0 0 0'          # The distance of the center of the shape from the center of the axis
[../]                                     # End of entry
```

Input 3.4 Cylinder Region Diffusion Coefficient Input

```
[./RectRegionDiffusionCoef]          # The reference name
  type = AnisoArrheniusDiffusionCoefFromPostprocessor  # The name of the kernel
  diffusion_coef = 'D'               # The name of the material property referenced by other objects
  D_xx = 'sim1_D_xx sim2_D_xx'       # The name of the postprocessors with the value of the D_xx (The first value is the center, the second is the boundary)
  D_xy = 'sim1_D_xy sim2_D_xy'       # The name of the postprocessors with the value of the D_xy
  D_xz = 'sim1_D_xz sim2_D_xz'       # The name of the postprocessors with the value of the D_xz
  D_yx = 'sim1_D_yx sim2_D_yx'       # The name of the postprocessors with the value of the D_yx
  D_yy = 'sim1_D_yy sim2_D_yy'       # The name of the postprocessors with the value of the D_yy
  D_yz = 'sim1_D_yz sim2_D_yz'       # The name of the postprocessors with the value of the D_yz
  D_zx = 'sim1_D_zx sim2_D_zx'       # The name of the postprocessors with the value of the D_zx
  D_zy = 'sim1_D_zy sim2_D_zy'       # The name of the postprocessors with the value of the D_zy
  D_zz = 'sim1_D_zz sim2_D_zz'       # The name of the postprocessors with the value of the D_zz
  x = 1                              # The distance of the x from the center
  y = 2                              # The distance of the y from the center
  z = 1.5                            # The distance of the z from the center
  position_from_center = '0 0 0'     # The distance of the center of the shape from the center of the axis
[../]                                # End of entry
```

Input 3.5 Rectangular Region Diffusion Coefficient Input

```
[./D_B_SiCa]                         # The reference name
  type = DiffusionSiliconCarbide     # The name of the kernel

  diffusion_coef = 'D_B_SiCa'        # The name of the material property referenced by other objects
  T = 'temp'                         # The temperature of the mesh
[../]                                # End of entry
```

Input 3.6 'DiffusionProperties' of Silican Carbide Input

```
[./Thermal_StSt]                     # The reference name
  type = ThermalStainlessSteel       # The name of the kernel
  T = temp                           # The temperature of the mesh
[../]                                # End of entry
```

Input 3.7 'ThermalProperties' of Stainless steel Input

```
[./PointValueOnLine]                # Must be the name of as the action! This cannot change, because type is not needed

  variable = u                      # The variable of the kernel
  point1 = '0 0.5 0'                # The starting point
  point2 = '10 0.5 0'               # The ending point
  num_out_per_unit 100              # The number of points on the line between point1 and point2
[../]                               # End of entry
```

Input 3.8 'PointValueOnLineAction'

CHAPTER 4

RESULTS

After developing these objects, an analog case was developed to show the capability of these tools. Even though Hollandite is in the early stages of development some modeling details are known that can help design a great analog case. As previously discussed some of the key modeling details are the diffusion coefficient will be path, phase, direction and temperature dependent and there will be decay heat causing a temperature gradient (at least early on). To capture these key details a multiscale simulation of Cesium diffusing through silicon carbide with a second phase precipitation of graphite was simulated. Again Cesium was chosen because it is an important radioisotope in nuclear waste and is more mobile when released into the environment. Silicon Carbide and Carbon were chosen as the solutes because there was extensive data on Cesium diffusing through Silicon Carbide and Carbon. Since Silicon Carbide is not the same as hollandite for all properties some silicon carbide properties were created to exemplify the capability of the created tools.

To capture all of these details, the analog case is defined as followed. There will be one main simulation with two sub-simulations. The main simulation will simulate the entire wasteform (silicon carbide in our case) including the canister. The canister will be a stainless steel cylinder with an inner and outer diameter of 0.56 m and 0.61 m and an inner and outer height of 2.95 m and 3.00 m, respectively. This can be seen in Figures 3.6, 3.7, and 3.8. In these Figures, orange is the

wasteform (Silicon Carbide and Carbon) and silver is the material of the canister (stainless steel in this case). The stainless steel is modeled because some of the cesium may decay in the stainless steel before being released into nature. Effective properties will be used for the wasteform. The thermal properties of stainless steel and the properties of the wasteform used to find the effective properties will be found from literature and can be seen in Figure 4.2, Figure 4.3, and Figure 4.4. The decay heat function and boundary condition for the case will be the same as the thermal model validation case. The values can be seen 3.10. The concentration is also computed in the main simulation. The cesium will also decay, making the concentration decrease over time. For this simulation, cesium 134, 136, and 137 will be simulated with half-lives of 2.06 years, 13 days, and 30.1 years, respectively. For the intial ammount of cesium, a typical waste canister contains 1.5 fuel assemblies. From a tabel in Nuclear reactor physics [18] the discharge of each isotope can be found as a Ci per tonne of heavy metal. Knowing a PWR assembly contains 0.25 tonne of heavy metal [1], the initial amount of the isotopes can be found. They are shown in Table 4.2. Since cesium is highly mobile in the environment, it is assumed to be released immediately from the boundary. Thus, the problem is assumed to be diffusion controlled, making the boundary of the canister have a constant concentration of zero. The diffusion coefficient for cesium in stainless steel is an Arrhenius equation found in the literature [12]. The diffusion coefficient for cesium in the wasteform is a linear fit to the effective diffusion coefficient, found from the two sub-simulations. The two sub-simulations simulate microstructure diffusion of a region of the wasteform. The first region is at the center of the wasteform (Sub Simulation 1), and the second region is at the edge of the wasteform (Sub Simulation 2). This can be seen in Figure 4.5. The sub simulation will be executed at the beginning of each main app time step. The microstructure of the wasteform is steady and will not change over time. Two microstructures were chosen to show the ability of modeling different microstructures in the same simulation. While

these two microstructure are not the microstructures of the wasteform (or silicon carbide), they were chosen because it is typical of a material under a large temperature gradient. To replicate anisotropic diffusion with grain boundaries, some diffusion coefficients were assumed based on the known bulk diffusion coefficient of cesium in silicon carbide and graphite. The diffusion coefficient for the sub-apps are in the form of the Arrhenius equation. Figure 4.6 shows the diffusion coefficients used versus temperature. The sub apps then use the Arrhenius equation to find the diffusion coefficient of the microstructure region. The temperature for the Arrhenius equation is supplied from the main app. Since the microstructure is small compared to the main app, there is not a notable temperature gradient over the microstructure, making it possible to use the single temperature value at the point of the region. The Asymptotic Expansion Homogenization method is then used to calculate the effective diffusion coefficient. The effective diffusion coefficient of each sub-app is then pushed back to be used in the main app.

Figure 4.7 shows the mass of cesium 134 contained over time. The dark blue dashed line shows the mass of Cesium 134 in the Wasteform (orange in Figure 3.8) and the orange dashed line shows the mass in the just the stainless steel (silver in Figure 3.8). The red line shows the mass in the waste and stainless steel (orange and silver in Figure 3.8). The mass of cesium 134 in the stainless steel, always being zero, shows that Cesium was not released from the wasteform and therefore none has been released from the canister into nature. This can also be seen in Figure 4.8, which is a plot of the mass of Cesium released over time. In this figure, the dark blue line shows the mass released from the wasteform (silver in Figure 3.8) into the canister (orange in Figure 3.8). The orange dashed line shows the mass released from the canister (orange in Figure 3.8) into nature. The mass released is found by knowing the release of barium and cesium at each time step is equal to the sum of the change in barium and cesium (ReleasePerTimeStep = $\Delta Cs + \Delta Ba$). The amount of cesium (or barium) released is equal to the percent of cesium (or barium)

in the canister times the release of both ($ReleasePerTimeStep_{Cs} = Cs/(Cs + Ba) \cdot ReleasePerTimeStep$). The total release at some time is then equal to the previous amount released plus the current time steps release ($Release_{Cs,i} = Release_{Cs,i-1} + ReleasePerTimeStep_{Cs,i}$). Again, both lines are zero, further confirming that no cesium has been released into the canister or even into nature. This is what would be expected, too. The diffusion coefficient of cesium in silicon carbide and graphite is extremely small, making diffusion extremely slow; the half-life of cesium 134 is also small, making the cesium decay quickly. The graphs are plotted in Mass versus time. The same thing can be seen for cesium 136 (Fig. 4.9, 4.10) and 137 (Fig. 4.11, 4.12).

Another analog case was developed with iodine 129, 131, and 132. This case is exactly the same, but iodine is diffusing through the silicon carbide graphite waste. Figure 4.13 shows the diffusion coefficients used versus temperature. Iodine was chosen because the diffusion coefficient is larger than cesium's diffusion coefficient in silicon carbide and graphite and iodine 129 has a very large half life. The initial amount of the isotopes can be found; they are shown in Table 4.3. For this simulation, iodine 129, 131, and 132 will be simulated with half-lives of 1.59e7 years, 8.04 days, and 2.285 hours, respectively.

The resualts were plotted the same way as the previous case and can be seen in Figures 4.14 through 4.19. It can be seen iodine 129 is released (Fg. 4.15). It can also be seen that some of iodine 131 is released from the wasteform but it is not released from the canister (Fg. 4.17). It all decays in the canister before being released.

The previous analog cases modeled different radionuclides (cesium and iodine) decaying and diffusing through a multiphase wasteform (Silicon Carbide and Graphite) in a stainless steel canister. A final case was created to model hollandite ceramic wasteform with current known properties. This case will be a worse case scenario because of two things. The first is, the release of Cesium and Barium

will not just depend on diffusion but will also depend on a voltage. Cesium or Barium will be diffusing as positive ions, therefore, to be released from the hollandite it must be balanced by something to account for the change in the charge. This case will be a worse case because it will assume the are the necessary items to balance this change in charge. The second reason is this case, similar to the analog case, will assume the cesium will be released once it reaches the boundary. The geometry and boundary conditions of the cesium are the same as the previous analog cases. Since the hollandite will have a higher waste loading the heat per canister will be higher making the temperature in hollandite higher than the temperature in borosilicate glass (this is why the thermal profile from the analog cases cannot be used). To account for this the heat from the decay of cesium 134 and 137 per gram was found. Figure 4.1 shows the heat output of the savannah river site borosilicate glass case and the heat output of a similar cesium loading case only accounting for the decay of cesium 134 and 137. The heat outputs are similar showing that only accounting for the decay of cesium 134 and 137 is a valid argument. The decay heat output can be seen in Table 4.4. In the future when other phases/radionuclides are modeled the decay heat of the other radionuclides can be accounted for producing a more accurate thermal model. The boundary of the canister was set to convection with $h = 50W/m^2 - K$ at $T_\infty = 300K$. Only the wasteform (the orange in Figure 3.8) in the simulations were changed. Therefore, only the material properties (such as specific heat, thermal conductivity, density, initial cesium concentration, and diffusion coefficient) and microstructure had to be changed. The specific heat and thermal conductivity for hollandite were found in the literature and can be seen in Figure 4.2 and Figure 4.3 [4]. The density of hollandite is $3920kg/m^3$. Knowing the initial composition of hollandite ($Cs_{1.33}Ga_{1.33}Ti_{6.67}O_{16}$) and the density, the initial concentration of cesium can be found.

$$C = \rho \frac{M_{Cs}}{M_{Hollandite}} = 374.68 \frac{kg}{m^3} \tag{4.1}$$

where M_{Cs} equals $1.33 \cdot 135.99$ and $M_{Hollandite}$ equals $1.33 \cdot 135.99$ + $1.33 \cdot 69.72$

+ $6.67 \cdot 204.3$ + $16 \cdot 16$. The amount of each cesium isotopes discharged from a reactor per ton of heavy metal is known and can be seen in Table 4.5 [18]. After the waste is discharged from a reactor, it is cooled for five years before putting it in the ceramic. The amount of cesium discharged will be different than the amount of cesium isotopes in the wasteform because of decay while cooling. The amount per tonne of heavy meatal and percent after five years can be seen in Table 4.5. The table also shows the amount initially in the wasteform, which is the concentration found earlier multiplied by the percent and the volume of the waste to get mass. For the selected composition, the initial barium concentration is 0. The diffusion coefficient for cesium and barium in hollandite were found from ionic conductivity tests. The ionic conductivity temperature dependence can be described by the Arrhenius equation [20].

$$\sigma = \sigma_0 exp\left(\frac{-E_a}{kT}\right) \tag{4.2}$$

where σ_0 is the conductivity preexponential, E_a is the activation energy in eV, k is Boltzmann's constant ($8.617 \cdot 10^{-5}\ eV/K$) and T is the absolute temperature. As previously discussed, the diffusion coefficient can be described by the Arrhenius equation (Eq. 2.19). The relationship of ionic conductivity and diffusion can be found through the Nernst-Einstein relation.

$$D_0 = \frac{RT}{z_i^2 F^2}\frac{M}{\rho}\sigma_0 \tag{4.3}$$

Where D_0 is the diffusion coefficient preexponential, R is the gas constant ($8.3145\ J/(Kmol)$), T is the absolute temperature at 298 K, z is the charge number of the ion (+1 for Cs, and +2 for Ba), F is Faraday's constant ($9.6485 \cdot 10^4\ Coulomb/mol$), M is molar mass ($1731.4\ g/mol$ for hollandite), ρ is density ($3.92\ g/cm^3$) and σ_0 is the conductivity preexponential S/cm) [20]. Conductivity tests were done on many different compositions of hollandite. As previously described, hollandite forms long rod-like features in the microstructure. Xu et al. found that as the cesium concentration increased the ring size and rod length increased. The crystal struc-

tures involve tunnels, along which the Cs would reside and provides for highly anisotropic preferred diffusion in the tunnel direction. The small grains will be dominated by grain boundary diffusion because the tunnels will be frequently interrupted, causing the tunnels to be less effective. The large grains will be dominated by bulk because the tunnels will move the ions further without being interrupted by grain boundaries. Therefore, when these rods are large, the diffusion will be dominated by bulk diffusion; when they are small, the diffusion will be dominated by the grain boundary. The high cesium concentration conductivity was then used to approximate the bulk diffusion coefficient (due to large grains) and the low cesium concentration conductivity (small grain) was used to approximate the grain boundary diffusion coefficient. The diffusion coefficient of cesium and barium in hollandite can be seen in Table 4.6 and Figure 4.20. The microstructure used will be a general 3D microstructure created in the phase field module of MOOSE. The microstructure will have the same ratio of bulk versus total area as hollandite. This ratio can be found from ρ_a/ρ_t (0.84 for this composition) where ρ_a is the actual density and ρ_t is the theoretical density.

The resualts were plotted the same way as the previous cases and can be seen in Figures 4.21 through 4.27. The temperature profile can be seen in Figure 4.30. This case also included graphs of the mass of barium. It can be seen that, similar to the silicon carbide analog cases, cesium was not released from the canister. The same simulation but without a stainless steel canister can be seen in Figures 4.31 through 4.39

Table 4.1 Thermal Properties of Analog Case

Material	Specific Heat $J/(kgK)$	Thermal Conductivity $W/(mK)$	Density kg/m^3
SiC	$670 - 1100$	$100 - 300$	3210
C	750	25	1600
S S	477	14.9	7900

Table 4.2 Cesium Discharge

Isotope	Ci per THM	Ci per Canister	g per Canister	g per m^3
Cs 134	$2.718 \cdot 10^5$	$1.02 \cdot 10^5$	78.703	108.69
Cs 136	$6.962 \cdot 10^4$	$2.61 \cdot 10^4$	0.354	0.48818
Cs 137	$1.115 \cdot 10^5$	$4.18 \cdot 10^4$	482.3	666.07

Table 4.3 Iodine Discharge

Isotope	Ci per THM	Ci per Canister	g per Canister	g per m^3
I 129	$3.2190 \cdot 10^{-2}$	$1.207 \cdot 10^{-2}$	69.259	95.646
I 131	$1.028 \cdot 10^6$	$3.855 \cdot 10^5$	3.1096	4.2943
I 132	$1.511 \cdot 10^6$	$5.666 \cdot 10^5$	0.054537	0.075315

Table 4.4 Heat Output of Final Case

Time (years)	Power (W)
0	99782.256
0.82136	137100.914
8.2136	68202.731
82.136	11421.279
273.785	136.365

Table 4.5 Cesium Discharge and Initial Mass in Hollandite

Isotope	Discharged (g/tHM)	Half-Life	After 5 years (g/tHM)	Percent (%)	Initially in Hollandite (kg)
Cs 134	271800	2.06 y	50535.216	33.711	92.204
Cs 136	69620	13 days	0	0	0
Cs 137	111500	30.1 y	99373.339	66.289	181.31

Table 4.6 Preexponential and activation energy in Hollandite

Isotope	Bulk m^2/s	GB m^2/s
Cs	$2.0637 \cdot 10^{-8} exp\left(\frac{-0.803}{kT}\right)$	$2.0637 \cdot 10^{-8} exp\left(\frac{-0.976}{kT}\right)$
Ba	$5.159 \cdot 10^{-9} exp\left(\frac{-0.803}{kT}\right)$	$5.487 \cdot 10^{-9} exp\left(\frac{-0.976}{kT}\right)$

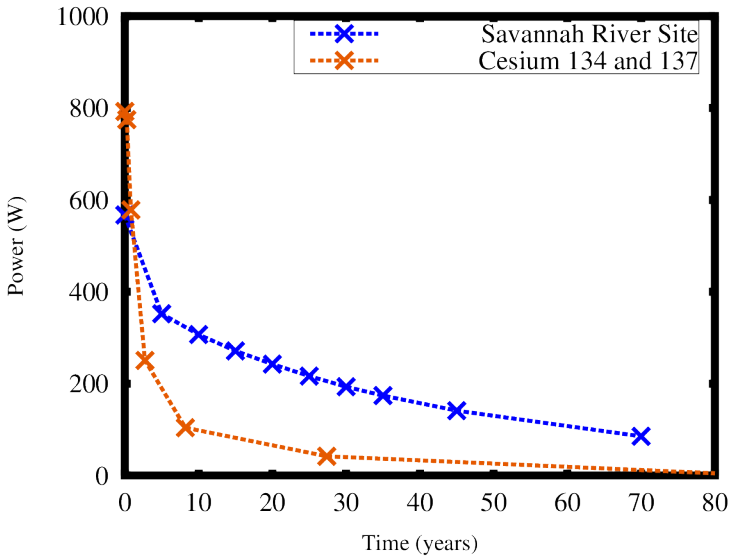

Figure 4.1 Heat Output

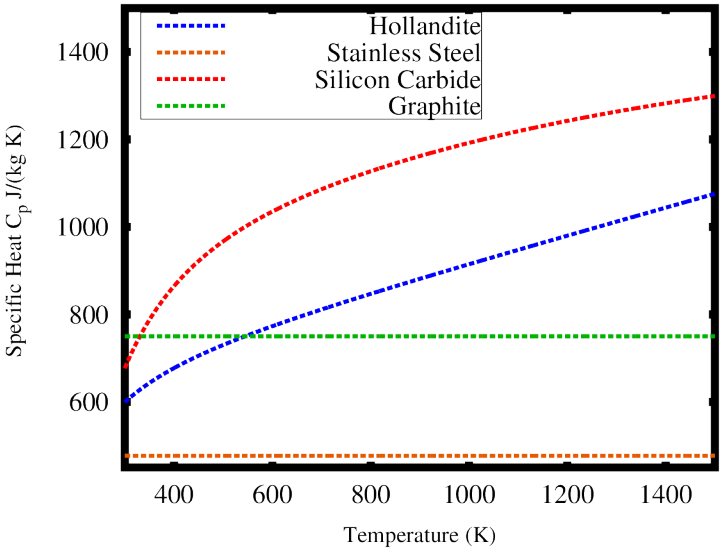

Figure 4.2 Specific Heat

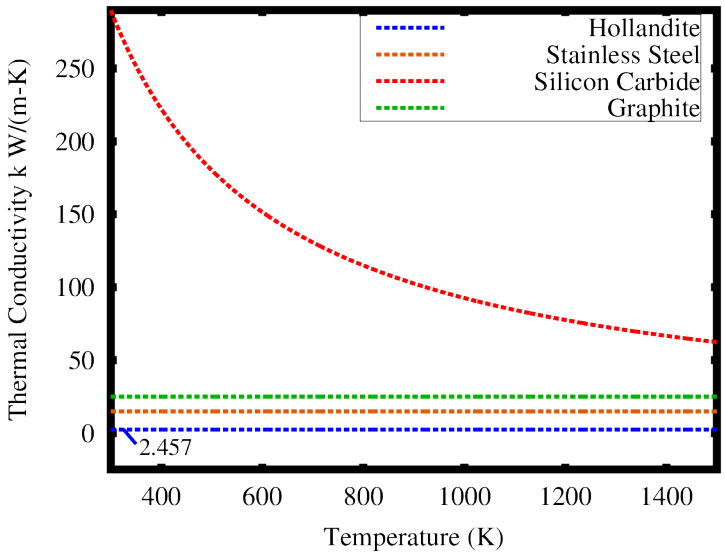

Figure 4.3 Thermal Conductivity

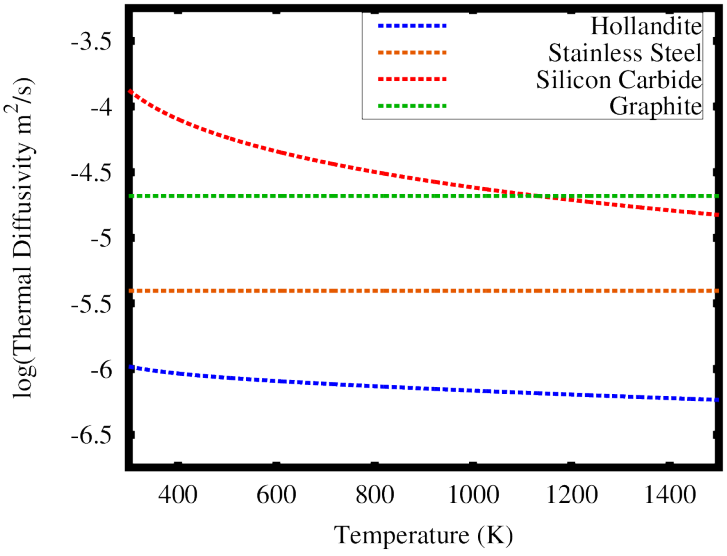

Figure 4.4 Thermal Diffusivity

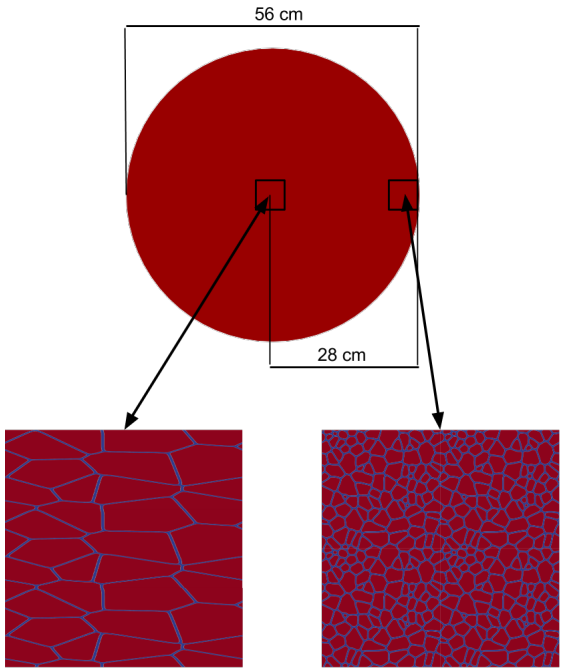

Figure 4.5 Analog Case

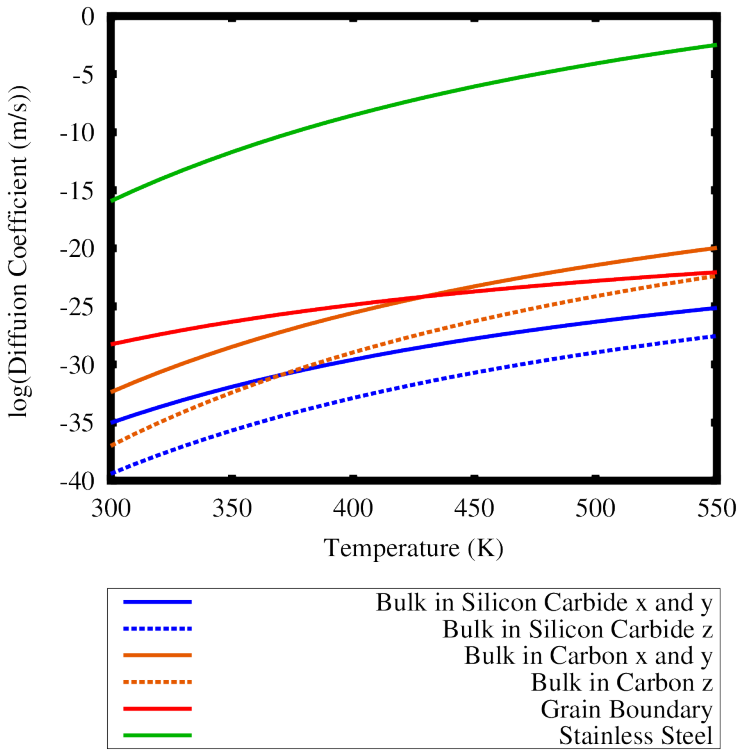

Figure 4.6 Cesium Diffusion Coefficients

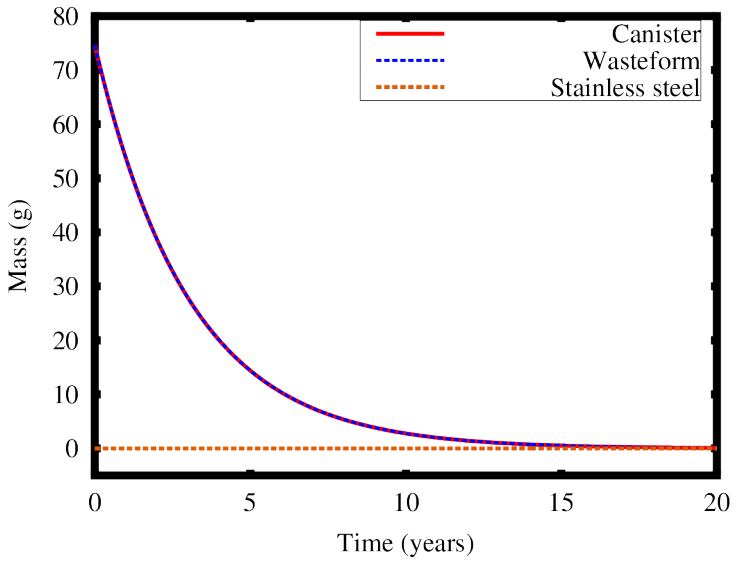

Figure 4.7 Cesium 134 Mass Contianed in SiC-C Phase

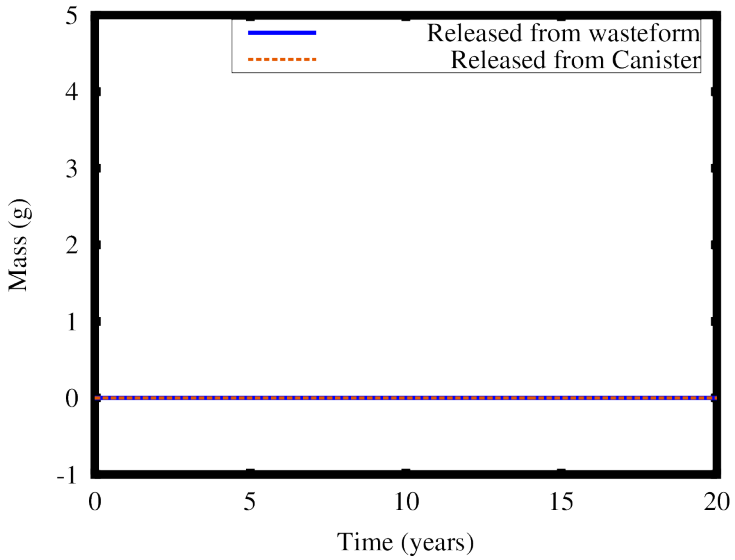

Figure 4.8 Cesium 134 Mass Released from SiC-C Phase

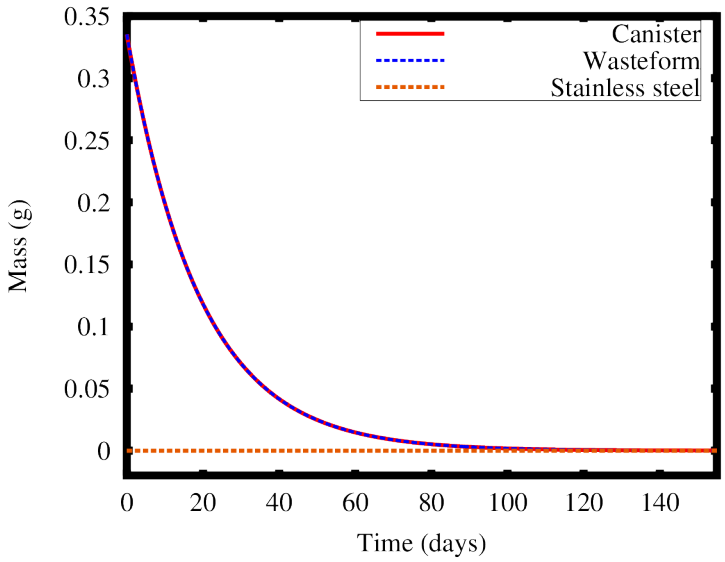

Figure 4.9 Cesium 136 Mass Contianed in SiC-C Phase

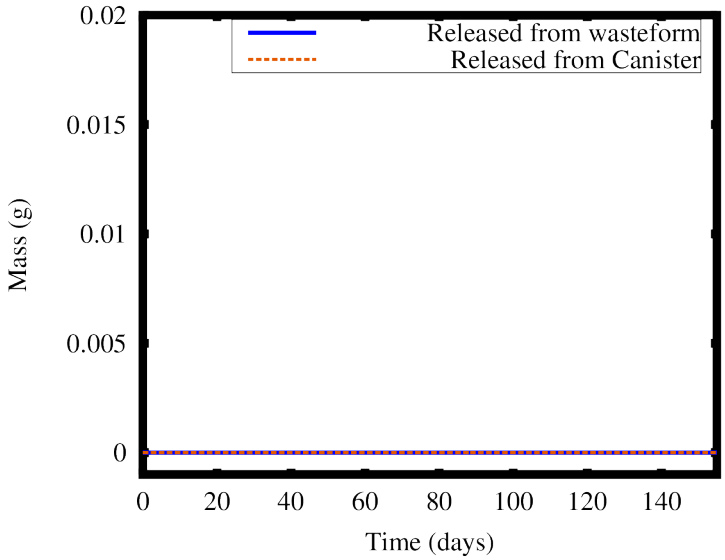

Figure 4.10 Cesium 136 Mass Released from SiC-C Phase

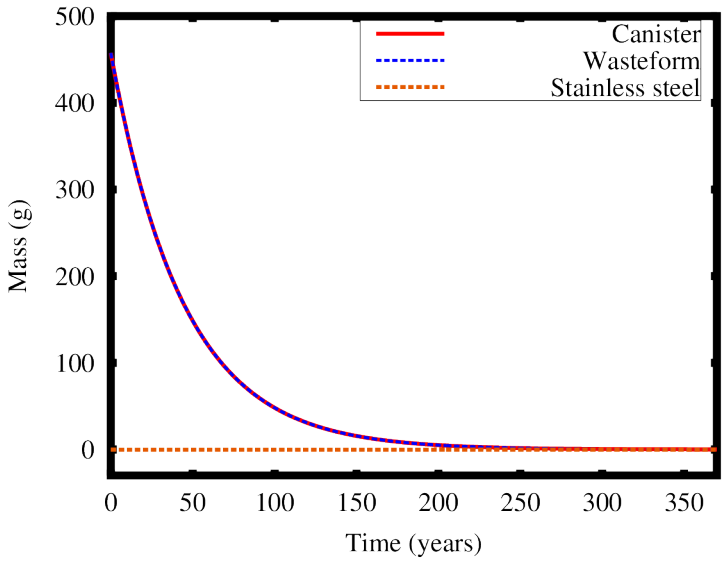

Figure 4.11 Cesium 137 Mass Contianed in SiC-C Phase

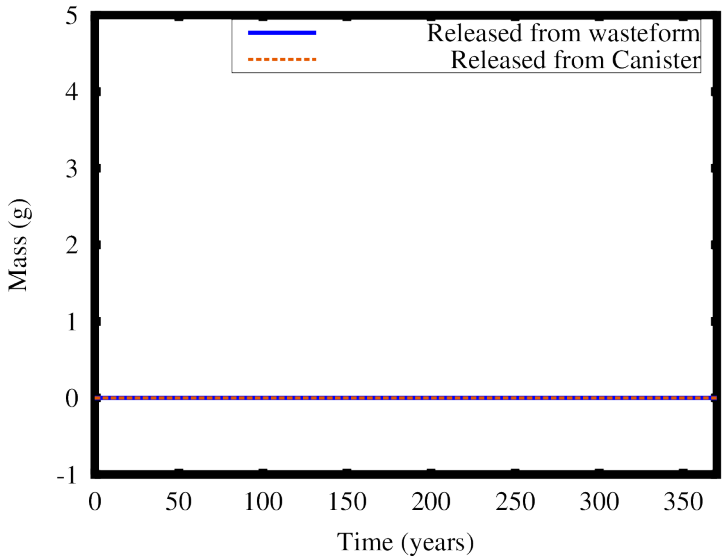

Figure 4.12 Cesium 137 Mass Released from SiC-C Phase

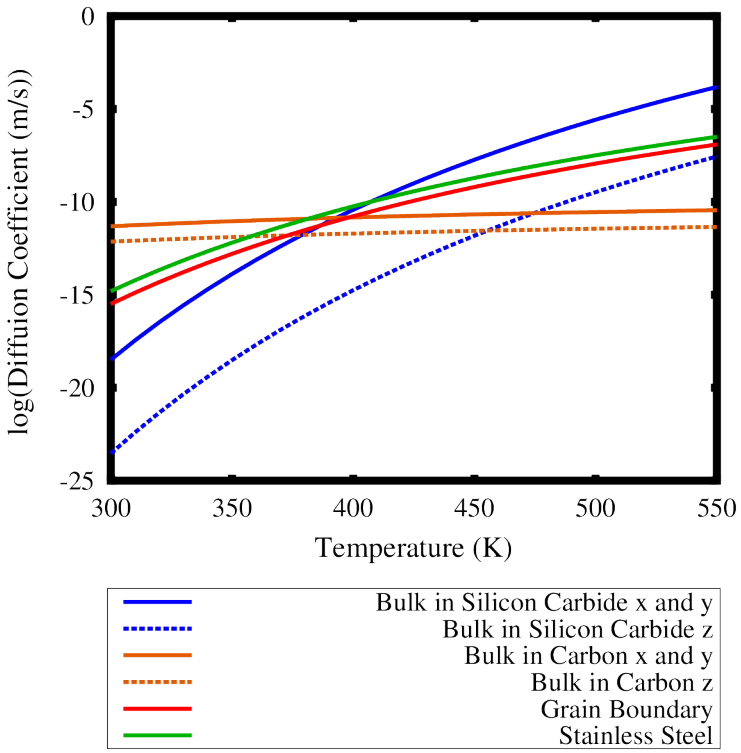

Figure 4.13 Iodine Diffusion Coefficients

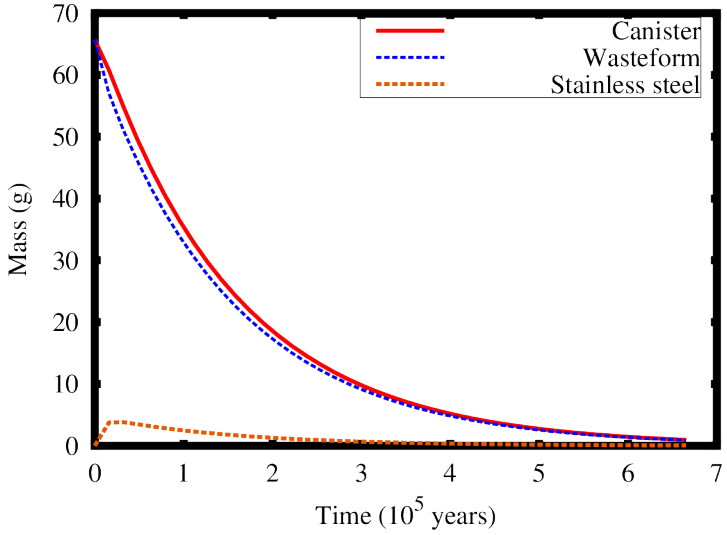

Figure 4.14 Iodine 129 Mass Contianed in SiC-C Phase

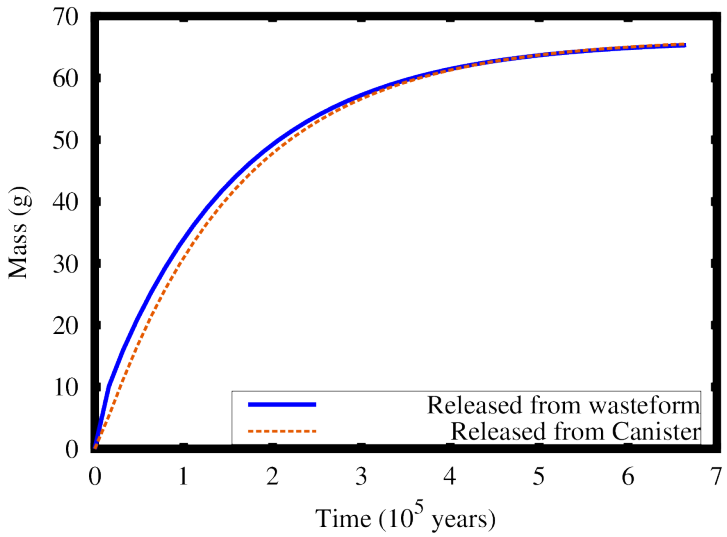

Figure 4.15 Iodine 129 Mass Released from SiC-C Phase

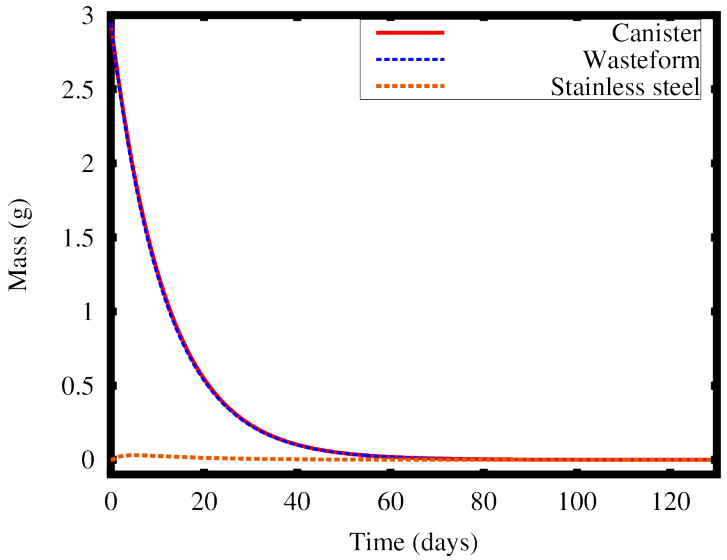

Figure 4.16 Iodine 131 Mass Contianed in SiC-C Phase

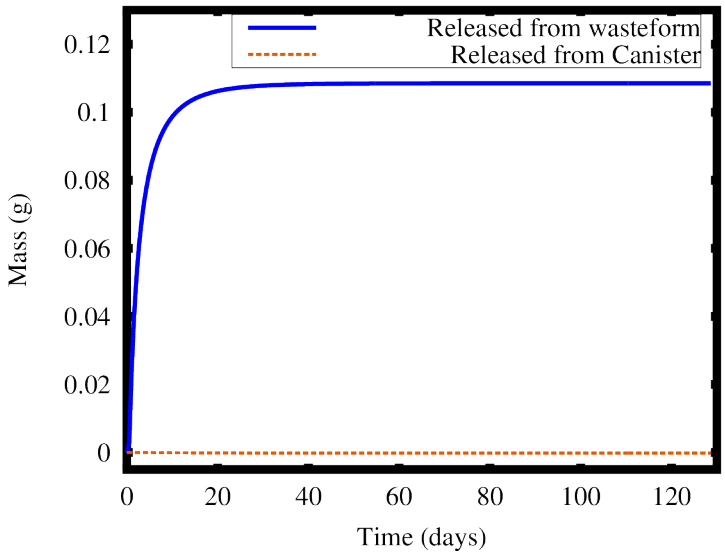

Figure 4.17 Iodine 131 Mass Released from SiC-C Phase

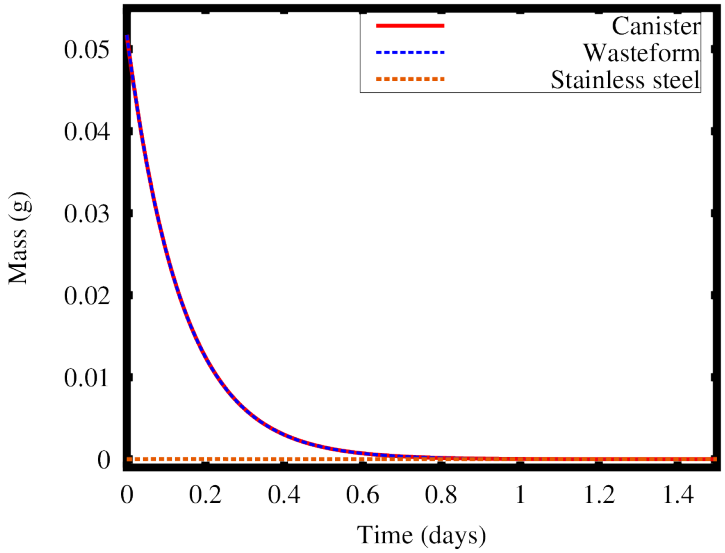

Figure 4.18 Iodine 132 Mass Contianed in SiC-C Phase

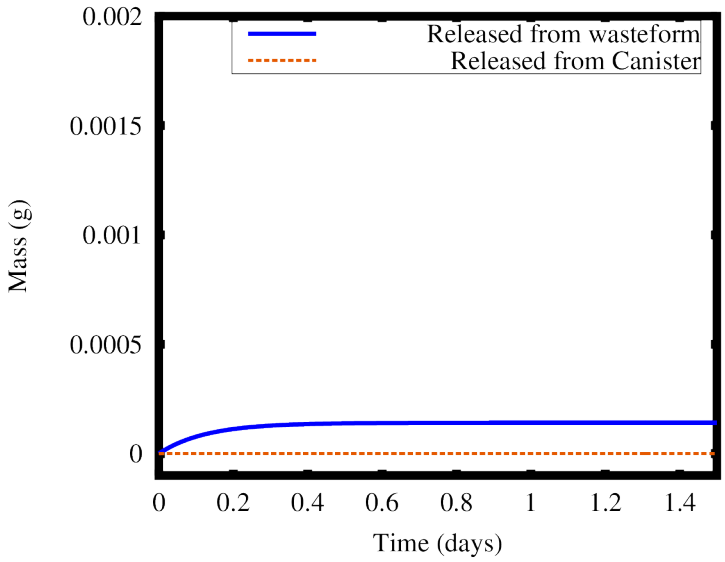

Figure 4.19 Iodine 132 Mass Released from SiC-C Phase

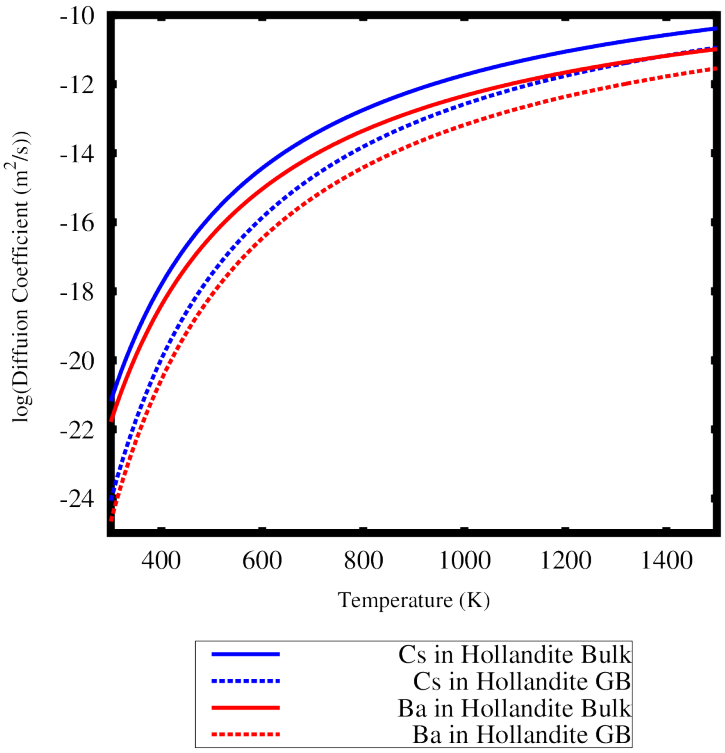

Figure 4.20 Diffusion Coefficient in Hollandite

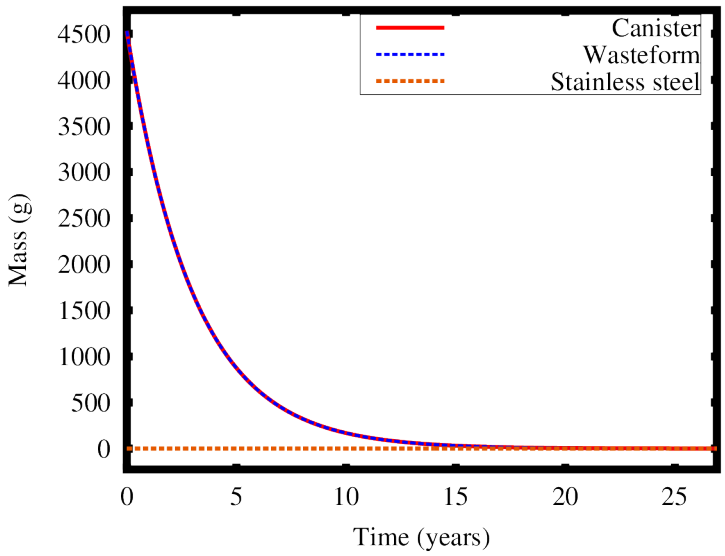

Figure 4.21 Cesium 134 Mass Contianed in Hollandite

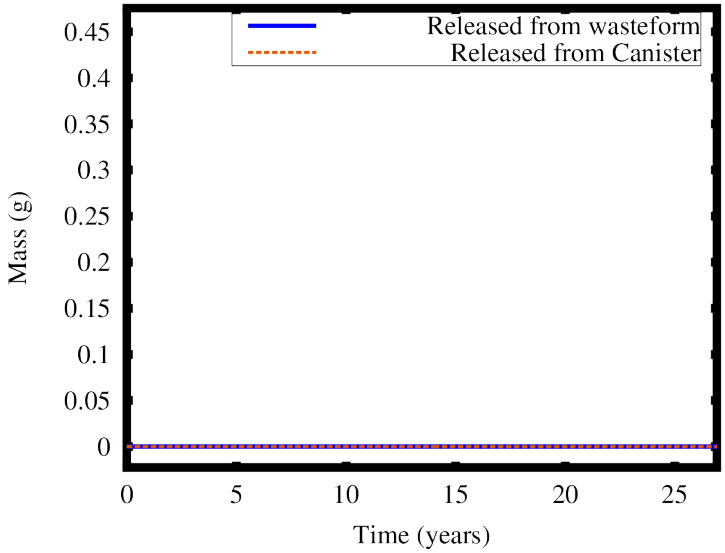

Figure 4.22 Cesium 134 Mass Released from Hollandite

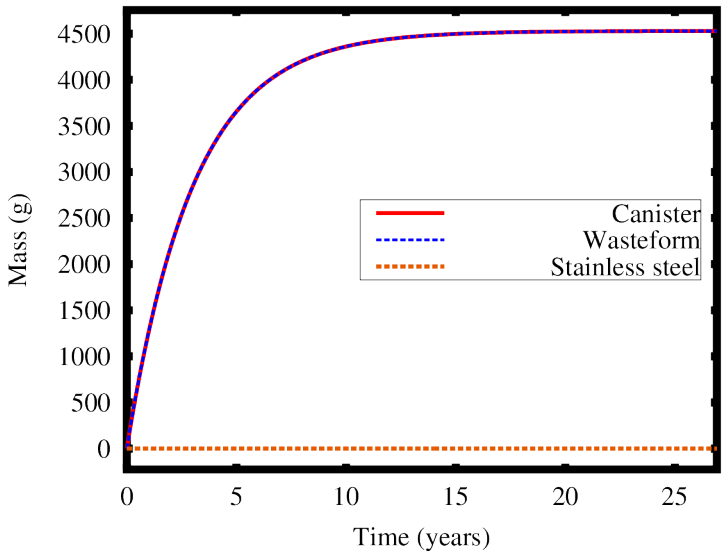

Figure 4.23 Barium 134 Mass Contianed in Hollandite

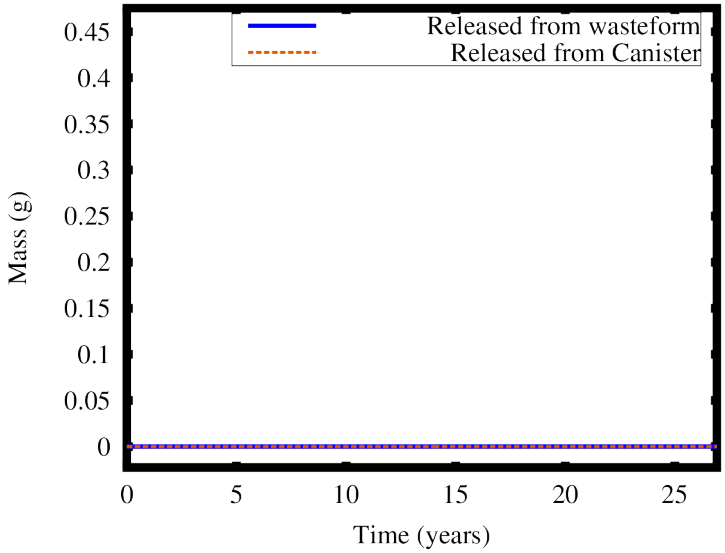

Figure 4.24 Barium 134 Mass Released from Hollandite

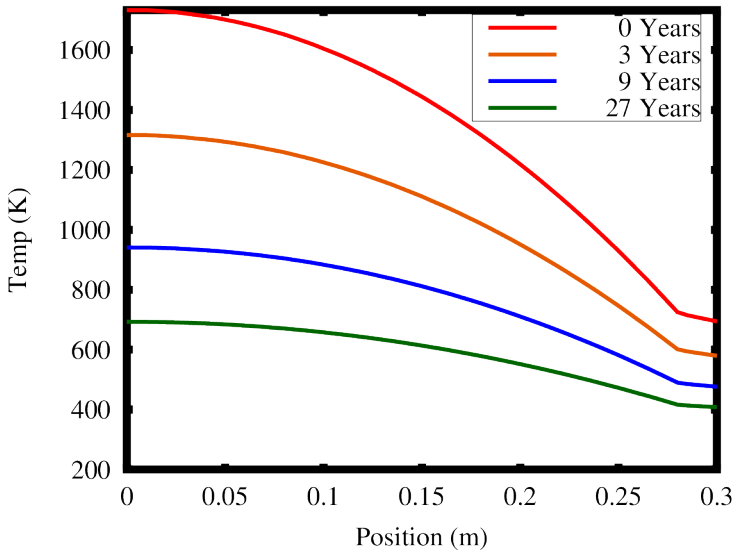

Figure 4.25 Temperature Profile of Cesium 134 Case

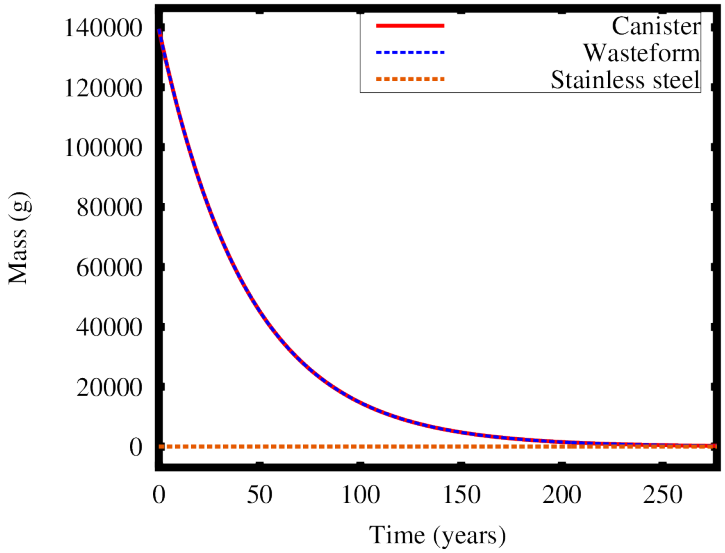

Figure 4.26 Cesium 137 Mass Contianed in Hollandite

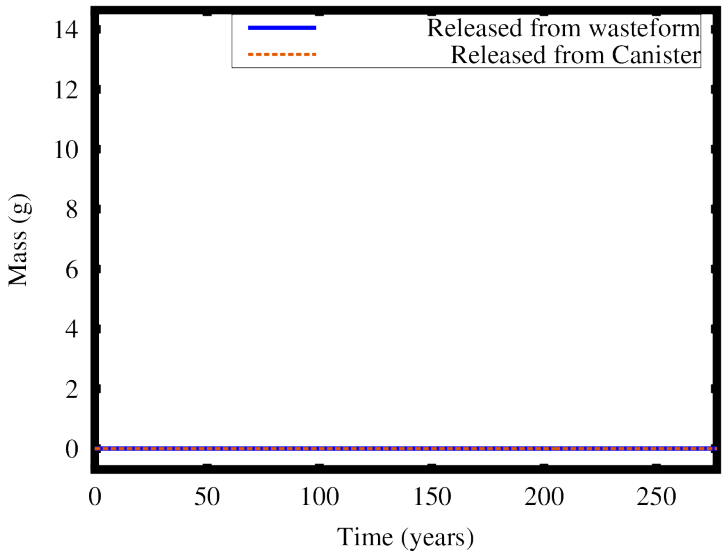

Figure 4.27 Cesium 137 Mass Released from Hollandite

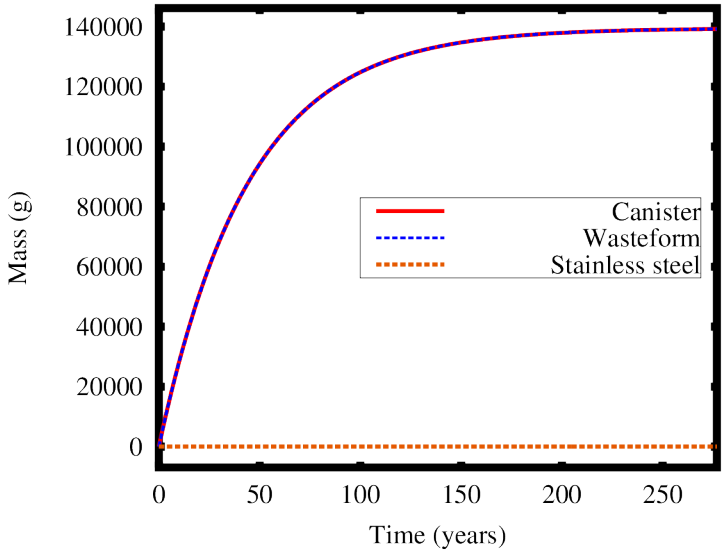

Figure 4.28 Barium 137 Mass Contianed in Hollandite

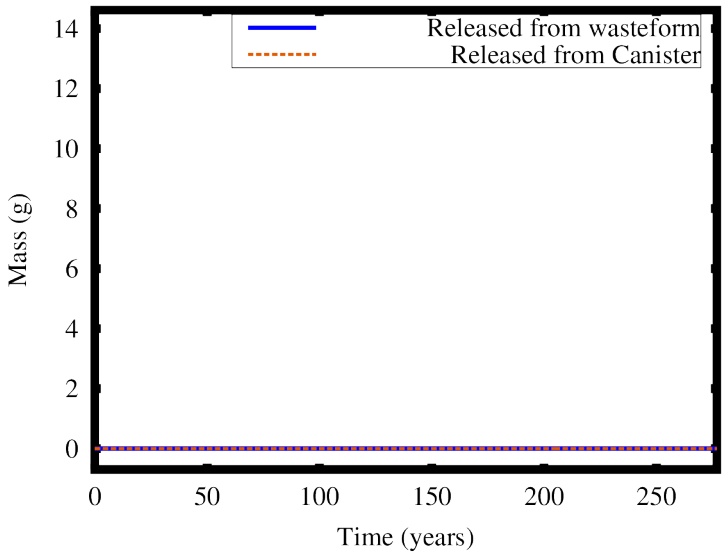

Figure 4.29 Barium 137 Mass Released from Hollandite

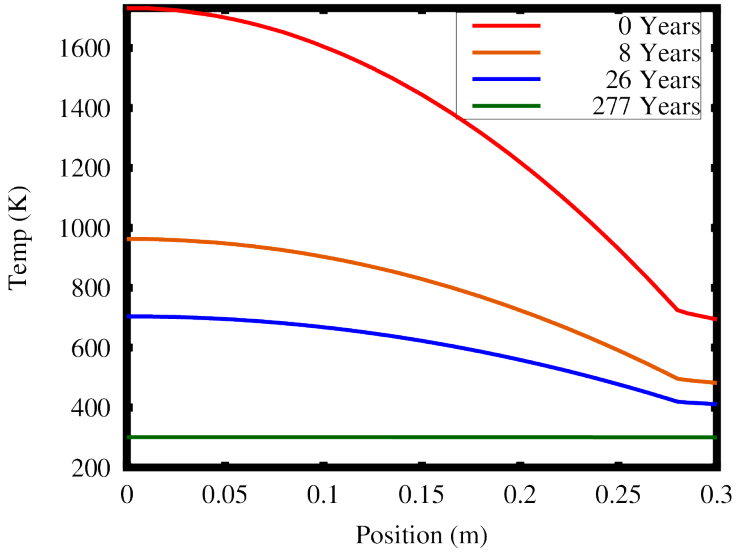

Figure 4.30 Temperature Profile of Cesium 137 Case

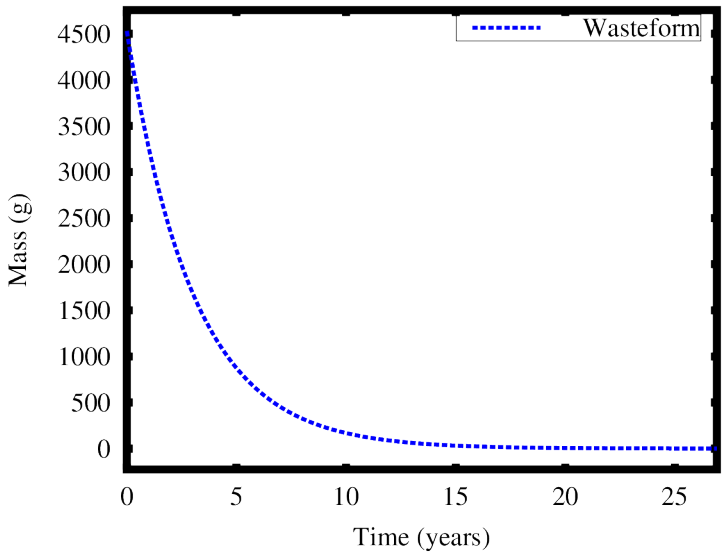

Figure 4.31 Cesium 134 Mass Contianed in Hollandite No Stainless Steel

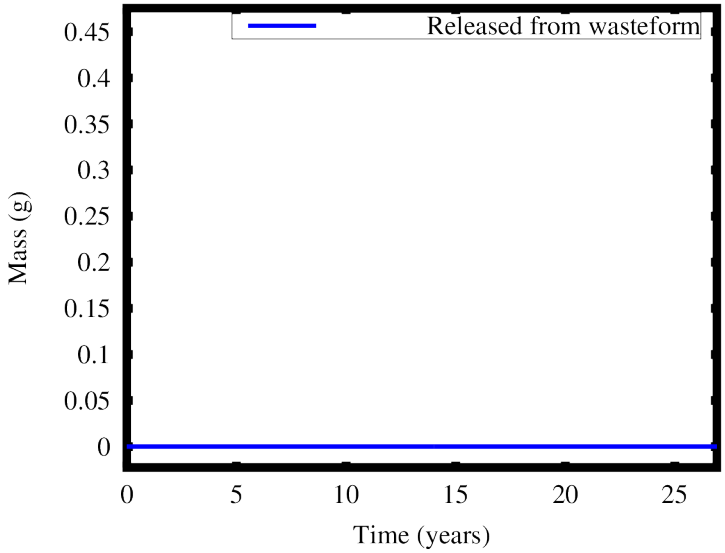

Figure 4.32 Cesium 134 Mass Released from Hollandite No Stainless Steel

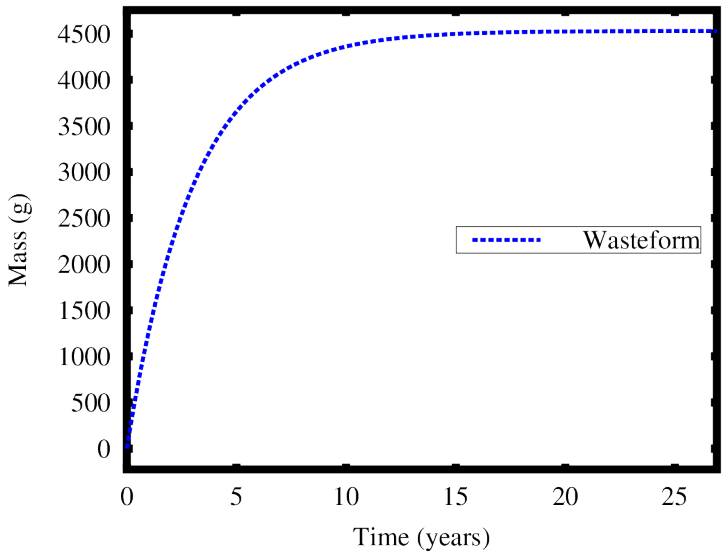

Figure 4.33 Barium 134 Mass Contianed in Hollandite No Stainless Steel

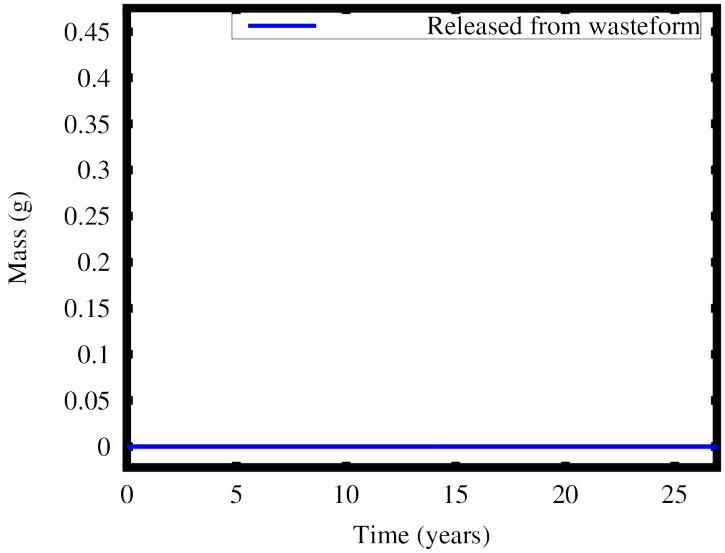

Figure 4.34 Barium 134 Mass Released from Hollandite No Stainless Steel

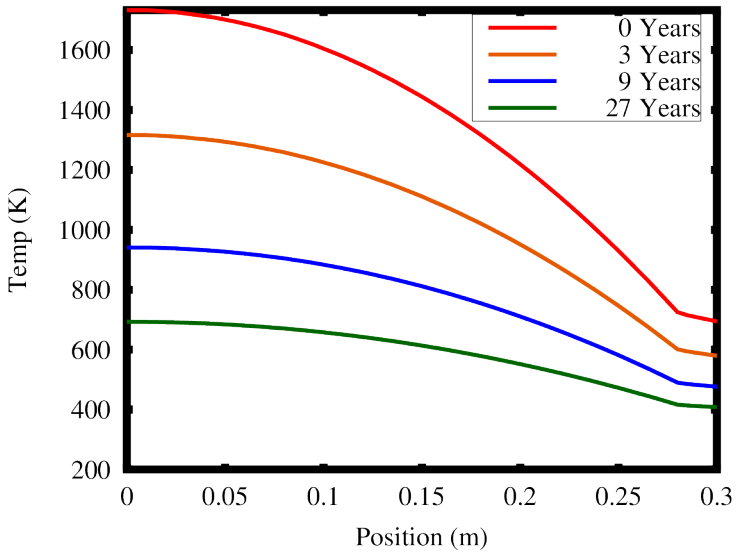

Figure 4.35 Temperature Profile of Cesium 134 No Stainless Steel Case

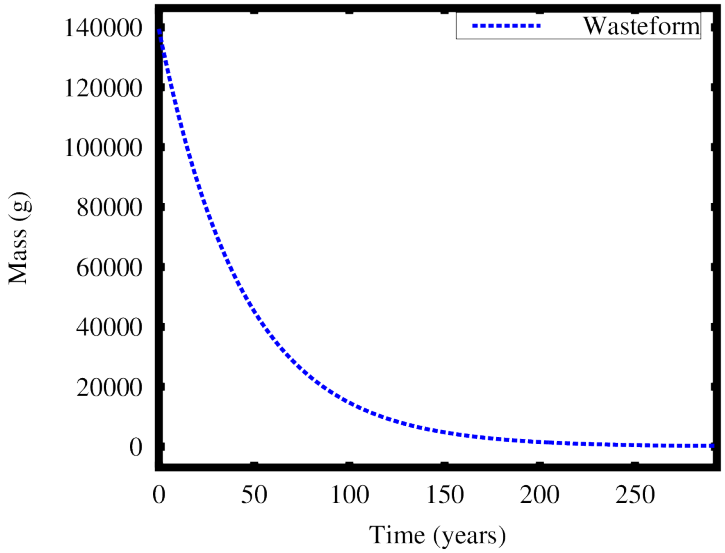

Figure 4.36 Cesium 137 Mass Contianed in Hollandite No Stainless Steel

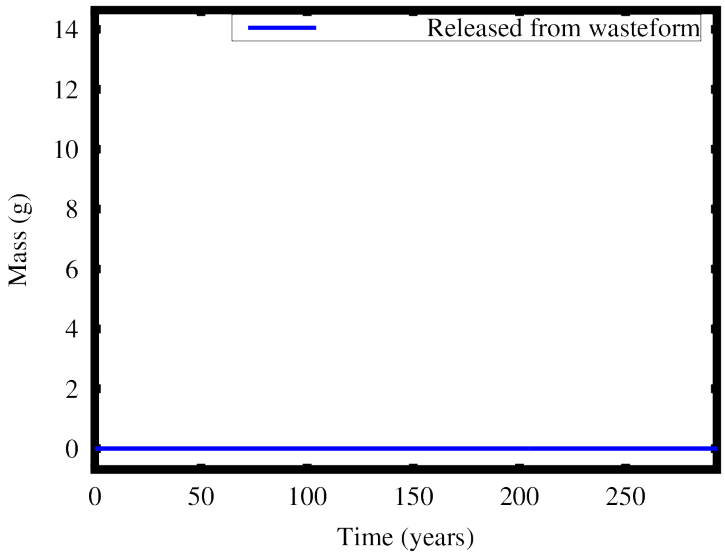

Figure 4.37 Cesium 137 Mass Released from Hollandite No Stainless Steel

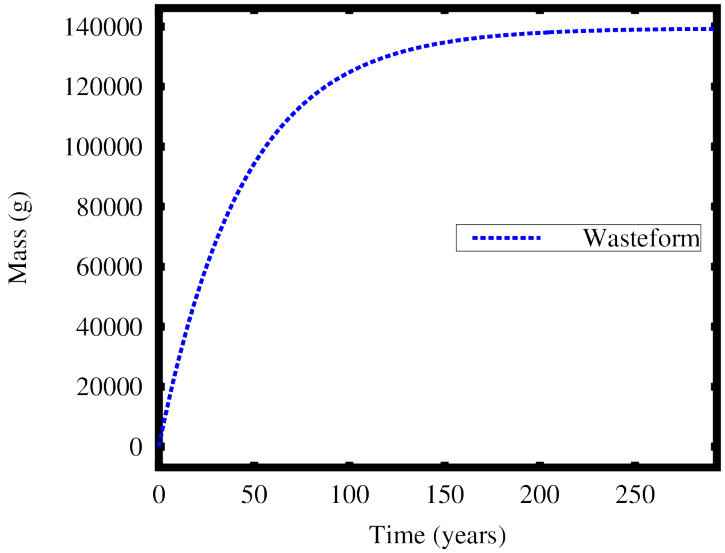

Figure 4.38 Barium 137 Mass Contianed in Hollandite No Stainless Steel

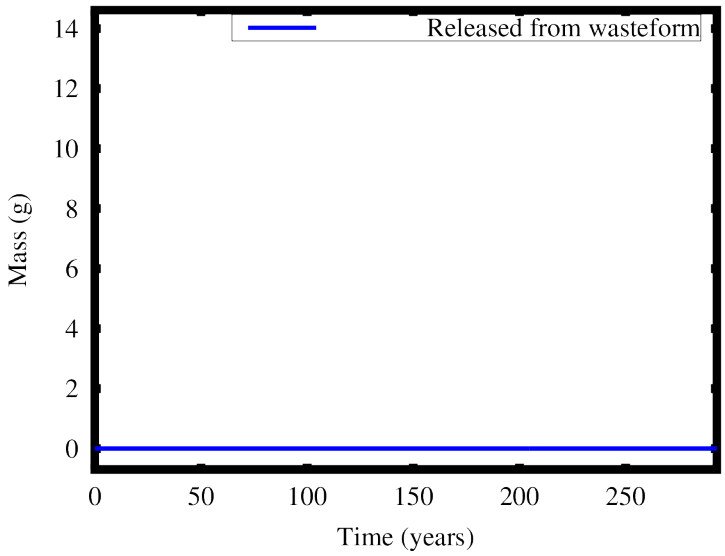

Figure 4.39 Barium 137 Mass Released from Hollandite No Stainless Steel

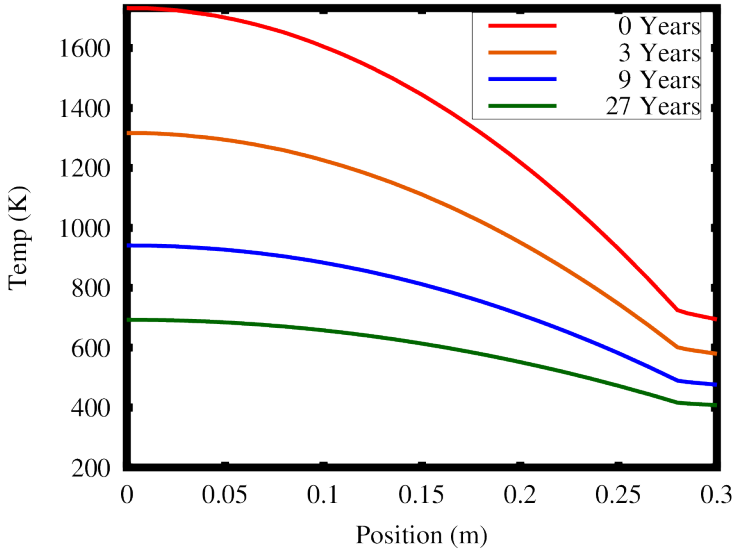

Figure 4.40 Temperature Profile of Cesium 137 No Stainless Steel Case

CHAPTER 5

CONCLUSIONS

5.1 CONCLUSION

The purpose of these analog simulations was to demonstrate the developed tools and methods for analyzing a possible wasteform. Many *kernel*, *material*, and *postprocessor* objects have been created for this. These tools can help increase the wasteforms ability to contain the radionuclides by providing useful results. By using effective properties, the developed tools are able to predict useful and accuret results. To show the final hollandite case was simulated only using the fastest path. The results can be seen in Figures 5.1 through 5.3. These results show that using only the fast path the model inaccurately predicts Cesium and Barium release. These tools and methods were then used to analyze hollandite at its current stage. With the current data, it can be seen that hollandite is a promising wasteform. As predicted, it contained the cesium and it held 400 times more cesium than the glass wasteforms.

5.2 FUTUREWORK

The tools added to MOOSE through TREX is a great launching pad for modeling nuclear waste. Many of the fundamental phenomena for modeling nuclear waste have been included or improved in TREX. While these phenomena accurately describe the physics, there are three main ways the proposed wasteform

model can be refined. The first is to add more physics to model, the second is to have more data on the wasteform, and the third is to model different scenarios. .

Different physics that could be added to refine the model are microstructure modeling, leaching, and concentration release at the boundaries. Microstructure modeling is most important for the initial microstructure of the wasteform. The alternative (or done together) is to use SEM images to recreate the microstructure (this will be discussed more). The microstructure may not change after fabrication but the benefit of modeling it is the ability to model variations. To model the microstructure, the phase field module could be used as a starting point. As of now, MOOSE can model UO_2. Secondly, leaching could be added to sharpen the model. When the waste is being stored, water could seep into the wasteform and cause leaching. Depending on the canister and wasteform, the amount of an isotope released could increase. Lastly, modeling the concentration release at the boundary could decrease the amount of an isotope released because the worse case scenario is currently being modeled. All the radionuclide is released when it reaches the boundary. An equation from different tests could be developed to find a better function for concentration release at the boundary.

More data on the wasteform that could help is SEM images and thermal properties. To model the microstructure of the wasteform from images, layered images of the microstructure would be needed. Layers would be needed to recreate the microstructure in 3D. MOOSE already has a method for using images as initial conditions. As of now, thermal properties used were from a different composition of hollandite; having thermal conductivity and specific heat could help improve the model.

The different scenarios that could be modeled are different types of canisters, canisters with imperfections and no canister (like the last case done). Since there was no cesium release in this model when there was no canister it was not important to model different canisters with imperfections. If the future refined models

release cesium when there is no canister it would be helpful to test the amount cesium released if a canister had a crack or a chip in it.

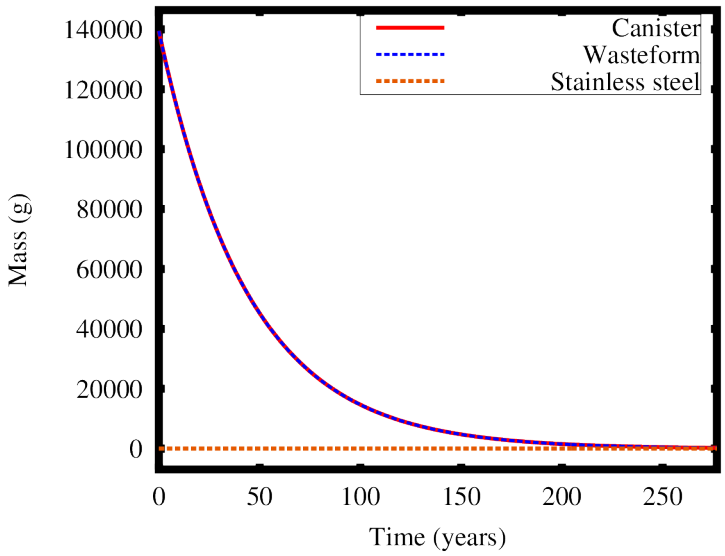

Figure 5.1 Cesium 137 Mass Contianed in Hollandite Fast Path

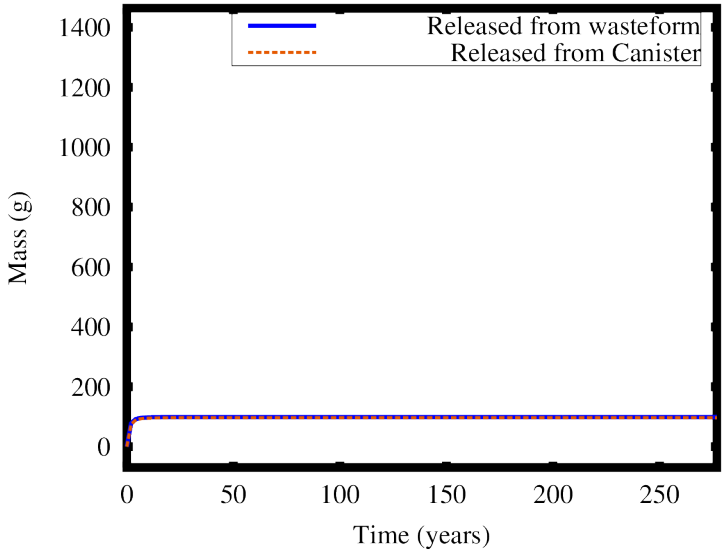

Figure 5.2 Cesium 137 Mass Released from Hollandite Fast Path

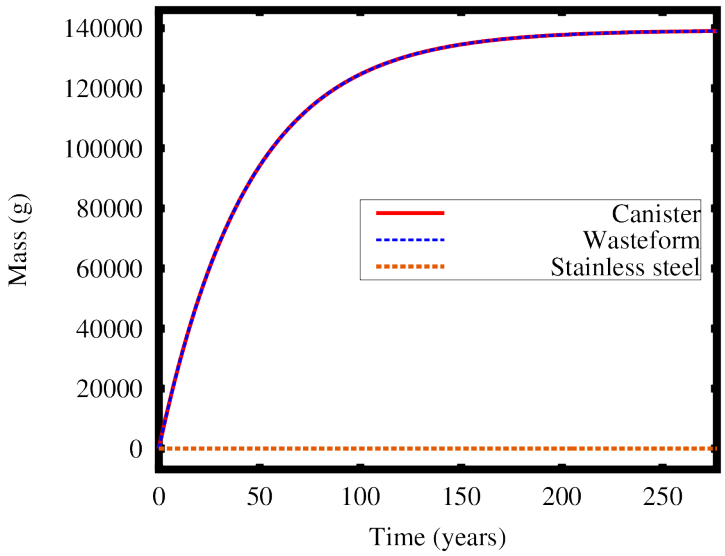

Figure 5.3 Barium 137 Mass Contianed in Hollandite Fast Path

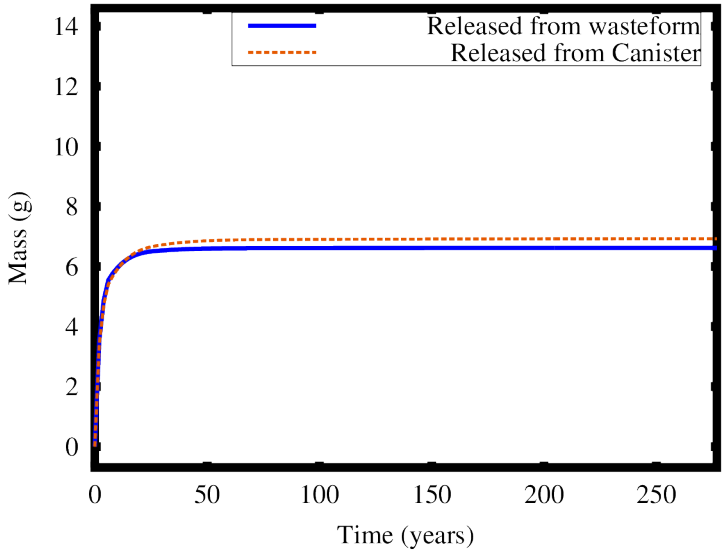

Figure 5.4 Barium 137 Mass Released from Hollandite Fast Path

BIBLIOGRAPHY

[1] World Nuclear Association. The nuclear fuel cycle. http://www.world-nuclear.org/information-library/nuclear-fuel-cycle/introduction/nuclear-fuel-cycle-overview.aspx, 2016. Accessed: 2016-08-15.

[2] Jean-Louis Auriault and Jolanta Lewandowska. Effective diffusion coefficient: from homogenization to experiment. Transport in porous media, 27(2):205–223, 1997.

[3] Theodore L Bergman, Frank P Incropera, David P DeWitt, and Adrienne S Lavine. Fundamentals of heat and mass transfer. John Wiley & Sons, 2011.

[4] WJ Buykx. Specific heat, thermal diffusivity and thermal conductivity of synroc, perovskite, zirconolite and barium hollandite. Journal of Nuclear Materials, 107(1):78–82, 1982.

[5] LM Carter, JD Brockman, JD Robertson, and SK Loyalka. Icp-ms measurement of iodine diffusion in ig-110 graphite for htgr/vhtr. Journal of Nuclear Materials, 473:218–222, 2016.

[6] Robert DeHoff. Thermodynamics in materials science. CRC Press, 2006.

[7] E Friedland, NG Van Der Berg, Johan B Malherbe, Jacobus Johannes Hancke, J Barry, E Wendler, and W Wesch. Investigation of silver and iodine transport through silicon carbide layers prepared for nuclear fuel element cladding. Journal of Nuclear Materials, 410(1):24–31, 2011.

[8] Derek Gaston, Chris Newman, Glen Hansen, and Damien Lebrun-Grandie. Moose: A parallel computational framework for coupled systems of nonlinear equations. Nuclear Engineering and Design, 239(10):1768–1778, 2009.

[9] JD Hales, MR Tonks, K Chockalingam, DM Perez, SR Novascone, BW Spencer, and RL Williamson. Asymptotic expansion homogenization for multiscale nuclear fuel analysis. Computational Materials Science, 99:290–297, 2015.

[10] Kimio Hayashi and Kousaku Fukuda. Diffusion coefficients of cesium in unirradiated graphite and comparison with those obtained from in-pile experiments. Journal of nuclear materials, 168(3):328–336, 1989.

[11] Rolf E Hummel. Understanding materials science: history, properties, applications. Springer Science & Business Media, 2004.

[12] Hj Matzke and G Linker. Study of the diffusion of cesium in stainless steel using ion beams. Journal of Nuclear Materials, 64(1-2):130–138, 1977.

[13] Michael I Ojovan and William E Lee. Glassy wasteforms for nuclear waste immobilization. Metallurgical and Materials Transactions A, 42(4):837–851, 2011.

[14] M John Plodinec, George G Wicks, and NE Bibler. Assessment of savannah river borosilicate glass in the repository environment. Technical report, Du Pont de Nemours (EI) and Co., Aiken, SC (USA). Savannah River Lab., 1982.

[15] Junuthula Narasimha Reddy. An introduction to the finite element method, volume 2. McGraw-Hill New York, 1993.

[16] AE Ringwood. Disposal of high-level nuclear wastes: a geological perspective. Mineral. Mag., 49(2):159, 1985.

[17] David Shrader, Izabela Szlufarska, and Dane Morgan. Cs diffusion in cubic silicon carbide. Journal of Nuclear Materials, 421(1):89–96, 2012.

[18] Weston M Stacey. Nuclear reactor physics. John Wiley & Sons, 2007.

[19] BS Tomar, Sumit Kumar, VK Shrikhande, and GP Kothiyal. Study of cesium diffusion in borosilicate glass by heavy ion rutherford backscattering spectrometry. Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms, 227(3):391–396, 2005.

[20] Yun Xu, Yi Wen, Rob Grote, Jake Amoroso, Lindsay Shuller Nickles, and Kyle S Brinkman. A-site compositional effects in ga-doped hollandite materials of the form baxcsyga2x+ yti8- 2x- yo16: implications for cs immobilization in crystalline ceramic waste forms. Scientific reports, 6:27412, 2016.

[21] Jingzhi Zhu, Long-Qing Chen, Jie Shen, and Veena Tikare. Microstructure dependence of diffusional transport. Computational materials science, 20(1): 37–47, 2001.