Developments in 3-D Nuclear Analysis: Model Visualization and Robust Activation Analysis

Lucas Mynsberge

January 2014

UWFDM-1420

M.S. thesis.

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

Developments in 3-D Nuclear Analysis: Model Visualization and Robust Activation Analysis

Lucas Mynsberge

Fusion Technology Institute
University of Wisconsin
1500 Engineering Drive
Madison, WI 53706

http://fti.neep.wisc.edu

January 2014

UWFDM-1420

M.S. thesis.

DEVELOPMENTS IN 3-D NUCLEAR ANALYSIS: MODEL VISUALIZATION AND ROBUST ACTIVATION ANALYSIS

by

Lucas Mynsberge

A thesis submitted in partial fulfillment of

the requirements for the degree of'

Master of Science

(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN-MADISON

2014

I Abstract

The main thrust of this research has been in designing and developing two tools to assist in performing nuclear systems analysis from modeling to induced activation. In order to do this, an extensive amount of careful code development was performed in miniscule steps to guarantee accuracy as the work traversed over various forms and types of files.

Two useful solutions are provided by the development and implementation of this work. The first tool allows an open source method of preparing a model for neutronics analysis that greatly reduces human error and provides a systematic approach to be repeated for all variations of models. Despite the intent to be used on nuclear systems, the code's versatility permits its use on fluid dynamics, stress analysis, and other mesh-based analysis operations with minimal adaptation.

The second tool reorders an already developed 3-D meshed analysis workflow in order to increase its usefulness as an activation analysis tool as well as, once again, reducing human error due to small computational mistakes. Whereas previous activation systems were limited to 1-D analysis in areas such as waste disposal rating (WDR), the effect of heterogeneity on the system can now be closely examined and 1-D predictions can be compared to the more accurate 3-D data.

In the development of these tools, a nuclear fusion power system is analyzed from its 3-D CAD geometry through a neutronics analysis and then further into an induced activation analysis. The increased capabilities provided by this work have allowed much more detailed material and MCNP tally information to be almost automated. The resulting information is then utilized to compare to the original, direct methods in order to demonstrate the improvements and also validate the developments.

Despite being used on only one nuclear fusion system, the tools presented along with the ones already in use for the study, lend themselves to a plethora of scenarios within the nuclear community. Throughout this thesis, the basis of the data types manipulated in the code and their format throughout the analysis process are deeply examined. Supplementing this at regular intervals are validation steps convincing the workflow's success. The validation steps performed have all fallen within expected ranges. There remain numerous aspects and improvements that would be necessary for production level distribution, but the developments introduced should serve the UW-Madison Fusion Technology Institute very amicably in the future, and even more so as small changes are made.

II Acknowledgements

My initial thank you goes to Dr. Laila El-Guebaly, my research advisor and nuclear analysis mentor. Since my joining of FTI in my sophomore year, I have never been pushed as hard, learned as much, or been as pleased with my job/college career. I was never shy of work and over the years I gained an immense appreciation of the work it requires to iterate through fusion power plant systems. In addition, allowing me to supplement this analysis with programming made me extremely grateful. Despite the mistakes and frustrations I created (as well as barely walking after an Ironman at work the next day), I couldn't have imagined doing anything else.

Additional thanks need to go out to Dr.'s Douglass Henderson and Timothy Tautges. Professor Henderson encouraged me throughout my college career to work hard and was always helpful or just around for a nice talk about motorcycles and athletics. Dr. Tautges helped me to realize my precarious balance between computer scientist and nuclear engineer. He forced me to find solutions myself as often as possible, which made me a better programmer and person.

Dr. Paul Wilson also deserves great thanks. From a professor in courses, to assisting me in editing computer software, despite his hectic schedule, he always seemed to be somewhat available. If I can become involved in half as many things as he is, I will be an excellent nuclear engineer. In addition, Dr. Andrew Davis received the unfortunate privilege of being directly across from my office. The incessant pestering of quick questions were probably just long enough to be distracting.

Next, I would like to thank my peers. Amir Jaber taught me almost everything I know about analysis and its frustrations. Elliot Biondo and Eric Relson also served as question sinks and I'm thankful they were around as much as they were.

I never would've gotten here without all the administrative personnel either such as Betsy Wood, Kathy Wegner, Michael Corradini, and Jake Blanchard. I've never been so relieved to have them take care of the paperwork.

One of the last thanks goes out to those directly responsible for my opportunity to perform this work: the DOE and the ARIES project. They directly funded my research appointment and have seen the need for advancing our nuclear fusion computer systems technology.

Finally, come my thanks to my family and friends who have provided continuous encouragement and support. Even when I think I'm going crazy. I'm sure I missed many, but I thank you nonetheless.

Table of Contents

Table of Figures

1 Introduction

This thesis encompasses the development of software to more completely perform complex analysis on nuclear systems. A software plugin for ParaView serves to prepare and model the geometry, material, and tally definitions for various reactor geometries in a more simplistic, user-friendly , and open-source environment. An addition to the R2S-ACT (Rigorous 2-Step ACTivation) [1] code provides the user with more flexibility in the activation data collected by the ALARA software including outputting Waste Disposal Ratings (WDR) on complex 3D geometries. Throughout the procedures, validation steps are performed in order to ensure accuracy.

Chapter 1 covers some of the guidelines and processes nuclear analysts go through to obtain data on a fusion system. It begins with the workflow currently in place that is executed. The chapter culminates with a discussion and description of the physics involved in nuclear fusion plants.

As Chapter 2 begins, the physics and code development methods behind the edited software as well as the intermediary tools are introduced. Specifically, the ITAPS/MOAB database, ParaView and the original R2S-ACT code is explained in great detail. In Chapter 3, the programming necessary to develop these new tools in conjunction with the current workflow is extensively discussed. Additionally, its future role is hypothesized.

Chapter 4 introduces a series of validation steps done to verify and confirm the success of the created tools. The old method of analysis is compared to the new method, and the new method is checked for consistency and robustness. Finally, Chapter 5 rounds out the entire nuclear analysis workflow by introducing the ARIES-ACT-1 fusion power plant study and driving it through the workflow.

1.1 Motivation

In the analysis of nuclear systems, it is paramount to obtain data that most realistically represents the system in question. Although there are other radiation worries, two of the most common are neutron irradiation and photon irradiation from neutron-induced activation. To obtain the high-level of accuracy desired for these systems, a complex 3-D geometry is a must. Previous studies have shown that when it comes to values such as damage, the difference between 3-D geometries and 1-D geometries can be up to a factor of 3 [2] depending on the accuracy of the 1-D normalization factor. The previous workflow had imperfections that made the 3-D analysis more difficult than it should be; in addition, there were multiple ways to do each step, which introduced an undesirable variability between users. The ParaView plugin suite aims to more accurately “tag” the reactor system's geometry and prepares it for neutronics analysis. From here, the reactor system enters the R2S-ACT workflow. With previous workflows, interested users could perform activation analyses of complex 3D nuclear systems, but due to the formulation of the code, information such as the WDR was not easily collected. By varying the way the code tags and interacts with the mesh, this information is easily calculated by the computer.

1.2 Neutronics and Activation Analysis

Although any neutron irradiation and activation can pose challenges, the high-fluxes and 14.1 MeV neutrons associated with fusion add a whole level of complexity to power plant studies. When contrasted against fission energy's approximate 2.5 MeV neutrons, the energy spectrum bombarding the wall and other components within a fusion reactor vessel create a much higher energy spectrum irradiation flux. This high flux benefits the reactor by

producing large amounts of heat; however, it also causes more damage to vessel components and an increased amount of activation.

Since it is not realistic to test all these properties with physical experiments, computational calculations are performed to approximate the effects of the physics in the reactor. They also serve to calculate desired data about the reactor model. The first section introduces a quick overview of the workflow performed in these analyses. The next few sections aim to define the physics and relay the significance of this high fusion flux.

1.2.1 The Nuclear Analysis Workflow

The workflow pathway starts with a reactor design and progresses through a series of pre-processing steps, analysis steps, and bridges between these. Figure 1 below provides a graphic of the workflow used for these types of analysis. In order to develop a feasible reactor model, an iterative process is performed: results from the workflow are analyzed, changes to the basic design model are made, and another iteration of the workflow occurs. This process continues until a final design is decided upon.

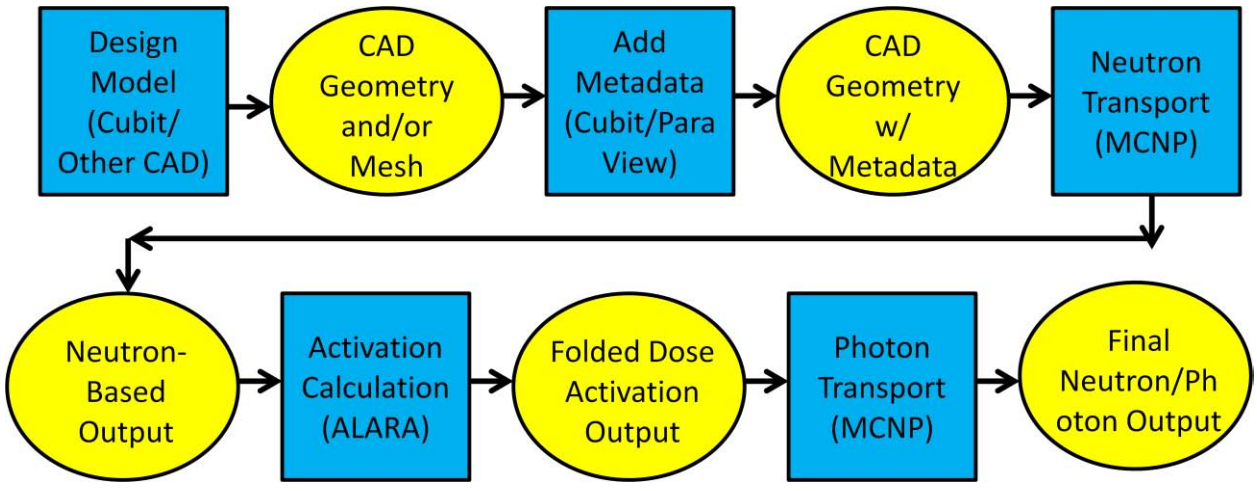


Figure 1. The workflow pathway for nuclear analysis. Blue shows programs and user manipulations and yellow shows inputs and outputs.

The first step in the process is to design the CAD model of the reactor. On the initial design, this process can take an extensive amount of time on the order of a full day or two of design. It requires building a robust model that meets all the requirements of the desired analysis outputs. From here, the model must be labeled with important metadata such as what materials compose which components, their density, and desired data outputs on various components. This step will take some time depending on the availability of densities and how many different materials the model has. The next step is to perform the neutron transport. In the workflow used here, this is a Monte Carlo transport, so it can take anywhere from an hour or two to multiple days depending on the computing power, size of the model, and statistical accuracy required. The next step takes that output and performs a deterministic activation calculation. This ranges from a few minutes to a couple of hours. Once the neutron activation data is collected it can be plugged into a photon Monte Carlo transport problem to get the final results.

Although this is the full workflow, there are many outputs along the way that are desired for analysis. Sometimes, reactor models do not even need to undergo the entire workflow. This entire process on average takes a few days to a week for the first iteration, and less thereafter. In the first iteration, much time is spent by user creating the model and preparing the programs to work properly over actually running the programs. Subsequently, the next iterations mainly require only runtime and some additional metadata work. The next sections look at some of these outputs.

1.2.2 Neutron Wall Loading, Tritium Breeding Ratio, and Nuclear Heating

Obtaining correct characterizations of neutron wall loading (NWL), tritium breeding ratio (TBR) and nuclear heating on a fusion power plant system is the first step in understanding the whole plant's operating parameters. These three neutronics effects are limited to the instant flux and are not concerned with induced activation. Results collected from these analyses affects the design parameters of entire fusion devices when heat loads are too high or TBR is too low [3].

To commence an analysis, NWL is found by recording (tallying) the neutron current that passes through a surface at the first wall (FW) in a magnetic confinement fusion vessel. The distribution of this neutron current density in the x, y, and z-axes provides important information in order to define the radial builds based on the peak inboard and outboard values. These values are normalized to the fusion power the reactor is expected to generate. Once this initial radial build is defined, more intricate complexities in the geometry can be added.

One of the most important analysis balances in the fusion community is the tritium breeding ratio (TBR) since it can neither be too high nor too low. Since most potential fusion power plants are examining the use of Deuterium-Tritium fuel, generation of tritium is required for self-sufficiency. Fusion power plants are usually estimated at anywhere from 2-3 GW of fusion power. The frequent claim for tritium estimates are about 55.6 kg/GW of tritium for a single full power year of operation (FPY).

The tritium breeding ratio is calculated based on the ratio of how many tritium atoms are produced (bred) for each tritium atom consumed in the fusion reaction. This means plants require large amounts of tritium to be self-sustaining, but they must be careful because they

can't generate too large of amounts either. Generating too much tritium would put the plant at a disadvantage because they'd have licensing issues, storage, and safety concerns. Ultimately, a net TBR value at 1.01 is preferred [4]. When performing the neutronics calculations, however, the calculated TBR is called "design TBR." This must be larger to account for known deficiencies in nuclear data and modeling and unknown uncertainties in design elements. Over the years, deficiencies have decreased, but they still are present and currently a TBR of 1.05 is preferred. The figure below is an excellent example of the deficiencies mentioned.

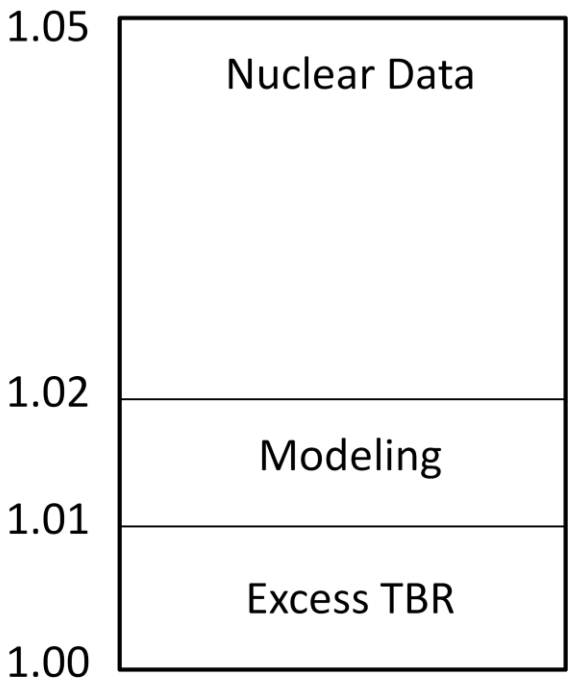


Figure 2. The effect of known and unknown discrepancies between design TBR and net TBR. [4]

The current nuclear analysis workflow has provided an indispensably close look at the effect small changes on complex 3D geometries have on TBR [5]. This narrowing towards a more exact replica of the real-world power plant helps to eliminate some of the

known deficiencies in modeling shown in figure 2. In figure 3 below, there are eight different steps. Each step represents the addition of more complexity to the geometry model of the reactor. In order to gather these results, each step requires that a model be created with the current step's geometry features, the proper material information be labeled and the proper cell tallies with relevant multipliers (response functions). This cyclic pattern through the neutronic workflow is effective, but improvements must be made to simplify the selection of tallies and materials that are used repeatedly throughout a user's analysis.

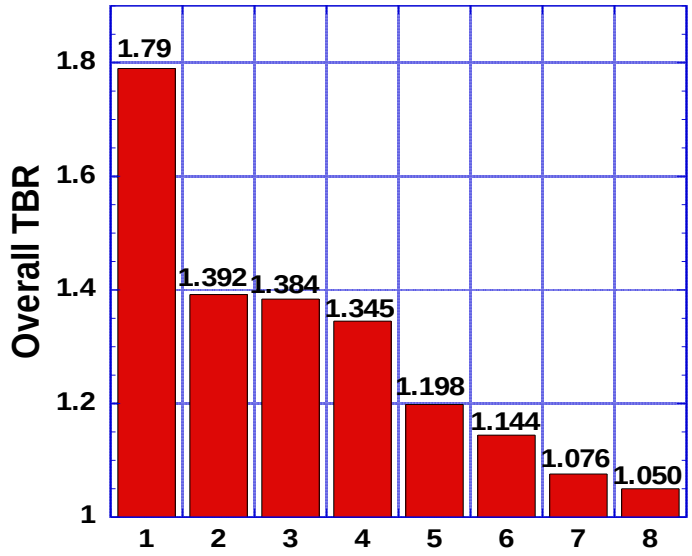


Figure 3. These eight TBR steps follow a 3D geometry as it becomes more complex

The next major neutron irradiation data gathered is nuclear heating in order to estimate the actual electricity the power plant could produce. In computer codes, cross sections for isotopes are referenced against the flux and the amount of heat produced in the reaction and then summed over the whole geometry and normalized by the density of the materials. Each component of the reactor has its heat added together and it's compared to the fusion power to see how much recovered energy is gathered. This step requires tagging

numerous materials and adding them all to a heating tally. In order to better depict heating results, a workflow tool that allowed color coding by material would increase error checking on the user's part and present a better depiction of composition for both papers and presentations.

When it comes to neutrons, there are many more values that studies desire to know for 3D geometries. Displacements per atom (a material damage estimate) and helium production both serve to dictate how often components need to be replaced in the high flux environment and whether or not it is possible to re-weld various components. These calculations are currently possible to perform, but would greatly benefit from a stream-lined workflow with visualization capabilities.

1.2.3 Specific Activity, Decay Heat, Clearance, and WDR

Neutron induced activation adds an entirely new level of complexity to already complex 3D problems. The activation is very dependent on the spatial distribution in the flux, and miniscule impurities can become large contributors to final results. These values provide very meaningful evaluation steps for studies examining the reactor during operation, maintenance periods, and shutdown.

Specific activity is very common in power plant studies purely to get an idea of radiation density. The values found for specific activity are the linking factor to the other activation results. From specific activity, computers can easily calculate waste disposal rating, recycling dose, clearance index, decay heat, and more simply through the properties of the component and material being examined.

The activity is given by equations 3 and 4,

$$A = N\lambda = -\frac{dN}{dt} \quad (1)$$

$$N = N_0 e^{-\lambda t} \quad (2)$$

where gamma is the decay constant and N is the atom number density. These equations give the activity in decays per second by understanding this rate of decay is described by the original number times some constant. Integrating finds the resultant equation for how many atoms there will be at any time, t.

In activation codes, the initial concentrations of atoms are defined by the user. The code then utilizes a flux and the reactions of that flux with the atoms to build a large matrix that keeps track of both the atoms being added to each isotope from reactions, and the atoms being taken away by decay. Equation 5 below describes this scenario. N is the vector of atom densities, and Γ is the matrix that keeps track of the time evolution of the system.

$$\frac{dN}{dt} = \Gamma N, \qquad \Gamma = \Gamma(decay) + \Gamma(reaction) \quad (3)$$

Decay heat information in a nuclear system prepares the designers to assess short term adjustments to the reactor during accident scenarios. As was learned in Fukushima [6], in the event of an accident decay heat becomes the prime focus of worry. By obtaining this information, the amount of back-up cooling can be estimated from hypothetical accident scenarios and the reactor design will be more robust. In addition to concern over accidents, decay heat provides an indication of the need for active cooling during shutdown. By ensuring this low decay heat, the downtime of the reactor is minimal and the utility would stand to profit the most. When calculating the decay heat, the specific activity is needed with high resolution in each real-world component.

Decay heat calculations multiply decay reactions by each of the various types of energy releases per disintegration for every isotope in the model. By taking the energy releases of heavy-particles (alphas, neutrons, fission fragrments), light particles (Auger electrons, positrons, betas), and EM radiation (gammas, X-rays, Bremsstrahlung) and multiplying it by the isotope concentration and decay constant, a relative heating value is found. Equations 6, 7, and 8 describe this by multiplying the decay constant times current number of those particles times the energy value for each isotope.

$$H_{HP}(t) = \sum_{i=1}^{M} \lambda_i N_i(t) E_{HP}^i \tag{4}$$

$$H_{LP}(t) = \sum_{i=1}^{M} \lambda_i N_i(t) E_{LP}^i \tag{5}$$

$$H_{EM}(t) = \sum_{i=1}^{M} \lambda_i N_i(t) E_{EM}^i \tag{6}$$

Summing these up for each isotope, and then for all three of the various types of heating, gives the total decay heat at any point, t, in time.

$$H_{Total}(t) = H_{HP}(t) + H_{LP}(t) + H_{EM}(t) \tag{7}$$

If total decay heat up to that specific time was desired, a simple integration over time would suffice.

Recycling, clearance and waste disposal rating assessments most directly motivated the alterations to the R2S-ACT workflow. In order for a fusion power plant to operate for an extended time, various components need to be replaced periodically until the end of the plant life. The time the components spend in the reactor affect their waste disposal rating (WDR), recycling dose, and clearance index at the time of replacement. This needs to be at a limit at

shutdown and then its activity can be monitored as a function of decay time. Extensive effort has been invested in understanding the radiation restrictions that will be imposed on various fusion components [7]. With limitations non-existent for some elements and seemingly beyond safe for others, collecting the data for recycling, clearance, and waste disposal rating allows potential plans to prepare for radwaste management. When finding these values, however, it is required that the full components be extracted and that the value is calculated over the fully compacted component. The goal of the R2S-ACT developments are to make gathering this information on complex, higher-order surfaces an easy task, simply handled by the computer and assessed by the researcher.

Recycling of materials used in fusion reactors is paramount to their sustainability if the community wants to minimize the radwaste stream and enhance economics. Examining the waste volume of the Z-Pinch device reveals that if the Recyclable Transmission Lines (RTL) are not recycled they will produce about 7 million m^3 of waste over 40 years [8]. When recycled, this value drops to 500 m^3 over 40 years. In order to calculate recycling ability, the recycling does rate is examined to see when it falls below various limits (see figure 4 for a look at the ARIES-CS components). The difficulties in recycling fusion reactor components are the radiation-resistant handling equipment, a large enough interim storage facility, the energy demand of recycling, recycling plant capacity, and more. These are being worked out by the fusion community in order to utilize this necessary step in the success of these various reactor designs.

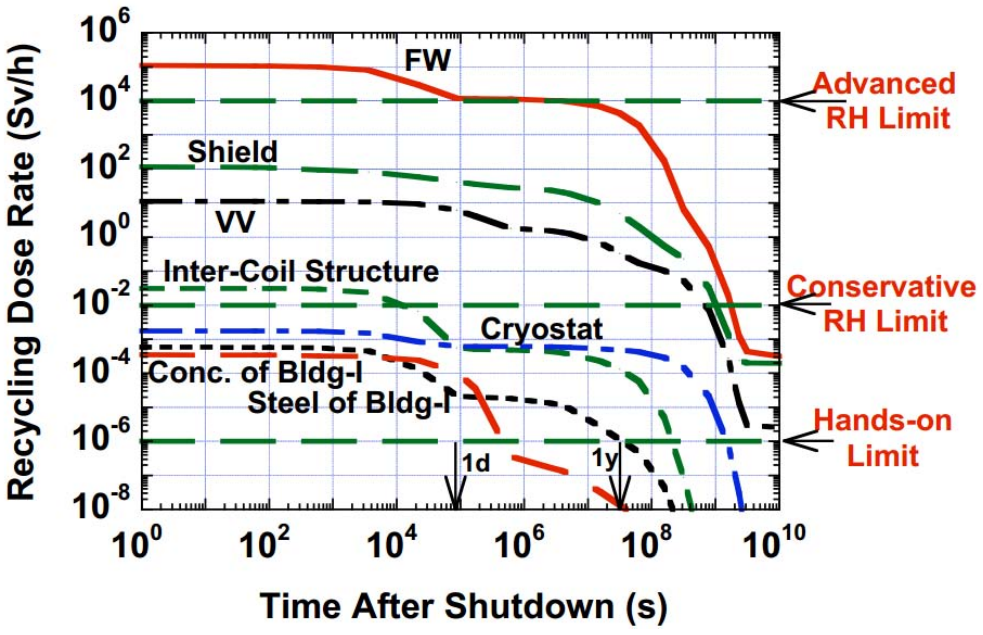


Figure 4. All ARIES-CS components could potentially be recycled in less than one year with advanced RH equipment.

For an activation code to calculate clearance, it simply needs the activity of an isotope in time. It takes that answer and compares it to a clearance level set by some governing body (such as the EU, US NRC, IAEA). Since materials are very seldom composed of one isotope, these clearances are summed up for all isotopes.

$$CI = \sum_{i}^{n} \frac{A_i}{L_i} \tag{8}$$

If the resultant value is less than or equal to one the material is cleared.

For IAEA, clearance levels follow the formula [9]

$$L_i = \min\left\{\frac{1}{E_{\gamma,i} + 0.1E_{\beta,i}}, \frac{ALI_{inh}}{1000}, \frac{ALI_{ing}}{100000}\right\} \tag{9}$$

E_γ, is the effective gamma energy in MeV, E_β is the effective beta energy in MeV, ALI_{inh} is the most restrictive annual limit for inhalation, and ALI_{ing} is the most restrictive annual limit for ingestion, both in Bq.

Neutron induced activation adds a new tool for the preparation of nuclear fusion systems. Since the equations can become extremely large and complex, they lend themselves to being solved computationally. The need for computational solutions makes it essential to have accurate nuclear data libraries for cross-sections and reaction data as well. The gamma rays produced by this activation can also be of interest to an analyst. The dose increases more when both neutrons and photons are taken into consideration.

Although only some of the desired results for power plant studies have been examined, it is clear that in order to properly design and build a fusion power plant to benefit the future, consistent, accurate, and simple workflows must be designed that allow iterations upon iterations of models until the perfect nuclear system design is found.

1.3 Tools

Since hands-on experiments are so limited for extensive nuclear analysis studies, nuclear engineers require a plethora of computational tools in order to collect the valuable information mentioned in the preceding sections. The tools used at UW-Madison are a collection of "in-house" created tools and other research institutions' tools. Starting from a CAD model, the workflow progresses through the MCNP transport code [10] , a series of Python scripts, and finally into the ALARA [11] activation code. These tools are described in more detail next. Figure 1 above mentions a few of these and where they fall in the workflow.

1.3.1 Cubit

Cubit was the geometry engine utilized and it was developed by Sandia National Laboratories [12]. It functions as a solid modeling program and mesh generation software. It

is not necessary to use Cubit in order to create the reactor system. Any solid modeling program can be used to generate the basic reactor design. Cubit does need to be used eventually, though, for very specific reasons. Two functions known as imprinting and merging allow Cubit to traverse the geometry and remove any and all redundant surfaces where there are two or more adjacent volumes. If a model is poorly designed, this step will be partially or wholly unsuccessful. If the geometry underwent radiation transport, numerous histories would be lost and the results could very well be erroneous.

In addition to simplifying the model, Cubit is used to group geometric entities in order to identify and assign materials and densities. DAG-MCNP [13] passes this information directly to the transport analysis in MCNP.

1.3.2 Direct Accelerated Geometry for Monte Carlo (DAGMC) and MCNP5

MCNP5 is a neutral particle, Monte Carlo transport code produced by Los Alamos National Laboratory. It transports neutrons, gamma rays, and even electrons, as well as performing coupled transport (secondary gamma rays caused by neutron interactions). The code can handle a multitude of problems ranging from criticality calculations to specific, defined sources. In order to obtain significant data from designed experiments, tallies are calculated ranging from cell fluxes to surface fluxes, energy deposition, and more. These tallies can also be recorded in the form of structured meshes. It is this feature that facilitates the R2S-ACT workflow, by gathering the neutron fluxes in a structured mesh, sending them through a series of Python scripts, and then passing them to ALARA.

DAG-MCNP [13] is a modified version of MCNP created by UW-Madison. This Direct Accelerated Geometry for Monte Carlo (DAG-MC) capability utilizes ray tracing on

faceted geometries, which enables complex, higher-order surfaces to be modeled in MCNP through the use of CAD-based designs. Because realistic 3D reactors often contain these complex surfaces, the code provides increased accuracy in results for a trade-off of speed. This speed is a welcome sacrifice, since it enables this much more accurate modeling; using native MCNP would require an incredible amount of hours to equivalently model these complex geometries. Fortunately, the speed of DAG-MCNP is on the average no slower than an order of magnitude lower [14] than native MCNP5. The DAG-MCNP code also has the ability to add unstructured mesh tallies to a geometry. This is very beneficial to get distribution maps of tallies of interest that aren't restricted to geometries contained within a structured mesh.

1.3.3 R2S-ACT Python Scripts

The set of R2S-ACT scripts (Robust 2-Step ACTivation) serves to couple the transport analysis of DAG-MCNP/MCNP with the activation software ALARA. Since these programs are not designed together, Python was facilitates the transfer of data from DAG-MCNP to ALARA. Python was selected due to its ease of use, versatility, and simple, straightforward syntax. In order to interact with the mesh passed from DAG-MCNP to ALARA, PyTAPS [15], a Python interface to MOAB (covered in the next chapter) is used. PyTAPS allows simple manipulation of meshes and is just as straightforward as Python.

1.3.4 Analytic and Laplacian Adaptive Radioactivity Analysis (ALARA) Code

The Analytic and Laplacian Adaptive Radioactivity Analysis (ALARA) code [11] calculates the induced activation resulting from neutron irradiation. Within the R2S-ACT workflow, it utilizes the neutron flux found from MCNP in order to calculate the photon

source density in each volume element throughout the defined mesh. In addition to MCNP's neutron flux, an irradiation schedule, the mesh geometry, and the material properties such as neutron activation cross-section and decay constants are provided--some in previously calculated libraries.

ALARA has a variety of output formats that it allows for data. For induced activation, the photon source can be output and then utilized by R2S-ACT again in MCNP. The user can also select zone, interval output, and the units of the output. For the R2S-ACT workflow changes implemented, it will be desired to have a zone output where each zone is representing one full volume from the original CAD model. The cooling steps are controlled by the user to allow the decay process to be captured statistically.

2 Development Interfaces and Environments

The Mesh Oriented datABase [16] stores and evaluates mesh data; it can handle both structured and unstructured meshes with ease. Metadata can easily be applied to the mesh. One of MOAB's prime benefits is its speed and efficiency since it processes mesh in groups rather than by accessing individual entities. Individual entity access is not eliminated, however. MOAB is accessed through a C++ interface for this thesis and, although not used here, has parallel functionality. This section delves into the fundamental types that compose MOAB and its benefit to nuclear analysis and beyond.

2.1 MOAB's Fundamental Types

The entire MOAB data model uses simply four types: mesh interfaces, mesh entities, sets, and tags. These objects are addressed via entity handles rather than pointers. This allows

easy access to the types and the values can be easily changed while the address doesn't. The MOAB interface is the gateway to all member functions that MOAB provides.

MOAB provides support for almost all imaginable mesh entities through enumeration in the interface. These range from vertices to polyhedrons. The various entities are ordered by dimension from lowest (vertex) to highest and they can be iterated over in loops. The entity handles are organized so that entities of similar type are stored together and then the individual entity id is labeled after, which allows for efficient grouping based on dimension. From a programming point of view, this reduces memory and increases speed that would be required by calling a function; rather the entity handle can just be examined by the code. In addition, this allows easy reordering and the creation of large groups that are closely linked. Since meshes frequently get large, MOAB has specific grouping that stores large sets of entity handles in a memory-efficient manner.

The vertices also form the basis for entity adjacencies. Higher order entities only have a topological relation to the lowest order (vertices) adjacency. For example, when storing a square mesh, only the four vertices composing each highest-dimension mesh element are stored as adjacency, which greatly reduces the memory requirements. Nonetheless, the faces and edges, and their adjacencies to higher or lower-dimension entities, composing the mesh element can still be found when requested by the user. As meshes drastically increase in size, especially as they do for nuclear fusion systems, the memory usage is much less than other data models.

Entity sets are groups of other entities and even entity sets. Sets can be utilized to group entities for application purposes, to describe geometric relations, or for parallel

operations. In order to relate entity sets, parent and child relationships can be applied. The analysis workflow analyzed here uses these by representing each geometric volume as an entity set and then groups these entity sets into a new set representing materials, tallies, etc. This becomes very useful in a variety of scenarios. Containing all entities and entity sets is a the "root set," which explicitly is the instance of the MOAB interface. This provides easy access to all members of any instance of a mesh. Within these entity sets, entities can be contained in a set where order is kept and thus the same handle can be repeated an arbitrary number of times, or the entity set can is ordered only by handle and duplications are not allowed.

Once entity sets are created, it is common and convenient to associate these with tags. These tags place metadata on the mesh of almost any type depending on the needs of the application. Similarly to entities and entity sets, tags are accessed via handles and thus the metadata associated with them can easily be changed without changing relationships between the tag and its interaction with the other three MOAB types. Every MOAB tag contains the following: name, size, storage type, data type, and a handle. All MOAB tags can be divided into three types: dense, sparse, or bit. Dense tags store their values in large arrays matching a series of entity handles. This makes them efficient when applying the same tag to a large group of entities. Sparse tags store a tuple of entity handle and tag value per sparse tag and are sorted by entity handles. Bit tags are very similar to dense tags, with the ability to allocate bit-size amounts to entities. If the data types of tags are known, they can be saved on MOAB files and the information is easily transported between systems. In the tools designed and

already being utilized, MOAB is used with very specific tag conventions to maximize the simplicity of passing from one analysis tool to the next.

2.2 Developing with MOAB's Interface

MOAB's use for any and all meshing situations cannot be overlooked. The interface is developed to function efficiently and quickly on numerous entities. The interface lends itself to nuclear analysis by allowing meshed 3D elements to be associated with a very specific geometry and store relationships and connectivity to the rest of the system. Important information can be tagged on these meshes to facilitate analysis in another program. The interface is designed to avoid copying and instead bringing data directly into MOAB's native format.

By implementing only the most basic relationships in large sets of mesh data and eliminating unnecessary constraints, MOAB increases its versatility and allows the user to decide connectivity and relations. Despite this simplistic approach, the underlying interface can obtain these increased relationships when it is required. Even at the lowest dimension (vertices) MOAB only connects the minimal amount of vertices to its topological entity. Thus, if a mesh has vertices at line midpoints or vertices in the middle of a mesh element, they can be ignored for mesh information. Yet with a few simple calls, these "hidden" vertices can be requested and connected or related.

Since entity lists and sets are such a big part of MOAB, there are many useful interactive tools that have been developed to save time and memory computationally. In order to find relationships between entities in a range of entities, adjacency information is

kept. This allows returning related higher- or lower-dimension entities, which can be used for further analysis or comparison.

MOAB has a built-in ability to effectively process Cubit files and preserve geometric topology and metadata associated with the solid model. As mentioned in the information about Cubit, for nuclear analysis the tallies and materials are described within Cubit in the forms of groups and names on groups. MOAB can process these files and collect the entity ids and names (such as volumes and surfaces) , groups and blocks that these entities are stored in, and mesh schemes executed within Cubit. With this stored mesh representation, the geometry can be effectively analyzed by programs such as MCNP that utilize the MOAB interface.

2.3 The Visualization ToolKit and ParaView

The Visualization Toolkit (VTK) is an open source software (designed by Kitware, Inc) that allows visualization and imaging across a variety of platforms and uses. Although based on C++, it contains many interfaces for alternate development. VTK is utilized as the basis for the development of the visualization software known as ParaView. ParaView [17] displays and operates on large sets of data using filters and other options. It can also create its own data in the VTK format. ParaView is open source as well and contains a very specific pipeline that enables developers to add features to the software. With the newest ParaView release, the developers took a more simplistic approach, allowing other developers to not only add features, but take them away or alter their functionality as well.

2.3.1 Visualizing and Manipulating with ParaView

Visualization takes extensive lists of seemingly meaningless raw data and converts it to images that are both viewable and informative. By utilizing fields, tensors, and other factors, visualization helps the design and analysis in a variety of circumstances way beyond nuclear analysis.

ParaView has the functionality to support large 2- and 3- dimensional datasets. It has the ability to run on anything from a single-processor to multi-processor supercomputers. It has proven itself as a superior visualization tool through its open-source nature, scalability, commercial support and updates, a user-friendly GUI easily accessible without any programming experience, a modular architecture, and changeability. The actual GUI is easily manipulated since ParaView's main strucutre is in the underlying libraries and not the most commonly seen client controlling the interface. Figure 5 shows a loose explanation of its architecture.

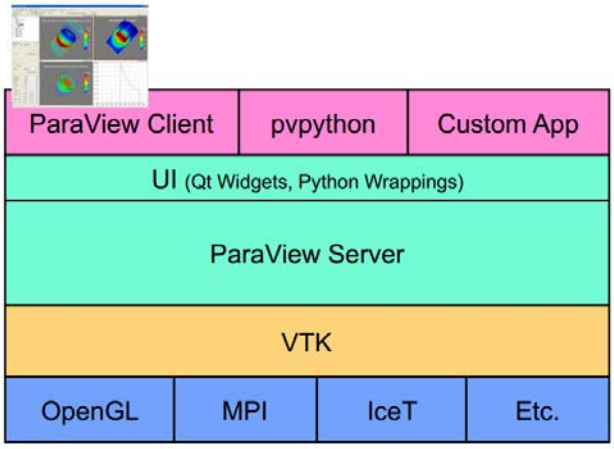


Figure 5. The ParaView client used is only a fraction of the true depth and versatility of the software. [17]

The first step in ParaView is by obtaining data to be visualized. ParaView performs this most commonly by reading in data from accepted file types--readers. It also possess the ability to generate data by building various "sources" provided by the ParaView client. This data can then be loaded into the ParaView Server and rendered. The parameters utilized in rendering are completely adjustable by the user including orientation, data representation (such as wireframe models, points, or 3D glpyhs), and field coloring proportional to the data on the mesh.

In order to maximize options and provide information and applications beyond the basic structure of the data, ParaView possesses numerous filters which operate on the current data to produce a different subset of that data. Filters can be applied to readers and sources to edit raw files. They can also be applied on other filters until the most accurate representation of the data desired by the user is obtained.

Finally, the data from VTK is rendered onto the screen. Additional data and filters can continuously be applied to create more concrete visualizations. One of the benefits ParaView provides is a pipeline allowing any or all of these to be hidden or shown at any time. In addition, multiple views can be provided in distinct windows to provide users with more information on the data. It is not limited to mesh visualizations, however; it also provides the ability to plot and graph the data according the user's wishes.

ParaView provides another very useful tool for interacting with this nuclear analysis workflow and that's selection abilities. ParaView's built-in selection features are limited, but the user is allowed to select any subset of the entire dataset. By using this selection,

information about this subset can be ascertained. As will be seen, this will be extended to allow editing and manipulation of the entire data model.

Visualization with ParaView is by no means limited to the topics discussed above. It has the features to perform animations, annotate graphs, work on large projects across multiple servers and even execute Python batch scripts. These features and the ability to add on abilities make ParaView a prime candidate to visualize nuclear fusion models and manipulate them. This will provide the nuclear engineer with a better understanding of the model and more concretely reveal the metadata surrounding the model.

2.3.2 Expandability through Plugins and Custom Applications

ParaView allows users to increase its functionality by adding readers, writers, filters, custom GUI components, and new views for data display. These plugins are accessed as shared libraries loaded by ParaView. In order to successfully use a self-developed plugin, ParaView must be a special build with shared libraries and the proper header files, the binary release will not allow it. Existing, pre-packaged plugins can be enabled by writing a server manager configuration XML file. In order to write a completely new plugin, a C++ header and program file are required to be written and compiled into a library. A server manager configuration XML file is also needed in order to connect the VTK pipeline to the C++ code.

Although not extensively analyzed, ParaView was recently restructured to take on a whole different workflow. This change has allowed the user to completely edit ParaView functionality, whereas plugins can only add features to existing behavior. By writing a custom application, pieces can be removed, operations can do something entirely different, and also features can be added. The most basic instance of writing a ParaView-brand custom

application [18] is to use the libraries and dependencies, but start with just a simple Qt-based window that then can have features slowly added. This new ParaView architecture is centralized on the idea that there are "Reactions," which responds to user actions, and "Behaviors," which allows abstract application editing related to ParaView's existence.

By utilizing ParaView's plugin abilities, features can be edited to simplify nuclear analysis workflow with relative ease. Using the plugin to import the specific geometry, and additional plugins to edit the datasets and save the data with new metadata and tagged information will prepare a complex 3-D reactor model that can immediately enter the R2S-ACT workflow.

2.3.3 The VTK File Format and Conversion to MOAB

Since ParaView is built on top of VTK, the MOAB data is pushed into the VTK library. The MOAB-based geometry file (.h5m) must be properly transferred to the VTK representation in ParaView, where it is then visualized through VTK. Part of this work had already been completed by a plugin called "vtkMoabReader." This tool converts the MOAB data file to VTK information by utilizing MOAB's tags, sets, and range functions. Figure 6 below represents the process performed by the library. The first operation is to grab the MOAB entities associated with specific tags. In this case, the reader looks at volumes, surfaces, materials, Dirichlet sets, Neumann sets, and boundary sets. Each of the entities within these sets is added to a VTK dataset. This dataset can then be visualized in ParaView directly, then.

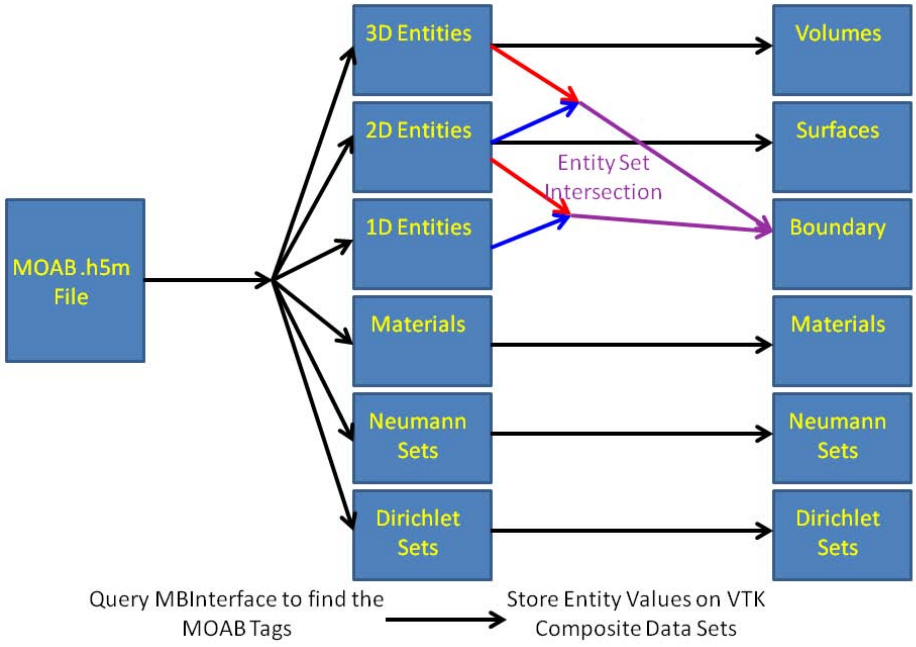


Figure 6. A representation of the process required to convert MOAB data to VTK data.

The MOAB-based geometry file is not actually composed of any 3-dimensional entities. Groups of 0-,1-, and 2-dimensional entities (points, curves, and surfaces) are put into MOAB's entity sets. These entity sets are then tagged with a tag key called "CATEGORY_TAG." The tag value of these entity sets can then be either "Volume," "Surface," Curve," "Vertex," or "Group." These are the geometric entities that the user cares about when adding metadata and that is desired to be visualized in ParaView.

2.4 R2S-ACT Workflow

The Rigorous 2-Step ACTivation (R2S-ACT) workflow provides a gateway from neutral particle transport analysis to activation analysis. This connection was originally desired to determine photon biological dose rate resulting from induced activation on complex geometries. The workflow utilizes MCNP and ALARA along with the a Python script suite in order to facilitate the workflow. Figure 7 shows a graphic of the workflow. The

red triangles represent data sinks (anywhere results can be obtained), the purple represents the actual R2S portion executed with Python scripts, and the blue squares show external software.

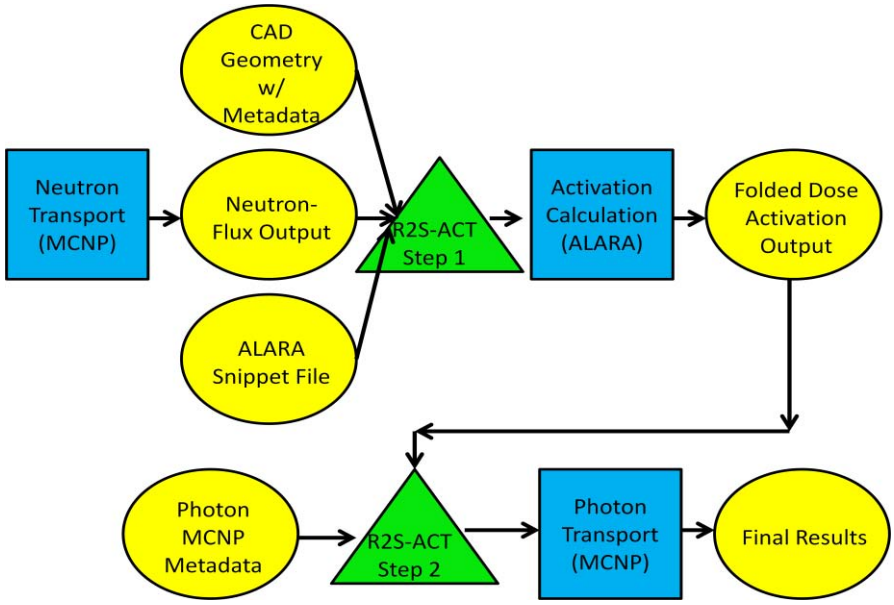


Figure 7. 3-D activation workflow depicting the processes performed for full analysis. Data can be collected after any step.

The starting point in the workflow requires the CAD geometry for the nuclear system to be properly created and the MCNP input to be prepared. The MCNP file needs to contain the neutron source definition, any desired tallies, material information, and most importantly MCNP's mesh tally feature. This is split up into a 175 neutron group energy structure. The material identification number and density also must be tagged onto the geometry of the CAD model by utilizing Cubit's groups. Additional tallies can also be added by placing them on the Cubit model. This collection of information can then undergo the neutron transport analysis.

Next is the first step of the Python scripts. This step takes the neutron flux mesh file that MCNP produces and overlays it on the geometry file provided. After this, ray-firing is performed on the combined geometry/mesh at each voxel in the mesh. This ray-firing determines the fractions of materials in each cell. Using Python dictionary storage features, these fractions are stored for each voxel. The scripts then utilize PyTAPs to obtain any material tags from the geometry that are within those voxels. By using these fractions of materials and the density values on the tags, a homogeneous mixture for each voxel is created and added to an ALARA input file along with the size of the voxel for that mixture. In addition, each material taken from the Cubit geometry is added in ALARA's material definition stage. The script then writes the flux file for each group and voxel by reading in the meshed neutron flux file and listing it in the order matching the voxels. In R2S-ACT, the voxels are listed in 'zyx' order along axes.

At this point, the Python scripts have created an ALARA input file that is ready to undergo analysis. Using ALARA to run the program produces photon source information at a specific cooling time step after irradiation as decided by the user. This source information at each point in the mesh is then sent to step two of the R2S-ACT workflow.

The second step of the Python script uses the same MCNP input as before, but with an altered source definition. It eliminates the original source definition and writes one based on the calculated photon source in each voxel. When running MCNP for the final time, these photon sources will be generated in the corresponding area of the reactor model and any desired tally will be calculated. It is important to remember that this photon response is just

for a single time step in the reactor's cooling process. By repeating the analysis at various time steps, a predictive model can be used to demonstrate the effect of activation over time.

The R2S-ACT workflow has been used multiple times within UW-Madison's FTI research group and is constantly being improved to grow with the needs of nuclear analysis. One of the current projects is to enable the workflow to successfully work on unstructured meshes in addition to structured meshes. It is one more tool that assists a nuclear analyst in defining and modifying a nuclear fusion system. The developments that will be demonstrated aim to increase robustness and user options, so it can be applied to a wider range of nuclear applications.

3 Expanding the Workflow's Abilities

As technology improves and nuclear analysis attempts to obtain every piece of information about a nuclear system, the analysis software used needs to progress and change with it. Especially in the case of nuclear fusion plants, collecting as much activation data about a system as possible helps engineers and designers to predict the exact responses of a reactor during accidents and after shutdown, cost, shielding requirements, and more. In order to assist in the development and increased accuracy of data manipulation, two improvements are made to the nuclear analysis workflow.

The first improvement utilizes visualization software in order to obtain a depiction of the nuclear system modeled in great detail. The user is then able to select subsets of the reactor system, whether that be various volumes, surfaces, materials, or tallies and either add metadata information, remove it, or edit it to prepare the model for analysis. The

development is intended to add metadata information that can be used by MCNP to perform neutron transport, but the underlying developments require only minor tweaks to be applied to fluid dynamics, thermodynamics, and other analysis. One of the most important parts of this change in the workflow is that the metadata is now recorded in the same "language" (MOAB) that is used by the analysis; this provides simplification.

The second development takes the R2S-ACT workflow discussed above and introduces an alternate method of collecting the mesh information. It iterates through the mesh geometry and extracts Cubit volume information rather than material information. It then uses this to calculate volumes in specific zones. This new method allows all ALARA responses to be accurately obtained in a real system.

3.1 Utilizing Visualization in Model Preparation

Computer visualization of geometric models of nuclear fusion systems transforms seemingly irrelevant streams of data into understandable depictions of reactor responses. ParaView has been used extensively to demonstrate the effect of nuclear heating and tritium breeding ratio on the nuclear fusion systems studied at UW-Madison. This section covers the development of a plugin to increase ParaView's application ability by improving tools for selecting, reporting, and modifying metadata of the model.

3.1.1 ParaView's Improvements on Cubit

Previously, Cubit has been solely utilized to prepare complex 3D geometries for MCNP by tagging on material and tally information by assigning geometric entities to "groups" and those groups are given names defining the material or tally information. These groups are read into MOAB and altered and extracted to become MOAB tags on the

geometry. The inherent clumsiness of this method arises because of human error and inaccuracy with the MOAB-based analysis. When it comes to human error, the group names are required to be in some format such as "mat_#_rho_{density value}" or "tally_#_{tally type}." The difficulty with this is misspellings will render the group just ignored when passed to DAGMC. Also, the format has changed occasionally with new releases of DAGMC, which again risks important metadata being ignored. Computationally, the analysis code must parse these strings and extract the significant data. Finally, MOAB tags are intended to have single key-value pairs such as {key: MATERIAL_TAG, value: Steel} and then another tag key-value pair would provide the density such as {key: DENSITY_TAG, value: 0.0444} and not "mat_1_rho_0.0444," where "1" is later associated with steel. Figure 8 demonstrates the non-natural method of Cubit and compares it to MOAB.

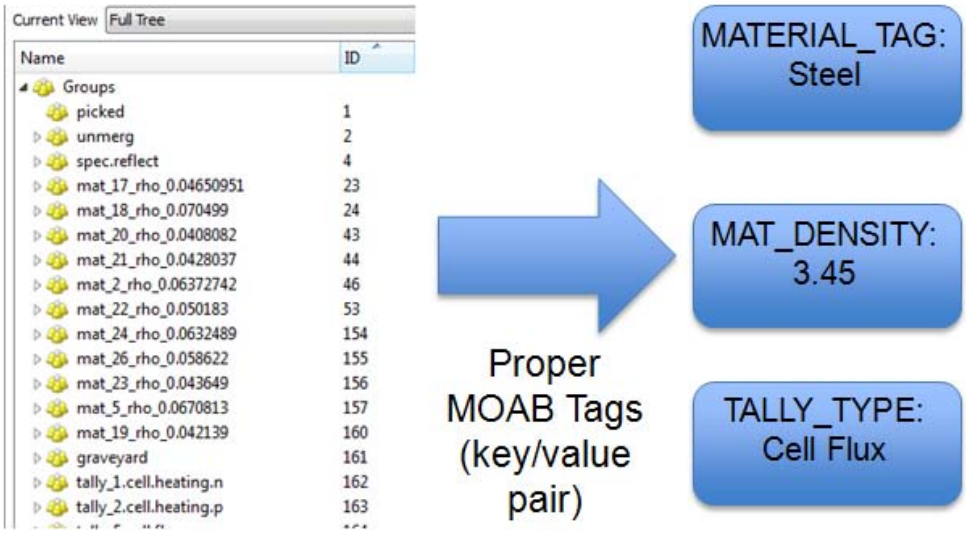


Figure 8. On the left, Cubit uses groups and names on groups. On the right, MOAB assigns values for the relevant keys.

ParaView can directly add tags to the MOAB geometry with the new plugin. In addition, Cubit visualizes geometric models in a different language than ParaView, which

increases discrepancies and translation errors. With the metadata labeling, ParaView will enable the depiction of materials or tallies by gradient color scales; this feature serves as an additional check of accuracy to ensure the right topology belongs to the right materials and tallies. Another benefit of developing this visualization plugin is that it removes sources of human error such as spelling errors that would render the tag unreadable, and speed of processing. Ultimately, ParaView will hopefully become the hub for complex 3D geometry analysis--serving to prepare the model for analysis and reviewing the results of the analysis on top of the geometry once completed.

3.1.2 Developing a user-friendly GUI

A good or bad program can come down to its Graphical User Interface (GUI). The GUI for the ParaView development utilized Qt to create and develop what are known as "widgets." When researching and brainstorming this plugin, careful consideration was taken to develop a GUI with three main components: simplicity, robustness, and expandability.

In making it simple, the GUI was developed to have the fewest buttons and options available that still allow for as much manipulation as the data as desired. Although in the current workflow, it is used for specific metadata, the goal was to design it with enough versatility to handle other engineering workflows and analyses. This was achieved by limiting too much specificity. Figure 9 shows a view of the plugin. The first box allows the user to select what "picking" mode to be in. Since it is possible for a point to belong to multiple geometric values (i.e. a curve, a surface, and a volume), this tells the underlying algorithms which one the user wants. Below that box is the tree hierarchy. This tree shows the different selectable entity sets and the geometric value that they are defined as. Below

that is the name of the current MOAB tag keys on the model in a list, as well as an option to add a new tag. Values for the desired tag key are entered in an editable space and finally the options to apply--add the tag value--delete, and write the MOAB file to an output round out the GUI.

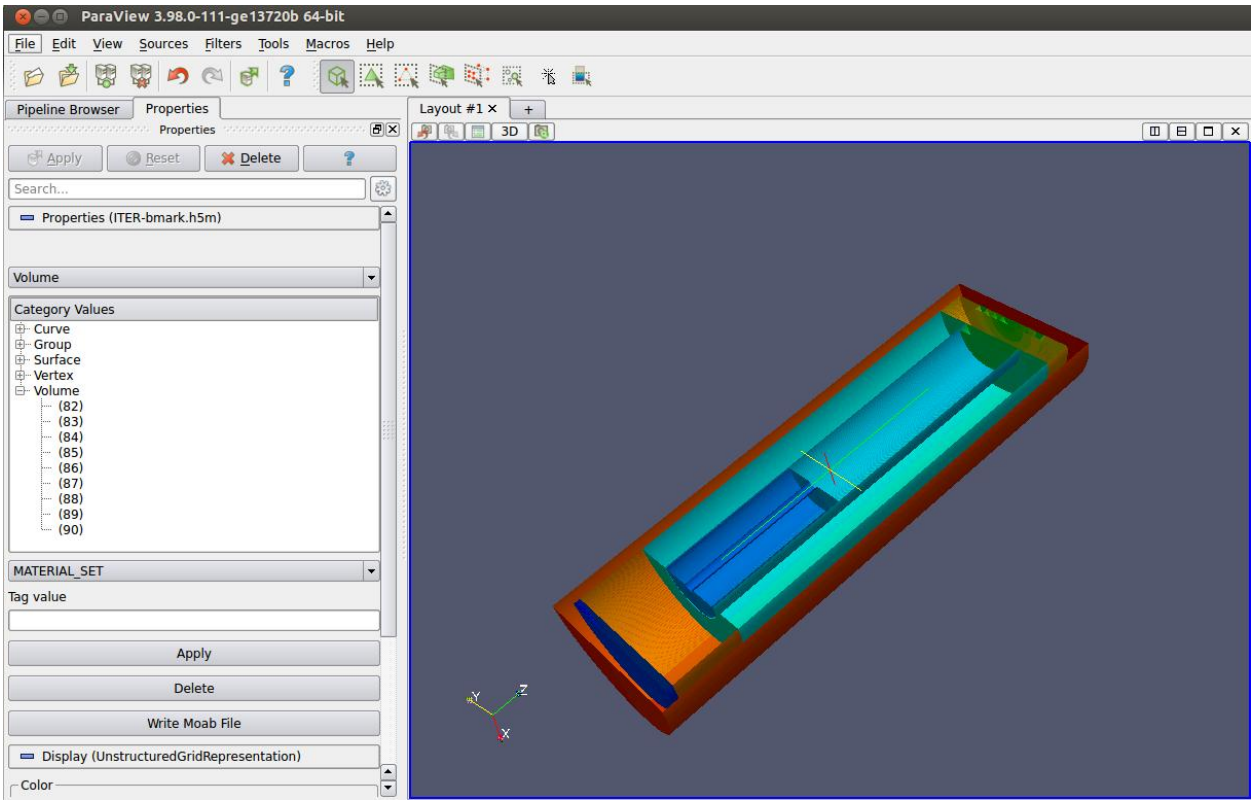


Figure 9. A figure of a geometric model and the plugin's view and widgets.

Expandability was a key component in the design so future needs could increase the geometry representations and data application. It would be relatively simple to add additional hierarchies within the tree with minimal code structural changes. An additional feature is the eventual development of smart boundary condition selection to apply graveyards and reflecting boundaries within ParaView in addition to material and tag information. Defining

neutron or source information would also be easy to add to the GUI once the proper implementation into the code was figured out.

3.1.3 Linking MOAB entities to VTK representations

Utilizing the basic "vtkMoabReader.cxx" ParaView reader plugin, relevant MOAB entity information was relayed into the VTK data model. Once the geometry has been converted into a ParaView-viewable format, selection occurs and executes with VTK algorithms and data structures. As the reader iterates over all desirable 2D or 3D entities, it transforms MOAB points and cells into VTK cells. In order to maintain the MOAB references within the VTK environment, each relevant VTK cell is labeled with field data that contains its MOAB equivalent's entity handle. Once in the ParaView GUI, selection obtains the VTK cell id, which has associated field data that provides the MOAB entity handle it is referencing.

At this early step, ParaView's visualization benefits have already become visually apparent. Figure10 shows an ITER benchmark that was used to check the progress of the plugin throughout development. This model is viewed in Cubit. Its exact construction details such as materials and purpose of cells can be seen in Appendix A, but is immaterial to the development. By adding in the MOAB entity information, Figure 11 shows how the visualization of the model has changed within ParaView. Distinct properties such as surface, volumes, materials, etc. can be visualized by examining color gradients.

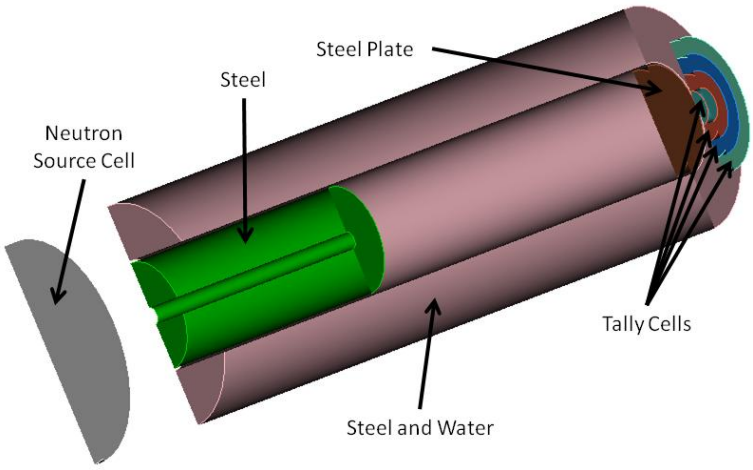


Figure 10. A Cubit solid model of the ITER benchmark

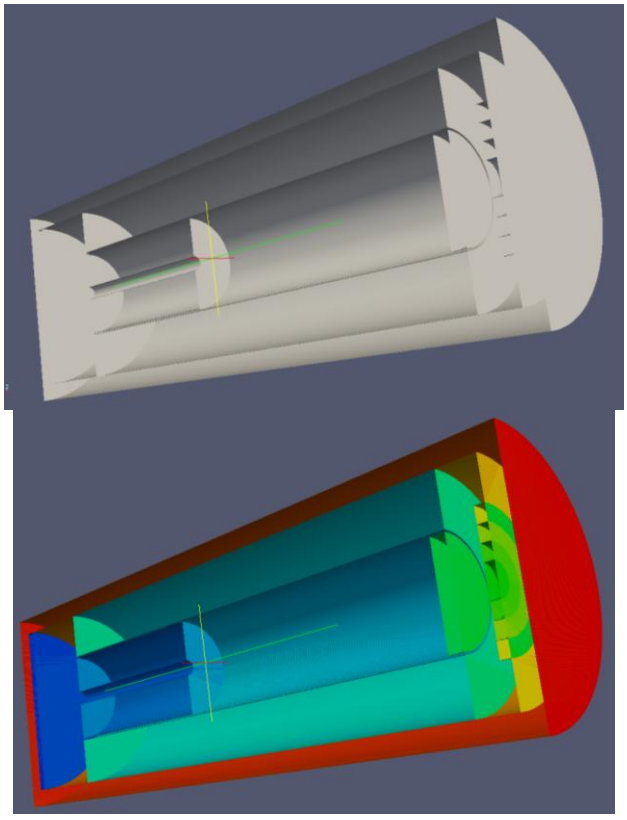


Figure 11. ParaView before (left) and after (right) allowing visualization based off of MOAB entity handles.

In addition to placing data on the VTK model representing the entity handles, the MOAB entities are tagged with data that ties them to the specific CATEGORY_TAG value

(volume, surface, etc.). This is necessary since an entity like a point could correlate to multiple CATEGORY_TAG values (a vertex, curve, surface, etc.). In order to have this data on the individual entities, a category bit masking tag is added on them. This bit masking tag assigns a 0 or 1 bit for each value. This set of bits is stored as an integer to minimize memory overhead. A 1 bit is used to initialize the tag. For each CATEGORY_TAG value, the integer is left bit-shifted; if the entity being tagged is a member of that value, the 0 bit is switched to a 1. The CATEGORY_TAG values are iterated over alphabetically. For example, if a point is a member of a volume and a group, but not a vertex, curve or surface, it's tag would be 101001 (initial-1; curve-no; group-yes; surface-no; vertex-no; volume-yes). In the next section, this bit masking is utilized to find the entity set that the user desires.

3.1.4 VTK selection connected to manipulating model

At this point, a prepared model is viewable in ParaView and a GUI panel exists with tag information selection. In order to connect the GUI selection in ParaView to MOAB entity sets, the user must first select a "picking" mode on the GUI panel. This drop down box contains all the possible category tag values introduced when the model was loaded.

Now, the user can select a single VTK cell up to all the VTK cells. Grabbing this set of cells, the field data from each cell is then extracted, which contains the corresponding MOAB entity handles. Placing these handles into a MBRange allows them to be fed into an algorithm that will predict what entity set the user most likely desired. Each MOAB entity handle is used to get the entity and corresponding bit masking tag. This tag is compared to the current selection mode, which will only ever have one bit that's 1 (true), except for the first bit (always 1). Any entities that don't have this bit true are ignored. The remaining

entities are iterated over to find out, which entity set(s) contain them. These entity sets are returned as the result of this algorithm.

One of the key concepts analyzed in performing this picking, selecting, and tagging was ensuring speed was optimized. In getting from the selection of VTK cells to returning entity handles, a high speed and memory cost can occur when a user selects every single VTK cell and thus, all the MOAB entities. In order to increase the speed, the code first checks the number of entity handles (from VTK cells) selected and the number of entity sets, whichever group is smaller is the group selected to iterate through. As this group is iterated through it is compared to the other group. If they value the iterator is pointing to is in the "other group," it is added to the list of results. Originally, the algorithm only ever iterated over the selection. By using the original method, the speed of returning the proper entity sets for a model with 4.1 million was 609 seconds. This dropped 30% by iterating over the smaller group.

The next step takes the user's desired entity set(s) and applies the tag key/value pair. The user selects the tag key on the model or chooses to create a new one. After entering a value and hitting either "Apply" or "Delete" the tag key/value is either added to the entity set or removed, respectively. When the user has finished adding all pertinent metadata, the MOAB file can be written by clicking the "Write MOAB to File" button.

3.2 Increase R2S-ACT Versatility

3.2.1 Limitations of original R2S-ACT

In the original version of the R2S-ACT workflow, Python iterated through each voxel and found the Cubit volumes that were in that voxel. Then, that volume information was used to obtain the material information. This material information was then stored along with the fraction of that material in each voxel. This worked effectively, but it introduced a problem when it came to obtaining activation data such as clearances and WDR. The problem was that each voxel was given the proper size in the ALARA input, and it was defined as a mixture of all materials that were present in that voxel. As models became more complex with higher-order surfaces, the small voxels on bordering surfaces were all different based on how it split the two or more conjoining entities. Clearance and WDR are volume integrated quantities and since the ALARA geometry did not keep original volume's information intact, it couldn't be calculated.

An example would be to have two voxels (a, b) containing two different cells (Cell 1, 2) and two different materials (mix_x, mix_y). The volume of each cell will be V_{a1}, V_{b1} and V_{a2}, V_{b2}, respectively, and this can be seen in Figure 12 below. Each voxel has an associated neutron flux, $\phi_{a/b}$. Now, R2S-ACT finds the volume fraction, associates it with the material and defines a mixture. If the WDR is desired in "Cell 1", this is lost since ALARA is really calculating the WDR over the combination of "Cells 1 and 2" within "voxel a", which could be drastically different from just "Cell 1" depending on B's material composition. In real reactor designs, some components have thousands of voxels or more, if each one of those was averaged with a wrong material, the final result would be entirely erroneous.

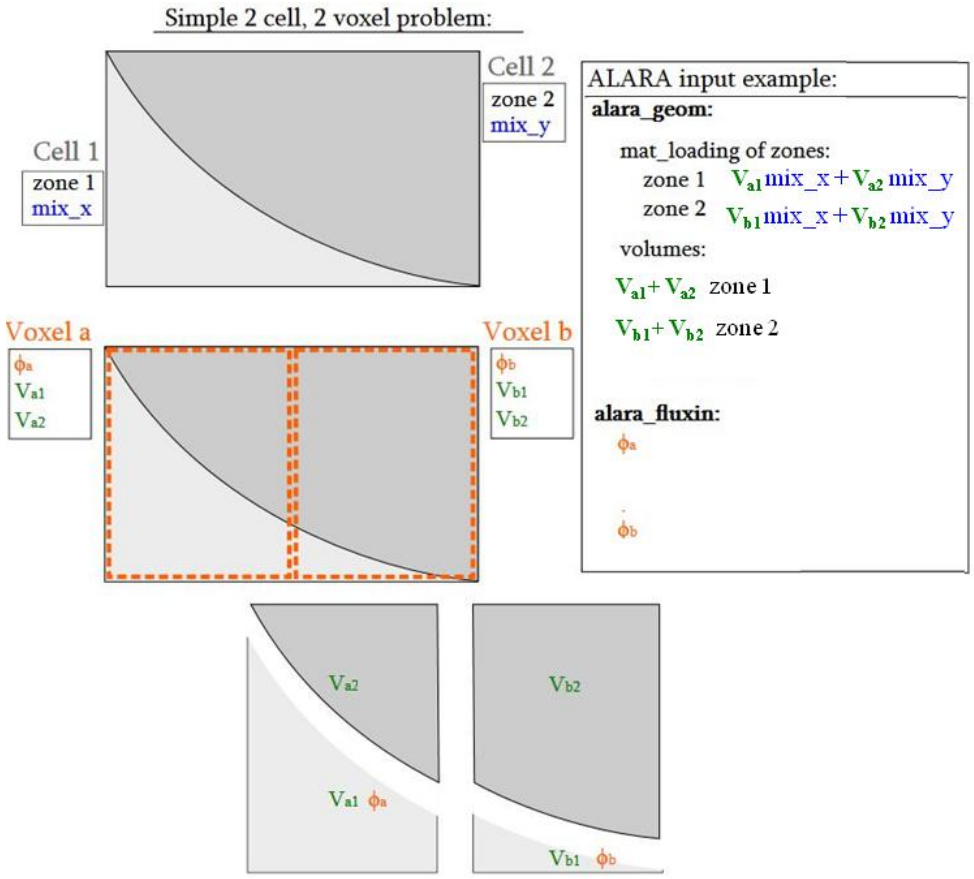


Figure 12. A depiction of the original R2S-ACT method

3.2.2 Implementing Cell-Based Calculations

To combat the limitations of the original R2S-ACT workflow, the Python script known as "mmgrid.py" was edited to iterate through the geometry and organize the information based off of cell instead of material/mixture. Figure 13 provides a good contrast against figure 12 and summarizes the steps performed for the ALARA input file.

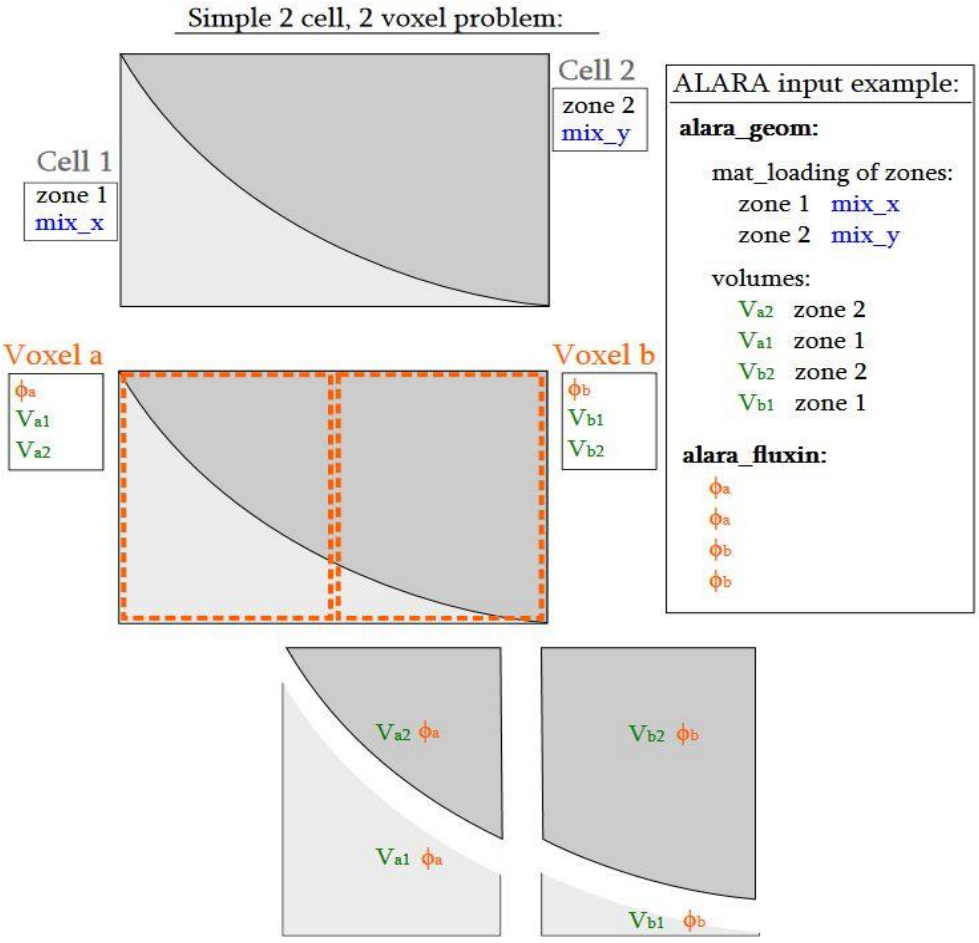


Figure 13. A depiction of the improved R2S-ACT workflow method.

In order to implement this, the cell information needed to be added as a tag on the meshed geometry. The volume list was called using PyTAPs and every cell was assigned a tag in the format of "Cell_#." Next required manipulation of the ray-firing algorithm in order to keep track of cells rather than materials. When ray-firing, sampling of the cell is performed by adding a normalized distance for each cell it encounters when in a voxel. A summary of the algorithm is provided.

```
for each dimension (x, y, and z)
Go in direction (u,v,w)
      for mesh square encountered
            for  fragment of cell encountered
                  Add normalized distance traveled for this cell to
an                array of all the ordered cells in the geometry
```

This summed value is found for each voxel and then normalized by the ray-tracing scores made in each voxel. This ray-tracing feature was still maintained for materials as well. In this way, accurate material information on a per voxel basis is also available for the mesh in addition to the cell information. With the information, each cell fraction in each voxel is iterated through and assigned a tag with its name ("Cell_#"), its material, and a ray-tracing error.

3.2.3 Create of Equivalent Geometry and Flux

At this point, there exists a structured mesh with numerous voxels with volume fractions of the cells that they contain. This mesh needs to be examined with PyTAPs and the ALARA input needs to be written from this. In "write_alara_geom.py," the following will be performed: write the geometry information, write material information, write the mixtures derived from the materials for each zone.

The ALARA input format controlled by "write_alara_geom.py" is summarized below. A full input file example can be seen in Appendix B.

```
geometry rectangular
      {Volume}     zone_#
      ....       ....
end
mat_loading
      zone_#                mix_#
```

```
    ....            ....
end
mixture mix_#
    material {name_from_library}    {Rel. Density}
    {Fraction in Mix}
end
....(add more mixtures)
```

In ALARA, zones are large areas over which data is desired to be collected. When making the modifications, the zone became representative of each cell. Zones were selected to be cells to maintain the idea of simplicity and allow the user maximum flexibility. Another option would have been to make zones correlate to materials; in some instances, however this would fail. For example, imagine that there is one component, Cell #1, made of Mix #1 and it is close to the plasma in a nuclear fusion reactor. Now imagine you have another component, Cell #2, that's beyond the vacuum vessel, but also composed of Mix #1. These components should not be combined if WDR is desired. This problem could not be fixed by the user either since they would have to know which volumes were for each cell in the input. To eliminate this problem, the zones are cell-based. If WDR is desired across multiple cells, then the user can change the output to collect results based off of mixture instead of zone.

In keeping with simplicity, the mixtures are numbered matching to the material identification numbers of the materials placed on the Cubit model. The material definitions within each mixture have a name of the form "mat#rho#"--pulled from the geometry--with 100% relative density and fraction in mix. In most cases, the material list will need to be altered to match the material library the user has defined. The last step is to simply match the zones to their respective mixes. This information is on the mesh already, so the program simply iterates over all the zones (cells), and finds out what mixture is associated with it.

With the geometry details of the ALARA input, the file is generated and the user has an executable ALARA input deck.

The last modification to the R2S-ACT workflow is to print out the correct number of fluxes for all the volumes. The old workflow only required one flux for each voxel. As figure 13 reveals, each voxel needs a flux repeated for each cell within its boundaries. The only necessary change was to add a loop over each cell within each voxel after entering the loop containing each individual voxel. This adds minimal complexity to the code and should trivially slow down the step, since writing the fluxin file only takes a small fraction of the time that the ray-firing takes.

3.3 The Future of the Nuclear Analysis Workflow

As these developments were added to the nuclear analysis workflow, the research and results obtained by the users of this software will become more accurate and meaningful. There is always the need for more advancements, nonetheless. As soon as improved methods are introduced, new ideas and improvements are already underway. On the ParaView side, more generalizations of the plugins can be developed along with easier selection and manipulation of the data, including, hopefully, editing the geometry rather than just metadata. The R2S-ACT's next big improvement will be to fully support unstructured meshes. DAG-MCNP already possesses the capability to create an unstructured mesh to perform neutronics analysis, adding that ability to the Python scripts and utilizing it in ALARA would provide increased accuracy in collected results. This is due to the fact that a conformal mesh could almost identically match the curves of the real cells within a reactor system.

4 Validation and Testing of Developments

One of the most pertinent aspects of any scientific computing development project is to ensure the program does as desired when ultimately finished. A series of excellent practices is provided by Wilson, et.al. [19] and includes focusing on writing programs simply for people, making small changes, validating along the way, and a handful of other key concepts. These guidelines were followed with this nuclear analysis pathway development in order to result in a successful series of programs. This section goes over the testing and validation used in each of the code developments introduced.

One of the first methods used to develop the code was setting up revision control. Github [x] was used in order to provide ultimate versatility and store data online. This allowed the syncing and use of the altered code from computers in any location with internet access. On top of this feature, revision control allows use of "commit" messages to keep track of code changes. This provides additional comment-like structures to development and changes can be reverted or accepted. Probably the most important aspect of revision control is the ability to perform "pull requests" and gain peer review of progress. This allows additional programmers to help correct since small errors such as forgetting a semi-colon, which cause major frustration.

Another development tool used was unit tests. These are small snippets that check to see that portions work as expected, and just as importantly, fail when they are expected to. This in conjunction with printing information to the command line, helped provide a general understanding and confirmation that the code was working as expected.

The final step was to evaluate real-models from simple solutions to complex solutions that will be referred to as benchmarks. These were specifically designed with the intention to

test some form of the code along the way. Due to the relative simplicity, these validation models could objectively be compared to known-as-accurate results garnered through a different process.

4.1 Validating the R2S-ACT Workflow Modifications

The R2S-ACT workflow modifications broke a script called "r2s_step1.py" that fired rays through the meshed geometry, wrote the ALARA geometry file, and wrote the ALARA flux file. There were three major benchmarks: one to produce the same results that the original workflow would have, one to test simple cell divisions of voxels, and one to analyze more complex divisions.

The basis of all three benchmarks was a parallelepiped 5 cm by 5 cm with a 20 cm depth designed in Cubit. Reflecting boundaries were placed on the x and y surfaces and a graveyard (zero importance/particle termination) was placed on the z-direction caps. A 1 MeV mono-directional source was placed at the furthest end and directed down the geometry. A 175 neutron group flux was calculated for each interval. Figure 14 below depicts these properties. During the ray-firing step, except where specified, the code fires 50 rays per mesh row.

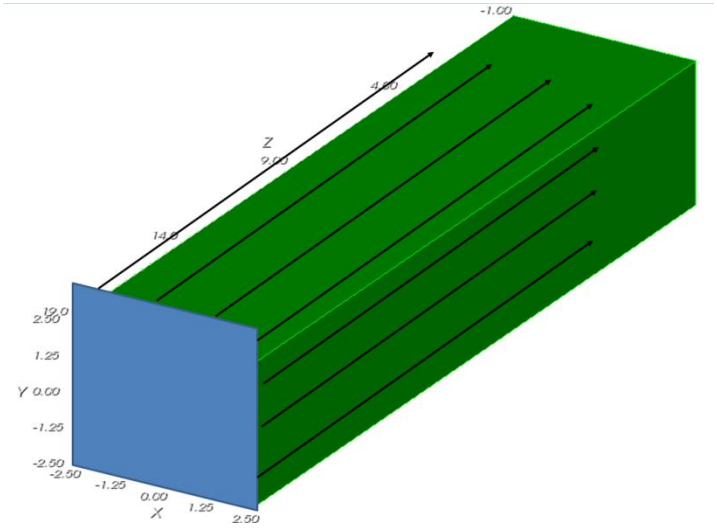


Figure 14. The basic geometry that all the benchmarks are derived from. The neutron source is also shown.

For the subsequent benchmarks, there will be two equivalent tests at each level. The first one will now be referred to as R2S-ACT (this is the version with the modifications). The second one will be referred to as "native MCNP/ALARA." The second one will be used as the reference case in all scenarios since it only utilizes analysis programs and code that has been extensively validated (MCNP and ALARA). For the volume values used in ALARA, the native MCNP/ALARA tests use Cubit to find the analytic volume. In order to create the flux neutron flux file for native cases, MCNP f4 tallies (cell flux: neutrons/cm^3) are used with energy bins identical to the fmesh4 card used in the R2S-ACT method. Each of those fluxes is then manually copied and pasted into a "manual-fluxin" file for ALARA to process. In this way, the R2S-ACT workflow is never utilized for the reference case. The input files for MCNP and ALARA, as well as some output can be found in Appendix C.

4.1.1 Benchmark #1

The first, and simplest, benchmark was designed simply to test that accurate volume information was getting input from the ray-tracing into the ALARA input. Mainly, it ensured that the code had all been written correctly.

For this benchmark, the entire model is composed of 100% Fe at 7.874 g/cm^3. The flux normalization was 10^8 n/s with a 10 year flux and a 0.15 year resting period. By choosing the material such that it encompassed the whole volume of each voxel, this ensured that both the original R2S-ACT workflow and the cell-based modified R2S-ACT workflow would provide the same results. In addition, a second model could be created that manually "slices" the rectangle into the same volume elements as the voxels in the mesh. These results are then matched for accuracy. The two models used are shown below in figure 15.

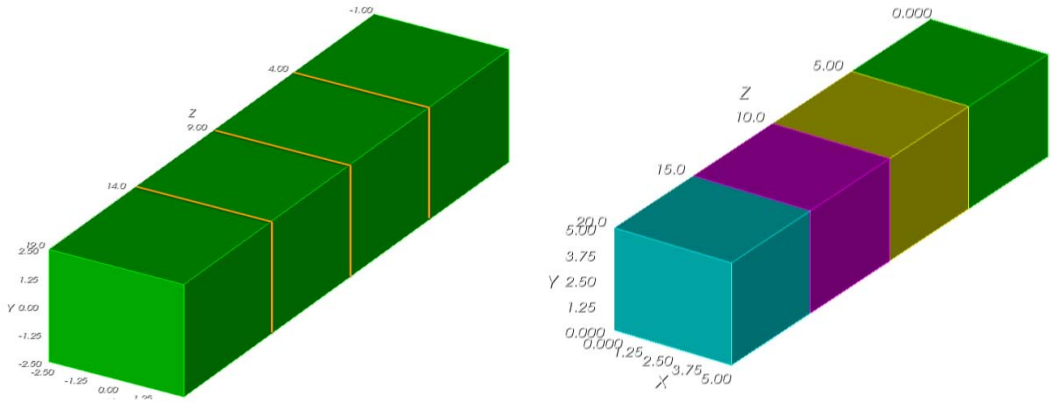


Figure 15. The figure on the left utilizes R2S-ACT, the figure on the right uses only native MCNP/ALARA.

The first check was to ensure the ALARA input had been properly created. The simplicity of this model lends itself to checking the input and it matches perfectly. The interval sizes are exactly 125 cm^3, as expected, and the mixture and zone definitions are properly defined for the cell. Next, the activation of the elements is analyzed. The specific

activity in each interval or voxel, is shown below in Table 1. It is found to be the same for each interval in the native ALARA step (Figure 15 on the right) and the R2S Benchmark #1.

Table 1. Interval-based activities for benchmark #1

Interval	Base Case [Ci/cm^3]	Benchmark #1 [Ci/cm^3]	Volume [cm^3]	Total Activity [Ci]
1	6.8794E-12	6.8794E-12	125	8.5993E-10
2	4.7616E-13	4.7616E-13	125	5.9520E-11
3	8.0451E-14	8.0451E-14	125	1.0056E-11
4	1.9557E-14	1.9557E-14	125	2.4446E-12
		Total::	500	9.3195E-10

The average zone activity is then calculated from the intervals and compared to the activity calculated from figure 16, which simply finds the flux over the whole geometry. Table 2 shows that the same numbers are calculated. The benchmark confirms that it has retained the same computational technique as the material-based R2S method.

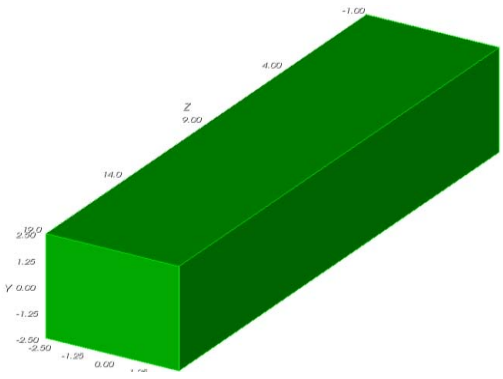


Figure 16. The benchmark geometry as one zone. The average flux over this one zone is the same as calculated with the mesh.

Table 2. Zone averaged activities for benchmark #2

	Specific Activity [Ci/cm^3]
Base Case Calculated Zone:	1.8639E-12
Calculated Zone Averaged (Benchmark #1):	1.863892E-12

4.1.2 Benchmark #2

The second benchmark aimed to confirm that two different cells split across voxels would be properly calculated. Figure 17 shows the R2S geometry in which there are five separate cells. In the R2S geometry, the first cell is 2.5 cm in depth in order to offset each full cell at the midpoint of each voxel. Also, an additional material was introduced. The green colored material is still the Fe at 7.874 g/cm^3 and the red colored material is W at 19.35 g/cm^3. The flux is normalized to 10^{16} n/s. An irradiation history of a 4.9 year pulse with a 0.1 year rest was used along with this normalization.

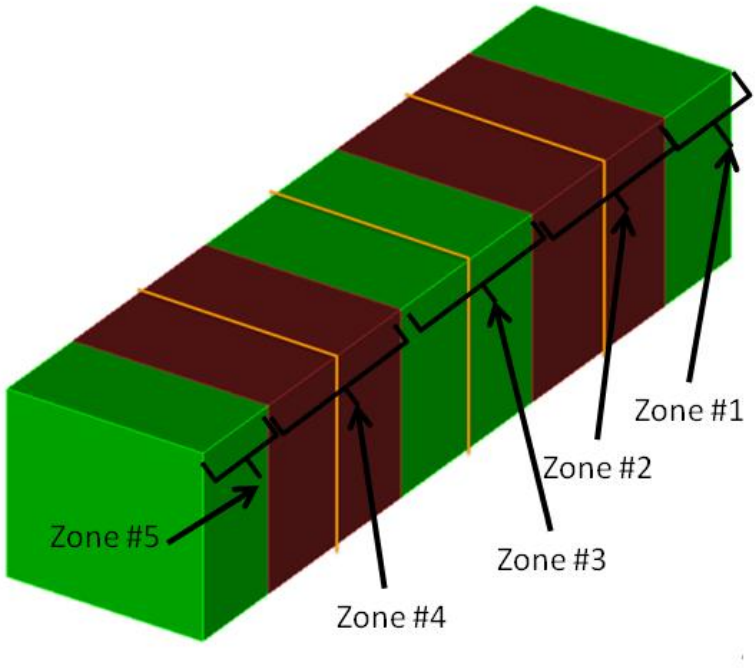


Figure 17. Benchmark #2 uses two different materials. The cells are split at exactly halfway by the mesh.

The native ALARA model has the same geometry except there are no cells split by voxels. Table 3 shows the interval and zone differences between the R2S model and the native ALARA model. Acquiring the flux in each full cell of the model in all 175-neutron groups, this information was entered into a fluxin file for the native input. In comparison, the R2S model would have eight fluxes of 175 groups (one for each cell in each mesh interval) and the native ALARA model would only have five fluxes of 175 groups (one for each cell in the entire geometry).

Table 3. The ALARA inputs for both variants are set up with the parameters in this table

Native ALARA			Benchmark #2 ALARA		
Volume	**Zone Label**	**Mixture Label**	**Benchmark #2 [cm^3]**	**Zone Label**	**Mixture Label**
62.5	Zone_1	Mix_1	63.1	Zone_1	Mix_1
125.0	Zone_2	Mix_3	61.9	Zone_2	Mix_3
125.0	Zone_3	Mix_1	63.1	Zone_3	Mix_1
125.0	Zone_4	Mix_3	61.9	Zone_2	Mix_3
62.5	Zone_5	Mix_1	62.376625	Zone_3	Mix_1
			62.683375	Zone_4	Mix_3
			63.016625	Zone_5	Mix_1
			61.983375	Zone_4	Mix_3

By printing output information to the shell during the execution of the "r2s_step1.py" script, the success or failure of the benchmark could be monitored. One of the output checks was to ensure the cell and materials matched for each voxel after ray firing. Since the number of cells is not equal to the number of materials, these lists would be of different size, but by recording their values at each voxel iteration, a proper comparison could be made because the list order does not change. Another output observation is ensuring the Python dictionaries for the cell fraction is as expected for each cell. This will mean something along the lines of "At voxel 1: {(10, 'Cell_10'): 0.50, (7, 'Cell_7'): 0.20, (3, 'Cell_3'): 0.3}" would appear and this would be compared to the Cubit fractions. These comparisons matched.

Performing the R2S-ACT ALARA step and comparing it to the native ALARA step, revealed that the code was successfully altered to gain results based on cells instead of

materials. The table below compares the "base case" (native ALARA) and benchmark #2 (R2S-ACT ALARA).

Table 4. Benchmark #2 specific activities at shutdown

Zone	Base Case [Ci/cm^3]	Benchmark #2 [Ci/cm^3]	% Discrepancy
1	4.8198E-03	4.8198E-03	0.00%
2	3.9859E-01	3.9894E-01	-0.09%
3	3.4698E-03	3.4635E-03	0.18%
4	1.8070E-01	1.7726E-01	1.90%
5	1.1984E-03	1.1984E-03	0.00%

The discrepancies between the zones is almost non-existent. The differences that do exist are due to some flux statistical errors from the MCNP input that ended up affecting the final answer. This fluctuation, when combined with the small discrepancies between the ray-tracing volumes calculated, can explain the small variance.

4.1.3 Benchmark #3

The final benchmark for the R2S-ACT modifications involved a more complex geometry that didn't involve even splitting of each voxel. Another key component of this model is the addition of a void region. The void region serves to ensure that the code does not fail when encountering an empty volume. For this benchmark, accuracy was found by printing out the volume of each cell in each voxel. This information was compared to Cubit's

calculation of the volume. Two materials were used in the figure below--tungsten and iron--along with the void.

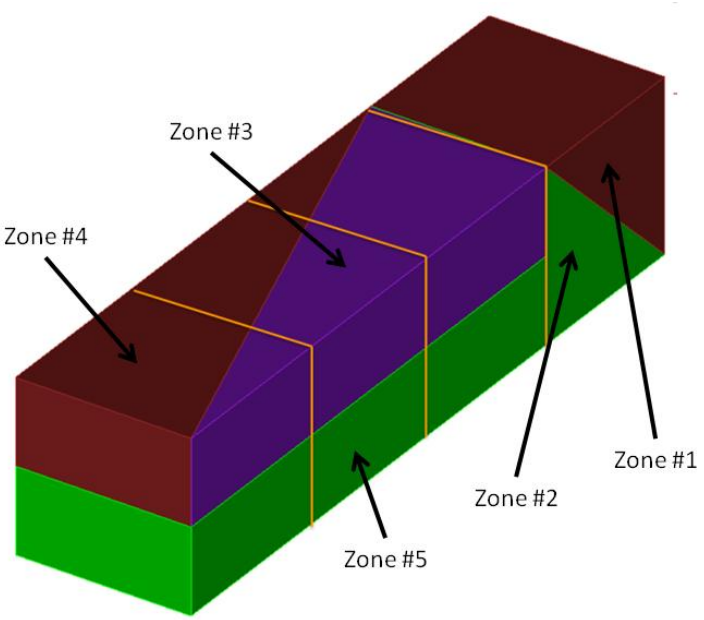


Figure 18. The final benchmark has a void region (purple) along with the tungsten (red) and iron (green) materials.

The first check in the process was once again to confirm volume calculations. By printing out the volume fractions for each voxel for each cell to the Unix shell, Table 5 was compiled. The fractions found by R2S agreed quite nicely with the analytic volume fractions. These values were less than the 2.5% maximum error that the ray-tracing algorithm computed.

Table 5. This table shows the volume fraction comparison between the R2S method and the analytic volume

Voxel #	Zone #	Analytic Volume Fraction (Cubit)	R2S Volume Fraction
Voxel # 1	1	50%	50.08%
	2	50%	49.92%
	3	0%	0%
	4	0%	0%
	5	0%	0%
Voxel # 2	1	0%	0%
	2	0%	0%
	3	41.67%	40.55%
	4	8.33%	7.83%
	5	50%	51.62%
Voxel # 3	1	0%	0%
	2	0%	0%
	3	25%	24.62%
	4	25%	24.40%
	5	50%	50.98%
Voxel # 4	1	0%	0%
	2	0%	0%
	3	8.33%	8.34%
	4	41.67%	41.24%
	5	50%	50.42%

Once again activity was compared between the meshed geometry and the original geometry placed into native ALARA. The fluxes were gathered for each cell from MCNP for the native version and input into the ALARA geometry. This was then compared against the meshed R2S ALARA results. The values along with the discrepancies between the two is shown in Table 6 below. In the first zone and last zones, the discrepancy was rather high.

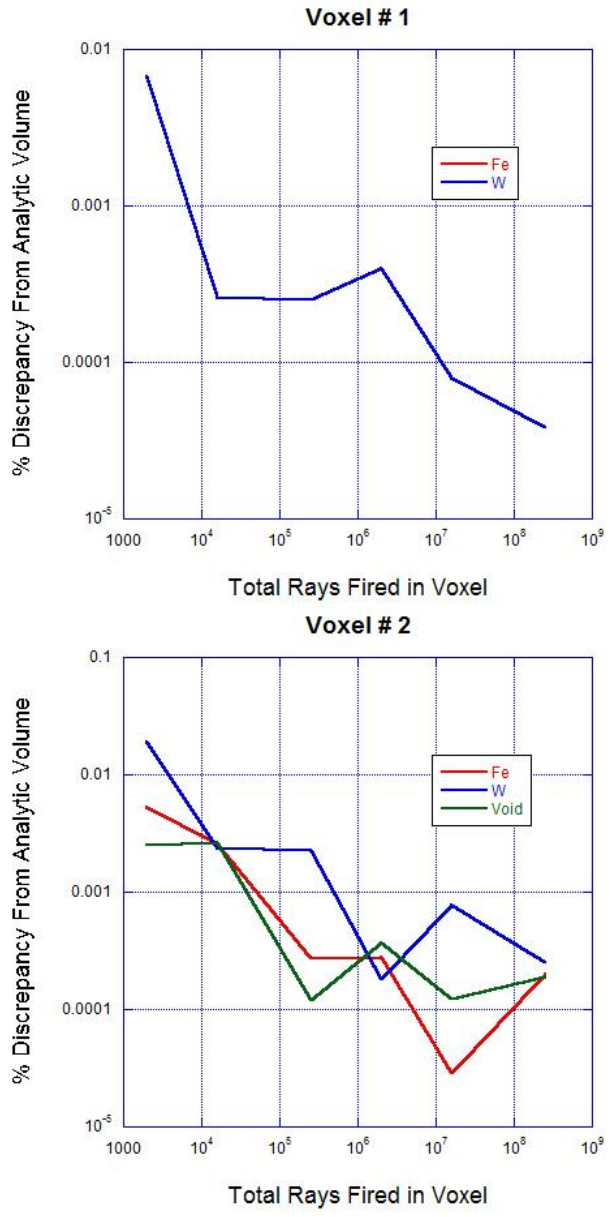
Table 6. The specific activities of benchmark #3

Zone #	Specific Activity (Native ALARA) [Ci/cm^3]	Specific Activity (R2S ALARA) [Ci/cm^3]	% Discrepancy
1	3.61450E+01	2.52890E+01	30.03%
2	0.00000E+00	0.00000E+00	--
3	3.80240E-01	3.86140E-01	-1.55%
4	4.97300E+01	4.83550E+01	2.76%
5	2.11380E-01	2.51990E-01	-19.21%

4.1.4 Ray-firing analysis

The error introduced does raise an interesting topic of discussion for nuclear analysis: how many rays should be fired for the 3D activation analysis. This question has many factors contributing to it including how complex the geometry's surfaces are through that voxel, how large the voxel is relative the geometry size, and the random numbers of the starting rays. An analysis of ray tracing on the third benchmark was performed to better understand the changes. More rays are expected to increase the accuracy of the volumes obtained. In order to offset the random number selection of the ray-firing, 100 runs were performed of each volume calculation and then averaged. The percent discrepancy from the accepted Cubit volumes is plotted as function of rays for each separate voxel and within each cell. The discrepancies are all expected to be under the maximum error observed. The volumes are

well within the error in all cases. The maximum is high enough that it is above the maximum of the plot. Figures 19-22 show this for each voxel.



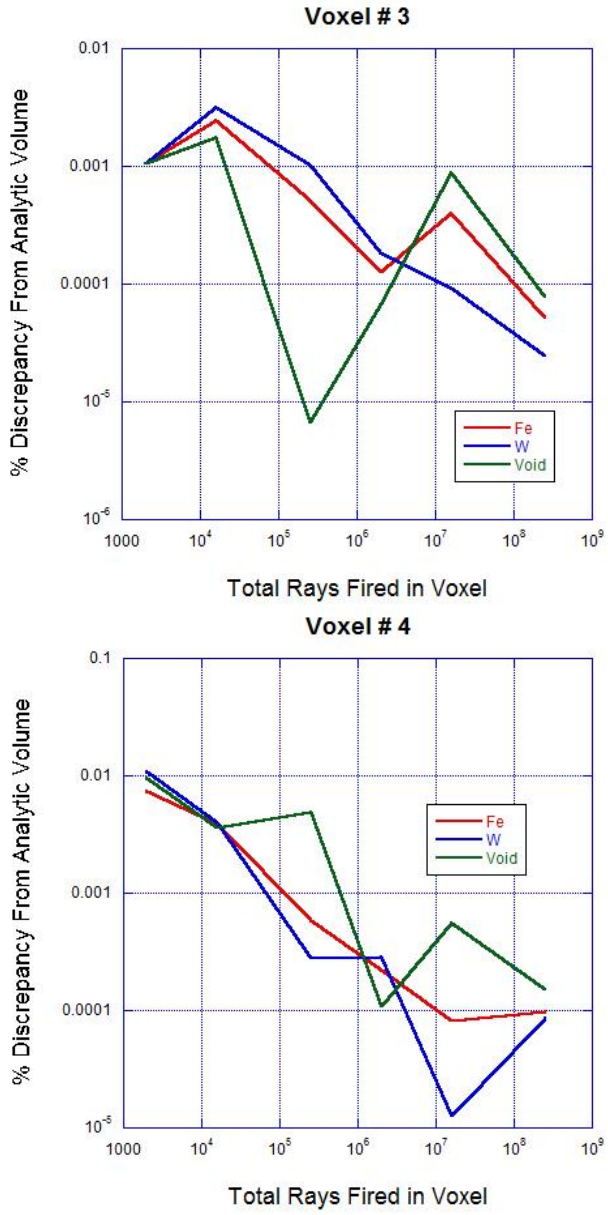


Figure 19, 20, 21, 22. The discrepancies between analytic and calculated volume for each voxel

The general trend of the plots reveals that more rays being fired calculates the volume more closely to the analytic volume, as expected. Voxel 3 seems to reveal that at some point, as long as the results are within the error, increasing rays isn't justified due to computer time. In order to take a more close look at how ray-tracing will perform on real-world problems

was examined by summing each cell's volume in each voxel over the entire geometry for benchmark #3 and then comparing that to the analytic cell volume. This results in figure 23 below. Once again, a downward trend is seen, but the benefit seems to be minimal once below expected error and statistical noise introduces random fluctuations.

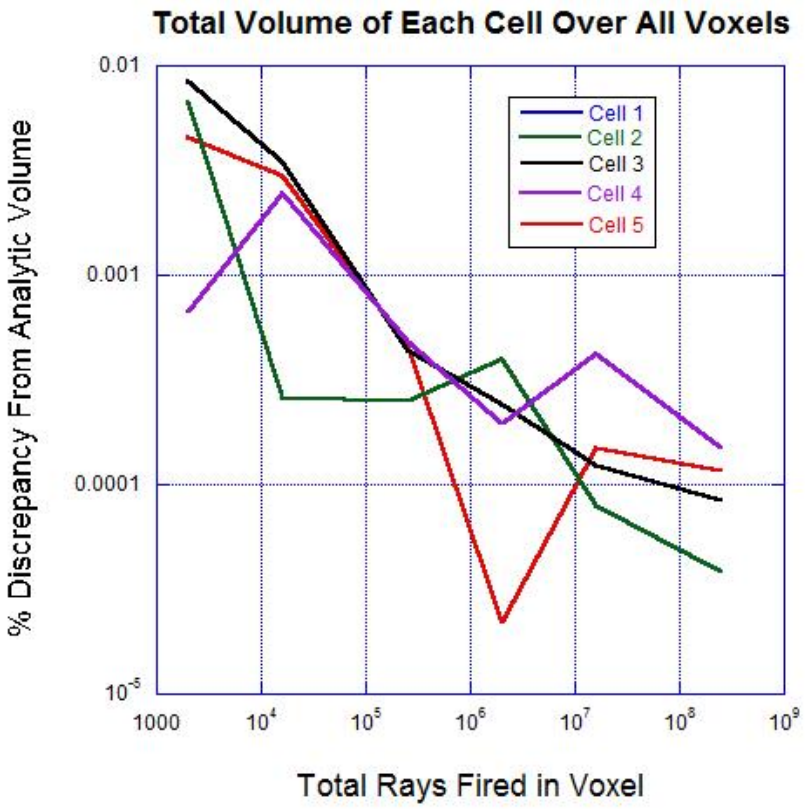


Figure 23. The discrepancy between cell volumes over the entire geometry due to ray tracing

5 Application of Developments

The code developments documented necessitate a real-world application in order to fully comprehend their usefulness to the users. There are multiple pathways that an analyst can use in order to gain accurate results from a nuclear fusion system. By using these

developed methods, a simple, clean pathway is revealed that works quickly and efficiently, and hopefully reduces problems that may arise from using other pathways. In order to be successful, a number of codes are used and many problems solved. This series of steps can result in error in application that would be unacceptable for a company or research institution. This chapter examines a real-world problem in the fusion community by utilizing this pathway and preventing user-error that may arise.

5.1 Developing the Geometry Model

The nuclear fusion power plant studied is the ARIES-ACT-1 (SiC/LiPb)--the most recent design in the ARIES series. The ARIES Team [20] has developed a multitude of nuclear fusion reactor studies. An extremely in-depth analysis is performed that provides detailed results and analysis from power generation to radwaste management to a cost analysis. This specific study utilizes a combination of Aggressive and Conservative Technologies (ACT), and in the case of ACT-1, looks at an SiC-based blanket with advanced physics. The fusion power of the device is 1804 MW and it has a major radius of 5.5 m. Figure 24 below labels some of the more important components of the device.

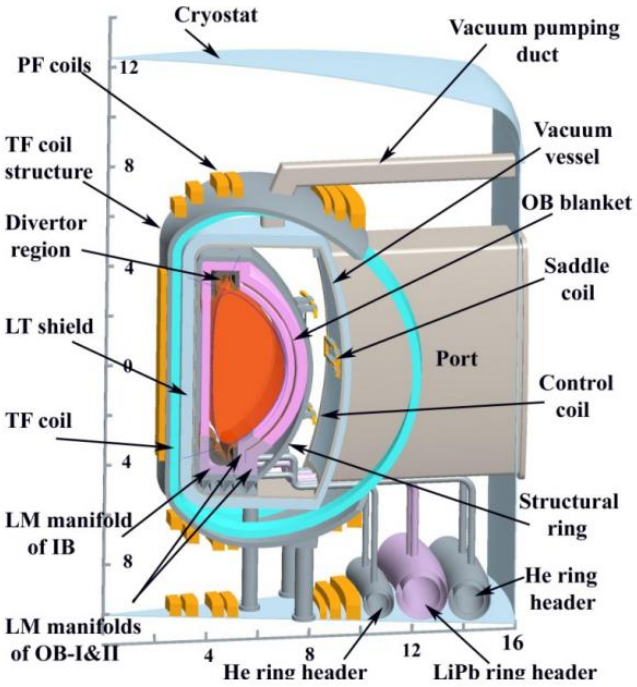


Figure 24. A slice of the ARIES-ACT-1 (SiC/LiPb) fusion device.

For ease of 3-D modeling, an 11.25° toroidal section of the plant, cut at the midplane was modeled like in figure 25. Reflecting boundary conditions were applied at the outside edges of the 1/64th model in order to simulate the full device. The plasma region is split into three nested source distributions for analysis in MCNP. The center has 63% intensity, then 32.5%, and then 4.5%. This model was created using Cubit as the solid modeling engine.

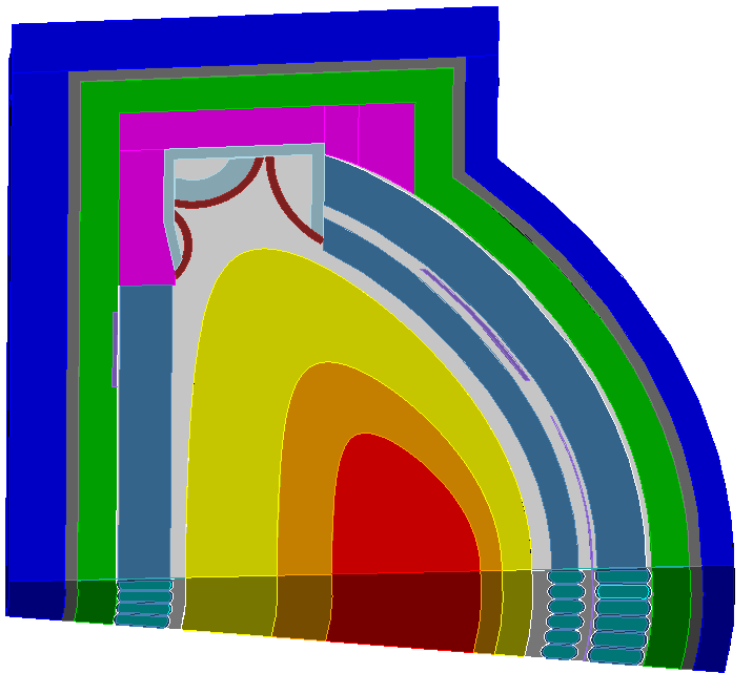


Figure 25. The modeled 1/64th device. The nested plasma source is shown in gradient colors.

5.2 Preparing Model for Neutronics Analysis

In order to add the metadata necessary to the model, it was opened in ParaView with the plugins activated. The DAG-MCNP analysis code is being edited to handle the proper tag key/value pairs when it comes to materials and MCNP tallies that will be added with this new, more accurate method.

Once visualized the proper material metadata is added. These are summarized in Table 7 below. The MATERIAL_TAG references the MCNP material number and the MATERIAL_DENSITY_TAG references the density in atoms/(barn-cm) if it is positive, and density in g/cm^3 if it is negative.

Table 7. The material metadata added to the ARIES-ACT-1 model.

MATERIAL_TAG	MATERIAL_DENSITY_TAG	MATERIAL_TAG	MATERIAL_DENSITY_TAG
1	0.030167	13	0.0725354
2	0.06372742	14	0.0859713
3	0.037611	17	0.04650951
4	0.0664162	18	0.070499
5	0.0670813	19	0.042139
6	0.0842419	20	0.0408082
7	0.0659506	21	0.0428037
8	0.0640882	22	0.050183
9	0.0804506	23	0.043649
10	0.0722694	24	0.0632489
11	0.0720699	26	0.058622
12	0.070274		

5.3 Entering the R2S-ACT Workflow

Once the model has been pre-processed and prepared for analysis, it begins to enter the R2S-ACT workflow. The ultimate goal of this study is to find the specific activity, waste disposal rating, recycling dose rate, and clearance index of the inboard first wall, which are shown in figure 26 below. First, a meshed neutron flux file from MCNP is required. In order to do this, an MCNP input is prepared. The input deck (contained in Appendix C) defines the constituents of the materials, the source, and tally definitions. Since an activation result is desired, the fmesh4 (similar to an f4 tally, but a mesh can be defined) card is the most important tally in the problem. The mesh is designed to be a bounding box around the four cells that compose the first wall. An approximately 2 cm by 2 cm by 2 cm mesh size is requested from the program. In addition, the flux is also being gathered in full individual cell

in order to perform validation later. The input is combined with the faceted geometry file, and MCNP runs for 1e8 particle histories.

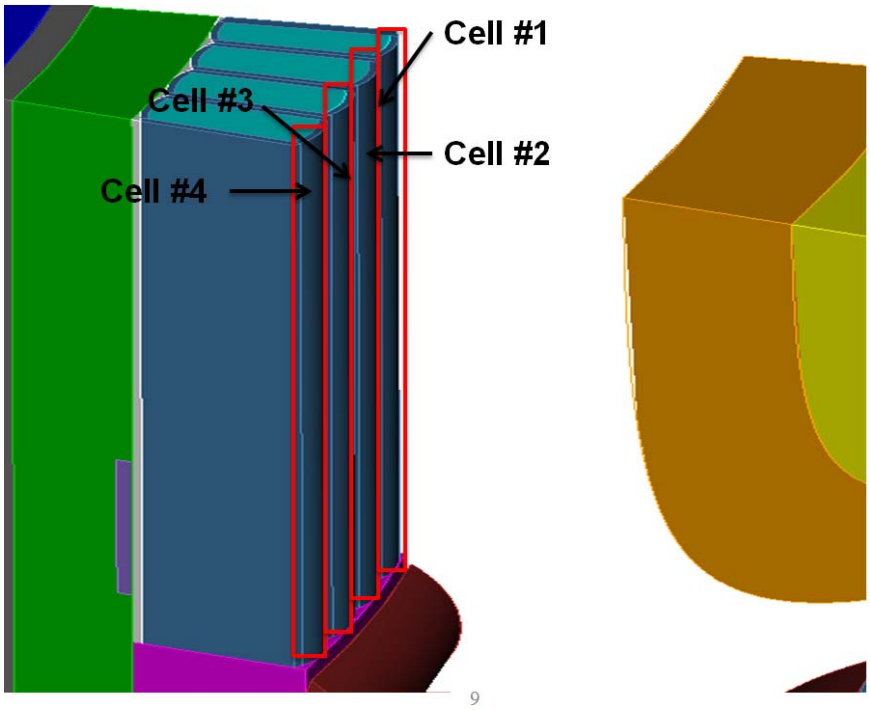


Figure 26. The red boxes are surrounding the curved inboard first walls of the device.

5.3.1 Discretized Mesh Source Creation (Step 1)

In order to create a valid input deck for ALARA next, the neutron flux mesh file has to go through processing in the Python "r2s_step1.py" step. A configuration step has defined all the names of the files needed--the geometry file, the input file, and the meshtal file--so the user can just run the Python script. For this analysis, ten rays are fired along each face in each voxel. This implies that 2000 rays are being fired per voxel. With the amount of voxels selected, the script must fire just shy of 24.5 million rays. The whole process takes approximately two hours to complete. Following this, the R2S-ACT workflow prepares the ALARA input file and the fluxin file for the problem.

Figure 27 below shows the meshed neutron flux file overlaid on a wireframe model of the device. As expected there is a higher neutron flux at the midplane (the reddish region) and the flux decreases further up the z-axis. The lack of symmetry toroidally is due to the fact that the structured mesh is a rectangular bounding box for the first wall cells, whereas the actual reactor geometry is rotating around the z-axis.

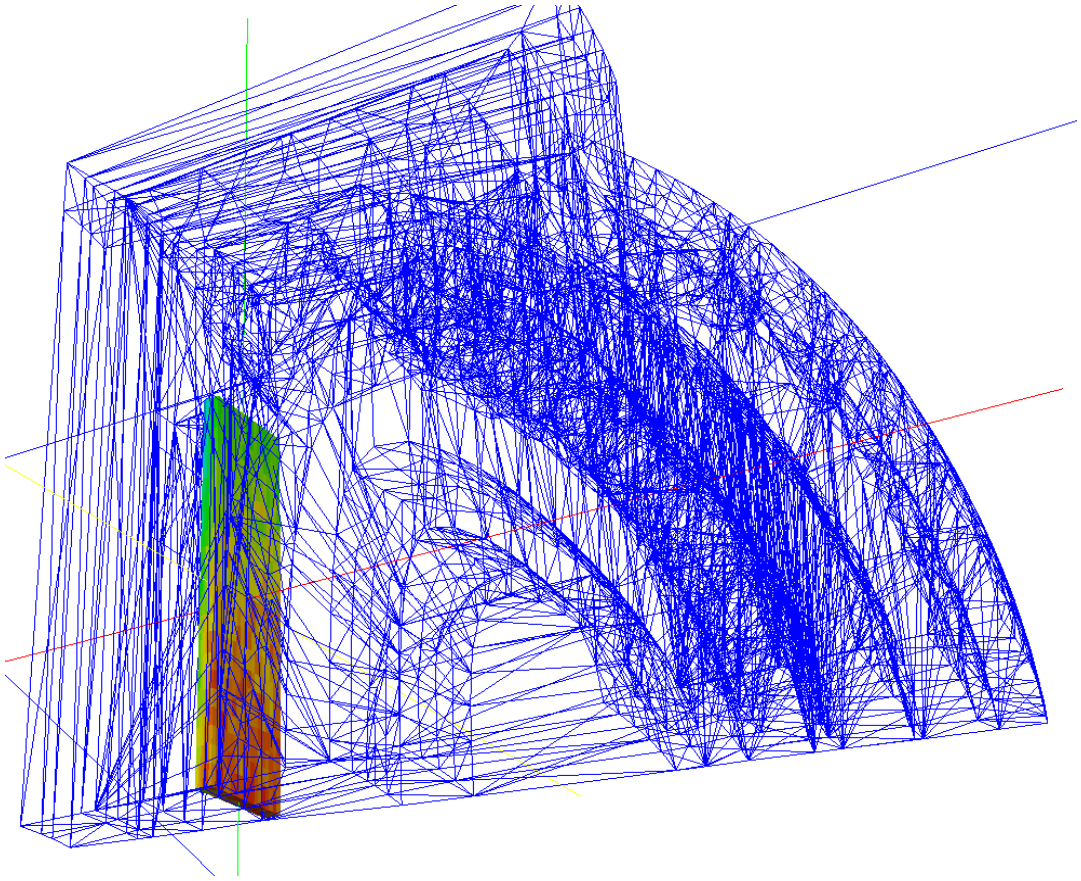


Figure 27. The neutron flux mesh file overlaid on the reactor geometry.

5.3.2 Performing the Activation Calculation (ALARA)

Following the completion of R2S-ACT's Step 1, the user is given two files necessary for finding the activation in the fusion system: "alara_geom" and "alara_fluxin." For the ARIES-ACT-SiC fusion system, specific activity, total decay heat, WDR, recycling dose, and clearance index are desired at a multitude of time steps following shutdown.

Utilizing the 1804 MW fusion power value for ARIES-ACT-1, the flux input file is normalized by $9.99727*10^{18}$ n/s. For this analysis, the inboard first wall was examined by irradiating it under this flux schedule for 3.8 FPY (Full Power Years) with 85% availability. In addition to this 3-D analysis, the results were compared to the 1-D results gathered from the PARTISN 1D flux input to ALARA using the average NWL over the appropriate first wall. The specific activity, total decay heat, recycling dose, IAEA clearance index, and FetterLo clearance, respectively, are plotted below for the inboard first wall, the inboard vacuum vessel, the outboard first wall, and the outboard vacuum vessel. Here, the 3D results are slightly lower at early times. This variance comes from the approximation made to estimate the 1-D normalization. The results also confirm that the method used to approximate the 3D heterogeneity effects are a very good 1D approximation of a complex 3D system. Also, in all results the 1D activation ends up very close to the 3D activation results, which confirms the accuracy of the method.

Figures 28 through 32 represent the model's inboard first wall.

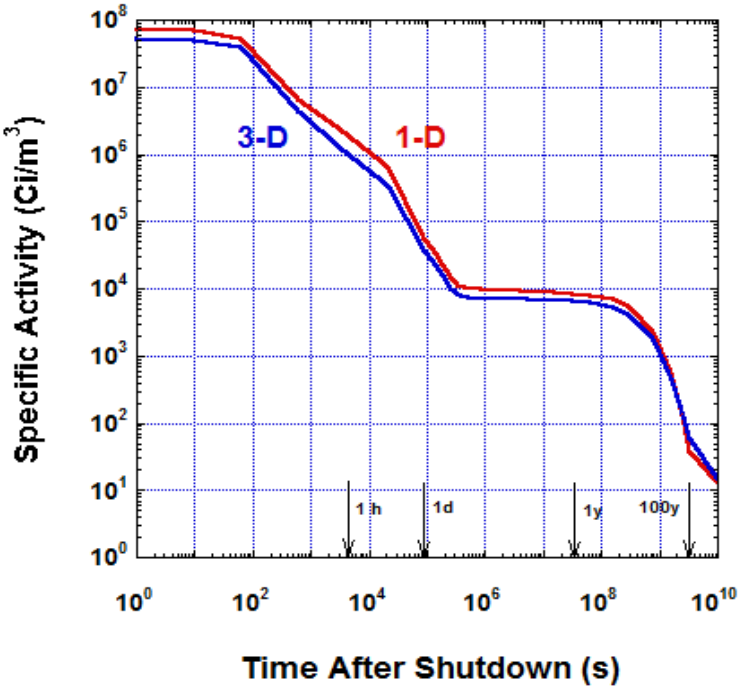


Figure 28. The specific activity for the inboard first wall.

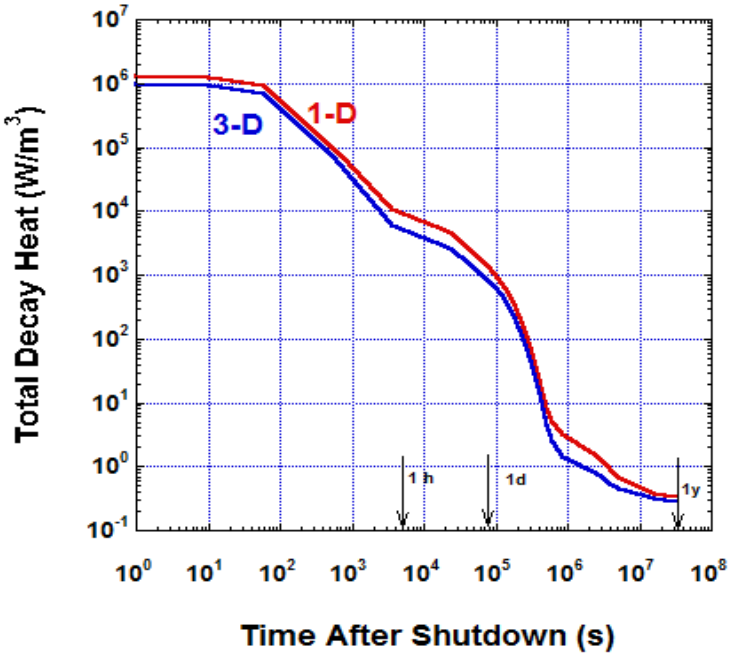


Figure 29. The total decay heat for the inboard first wall.

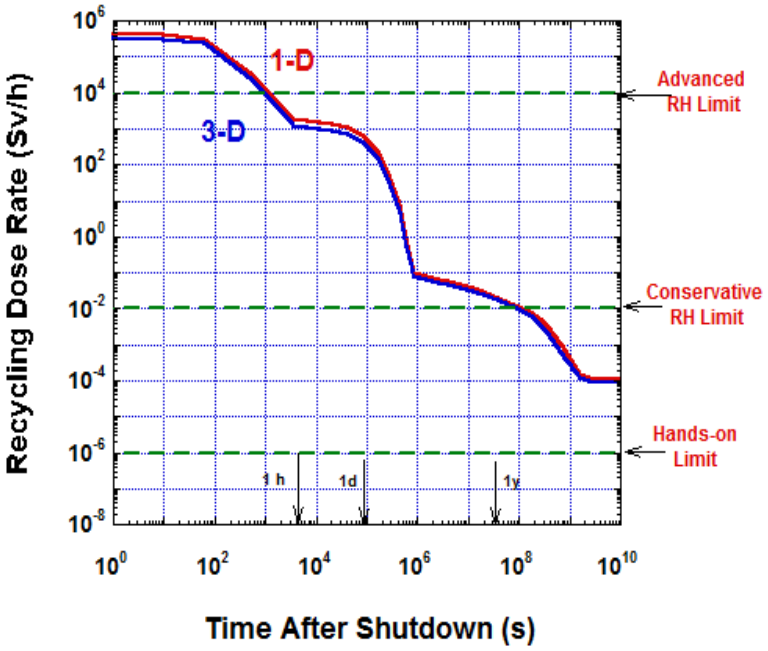


Figure 30. The recycling dose rate for the inboard first wall.

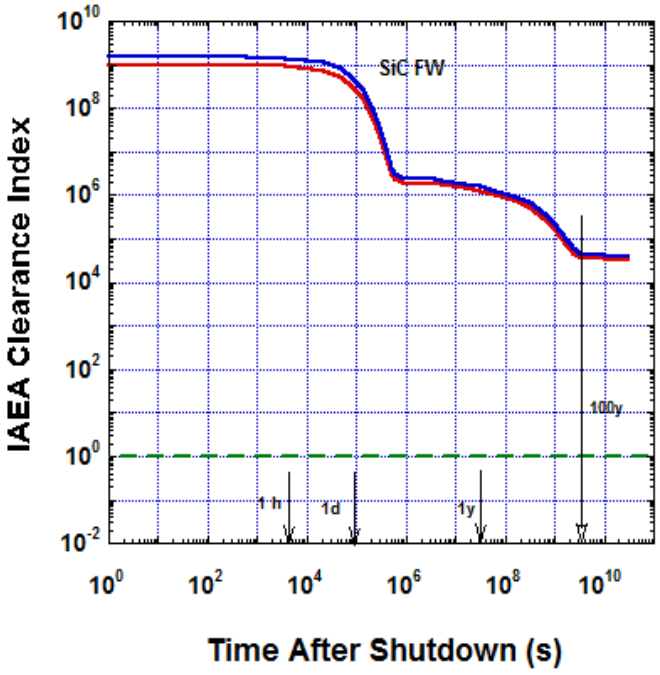


Figure 31. The IAEA clearance for the inboard first wall.

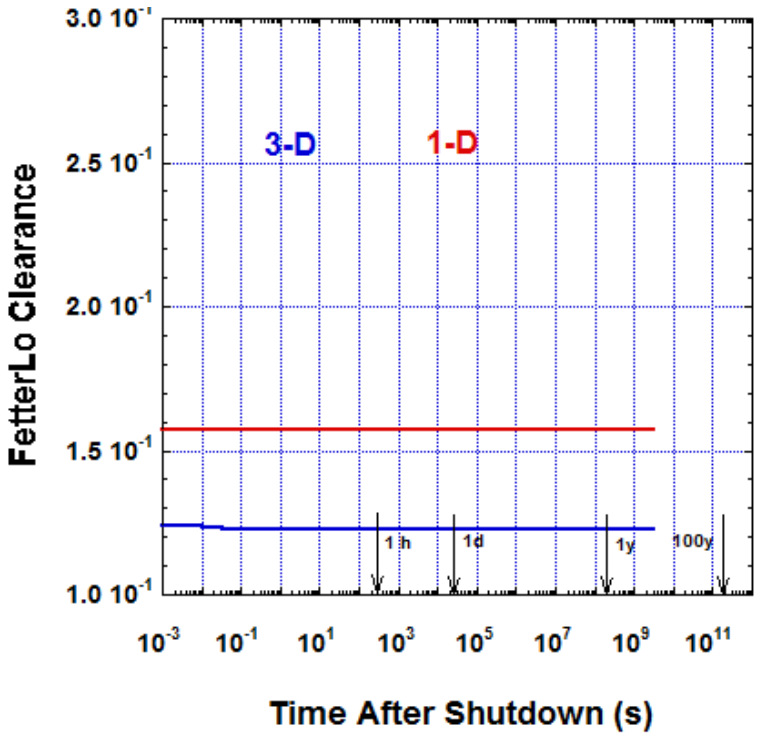


Figure 32. The FetterLo clearance for the inboard first wall.

The inboard vacuum vessel activation is plotted in figures 33-37.

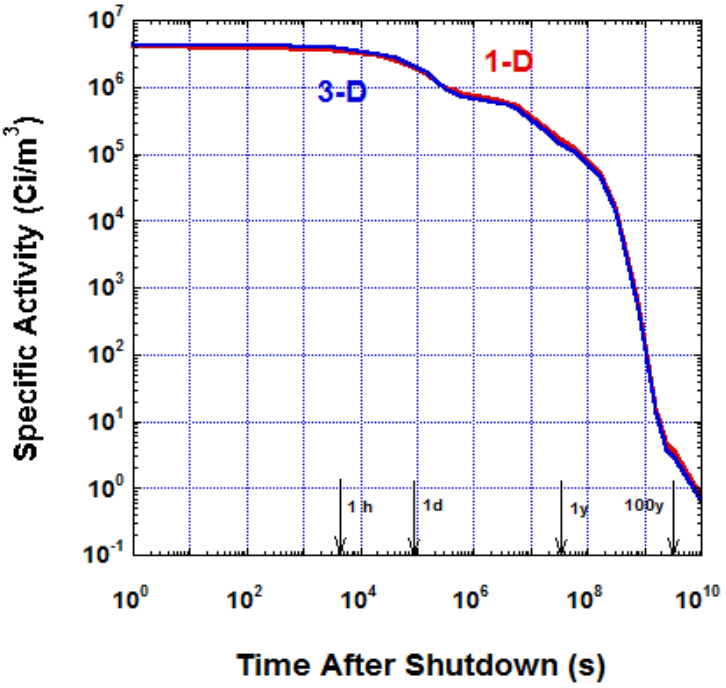


Figure 33. The specific activity for the inboard vacuum vessel.

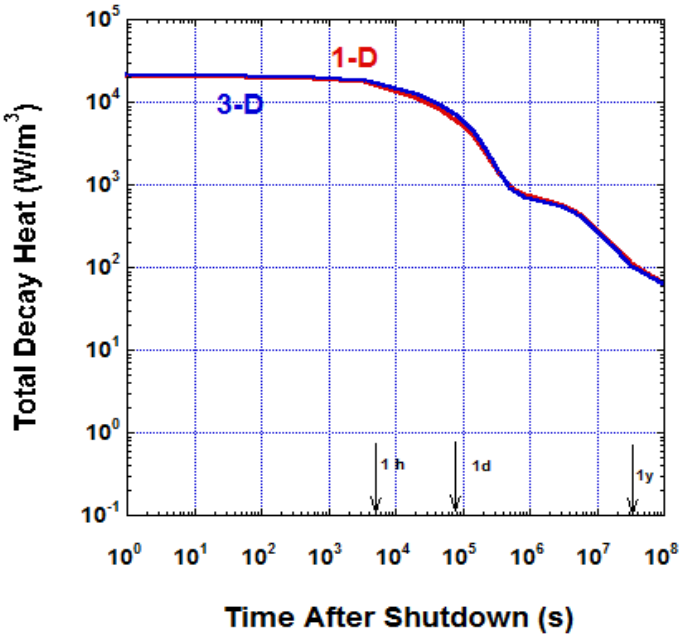


Figure 34. The total decay heat for the inboard vacuum vessel.

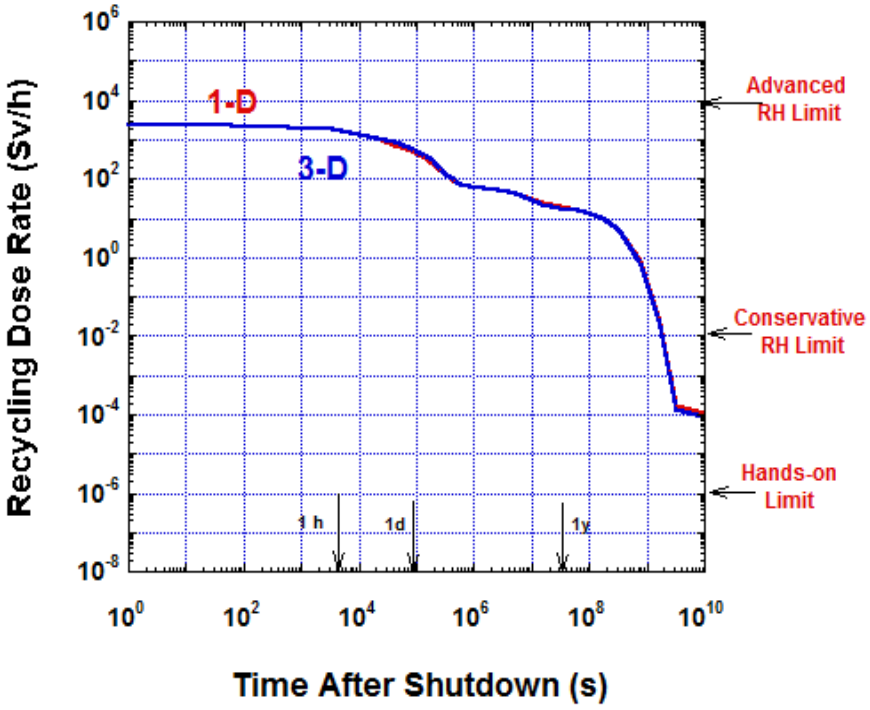


Figure 35. The recycling dose rate for the inboard vacuum vessel.

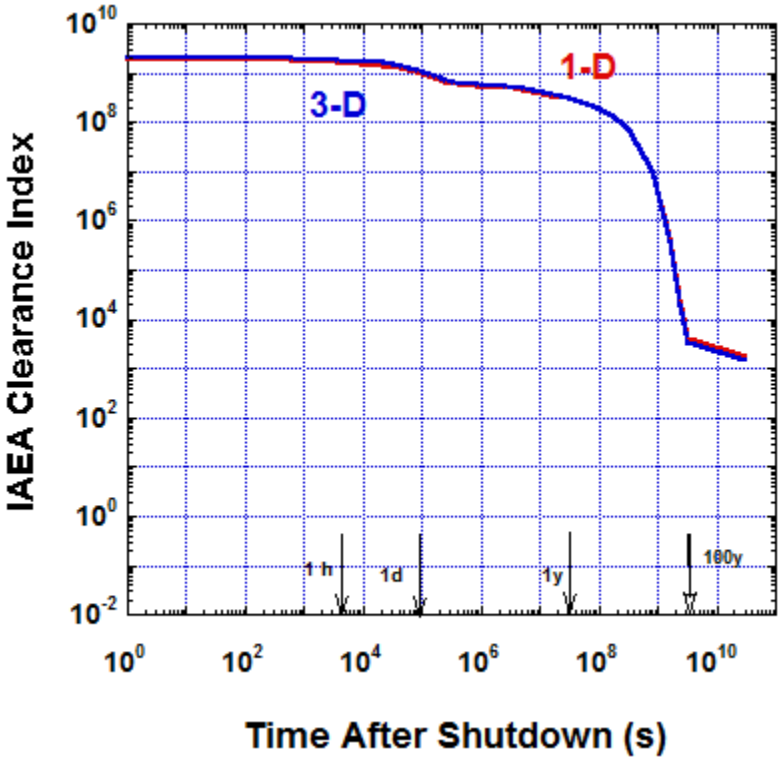


Figure 36. The IAEA clearance for the inboard vacuum vessel.

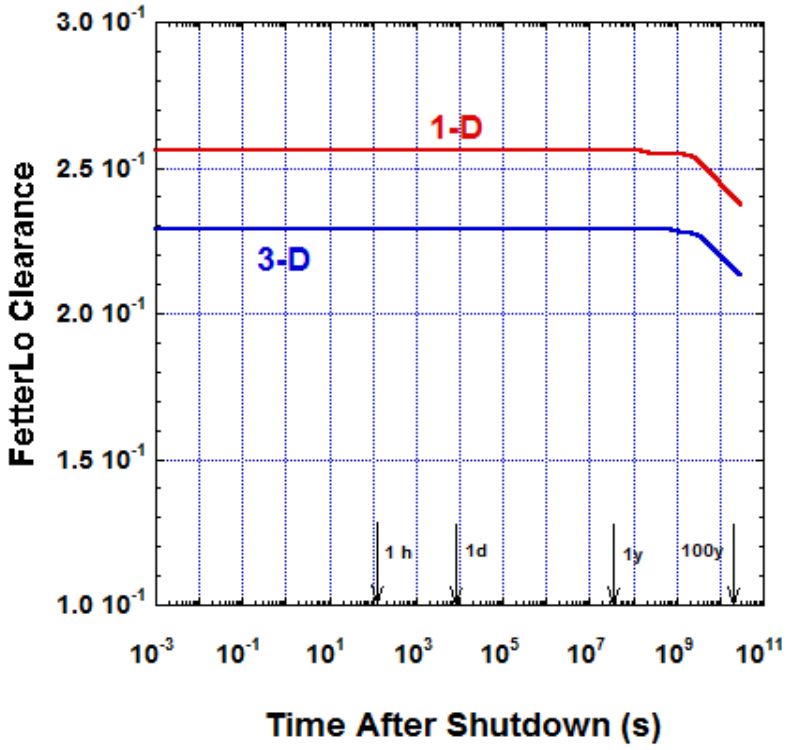


Figure 37. The FetterLo clearance for the inboard vacuum vessel.

The outboard first wall activation is plotted in figures 38-42.

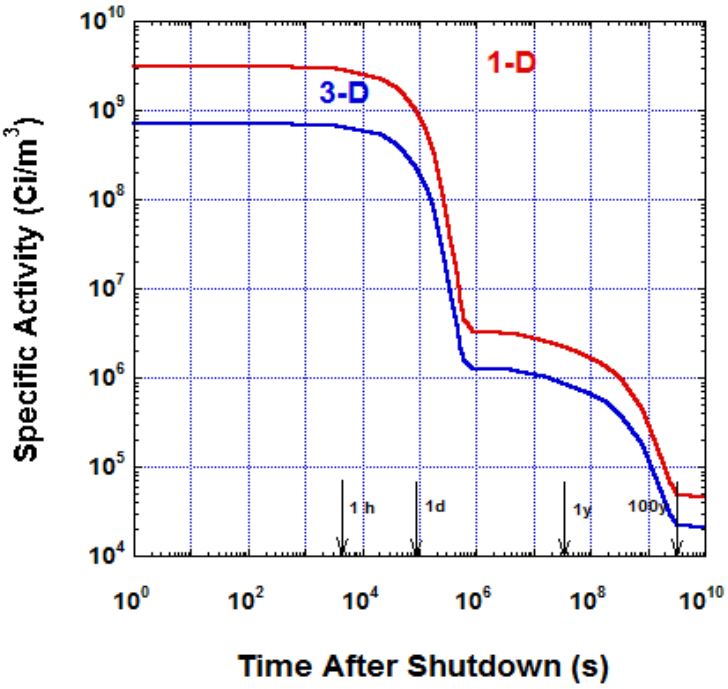


Figure 38. The specific activity for the outboard first wall.

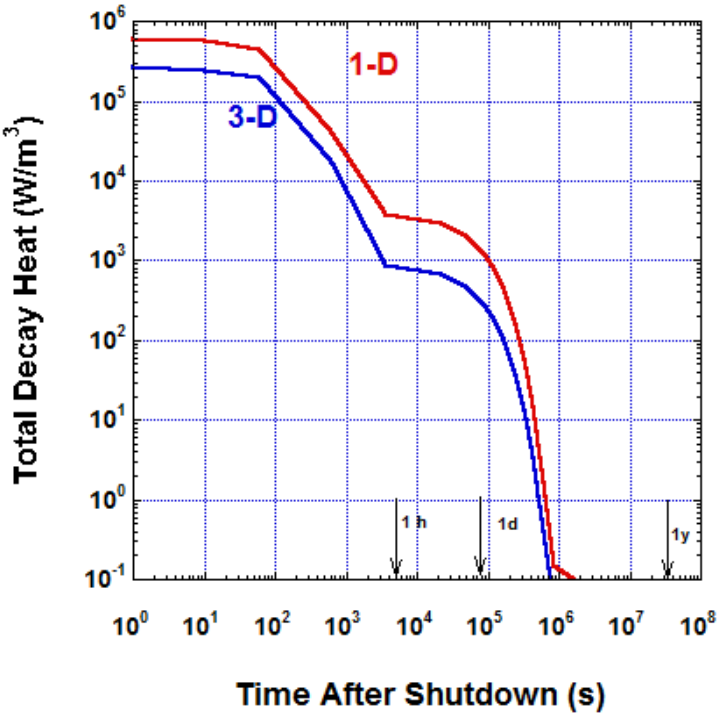


Figure 39. The total decay heat for the outboard first wall.

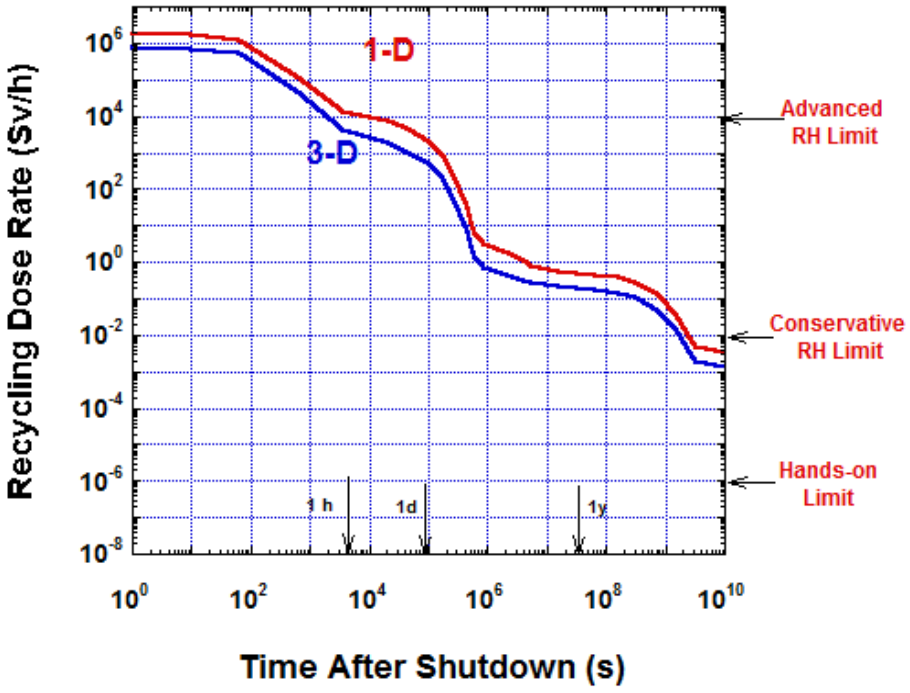


Figure 40. The recycling dose rate for the outboard first wall.

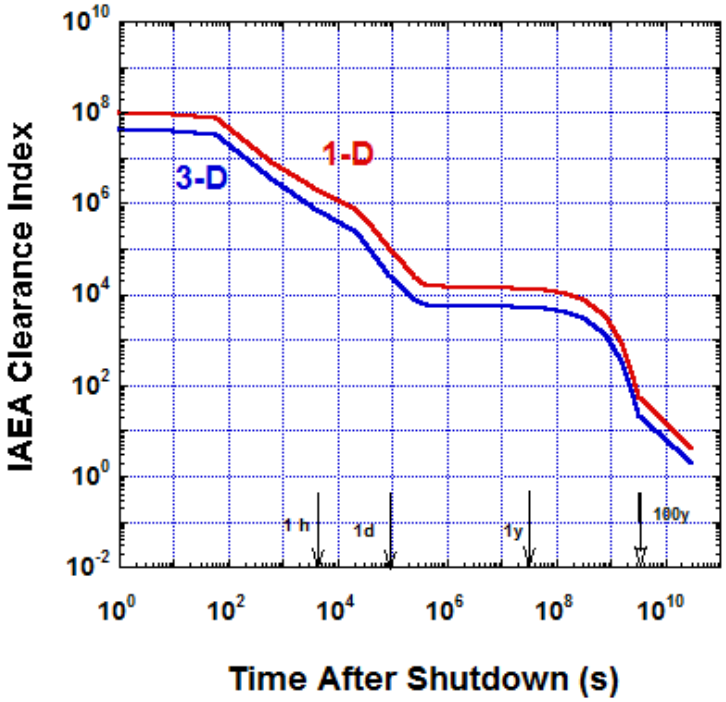


Figure 41. The IAEA clearance for the outboard first wall.

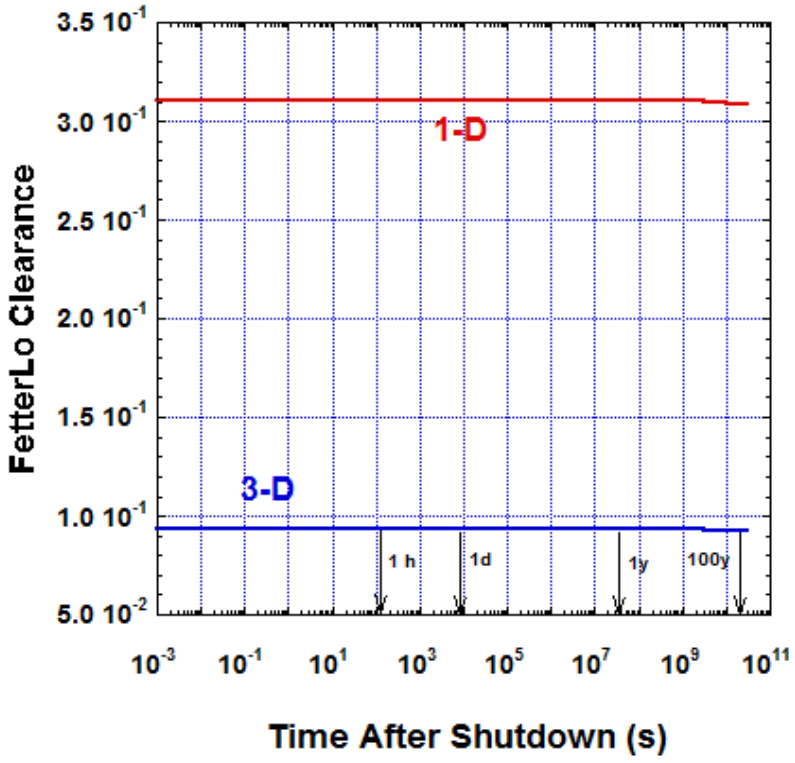


Figure 42. The FetterLo clearance for the outboard first wall.

The outboard vacuum vessel activation is plotted in figures 43-47.

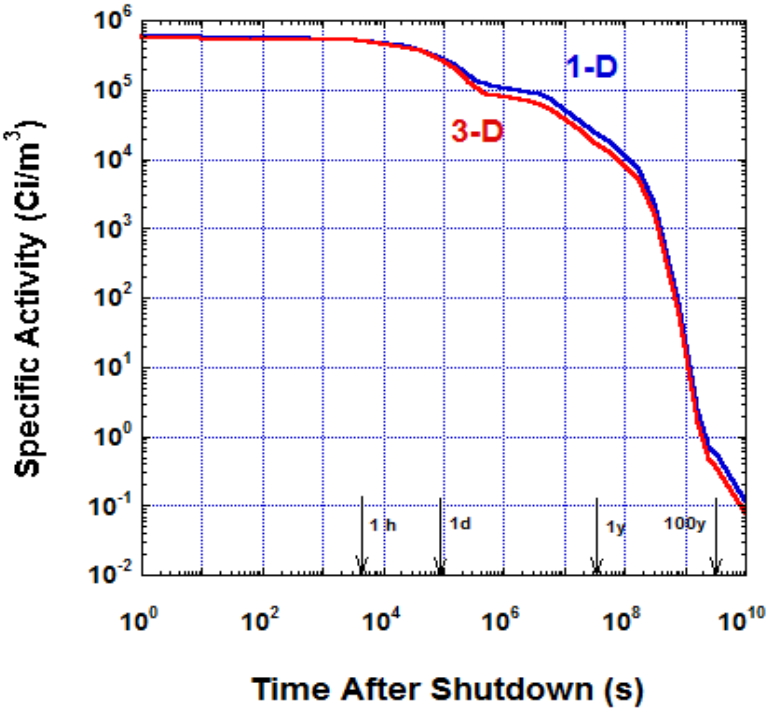


Figure 43. The specific activity for the outboard vacuum vessel.

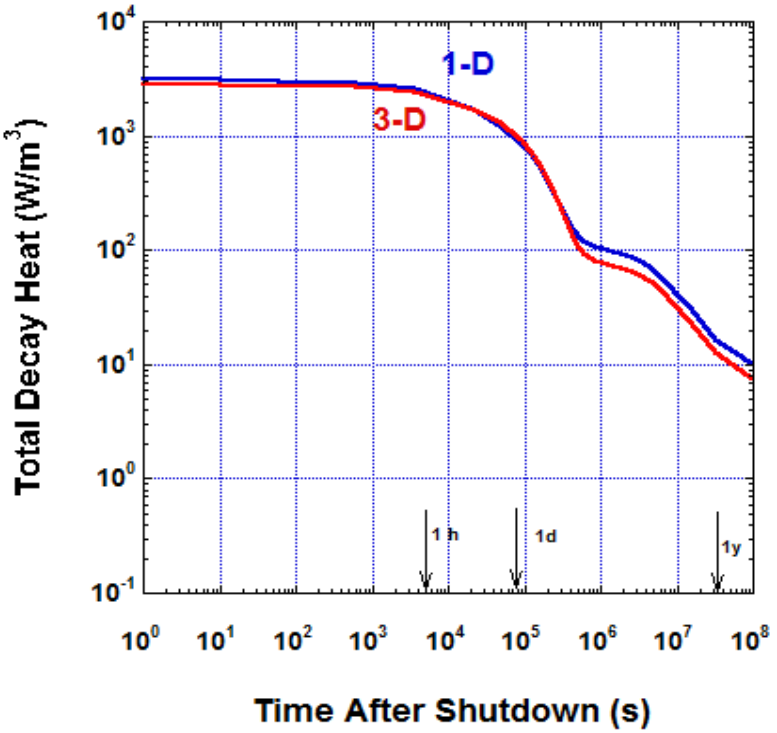


Figure 44. The total decay heat for the outboard vacuum vessel.

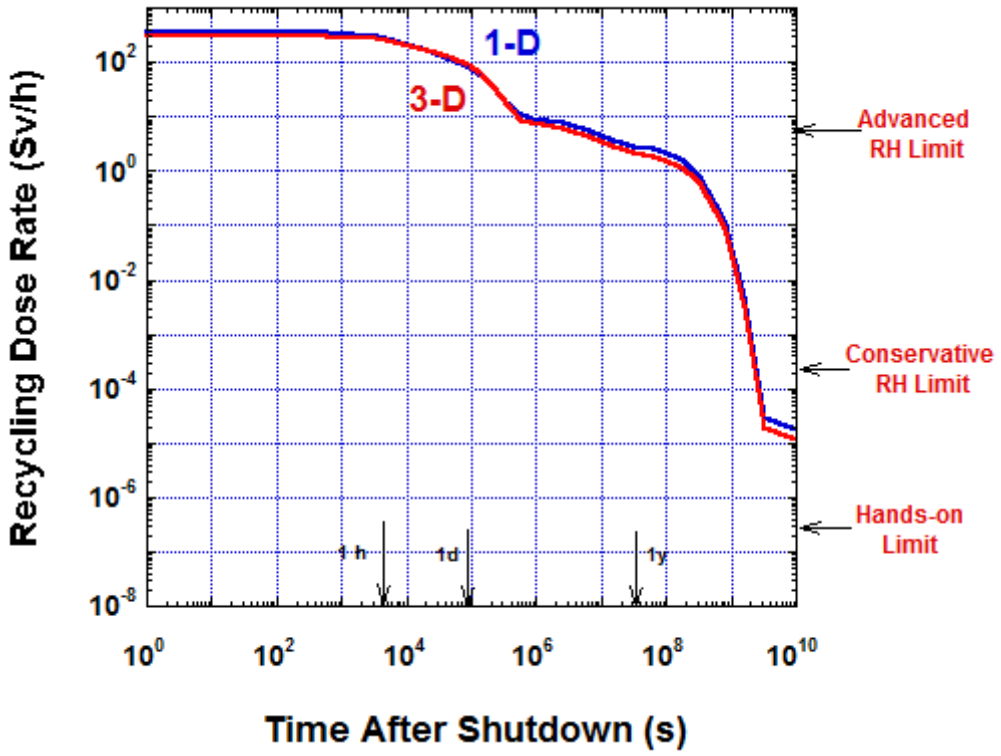


Figure 45. The recycling dose rate for the outboard vacuum vessel.

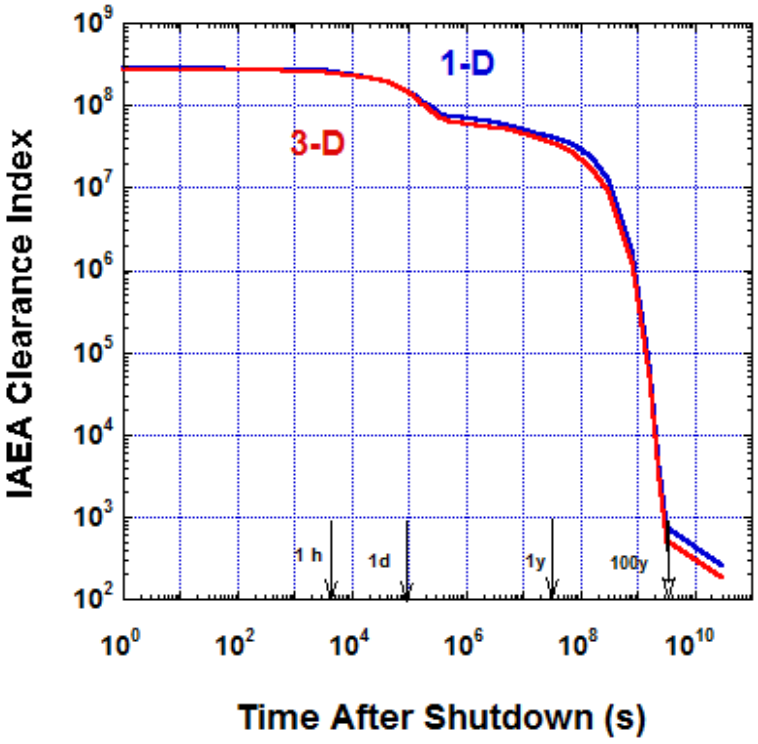


Figure 46. The IAEA clearance for the outboard vacuum vessel.

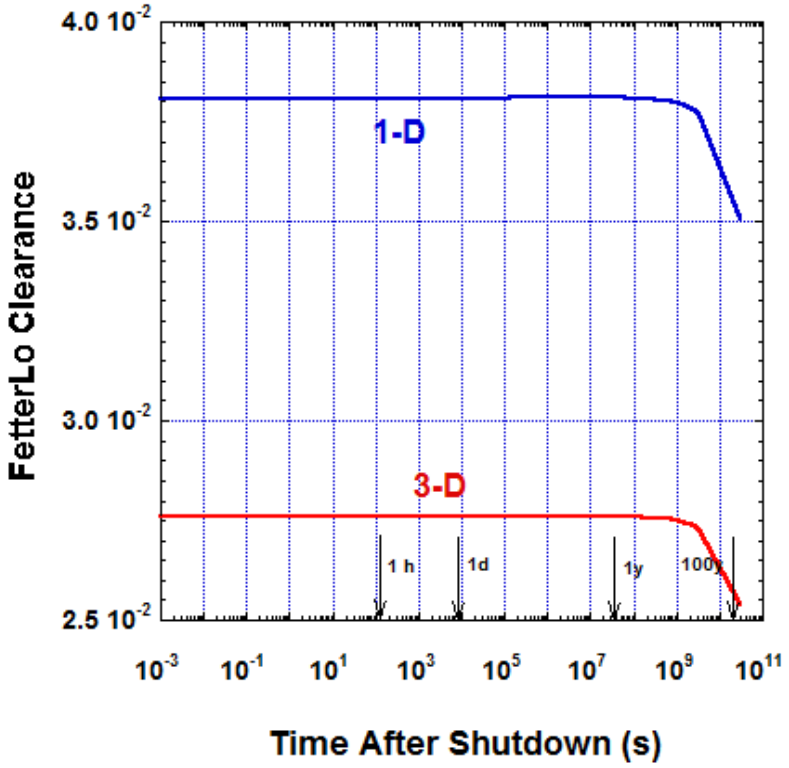


Figure 47. The FetterLo clearance for the outboard vacuum vessel.

5.3.3 ARIES-ACT-1 Model Validation

Taking a real-world model was the final test for the code alterations introduced to the R2S-ACT workflow. In addition to obtaining the specific activity, total decay heat, recycling dose, clearance index, and WDR values to check consistency in the R2S workflow were collected. The first validation of the workflow was performed by obtaining the volumes analytically from the Cubit model that was prepared and comparing them to the volumes entered into ALARA from the ray-firing operations. Table 8 below shows this comparison as well as the discrepancy between the two evaluations. During the ray-tracing of "r2s_step1.py" error is collected for each ray and propagated through to the end. The final ray-tracing error is output to the user, and in the case of this analysis was 2.35%. As a result of this, the discrepancies between the more accurate Cubit volume and R2S-ACT's volume is

expected to be lower than 2.35%. The data fits comfortably under this value. Since the final activation data is obtained for all four cells, this discrepancy is even lessened as can be seen by the final row.

Table 8. The volumes of the cells agree very nicely between Cubit's and R2S-ACT's ray-tracing.

Volumes (cm^3)	Cubit (analytic)	R2S-ACT	% Discrepancy
Cell #1	7076.57	7056.08	0.290%
Cell #2	8457.39	8640.29	-2.163%
Cell #3	8457.49	8399.93	0.681%
Cell #4	8457.49	8465.31	-0.093%
All Cells	8112.24	8140.40	-0.348%

The next validation step involved preparing a native ALARA input to compare to the R2S-ACT workflow ALARA. The f4 tally added to the input deck of MCNP at the start of the workflow now comes into play. The flux in each of the 175 neutron group energy bins was placed into an ALARA fluxin file (from highest energy to lowest energy, the opposite of MCNP). In the same cell order that the fluxes were placed in the fluxin file, the volumes of that respective cell were added in the geometry definition of ALARA. The materials were properly defined and related to these cells, and then the native activation step was performed. Table 9 below shows the comparison between the native and R2S-ACT methods by examining the activity in the four cells at shutdown. The agreement is within an acceptable

range of less than 2.35% that the ray-tracing could be off by. Once again, looking at the total activity makes the discrepancy even less.

Table 9. A comparison between native ALARA and the R2S-ACT workflow reveals only a small discrepancy.

Activity [Ci/cm^3]	Native	R2S-ACT	% Discrepancy
Cell #1	5.82E+01	5.93E+01	-1.99%
Cell #2	5.74E+01	5.84E+01	-1.64%
Cell #3	5.73E+01	5.84E+01	-1.94%
Cell #4	5.74E+01	5.76E+01	-0.47%
All Cells	5.75E+01	5.84E+01	-1.46%

6 Conclusions

This thesis has increased the effectiveness with which activation analyses can be performed by offering additional tools to find accurate results on complex 3D geometries. The first tool assists analysis by reducing human error, automating actions, and allowing visual checkpoints and tests to ensure the accuracy of a model before analysis is performed. The simplicity of the new tool also decrease total time in the analysis workflow. This time adds up when multiple iterations are performed after changes to geometry or new desired data. Indirectly, this leads to better analysis and allows the user to focus on the output data rather than worry with the model preparation. Once in the nuclear analysis workflow, the improved R2S-ACT workflow scripts allow users much needed activation solutions for their systems beyond simple 1D calculations. This greatly reduces user time, since performing a similar analysis on complex models by hand would take many user hours.

The tools introduced and developed within this body of work were evaluated by selecting a real-world nuclear fusion device from the ARIES project and undergoing the nuclear analysis process from start to finish. This evaluation showed the benefits of having a simple workflow to follow, but also revealed the difficulty of balancing complexity and efficiency. Examining complex 3D geometries provides more accurate data on the device, but at the expense of a large increase in computational demands. Ray-tracing analysis compared to time demonstrated the trade-off that must be accepted. Additional, simple tests confirmed the accuracy of the results when large numbers of rays for ray-tracing is permissible.

6. 1 Some Proposed Future Work

While performing this work and developing these tools, there were numerous instances of realizations where more work could be done and areas improved. Time proved a limiter, but hopefully, and eventually, they will be realized. Now that the plugin is developed and with the new capabilities of a completely unique, self-contained ParaView client, it would prove useful to develop a very simple, low-level ParaView client with only the bare essentials for nuclear analysis available. ParaView's size and versatility are nice for some visualization work, but for the purpose of this workflow, prove daunting. The amount of time it would take to develop this doesn't seem too intensive since most components and features of the underlying VTK model would be ignored. Nonetheless, creating a completely unique ParaView client application would require an excellent knowledge of that underlying model, so the learning curve would be high.

As frequently occured, the ParaView plugin could be further developed to have more error checking features to prevent the user from placing metadata on the model that may break the analysis execution, or even worse, go unnoticed. In addition, some work could be spent on increasing the robustness of the plugin for use on extremely large datasets. ParaView internally can handle this by creating instances of the client across many cores, but the plugin's response to this may be less than desirable.

The activation data collected from the R2S-ACT workflow could be improved by developing a powerful unstructured mesh workflow that can be used in the same ways as the current workflow. This could help reduce errors and assist even more in complex models. Another useful addition would be to develop post-processing tools for the output. ALARA provides heaps of output data, but not in a useful form. By spending time on developing post-

processing tools that could output the data in numerous ways based on a user's requests, ALARA's depth of data could be more routinely utilized by researchers.

References

[1] "UW-R2S," CNERG, 2013. [Online]. Available: http://svalinn.github.io/r2s-act/r2s-userguide.html. [Accessed 2013].

[2] M. E. Sawan and M. Z. Youssef, "Three-dimensional neutronics assessment of dual coolant molten salt blankets with comparison to one-dimensional results," *Fusion Engineering and Design*, vol. 81, pp. 505-511, 2006.

[3] L. Mynsberge, A. Jaber and L. El-Guebaly, "Three-Dimensional Evaluation of Tritium Breeding Ratio, Nuclear EHating Distribution, and Neutron Wall Loading Profile for ARIES-ACT-1 (SiC/LiPb) Design," UWFDM-1414, December 2012.

[4] L. El-Guebaly and S. Malang, "Toward the Ultimate Goal of Tritium Self-Sufficiency: Technical Issues and Requirements imposed on ARIES Fusion Power Plants," *Fusion Engineering and Design*, vol. 84, no. 12, pp. 2072-2083, December 2009.

[5] A. Jaber, L. El-Guebaly and L. Mynsberge, "3-D Modeling of Neutron Wall Loading, Tritium Breeding Ratio, and Nuclear Heating Distribution for ARIES-ACT DCLL Design," UWFDM-1408, April 2012.

[6] American Nuclear Society, "Fukushima Daiichi: ANS Committee Report," American

Nuclear Society, La Grange Park, IL, URL: http://fukushima.ans.org/report/Fukushima_report.pdf, June 2012.

[7] L. El-Guebaly, P. Wilson, D. Paige and ARIES and Z-Pinch Teams, "Status of US, EU, and IAEA Clearance Standards and Estimates of Fusion Radwaste Classifcations," UWFDM-1231, December 2004.

[8] L. El-Guebaly, V. Massaut, K. Tobita and L. Cadwallader, "Goals, Challenges, and Successes of Managing Fusion Active Materials," in *8th International Symposium on Fusion Nuclear Technology*, Heidelberg, Germany, 2007.

[9] International Atomic Energy Agency, "Clearance levels for radionuclides in solid materials," IAEA, IAEA-TECDOC-8S5, Vienna, Austria, 1996.

[10] X-5 Monte Carlo Team, "MCNP--A General Monte Carlo N-Particle Transport Code, Version 5," Los Alamose National Laboratory, Los Alamos, NM, April 24, 2003.

[11] P. P. H. Wilson, "ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis," Vols. UWFDM-1070 and UWFDM-1071, 1999.

[12] G. D. Sjaardema and e. al., *Cubit Geometry and Mesh Generation Environment Volume 1: User's Manual*, Sandia National Laboratories, May 1994.

[13] ""DAGMC Users Guide."," University of Wisconsin, 2008. [Online]. Available: http://svalinn.github.io/DAGMC/doc/usersguide/.

[14] P. P. Wilson, T. J. Tautges, J. A. Kraftcheck, B. M. Smith and D. L. Henderson, "Acceleration Techniques for the Direct Use of CAD-Based Geometry in Fusion Neutronics," *Fusion Engineering and Design*, vol. 85, no. 10-12, pp. 1759-1765, Dec. 2010.

[15] "PyTAPS v1.4 Documentation," URL: http://pythonhosted.org/PyTAPS/index.html. [Accessed June 2013].

[16] T. Tautges, J. Kraftcheck, B. Smith and H.-J. Kim, "Mesh-Oriented datABase (MOAB) Version 4.0 User's Guide".

[17] "ParaView Visualization Software User's Guide," URL: http://paraview.org/Wiki/ParaView/.

[18] T. M. Shead, "Customizing ParaView," URL: http://www.vgtc.org/PDF/slides/2008/visweek/tutorial6_shead_customizing.pdf. [Accessed March 2013].

[19] G. Wilson and e. al., "Best Practices for Scientifc Computing," *ArXiv e-prints*, Nov 2012.

[20] UC San Diego, "ARIES Project," URL: http://aries.ucsd.edu/ARIES/.

Appendix A

Shutdown Dose rate Calculational Benchmark

This note described the radiation transport and activation calculations to be carried out by each participant of a calculational benchmark.

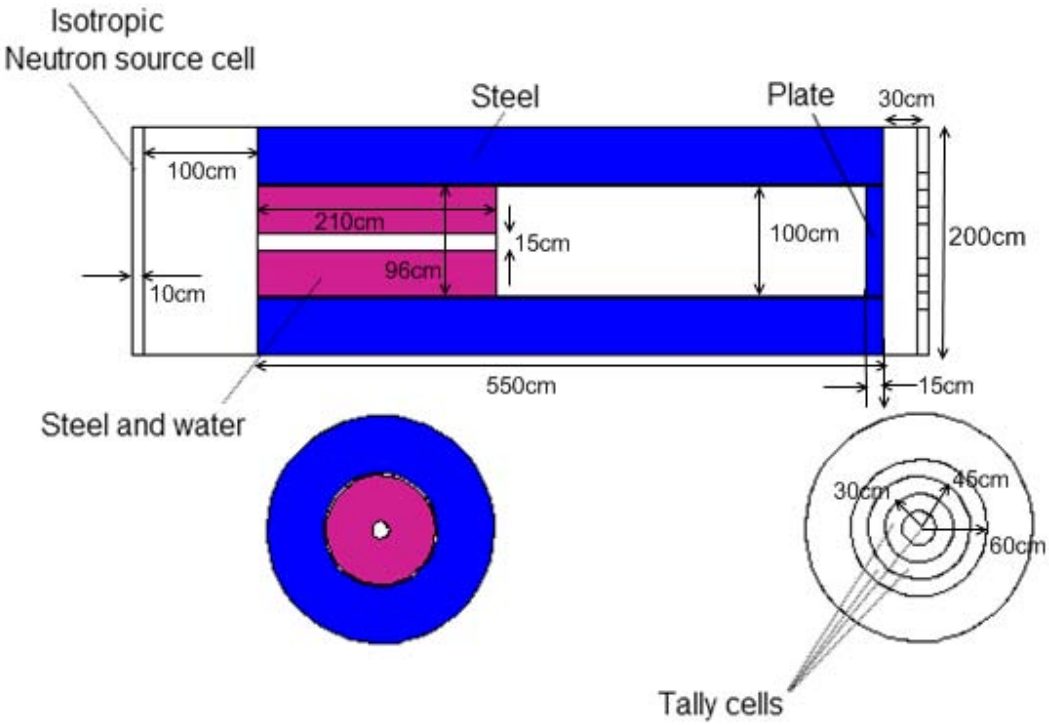


Figure 1: Problem geometry

Geometry

The geometry is cylindrical (see figure 1).

The radius of the outermost cylinder is 100 cm. All radiation is lost beyond this cylinder.

The source cell is 10cm thick. There is then a gap of 100 cm.

The material section consists of an outer steel cylinder is 550 cm long with a 50 cm radius hole through it. The first 210 cm of this hole is nearly filled with a steel and water cylinder which itself has a 7.5 cm radius hole through its centre. There is a 2 cm gap between the steel and water cylinder and the outer steel cylinder.

There is a 15 cm thick steel plate at the end of outer steel cylinder. There is a 2 cm gap between this plate and the outer cylinder.

The tally cells are four concentric circular cells in a void at the rear of the material. These cells begin 30 cm from the back of the steel and are 10 cm thick (i.e. the centre of the tally cells is 35 cm from the plate's rear face). The outer radii of these tally cells are 15, 30, 45, and 60 cm respectively.

Materials

Table 1: Material definitions
Steel & water

Element	Atom fractions
H	1.46E-01
B	4.02E-05
C	8.14E-04
N	2.17E-03
O	7.29E-02
Al	8.04E-04
Si	7.73E-03
P	3.50E-04
S	1.02E-04
K	5.55E-06
Ti	1.36E-03
V	3.41E-05
Cr	1.46E-01
Mn	1.42E-02
Fe	5.03E-01
Co	3.68E-04
Ni	9.06E-02
Cu	2.05E-03
Zr	9.52E-06
Nb	4.67E-05
Mo	1.13E-02
Sn	7.31E-06
Ta	2.40E-05
W	2.36E-06
Pb	1.68E-06
Bi	1.66E-06

Steel and water: density= 6.536 g/cm^3

Steel

Element	Atom fractions
B	5.14E-05
C	1.04E-03
N	2.78E-03
O	6.95E-05
Al	1.03E-03
Si	9.89E-03
P	4.48E-04
S	1.30E-04
K	7.10E-06
Ti	1.74E-03
V	4.36E-05
Cr	1.87E-01
Mn	1.82E-02
Fe	6.44E-01
Co	4.71E-04
Ni	1.16E-01
Cu	2.62E-03
Zr	1.22E-05
Nb	5.98E-05
Mo	1.45E-02
Sn	9.36E-06
Ta	3.07E-05
W	3.02E-06
Pb	2.14E-06
Bi	2.13E-06

Steel: density = 7.93 g/cm^3

Source

The neutron source is an isotropic 14 MeV neutron source emitted uniformly from within the source cell. The neutron production for activation calculations should be as described in table 2

Table 2: Neutron production scenario

Source Strength	Duration	No. of times
1.0714×10^{17}	2 years	1
8.25×10^{17}	10 years	1
0	0.667 years	1
1.6607×10^{18}	1.33 years	1
0	3920 sec	17
2.0×10^{19}	400 sec	
0	3920	4
2.8×10^{19}	400	

Tallies

One tally will be the biological gamma dose (Sv/hr) which results from the neutron activation of the materials 10^6 seconds after cessation of neutron production and averaged over each of the tally cells or as a description of the dose as a function of radius. The dose should be estimated from the gamma flux as prescribed in ITER_D_29PJCT - Recommendations on Computation of Dose from Flux Estimates.

The second tally will be neutron spectra in $n/cm^2/s$, normalised to a source strength of 2.0×10^{19} n/sec. The tally should be made at eight locations: averaged over the front (near neutron source) and rear (away from source) faces of the steel and water cylinder and the steel plate and at points at the centre of each of these faces. The energy bin bounds for the neutron spectra will be:

Lower Bound (MeV)	Upper Bound (MeV)
1.00E-11	1.00E-10
1E-10	1.00E-09
1E-09	1.00E-08
1E-08	1.00E-07
1E-07	1.00E-06
0.000001	1.00E-05
0.00001	1.00E-04
0.0001	1.00E-03
0.001	1.00E-02
0.01	0.1
0.1	1
1	10
10	13
13	14
14	15
15	16
16	20

The third tally will be the gamma ray spectra averaged over each of the tally cells indicated in figure 1 or as a description of the dose as a function of radius at a position in a plane parallel to and 35 cm from the rear face of the plate. The energy bins for the gamma ray spectrum will be:

Lower Bound (MeV)	Upper Bound (MeV)
0	0.1
0.1	0.4
0.4	0.6
0.6	0.8
0.8	1
1	1.22
1.22	1.44
1.44	1.66
1.66	2
2	2.5
2.5	3
3	4
4	5
5	8
8	10
10	And beyond

Appendix B

An example ALARA input file.

```
Specify the geometry type
geometry rectangular

dimension x 0.0
    100   100.0
    100   200.0
end

mixture mix_0
   # material <material_name> <relative density> <volume fraction>
  element     fe      1.0     1.0
end

mixture mix_1
  material
end

mat_loading
    zone_0     void
    zone_1     mix_1
end

# Specify the material, element, and data libraries.
material_lib ARIES_matlib
element_lib ARIES_elelib
data_library alaralib FENDL2

# Specify the cooling times desired for induced activation results.
cooling
   1   s
   1   m
   1   h
   1   d
   1   y
   1   c
end

# Specify the dump file that will hold the solution details.
dump_file   dump.file

# Specify desired ALARA output format
```

```
# Photon source card will generate a photon source file which can be used in DANTSYS.
output zone
      units Ci cm3
      constituent
      specific_activity
# number_density total_heat
     photon_source  FENDL2  phtn_src 21  1e4  1e5  2e5
       4e5  1e6  1.5e6  2e6  2.5e6  3e6  3.5e6
       4e6  4.5e6  5e6  5.5e6  6e6  6.5e6  7e6  7.5e6  8e6
       1e7  1.2e7  1.4e7
end

# Specify the fluxin file and normalization, if needed.
# flux <flux_defn> <flux_file> <norm_value> <group_skip> default
flux flux_1 rtflux 1.0e18 0 default



# Specify the irradiation schedule using âscheduleâpulsehistoryâ
mat_loading

# length of pulse, flux for pulse, pulse schedule, post-pulse time
   5 y   flux_1  pulse1        0 s
  10 h   flux_1 pulse2      100 m
end



pulsehistory    pulse1
# number of pulses, delay time between pulses
   1    0 s
end



pulsehistory    pulse2
  10   1 h
end

# Specify the value at which activation tree branches are cut.
truncation  1e-7

# Specify importance of impurities.
impurity    1e-6    1e-8
```

Appendix C

Benchmark #1 MCNP input file (using R2S-ACT)--Splitting of cells done in fmesh card.

```
Luke Mynsberge: ALARA 3D - Benchmark 1
c
c *************************
c Material Definitions
c *************************
c
c Iron Shield
c 100% Iron
m1 26000  1.0
c
c *************************
c Source Definition
c *************************
mode n
sdef pos=0 0 0.01 x=d1 y=d2 z=0.01 par=1 vec=0 0 1 erg=1.0
si1 0.1 4.9
sp1 0 1
si2 0.1 4.9
sp2 0 1
c
c *************************
c Tally Definitions
c *************************
c
f14:n 1
fc14 Cell flux
c
fmesh24:n geom=xyz origin=0.0 0.0 0.0
        imesh=5.0 iints=1
        jmesh=5.0 jints=1
        kmesh=5.0 10.0 15.0 20.0
        kints=1 1 1 1
        emesh=1.0000E-07 4.1399E-07 5.3158E-07 6.8256E-07 8.7642E-07
            1.1254E-06 1.4450E-06 1.8554E-06 2.3824E-06 3.0590E-06
            3.9279E-06 5.0435E-06 6.4760E-06 8.3153E-06 1.0677E-05
            1.3710E-05 1.7603E-05 2.2603E-05 2.9023E-05 3.7267E-05
            4.7851E-05 6.1442E-05 7.8893E-05 1.0130E-04 1.3007E-04
            1.6702E-04 2.1445E-04 2.7536E-04 3.5380E-04 4.5400E-04
            5.8295E-04 7.4852E-04 9.6112E-04 1.2341E-03 1.5846E-03
            2.0347E-03 2.2487E-03 2.4852E-03 2.6126E-03 2.7465E-03
            3.0354E-03 3.3546E-03 3.7074E-03 4.3107E-03 5.5308E-03
            7.1017E-03 9.1188E-03 1.0595E-02 1.1709E-02 1.5034E-02
```

1.9305E-02 2.1875E-02 2.3579E-02 2.4176E-02 2.4788E-02
2.6058E-02 2.7000E-02 2.8500E-02 3.1828E-02 3.4307E-02
4.0868E-02 4.6309E-02 5.2475E-02 5.6562E-02 6.7379E-02
7.2000E-02 7.9500E-02 8.2500E-02 8.6517E-02 9.8037E-02
1.1109E-01 1.1679E-01 1.2277E-01 1.2907E-01 1.3569E-01
1.4264E-01 1.4996E-01 1.5764E-01 1.6573E-01 1.7422E-01
1.8316E-01 1.9255E-01 2.0242E-01 2.1280E-01 2.2371E-01
2.3518E-01 2.4724E-01 2.7324E-01 2.8725E-01 2.9452E-01
2.9720E-01 2.9850E-01 3.0197E-01 3.3373E-01 3.6883E-01
3.8774E-01 4.0762E-01 4.5049E-01 4.9787E-01 5.2340E-01
5.5023E-01 5.7844E-01 6.0810E-01 6.3928E-01 6.7206E-01
7.0651E-01 7.4274E-01 7.8082E-01 8.2085E-01 8.6294E-01
9.0718E-01 9.6164E-01 1.0026E+00 1.1108E+00 1.1648E+00
1.2246E+00 1.2873E+00 1.3534E+00 1.4227E+00 1.4957E+00
1.5724E+00 1.6530E+00 1.7377E+00 1.8268E+00 1.9205E+00
2.0190E+00 2.1225E+00 2.2313E+00 2.3069E+00 2.3457E+00
2.3653E+00 2.3852E+00 2.4660E+00 2.5924E+00 2.7253E+00
2.8650E+00 3.0119E+00 3.1664E+00 3.3287E+00 3.6788E+00
4.0657E+00 4.4933E+00 4.7237E+00 4.9659E+00 5.2205E+00
5.4881E+00 5.7695E+00 6.0653E+00 6.3763E+00 6.5924E+00
6.7032E+00 7.0469E+00 7.4082E+00 7.7880E+00 8.1873E+00
8.6071E+00 9.0484E+00 9.5123E+00 1.0000E+01 1.0513E+01
1.1052E+01 1.1618E+01 1.2214E+01 1.2523E+01 1.2840E+01
1.3499E+01 1.3840E+01 1.4191E+01 1.4550E+01 1.4918E+01
1.5683E+01 1.6487E+01 1.6905E+01 1.7333E+01 1.9640E+01

```
c
ctme 10
lost 500 500
```
Benchmark #1 MCNP input file (not using R2S-ACT)--Splitting of cells done in model.
```
Luke Mynsberge: ALARA 3D - Benchmark 1
c
c **************************
c Material Definitions
c **************************
c
c Iron Shield
c 100% Iron
m1 26000  1.0
c
c **************************
c Source Definition
c **************************
mode n
sdef pos=0 0 0.01 x=d1 y=d2 z=0.01 par=1 vec=0 0 1 erg=1.0
si1 0.1 4.9
```

```
sp1 0 1
si2 0.1 4.9
sp2 0 1
c
c ************************
c Tally Definitions
c ************************
c
f14:n 1 2 3 4 T
fc14 Cell fluxes for all areas
e14         1.0000E-07 4.1399E-07 5.3158E-07 6.8256E-07 8.7642E-07
            1.1254E-06 1.4450E-06 1.8554E-06 2.3824E-06 3.0590E-06
            3.9279E-06 5.0435E-06 6.4760E-06 8.3153E-06 1.0677E-05
            1.3710E-05 1.7603E-05 2.2603E-05 2.9023E-05 3.7267E-05
            4.7851E-05 6.1442E-05 7.8893E-05 1.0130E-04 1.3007E-04
            1.6702E-04 2.1445E-04 2.7536E-04 3.5380E-04 4.5400E-04
            5.8295E-04 7.4852E-04 9.6112E-04 1.2341E-03 1.5846E-03
            2.0347E-03 2.2487E-03 2.4852E-03 2.6126E-03 2.7465E-03
            3.0354E-03 3.3546E-03 3.7074E-03 4.3107E-03 5.5308E-03
            7.1017E-03 9.1188E-03 1.0595E-02 1.1709E-02 1.5034E-02
            1.9305E-02 2.1875E-02 2.3579E-02 2.4176E-02 2.4788E-02
            2.6058E-02 2.7000E-02 2.8500E-02 3.1828E-02 3.4307E-02
            4.0868E-02 4.6309E-02 5.2475E-02 5.6562E-02 6.7379E-02
            7.2000E-02 7.9500E-02 8.2500E-02 8.6517E-02 9.8037E-02
            1.1109E-01 1.1679E-01 1.2277E-01 1.2907E-01 1.3569E-01
            1.4264E-01 1.4996E-01 1.5764E-01 1.6573E-01 1.7422E-01
            1.8316E-01 1.9255E-01 2.0242E-01 2.1280E-01 2.2371E-01
            2.3518E-01 2.4724E-01 2.7324E-01 2.8725E-01 2.9452E-01
            2.9720E-01 2.9850E-01 3.0197E-01 3.3373E-01 3.6883E-01
            3.8774E-01 4.0762E-01 4.5049E-01 4.9787E-01 5.2340E-01
            5.5023E-01 5.7844E-01 6.0810E-01 6.3928E-01 6.7206E-01
            7.0651E-01 7.4274E-01 7.8082E-01 8.2085E-01 8.6294E-01
            9.0718E-01 9.6164E-01 1.0026E+00 1.1108E+00 1.1648E+00
            1.2246E+00 1.2873E+00 1.3534E+00 1.4227E+00 1.4957E+00
            1.5724E+00 1.6530E+00 1.7377E+00 1.8268E+00 1.9205E+00
            2.0190E+00 2.1225E+00 2.2313E+00 2.3069E+00 2.3457E+00
            2.3653E+00 2.3852E+00 2.4660E+00 2.5924E+00 2.7253E+00
            2.8650E+00 3.0119E+00 3.1664E+00 3.3287E+00 3.6788E+00
            4.0657E+00 4.4933E+00 4.7237E+00 4.9659E+00 5.2205E+00
            5.4881E+00 5.7695E+00 6.0653E+00 6.3763E+00 6.5924E+00
            6.7032E+00 7.0469E+00 7.4082E+00 7.7880E+00 8.1873E+00
            8.6071E+00 9.0484E+00 9.5123E+00 1.0000E+01 1.0513E+01
            1.1052E+01 1.1618E+01 1.2214E+01 1.2523E+01 1.2840E+01
            1.3499E+01 1.3840E+01 1.4191E+01 1.4550E+01 1.4918E+01
            1.5683E+01 1.6487E+01 1.6905E+01 1.7333E+01 1.9640E+01
```

```
c
ctme 10
lost 500 500
Benchmark #1 ALARA input file

geometry rectangular

volume
        125.0   zone_0
        125.0   zone_1
        125.0   zone_2
        125.0   zone_3
end

mixture         mix_0
        material        mat1_rho-7.874          1       1.0
end

mixture pseudo_void
        material        pseudo_void   1       1.0
end

mat_loading
        zone_0 mix_0
        zone_1 mix_0
        zone_2 mix_0
        zone_3 mix_0
end


# ALARA Snippet file: See ALARA user manual for additional syntax information

# Specify the material, element, and data libraries.
material_lib    matlib
element_lib     elelib
data_library alaralib FENDL2

# Specify the cooling times for which activation results are desired
cooling
   1   s
   1   m
   1   h
   1   d
   1   y
end
```

```
# Specify the fluxin file and normalization, if needed
#    flux name  flux file     norm   shift   unused
flux flux_1     alara_fluxin  1.0e6    0     default

# Specify the irradiation schedule using "schedule" and "pulsehistory"
# Syntax is found in the ALARA user manual
schedule    total
   10.0 y   flux_1  pulse_once    0  s
end

# A pulse history is applied to each flux in the schedule. Pulse syntax is:
#  pulsehistory    pulse_name
#       num_pulses   delay_between_pulses
#  end
pulsehistory    pulse_once
   1   5.0 s
end
pulsehistory    pulse_thrice_wait_some
   3   0.1 y
end

# Specify desired ALARA output (e.g. constituant, specific activity).
# Photon source card must be present to produce the pthn_src file for step2.
output interval
      units Ci cm3
      constituent
      specific_activity
      # photon_source  FENDL2  phtn_src 42  1e4  2e4
      #   3e4  4.5e4  6e4  7e4  7.5e4  1e5  1.5e5  2e5  3e5
      #   4e5  4.5e5  5.1e5  5.12e5  6e5  7e5  8e5  1e6  1.33e6
      #   1.34e6  1.5e6  1.66e6  2e6  2.5e6  3e6  3.5e6
      #   4e6  4.5e6  5e6  5.5e6  6e6  6.5e6  7e6  7.5e6  8e6
      #   1e7  1.2e7  1.4e7  2e7  3e7  5e7
end

#other parameters
truncation  1e-12
impurity    5e-6    1e-3
dump_file   dump.file
```

Some ALARA output for Benchmark #1

```
Zone #2: zone_1
          Relative Volume: 500
          Containing mixture: mix_1
```

```
Constituent: mat1_rho-7.874
        Volume Fraction: 1    Relative Volume: 500
Specific Activity [Ci/cm3]
isotope shutdown        1 s        1 m        1 h        1 d        1 y
================================================================================
                                Looking for data for 250550
                                Setting NuclearData members.
                                Looking for data for 260540
                                Setting NuclearData members.
                                Looking for data for 260550
                                Setting NuclearData members.
fe-55   1.7886e-12  1.7886e-12  1.7886e-12  1.7886e-12  1.7874e-12  1.3894e-12
                                Looking for data for 260560
                                Setting NuclearData members.
                                Looking for data for 260570
                                Setting NuclearData members.
                                Looking for data for 260580
                                Setting NuclearData members.
                                Looking for data for 260590
                                Setting NuclearData members.
fe-59   7.5280e-14  7.5280e-14  7.5279e-14  7.5231e-14  7.4116e-14  2.5967e-16
                                Looking for data for 270590
                                Setting NuclearData members.
================================================================================
total    1.8639e-12  1.8639e-12  1.8639e-12  1.8638e-12  1.8615e-12  1.3897e-12
** Zone totals are the same as those of the single constituent.
```