DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

SANDIA REPORT
SAND2025-11634
Printed September 2025

Sandia National Laboratories

MPACT Safeguards Modeling: FY25 Update

Nathan Shoman, Philip Honnold, Ramon Pulido, Tania Rivas, Anna Taconi

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

Sandia National Laboratories develops and maintains several open-source software packages to support material accountancy analyses. This includes the Material Accountancy Performance Indicator Toolkit (MAPIT), the Fissile Facility Flow Modeler (F3M) and the Separation and Safeguards Performance Model Library (SSPM-L).

MAPIT is responsible for performing statistical safeguards analyses on bulk and itemized data from nuclear fuel cycle facilities and can operate on real or synthetic data. MAPIT is the only open-source software for such analyses. F3M is a library of modules, built in MATLAB Simulink, that contain pre made blocks to represent different generic fuel cycle processes. These blocks can be used together in a modular fashion to represent and simulate nuclear fuel cycle processes with the goal of improving facility-level accountancy during the design phase. F3M is also an open-source library. Finally, the SSPM-L library is a series of completed models built from F3M. The library includes facility models such as a generic PUREX facility and a fuel fabrication facility. The SSPM-L library is not open source, but is available to collaborators with a relevant use case.

These tools include modeling and simulation pipelines to simulate nuclear fuel cycle facilities and the underlying software needed to simulate measurement uncertainty and perform statistical analyses. Together, these tools can perform end-to-end nuclear material accountancy analyses. This report documents the various improvements made to these tools in FY25. Specifically, we added new statistical test, new statistical modeling capabilities, new fuel cycle facility models, and launched a new open-source model component library.

This page intentionally left blank.

ACKNOWLEDGMENTS

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

This work is supported by the DOE/NE Office of Materials and Chemical Technologies (NE-43) - Materials Protection, Accounting, and Control Technologies (MPACT) program.

This page intentionally left blank.

CONTENTS

This page intentionally left blank.

LIST OF FIGURES

This page intentionally left blank.

1. INTRODUCTION

Nuclear fuel cycle facilities face significant economic challenges in meeting safeguards, security, and environmental regulations. Prior experience has demonstrated that retrofits to meet regulatory requirements can be costly and time consuming. It is therefore advantageous to consider these goals during the facility design phase. Sandia National Labs develops and maintains several software packages that support the design and implementation of material control and accountancy (MC&A). This includes the Material Accountancy Performance Indicator Toolkit (MAPIT), Fissile Facility Flow Modeler (F3M), and the Separation and Safeguards Performance Model Library (SSPM-L). These tools are a mix of open-source and closed-source software tools, that together, can help answer "what-if" questions for accountancy of bulk nuclear facilities. Figure 1-1 below shows how the different tools work together to support accountancy analyses.

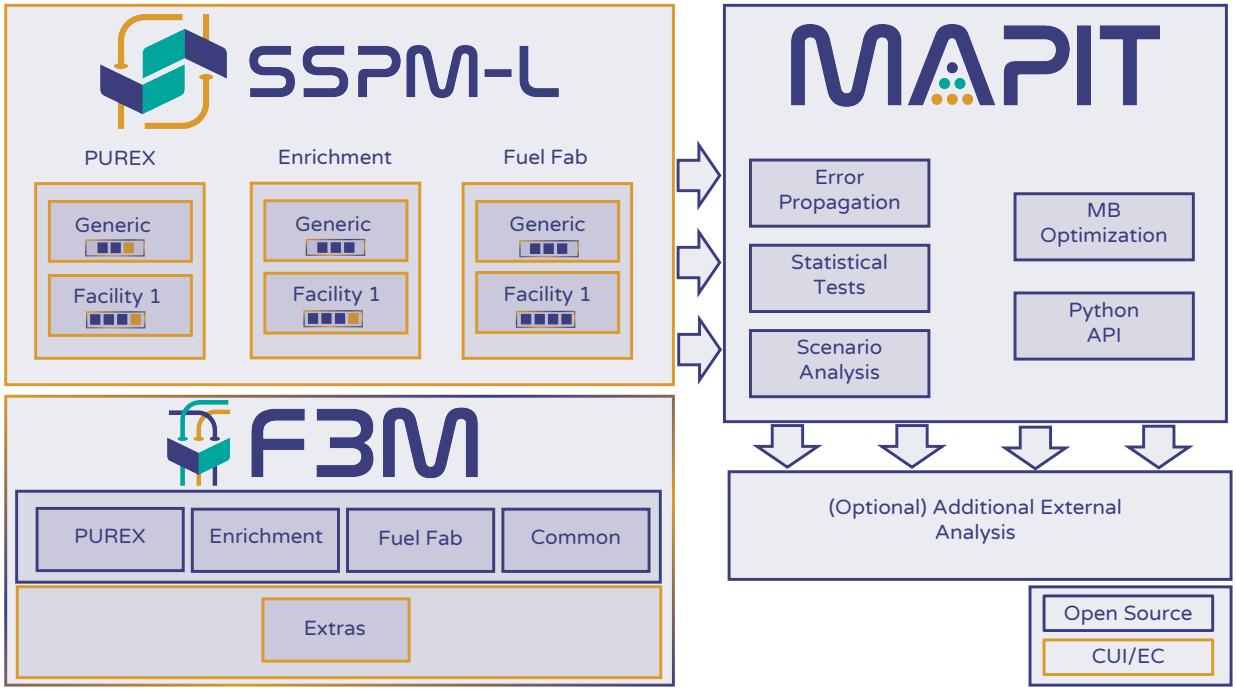

Figure 1-1: MAPIT, F3M, and SSPM-L architecture overview. Small colored blocks for each flow sheet in the SSPM-L reference the different libraries from F3M that are used.

A brief description of each tool is as follows:

- **MAPIT:** MAPIT (Material Accountancy Performance Indicator Toolkit) is a Python package designed to aid in safeguards analysis of bulk materials. The inherent flexibility is designed to allow safeguards practitioners ask the "what if?" questions while providing transparency into commonly employed statistical tests. MAPIT provides both a graphical user interface (GUI) and an application program interface (API). The API can be used with other popular Python libraries to extend functionality and integrate with other analytical workflows.
  - MAPIT code
  - MAPIT documentation
    - Complete function and theory documentation for MAPIT is available here
  - MAPIT examples and notional data
- **F3M:** The Fissile Facility Flow Modeler (F3M) is an open source series of modules intended to facilitate the development of nuclear facility models in MATLAB Simulink for safeguards research. F3M consists of several libraries divided into facility type. This library is targeted towards modeling material flows and operations without the inclusion of specific operational details. For example, one generic block is the flow batcher. This block exposes parameters that describe the behavior of a batching operation such as batch target size and residual size, but does not describe dimensions or mechanics for how this process might be carried out. The most relevant details for safeguards research are material flows, quantities, and timings, which is the primary focus of this library.
  - F3M code
  - F3M documentation
    - Complete function and theory documentation for F3M is available here
- **SSPM-L:** The Separation and Safeguards Performance Model Library (SSPM-L) is an evolution of the SSPM framework, which started development nearly 15 years ago. The SSPM-L is a series of generic models built on the F3M component library and readily integrated with MAPIT. Each model in the library is tied to references where possible with explicit ties between the model's operating parameters and literature references. These parameters are documented in a model card that accompanies each model. The intent of the SSPM-L model library is to either perform generic accountancy analyses or to serve as a base for customized, facility-specific models. Since SSPM-L is based on F3M, any bug fixes or improvements made in one model can be easily propagated to the others. Many of the SSPM-L models are CUI and are not available on an open repository like F3M and MAPIT. Users with a relevant use case can obtain SSPM-L without extensive paperwork (no government use notice required), but must still protect the models as they would any other CUI information.

This report is focused on updates and improvements made in FY25 and is not intended to serve as a comprehensive overview. Much of the underlying software documentation for MAPIT and F3M is

hosted on our GitHub repos [1,2] whereas SSPM-L documentation is in model cards that accompany the model file.

[1] MAPIT repo
[2] F3M repo

This page intentionally left blank.

2. MAPIT

MAPIT is the an open source material accountancy toolkit supported by MPACT. Efforts in FY25 focused on general improvements and responses to users filing bug reports or reporting usability issues.

2.1. Systematic error calibration

MAPIT uses a multiplicative error model [1, 2] to apply simulated measurement error to user supplied data. The model is described in Equation 2.1.

$$\widetilde{x_t} = x_t(1 + r_t + s_t) \tag{2.1}$$

Where:

- $\widetilde{x_t}$ is the observed value (i.e., what is actually measured) at time t
- x is the true, but unknowable value at time t
- r_t is relative random error of x
  - Specifically r_t is a random variate of the distribution $\mathcal{N}(0, \delta_r^2)$ where δ_r^2 is the random relative standard deviation.
- s_t is the relative systematic error of x
  - Specifically, s_t is a random variate of the distribution $\mathcal{N}(0, \delta_s^2)$ where δ_s^2 is the systematic relative standard deviation.

Here, random error refers to sources of error that can be reduced through repeated measurements of the same item. Systematic errors refers to short-term biases that are generally irreducible. These systematic biases can arise from a variety of sources such as calibration errors. Regardless of the measurement type, errors are characterized by a mean zero normal distribution with non-zero standard deviation. The distributions characterizing the random and systematic error can vary based on a variety of factors such as measurement type, measurement system, and even the specific isotope measured.

Systematic errors behave as a bias. Consequently, the systematic variate, S_t, from the multiplicative model described above (Equation 2.1), is not updated at every timestep. This contrasts with the random variate which *is* updated at each time step. The systematic variate is held constant and only updated on a periodic basis that corresponds to a specified calibration period. The specifics of the calibration period are measurement system specific.

Historically, MAPIT hasn't supported sensor recalibration. Sensor recalibration would need to be modeled as systematic errors that are updated at a regular periodicity corresponding to the recalibration. This feature was added in FY25. Now users can select a sensor recalibration period. A new systematic variate is drawn and applied at each interval specified, instead of limiting users to one applied systematic variate over the entire run. This is integrated into the GUI, as shown in Figure 2-1.

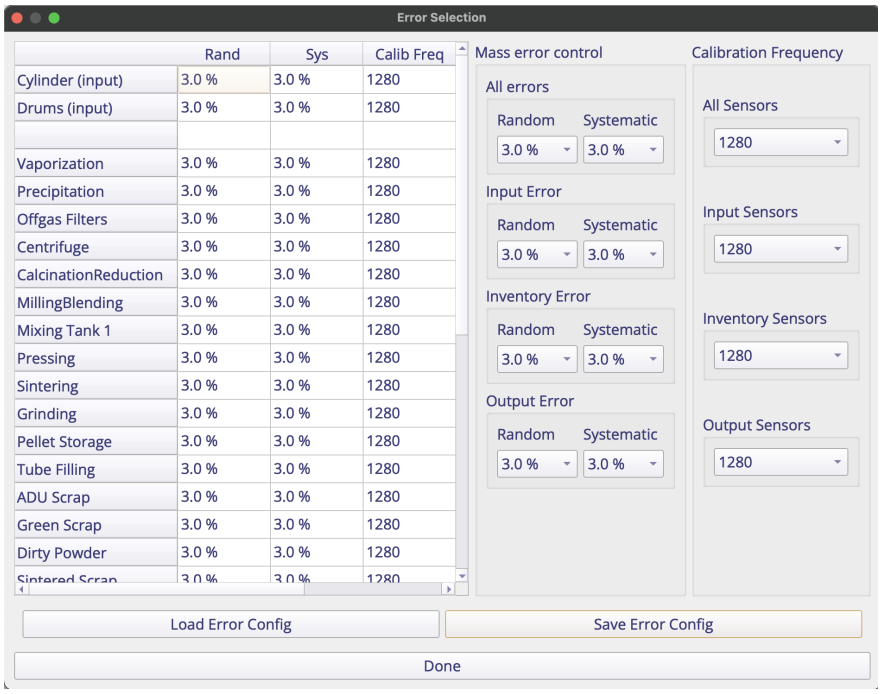

Figure 2-1: Recalibration GUI dialog. Per-sensor calibration frequencies can be set here through the drop downs or the edit box.

This functionality is also incorporated in the API, as shown below. The new arguments `inputCalibrationPeriod`, `inventoryCalibrationPeriod`, and `outputCalibrationPeriod` take a list of values that should be equal in length to the correspond list on data inputs. The list of calibration periods are matched, in order, to the data series for each measurement type. A value of `None` indicated that there is no recalibration and the same systematic error should be used for the duration of the data set. The code block below shows the new arguments for setting up the material balance area object.

MBSetup.py

```
from MAPIT.core import StatsProcessor
MB1 = StatsProcessor.MBArea(rawInput = indat,
        rawInventory = invdat,
        rawOutput = outdat,
        rawInputTimes = inTdat,
        rawInventoryTimes = invTdat,
        rawOutputTimes = outTdat,
        inputErrorMatrix = inputErrorMatrix,
        inventoryErrorMatrix = inventoryErrorMatrix,
        outputErrorMatrix = outputErrorMatrix,
        mbaTime = mbaTime,
        iterations = 50,
        inputCalibrationPeriod=[None]*len(indat),
        inventoryCalibrationPeriod=[None]*len(invdat),
        outputCalibrationPeriod=[None]*len(outdat),
        inputTypes=['continuous']*len(indat),
        outputTypes=['continuous']*len(outdat)
    )
```

2.2. Discrete item support

Historically, MAPIT has considered all material balance calculation terms to be "continuous" in that they are *rates* (e.g., kg/hr, L/hr, etc.). Consequently, MAPIT assumed that all terms would require numerical integration as material balance terms must have mass units. Recent improvements to F3M, namely integrating SimEvents for discrete entity tracking, has led to the need for both continuous and discrete mass handling within MAPIT. In FY25, we added the capability for MAPIT to handle discrete terms.

The key difference between continuous and discrete terms is that the latter doesn't need to be integrated. We created complementary blocks in F3M for recording discrete data streams that only record data when entities are present. As such, we can then simply sum all the terms in a material balance period without the need for integration. We adapted all existing statistical tests to handle both material types. For example, the input terms for the standard error of the inventory difference (SEID) calculation are shown below. Note the option for continuous or discrete data.

StatsTests.py

```
if inputTypes[j] == 'continuous':
  AFTS = AuxFunctions.trapSum(logicalInterval,processedInputTimes[j],inputAppliedError[j])
elif inputTypes[j] == 'discrete':
  AFTS = inputAppliedError[j][:, logicalInterval].sum(axis=1)
else:
  raise Exception("inputTypes[j] is not 'continuous' or 'discrete'")
```

Functionality for handling different item types has been integrated into both the GUI and the API. In the GUI, when bringing your own data, a simple popup prompts users to specify the type of measurement for inputs and outputs [3]. The API implements this by requesting a list of strings for the input (`inputTypes`) and outputs (`outputTypes`) that specify `continuous` or `discrete` for each data stream.

2.3. GEMUF Test integration

We implemented the GEMUF (geschatzter, or estimated, MUF) [3] in FY25. We previously implemented SITMUF, which attempts to develop a sequence of *residuals* wherein the MUF sequence is converted to a standardized sequence and monitored for trend changes. In contrast, GEMUF attempts instead to develop a distance-based metric to detect anomalies via MUF and the covariance matrix (but not necessarily a whitened residual).

Recall that the **muf** sequence [1, 2, 4] is defined as follows:

$$\mathbf{muf} = \{\text{muf}_0, \text{muf}_1, ...\text{muf}_n\} \tag{2.2}$$

with

$$\text{muf}_i = \sum_{l \in l_0} \int_{t=\text{MBP}_{i-1}}^{\text{MBP}_i} I_{t,l} - \sum_{l \in l_1} \int_{t=\text{MBP}_{i-1}}^{\text{MBP}_i} O_{t,l} - \sum_{l \in l_2} (C_{i,l} - C_{i-1,l}) \tag{2.3}$$

The covariance matrix contains the covariance between different material balances in the sequence. For example, consider the entry σ_{2n}^2 of the covariance matrix below. This term is the variance between material balance n and 2.

$$\Sigma = \begin{pmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \dots & \sigma_{2n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \dots & \sigma_{nn}^2 \end{pmatrix} = \begin{pmatrix} \Sigma_{i-1} & \sigma_{i-1} \\ \sigma_{i-1}{}^T & \sigma_{i,i} \end{pmatrix} \tag{2.4}$$

The simplest statistical test to detect a loss in the MUF sequence would be a simple hypothesis test:

$$\begin{aligned} &H_0 \ : \ E(\text{muf}_i) = 0 \text{ for } i \in \{1, 2, ..., N\} \\ &H_1 \ : \ E(\text{muf}_i) = M_i \text{ for } i \in \{1, 2, ..., N\} \\ &\text{where} \\ &\sum M_i = M > 0 \end{aligned} \tag{2.5}$$

[3] Inventories are assumed to always be masses, not flows.

For all loss patterns, $\mathbf{M}_N^T = \{M_1, M_2, ...M_N\}$, where M_i is the loss in period i, the optimal test to compare H_0 and H_1 is a Neyman-Pearson test. Siefert [3] showed the test statistic for such an optimal test can be defined as:

$$Z = M_N^T \Sigma_N^{-1} \mathbf{muf}_N \tag{2.6}$$

With the test itself formulated as:

$$Z \begin{cases} > \mathrm{k}_\alpha : & \text{reject } H_0 \\ \leq \mathrm{k}_\alpha : & \text{reject } H_1 \end{cases} \tag{2.7}$$

There's two challenges with this test. First, the test doesn't provide sequential decisions (not necessarily a problem considering the test can still be calculated sequentially, which we will do). This can be remedied by simply calculating the test statistic for each period and making decisions as such:

$$ZG_i = M_i^T \Sigma_N^{-1} \mathbf{muf}_i \tag{2.8}$$

with decision process:

$$ZG_N \begin{cases} > s(N) : & \text{reject } H_0 \\ \leq s(N) : & \text{no reject } H_1 \end{cases} \tag{2.9}$$

Second, and more problematic, is the requirement that the loss pattern, M_N is known. It is reasonable to approximate M_N as $M_N \approx \mathbf{muf}_N$ such that $\hat{M_N} = \mathbf{muf}_N$ by considering that $E(\mathrm{muf}_i) = M_i$.

The test statistic can then be formulated as:

$$\begin{aligned} Z &= \left(\hat{M_N}\right)^T \Sigma_N^{-1} \mathbf{muf}_N \\ &= \mathbf{muf}_N{}^T \Sigma_N^{-1} \mathbf{muf}_N \end{aligned} \tag{2.10}$$

Siefert noted that using a single MUF value at each step (i.e., M_i) can lead to significant variance. It was proposed to use a weighted value such that:

$$M_i = \frac{1}{7}(\mathrm{muf}_{i-2} + \mathrm{muf}_{i-1} + 3\mathrm{muf}_i + \mathrm{muf}_{i+1} + \mathrm{muf}_{i+2}) \tag{2.11}$$

This approach has lower variance, but is no longer unbiased. In MAPIT we implement both approaches, the use of the single MUF value is referenced as V1 and the use of the weighted values is V5B3, following the notation in the original paper. The code implementation (V1 shown below) only requires a few lines of code as much of the complexity for this test involves calculating the covariance matrix, which we already implemented and use for SITMUF.

StatsTest - GEMUF

```
    for ZR in range(1, int(nMBP)):
        IDs[ZR] = MUF[k,ZR*MBP]
        tempID = IDs[:ZR]
        tempcovmatrix = covmatrix[k,:ZR,:ZR]
        ZG = np.matmul(np.matmul(np.transpose(tempID), np.linalg.inv(tempcovmatrix)), tempID)
        GEMUFCalcsV1[k,int((ZR - 1) * MBP):int(ZR * MBP)] = np.ones((MBP,)) * ZG
```

2.4. Miscellaneous improvements

We made a few other under-the-hood improvements that are smaller in impact, but important for maintaining the tool:

- **Conda-Forge Deployment:** We added MAPIT to the conda-forge build repository in FY25. Anaconda is a package management platform for Python that help to manage install and deployment of Python code. Conda-forge is the community managed list of Python packages, which MAPIT is now a part of. This enables end users to very easily update and deploy MAPIT. Now, MAPIT can be installed with a single line of code `conda install MAPIT -c conda-forge` without managing dependencies or environments.

Name	Downloads	Version	Platforms
recipe mapit	downloads 1.7k	conda-forge v1.5.0b0	platform noarch

Current MAPIT statistics on conda-forge

- **Improved Parallel Behavior:** Previously, MAPIT relied on the `ray` library to perform parallel processing of large scale accountancy calculations. In FY25, we replaced `ray` with a simpler `concurrent.futures` workflow. This helps shrink the footprint of MAPIT while reducing code complexity.
- **Improved GUI Thresholds:** We improved the functionality of the graphical thresholding and associated statistics in the MAPIT GUI to allow for higher order thresholds. Previously, MAPIT could only perform a single threshold at a constant value (i.e., order 0). We added the ability to specify an order 1 threshold (i.e., ax+b) to support analysis with the GEMUF test, which has a linear test statistic that changes with time.

This page intentionally left blank.

3. F3M

The Fissile Facility Flow Modeler (F3M) is the open source library of Simulink blocks that can be used to compose models of nuclear fuel cycle facilities to support material control and accountancy research. The largest effort for F3M in FY25 was launching the library in the public domain. We also continued to add a few new blocks to the library and fixed several bugs as we completed new SSPM-L models.

3.1. Block improvements

The majority of work conducted on F3M itself during FY25 focused on bug fixes in and around discrete item handling. As this was a new capability added to MAPIT, some iteration was required to ensure the data recorder blocks worked properly. The one notable block that was added is our first custom block, which was created to support the fuel fabrication model. This block, the "pop queue" block was written by us to behave like a modified entity queue. Typical Simulink queues have capacities for entities (items) that, once reached, no longer accept items. For example, if a queue has a capacity of 25 items, then item 26 cannot enter. There are some instances where we wanted a constant item queue, regardless of the arrival of new entities. Here, when the pop queue is full, it pushes out the oldest entity to make space for the new entity. In that way, the oldest entities are "popped" out when new ones arrive.

Custom Simulink blocks must be coded in MATLAB, not composed of existing blocks. The code for the pop queue is relatively straightforward and can be found in Appendix A.

3.2. Initial public release

We released our initial version of F3M on our public facing GitHub repository in FY25. Although not necessarily a "technical" achievement, this required extensive work with our legal and export control teams at Sandia. We have ultimately been able to release F3M with a permissive license with the hopes that other stakeholders will be able to use our blocks to help create models of their own.

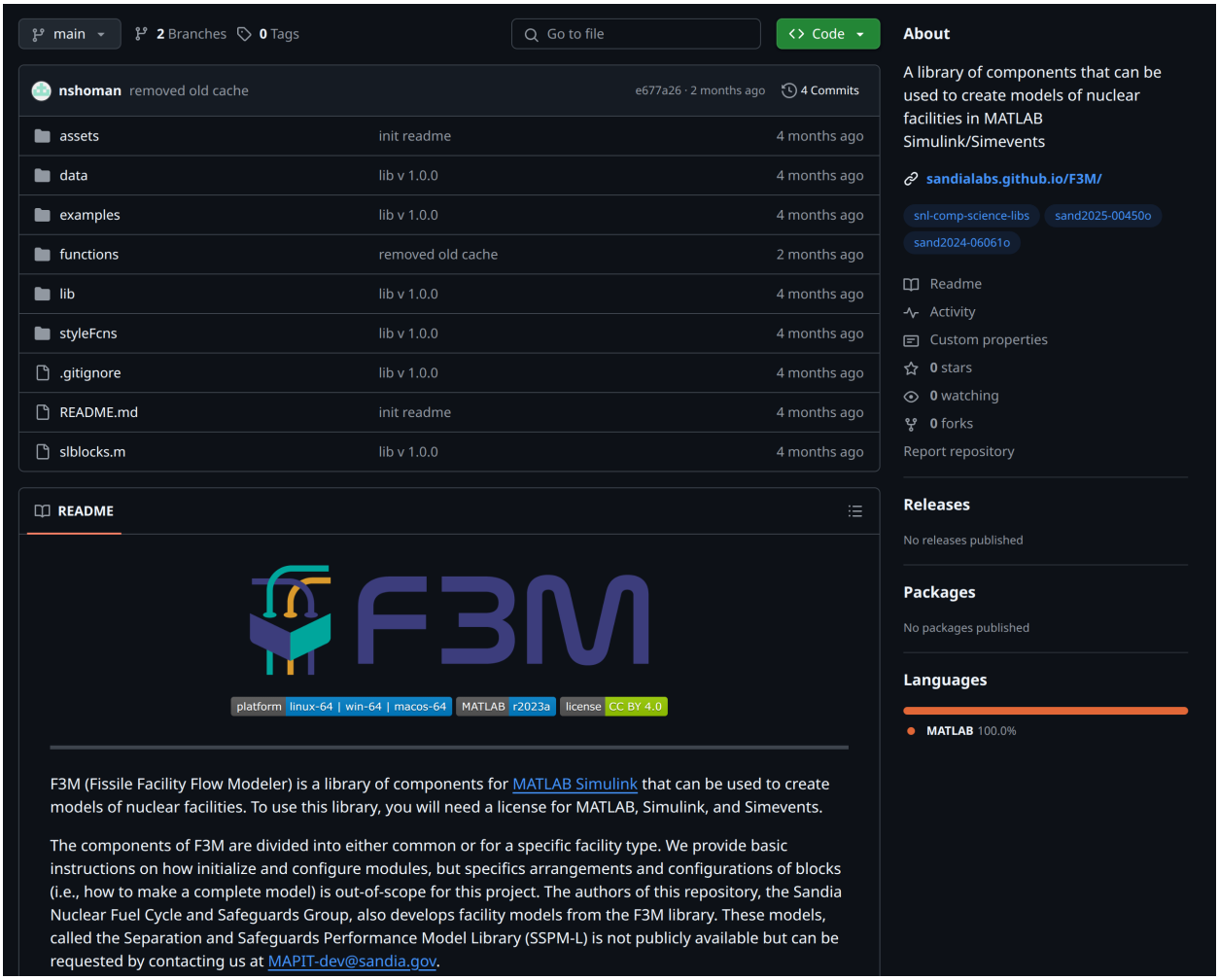

Figure 3-1: Initial release of F3M onto GitHub.

3.3. Finalized documentation

Along with our initial release of F3M as open-source code, we completed all the complementary documentation. Each block has documentation explaining how it's used, what different options mean, and any important use details. An example of a completed document page can be seen below in Figure 3-2.

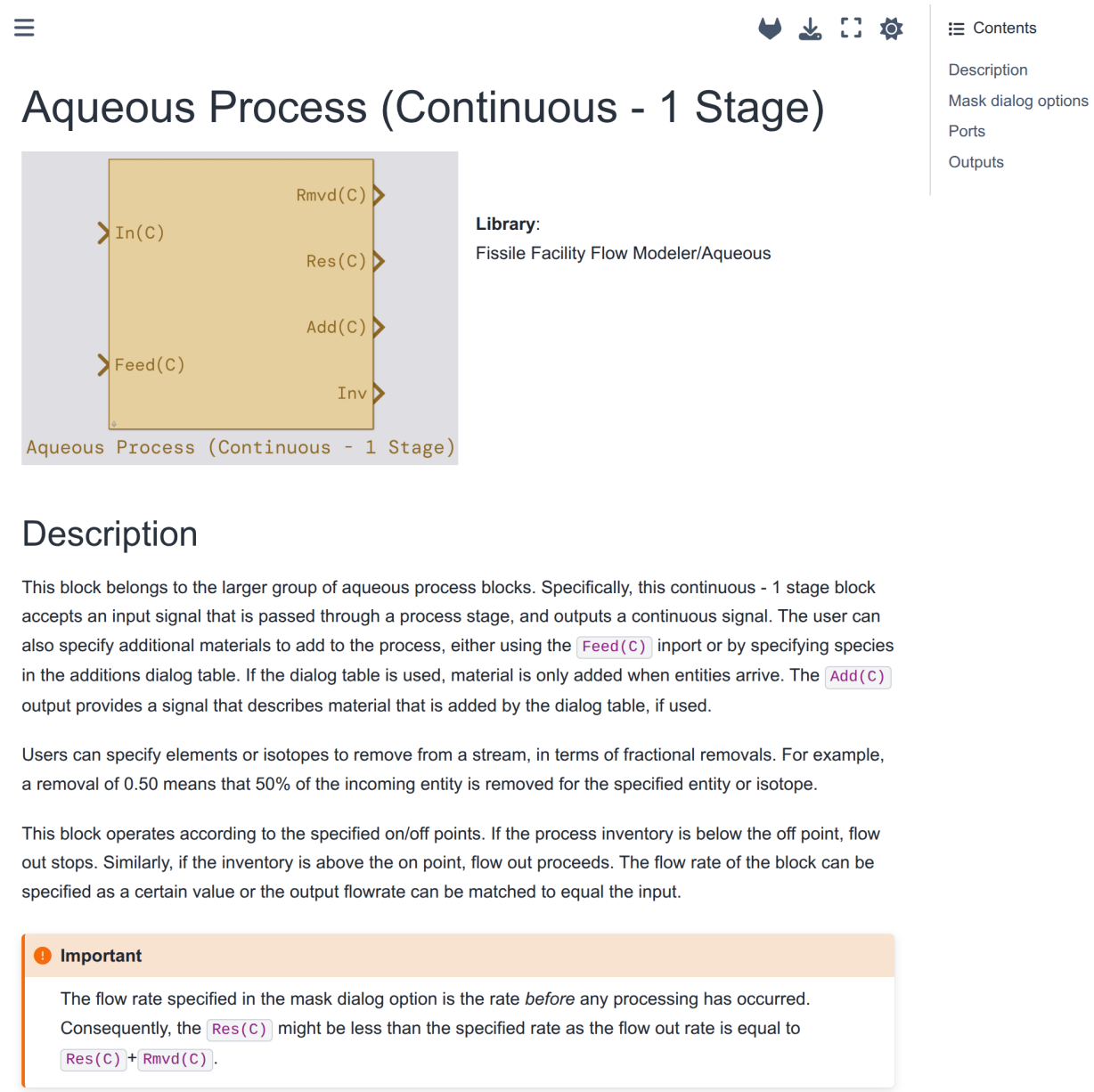

Figure 3-2: Example of F3M block documentation.

This page intentionally left blank.

4. SSPM-L

The most substantial work in the modeling and simulation software packages in FY25 was for SSPM-L. SSPM-L is the library of models composed from F3M blocks. The model files themselves are accompanied by initialization files that closely tie model parameters to references. In FY25, we completed three new models, bringing the total in the library to four. In many instances, we leveraged the new discrete item capabilities developed for F3M to improve both accuracy and simulation efficiency. F3M and SSPM-L is actively being used in several collaborations with industry, so these new models will greatly improve our ability to expand support for domestic vendors.

4.1. Fuel Fabrication

A generic LEU fuel facility based on a number of references (described and linked in the model card) and leverages the F3M library. This initializer file is included to enable easier changing of the facility throughput. All the calculations below are based on a series of Reference and should only be modified if being used to represent a different facility. By default, we assume 300 MTIHM/yr throughput with 270 operational days.

The flowsheet itself is largely derived from a NRC training module on fuel fabrication (publicly available here) along with STR-150. Further details of the individual unit operation can be found either in the initialization file or in the Simulink model itself in the block properties.

This model adopts the F3M notation of tracking 1675 isotopes. The two auxiliary slots of the F3M signal are utilized in the model. The first slot, `at1` is the bulk liquid flow and the second slot, `at2` is the bulk solid flow. These aren't currently used, but could be useful in the future to relax a number of simplifying modeling assumptions. This model represents a conversion from the legacy SSPM model to the new SSPM-L format with few changes to the underlying operational flowsheet.

A screenshot of the model can be seen in Figure 4-1 below.

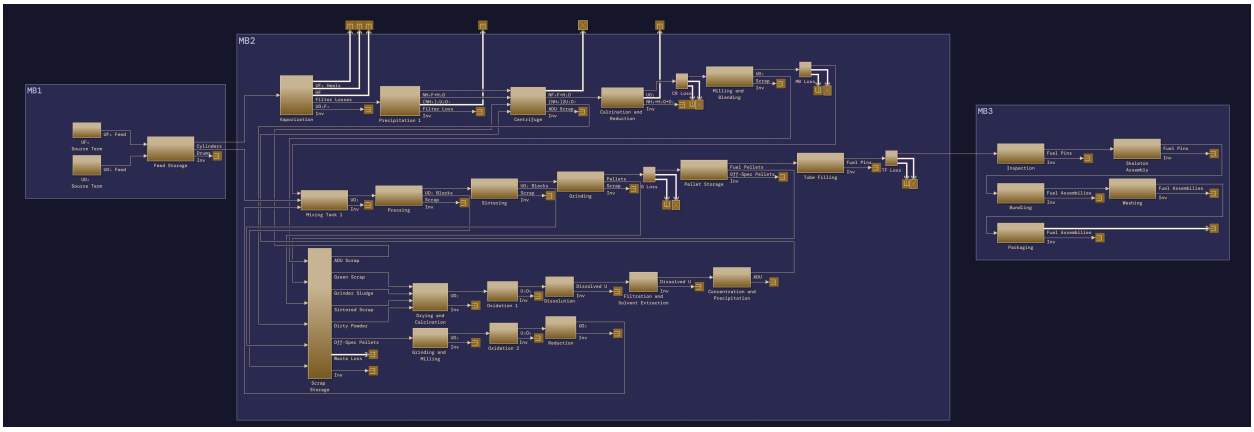

Figure 4-1: SSPM-L Fuel Fabrication model flowsheet

4.2. TRISO Compact

This model is a generic TRISO fuel fabrication model. It is assumed that the TRISO particles are used to form fuel compacts, which ultimately form hexagonal fuel assemblies used for prismatic HTGR designs. The flowsheet is largely derived from an open source IAEA document (IAEA-TECDOC-CD-1645 [5]). The process starts with a HALEU UO_2 to produce the coated TRISO particles (uranium oxycarbide kernel manufacturing). Then, particles are coated with the various carbide layers (coated particle manufacturing). Next, particles are assembled into compacts before finally being assembled into fuel elements.

This model does contain some notional representations of chemical processes, but does not include chemical models themselves. That is, we might have a process block that adds 10kg of carbon to a batch of TRISO fuel particles to simulate the deposition of carbide layers, but we don't have chemical models to simulate that process. We can only specify simple additions or removals to loss terms and can't model chemical interactions in a detailed way.

This model adopts the F3M notation of tracking 1675 isotopes. The two auxiliary slots of the F3M signal are utilized in the model. The first slot, at1 is the bulk liquid flow and the second slot, at2 is the bulk solid flow. Here, bulk flow represents a material that is not tracked on an elemental/isotopic basis, but is nonetheless needed for mass balances and model simulation. In this model, at1 and at2 are sometimes used to represent solvent masses and flows.

A screenshot of the model can be seen in Figure 4-2 below.

Figure 4-2: SSPM-L TRISO Fuel Fabrication (compact) flowsheet

4.3. Generic Electrochemical Reprocessing

The final model added in FY25 is a generic electrochemical reprocessing facility. Unlike the other SSPM-L models, the EChem model has a much looser relation to related literature. The original SSPM model (circa 2020) avoided direct references to conceptual flowsheet design over sensitivity concerns that they might one day become real facilities. Consequently, this model loosely follows flowsheet references but relies on notional process values based on rule-of-thumb engineering.

As with all of the other models, the generic electrochemical model includes an initialization file that explains the model assumptions while allowing users to change some parameters. It model adopts the F3M notation of tracking 1675 isotopes. The two auxiliary slots of the F3M signal are utilized in the model. The first slot, `at1` is the bulk liquid flow and the second slot, `at2` is the bulk solid flow. Here, bulk liquid flow, `at1` represents a material that is not tracked on an elemental/isotopic basis, but is nonetheless available in the model simulation. For this EChem model, at1 is used to reference the bulk salt component, which is not broken down into elemental or isotopic constituents. `at2` is largely unused here.

A screenshot of the model can be seen in Figure 4-3 below.

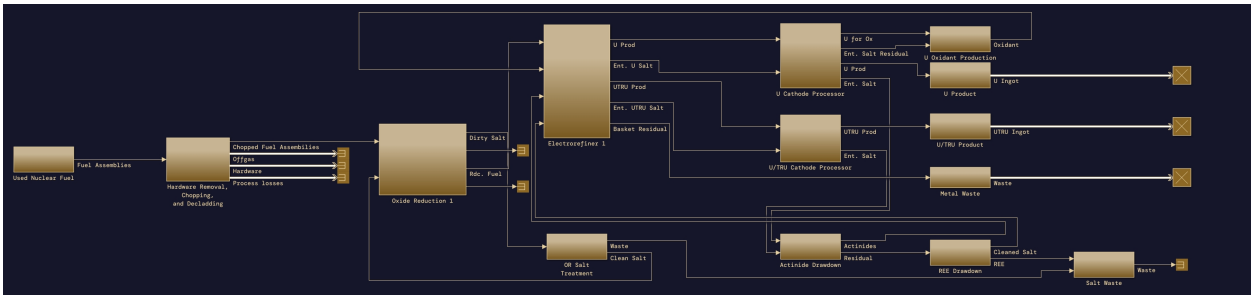

Figure 4-3: SSPM-L Electrochemical reprocessing flowsheet

5. SUMMARY

This report covers improvements to Sandia's material control and accountancy software in FY25. The most significant improvement was for SSPM-L where three new models were added to the library. We added new blocks to F3M, including our first totally custom block, to support these models. We ensured continued integration between our modeling (F3M and SSPM-L) tools with our analytical tools (MAPIT) by adding support for discrete items. We also continued to support and address user concerns by fixing several bugs raised by the community. Finally, we added the GEMUF test to MAPIT, to expand the number of statistical tests available to users.

This page intentionally left blank.

REFERENCES

[1] S. F. DeMuth, *Proliferation Resistance and Safeguards*, pp. 3421–3538. Boston, MA: Springer US, 2010.

[2] T. Burr and M. S. Hamada, "Revisiting statistical aspects of nuclear material accounting," *Science and Technology of Nuclear Installations*, March 2013.

[3] R. Seifert, *The GEMUF test: A new sequential test for detecting loss of material in a sequence of accounting periods*. IAEA., 1987.

[4] R. R. Picard, "Sequential analysis of material balances," *Journal of Nuclear Materials Management*, vol. 15, 1987.

[5] I. A. E. Agency, *High temperature gas cooled reactor fuels and materials*. IAEA, 2010.

This page intentionally left blank.

APPENDIX A. POP QUEUE

The following code block is used to create the new pop queue block:

```
classdef F3_popQueue < matlab.DiscreteEventSystem

    % Push-Pop FIFO like entity queue
    % Queue that holds entities to a capacity
    % Essentially the queue fills to capacity
    % then pushes out FIFO entities as new ones
    % arrive

    properties(Nontunable)
        Capacity=10; % Queue size - can be adjusted by dialog also
    end

    properties(DiscreteState)
        Count
    end


    methods (Access = protected)

        % Single input and output
        function num = getNumInputsImpl(~), num = 1; end
        function num = getNumOutputsImpl(~), num = 1; end

        % Type entity, same as generator string
        function types = getEntityTypesImpl(obj)
            types = obj.entityType('Entity');
        end

        % types for in/out ports
        function [inT, outT] = getEntityPortsImpl(~)
            inT = {'Entity'};
            outT = {'Entity'};
        end

        % create FIFO storage and connect input1 to storage
        % and storage to output 1
        function [specs, I, O] = getEntityStorageImpl(obj)
            specs = obj.queueFIFO('Entity',obj.Capacity+1);
            I = 1;
            O = 1;
        end

        % set the value of our internal counter
        function resetImpl(obj)
            obj.Count = 0;
        end

    end
```

```
    methods

        % Function to iterate over entities
        function [entity, events, next] = iterate(obj, storage, entity, tag, status)
            events = obj.initEventArray;
            next = false;

            % if an entity is tagged to be popped and its next
            % up then try to send it on
            if strcmp(tag,'pop') && status.position == 1
                events(end+1) = obj.eventForward('output',1,0);
            end
        end

        % when an entity leaves
        function event = exit(obj, storage, entity, dest)

            % reduce the internal count of queued entities
            obj.Count = obj.Count - 1;

            % allow additional entities in
            event = obj.eventTestEntry(storage);
        end

        % on entity arrival
        function [entity, events] = entry(obj, storage, entity, ~)

            % update count
            obj.Count = obj.Count + 1;
            events = obj.initEventArray;

            % if theres too many entities, we need
            % to pop, so mark the position 1 entity
            % in the queue to leave
            if obj.Count > obj.Capacity
                events = obj.eventIterate(storage,'pop',1);
            end
        end
    end


end
```

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Nathan Shoman	8845	nshoman@sandia.gov
Philip Honnold	8845	phonnol@sandia.gov
Ramon Pulido	8845	rpulido@sandia.gov
Tania Rivas	8845	tnrivas@sandia.gov
Anna Taconi	8845	ataconi@sandia.gov
Scott Sanborn	8845	sesanbo@sandia.gov
Benjamin Cipiti	8845	bbcipit@sandia.gov
Technical Library	1911	sanddocs@sandia.gov

Email—External

Name	Company Email Address	Company Name
Mike Browne	mcbrowne@lanl.gov	DOE
Tansel Selekler	tansel.selekler@nuclear.energy.gov	DOE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.