Benchmark of Cyclus Fuel Cycle Simulation Tool in Transition Scenario

Jin Whan Bae[1], Joshua L. Peterson-Droogh[2], Kathryn Huff[1]

[1] *Dept. of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana-Champaign, Urbana, IL*
[2] *Oak Ridge National Laboratory, Oak Ridge, TN*

Abstract

Numerous nuclear fuel cycle system modeling codes have been developed to perform fuel cycle transition analyses from a once-through cycle to an advanced fuel cycle. Verification studies compare different fuel cycle analysis tools against each other to test agreement and identify sources of difference. This paper benchmarks Cyclus, the agent-based, open-source fuel cycle simulation code, against a verification study [1] for DYMOND [2], VISION [3], ORION [4], and MARKAL [5]. The study reveals that Cyclus results match the spreadsheet results closely, with minor differences caused by reactor module behavior.

---

1. Introduction

Fuel cycle simulators act as an crucial tool to aid decision in policy and fuel cycle strategies. To meet this need from various institutions, a multitude of fuel cycle simulators were developed, using different methods and different structures to simulate the material flow in the nuclear fuel cycle. The difference in the algorithm of fuel cycle analysis codes combined with a small user community make validation studies necessary to gain confidence of the capability of the code as well as its agreement with other analysis codes.

This study benchmarks Cyclus' results against that of other well-known codes, such as DYMOND [2], VISION [3], ORION [4], and MARKAL [5]. We take the input parameters and results from a validation study [1] already done for the mentioned tools for a transition scenario from an open fuel cycle to an advanced fuel cycle with reprocessing. In the benchmark [1], the 'model solutions' generated from an excel worksheet are compared to each code results, and the results show excellent agreement.

1.1. Cyclus

Cyclus is an agent-based fuel cycle simulation framework [6], which means that each reactor, reprocessing plant, and fuel fabrication plant is modeled as an agent. A Cyclus simulation contains prototypes, which are fuel cycle facilities with pre-defined parameters, that are deployed in the simulation as `facility` agents. Encapsulating the `facility` agents are the `institution` and `region`. A `region` agent holds a set of `institutions`. An `institution` agent can deploy or decommission `facility` agents. The `institution` agent is part of a `region` agent, which can contain multiple `institution` agents. Several versions of `Institution` and `region` exist, varying in complexity and functions [7]. `DeployInst` is used as the institution archetype for this work, where the institution deploys agents at user-defined timesteps.

At each timestep (one month), agents make requests for materials or bid to supply them and exchange with one another. A market-like mechanism called the dynamic resource exchange [8] governs the exchanges. Each material resource has a quantity, composition, name, and a unique identifier for output analysis. The timestep execution in Cyclus follows `Build, Tick,` dynamic resource exchange (DRE), `Tock, and Decommission`, as illustrated in figure 1. The `Tick`, and `Tock` phases are for each agent to perform actions, such as transmutation, separation, or generation of materials before and after the market exchange phase.

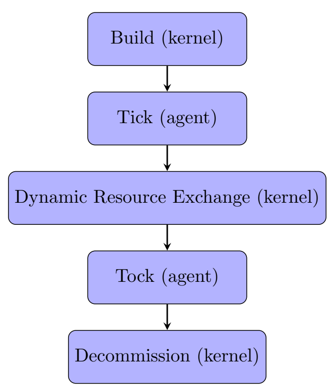

Figure 1: Cyclus timestep execution steps.

The modularity of Cyclus allows a low barrier of entry for developers, since developers can create an archetype (e.g. Reactor module, Reprocessing module) without extensive knowledge of the Cyclus framework.

2. Methodology

The benchmark paper [1] has comprehensive simulation parameters that allow reproduction of the transition scenario in Cyclus. In this study, we used the Cycamore [6] archetype library to model all fuel cycle facilities. Cycamore libraries contain simple fuel cycle facility models. For example, the Reactor module does depletion calculations through user-defined recipes.

Cyclus outputs files in either `.sqlite` or `.h5` format. In this study, we used the `.sqlite` format and analyzed the output file using a python script.

After post-processing of the output data, we overlap the results with the benchmark's solutions for comparison.The input file and analysis procedures are all in [zenodo].

The analysis and benchmark is performed iteratively, where the we improve the original result by communicating with the authors of the benchmark. From the original result, the reasons for the differences were analyzed and small edits in the source code were made accordingly. Major differences in the source code are not edited but simply explained in detail as to how they contribute to the difference.

3. Fundamental Code Differences in Cyclus

Cyclus has fundamental code differences than some fuel cycle analysis codes used in the benchmark [1].

Cyclus has a default timestep of a month. In order to take this into account, we calculate the annual value (i.e. average value for inventory values and sum of 12 months for throughput values) for each result. The timestep in Cyclus can be changed into a year without changing the source code, but Cyclus' timestep execution (figure 1) causes a delay in the material flow. Thus, having the timestep be 12 months allows the lessening of the impact of the delay due to the Cyclus timestep execution.

Similarly, Cyclus has discrete execution steps per timestep that might cause delays or contort the results from other simulators. For example, decommissioning of facilities occur at the end of a timestep, while building of facilities occur at the beginning of a timestep.

The Cycamore recipe reactor depletes half of its core when decommissioned, whereas the codes in the benchmark [1] deplete all its fuel when decommissioned. This causes a major discrepancy for transuranic elements (TRU) inventory. For this study, we changed the Cycamore source code to deplete all its assemblies to the depleted recipe. Also, the Cycamore recipe reactor treats each batch (and assembly) as a discrete material, while some codes have continuous fuel discharge. This produces differences in the results because the batches in the benchmark [1] are in fractions. In this study, the Light Water Reactor (LWR) batch size and cycle time is increased, while decreasing the batch number to keep the core size constant. We simply round up the Sodium-Cooled Fast Reactor (SFR) batch number, while the batch size and cycle time are kept constant. This increases the core size by 1.08%, which is negligible, but will be discussed in the results section. We list the differences in table 1.

Note that all the differences could have been mediated by changing the archetype source codes. However, the only change made was the reactor depletion behavior at decommission due to its large impact. Note that the goal of this study is to show current Cyclus agreement with other codes and identify differences, not to alter Cyclus to match the other codes.

Table 1: Difference in Batch number and core size

Category	Benchmark [1]	This study
LWR Batches	4.5	3
LWR Batch size [tHM]	19.91	29.86
LWR Core size [tHM]	89.59	89.59
LWR Cycle time	1 year	1.5 years
SFR Batches	3.96	4
SFR Batch size [tHM]	3.95	3.95
SFR Core size [tHM]	15.63	15.8

4. Results

We represent each CYCLUS result as a solid line, and the benchmark solution as a dotted line for visualization. The results are simply a reproduction of the plots displayed in the benchmark. We obtained the benchmark solutions through personal contact with benchmark paper's author Bo Feng at Argonne National Laboratory.

Figure 2 shows the deployed reactor capacity, and figure 3 shows the LWR retirement and SFR deployment timeseries. The two plots show exact agreement with the benchmark solutions.

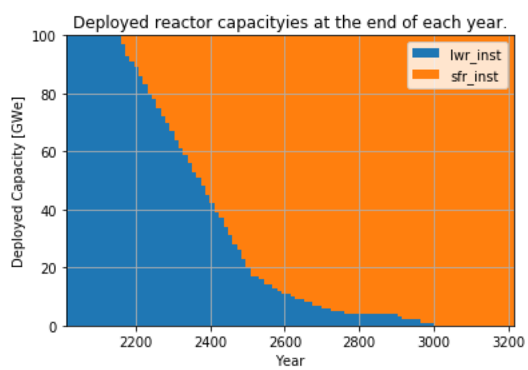

Figure 2: Deployed reactor capacities at the end of each year.

Figure 4 shows the annual fuel loading rate. The initial fuel loading for 100 LWR reactors were edited to match the plot in the verification study results. Note the oscillations for the LWR fuel loading is caused by the refueling period being 18-month refuel cycle for all LWR reactors aggregated into 12-month groups. Note also that the total values are equal for both plots.

Although indistinguishable in figure 4, there is a small difference with SFR fuel loading proportional to the core mass difference, as mentioned in the previous section. Figure 5 shows the differences normalized by the core mass dif-

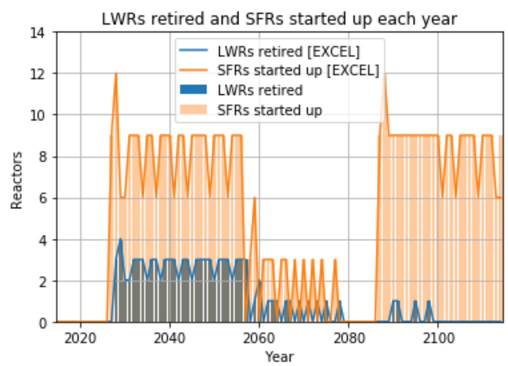

Figure 3: LWRs retired and SFRs started up each year.

ferences, overlapped with the SFR deployment. This shows that the differences only occur during deployment due to the difference in core mass.

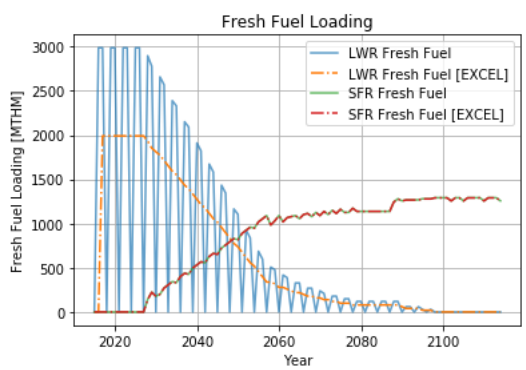

Figure 4: Annual fresh fuel loading rates (first cores and reload fuel).

Figure 6 shows the inventory of discharged Used Nuclear Fuel (UNF) in mandatory cooling stage. It also oscillates between the benchmarks solution,and converges, caused by the influx and the outflux of UNF into and out of the storage facility. Note that for most plots the SFR inventory and fuel loading exactly matches the benchmark solutions, minus the small difference due to core size.

Figure 7 shows similar results for the inventory of cooled UNF waiting for reprocessing. Unlike the previous plot, however, the oscillation peaks meet with the benchmark solution. This is because the cooled UNF inventory is measured by the cumulative sum of UNF that has been cooled subtracted by the UNF reprocessed at that timestep. Thus, the peaks in the oscillation correspond to the cooled inventory in the storage facility before it sends its inventory to

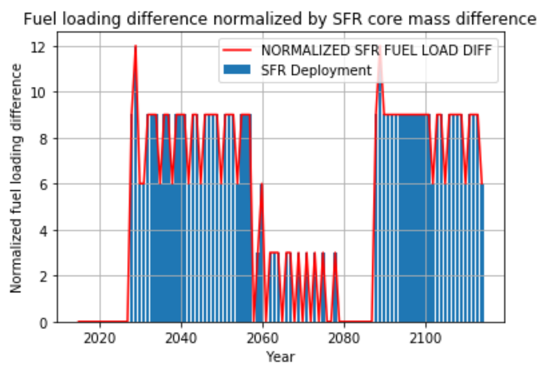

Figure 5: Difference of annual fresh SFR fuel loading rates (Cyclus - Benchmark) normalized by the core mass difference of an SFR due to fractional batch size.

reprocessing.

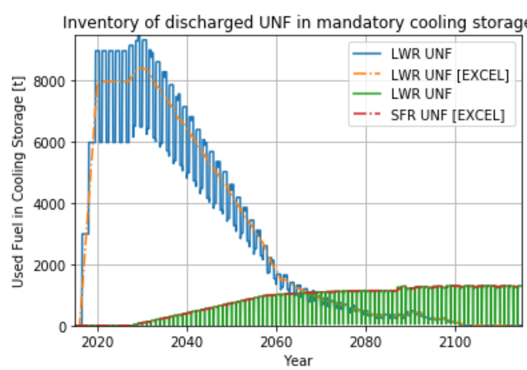

Figure 6: Inventory of discharged UNF in mandatory cooling storage.

Figure 8 shows the reprocessing throughput, which also oscillates between the benchmark solution. Note that no oscillation exists in the beginning because the LWR UNF reprocessing plant throughput is maximized at 2,000 tons per year.

Figure 9 shows the inventory of unused TRU recovered from UNF. The Cyclus results follows the benchmark solutions closely. However, the difference in core size causes Cyclus results to be smaller, since more TRU is used to start up the newly deployed SFRs. The difference decreases as the SFRs decommission, discharging more UNF (thus TRU) than the benchmark.

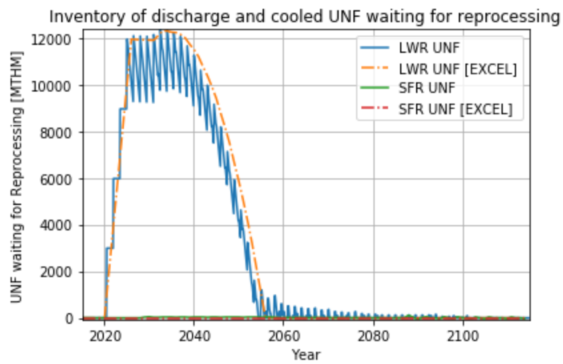

Figure 7: Inventory of discharged and cooled UNF waiting for reprocessing.

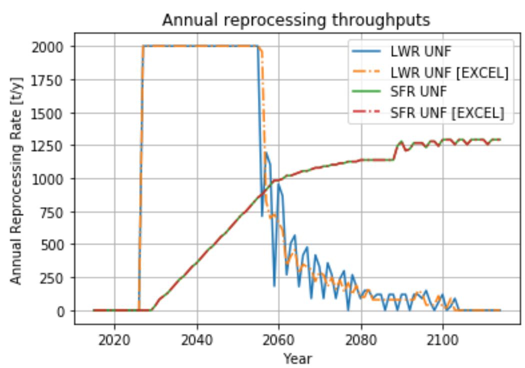

Figure 8: Annual reprocessing throughputs.

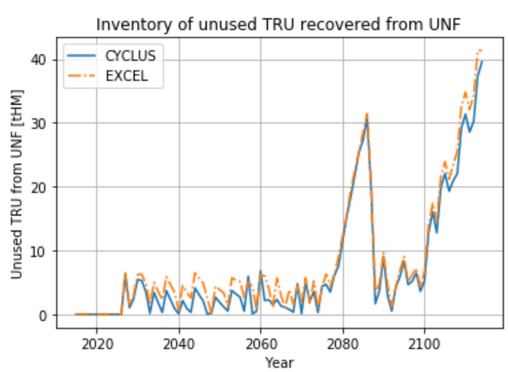

Figure 9: Inventory of unused TRU recovered from UNF.

5. Discussion

We benchmarked Cyclus, the agent-based fuel cycle simulator with results from another verification study and saw good agreement in a transition scenario.

Throughout this work, two major differences were identified that led to the deviation of Cyclus results to that of the excel sheet. First, the Cycamore reactor depletes only half of its core when decommissioned. Second, Cyclus, unlike other codes examined in the benchmark (except ORION), ~~only has~~ discrete batches for fuel discharge. We change the first issue by changing one line in the source code. However, we did not change the second issue intentionally to show that the final results still match the benchmark solutions.

This study proves Cyclus as a capable tool for modeling fuel cycle transition scenarios, and shows promise for expansion and future development.

6. Acknowledgments

The work done was funded through the Nuclear Engineering Science Laboratory Synthesis (NESLS) program. We thank Eva Davidson from Oak Ridge National Laboratory (ORNL) and Bo Feng from Argonne National Laboratory (ANL) for their aid in providing benchmark solutions and insight for this work.

References

[1] B. Feng, B. Dixon, E. Sunny, A. Cuadra, J. Jacobson, N. R. Brown, J. Powers, A. Worrall, S. Passerini, R. Gregg, Standardized verification of fuel cycle modeling 94 300–312. doi:10.1016/j.anucene.2016.03.002. URL http://www.sciencedirect.com/science/article/pii/S0306454916301098

[2] A. M. Yacout, J. J. Jacobson, G. E. Matthern, S. J. Piet, A. Moisseytsev, Modeling the nuclear fuel cycle, in: The 23rd International Conference of the System Dynamics Society," Boston, Citeseer.

[3] J. J. Jacobson, A. M. Yacout, G. E. Matthern, S. J. Piet, D. E. Shropshire, R. F. Jeffers, T. Schweitzer, Verifiable fuel cycle simulation model (VISION): a tool for analyzing nuclear fuel cycle futures 172 (2) 157–178.

[4] R. Gregg, C. Grove, Analysis of the UK nuclear fission roadmap using the ORION fuel cycle modelling code, in: Proc of the IChemE nuclear fuel cycle conference, Manchester, United Kingdom.

[5] C. Shay, J. DeCarolis, D. Loughlin, C. Gage, S. Yeh, S. Vijay, E. L. Wright, EPA US national MARKAL database: database documentation.

[6] K. D. Huff, M. J. Gidden, R. W. Carlsen, R. R. Flanagan, M. B. McGarry, A. C. Opotowsky, E. A. Schneider, A. M. Scopatz, P. P. H. Wilson, Fundamental concepts in the cyclus nuclear fuel cycle simulation framework 94 46–59. doi:10.1016/j.advengsoft.2016.01.014. URL http://www.sciencedirect.com/science/article/pii/S0965997816300229

[7] K. Huff, M. Fratoni, H. Greenberg, Extensions to the cyclus ecosystem in support of market-driven transition capability, Tech. rep., Lawrence Livermore National Laboratory (LLNL), Livermore, CA (2014).

[8] M. J. Gidden, An agent-based modeling framework and application for the generic nuclear fuel cycle, Ph.D. thesis, The University of Wisconsin-Madison (2015).