Once-Through Benchmarks with CYCLUS, a Modular, Open-Source Fuel Cycle Simulator

Matthew J. Gidden, Paul P.H. Wilson, Kathryn D. Huff, Robert W. Carlsen
Department of Nuclear Engineering & Engineering Physics, University of Wisconsin - Madison, Madison, WI, 53703
gidden@wisc.edu

INTRODUCTION

The CYCLUS project, based at the University of Wisconsin - Madison, is an open source platform for exploring the long-term impact of alternative nuclear fuel cycles. The CYCLUS core provides the infrastructure for an agent-based approach, allowing user-provided modules to define the behavior of fuel cycle facilities as they interact to exchange materials. An important consequence of this approach is that innovative facility and material exchange concepts can be introduced to a consistent framework allowing for more rigorous comparison. The CYCLUS team has recently grown and now incorporates a variety of expertise: output visualization capability through collaboration with the University of Utah, server-client communication via the University of Idaho, input visualization and control with the University of Texas - Austin, and social communication expertise through collaborators at UW-Madison to assist the mission-critical goal of relevancy vis-à-vis policy makers. Accordingly, the CYCLUS project is expanding efforts in the realms of both structural capability and benchmarking calculations.

A series of once-through fuel cycle scenarios are being conducted using the CYCLUS core and accompanying modules. Where needed, additional modules have been added, including a region model that intelligently makes building decisions given a demand function. The results of these scenarios are then compared with VISION [1] to provide a benchmark of the CYCLUS results.

CYCLUS DESIGN AND DEVELOPMENT

The paradigm under which CYCLUS has been developed and is distributed is unique and offers a number of advantages regarding its potential wide-spread adoption by a diverse user base. CYCLUS is an open source collection of libraries that provides out-of-the-box capability as well as a platform on which to develop additional modules. At present, the CYCLUS suite includes a core library, a module development toolkit, and a basic module pack. The module development toolkit includes a growing collection of standard and commonly used capabilities, such as an interface to optimization problem solvers and resource buffers for storing inventories.

Open Development

As one of the leading principles guiding the development of CYCLUS, an open-source repository provides a high degree of flexibility and a large amount of exposure to potential collaborators and developers. The CYCLUS repository is publicly available via GitHub [2]. A number of tools are available to CYCLUS developers in order to maintain software development best-practices, including distributed version control, automatic documentation, and in-depth issue management. Additionally, CYCLUS developers have taken advantage of existing, external open-source libraries in order to conform to current standards, including an expansion of the C++ standard library and fully benchmarked linear and integer programming solvers. Perhaps the most important attribute of the open development paradigm is that it allows for unfettered access to the CYCLUS core and basic module source code. Combined with modular software development, CYCLUS is an easily transferable framework on which to build a strong fuel cycle simulation community.

Dynamic Module Capability

Incorporation of dynamically loadable, independently constructed modules is another key design concept in CYCLUS. Whereas the connections between staticly loaded libraries is determined at compile time, dynamically loaded libraries are linked at run time. This allows for the core simulation engine comprising CYCLUS to be physically separate from the various modules determining the specific behavior that occurs in each simulation. Because interaction between the core and modules occurs via dynamic linking, module development is encapsulated and can be performed without involving the simulation engine, increasing efficiency and decreasing the overall programming experience required. This separation not only allows for multiple developers to work towards a common goal, e.g. fully describing a target scenario, but also assists in separating simulation concerns into reasonably sized, independently testable modules.

Optimization

The CYCLUS project has recently added a linear, integer, and mixed integer program wrapper named CYCLOPTS [3] to its tool set. CYCLOPTS relies on the open-source solver Coin-OR Branch and Cut [4], providing a simple interface to describe such optimization problems. At the present time, developers have the ability to define a set of variables and to describe an objective function and series of constraints that utilize some subset of those variables. Variables currently come in two flavors, integer and linear, corresponding to their respective solution techniques. Additional work will be performed to add cutting plane support as well as a suite of unit tests.

VERIFICATION AND TESTING

Core Functionality Benchmarks

To verify the functionality of the CYCLUS core, basic problems using a single reactor type were performed and compared to results from both VISION and GENIUS [5], another fuel cycle simulator and predecessor to CYCLUS. Table 1 describes the parameters varied in the functionality benchmark.

Problem No.	No.Reactors	Growth Type
1	1	none
2	10	none
3	1	linear
4	1	exponential

TABLE 1: Core Functionality Benchmarking Parameters.

Once-Through Fuel Cycle Benchmarks

Once-through simulation results were compared with similarly defined scenarios in VISION. For CYCLUS, a subset of real-world facilities and their interactions were included: mines, enrichment facilities, reactors, and storage facilities. In order to drive the simulation, an electricity demand function is provided to the simulation, which determines the supply requirement for reactors at a given time step. Once the simulation is complete, an analysis of material flows and facility deployment as a function of time is provided. Of specific concern with respect to the once-through fuel cycle is long term used fuel production and the corresponding storage requirements.

Scenario Construction

Reactor deployment decisions are driven by the region in which the reactors exist. An electricity demand is defined for that region, either as a linear or exponential function. The region is also provided with a set of available facilities (reactors) that can meet this demand. At any time step in which there exists a demand gap, i.e. there exists more demand than supply, a build decision is made. This decision is modeled as the following integer program:

$$\min \quad \sum_{i \in I} n_i * c_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in I} n_i * \phi_i \geq \Phi \tag{1b}$$

$$n_i \in [0, \infty) \quad \forall\, i \in I \tag{1c}$$

$$n_i \ \ integer \quad \forall\, i \in I \tag{1d}$$

where Φ is the unmet demand, I is the set of facilities capable of meeting the demand, and, for each facility in I, c_i is the cost of building, and ϕ_i is the nameplate capacity. Finally, n_i is the optimized number of facilities to build of type i.

While electricity demand directly corresponds to reactor deployment, supporting facilities, e.g. enrichment facilities, mines, and storage facilities, depend on overall reactor fuel input and output requirements. There are multiple strategies that can be used to determine supporting facility deployment, of which a minimal constraint approach is first analyzed followed by a supply-demand constraint approach, i.e. when the demand for a resource, e.g. fresh fuel, reaches a percentage of the total supply, a facility meeting that demand is deployed.

Benchmarking and Parameter Variation

VISION uses an Excel spreadsheet input paradigm which allows one to develop new scenarios or vary the parameters of the provided "Base Case Scenarios". The once-through base case scenario is used and some of its corresponding parameters are varied in order to define a relative benchmark between CYCLUS and VISION. Specifically, the following parameters are varied:

- number of reactor types (e.g. LWRs and HWRs)
- electricity demand rate
- storage availability date (i.e. when a repository is available)

RESULTS & CONCLUSIONS

Results

CYCLUS has performed well in early verification comparisons with VISION. Results of the core functionality benchmark problems, which report total mass ejected from reactors as a function of time, are shown below in Fig. 1 and Fig. 2; note that the benchmark values used for VISION and GENIUS have been previously reported in Oliver2009 [6].

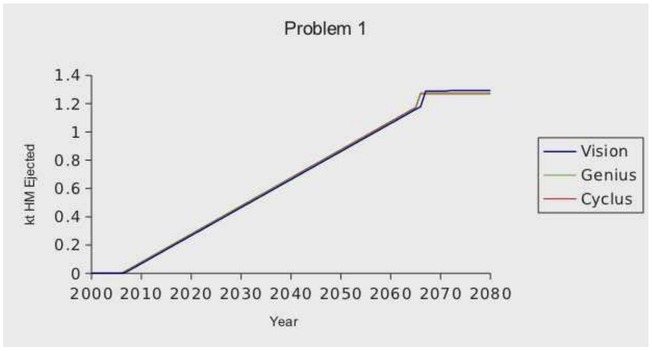

Fig. 1: Results from Problem 1.

A fundamental and expected result from this work is the effect of continuous material flow simulators, such as VISION, versus the discrete nature of GENIUS and CYCLUS. This difference accounts for the small discrepancies during the final core offload(s) [6].

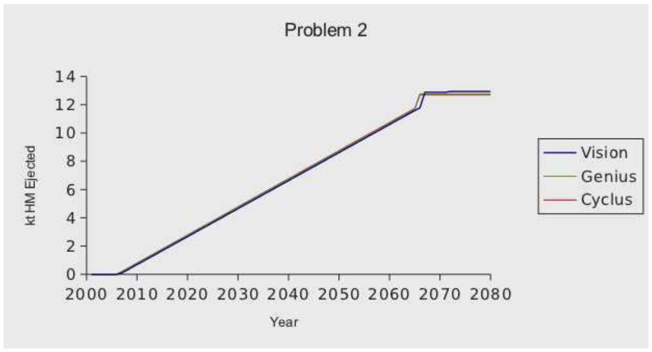

Fig. 2: Results from Problem 2.

Conclusions

The CYCLUS core has been shown to provide expected behavior of discrete material flows for simple reactor cases. CYCLUS additionally has the capability not only to model a variety of growth scenarios with varying fuel storage strategies, but also to emulate the reactor deployment algorithm of VISION, i.e. explicitly declaring reactor types to be built given a defined distribution, by adding an additional constraint in the decision making problem solved by the region in which the reactors exist.

Further work is currently underway to refine building strategies for non-reactor facilities; however, the next major project goal of the CYCLUS team is to provide recycle capability. Once completed, additional benchmarks will be performed to analyze code behavior using recycle scenarios.

ACKNOWLEDGMENTS

This research is being performed using funding received from the DOE Office of Nuclear Energy's Nuclear Energy University Programs. The author thanks the NEUP for its generous support.

REFERENCES

1. J. J. JACOBSON ET AL., *VISION User Guide - VISION (Verifiable Fuel Cycle Simulation) Model*, Idaho National Lab, inl/ext-09-16645 ed. (2009).
2. P. WILSON, M. GIDDEN, K. HUFF, and R. CARLSEN, "Cyclus: A Nuclear Fuel Cycle Code from the University of Wisconsin Madison," (June 2012), http://cyclus.github.com/.
3. M. GIDDEN, "Cyclopts: An Optimization Wrapper for Cyclus," (June 2012), https://github.com/cyclus/cyclopts.
4. J. J. FORREST, "Coin-OR Branch and Cut v2.7," (June 2012), https://projects.coin-or.org/Cbc.
5. C. JUCHAU, *Development of the global evaluation of nuclear infrastructure and utilization scenarios (GENIUS) nuclear fuel cycle systems analysis code*, Master's thesis, Idaho State University (2008).
6. K. OLIVER, *GENIUSV2: Software design and mathematical formulations for multi-region discrete nuclear fuel cycle simulation and analysis*, Master's thesis, University of Wisconsin - Madison (2009).