Cyclus Once-Through Fuel Cycle Benchmarks

Comparisons with VISION

Matthew J. Gidden

University of Wisconsin-Madison

April 13, 2012

Outline

Introduction : Purpose

Fuel cycle simulators are designed to answer policy-related questions regarding transitions from one equilibrium state to another.

Introduction : Purpose

Fuel cycle simulators are designed to answer policy-related questions regarding transitions from one equilibrium state to another.

A simulator answers the following questions as a function of its parameter space:

- how much material exists
- where does that material reside
- from/to where and when is material transported
- what kinds of facilities are needed
- when is each type of facility needed

Introduction : Simulators Today

In general, simulators differ in the following respects

- material flows: continuous or discrete
- facility deployment: individual or fleet-based
- regional modeling

Introduction : Simulators Today

In general, simulators differ in the following respects

- material flows: continuous or discrete
- facility deployment: individual or fleet-based
- regional modeling

Some codes currently in use today:

- VISION (INL)
- NUWASTE (NWTRB)
- CAFCA (MIT)
- DANESS (ANL)
- NFCSim (LANL)
- COSI (CEA)
- MARKAL (BNL)

... to name a few.

Introduction : VISION

VISION has surfaced as the industry standard simulator. It is well represented in the literature and can model most aspects of the fuel cycle. [3]

- continuous material flows
- fleet-based facility deployment
- some regional modeling capability
- input/output via Excel
- simulation engine via Powersim

However, Powersim has its limitations...

- variables as a function of software level... did you get the academic or professional version?
- it's (relatively) expensive
- it's third party software

Development : From GENIUS to Cyclus

The predecessor of Cyclus was a simulator named GENIUSv2 [2]

- also written in C++
- similar structures (regions - institutions - facilities)
- designed to overcome other simulator's difficulties
- not designed to proliferate a community

Development : Developer Environment

Cyclus is designed to accommodate any kind of user or developer by allowing flexible participation [1]

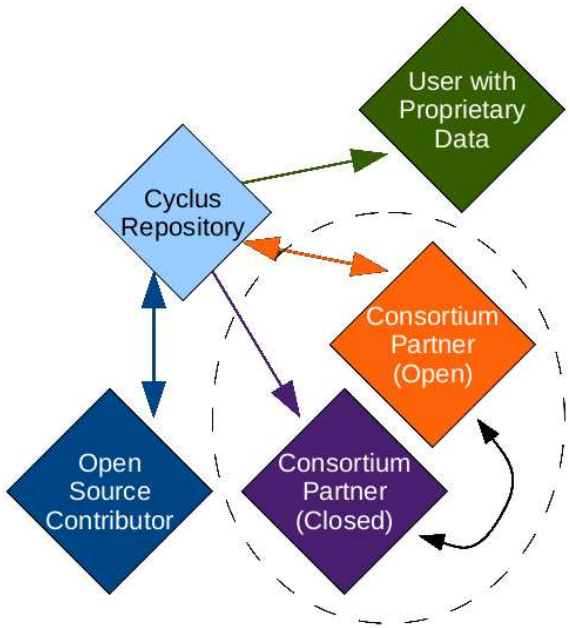


Figure: The Cyclus Participation Paradigm

Development : Code Repository

The Cyclus code repository is publicly available on GitHub.

General information can be found at cyclus.github.com.

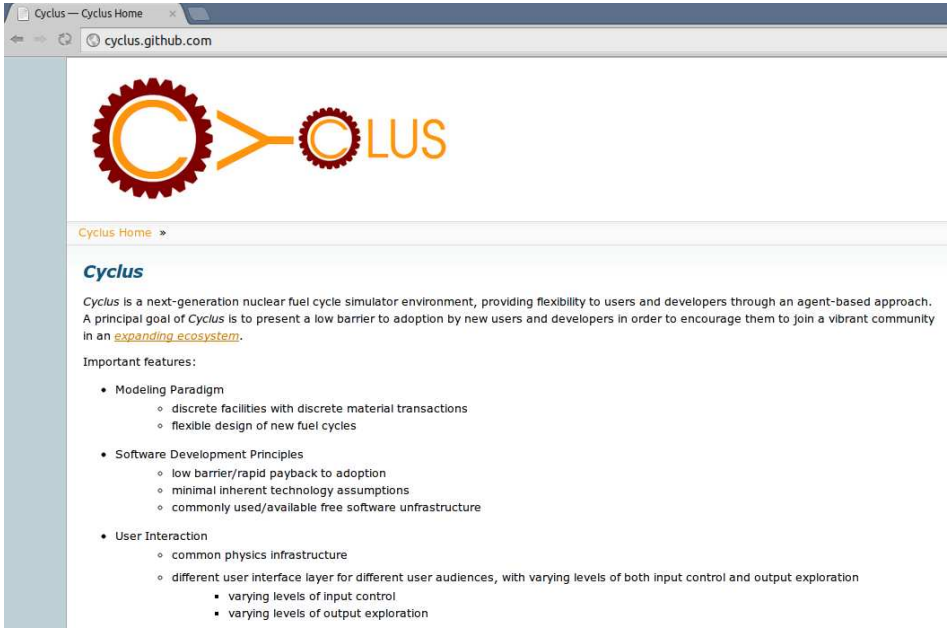


Figure: The Cyclus Github Landing Page

Development : Code Repository

The Cyclus code repository is publicly available on GitHub.

The Cyclus source code can be found at github.com/cyclus/core.

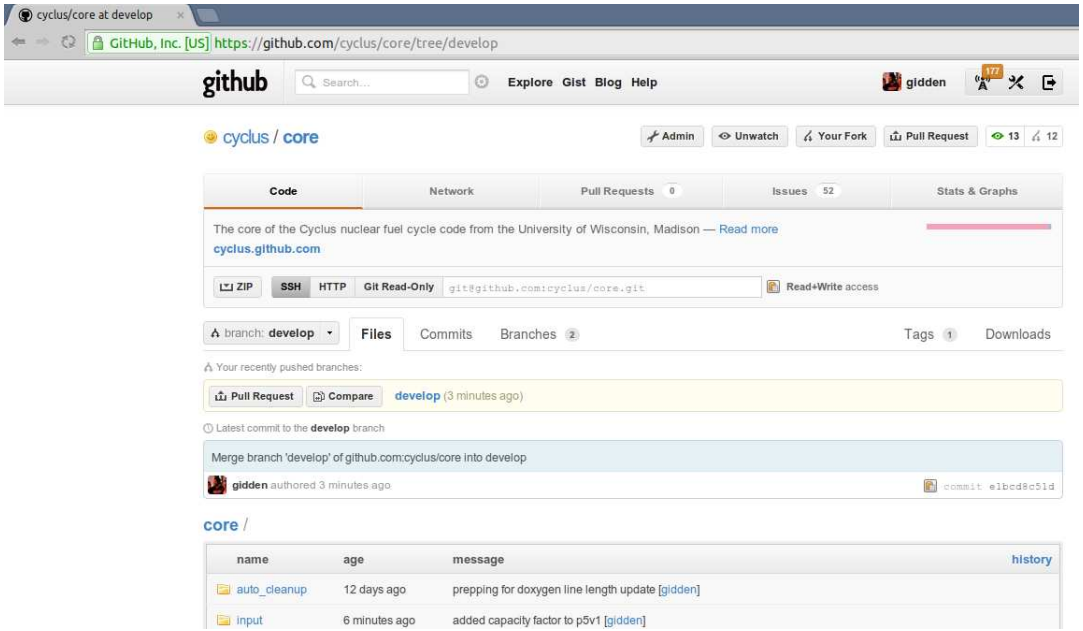
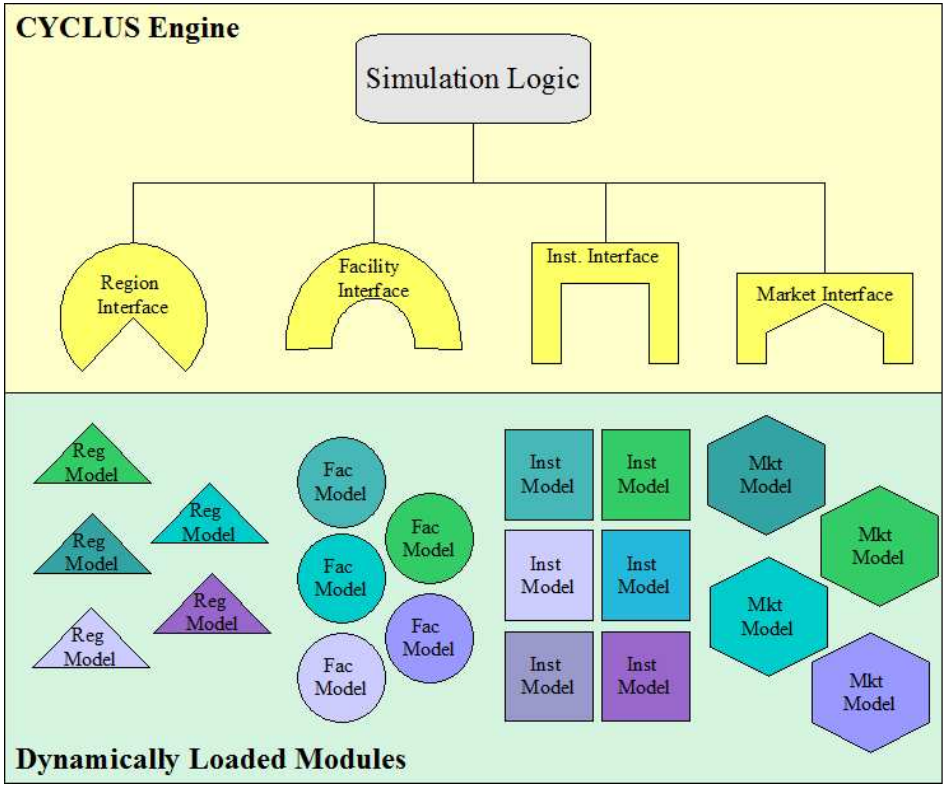


Figure: The Cyclus Source Code Repository

Development : Model Modularity

Combined with the availability of the Cyclus engine source code, modularity of models in Cyclus plays a pivotal role in reducing its barriers to entry.

The Cyclus engine defines a Model's API, to which all models must conform. This allows models of each archetype to be interchangeable.



Engine : Parent-Child Relationship and Time Steps

In the Cyclus Engine, Models communicate messages and commands through their parent-child relationship.

- all models have parents
- all children are 'owned' by their parent
- the region-institution-facility relationship is modeled through this relationship

Simulation time steps are decomposed into:

- tick: system requirements determined
- resolve: optimized solution of requirements found
- tock: solution executed

Engine : Black-Box Paradigm

Facilities are treated as black boxes, i.e. only what comes in and out is recorded.

- facilities can be thought of as nodes: sources, sinks, or intermediaries
- material is passed between resources as a combination of:
  - commodities (e.g. fresh/spent fuel)
  - recipes (e.g. 51 GWd/tHM burnup)
  - quantity

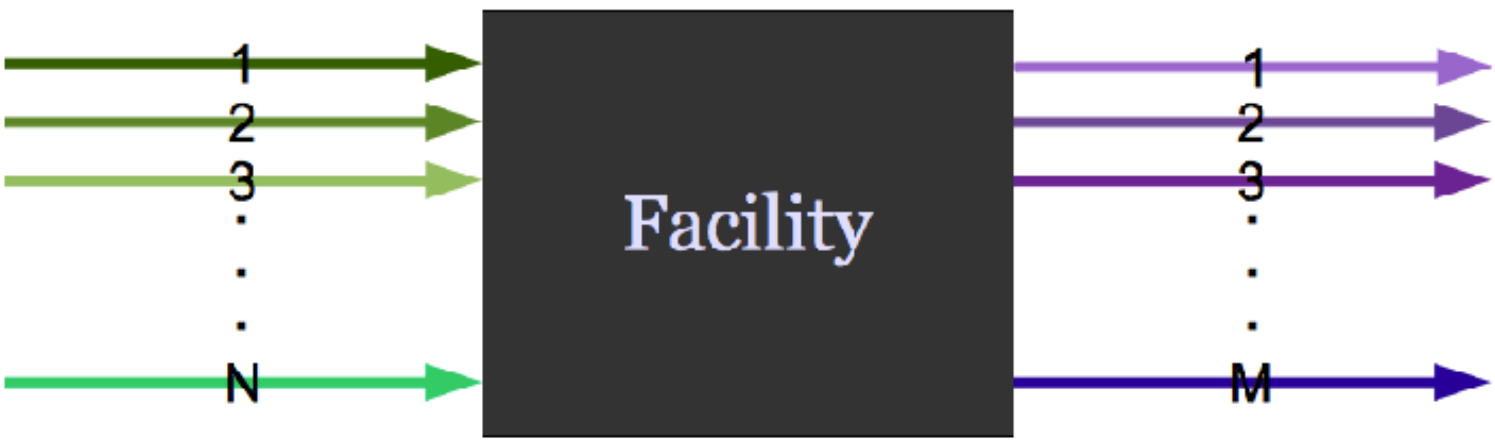


Figure: Facilities are Black Boxes

Benchmarks : Overview

This suite of benchmark problems is designed to test Cyclus' build directive and gross material flows.

VISION is not a discrete material code, thus approximations must be made. As per Kyle Oliver's masters thesis, the following parameters are used:

Reactor Parameters for Benchmark Problems

Parameter	Value
Construction + License Time	6 years
Operation Time	60 years
Power Capacity	1050 MWe
Capacity Factor	0.9
Cycle Time	12 months
Burnup	51 GWd/tHM
Batches per Core	5

Table: Full reactor parameters for the four benchmark problems

Note that using these parameters, a full core loading is 1.273 tHM.

Benchmarks : Specifics

Each simulation begins in the year 2000 and is run for 100 years. The decay module in Cyclus has been turned off in order to compare results with given data.

- Problem 1 - 1 Reactor
  - A single reactor is ordered in 2000, decommissioning at the end of its lifetime
- Problem 2 - 10 Reactors
  - 10 such reactors are ordered
- Problem 3 - Linear Growth
  - 1 reactor per year is built
  - Note: 2 reactors are built in years that old reactors retire

Facility Modules

Three types of facilities are used in the each benchmark:

- Source & Sink Facilities
  - Simple Facilities – already existed
  - Offer or request a certain amount of a commodity each time step
- Batch Reactor Facility
  - Developed for these benchmarks
  - Beginning Phase: Requests a full core
  - Operation Phase: Sits until cycle complete
  - Refuel Phase: Ejects a batch and requests a batch
  - Ending Phase: Ejects a full core

Results : Problem 1

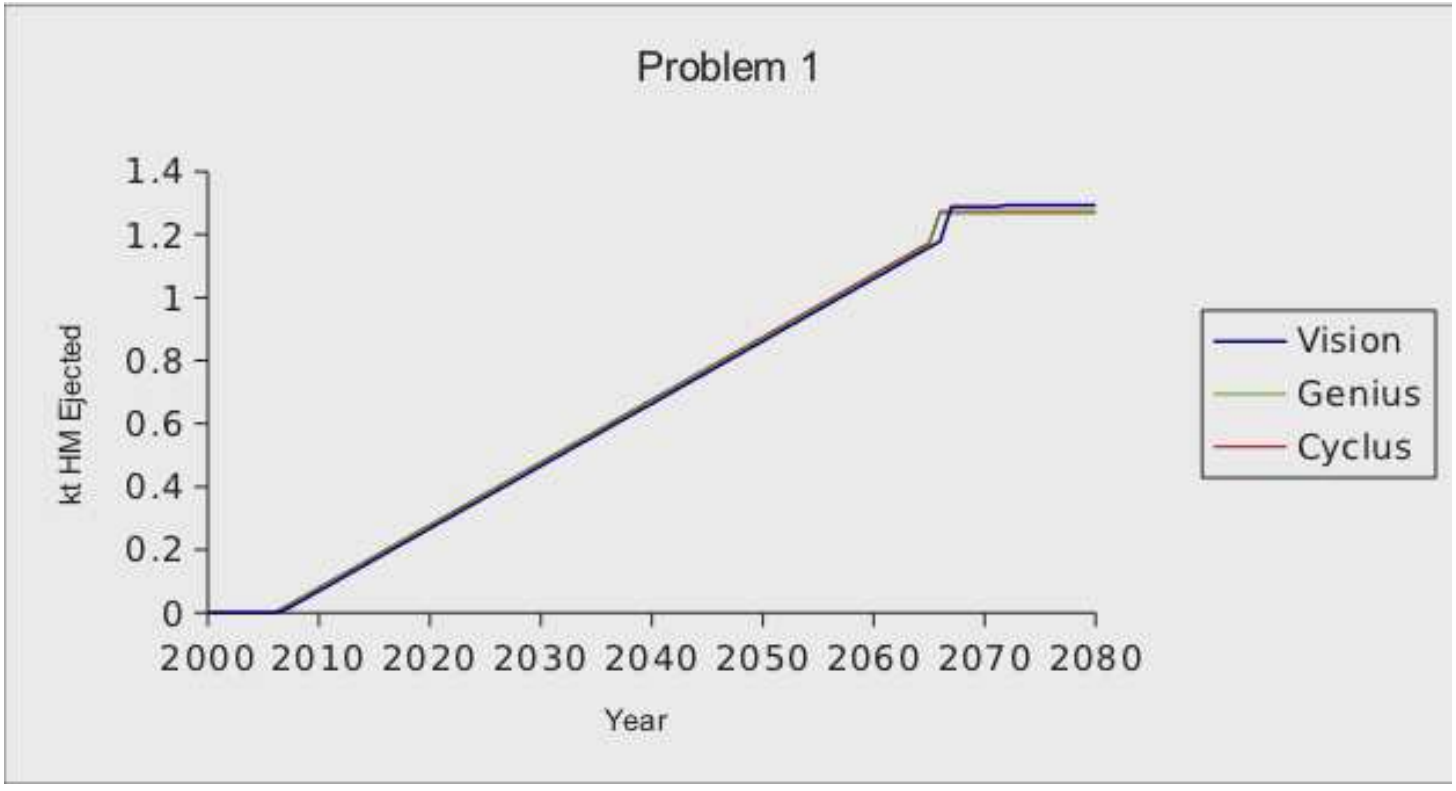


Figure: Problem 1 Results

Results : Problem 2

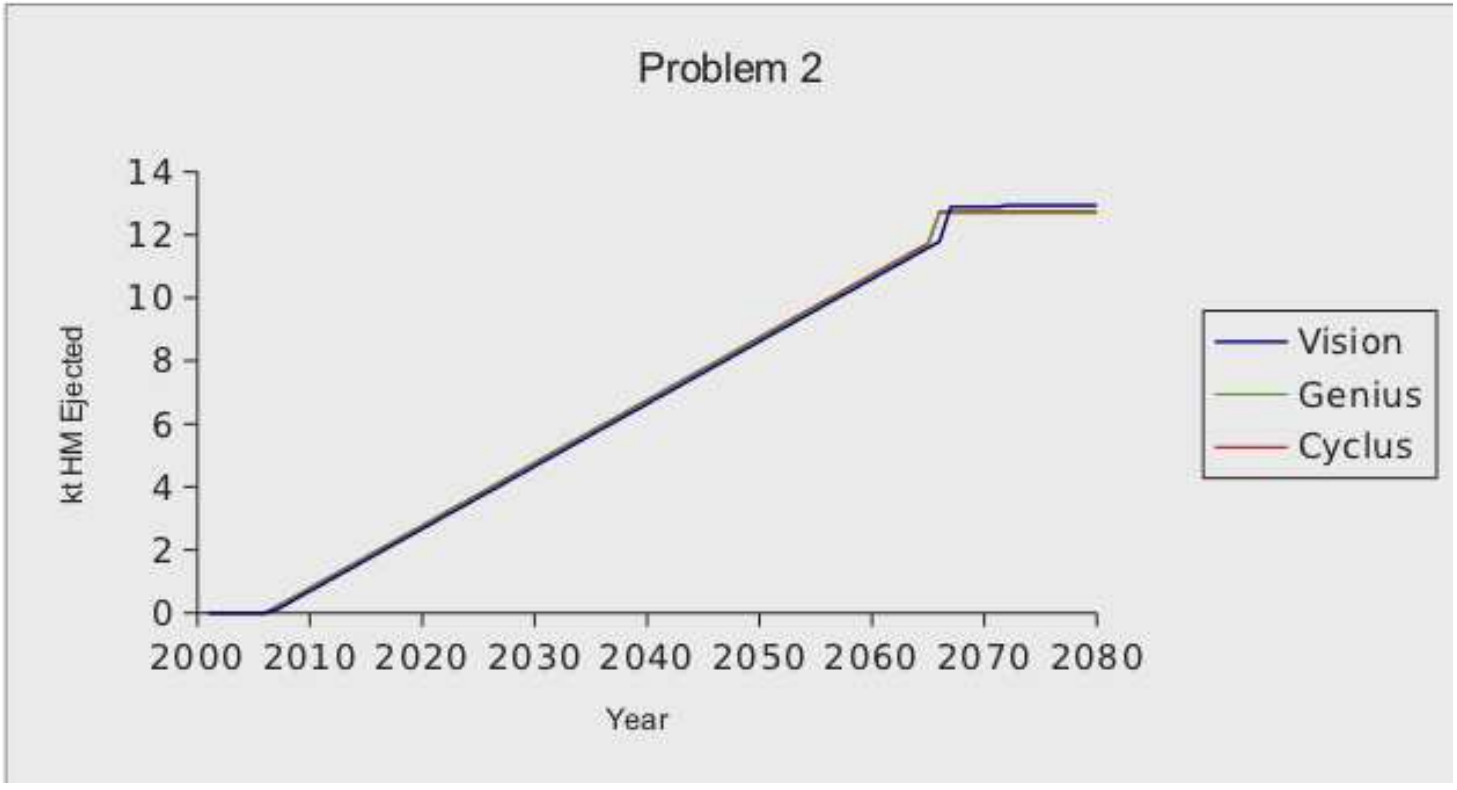


Figure: Problem 2 Results

Results : Problem 3

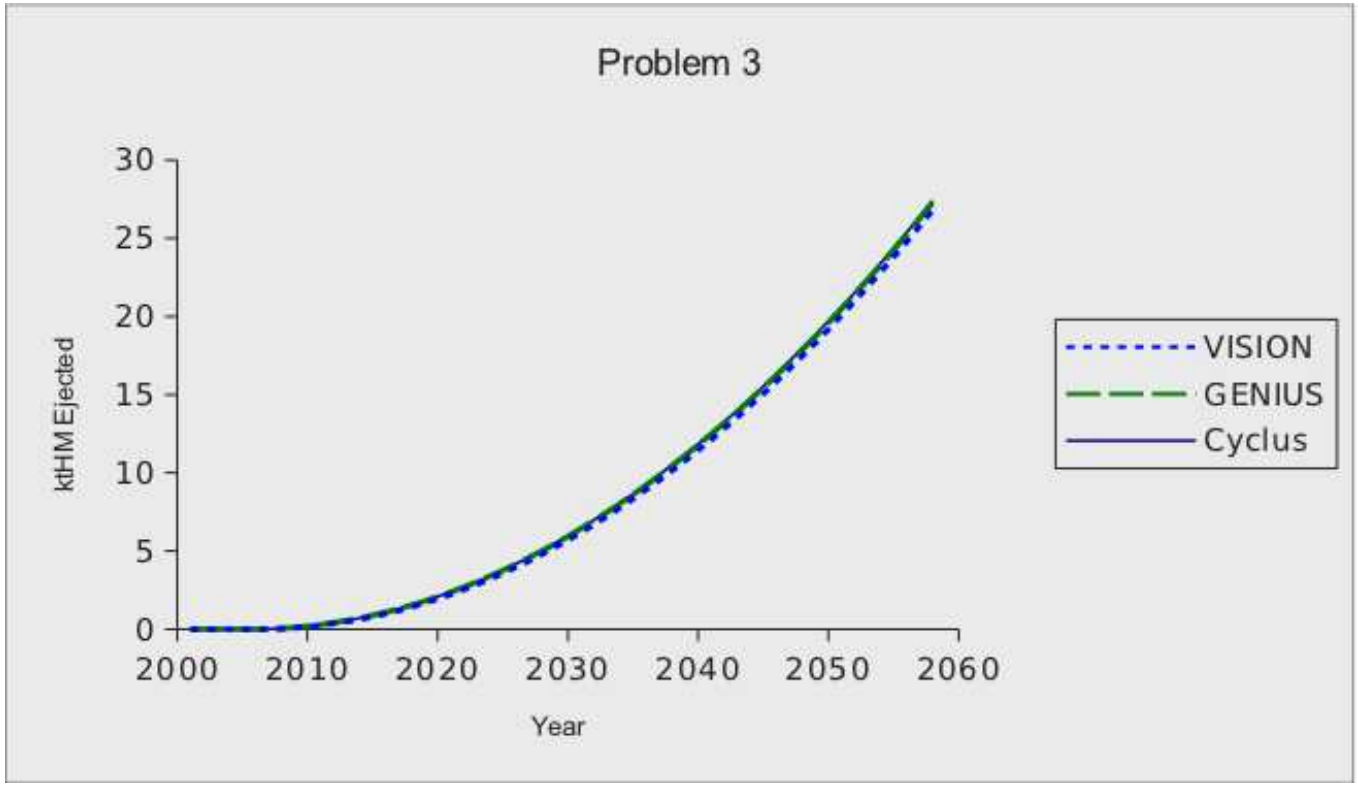


Figure: Problem 3 Results

Results : Future Work

- ... lots?
- memory leaks & efficient logging
- input/visualization team
- remaining VISION once-through benchmarks
- release 0.2!

Acknowledgments

First and foremost, I would like to thank the members of the Computational Nuclear Engineering Research Group at UW

- Katy Huff
- Robert Carlsen
- Prof. Paul Wilson

And, of course, I would like to thank the DOE Office of Nuclear Energy's Nuclear Engineering University Programs for providing funding.

NEUP Nuclear Energy University Programs

U.S. Department of Energy

References

[1] K. Huff.
An integrated used fuel disposition and generic repository model.
Technical report, University of Wisconsin, Madison, WI, 2011.

[2] K. M Oliver.
GENIUSv2: software design and mathematical formulations for multi-region discrete nuclear fuel cycle simulation and analysis.
Master's thesis, UNIVERSITY OF WISCONSIN, 2009.

[3] A. M. Yacout, J. J. Jacobson, G. E. Matthern, S. J. Piet, D. E. Shropshire, and C. Laws.
VISION – verifiable fuel cycle simulation of nuclear fuel cycle dynamics.
In *Waste Management Symposium*, 2006.