1 Introduction

The basic premise of this project is to create a quick, modular reactor depletion model for various reactors in Cyclus. This is done by using two codes, RAVEN [**?**] and SERPENT [**?**] . SERPENT is a monte-carlo reactor physics burnup calculation code, and RAVEN is a parametric and probabilistic analysis tool. The goal of this project is to create an infrastructure to implement a RAVEN Reduced-Order Model (ROM) into a Cyclus reactor module to do depletion calculations of the fuel, and calculate the keff of the core.

In this development, the Molten Salt Breeder Reactor (MSBR) [**?**] reactor ROM is generated.

Thrust 1. Generate SERPENT output files in varying input space

To generate a ROM, large amounts of data tends to make the ROM more accurate. Before we start, the following is the input and output space for the SERPENT ROM:

Input Space	'Fresh' Composition , Depletion time
Output Space	'Depleted Composition', BOC k_{eff}, EOC k_{eff}

Luckily, Andrei has been working on on-line reprocessing in MSBR and already running multiple SERPENT runs (!!!Make this less bro-y). We obtained the SERPENT input and output data from Andrei and converted into a csv file, with the given input and output space. (hdf5_to_csv.py) Table **??** lists the datasets in the hdf5 generated by Andrei, with details. Note that the depletion time in Andrei's SERPENT runs are kept constant at three days. The hdf5 file is curated to a csv file to the form shown in **??**.

Aside: SERPENT-RAVEN interface

By creating the SERPENT-RAVEN interface, RAVEN can run SERPENT with varying parameters to generate SERPENT run data. For example, a user can choose a variable (depletion time, U-233 Composition, etc.) and have RAVEN choose its value by sampling from a distribution. In a RAVEN run, RAVEN will run multiple runs of SERPENT, save the data in csv, dump all the outputs of SERPENT in a separate directory. An example of this is shown in:

Thrust 2. Generate ROM of SERPENT

Thrust 3. Validate ROM calculation with SERPENT runs

Thrust 4. Implement ROM to Cyclus Module

Andrei's data	Details
core adensity after reproc	The input for SERPENT run (the composition that gets depleted)
core adensity before reproc	The depleted composition of previous core adensity after reproc
keff BOC	keff of 'core adensity after reproc'
keff EOC	keff of 'core adensity before reproc'
Th tank adensity	Composition in thorium tank
iso codes	isotope codes array
noble adensity	noble gases composition
tank adensity	composition in Pa tank

Curated dataset	Method
'Fresh' composition	all but the last element in core adensity after reproc
Depletion time	array of 3's with length of keff BOC
BOC keff	BOC keff
Depleted composition	all but the first element in core adensity before reproc
EOC keff	EOC keff