Kraken workshop

SMR modelling with Kraken (Ants – Kharon – SuperFINIX)

21/04/2024 VTT – beyond the obvious

A look at calculation models
(PHYSOR_2024_LDR_lite_model.tgz)

Calculation models

- Ants model. https://serpent.vtt.fi/kraken/index.php/Ants_input_manual
  - 01_inputs/Ants.inp
  - 01_inputs/ants_includes/
    - core.inc
    - fuel_assemblies.inc
    - control_rods.inc
    - radial_reflector.inc
  - 01_inputs/ants/gc/
- Kharon model.
- SuperFINIX model.
- FINIX model.

Calculation models

- Ants model.
- Kharon model. https://serpent.vtt.fi/kraken/index.php/Kharon_input_manual
  - 01_inputs/Kharon.inp
- SuperFINIX model.
- FINIX model.

Calculation models

- Ants model.
- Kharon model.
- SuperFINIX model. https://serpent.vtt.fi/kraken/index.php/SuperFINIX_input_manual
  - 01_inputs/SuperFINIX.inp
- FINIX model.

Calculation models

- Ants model.
- Kharon model.
- SuperFINIX model.
- FINIX model.
  - 01_inputs/finix_files/
    - finix.options
    - finix.rods
    - finix.scenario

https://serpent.vtt.fi/kraken/index.php/File:VTT-R-01103-19_FINIX-1.19.12-manual.pdf

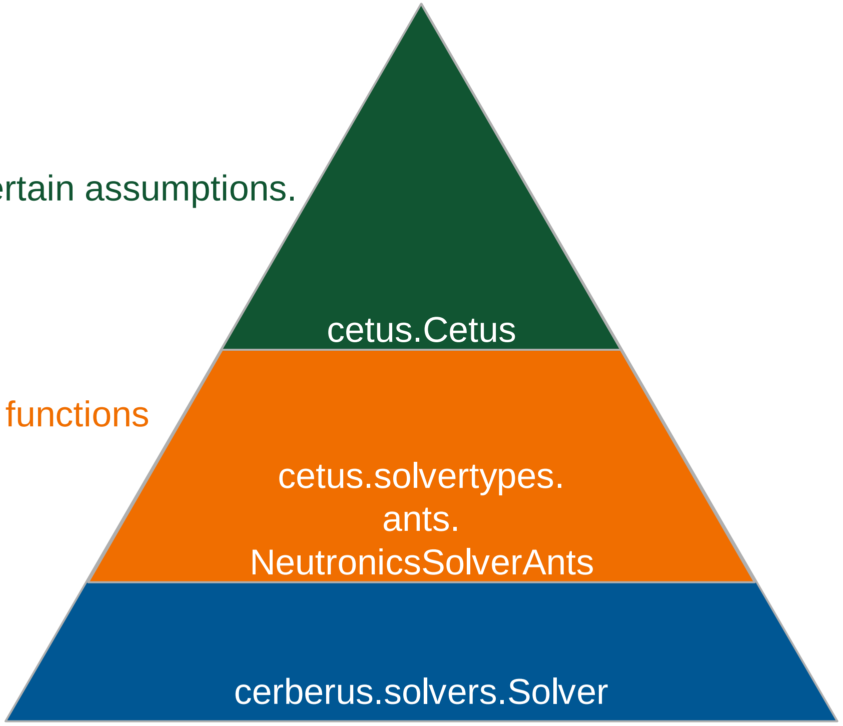
Modelling

Three possible approaches

Automated simulation of fuel cycles with certain assumptions.

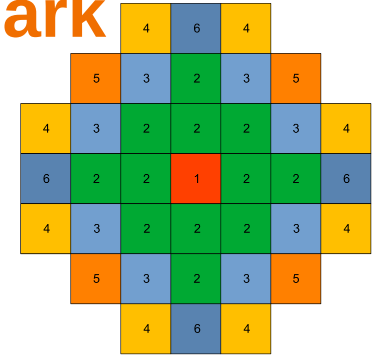
Specific solver used by Cetus, automated for functions required for fuel cycle modelling.

Generic, manual, flexible, can do anything.

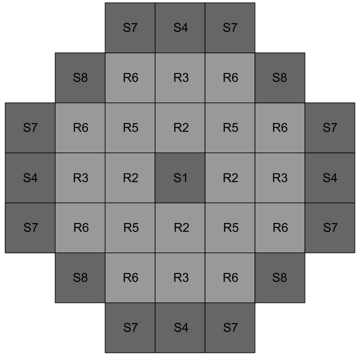


Solving the LDR lite benchmark task 3 (02_LDR_lite_benchmark/)

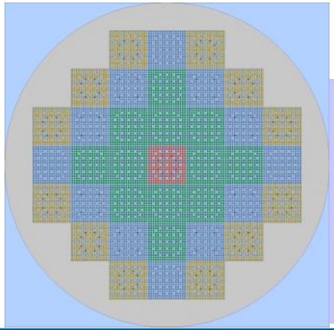
- Evaluate for all-rods-out (300 K):
  - k_{eff}
  - Assembly power distribution
  - Pin power distribution
  - Axial power distribution
  - Control rod group worths

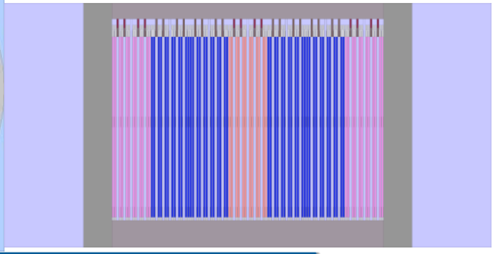


Assembly type	Description
1	1.50 wt-% ^{235}U, 8 rods with 6.00 wt-% Gd_2O_3
2	1.40 wt-% ^{235}U
3	1.80 wt-% ^{235}U, 4 rods with 6.00 wt-% Gd_2O_3
4	2.40 wt-% ^{235}U, 8 rods with 5.00 wt-% Gd_2O_3
5	2.40 wt-% ^{235}U, 8 rods with 9.00 wt-% Gd_2O_3
6	1.80 wt-% ^{235}U



CR group type	Description
RX	Regulating group
SX	Safety group





Generic, manual, flexible, can do anything.

cerberus.solvers.Solver

Solving the LDR lite benchmark task 3 (02_LDR_lite_benchmark/)

- Start up Ants and set state to be simulated
  - Ants_if_fuel_temperature
  - Ants_if_coolant_density
  - Ants_iv_xenon_state
  - Ants_iv_samarium_state
  - Ants_iv_total_power
- Converge Ants neutronics:
  - krakentools.ants.converge_ants()
- Get some results:
  - Ants_ov_keff
  - Ants_of_supernode_power
  - Ants_of_power

Solving the LDR lite benchmark task 3 (02_LDR_lite_benchmark/)

- Start up Ants and set state to be simulated
  - Ants_if_fuel_temperature
  - Ants_if_coolant_density
  - Ants_iv_xenon_state
  - Ants_iv_samarium_state
  - Ants_iv_total_power
- Converge Ants neutronics:
  - krakentools.ants.converge_ants()
- Get some results:
  - Ants_ov_keff
  - Ants_of_supernode_power
  - Ants_of_power

```
Initial keff is 1.08117

Assembly powers:

               0.63 0.82 0.63
          0.74 1.13 1.20 1.13 0.74
     0.63 1.13 1.31 1.35 1.31 1.13 0.63
     0.82 1.20 1.35 1.18 1.35 1.20 0.82
     0.63 1.13 1.31 1.35 1.31 1.13 0.63
          0.74 1.13 1.20 1.13 0.74
               0.63 0.82 0.63


...
```

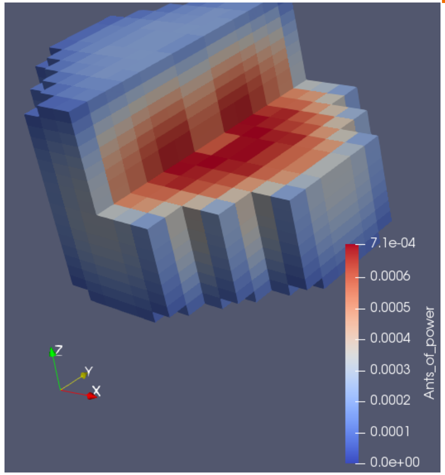
Solving the LDR lite benchmark task 3 (02_LDR_lite_benchmark/)

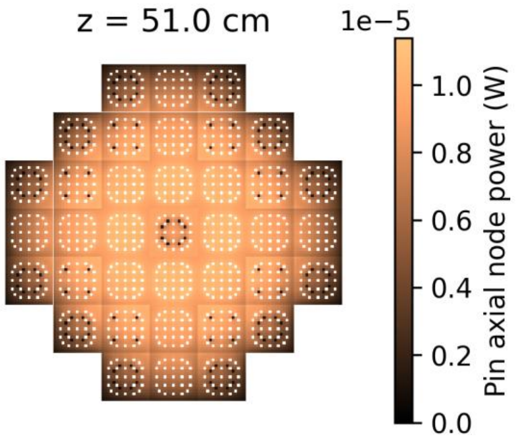
- Move control rods as needed:
  - Ants_iv_cr_bank_height_<name>
- Converge neutronics and get new k_{eff}
  - krakentools.ants.converge_ants()
  - Ants_ov_keff

```
Calculating reactivity worth of group S1...
Reactivity worth was 858.6 pcm

Calculating reactivity worth of group R2...
Reactivity worth was 2175.8 pcm

Calculating reactivity worth of group R3...
Reactivity worth was 2165.7 pcm

Calculating reactivity worth of group S4...
Reactivity worth was 1339.9 pcm

Calculating reactivity worth of group R5...
Reactivity worth was 2445.1 pcm

Calculating reactivity worth of group R6...
Reactivity worth was 3217.2 pcm

Calculating reactivity worth of group S7...
Reactivity worth was 1632.9 pcm

Calculating reactivity worth of group S8...
Reactivity worth was 1205.4 pcm
```

Solving the LDR lite benchmark task 3 (02_LDR_lite_benchmark/)

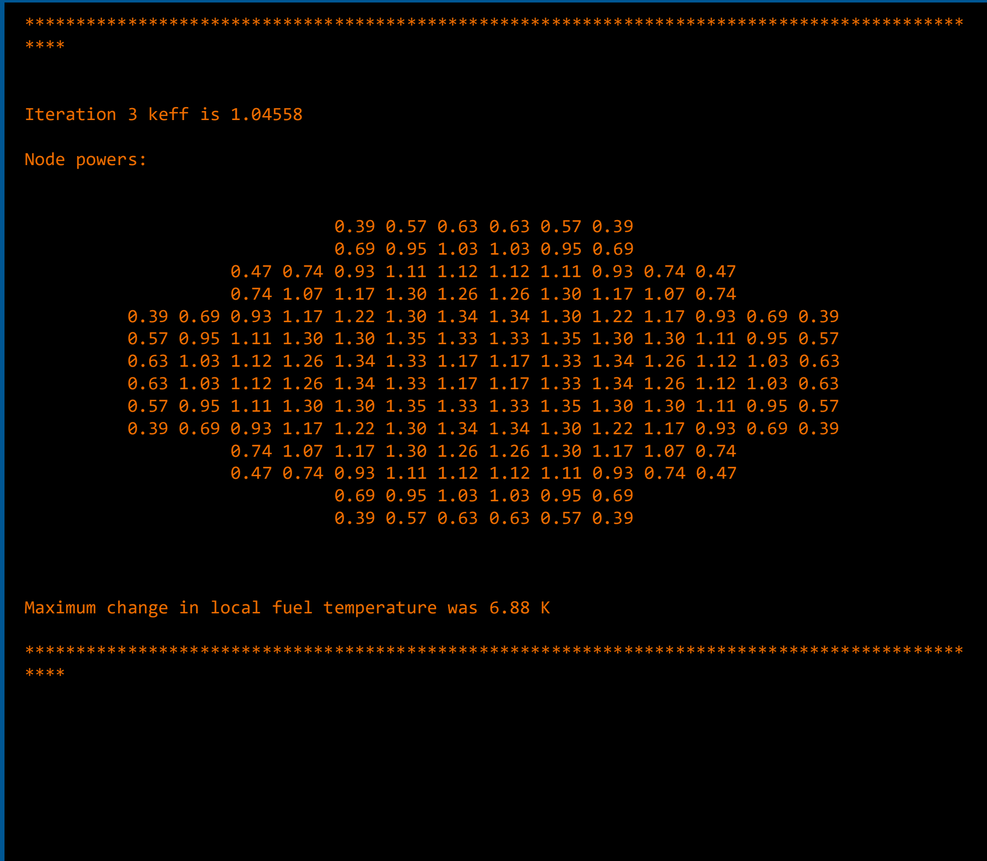
- Fields on Cartesian and hexagonal meshes can be saved into FOAM format for Paraview visualization.
  - transferrable.write_foam()
- Values of fields/variables can be written to files with
  - transferrable.write_simple()
  - transferrable.write_simple(save_binary=False)
- Most of the visualization is left to the user
  - See plot_pinpowers.py





Modelling a powered state 03_coupled_powered_state (a coupled problem)

- Set up and start up:
  - Ants for neutronics.
  - SuperFINIX for fuel behaviour.
  - Kharon for thermal hydraulics.
- Get all of the required fields for the coupled problem:
  - Fuel temperature.
  - Coolant density (and temperature).
  - Cladding temperature.
  - Fission power.
- Set up interpolators between the coupled fields:
  - Automatically generate them based on cell centerpoints due to well built models.
- Set up coupled calculation sequence:
  - Simple Picard iteration with solvers solving in turns and passing data from one to the other.



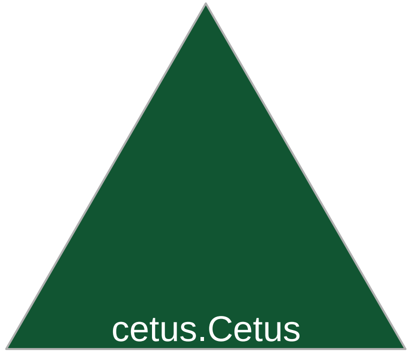
```
**********************************************************************************************
****


Iteration 3 keff is 1.04558

Node powers:


                              0.39 0.57 0.63 0.63 0.57 0.39
                              0.69 0.95 1.03 1.03 0.95 0.69
                    0.47 0.74 0.93 1.11 1.12 1.12 1.11 0.93 0.74 0.47
                    0.74 1.07 1.17 1.30 1.26 1.26 1.30 1.17 1.07 0.74
          0.39 0.69 0.93 1.17 1.22 1.30 1.34 1.34 1.30 1.22 1.17 0.93 0.69 0.39
          0.57 0.95 1.11 1.30 1.30 1.35 1.33 1.33 1.35 1.30 1.30 1.11 0.95 0.57
          0.63 1.03 1.12 1.26 1.34 1.33 1.17 1.17 1.33 1.34 1.26 1.12 1.03 0.63
          0.63 1.03 1.12 1.26 1.34 1.33 1.17 1.17 1.33 1.34 1.26 1.12 1.03 0.63
          0.57 0.95 1.11 1.30 1.30 1.35 1.33 1.33 1.35 1.30 1.30 1.11 0.95 0.57
          0.39 0.69 0.93 1.17 1.22 1.30 1.34 1.34 1.30 1.22 1.17 0.93 0.69 0.39
                    0.74 1.07 1.17 1.30 1.26 1.26 1.30 1.17 1.07 0.74
                    0.47 0.74 0.93 1.11 1.12 1.12 1.11 0.93 0.74 0.47
                              0.69 0.95 1.03 1.03 0.95 0.69
                              0.39 0.57 0.63 0.63 0.57 0.39



Maximum change in local fuel temperature was 6.88 K

**********************************************************************************************
****
```

Fuel cycle simulation with Cetus

(04_fuel_cycle)

- Cetus class handles the simulation
  - cetus = Cetus()
- Add solvers to simulation with chosen options
  - cetus.add_solver()
- Add control rod groups
  - cetus.add_control_rod_group()
- Specify calculation time points (in days)
  - cetus.set_time_points()
- Options for fuel cycle simulation
  - EvalOptions, OutputOptions and CalcOptions
- Interpolation between core physics fields
  - cetus.th_to_nt_cool_density = Interpolator() etc.
- Run simulation
  - cetus.run_simulation()

Cetus handles automated fuel cycle simulations with certain assumptions



Fuel cycle simulation with Cetus

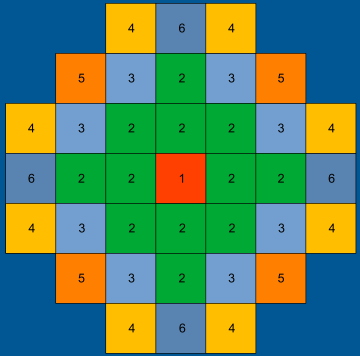
- Some results are: **(04_fuel_cycle)**
  - Written to terminal.
  - Saved to output/results.txt in a human readable format.
  - Saved to output/tabular_results.txt in a tabular ASCII format.
  - Saved to output/<time_folder> as binary files.
  - Saved to foam_output/<mesh_folder>/ as FOAM data.

```
from terminal output:
 Calculating control rod group insertion worths
     - Control rod group S1 insertion worth: 919 pcm
     - Control rod group R2 insertion worth: 2142 pcm
     - Control rod group R3 insertion worth: 2454 pcm
     - Control rod group S4 insertion worth: 1677 pcm
     - Control rod group R5 insertion worth: 2538 pcm
     - Control rod group R6 insertion worth: 4301 pcm
     - Control rod group S7 insertion worth: 2370 pcm
     - Control rod group S8 insertion worth: 1647 pcm
    Finding rod with highest extraction worth
    All groups are already fully extracted
  Calculating uniform Doppler coefficient
     - Uniform Doppler with delta of 10.0 K: -0.8085 pcm/K
     - Uniform Doppler with delta of 20.0 K: -0.8885 pcm/K
```

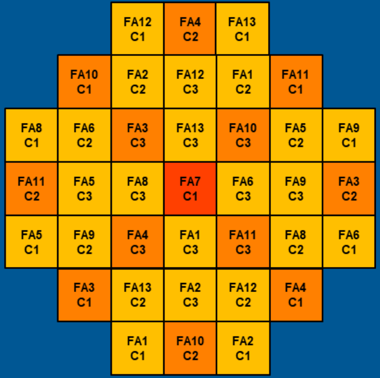
```
from output/results.txt:
Time (days):             480.00000
Time (EFPD):             480.00000
Ave burnup (MWd/kgU):    5.14
Min burnup (MWd/kgU):    0.80
Max burnup (MWd/kgU):    9.56
Keff:                    1.00000
Boron (ppm):             0.000
Power (MW):              4.273E+01
Assembly mflow (kg/s):   6.6000
Outlet pressure (Pa):    6.20000E+05
Ave cool rho (kg/m^3):   947.1
Ave fuel I135 (1/m^3):   1.66176E+21
Ave fuel Xe135 (1/m^3):  9.51686E+20
Ave fuel Xe135m (1/m^3): 0.00000E+00
Ave fuel Pm149 (1/m^3):  2.66164E+21
Ave fuel Sm149 (1/m^3):  1.11091E+22

...
```

Fuel reloading/shuffling (05_shuffling_and_fuel_cycle)

- Reloading/shuffling has not yet been automated with Cetus.
- If assembly types in core loading change, need a new Ants core input.
- Need to move some fields from end of previous simulation to start of next simulation:
  - **Ants_of_burnup → Ants_if_burnup**
  - **Ants_of_number_density_... → Ants_if_number_density**
    - For xenon, samarium and plutonium chains.
  - FINIX restarts are compressed by Cetus to output/final/finix_restarts.tgz:
    - Need to be decompressed to new SuperFINIX working directory and fed to SuperFINIX with **sf_if_rod_idx**
      - SuperFINIX/tests/06_string_shuffle/
      - SuperFINIX/tests/07_fresh_fuel_shuffle/
      - SuperFINIX/tests/08_multiple_fuel_types_shuffle/
- Using reload map shown on the right, reload from end of first cycle and run next cycle.



Assembly type	Description
🟥 1	1.50 wt-% ^{235}U, 8 rods with 6.00 wt-% Gd_2O_3
🟩 2	1.40 wt-% ^{235}U
🟦 3	1.80 wt-% ^{235}U, 4 rods with 6.00 wt-% Gd_2O_3
🟨 4	2.40 wt-% ^{235}U, 8 rods with 5.00 wt-% Gd_2O_3
🟧 5	2.40 wt-% ^{235}U, 8 rods with 9.00 wt-% Gd_2O_3
🟦 6	1.80 wt-% ^{235}U



```
from terminal output:
SuperFINIX restart, y points up

               x    8    x
          x    3    7    1    x
     x   15   36    5   22    9    x
    34   28   14    x   10   24    4
     x   29   16   33    2   23    x
          x   37   31   35    x
               x   30    x
```

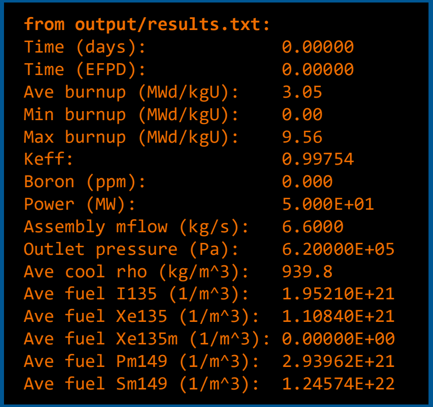
Fuel reloading/shuffling

(05_shuffling_and_fuel_cycle)

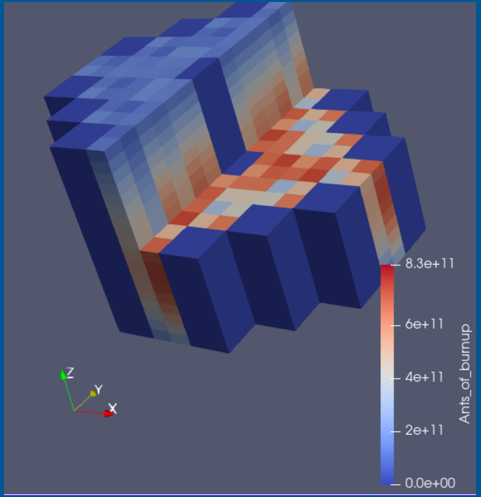
- Reloading/shuffling has not yet been automated with Cetus.

- If assembly types in core loading change, need a new Ants core input.
- Need to move some fields from end of previous simulation to start of next simulation:
  - **Ants_of_burnup → Ants_if_burnup**
  - **Ants_of_number_density_... → Ants_if_number_density**
    - For xenon, samarium and plutonium chains.
  - FINIX restarts are compressed by Cetus to output/final/finix_restarts.tgz:
    - Need to be decompressed to new SuperFINIX working directory and fed to SuperFINIX with **sf_if_rod_idx**
      - SuperFINIX/tests/06_string_shuffle/
      - SuperFINIX/tests/07_fresh_fuel_shuffle/
      - SuperFINIX/tests/08_multiple_fuel_types_shuffle/

- Using reload map shown on the right, reload from end of first cycle and run next cycle.



```
from output/results.txt:
Time (days):                0.00000
Time (EFPD):                0.00000
Ave burnup (MWd/kgU):       3.05
Min burnup (MWd/kgU):       0.00
Max burnup (MWd/kgU):       9.56
Keff:                       0.99754
Boron (ppm):                0.000
Power (MW):                 5.000E+01
Assembly mflow (kg/s):      6.6000
Outlet pressure (Pa):       6.20000E+05
Ave cool rho (kg/m^3):      939.8
Ave fuel I135 (1/m^3):      1.95210E+21
Ave fuel Xe135 (1/m^3):     1.10840E+21
Ave fuel Xe135m (1/m^3):    0.00000E+00
Ave fuel Pm149 (1/m^3):     2.93962E+21
Ave fuel Sm149 (1/m^3):     1.24574E+22
```



Burnups in Ants model after shuffling.
From shuffled/Ants_mesh_node/0.3/
Note burnup values in SI-units
(J instead of MWd)

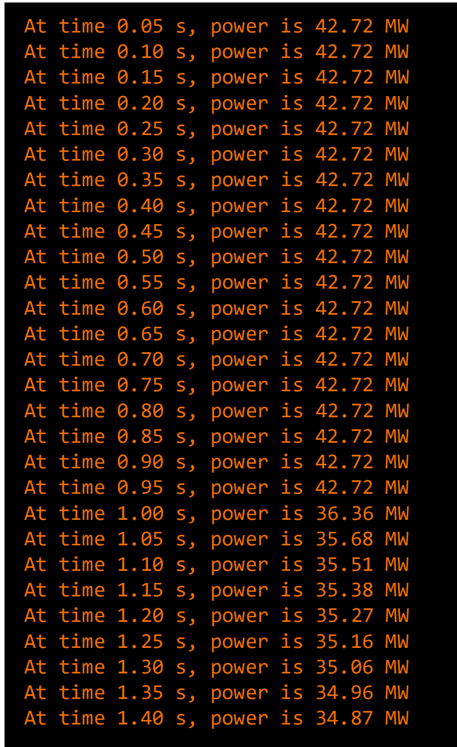
Transient simulation (06_Ants_transient)

- Simply using Ants as a cerberus.solvers.Solver

- Transient restart is loaded using krakentools.ants.load_ants_restart_for_transient()
  - In a general case, a transient restart can be written using krakentools.ants.save_ants_restart()

- Xenon and Samarium states need to be set to something sensible.

- Moving through time handled with
  - solver.set_current_time_interval() for first call
  - solver.advance_to_time_interval() for subsequent calls



```
At time 0.05 s, power is 42.72 MW
At time 0.10 s, power is 42.72 MW
At time 0.15 s, power is 42.72 MW
At time 0.20 s, power is 42.72 MW
At time 0.25 s, power is 42.72 MW
At time 0.30 s, power is 42.72 MW
At time 0.35 s, power is 42.72 MW
At time 0.40 s, power is 42.72 MW
At time 0.45 s, power is 42.72 MW
At time 0.50 s, power is 42.72 MW
At time 0.55 s, power is 42.72 MW
At time 0.60 s, power is 42.72 MW
At time 0.65 s, power is 42.72 MW
At time 0.70 s, power is 42.72 MW
At time 0.75 s, power is 42.72 MW
At time 0.80 s, power is 42.72 MW
At time 0.85 s, power is 42.72 MW
At time 0.90 s, power is 42.72 MW
At time 0.95 s, power is 42.72 MW
At time 1.00 s, power is 36.36 MW
At time 1.05 s, power is 35.68 MW
At time 1.10 s, power is 35.51 MW
At time 1.15 s, power is 35.38 MW
At time 1.20 s, power is 35.27 MW
At time 1.25 s, power is 35.16 MW
At time 1.30 s, power is 35.06 MW
At time 1.35 s, power is 34.96 MW
At time 1.40 s, power is 34.87 MW
```

VTT

beyond
the obvious