Kraken workshop

Introduction to the Cerberus based reactor simulator for fuel cycle analyses

14/05/2022 VTT – beyond the obvious

Outline

- Introduction
- Capabilities of the simulator
- Examples on calculation sequences the simulator automates
- Requirements for running a simulation
- Demo: First operating cycle of a SMR core
- Conclusions

Introduction

- A relatively simple fuel cycle simulator leveraging the Python API Cerberus provides for communicating with individual solvers
- Included in the Cerberus package as a module **cerberus.simulator**
- Utilizes the following solvers:
  - Neutronics: Ants or Serpent
  - Thermal hydraulics: Kharon
  - Fuel behaviour: SuperFINIX
- Demonstrates how common calculation sequences can be easily packaged into their own modules using the Python API
- The simulator has been used mostly in the Finnish district heating reactor project [1,2]

[1] J. Leppänen *et al.* "A Finnish district heating reactor: Background and general overview". Proceedings of ICONE-28, August 4-6, 2021, Virtual Conference, USA.

[2] J. Leppänen *et al.* "A Finnish district heating reactor: Neutronics design and fuel cycle simulations". Proceedings of ICONE-28, August 4-6, 2021, Virtual Conference, USA.

Capabilities of the simulator

- Reactivity control based on:
  - Control rod movement (multiple simple algorithms)
  - Boron in the coolant
  - Inlet temperature
  - Inlet flowrate
  - Reactor power
- Automatic evaluation of:
  - Various reactivity coefficients (Uniform/distributed Doppler, moderator etc.)
  - Control rod worths
  - Shutdown margins
- Switching from Ants based simulation to Serpent based simulation or vice versa requires only minor modifications

Automated calculation sequence: Uniform Doppler coefficient

Steps:

1. Get initial k_{eff} from the neutronics solver
2. Switch critical boron iteration off in the neutronics solver if it was on
3. Fix Xenon distribution in the neutronics solver
4. Store initial temperature field for the neutronics solver
5. Modify the temperature field for the neutronics solver (increase or decrease fuel temperatures by a fixed amount ΔT)
6. Update the neutronics solution
7. Get new k_{eff} from the neutronics solver
8. Calculate uniform Doppler coefficient based on the initial k_{eff}, new k_{eff} and ΔT
9. Restore initial temperature field for the neutronics solver
10. Switch critical boron iteration on if it was on initially
11. Unfix xenon distribution

Automated calculation sequence: Cold shutdown margin

Steps:

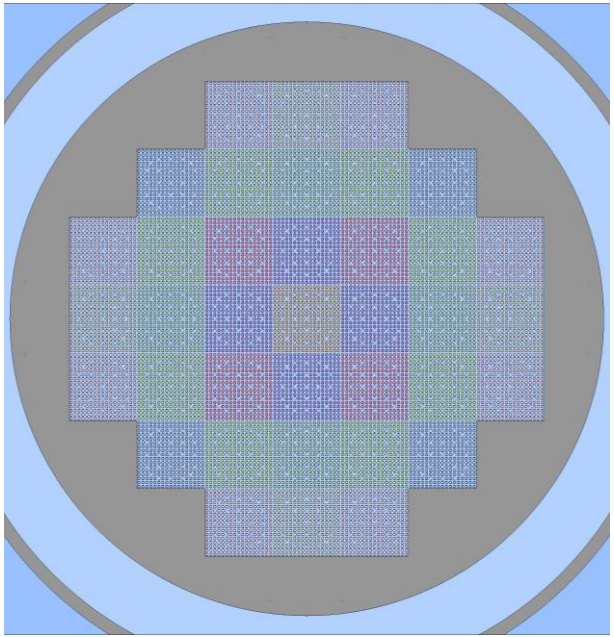
1. Switch critical boron iteration off in the neutronics solver if it was on
2. Store initial Xenon distribution for the neutronics solver
3. Fix Xenon distribution to zero in the neutronics solver
4. Store initial coolant temperature/density and fuel temperature fields for the neutronics solver
5. Set coolant temperature/density and fuel temperature fields for the neutronics solver to cold conditions
6. Store initial positions of the control rod groups
7. Fully insert all control rod groups
8. Find the control rod with the highest worth by repeating the following procedure for each rod x:
    1. Fully extract rod x
    2. Update the neutronics solution
    3. Get new k_{eff} from the neutronics solver
    4. Calculate reactivity and compare to current maximum reactivity
    5. Fully insert rod x
9. Cold shutdown margin is the negative of the maximum reactivity obtained from the previous step
10. Restore initial Xenon distribution for the neutronics solver
11. Restore initial coolant temperature/density and fuel temperature fields for the neutronics solver
12. Restore initial positions of the control rod groups
13. Switch critical boron iteration on if it was on initially
14. Unfix xenon distribution

Requirements for running a simulation

- Solver executables (Ants or Serpent, Kharon, SuperFINIX)
- Input files for each solver
- Interpolation matrices used in transferring field data between the solvers when solving the coupled burnup problem
- A Python script to run the simulation. The following is defined in the script:
  - Solvers/solver input files
  - Interpolation
  - Control rods
  - Depletion steps
  - Convergence criteria for the coupled problem (and neutronics if using Ants)
  - Type of reactivity control
  - Options for reactivity coefficient calculation
  - etc.

Demo: First operating cycle of a SMR core

- Same district heating SMR core model as in the previous Cerberus presentation
- Coupled burnup calculation at nominal power over the first operating cycle up to 2750 EFPD (~ 21.8 MWd/(kgU))
- Linear extrapolation/Linear interpolation as burnup algorithm
- Evaluated at each time point:
  - Reactivity coefficients:
    - UDP: Uniform Doppler temperature variation
    - DDP: Distributed Doppler temperature variation
    - MTC: Uniform moderator inlet temperature variation
    - BOR: Uniform moderator boron content variation
    - POW: Distributed system power variation
  - Shutdown margins:
    - Hot instantaneous
    - Cold long-term
  - Control rod group worths



Summary

- The reactor simulator module is capable of basic modelling of fuel cycles by utilizing VTT's Ants/Serpent, Kharon and SuperFINIX solvers
- The simulator does not read a separate input file and the calculation options are defined in the Python script used to run the simulation
  - The solvers require their own input files!
- The simulator module is included in the Cerberus Python package

beyond
the obvious

Riku Tuominen
riku.tuominen@vtt.fi

@VTTFinland

www.vtt.fi