

Inventory Interrogation

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Inventory files

- ORIGIN concentration file (**F71**)
 - Binary format
 - Generated by depletion codes: TRITON, Polaris, ORIGIN, ORIGAMI
 - Content can be viewed only with Fulcrum and with OBIWAN
 - Useful for data transfer between SCALE sequences
 - Inconvenient for data transfer and inspection for non-SCALE users
- New, additional inventory file: JSON-formatted **inventory interface file (II.JSON)**
 - Simplifies data exchange between SCALE and other codes (such as MELCOR)
 - Allows inspection of data in any text editor
 - Good editors permit collapsing of data groups for easier navigation (e.g., Sublime, Notepad++, Visual Studio Code)
 - Readable with scripting languages such as Python
 - Generated from an F71 file with OBIWAN

Inventory Interface File – Content

An II.JSON file contains the following data:

- Relevant nuclear data for each nuclide:
 - Nuclide ID and element name
 - Molar mass
 - Atomic and mass number
 - Decay constant
 - Decay energy
- Each **case** in a file contains the following data:
 - Nuclide-wise inventory (default: in moles) for this case, at one or more time points (default: in seconds)
 - Volume of this case (default: in cm³)
- Each case can have a different time vector and use different units, but consistency is recommended
- A case corresponds to the case on an ORIGEN F71 concentration file; it may correspond to a specific material in a TRITON model, or the material depleted in an ORIGEN or ORIGAMI calculation

Providing relevant nuclear data makes II.JSON a standalone file that can be used to calculate decay heat, mass, activity, and other derived quantities from the number of atoms/moles of each nuclide.

Generation of II.JSON files



Inventory Interface File – Generation

```
obiwan view -format=ii.json -allow_s="[2,3]" my.f71 > my.ii.json
```

- II.JSON files are generated with OBIWAN's view mode
- The data are printed on screen but can be re-directed to a file via ">"
- Additional options can be used:
 - **allow_s** allow only sublibs, e.g., "-allow_s=[2,3]"
 - **allow_i** allow only isomeric states, e.g., "-allow_i=[1]"
 - **allow_z** allow only atomic numbers, e.g., "-allow_z=[92]"
 - **allow_a** allow only mass numbers, e.g., "-allow_a=[135,136]"
 - **min_z** allow only atomic numbers greater or equal
 - **max_z** allow only atomic numbers less than or equal
- Example: Limit to sublibraries 2 and 3 for II.JSON files for MELCOR to avoid consideration of light nuclides from the fuel (e.g., O or C) to be included

Hands on: II.JSON file generation

- Use OBIWAN to generate an II.JSON file of your TRITON F71 file
 - Pipe the output into a file with ending “ii.json”
 - Apply option **-allow_s="[2,3]”**
- Repeat the above with an F71 file from an ORIGEN decay calculation
- Open the F71 file with Fulcrum and the II.JSON file with a text editor:
 - How many cases do you find? What are the case numbers?
 - Which units are used for the nuclide amounts and the time array?
 - Are the time arrays consistent among all cases?

Inventory visualization



Interrogation of ORIGEN F71 files with Fulcrum

- Data from ORIGEN F71 files can be visualized with Fulcrum (we already did this during this training)
- Benefits:
 - Quick inspection of data on file
 - Different units for nuclide amounts and for the time
 - Flexible selection of nuclides/elements including selection via ranking
- Limitations:
 - Limited plot capabilities (curves, markers, but no stacked plots, no mixed linear/logarithmic scales)
 - No capability for determining differences between data
 - Many clicks are needed to optimize each plot (no “memory” of plot settings)
 - Later modifications to a plot require saving plot as SPF, but addition of data later is not permitted

Interrogation of II.JSON files with Notebooks

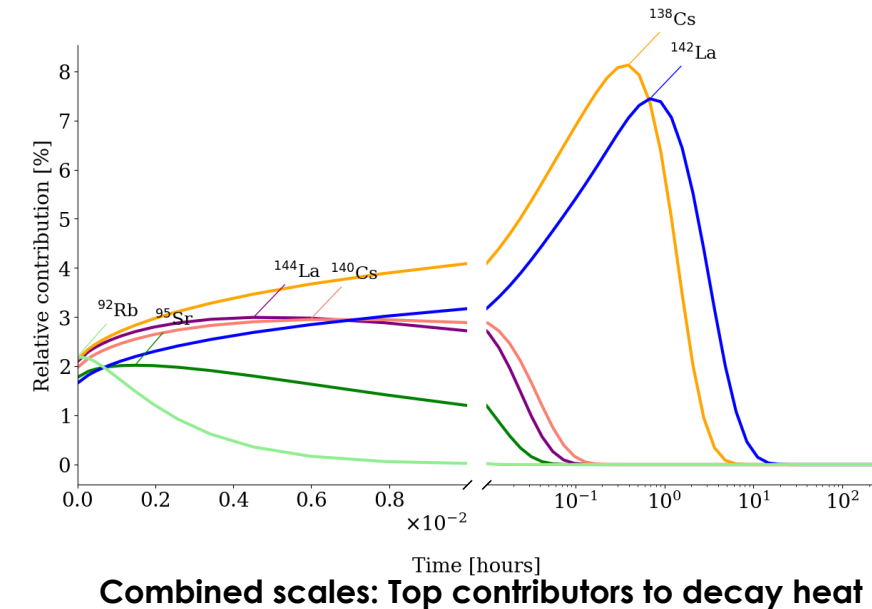
- Data from II.JSON files can be visualized with Jupyter Notebooks (a web-based interactive development environment) that we developed and share in our public repository*
- Using an existing python3 environment, the Jupyter Notebooks can be run in a browser and themselves show minimal Python code
 - You specify your files, choose your settings, and then click to generate plots
 - The Notebooks use “hidden” Python scripts that are stored elsewhere
- Using notebooks does not require solid programming skills, just some common sense and the patience for some trial-and-error
- Benefits:
 - One Notebook for each analysis
 - Simple reproducibility of plots if you want to change settings later
 - One big file in which all information is stored instead of trying to find the plots spread throughout directories and identifying which F71 was used to generate them

*Public repository: code.ornl.gov/scale/analysis/non-lwr-models-vol3/

Interrogation of II.JSON files with Notebooks

Our Jupyter Notebooks support:

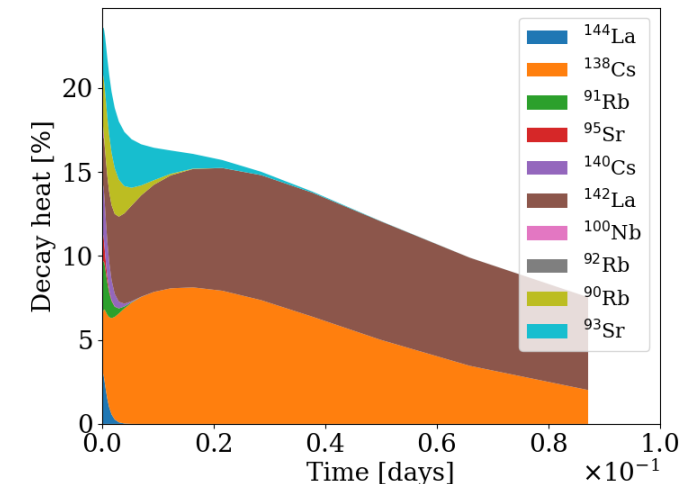
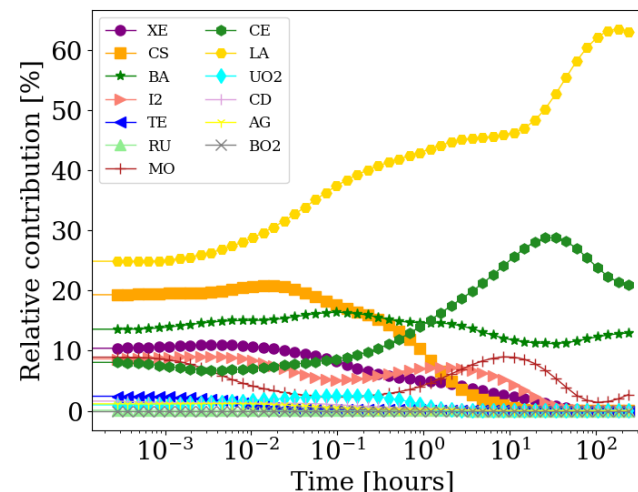
- Plot decay heat and nuclide amounts in different units
- Plot differences between data sets
- Different types of plots (curves, stacks, different scales) with flexible setting manipulation
- Selection of data based on top contributions, if desired
- Generation of MELCOR DCH files



Top 5 isotopes from t=0.000e+00 to t=1.000e+01 days:

no	nuclide	W	% total
1	La140	2.85e+05 W	4.74 %
2	Cs138	2.20e+05 W	3.66 %
3	La142	1.84e+05 W	3.07 %
4	Y94	1.39e+05 W	2.31 %
5	Sr93	1.31e+05 W	2.18 %

Top contributing isotopes to decay heat over a specified time range



Hands-on: ll.JSON file Interrogation – Decay heat

- Navigate to directory “jupyter_notebooks” and start a Jupyter Notebook session
- Open the “DecayHeatAnalysis” Notebook:
 - Note where the path to the directory with the Python scripts is provided
 - Note where the ll.JSON file is provided
 - Why do you think the provided mass is not equal to 1?
 - What do you think the power is used for?
 - What is the difference between printing top contributors to decay heat with no time provided, doing so with a specific time provided, and doing so with a time interval provided?
 - Check out the different types of plots that are available
- Try to execute the different cells top to bottom – check whether everything works

Hands-on: II.JSON file Interrogation – Fuel inventory

- Open the “FuelInventory” Notebook:
 - Note where the path to the directory with the Python scripts is provided
 - Note where the II.JSON file is provided
 - What is the scaling factor used for? For which nuclide amount unit is the scaling factor relevant?
 - Check out the different types of plots that are available
- Try to execute the different cells top to bottom – check whether everything works

Hands-on: H.JSON file Interrogation – Decay heat

- Make a copy of the “DecayHeatAnalysis.ipynb” file and save it with a new name
- Load your H.JSON file from the ORIGEN decay calculation
 - Modify the mass and power adequately
- Identify the top 5 actinide and fission product contributors to the decay heat
 - In the first 5 seconds of the decay
 - In the last 5 seconds of the available time interval
- Create different plots with these identified isotopes

Interrogation with
your own files

Hands-on: HJSON file Interrogation – **Fuel inventory**

- Make a copy of the “FuelInventory.ipynb” file and save it with a new name
- Load your HJSON file from the TRITON depletion calculation
 - Modify the scaling factor adequately
- Create different plots:
 - Actinides: U-235, U-238, Pu-239, Pu-240, Pu-241, Pu-242
 - Fission Products: Xe-135, I-131, Cs-137, Sr-90, Kr-85, Nd-148, Gd-154

Interrogation with
your own files