JADE

Release v1.4.0

Davide Laghi

Oct 29, 2021

JADE USER GUIDE:

1 JADE in a nutshell **3**

2 Installation **5**

3 Folder Structure **7**
- 3.1 Benchmark inputs . . . 8
- 3.2 Code . . . 8
- 3.3 Configuration . . . 9
- 3.4 Experimental results . . . 9
- 3.5 Quality . . . 9
- 3.6 Tests . . . 9
- 3.7 Utilities . . . 9

4 Configuration **11**
- 4.1 Main Configuration . . . 11
- 4.2 Activation File . . . 14
- 4.3 Benchmark run configuration . . . 14
- 4.4 Benchmark post-processing configuration . . . 14

5 Usage **19**
- 5.1 Quality check menu . . . 20
- 5.2 Computational Benchmark menu . . . 20
- 5.3 Experimental Benchmark menu . . . 21
- 5.4 Post-processing menu . . . 22

6 Default Benchmarks **25**
- 6.1 Computational Benchmarks . . . 25
- 6.2 Experimental Benchmarks . . . 42

7 Post-Processing Gallery **49**
- 7.1 Excel output . . . 49
- 7.2 Plots Atlas . . . 53

8 Utilities **59**
- 8.1 Print available libraries . . . 59
- 8.2 Restore default configurations . . . 59
- 8.3 Translate an MCNP input . . . 60
- 8.4 Print materials info . . . 62
- 8.5 Generate material mixture . . . 63
- 8.6 Switch material fractions . . . 63
- 8.7 Change .ace libraries suffix . . . 64

8.8 Produce D1S-UNED reaction files . . . 64

9 Tips & Tricks **65**
9.1 External Run of a benchmark . . . 65
9.2 Change the plots fontsizes . . . 65

10 Insert Custom Benchmarks **67**
10.1 Insert Custom Computational Benchmark . . . 67
10.2 Insert Custom Experimental Benchmark . . . 68

11 Modify documentation **71**

12 General OOP scheme **73**
12.1 Utilities . . . 74
12.2 Benchmarks generation and run . . . 74
12.3 Post-processing . . . 74

13 JADE Session **75**
13.1 main module . . . 75
13.2 libmanager module . . . 76
13.3 state module . . . 77
13.4 configuration module . . . 79

14 Input Generation **81**
14.1 matreader module . . . 81
14.2 testrun module . . . 87
14.3 inputfile module . . . 89
14.4 parsersD1S module . . . 93

15 Post-Processing **97**
15.1 output module . . . 97
15.2 expoutput module . . . 100
15.3 plotter module . . . 102

16 JADE Testing **105**

17 License **107**
17.1 GNU GPLv3 License . . . 107
17.2 Pyne License . . . 119

18 Contributors **121**

19 List of Publications and Contributions **123**
19.1 Publications featuring JADE . . . 123
19.2 Benchmarks Related Publications . . . 123
19.3 Miscellaneous . . . 124

20 Indices and tables **125**

Version: v1.4.0

JADE is a new tool for nuclear libraries V&V. Brought to you by NIER, University of Bologna (UNIBO) and Fusion For Energy (F4E).

 When using JADE for scientific publications you are kindly encouraged to cite the following papers:

- Davide Laghi et al, 2020, "JADE, a new software tool for nuclear fusion data libraries verification & validation", Fusion Engineering and Design, **161** 112075, doi: https://doi.org/10.1016/j.fusengdes.2020.112075.
- D. Laghi, M. Fabbri, L. Isolan, M. Sumini, G. Shnabel and A. Trkov, 2021, "Application Of JADE V&V Capabilities To The New FENDL v3.2 Beta Release", *Nuclear Fusion*, **61** 116073. doi: https://doi.org/10.1088/1741-4326/ac121a

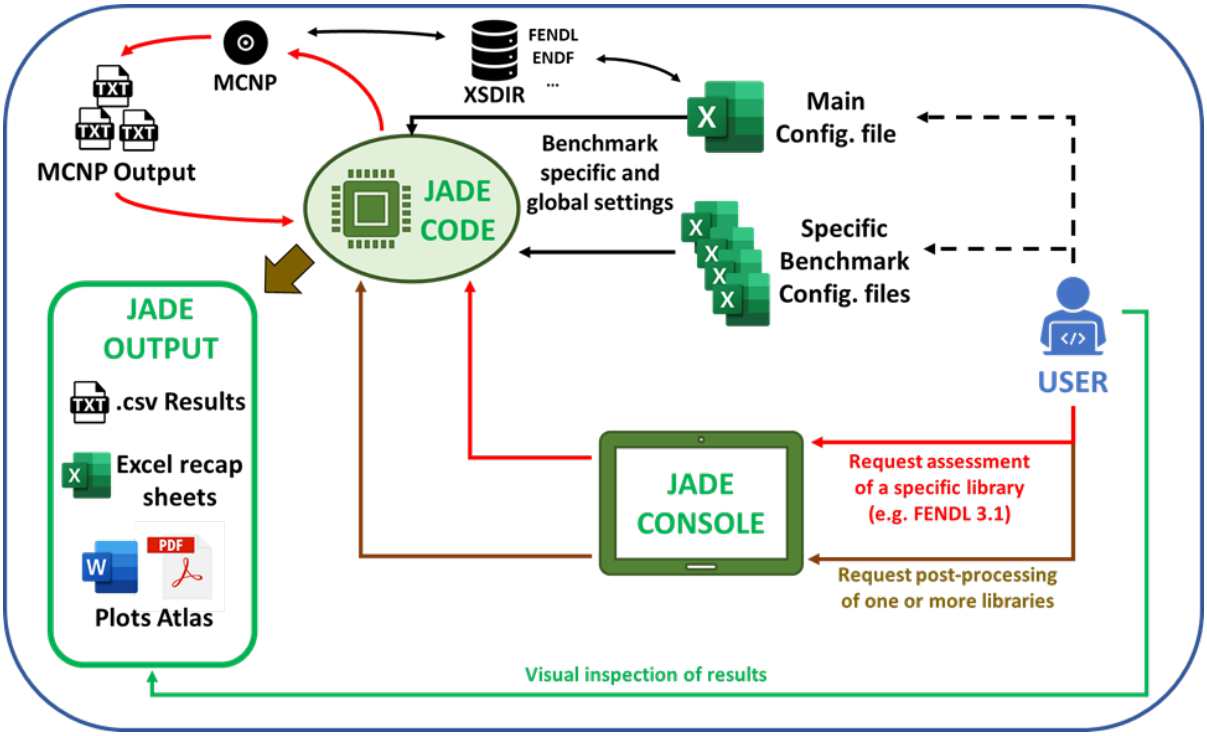
For additional information contact: d.laghi@nier.it

For additional information on future developments please check the issues list on the GitHub repository [link].

CHAPTER

ONE

JADE IN A NUTSHELL



JADE is a new tool for nuclear libraries V&V. Brought to you by NIER, University of Bologna (UNIBO) and Fusion For Energy (F4E). JADE is an open source, Python 3 based software able to:

- automatically build a series of MCNP input file using different nuclear data libraries;
- automatically run simulations on such inputs;
- automatically parse and post-process all the generated MCNP outputs.

The benchmarks implemented by default are divided between computational and experimental benchmarks. The post-processing output includes:

- raw data in .csv files containing the entire tallied output from the simulations;
- formatted Excel recap files;
- Word and PDF atlas collecting the plots generated during the post-processing.

Additional JADE features are:

- the possibility to implement user-defined benchmarks;

- operate on the material card of an MCNP input (e.g. create material mixtures or translate it to a different nuclear data library);
- print a recap of the material composition of an MCNP input.

When using JADE for scientific publications you are kindly encouraged to cite the following papers:

- Davide Laghi et al, 2020, "JADE, a new software tool for nuclear fusion data libraries verification & validation", Fusion Engineering and Design, **161** 112075, doi: https://doi.org/10.1016/j.fusengdes.2020.112075.
- D. Laghi, M. Fabbri, L. Isolan, M. Sumini, G. Shnabel and A. Trkov, 2021, "Application Of JADE V&V Capabilities To The New FENDL v3.2 Beta Release", *Nuclear Fusion*, **61** 116073. doi: https://doi.org/10.1088/1741-4326/ac121a

For additional information contact: d.laghi@nier.it

For additional information on future developments please check the issues list on the GitHub repository [link].

CHAPTER

TWO

INSTALLATION

The preferred way to install JADE is through a conda virtual environment, meaning that an Anaconda or Miniconda installation is required.

1. Extract the zip into a folder of choice (from now on `<JADE_root>`);
2. Rename the folder containing the Python scripts as 'Code' (`<JADE_root>\Code`);
3. Open the global configuration file: `<JADE_root>\Code\Configuration\Config.xlsx`; here you need to properly set the environment variables specified in the 'MAIN Config.' sheet (i.e. xsdir Path, and multithread options);
4. Open an anaconda prompt shell and change directory to `<JADE_root>\Code`. Then create a virtual environment specific for jade:

   ```
   conda env create --name jade --file=environment.yml
   ```

   This ensures that all dependencies are satisfied. The environment can be activated using:

   ```
   conda activate jade
   ```

   And deactivated simply using:

   ```
   conda deactivate
   ```

5. Finally, when the environment is activated, in order to start JADE type:

   ```
   python main.py
   ```

6. On the first usage the rest of the folders architecture is initialized.

See also:

check also *Main Configuration* for additional information on the minimum environment variables to be set.

CHAPTER

THREE

FOLDER STRUCTURE

The following is a scheme of the JADE folder structure:

```
<JADE_root>
    |--------- Benchmark inputs
    |             |----- <inputfile>.i
    |             |----- ...
    |             |----- <Inputfolder>
    |             |----- ...
    |             |----- VRT
    |
    |--------- Code
    |             |----- default_settings
    |             |----- docs
    |             |----- install_files
    |             |----- templates
    |             |----- tests
    |
    |--------- Configuration
    |                 |----------- Benchmarks Configuration
    |                 |----------- Config.xlsx
    |
    |--------- Experimental results
    |                    |------------ <Benchmark name 1>
    |                    |------------ [...]
    |
    |--------- [Quality]
    |
    |--------- Tests
    |             |--------- MCNP simulations
    |             |                  |----------- <Lib suffix 1>
    |             |                  |                  |---------- <Benchmark name 1>
    |             |                  |                  |---------- [...]
    |             |                  |----------- [...]
    |             |
    |             |--------- Post-Processing
    |                                |------------ Comparisons
    |                                |                  |-------- <lib 1>_Vs_<lib 2>_Vs..
↪..
    |                                |                  |                  |-------------
↪----- <Benchmark name 1>
    |                                |                  |                  |             ⌋
↪               |----------- Atlas
    |                                |                  |                  |             ⌋
↪               |----------- Excel
```

(continues on next page)

(continued from previous page)

```
    |                                |                    |                    |
    |                                |                    |                    |-------------
↪----- [...]
    |                                |                    |
    |                                |                    |------- [...]
    |                                |
    |                                |------------ Single Libraries
    |                                                     |----------- <Lib suffix 1>
    |                                                     |                    |----------
↪<Benchmark name 1>
    |                                                     |                    |           ⌟
↪       |----------- Atlas
    |                                                     |                    |           ⌟
↪       |----------- Excel
    |                                                     |                    |           ⌟
↪       |----------- Raw Data
    |                                                     |                    |
    |                                                     |                    |----------⌟
↪[...]
    |                                                     |
    |                                                     |----------- [...]
    |
    |-------- Utilities
```

<JADE_root> is the root folder chosen by the user. As described in *Installation* section, the JADE GitHub repo should be renamed and placed inside the root directory as <JADE_root>\Code.

All folders parallel to the <JADE_root>\Code will be created after the first JADE run.

Hereafter, a general overview of the different JADE tree branches is presented.

3.1 Benchmark inputs

<JADE_root>\Benchmark inputs contains all the inputs of the default benchmarks avaialble in the JADE suite. This is the folder where eventual user defined benchmark inputs should be positioned. In case of benchmarks that are composed by more than one run, all the inputs are reunited in a sub-folder (e.g. <JADE_root>\Benchmark inputs\Oktavian.

<JADE_root>\Benchmark inputs\VRT folder is especially important. Here, in specific subfolders, can be found all additional files required by the benchmark inputs (e.g. weight window files, irradiation files, reactions file, etc.)

3.2 Code

<JADE_root>\Code contains the JADE GitHub repo itself. All the source code is contained here toghether with the following subfolders:

default_settings Contains all JADE default settings. On the first JADE instance these are copied to the <JADE_root>\Configuration folder. They can be restored by a dedicated utility function available from the main menu.

docs Contains all files related to this documentation. Here, local version of the documentations can be found.

install_files Contains files to be used during the first JADE run. They have not any appeal to the general user.

templates Contains all the Microsoft Office and Word templates to be used during post-processing. In case of user-defined benchmarks that are based on specific templates, these need to be added here.

tests Contains all the .py files to be run with pytest and folder containing files useful for the testing activity

3.3 Configuration

`<JADE_root>\Configuration` stores the main JADE configuration file `Config.xlsx` and all benchmark-specific configuration files that are stored in `<JADE_root>\Code\Benchmarks Configuration`.

See also:

Configuration for additional description of the configuration files.

3.4 Experimental results

`<JADE_root>\Experimental results` stores all the experimental results needed for the post-processing of experimental benchmarks. In case of benchmarks that are composed by more than one run, all the inputs are reunited in a sub-folder (e.g. `<JADE_root>\Experimental results\Oktavian`.

3.5 Quality

NOT IMPLEMENTED

3.6 Tests

`<JADE_root>\Tests` reunites all the outputs of the benchmarks assessments.

MCNP simulations contains the results of the transport simulations.

Post-Processing contains all the results coming from the post-processing of results. These are divided between *Comparisons* and *Single Libraries*.

3.7 Utilities

`<JADE_root>\Tests` is where all outputs coming from the *Utilities* are reunited. Each utility generates a dedicated sub-folder when is used for the first time. Upon installation, the only sub-folder is `<JADE_root>\Tests\Log Files` that contains all log files produced by each JADE session.

CHAPTER

FOUR

CONFIGURATION

All configuration files are included in the `<JADE_root>\Configuration` directory. In principle, **the general user should only operate on the** *Main Configuration* **file**, while all other configuration files simply guarantee an additional level of personalization for the user.

Note: In case of user-defined benchmarks suitable *Benchmark run configuration* and *Benchmark post-processing configuration* files need to be produced.

Note: Every time a new D1S library is added to the user xsdir, in order to use it in JADE a specific sheet must be added in the *Activation File*.

4.1 Main Configuration

The most important configuration file is `<JADE_root>\Configuration\Config.xlsx`. This is **the only configuration file that the user must modify** before operating with JADE. Herafter, a description of the different sheets included in the file is given.

4.1.1 MAIN Config.

MAIN CONFIGURATION VARIABLES	
xsdir Path	C:\Users\d.laghi\Documents\MCNP\MCNP_DATA\xsdir_mcnp6.2
multithread	True
CPU	4

This sheet contains the JADE *ambient variables*:

xsdir Path Absolute path to the xsdir file that has been set to be used during MCNP simulations. If different codes are used that use different xsdir file (e.g. mcnp5 and mcnp6) the user should make sure that all libraries of interest are included in the xsdir file indicated in this variable.

multithread Under Windows operative system, MCNP allows to run on multithread using the `tasks` keyword. Setting this variable to `True` enables this capability.

CPU When **multithread** is set to `True`, **CPU** sets the argument that will be passed to `tasks` during MCNP runs.

4.1.2 Computational benchmarks

Computational benchmark additional options

Default Benchmarks

Description	File Name	OnlyInput	Run	Post-Processing	NPS cut-off	CTME cut-off	Relative Error cut-off
Sphere Leakage Test	Sphere.i	false	false	false	1.00E+04		
ITER 1D (by M. Sawan)	ITER_1D.i	false	false	false	1.00E+04		
Helium Cooled Pebble Bed Test Blanket Module (1D)	HCPB_TBM_1D.i	false	false	false	1.00E+03		
Water Cooled Lithium Lead Test Blanket Module (1D)	WCLL_TBM_1D.i	false	false	false	1.00E+03		
C-Model R181031 rev190715	C_Model.i	true	false	true	1.00E+03		

This table collects allows to personalize which *computational benchmarks* should be included in the JADE assessment. Each row controls a different benchmark, where the following options (columns) are available:

Description this is the extended name of the benchmark, this name will appear in specific outputs of the post-processing.

File Name name of the reference MCNP input file. These need to be placed in `<JADE root>\Benchmarks inputs`.

OnlyInput when this field is set to `True` the benchmark input is only generated but not run. This can be useful when the user wants to run the benchmark on a different hardware with respect to the one where JADE is being used.

See also:

External Run of a benchmark

Run the benchmark will be run during an assessment only if this field is set to `True`. This allows to customize the selection of benchmarks to be run during an assessment or avoid to re-run benchmarks that were already simulated in the past.

Post-Processing this field works exactly as the `Run` one but for the post-processing operations.

The last three options available for each benchmark control the MCNP STOP card parameters that help regulating the simulation lenght:

NPS cut-off this is equivalent to the `NPS` entry in the MCNP STOP card. It sets a maximum amount of histories to be simulated. Only integers are allowed.

CTME cut-off this is equivalent to the `CTME` entry in the MCNP STOP card. It sets a maximum computer time after which the simulation will be interrupted. Only integers are allowed.

Relative Error cut-off this is equivalent to the `F` entry in the MCNP STOP card. The sintax of this entry is:

F<*k*>-<*e*> (example: F16-0.0005)

This stops the calculation when the tally fluctuation chart of tally *k* has reached a relative error lower than *e*.

Custom input New in version v1.3.0: This columns allows to provide custom inputs to the different benchmarks. For the moment, this is used only in the *Sphere Leakage* and *Sphere SDDR* benchmarks where, if a number *n* is specified, this will limit the test to the first *n* isotope and material simulations (useful for testing).

Code New in version v1.3.0: This column is needed to specify which kind of code needs to be used for each benchmark. The available codes are defined at the very beginning of the `<JADE root>\Code\testrun.py` by a dictionary linking tags to be used in the config. file and actual names of the executables to be used.

```
CODE_TAGS = {'mcnp6': 'mcnp6', 'D1S5': 'd1suned3.1.2'}
```

Note: All three STOP parameters can be simultaneously defined during a simulation. The first cut-off criteria reached will be the one triggering the end of the calculation.

4.1.3 Experimental benchmarks

Computational benchmark additional options

Default Benchmarks

Description	File Name	OnlyInput	Run	Post-Processing	NPS cut-off	CTME cut-off	Relative Error cut-off
Oktavian Experiment	Oktavian	false	true	true	1.00E+06		

The structure of the sheet is exactly the same as the *Computational benchmarks* one. Nevertheless, in this table are indicated the settings for the experimental benchmarks.

4.1.4 Libraries

Suffix	Name	Default
21c	FENDL 2.1c	
30c	FENDL 3.0	
31c	FENDL 3.1d	
32c	FENDL 3.2 beta	
70c	ENDF VII	
00c	ENDF VIII	yes

This table simply consists of a glossary where the user can associate more explicit names to the nuclear data libraries suffixes available in the xsdir file. This allows for a clearer post-processing output.

Warning: Do not use invalid filename characters (e.g. `"\"`) in the names assigned to the libraries!

4.2 Activation File

Parent	MF	MT	Reaction	Daughter
O16	13	103	(n,p)	N16
F19	13	16	(n,2n)	F18
Na23	13	102	(n,γ)+(n,γ)*	Na24
Al27	13	107	(n,α)+(n,α)*	Na24
Cr50	13	102	(n,γ)	Cr51
Cr52	13	16	(n,2n)	Cr51
Mn55	13	16	(n,2n)	Mn54
	13	102	(n,γ)	Mn56
Fe54	13	103	(n,p)	Mn54
	13	107	(n,α)	Cr51
Fe56	13	103	(n,p)	Mn56
	13	105	(n,t)	Mn54
Fe57	13	28	(n,np)	Mn56
	13	104	(n,d)	Mn56
Fe58	13	102	(n,γ)	Fe59
	13	105	(n,t)	Mn56

The `<JADE_root>\Configuration\Activation.xlsx` file stores all the reactions available in the different versions of the D1S-UNED activation libraries. For each library a sheet needs to be added having as name the suffix used in the xsdir file for the library. Only three columns in the table are mandatory and these are the **Parent**, **MT** and **Daughter** ones.

4.3 Benchmark run configuration

TBD

These are used only for *Sphere Leakage* and cannot be generalized.

4.4 Benchmark post-processing configuration

It is possible to control (to some extent) the post-processing of each benchmark via its specific configuration file. These files are located in the `<JADE_root>\Configuration\Benchmarks Configuration` folder and their name must be identical to the one used in the `File Name` field in the main configuration file (using the .xlsx extension instead of the .i). These files are available only for computational benchmarks, since the high degree of customization needed for an experimental benchmark makes quite difficult to standardize them. While computational benchmarks can be added to the JADE suite without the need for additional coding, this is not true also for experimental one.

The files contain two main sheets, that respectively regulate the Excel and the Word/PDF post-processing output.

4.4.1 Excel

Tally	x	x name	y	y name	cut Y
4	Energy	Energy [MeV]	Cells	Cell	20
14	Energy	Energy [MeV]	Cells	Cell	20
6	Cells	Cell	tally	Value	
16	Cells	Cell	tally	Value	
26	Cells	Cell	tally	Value	
24	Cells	Cell	tally	Value	
34	Cells	Cell	tally	Value	
44	Cells	Cell	tally	Value	
54	Cells	Cell	tally	Value	
64	Cells	Cell	tally	Value	
74	Cells	Cell	tally	Value	
84	Cells	Cell	tally	Value	
94	Cells	Cell	tally	Value	
104	Cells	Cell	tally	Value	
114	Cells	Cell	tally	Value	
124	Cells	Cell	tally	Value	
134	Cells	Cell	tally	Value	
144	Cells	Cell	tally	Value	
154	Cells	Cell	tally	Value	
164	Cells	Cell	tally	Value	
174	Energy	Energy [MeV]	Cells	Cell	20
204	Cells	Cell	tally	Value	
214	Cells	Cell	tally	Value	

This sheet regulates the Excel output derived from the benchmark. It consists of a table where each row regulates the output of a single tally present in the MCNP input.

Hereinafter a description of the available fields is reported.

Tally tally number according to MCNP input file.

x, y select the binnings to be used for the presentation of the excel results of the specific tally. Clearly, the binning should have been coherently defined in the MCNP input too. MCNP allows different types of tally binning, they can be accessed using the tags reported in the table below.

Table 1: Allowed binnings

Admissible **x** and **y**
Energy
Cells
time
tally
Dir
User
Segments
Multiplier
Cosine
Cor A
Cor B
Cor C

As a result of the selected **x** and **y** option, the results of the post-processed tally will be display in a matrix format. In case only a single binning is defined in the MCNP input, the `tally` keyword should be used to signal to JADE to just to print the results in a column format.

Important: The main direction of an Excel file is considered to be the vertical one, which is the preferred scrolling direction. For this reason, the **x** direction is associated with the vertical direction in an Excel file and the **y** with the horizontal one.

Warning: No more than two binnings should be defined for a single MCNP tally due to the limitation of having to represent 2-D output. JADE may be able to to handle tallies with more than 2 binnings if some of them are constant values.

Tip: If a 1D FMESH is defined in the MCNP input, JADE will automatically transform it to a "binned" tally and handle it as any other tally using the `Cor A`, `Cor B` or `Cor C` field.

---

x name, y name These will be the names associated to the **x** and **y** axis printed in the excel file.

cut Y The idea behind JADE is to produce outputs that are easy to investigate simply by scrolling and concentrate on the main results highlighted through colors. Having a high number of bins both in the x and y axis may cause a problem in this sense, forcing the user to scroll on both axis. For this reason, a maximum number of columns can be set to solve this issue. This will cause the tally results not to be printed as a unique matrix but as sequential blocks each with a number of columns equal to **cut Y**.

4.4.2 Atlas

Tally	Quantity	Unit	Binned graph	Ratio graph
4	Neutron Flux	#/cm^2	True	False
14	Photon Flux	#/cm^2	True	False
6	Total Nuclear Heating	W/g	False	True
16	Neutron Heating	W/g	False	True
26	Photon Heating	W/g	False	True
24	DPA in Fe	dpa/FPY	False	False
34	Helium production in SS316	appm/FPY	False	False
44	Hydrogen production in SS316	appm/FPY	False	False
54	Tritium production in SS316	appm/FPY	False	False
64	DPA in Cu	dpa/FPY	False	False
74			False	False
84			False	False
94			False	False
104			False	False
114			False	False
124			False	False
134			False	False
144			False	False
154			False	False
164			False	False
174			False	False
204	Neutron Flux	#/cm^2	False	True
214	Photon Flux	#/cm^2	False	True

This sheet regulates the Atlas output (Word/PDF) derived from the benchmark. It consists of a table where each row regulates the output of a single tally present in the MCNP input. Hereinafter a description of the available fields is reported.

Tally tally number according to MCNP input file.

Quantity Physical quantity that will be plotted on the y-axis of the plot. For the x-axis the one specified in the Excel sheet under **x** will be considered. The quantity selected for plotting will always be the tallied quantity.

Important: when two binnings are specified in the Excel sheet, a different plot for each of the **y** bins will be produced. For example, let's consider a neutron flux tally binned both in energy (selected as **x**) and cells (selected as **y**). Then, a plot showing the neutron flux as a function of energy will be produced for each cell indicated in the tally.

Unit Unit associated to the Quantity.

<Graph type> Different columns can be added where it can be specified if a plot in the style indicated by the column name should be generated (`true`) or not (`false`). The available plot styles are *Binned graph*, *Ratio Graph*, *Experimental points* and *Grouped bars*.

See also:

Plots Atlas for an additional description of the available plot styles.

CHAPTER

FIVE

USAGE

Once JADE is correctly configured (for additional details see *Configuration*), the application can be started from the 'Code' folder with:

```
python main.py
```

The main menu is loaded at this point:

```
*********************************************
            Welcome to JADE v1.1.0
     A nuclear libraries V&V Test Suite
         Release date: 25/05/2021

               MAIN MENU

       Powered by NIER, UNIBO, F4E
*********************************************
MAIN FUNCTIONS

* Open Quality check menu               (qual)
* Open Computational Benchmark menu     (comp)
* Open Experimental Benchmark menu       (exp)
* Open Post-Processing menu             (post)
---------------------------------------------
UTILITIES

* Print available libraries         (printlib)
* Translate an MCNP input              (trans)
* Print materials info              (printmat)
* Generate material                 (generate)
* Switch fractions                   (switch)
---------------------------------------------
* Test installation                    (test)

* Exit                                 (exit)
```

This menu allows users to interact with the tool directly from the command prompt via pre-defined commands. The following main option are available typing from the main menu:

- `qual` not currently supported;
- `comp` opens the *Computational Benchmark menu*;

- `exp` opens the *Experimental Benchmark menu*;
- `post` opens the *Post-processing menu*;
- `exit` exit the application.

Additionaly to these main options, a series of "utilities" can be also accessed from the main menu. These are a collection of side-tools initially developed for JADE that nevertheless can be useful also as stand-alone tools. A detailed description of these functionalities can be found in *Utilities*.

5.1 Quality check menu

Not implemented.

5.2 Computational Benchmark menu

```
*********************************************
            Welcome to JADE v1.1.0
     A nuclear libraries V&V Test Suite
         Release date: 25/05/2021

         COMPUTATIONAL BENCHMARK MENU

       Powered by NIER, UNIBO, F4E
*********************************************

* Print available libraries          (printlib)
* Assess library                       (assess)
* Continue assessment                (continue)
* Back to main menu                      (back)
* Exit                                   (exit)

Enter action:
```

The following options are available in the computational benchmark menu:

- `printlib` print to video all the available nuclear data libraries in the xsdir file selected during JADE configuration;
- `assess` start the assessment of a selected library on the computational benchmarks. The library is specified directly from the console when the selection is prompted to video. The library must be contained in the xsdir file (available libraries can be explored using `printlib`);
- `continue` continue a previously interrupted assessment for a selected library. **Currently, this option is implemented only for the Sphere Leakage benchmark.**
- `back` go back to the main menu;
- `exit` exit the application.

The selection of the libraries is done indicating their correspondent suffix specified in the xsdir file (e.g. `31c`). Activation benchmarks need to be run separetly since they require two different libraries to be specified: one for activation and one for transport. Activation library must always be specified first (e.g. `99c-31c`).

Note: Whenever an assessment is requested, all the benchmarks selected in the main configuration file will be considered. In case the requested library was already assesed on one or more of the active benchmarks, the user will be asked for permission before overriding the results.

See also:

Configuration for additional details on the benchmark selection.

5.3 Experimental Benchmark menu

```
*********************************************
            Welcome to JADE v1.1.0
     A nuclear libraries V&V Test Suite
         Release date: 25/05/2021

         EXPERIMENTAL BENCHMARK MENU

       Powered by NIER, UNIBO, F4E
*********************************************

* Print available libraries          (printlib)
* Assess library                       (assess)
* Continue assessment                (continue)
* Back to main menu                      (back)
* Exit                                   (exit)

Enter action:
```

The following options are available in the experimental benchmark menu:

- `printlib` print to video all the available nuclear data libraries in the xsdir file selected during JADE configuration;
- `assess` start the assessment of a selected library on the experimental benchmarks. The library is specified directly from the console when the selection is prompted to video. The library must be contained in the xsdir file (available libraries can be explored using `printlib`);
- `continue` **[not implemented]**
- `back` go back to the main menu;
- `exit` exit the application.

The selection of the libraries is done indicating their correspondent suffix specified in the xsdir file (e.g. `31c`). Activation benchmarks need to be run separetly since they require two different libraries to be specified: one for activation and one for transport. Activation library must always be specified first (e.g. `99c-31c`).

Note: Whenever an assessment is requested, all the benchmarks selected in the main configuration file will be considered. In case the requested library was already assesed on one or more of the active benchmarks, the user will be asked for permission before overriding the results.

See also:

Configuration for additional details on the benchmark selection.

5.4 Post-processing menu

```
*********************************************
            Welcome to JADE v1.1.0
     A nuclear libraries V&V Test Suite
         Release date: 25/05/2021

         POST PROCESSING MENU

       Powered by NIER, UNIBO, F4E
*********************************************

* Print tested libraries            (printlib)
* Post-Process library                   (pp)
* Compare libraries                  (compare)
* Compare Vs Experiments             (compexp)
* Back to main menu                     (back)
* Exit                                  (exit)

Enter action:
```

The following options are available in the post-processing menu:

- `printlib` print all libraries that were tested and that are available for post-processing;
- `pp` post-process a single library;
- `compare` compare different libraries results on computational benchmarks;
- `compexp` compare different libraries results on experimental benchmarks;
- `back` go back to the main menu;
- `exit` exit the application.

For the `pp`, `compare` and `compexp` the selection of the libraries will be directly prompt to video. The selection of the libraries is done indicating their correspondent suffix specified in the xsdir file (e.g. `31c`). When comparing more than one library, the suffixes should be separated by a '-' (e.g. `31c-32c`). The first library that is indicated is always considered as the *reference library* for the post-processing. There may be a limitation on the number of libraries that can be compared at once depending on the post-processing settings.

Only one library at the time can be post-processed with the `pp` option. Nevertheless, when a comparison is requested that includes libraries that were not singularly post-processed, an automatic `pp` operation is conducted on them.

Warning: Please note that `printlib` will simply show all libraries for which at least one benchmark has been run.

Warning: Please note that part of the single post-processing of the libraries is used in the comparisons. Also, JADE does not perform any checks on the consistency between the two. This responsability is left to the user. The following is an example of incorrect usage that can lead to erroneous results:

1. a first assessment is run;
2. single post-processing is completed;
3. some configuration settings are changed and the assessment is re-run;
4. a comparison is requested.

In this case, JADE cannot know that the first single post-processing was done on a different benchmark run wiith respect to the requested comparison. As a result, the outputs coming from different assessments will be mixed up.

Note: Whenever a post-processing is requested, all the benchmarks selected in the main configuration file will be considered. In case one or more of the requested libraries were already post-processed on one or more of the active benchmarks, the user will be asked for permission before overriding the post-processing results.

See also:

Configuration for additional details on the benchmark selection.

CHAPTER

SIX

DEFAULT BENCHMARKS

This section describe more in detail all the default benchmarks that have been implemented in JADE dividing them between computational and experimental benchmarks.

Important:

- Not all benchmark inputs and related files can be distributed toghether with JADE due to licensing reasons. In case the user provide evidence of licensing rights on specific benchmarks, the JADE team will provide the missing input which, once copied in suitable JADE folders, allow to correctly run them.
- For some of the benchmarks, weight windows (WW) have been produced and necessary for the benchmark run. Unfortunately, these WW are often too heavy for them to be distributed with Git. These files must be downloaded separately and inserted in a suitable folder in `<JADE_root>\Benchmark inputs\VRT`.
- The benchmarks included in JADE can be also divided between **transport** benchmarks (ususally associated with classical MCNP) and **activation** benchmarks (usually associated with D1S-UNED). It is recommended to run these two benchmarks categories separately, mostly because they require a different input in terms of library to be assessed. If transport Benchmarks expect a single library (e.g. `31c`), activation one require two: an activation library and a transport one for all zaids that cannot be activated (e.g. `99c-31c`).
- In activation benchmarks, the library that is considered the assessed one is always the activation library (i.e. the first provided). No track is kept during the post-processing of which was the transport library used and it is responsability of the user to make sure that comparisons between activation libraries results are done in a coherent way. That is, the same transport library should be always used.

6.1 Computational Benchmarks

6.1.1 Sphere Leakage

The Sphere Leakage benchmark is arguably the most important benchmark included in the JADE suite. Indeed, it allows to test individually each single isotope of the nuclear data library under assessment plus some typically used material in the ITER project namely:

- Water;
- Ordinary Concrete;
- Boron Carbide;
- SS316L(N)-IG;
- Natural Silicon;
- Polyethylene (non-borated).

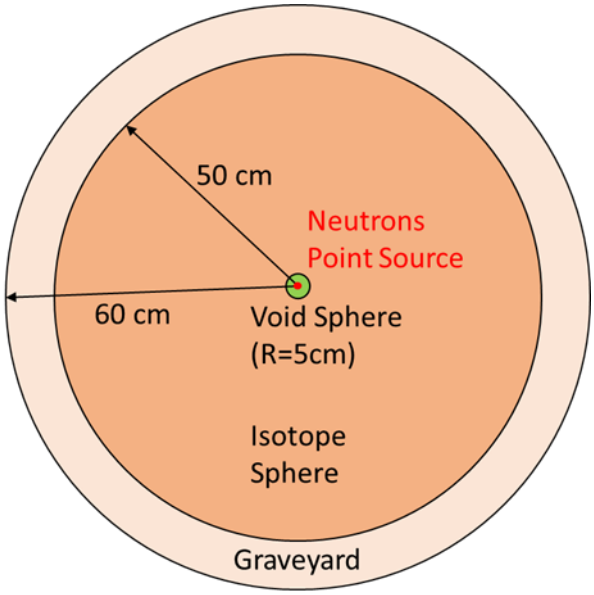


Fig. 1: Sphere Leakage geometrical model

Geometry and run parameters

The Sphere Lekage geometry consists of actually three concentric spheres. The inner one is void and has a radius of 5 cm. Here is located the uniform probability 0-14 MeV neutron point source. The second sphere has a radius of 50 cm and it is composed entirely by a single isotope or a typical ITER material. Finally, the last 60 cm radius sphere acts as a graveyard where particles importance is set to zero and the boundary of the model is defined.

Other two important settings that needed to be defined where the the choice of the sphere density and of the MCNP STOP card parameters. Since to impose a single density equal for all materials and zaids was not a valid option, in order to keep some kind of physical meaning in the results, as default densities the one computed using NTP (Normal Temperature and Pressure) conditions were used. These are are defined at 20 °C and 101325 Pa (1 atm). Even if these values work quite well with solids, they cause gases to perform poorly in terms of tally scoring. This happens due to the substantially lower density in NTP conditions for gases when compared to solids, resulting in too few interactions of the neutrons and secondary photons with the material. This has been proven to be especially true for hydrogen and helium, leading to the choice of selecting their liquid phase density instead. Another issue was encountered when simulating fissile isotopes like U235. A 1m diameter sphere containing a pure fissile isotopes at NTP density is very much super-critical and the high number of secondary particles (i.e. other neutrons) produced caused the simulations to fail due to memory limitations. For this reason, the density of these isotopes was imposed equal to 1 g/cc as if an aerosol was considered.

Finally, the STOP card parameters were optimized. MCNP allows to stop a simulation based either on: * the precision reached in a specified tally; * the number of histories (NPS); * the total elapsed computer time (i.e. the sum of computer time used by all CPUs).

The optimization of such parameters for each element was done through trial and error with the aim of finding a good balance between computational cost and enough precise results. These parameters are provided by default in JADE, but the user may modify them if necessary through benchmark-specific configuration files. Density values can be modified too. The files are located in `<JADE_root>\Configuration\Benchmarks Configuration\Sphere`.

Tallies

Both the transport of neutrons and of secondary photons are active and photons cut-off energy is left to the default value of 1 KeV. The following MCNP tallies are defined in the Sphere Leakage benchmark:

Tally n. 2 Fine neutron flux at the external surface of the filled sphere. The flux is binned in energy using the Vitamin-J cite{Sartori:vitaminJ} 175 energy group structure.

Tally n. 12 Coarse neutron flux at the external surface of the filled sphere. The flux is binned in 5 energy groups: 1e-6, 0.1, 1, 10 and 20 MeV.

Tally n. 32 Fine photon flux at the external surface of the filled sphere. The flux is binned in energy using the 24 group structure described in the FISPACT manual cite{manual:fispact}.

Tally n. 22 Coarse photon flux at the external surface of the filled sphere. The flux is binned in 5 energy groups: 0.01, 0.1, 1, 10 and 20 MeV.

Tally n. 4 Neutron heating computed in the filled sphere (F4+FM strategy).

Tally n. 44 Photon heating computed in the filled sphere (F4+FM strategy).

Tally n. 6 Neutron heating computed in the filled sphere (F6 strategy).

Tally n. 16 Photon heating computed in the filled sphere (F6 strategy).

Tally n. 14 Helium (He) ppm production in the filled sphere.

Tally n. 24 Tritium (T) ppm production in the filled sphere.

Tally n. 34 Displacement Per Atom (DPA) production in the filled sphere.

More details on the MCNP tally definition and especially on the difference between the heating computation using the F6 or the F4+FM strategy can be found in §ref{sec:tallies}.

See also:

Related papers and contributions:

- D. Laghi, M. Fabbri, L. Isolan, R. Pampin, M. Sumini, A. Portone and A. Trkov, 2020, "JADE, a new software tool for nuclear fusion data libraries verification & validation", *Fusion Engineering and Design*, **161** 112075

6.1.2 ITER 1D

The ITER 1D benchmark developed by Sawan M. is a popular 1-Dimensional neutronic model used for nuclear data benchmarking in the fusion community. This consists of a simple but realistic model of the ITER TOKAMAK where the inboard and outboard portion of the machine and the plasma region are modelled by means of simple concentric cylindrical surfaces.

Geometry and run parameters

As visible in the figure, the benchmarks geometry is uniquely composed by concentric cylindrical surfaces. A detailed description of the different layers is reported hereafter:

The plasma region includes a 14.1 MeV isotropic neutron source (characteristic of Deuterium-Tritium fusion reaction).

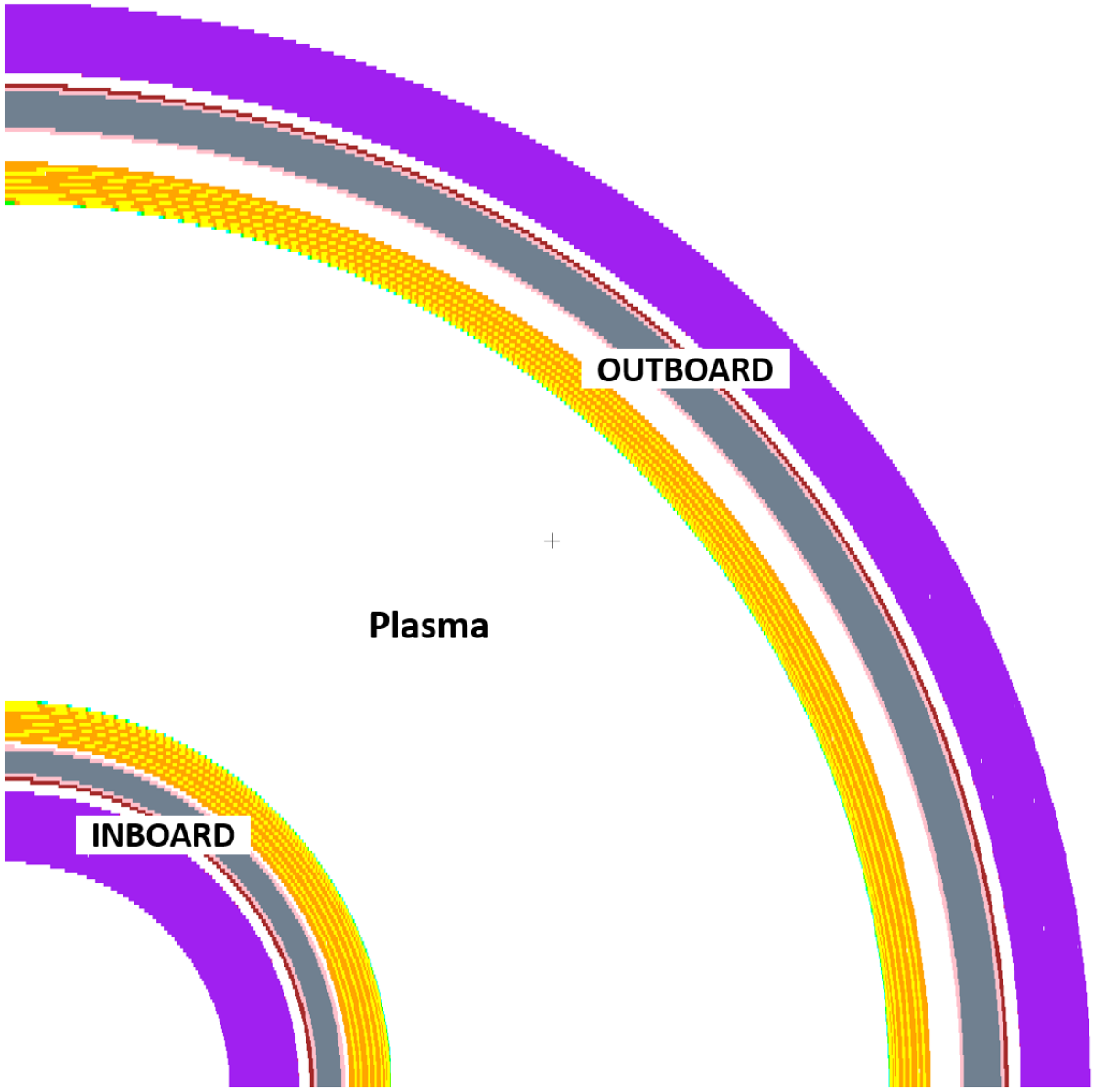


Fig. 2: ITER 1D MCNP geometry (quarter)

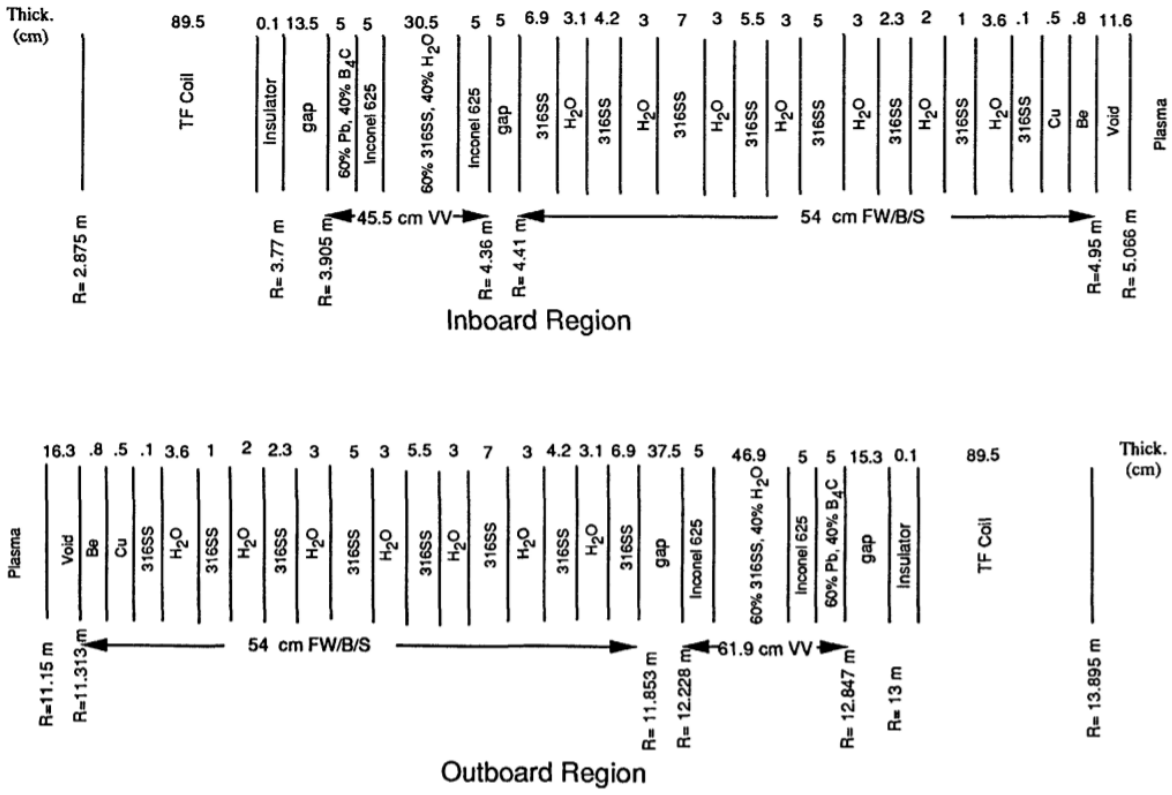


Fig. 3: Description of the layers composing the ITER 1D benchmark

Tallies

Many quantities are tallied in the ITER 1D benchmark, the following is a thorough description of them.

Tally n. 4 Neutron flux [#/cm^2] (binned in Vitamin-J 175 energy groups) in 97 different MCNP cells located across the radial direction.

Tally n. 204 Total neutron flux [#/cm^2] at the same locations as Tally n. 4.

Tally n. 14 Photon flux [#/cm^2] (binned in energy) at the same locations as Tally n. 4. The energy bins limits are 0.1, 1, 5, 10 and 20.

Tally n. 214 Total photon flux [#/cm^2] at the same locations as Tally n. 4.

Tally n.6 Total nuclear heating [W/g], i.e., neutron plus photon heating at the same locations as Tally n. 4.

Tally n. 16 Neutron heating [W/g] at the same locations as Tally n. 4.

Tally n. 26 Photon heating [W/g] at the same locations as Tally n. 4.

Tally n. 34 Helium production in steel.

Tally n. 44 Hydrogen production in steel.

Tally n. 54 Tritium production in steel.

Tally n. 64 Displacement per atom (DPA) in Cu.

Tally n. 74 Helium production in CuBeNi.

Tally n. 84 Hydrogen production in CuBeNi.

Tally n. 94 Tritium production in CuBeNi.

Tally n. 104 DPA in Nickel.

Tally n. 114 Helium production in Inconel.

Tally n. 124 Hydrogen production in Inconel.

Tally n. 134 Tritium production in Inconel.

Tally n. 144 Helium production in Be.

Tally n. 154 Hydrogen production in Inconel.

Tally n. 164 Tritium production in Inconel.

Tally n. 174 Fast (E>0.1 MeV) neutron fluence at magnets.

See also:

Related papers and contributions:

- M. Sawan, 1994, "FENDL Neutronics Benchmark: Specifications for the calculational and shielding benchmark", (Vienna: INDC(NDS)-316)

6.1.3 Test Blanket Module

Tritium and Deuterium are the two main ingredients of the fusion reaction that is foreseen to be tested in ITER and that should guarantee sustainable energy production in DEMO. Deuterium is abundant in normal sea water and can be extracted relatively easily. Unfortunately, the same cannot be said for Tritium, whose short half life means that is not readily available in nature. For this reason, in order to be sustainable, TOKAMAKs will need to be able to produce, or "breed" all the tritium needed for their fusion reactions. Tritium breeding is not foreseen for the ITER power plant and the open point will need to be closed by DEMO instead. Nevertheless, ITER is a unique opportunity to test various breeding concepts and find out what will be the best solution to implement in DEMO, leading to the creation of the Test Blanket Module (TBM) project. These are prototypes of blanket sections (and their ancillaries) which have the capability to breed and store tritium for later use.

Building on the historical ITER 1D model, in 2020, two additional benchmark were generated by the F4E neutronic team which had a specific focus on the ITER TBM experiments. The ITER 1D original model was focused on shielding application and did not feature any port in the outboard region. On the contrary, in the new benchmarks, the outboard region was substituted with 1D models of the two proposed European concepts for the TBM: the Helium Cooled Pebbled bed (HCPB) and the Water Cooled Lithium Lead (WCLL).

See also:

ITER 1D

Geometry and run parameters

The benchmarks are focused on the TBM set (i.e. the actual blanket module) and on the shielding section that can be found behind it. While the shield section does not really change between the two different TBM concepts, the TBM sets do.

Tallies

All Tritium production tallies that were defined for the ITER 1D benchmark were retained also in the TBMs ones. Additionally, a 1-dimensional FMESH was placed on the outboard region (from R=830 up to R=1084.2) composed by 2000 bins. The following quantities were tallied on such grid:

Tally n. 214 Neutron heating [MeV/cm^3/n_s.

Tally n. 224 Photon heating [MeV/cm^3/n_s].

Tally n. 234 Tritium production [atoms/cm^3/n_s].

Tally n. 244 Neutron flux [#/cm^3/n_s].

Tally n. 254 Photon flux [#/cm^3/n_s].

See also:

ITER 1D

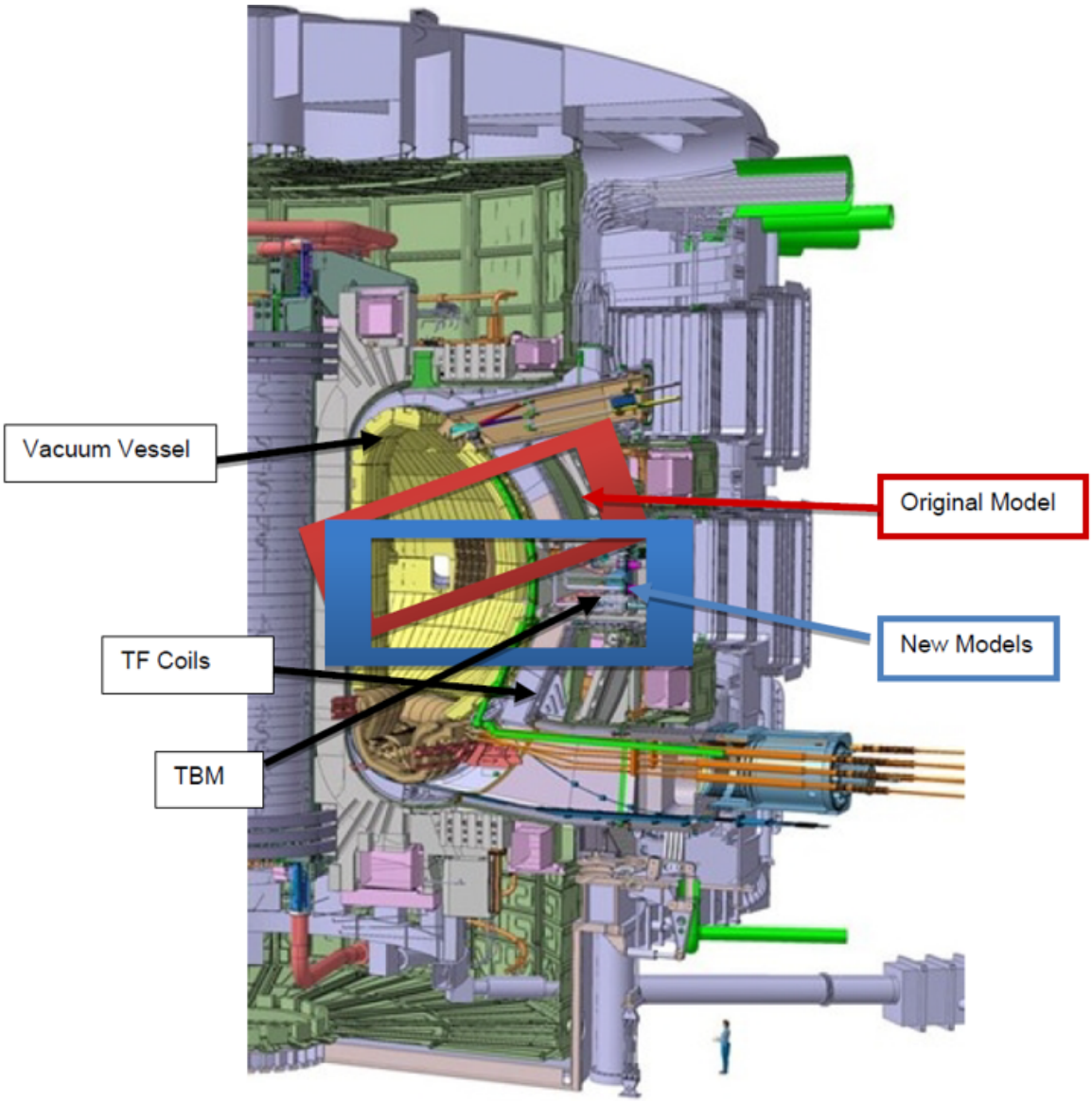


Fig. 4: Position of the MCNP model in the ITER tokamak

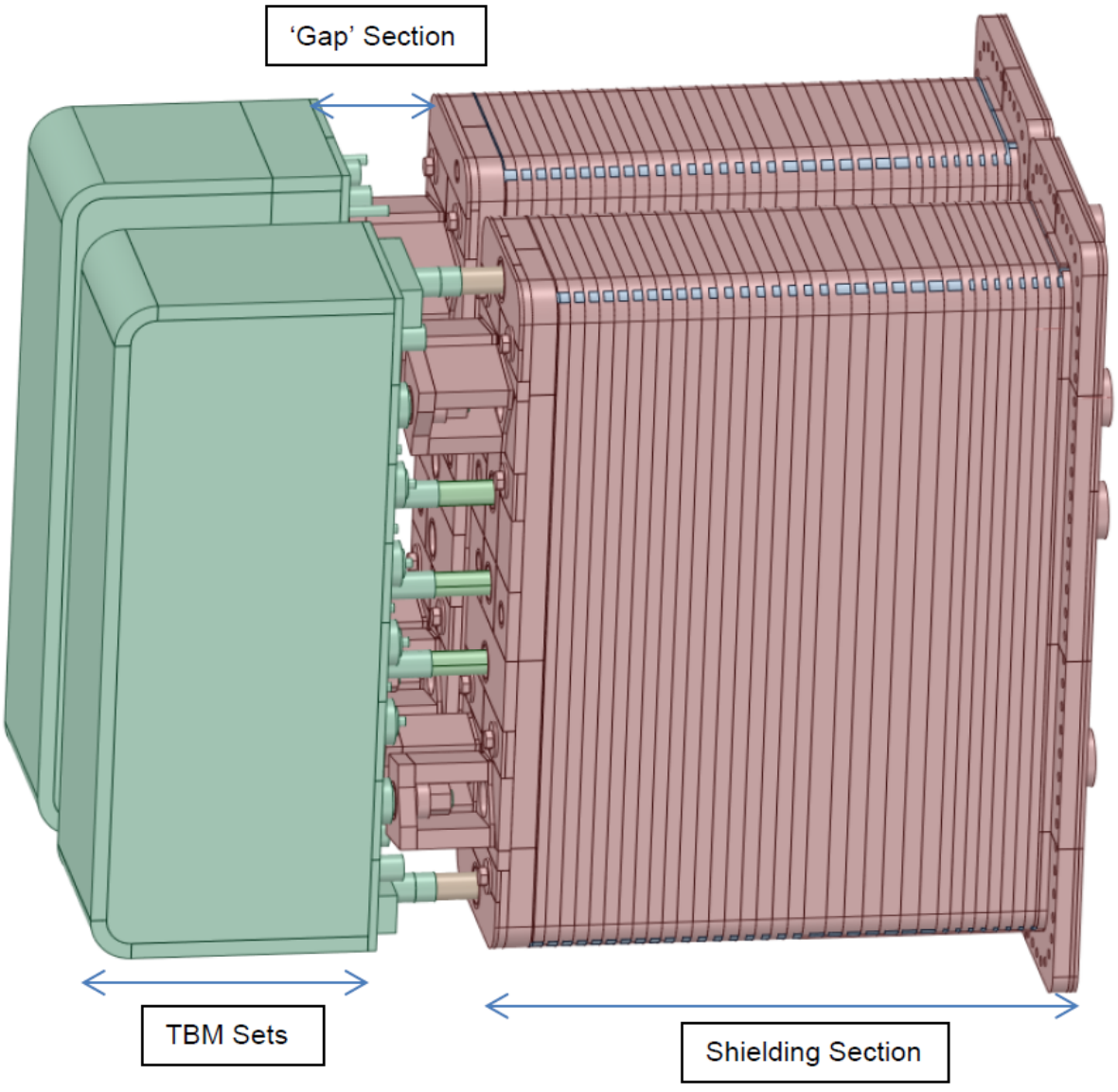


Fig. 5: CAD model of the TBM component

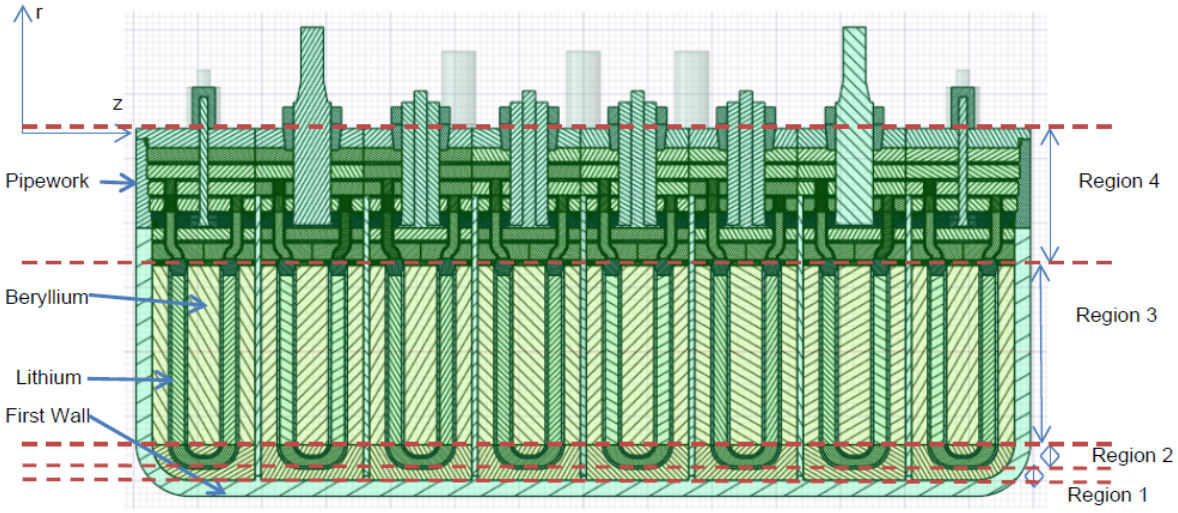


Fig. 6: Section of the CAD model of the HCPB TBM set

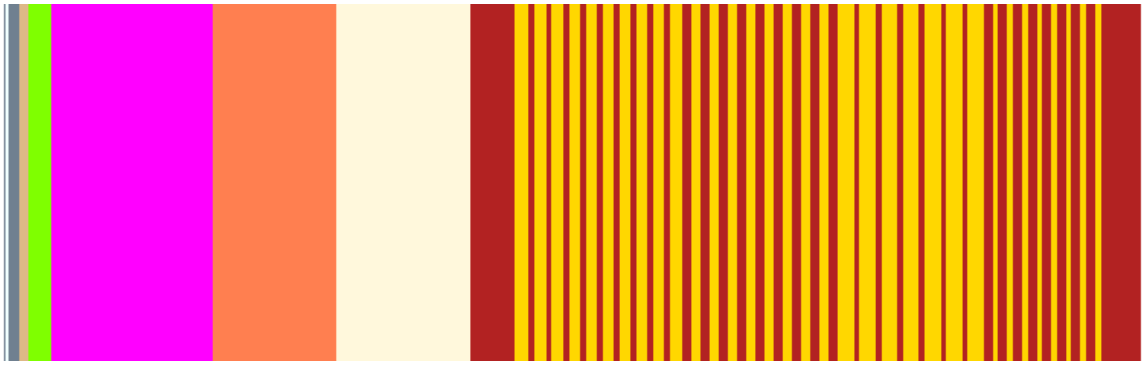


Fig. 7: Visualization of the TBM and and shielding section in the 1D MCNP geometry

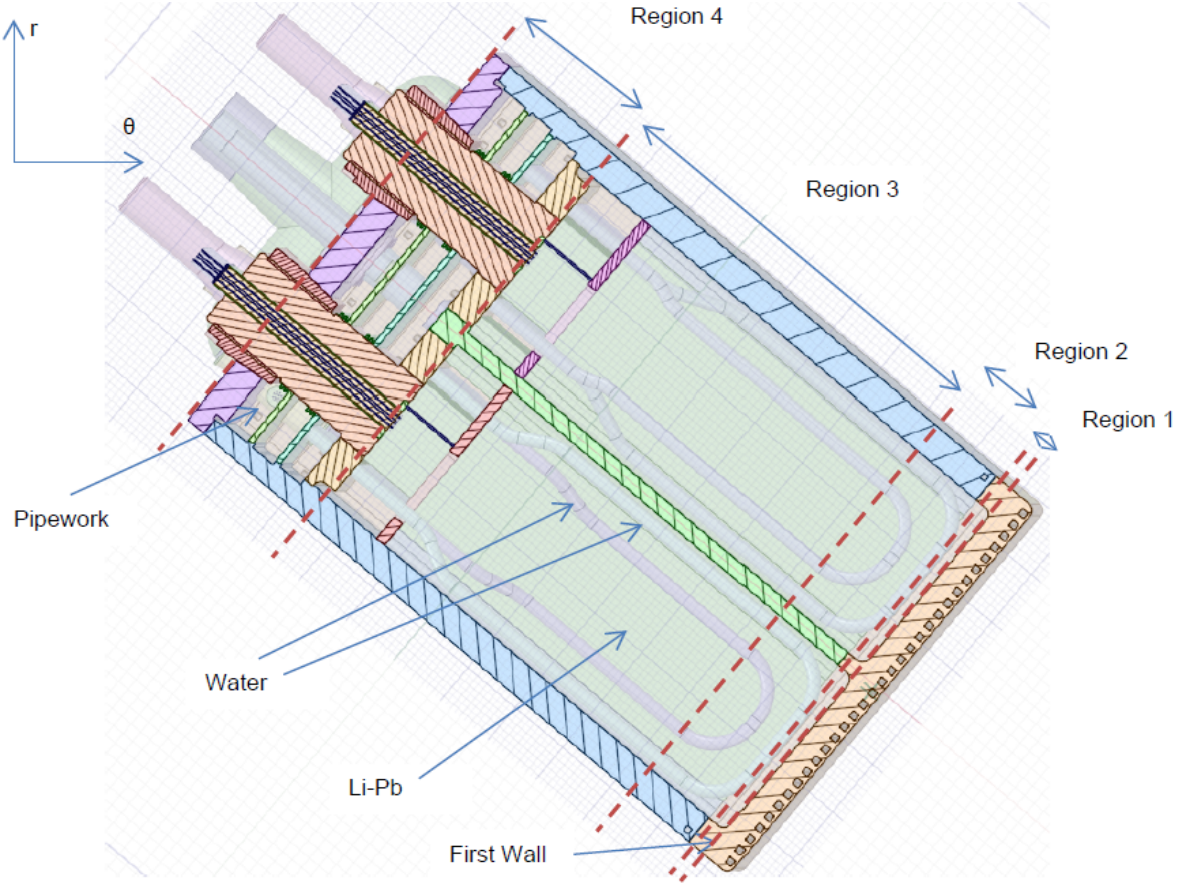


Fig. 8: Section of the CAD model of the WCLL TBM set

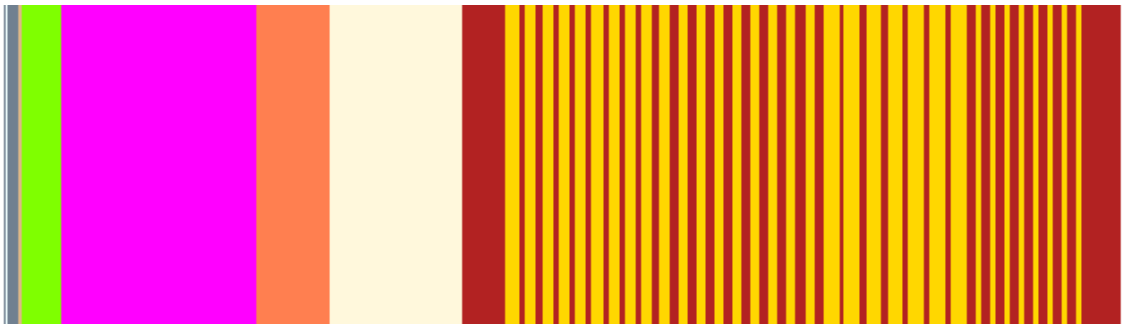


Fig. 9: Visualization of the TBM and shielding section in the 1D MCNP geometry

C-Model

Important: This benchmark input cannot be distributed directly with JADE. The user must request to obtain it from ITER organization and insert it in the `<JADE root>\Benchmarks inputs` directory renaming it 'C_Model.i'.

The NPS card needs to be removed from the input. It is recommended to also delete total bins from standard tallies for a clearer post-processing results.

During the long life of the ITER project, many neutronics models have been generated to represent the TOKAMAK machine. These are used to conduct neutronic analyses on the reactor in order to investigate many direct and indirect effects induced by neutrons like heat generation, particle generation, DPA, dose rate, etc. C-Model is an extremely detailed MCNP input of a 40° sector of ITER TOKAMAK. It was the most complete neutronic model available for the ITER machine until 2021, when E-lite was released which is a full 360° model of ITER that was conceived to overcome some limitation encountered using the C-Model for specific application. Nevertheless, since E-lite is an extremely heavy model, C-model is still considered the reference basic model of the ITER TOKAMAK for neutronic analyses.

Geometry and run parameters

Due to its complexity, a thorough description of the C-Model benchmark geometry is considered out of the scope of this work and can be found, instead, in a dedicated F4E report.

Tallies

The C-model standard tallies have been used. They include neutron current, neutron current and nuclear heating at different locations.

See also:

Related papers and contributions:

- D. Leichtle, B. Colling, M. Fabbri, R. Juarez, M. Loughlin, R. Pampin, E. Polunovskiy, A. Serikov, A. Turner and L. Bertalot, 2018, "The ITER tokamak neutronics reference model C-Model", *Fusion Engineering and Design*, **136** 742-746
- R. Juarez, G. Pedroche, M. J. Loughlin, R. Pampin, P Martinez, M. De Pietri, J. Alguacil, F. Ogando, P. Sauvan, A. J. Lopez-Revelles, A. Kolšek, E. Pol-unovskiy, M. Fabbri, and J. Sanz. "A full and heterogeneous model of the ITERtokamak for comprehensive nuclear analyses". In:Nature Energy 6 (2021), pp. 150–157
- E. Polunovskiy. "Description of ITER Nuclear Analysis Tokamak Reference Model C-model R181031". Technical Report [ITER IDM XETSWC v1.5]. Iter Organization, 2019.

6.1.4 Sphere SDDR

The Sphere SDDR benchmark is a variation of the the *Sphere Leakage* which is focused on isotopes activation and dose rate measurement. Once again, these kind of benchmarks allows to test all available isotopes in the library under assessment (this time being a D1S activation library) together with a few typical ITER materials. In particular, each single reaction channel (MT) of every isotope will be tested separately while, for the typical materials, all possible reactions foreseen by the library will be considered.

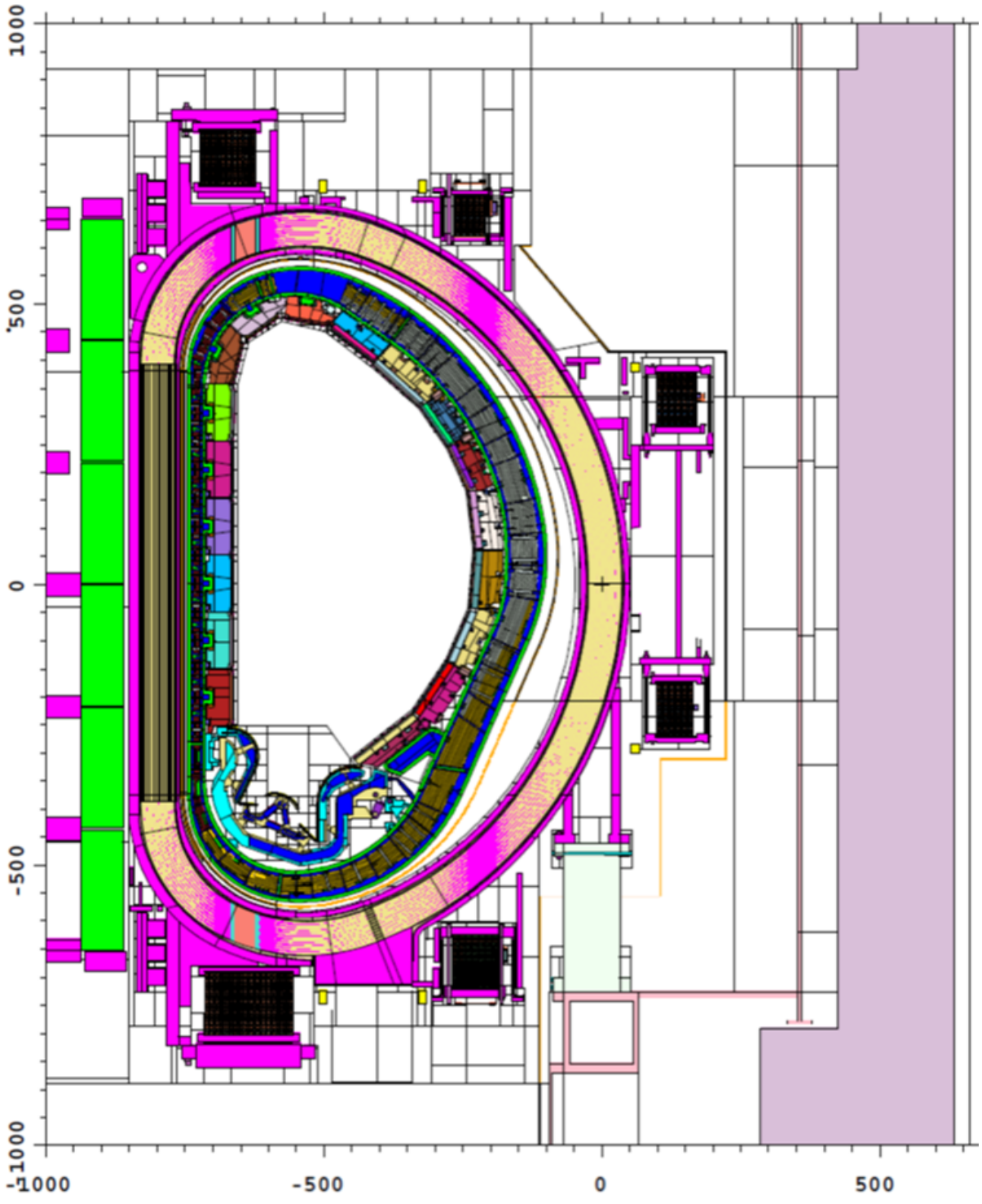


Fig. 10: C-model R181031. Origin (1050,200,0). Basis (0.982339, 0.187112, 0.000000) (0,0,1). Extent (1000,1000)

Geometry and run parameters

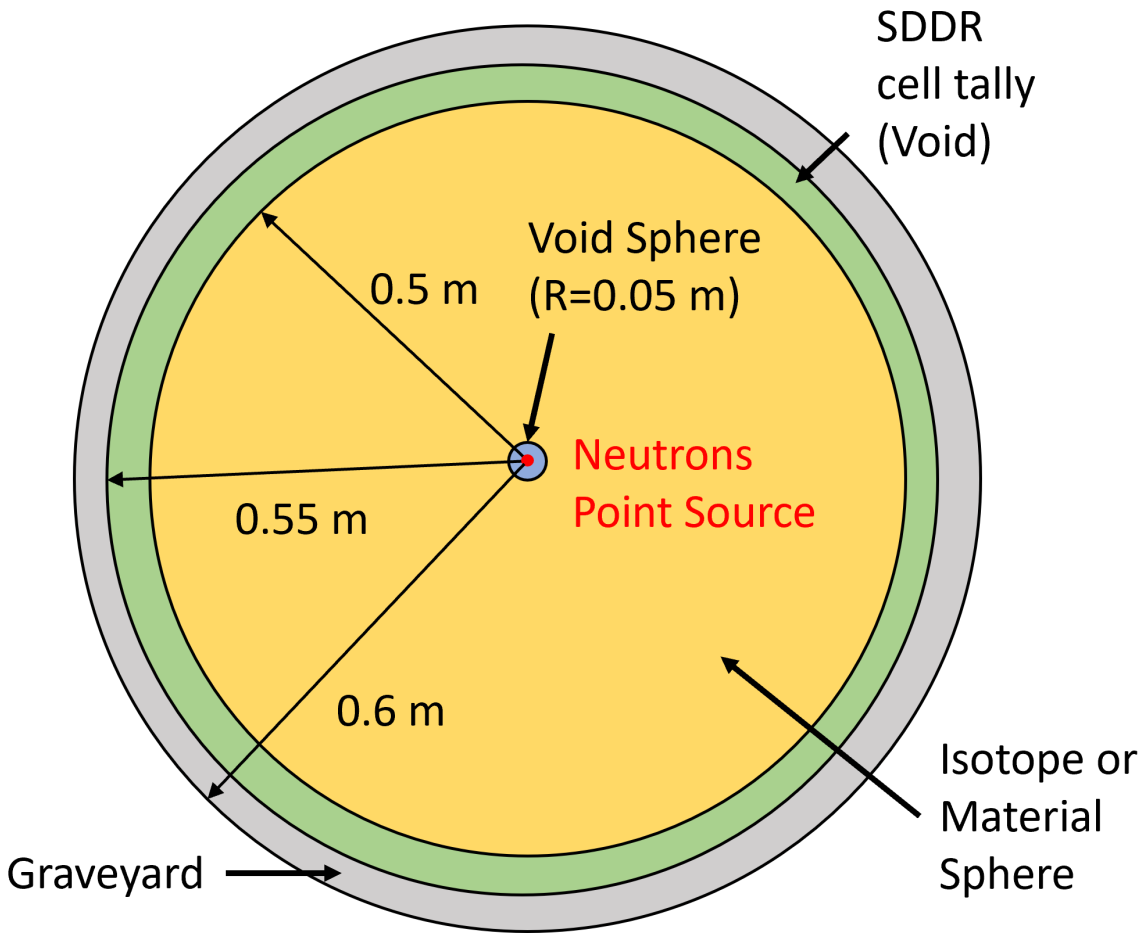


Fig. 11: Schematic view of the Sphere SDDR model

The geometry of the Sphere SDDR benchmarks, is practically the same as the one of the Sphere Leakage benchmark. The only difference is that externally to the filled sphere, a void spherical shell has been defined having a 10 cm radial thickness. This is the cell used to tally the contact shut down dose rate.

Similarly to what was done for the Sphere Leakage benchmark, the user have control on the densities to be applied for each element and material (default is set to NTP conditions with few exceptions) and control on the STOP parameters to be used.

SDDR parameters

The cool-down times that have been considered are 0 s, 2.7 h, 24 h, 11.6 d, 30 d and 10 y. For the isotopes simulations, since only one reaction is considered, relative comparisons at different cool-down times will not lead to different results, hence, during post-processing operations only the results at $0s$ are elaborated. This does not apply to materials simulations, where many different reactions are included.

The irradiation schedule considered for the Sphere SDDR benchmark is reported hereafter and represents an actual equivalent irradiation scenario foreseen for ITER blanket (mode SA2):

Table 1: Irradiation schedule (ITER mode SA2)

Source Intensity [n/s]	Δt irr.	Multiplicity
1.0714E+17	2 y	1
8.2500E+17	10 y	1
0	9 m	1
1.6667E+18	15 m	1
0	3290 s	->
2.0000E+19	400 s	17
0	3290 s	->
2.8000E+19	400 s	4

As previously discussed, the irradiation file and reaction file provided together with the MCNP input file are generated in two different ways depending on if the simulation is conducted on a single isotope or on a typical ITER material. In the first case, a single reaction is considered and the irradiation file will only contain the daughter of such irradiation. In the second case, all possible reactions that are available in the library and that can be originated in the material will be included. The irradiation file will be then generated accordingly.

Tallies

All neutron and photon related tallies defined in Sphere Leakage benchmark have also been imported in the Sphere SDDR benchmark. For photons, the time binning necessary to cover all the cool-down times of interest have been added. Tally n. 104 have been also defined to tally the contact shut down dose rate in air at all cool-down times in the additional spherical shell added for this specific purpose [Sv/h].

6.1.5 ITER CYLYNDER SDDR

The ITER Cylinder SDDR is a very popular computational benchmark for SDDR computation in ITER since it features dimensions, materials and streaming characteristics of a typical ITER equatorial port.

Geometry and run parameters

The ITER Cylinder SDDR is a simple yet effective benchmark. The model is composed by a 550 cm long, hollowed steel cylinder with internal and external radius respectively equal to 50 cm and 100 cm. The rear part of the cylinder is closed with a steel disk plate of 48 cm radius and 15 cm thick. The inner front part of the cylinder is filled with a smaller cylinder made of a water-steel mixture. This internal 48 cm radius cylinder is 210 cm long and features a central 15 cm diameter cylindrical hole. As it can be deducted from the given measures, a 2 cm gap is left between the main external hollow cylinder and its internal components.

A volumetric and isotropic neutron source is also defined. The volume of emission is a disk aligned with the front part of the cylinder assembly and positioned at a distance of 100 cm. The volume of the disk is 10 cm thick and has a radius equal to 100 cm.

A series of cells and surfaces is also defined for tallying purposes. The shutdown dose rate due to the activation of the assembly is evaluated in cell tallies located 30 cm past the end of the rear plate. The tally cells consist of concentric (hollow) disks which are 10 cm thick and are characterized from the following radii:

- from 0 cm to 15 cm (MCNP cell n. 10);
- from 15 cm to 30 cm (MCNP cell n. 11);
- from 30 cm to 45 cm (MCNP cell n. 12);
- from 45 cm to 60 cm (MCNP cell n. 13).

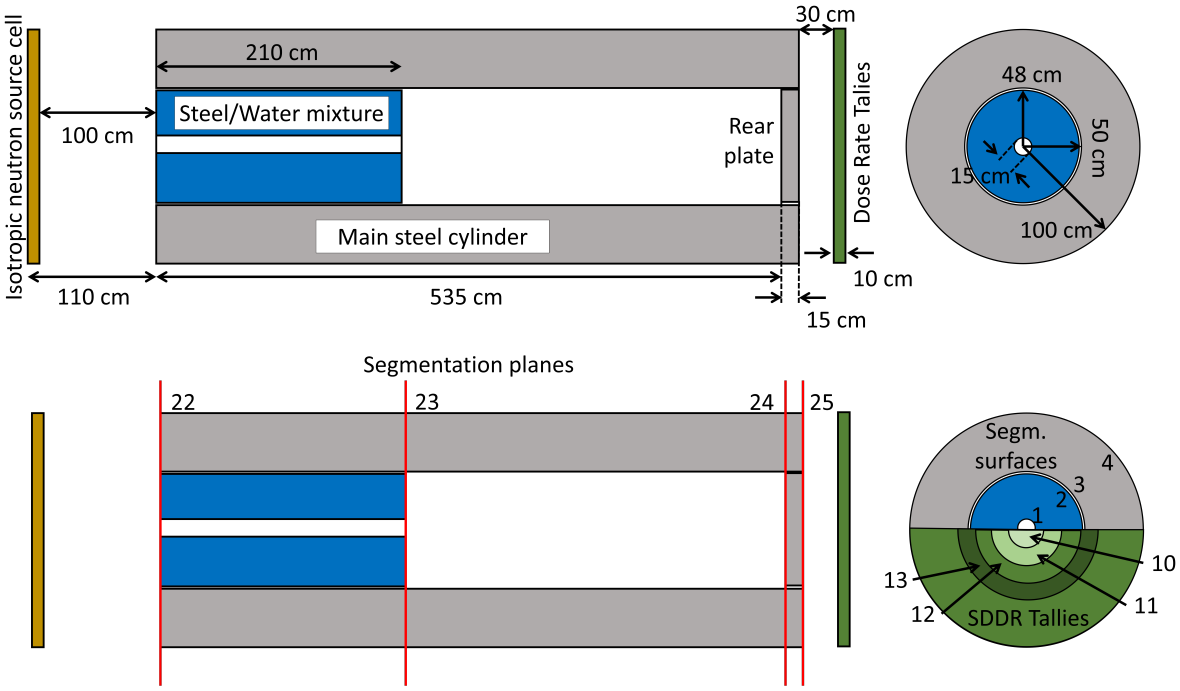


Fig. 12: ITER Cylinder SDDR benchmark geometry visualization

For flux tallying purposes, instead, the following cylindrical surfaces have been defined:

- n. 1, coincident with the external surface of the central hole;
- n. 2, coincident with the external surface of the water/steel cylinder;
- n. 3, coincident with the internal surface of the main steel cylinder;
- n. 4, coincident with the external surface of the main steel cylinder;

and the following planes orthogonal to the cylinder length:

- n. 22, coincident with the front of the assembly;
- n. 23, coincident with the rear of the water/steel cylinder;
- n. 24, coincident with the front of the rear plate;
- n. 25, coincident with the rear of the assembly.

SDDR parameters

The irradiation schedule considered for ITER Cylinder SDDR benchmark is reported hereafter:

Table 2: Irradiation schedule (ITER mode SA2)

Source Intensity [n/s]	Δt irr.	Multiplicity
1.0714E+17	2 y	1
8.2500E+17	10 y	1
0	9 m	1
1.6667E+18	15 m	1
0	3290 s	->
2.0000E+19	400 s	17
0	3290 s	->
2.8000E+19	400 s	4

Two different cool-down times were considered in the photon tallies: 0s and 1e6 s (approx. 11.5 days). That is, these are the time interval waited after the irradiation phase has finished before tallying the SDDR and the photon flux.

The possible reactions allowed during the simulation are listed in the following table:

Table 3: List of possible reactions considered during the ITER Cylinder SDDR benchmark

Parent	Daughter
Cr50	Cr51
Cr52	Cr51
Mn55	Mn54
Fe54	Mn54
Fe54	Cr51
Fe56	Mn54
Fe58	Fe59
Co59	Co58
Co59	Co60
Co59	Fe59
Ni58	Co58
Ni60	Co60
Ni61	Co60
Ni61	Co60
Ni62	Fe59
Cu63	Cu62
Cu63	Co60
Cu65	Cu66
Ta181	Ta182
W182	Ta182
W186	W187

Tallies

Neutron flux, (decay) gamma flux and SDDR are the only tallied quantities. The following is a description of the tallies defined in the benchmark:

Tally n. 202 Neutron flux per energy bin [#/cm^2/s]. The flux is tallied in 16 energy bins ranging between 1E-10 MeV to 20 MeV. The flux is also binned geometrically using all surfaces described in §ref{sec:cylgeom}.

Tally n. 242 Total neutron flux [#/cm^2/s]. Same as Tally n. 202 but without the energy binning.

Tally n. 14 Gamma flux per energy bin in cell 10 [#/cm^2/s]. The flux is tallied in 16 energy bins ranging from 0.1 MeV to 20 MeV. The flux is tallied at both cool-down times.

Tally n. 34 Gamma flux per energy bin in cell 11 [#/cm^2/s]. The flux is tallied in 16 energy bins ranging from 0.1 MeV to 20 MeV. The flux is tallied at both cool-down times.

Tally n. 44 Gamma flux per energy bin in cell 12 [#/cm^2/s]. The flux is tallied in 16 energy bins ranging from 0.1 MeV to 20 MeV. The flux is tallied at both cool-down times.

Tally n. 54 Gamma flux per energy bin in cell 13 [#/cm^2/s]. The flux is tallied in 16 energy bins ranging from 0.1 MeV to 20 MeV. The flux is tallied at both cool-down times.

Tally n. 74 Total gamma flux [#/cm^2/s]. The flux is tallied only by cell (i.e. 10, 11, 12 and 13).

Tally n. 124 SSDR behind the plate [Sv/h]. The SDDR is computed at all cell tallies (i.e. 10, 11, 12 and 13) and at both cool-down times.

See also:

Related papers and contributions

- M. Youssef, Feder R., Batistoni P., Fischer U., Jakhar S., Konno C., Lough-lin M., and Villari R. "Benchmarking of the 3-D CAD-based Discrete Ordinates code "ATTILA" for dose rate calculations against experiments and MonteCarlo calculations". In: Fusion Engineering and Design 88 (2013), pp. 3033–3040.
- R. Pampin, A. Davis, J. Izquierdo, D. Leichtle, M.D. Loughlin, J. Sanz, A.Turner, R. Villari, and P.P.H. Wilson. "Developments and needs in nuclearanalysis of fusion technology". In: Fusion Engineering and Design 88 (2013), pp. 454–460.

6.2 Experimental Benchmarks

6.2.1 Oktavian

Experimental results derived from Oktavian experiments are publicly accessible at the CoNDERC database which is mantained by the IAEA Nuclear Data Section and built upon the database of shielding experiments (SINBAD), hosted by the RSICC and jointly mantained with the NEA data bank.

OKTAVIAN is an experimental facility located at the Osaka University which has been operative since 1981. It consists of an intense deuterium-tritium (D-T) fusion neutron source (up to 3E+12 n/s) that has been used during the years for many experiments on high energy neutrons transport. Among them, many Time Of Flight (TOF) experiments were conducted and their results have been introduced in SINBAD. These experiments consists in placing the neutron source inside a sphere composed only by a specific material of interest and measuring the leakage photon spectra exiting from such sphere with the use of detectors. The photon energy measure is performed indirectly measuring the time of flight, which is then converted into a velocity.

Geometry and run parameters

An accelerated deuteron beam is led through a narrow tube to the centre of a sphere (every time composed by a different material) where pulsed 14.1 MeV monochromatic neutrons were produced by the d-t fusion reaction. The source is regarded to be 14 MeV monochromatic. Neutron leakage current spectrum of neutrons was measured in "absolute values" by the time-of-flight technique between 10 keV and 14 MeV, about 9.5 m from the sphere centre. Because of the presence of the collimators the detectors could not see the entire surface of the sphere, but only the solid angle of 17.28° from the sphere centre.

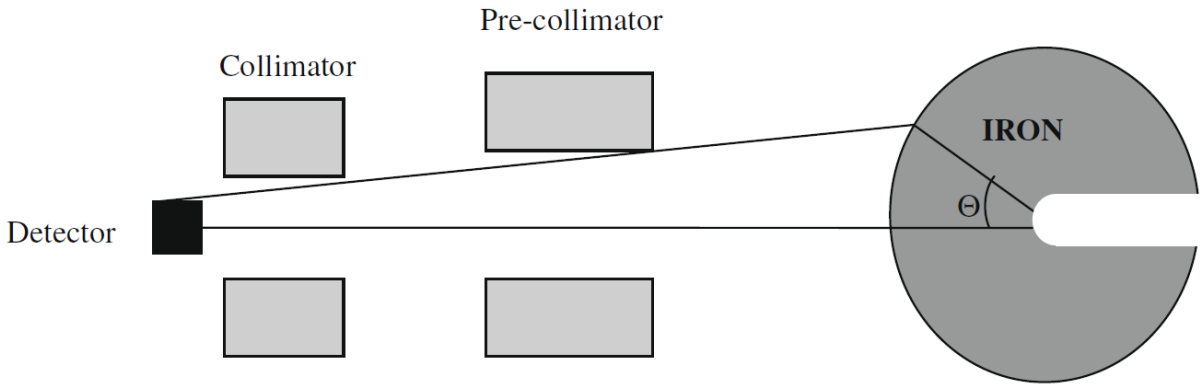


Fig. 13: Simplified layout of the OKTAVIAN Fe experimental setup (not in scale).

Tallies

Only two tallies are defined for each input:

Tally n. 21 Neutron leakage current $[#/cm^2]$ per source particle. 134 energy bins were defined spanning from 0.1 MeV to 20.6 MeV.

Tally n. 41 Photon leakage current $[#/cm^2]$ per source particle. 57 energy bins were defined spanning from 0.5 MeV to 10.5 MeV.

Since experimental results are provided as flux per unit lethargy, the tally results are manipulated as follows:

$$d\Phi_u = d\Phi/d(\log E)$$

See also:

Related papers and contributions:

- A. Milocco, A. Trkov and I. A. Kodeli, 2010, "The OKTAVIAN TOF experiments in SINBAD: Evaluation of the experimental uncertainties", *Annals of Nuclear Energy*, **37** 443-449
- I.Kodeli, E. Sartori and B. Kirk, "SINBAD - Shielding Benchmark Experiments - Status and Planned Activities", *Proceedings of the ANS 14th Biennial Topical Meeting of Radiation Protection and Shielding Division*, Carlsbad, New Mexico (April 3-6, 2006)

6.2.2 Frascati Neutron Generator

Important: This benchmark input cannot be distributed directly with JADE. The user must have a valid SINBAD license and contact the JADE team in order to obtain the input.

The Frascati Neutron Generator (FNG) is an experimental facility designed and built by ENEA (the italian New Technology, Energy and Ambient Body) in Frascati, Italy. The installation is able to produce 14 MeV neutrons based on the $T(d,n)\alpha$ fusion reaction and it is able to produce up to 5E+11 n/s.

One of the key experiments that have been conducted at the FNG is the neutron irradiation experiment, where a mock-up of the outer vacuum vessel region of ITER was irradiated by means of 14 MeV neutrons for a sufficiently long time in order to achieve activation levels similar to the ones that are expected to be reached at the ITER end of life. Two distinct irradiation campaigns were conducted in May and August 2000 and, among other things, the SDDR

values after different cooling time intervals were measured. Many benchmarks activities have been performed using the experiment in the past, and the benchmark is also included in the SINBAD database (listed as fng_dose).

Geometry

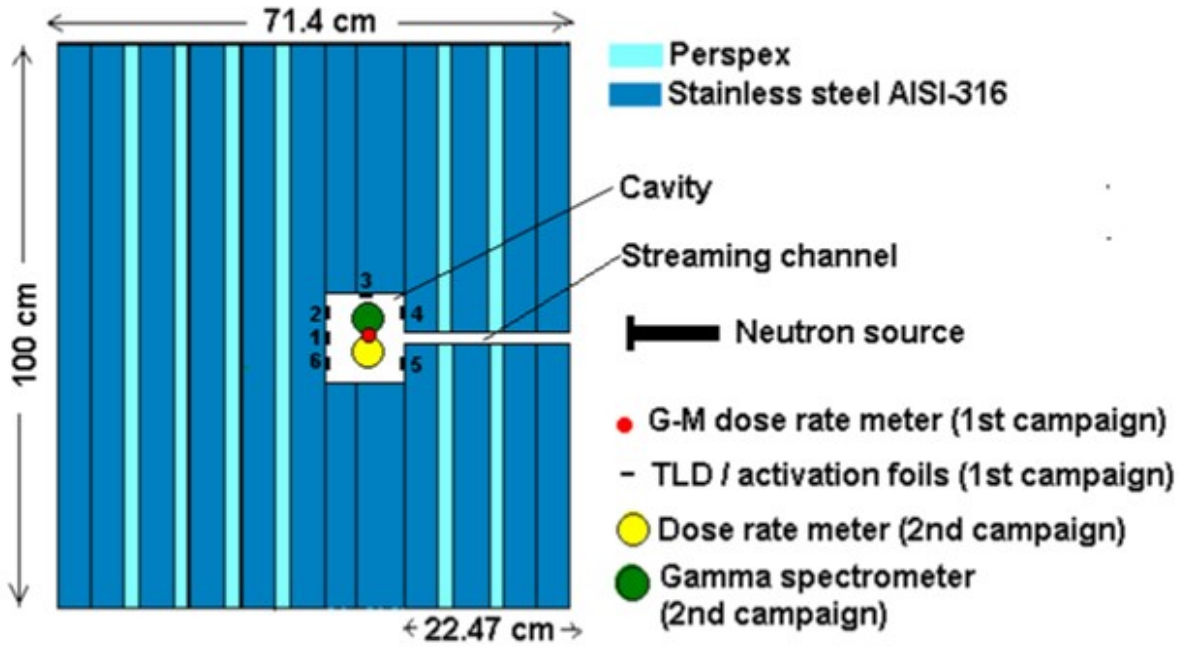


Fig. 14: FNG SDDR experiment layout

In the FNG, a deuterium beam is accelerated up to 300 KeV by means of a linear electro-static tube towards a target rich in tritium generating a 14 MeV neutron source. These are the neutrons that were used to irradiate the experimental assembly which consisted of a block of stainless steel and water equivalent material (perspex) with total thickness of 71.4 cm, and a lateral size of 100 cm x 100 cm. A cavity was obtained within the block (12.6 cm in the beam direction, 11.98 cm high) behind a 22.47 cm thick shield. A void channel (2.7 cm inner diameter) was included in front of the cavity to study the effect of streaming paths in the bulk shield. A squared box was used to locate detectors inside the cavity, with 2 mm thick bottom and lateral walls. Measurements were taken in the cavity, during the irradiation and after shut-down, to obtain the local neutron flux, the decay gamma-ray spectra and the dose rates for different cooling times.

JADE MCNP input template was realized starting from the MCNP inputs provided in the SINBAD database and, for this reason, it cannot be freely distributed together with the JADE source code. Cell and surface card were left untouched as well as the material composition. D1S-Uned specific cards were suitably added.

SDDR Parameters

The next two tables describe the equivalent schedules considered for respectively the 1st and 2nd irradiation campaign conducted at the FNG.

Table 4: Equivalent schedule of the 1st FNG irradiation campaign

Δt [s]	Δt [min]	Neutron Intensity [n/s]
19440	324	2.32E+10
61680	1028	0
32940	549	2.87E+10
54840	914	0
15720	262	1.90E+10
6360	106	0
8940	149	1.36E+10

Table 5: Equivalent schedule of the 2nd FNG irradiation campaign

Δt [s]	Δt [min]	Neutron Intensity [n/s]
1748	29	3.04E+10
7820	130	4.28E+10
54140	902	0
22140	369	4.29E+10
900	15	0
3820	64	3.38E+10
420	7	0
140	2	2.86E+10

The experimentally measured SDDR values at different cooling times are reported in the next tables for the 1st and 2nd irradiation campaigns.

Table 6: Experimental measure of the SDDR during 1^{st} FNG irradiation campaign

Cooldown Time [d]	Cooldown Time [s]	Experimental SDDR [Sv/h]	Relative Error
1	86400	2.46E-06	0.1
7	604800	6.99E-07	0.1
15	1296000	4.95E-07	0.1
30	2592000	4.16E-07	0.1
60	5184000	3.16E-07	0.1

Table 7: Experimental measure of the SDDR during 2^{nd} FNG irradiation campaign

Cooldown Time [s]	Cooldown Time [h]	Cooldown Time [d]	Experimental SDDR [Sv/h]	Relative Error
4380	1.22	0.05	4.88E-04	3.89E-02
6180	1.72	0.07	4.15E-04	3.86E-02
7488	2.08	0.09	3.75E-04	4.00E-02
11580	3.22	0.13	2.68E-04	3.73E-02
17280	4.80	0.20	1.73E-04	4.05E-02
24480	6.80	0.28	1.01E-04	3.96E-02
34080	9.47	0.39	5.06E-05	3.95E-02
45780	12.72	0.53	2.30E-05	3.91E-02
57240	15.90	0.66	1.17E-05	4.27E-02
72550	20.15	0.84	5.80E-06	3.97E-02
90720	25.20	1.05	3.56E-06	3.93E-02
132000	36.67	1.53	2.43E-06	3.70E-02
212400	59.00	2.46	1.78E-06	3.93E-02
345600	96.00	4.00	1.22E-06	4.10E-02
479300	133.14	5.55	9.52E-07	3.89E-02
708500	196.81	8.20	7.59E-07	3.95E-02
1050000	291.67	12.15	6.67E-07	3.90E-02
1670000	463.89	19.33	6.13E-07	3.92E-02
1710000	475.00	19.79	6.14E-07	3.91E-02

When simulating with the D1S approach, in order to reduce the computation time it is good practice to individuate the subset of decay isotopes which contribute the most to the dose rate. This subset will depend from the unirradiated material composition and the cool-down time that are considered. In order to do so, preliminary activation calculation are usually performed with the help of activation codes like FISPACT or ACAB. Fortunately these studies have been already conducted both during the D1S libraries initial V&V procedure and when the experimental results were tested for the first time. The next plot lists the isotopes contributing cumulatively to more than 95% of the dose rate during the first irradiation campaign.

At this point, the D1S reaction file can be generated: it will include all reactions that can originate in the material (i.e. that are also available in the activation library) which result in the creation of one of the daughters of interest. The D1S irradiation file will simply contain those daughters which are generated by at least one reaction. All of this implies that a comparison between two different libraries can often not be an exact one. Indeed, it is quite common that to a new library release corresponds an increase in the number of available reactions. Nevertheless, this is in line with the philosophy of JADE. If the Sphere benchmarks are the primary tools that should be used to identify specific inconsistencies at the single cross section level among libraries, all other benchmarks have a slightly different scope which is to show how big is the impact of these inconsistencies on more realistic applications.

Tallies

The only tallied result for the FNG benchmark is the dose rate at the dosimeter location inside the cavity (tally n.4).

See also:

Related papers:

- M. Martone, M. Angelone, and M. Pillon. "The 14 MeV Frascati neutrongenerator". In:Journal of Nuclear Materials 212-215 (1994). Fusion ReactorMaterials, pp. 1661–1664
- P. Batistoni, M. Angelone, L. Petrizzi, and M. Pillon. "Benchmark Experimentfor the Validation of Shut Down Activation and Dose Rate in a Fusion Device".In: Journal of Nuclear Science and Technology 39.sup2 (2002),

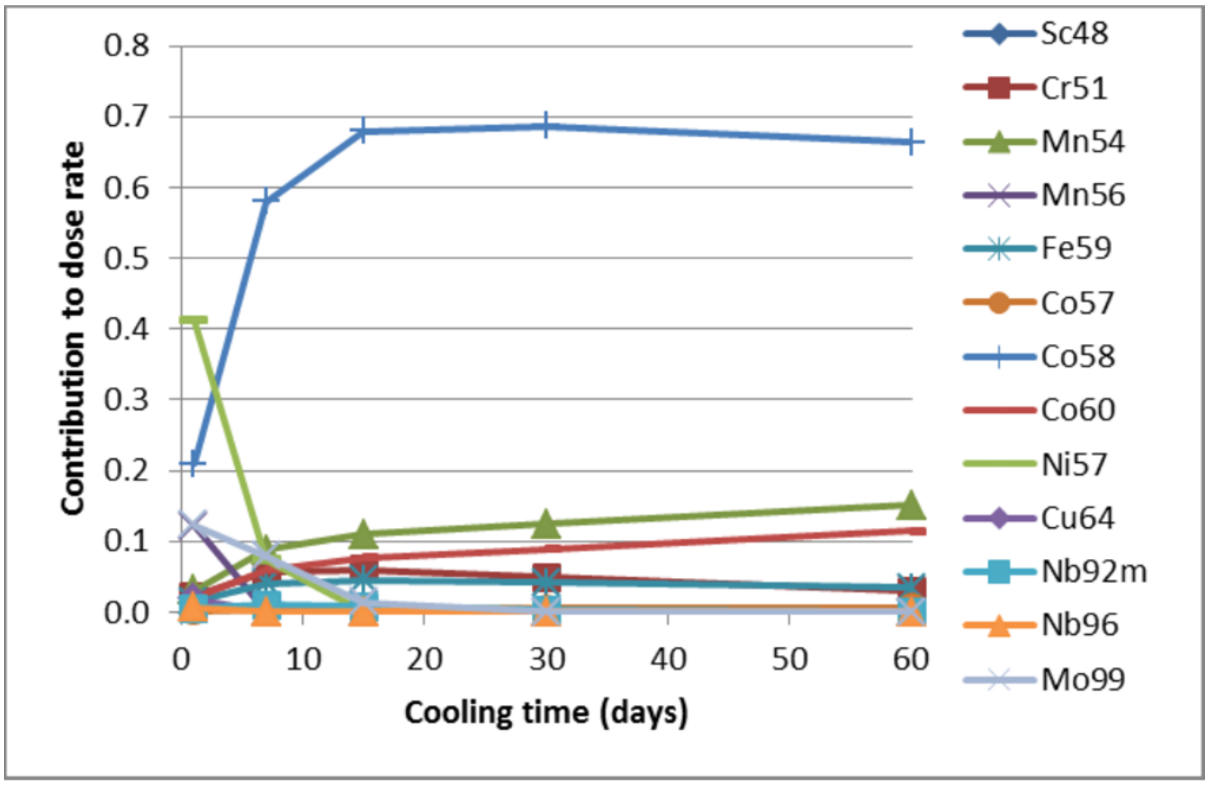


Fig. 15: Isotope contribution to the the dose during the first FNG irradiation campaign

pp. 974–977.

- K. Seidel, Y. Chen, U. Fischer, H. Freiesleben, D. Richter, and S. Unholzer."Measurement and analysis of dose rates and gamma-ray fluxes in an ITERshut-down dose rate experiment". In:Fusion Engineering and Design 63-64 (2002), pp. 211–215.
- R. Pampin, A. Davis, R.A. Forrest, D.A. Barnett, I. Davis, and M.Z. Youssef."Status of novel tools for estimation of activation dose". In:Fusion Engineeringand Design 85.10 (2010). Proceedings of the Ninth International Symposiumon Fusion Nuclear Technology, pp. 2080–2085.
- J. Sanz, O. Cabellos, and N. Garcia-Herranz. Inventory Code for Nuclear Applications: User's Manual V. 2008. RSICC. 2008.

CHAPTER

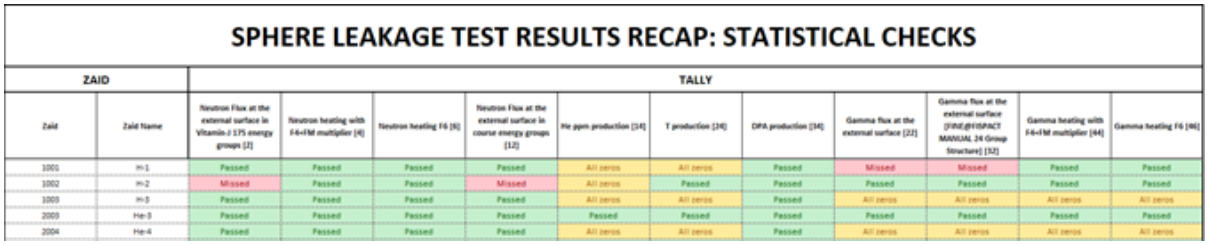
SEVEN

POST-PROCESSING GALLERY

7.1 Excel output

7.1.1 Benchmark specific

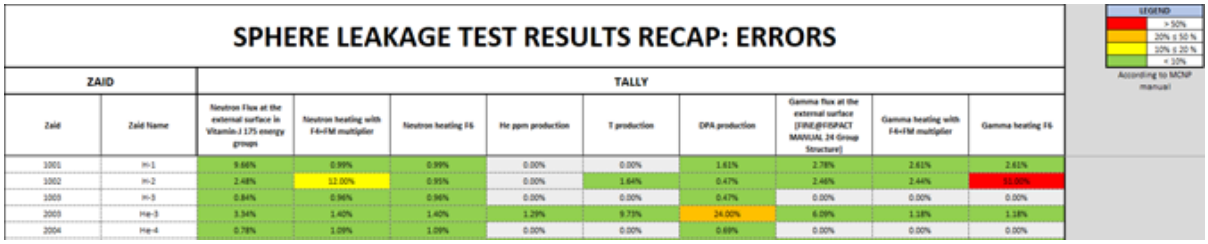
Sphere Leakage



SPHERE LEAKAGE TEST RESULTS RECAP: STATISTICAL CHECKS

ZAID		TALLY										
Zaid	Zaid Name	Neutron Flux at the external surface in Vitamin-J 175 energy groups [2]	Neutron heating with F4+FM multiplier [4]	Neutron heating F6 [6]	Neutron Flux at the external surface in course energy groups [12]	He ppm production [14]	T production [24]	DPA production [34]	Gamma flux at the external surface [22]	Gamma flux at the external surface [FINE@FISPACT MANUAL 24 Group Structure] [32]	Gamma heating with F4+FM multiplier [44]	Gamma heating F6 [46]
1001	H-1	Passed	Passed	Passed	Passed	All zeros	All zeros	Passed	Missed	Missed	Passed	Passed
1002	H-2	Missed	Passed	Passed	Missed	All zeros	Passed	Passed	Passed	Passed	Passed	Passed
1003	H-3	Passed	Passed	Passed	Passed	All zeros	All zeros	Passed	All zeros	All zeros	All zeros	All zeros
2003	He-3	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed	Passed
2004	He-4	Passed	Passed	Passed	Passed	All zeros	All zeros	Passed	All zeros	All zeros	All zeros	All zeros

Fig. 1: 10 MCNP statistical checks for each zaid and tally

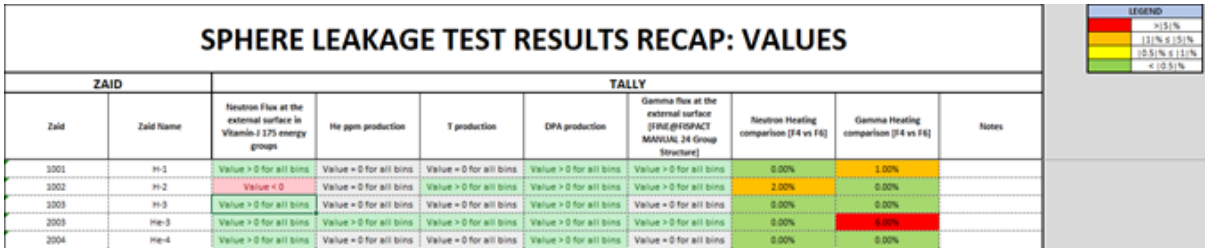


SPHERE LEAKAGE TEST RESULTS RECAP: ERRORS

ZAID		TALLY								
Zaid	Zaid Name	Neutron Flux at the external surface in Vitamin-J 175 energy groups	Neutron heating with F4+FM multiplier	Neutron heating F6	He ppm production	T production	DPA production	Gamma flux at the external surface [FINE@FISPACT MANUAL 24 Group Structure]	Gamma heating with F4+FM multiplier	Gamma heating F6
1001	H-1	9.66%	0.99%	0.99%	0.00%	0.00%	1.61%	2.78%	2.61%	2.61%
1002	H-2	2.48%	12.00%	0.95%	0.00%	1.84%	0.47%	2.46%	2.44%	55.00%
1003	H-3	0.84%	0.96%	0.96%	0.00%	0.00%	0.47%	0.00%	0.00%	0.00%
2003	He-3	3.34%	1.40%	1.40%	1.29%	9.73%	24.00%	6.09%	1.18%	1.18%
2004	He-4	0.78%	1.09%	1.09%	0.00%	0.00%	0.69%	0.00%	0.00%	0.00%

LEGEND
> 50%
20% ≤ 50 %
10% ≤ 20 %
< 10%
According to MCNP manual

Fig. 2: Statistical error associated with each tally for each zaid



SPHERE LEAKAGE TEST RESULTS RECAP: VALUES

ZAID		TALLY							
Zaid	Zaid Name	Neutron Flux at the external surface in Vitamin-J 175 energy groups	He ppm production	T production	DPA production	Gamma flux at the external surface [FINE@FISPACT MANUAL 24 Group Structure]	Neutron Heating comparison [F4 vs F6]	Gamma Heating comparison [F4 vs F6]	Notes
1001	H-1	Value > 0 for all bins	Value = 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	0.00%	1.00%	
1002	H-2	Value < 0	Value = 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	2.00%	0.00%	
1003	H-3	Value > 0 for all bins	Value = 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	Value = 0 for all bins	0.00%	0.00%	
2003	He-3	Value > 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	Value > 0 for all bins	0.00%	6.00%	
2004	He-4	Value > 0 for all bins	Value = 0 for all bins	Value = 0 for all bins	Value > 0 for all bins	Value = 0 for all bins	0.00%	0.00%	

LEGEND
> |5| %
|1| % ≤ |5| %
|0.5| % < |1| %
< |0.5| %

Fig. 3: Consistency checks on zaid tally results

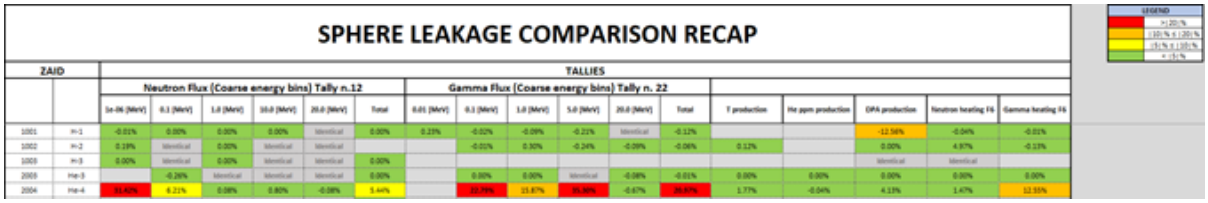


Fig. 4: Comparison of tally results for each zaid

Oktavian

"C/E (mean +/- σ)"

Particle	Energy Range [MeV]	FENDL 3.1d	FENDL 3.2 beta
Neutron	0.1 - 1	1.01 +/- 0.44	1.03 +/- 0.44
	1.0 - 5	0.9 +/- 0.08	0.9 +/- 0.08
	10.0 - 20	3.38 +/- 9.97	3.38 +/- 9.95
	5.0 - 10	0.91 +/- 0.11	0.9 +/- 0.11
Photon	0.1 - 1	0.42 +/- 0.26	0.44 +/- 0.27
	1.0 - 5	2.21 +/- 1.06	2.21 +/- 1.05
	10.0 - 20	18.45 +/- nan	17.96 +/- nan
	5.0 - 10	7.87 +/- 4.76	7.75 +/- 4.74

Fig. 5: C/E table summarized per energy range

LIBRARY:	31c

10 MCNP Statistical Checks

Tally Number	Tally name	Result
44	H in 316SS appm/FPY	Missed
54	T in 316SS appm/FPY	Missed
64	Cu dpa/FPY	Missed
74	He in CuBeNi appm/FPY	Missed
84	H in CuBeNi appm/FPY	Missed
94	T in CuBeNi appm/FPY	Missed
104	Ni dpa/FPY	Missed
114	He in Inconel appm/FPY	Missed
124	H in Inconel appm/FPY	Missed
134	T in Inconel appm/FPY	Missed
144	He in Be appm/FPY	Passed

Fig. 6: 10 MCNP statistical checks recap

LIBRARY: 32c

according to MCNP

LEGEND

> 50%
20% ≤ 50 %
10% ≤ 20 %
< 10%

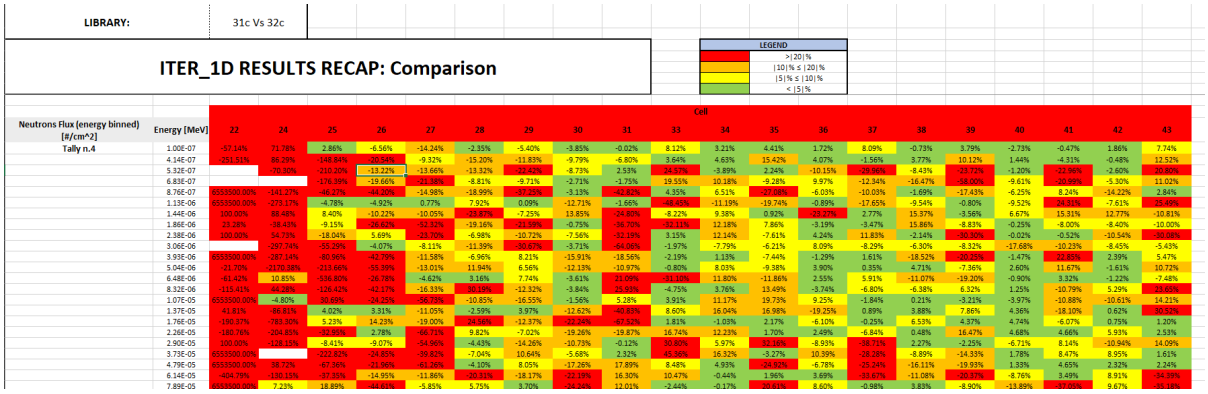
ITER_1D RESULTS RECAP: Errors

Neutrons Flux (energy binned) [#/cm^2]		Cell																			
Tally n.4	Energy [MeV]	22	24	25	26	27	28	29	30	31	33	34	35	36	37	38	39	40	41	42	43
	1.00E-07	0.9999	0.6455	0.2003	0.1262	0.1061	0.0888	0.0927	0.0844	0.1568	0.071	0.0623	0.0552	0.0532	0.0476	0.0448	0.0419	0.0406	0.0359	0.0343	0.0342
	4.14E-07	0.9999	0.6576	0.2709	0.1332	0.1108	0.0942	0.0904	0.0862	0.1556	0.0996	0.0634	0.0769	0.0523	0.0648	0.0446	0.058	0.0404	0.0552	0.0354	0.0553
	5.32E-07	0	0.7164	0.3838	0.1709	0.1424	0.1321	0.1199	0.1075	0.2255	0.164	0.0851	0.1663	0.0783	0.1344	0.0667	0.1208	0.0662	0.1274	0.063	0.1298
	6.83E-07											0									
	8.76E-07	0	0.7077	0.3926	0.172	0.1442	0.1144	0.129	0.1108	0.2389	0.1675	0.0897	0.1592	0.0771	0.1243	0.0701	0.1268	0.0623	0.1155	0.0655	0.134
	1.13E-06	0	0.9998	0.3266	0.1636	0.1452	0.1304	0.1302	0.1203	0.2367	0.1914	0.0903	0.1633	0.0773	0.1157	0.0689	0.1196	0.0624	0.1145	0.0662	0.154
	1.44E-06	0.9999	0.8442	0.3543	0.1742	0.1512	0.1324	0.1181	0.1058	0.2186	0.1739	0.0861	0.164	0.0787	0.1145	0.07	0.1209	0.0629	0.1033	0.0609	0.1631
	1.86E-06	0.9999	0.6581	0.3545	0.1634	0.157	0.1375	0.1254	0.1162	0.2288	0.1721	0.0855	0.1536	0.075	0.1143	0.0626	0.122	0.0652	0.1207	0.0642	0.1743
	2.38E-06	0.9999	0.5151	0.3185	0.1848	0.1548	0.1402	0.114	0.1146	0.2155	0.1716	0.0893	0.1553	0.0827	0.1217	0.068	0.1206	0.062	0.1178	0.0637	0.1683
	3.06E-06	0	0.7937	0.2756	0.1792	0.1554	0.1363	0.1177	0.1121	0.2405	0.1716	0.0919	0.1505	0.0803	0.1096	0.0689	0.1252	0.0638	0.1237	0.0651	0.1695
	3.93E-06	0	0.9999	0.3713	0.2005	0.149	0.142	0.1138	0.1095	0.2217	0.1645	0.0863	0.1654	0.084	0.1145	0.0667	0.115	0.0616	0.1256	0.0637	0.1479
	5.04E-06	0.9999	0.9999	0.3682	0.1769	0.1599	0.1377	0.1122	0.1107	0.1951	0.1528	0.0916	0.1566	0.0814	0.1035	0.0672	0.1169	0.0607	0.1256	0.07	0.1656
	6.48E-06	0.7096	0.853	0.3802	0.1927	0.1581	0.1287	0.1258	0.1129	0.1781	0.1623	0.0909	0.1451	0.0766	0.103	0.069	0.1116	0.0672	0.1241	0.0649	0.1641
	8.32E-06	0.9228	0.6615	0.3316	0.166	0.1598	0.1399	0.1171	0.1191	0.1775	0.1541	0.0878	0.1422	0.081	0.1143	0.0688	0.1226	0.0638	0.1244	0.064	0.1423
	1.07E-05	0	0.724	0.3292	0.1764	0.1669	0.1348	0.1135	0.1075	0.1658	0.1521	0.0927	0.1606	0.081	0.1141	0.0694	0.1163	0.0677	0.1148	0.0644	0.1487

Fig. 7: Statistical errors associated with the tally results

H in 316SS appm/FPY	Cells	Value
Tally n.44	21	-37.43%
	26	-24.62%
	27	-17.63%
	28	-7.39%
	29	-0.58%
	30	-22.90%
	33	-17.75%
	35	-3.39%
	37	-9.85%
	39	-6.89%
	41	-4.62%
	43	-4.75%
	45	-3.56%
	47	-5.62%
	55	-5.05%
	57	-4.42%
	59	-3.55%
	61	-2.28%
	63	-2.45%
	65	-5.27%
	67	-5.97%
	69	-2.21%
	72	0.51%
	73	-3.17%
	74	-5.71%
	75	-7.38%
	76	-8.82%
	77	0.33%
	78	3.28%
	79	0.68%
	80	-0.58%

Fig. 8: Print of a single binned tally



LIBRARY: 31c Vs 32c

ITER_1D RESULTS RECAP: Comparison

LEGEND
>|20|%
|10|% ≤ |20|%
|5|% ≤ |10|%
< |5|%

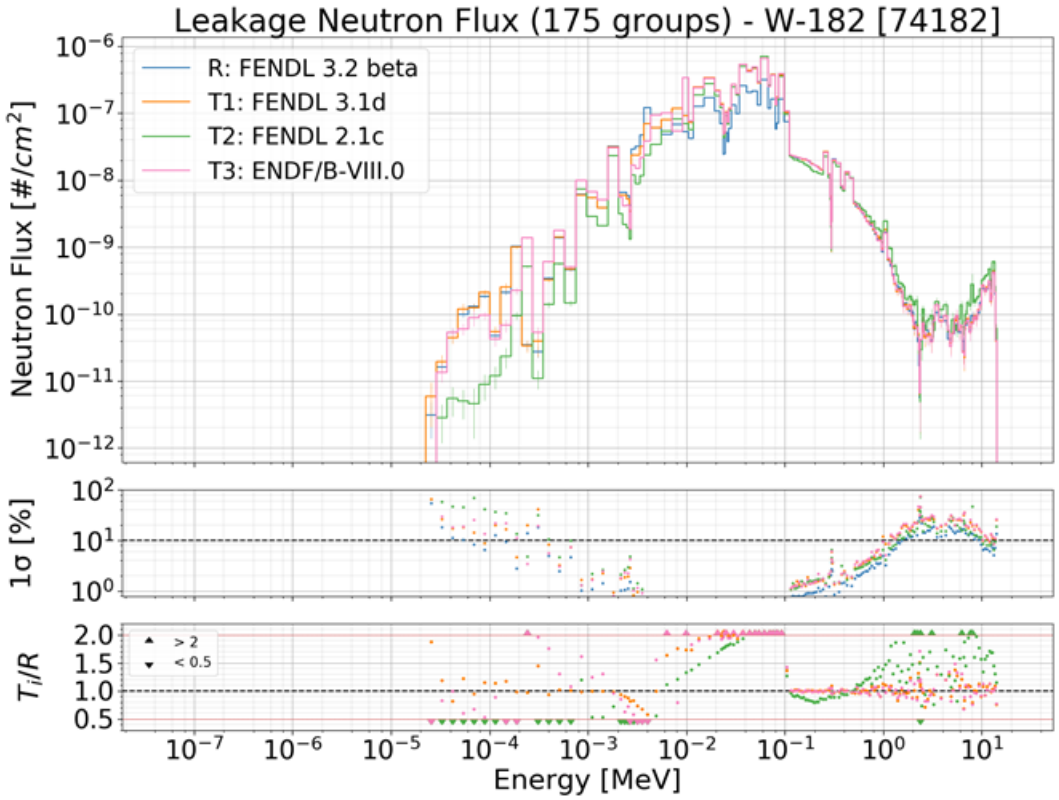
Neutrons Flux (energy binned) [#/cm^2] Tally n.4	Energy [MeV]	Cell 22	24	25	26	27	28	29	30	31	33	34	35	36	37	38	39	40	41	42	43
	1.00E-07	-57.14%	71.78%	2.86%	-6.56%	-14.24%	-2.35%	-5.40%	-3.85%	-0.02%	8.12%	3.21%	4.41%	1.72%	8.09%	-0.73%	3.79%	-2.73%	-0.47%	1.86%	7.74%
	4.14E-07	-251.31%	88.29%	-148.84%	-20.94%	-9.32%	-15.20%	-11.83%	-9.79%	-6.80%	3.64%	4.63%	15.42%	4.07%	-1.56%	3.77%	10.12%	1.44%	-4.31%	-0.48%	12.52%
	5.32E-07		-70.30%	-210.20%	-13.22%	-13.66%	-13.32%	-22.42%	-8.73%	2.53%	24.57%	-3.89%	2.24%	-10.15%	-29.96%	-8.43%	-23.72%	-1.20%	-22.96%	-2.60%	20.80%
	6.83E-07			-176.39%	-19.66%	-21.38%	-8.81%	-9.71%	-2.71%	-1.75%	19.55%	10.18%	-9.28%	9.97%	-12.34%	-16.47%	-58.00%	-9.61%	-20.99%	-5.30%	11.02%
	8.76E-07	6535500.00%	-141.27%	-46.73%	-24.20%	-14.98%	-18.99%	-37.85%	-1.19%	-42.92%	4.35%	6.51%	-20.84%	-6.03%	-10.03%	-1.69%	-17.43%	-6.25%	8.24%	-14.22%	2.84%
	1.13E-06	6535500.00%	-273.17%	-4.78%	-4.92%	0.77%	7.92%	0.09%	-12.71%	-1.66%	-48.45%	-11.19%	-19.74%	-0.89%	-17.65%	-9.54%	-0.80%	-9.52%	-24.35%	-7.61%	25.49%
	1.44E-06	100.00%	88.48%	8.40%	-10.22%	-10.05%	-23.87%	-7.25%	13.85%	-24.80%	-8.22%	9.38%	0.92%	-28.27%	2.77%	15.37%	-3.56%	6.67%	15.31%	12.77%	-10.81%
	1.86E-06	23.28%	-38.43%	-9.15%	-26.62%	-52.32%	-19.16%	-21.59%	-0.75%	-36.70%	-32.11%	12.18%	7.86%	-3.19%	-3.47%	15.86%	-8.83%	-0.25%	-8.00%	-8.40%	-10.00%
	2.38E-06	100.00%	54.73%	-18.04%	5.69%	-23.10%	-6.98%	-10.72%	-7.56%	-52.19%	3.15%	12.14%	-7.61%	4.24%	11.83%	-2.14%	-30.86%	-0.02%	-0.52%	-10.54%	-31.08%
	3.06E-06		-287.74%	-35.29%	-4.07%	-8.11%	-11.39%	-20.47%	-3.71%	-64.06%	-1.87%	-7.79%	-6.21%	8.09%	-2.29%	-6.30%	-8.32%	-17.68%	-10.23%	-8.45%	-5.43%
	3.93E-06	6535500.00%	-287.14%	-80.96%	-42.79%	-11.58%	-6.96%	8.21%	-15.91%	-18.56%	-2.19%	1.13%	-7.44%	-1.29%	1.61%	-18.52%	-20.25%	-1.47%	22.85%	2.39%	5.47%
	5.04E-06	-21.70%	-2170.38%	-213.66%	-55.39%	-13.01%	11.94%	6.56%	-12.13%	-10.97%	-0.80%	8.03%	-9.38%	3.90%	0.35%	4.71%	-7.36%	2.60%	11.67%	-1.61%	10.72%
	6.48E-06	-61.42%	10.85%	-536.80%	-26.78%	-4.62%	3.16%	7.74%	-3.61%	21.09%	-31.10%	11.80%	-11.86%	2.55%	5.91%	-11.07%	-19.20%	-0.90%	3.32%	-1.22%	-7.48%
	8.32E-06	-115.41%	-44.28%	-126.42%	-42.17%	-16.33%	-30.14%	-12.32%	-3.84%	-26.98%	-4.75%	3.76%	13.49%	-3.74%	-6.80%	-6.38%	6.32%	1.25%	-10.79%	5.29%	-24.83%
	1.07E-05	6535500.00%	-4.80%	-30.89%	-24.25%	-56.73%	-10.85%	-16.55%	-1.56%	5.28%	3.91%	11.17%	19.75%	9.25%	-1.84%	0.21%	-3.21%	-3.97%	-10.88%	-10.61%	14.21%
	1.37E-05	41.81%	-86.81%	4.02%	3.31%	-11.05%	-2.59%	3.97%	-12.62%	-40.83%	8.60%	16.04%	16.98%	-19.25%	0.89%	3.88%	7.86%	4.36%	-18.10%	0.62%	30.52%
	1.76E-05	-190.37%	-783.30%	5.23%	14.23%	-19.00%	-24.56%	-12.37%	-22.24%	-67.82%	1.81%	-1.03%	2.17%	-6.10%	-0.25%	6.53%	4.37%	4.74%	-6.07%	0.75%	1.20%
	2.26E-05	-180.76%	-204.85%	-29.86%	2.78%	-66.71%	9.82%	-7.02%	-19.26%	-19.87%	16.74%	12.23%	1.70%	2.49%	-6.84%	0.48%	16.47%	4.68%	4.66%	5.93%	2.53%
	2.90E-05	-100.00%	-128.15%	-8.41%	-9.07%	-54.96%	-4.43%	-14.26%	-10.73%	-0.12%	30.80%	5.97%	32.16%	-8.93%	-38.71%	2.27%	-2.25%	-6.71%	8.14%	-10.94%	14.09%
	3.73E-05	6535500.00%		222.82%	-24.85%	-39.82%	-7.04%	10.64%	-5.68%	2.32%	45.36%	16.32%	-3.27%	10.39%	-28.28%	-8.89%	-14.33%	1.78%	8.47%	8.95%	1.61%
	4.79E-05	6535500.00%	38.72%	-67.36%	-21.96%	-61.26%	-4.10%	8.05%	-17.26%	17.89%	8.48%	4.93%	-24.92%	-6.78%	-25.24%	-16.11%	-19.93%	1.33%	4.65%	2.32%	2.24%
	6.14E-05	-404.79%	-130.11%	-97.35%	-14.95%	-11.86%	-20.31%	-18.17%	-22.19%	16.30%	10.47%	-0.44%	1.96%	3.69%	-31.67%	-11.08%	-20.87%	-8.76%	3.49%	8.91%	-34.39%
	7.89E-05	6535500.00%	7.23%	18.89%	-34.69%	-5.85%	5.75%	3.70%	-24.34%	12.01%	-2.44%	-0.17%	29.61%	8.60%	-0.98%	3.83%	-8.90%	-13.89%	-17.06%	9.67%	-35.18%

Fig. 9: Print of a double binned tally

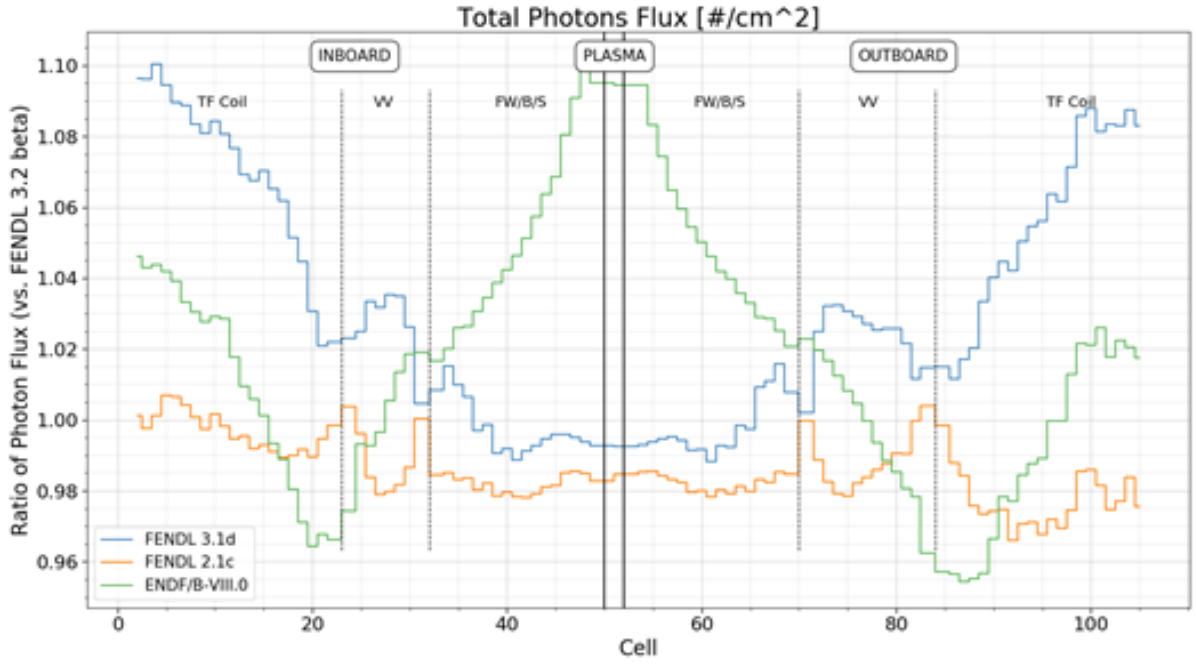
7.1.2 General output

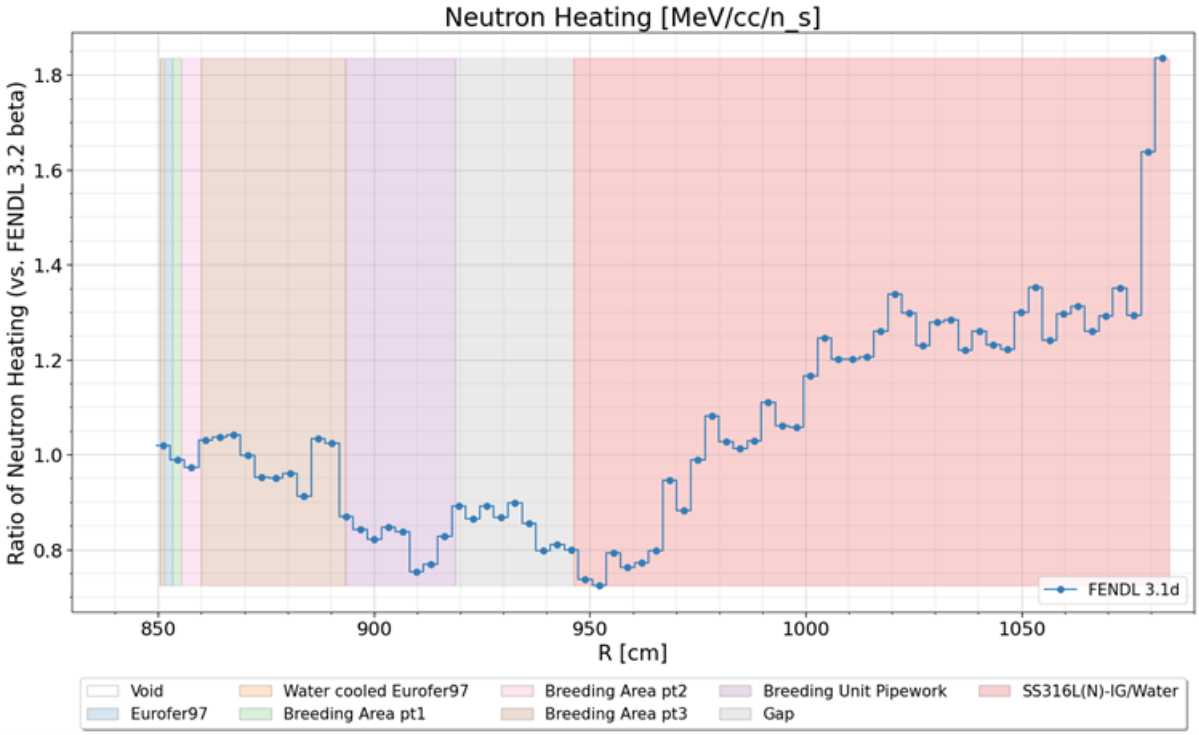
7.2 Plots Atlas

7.2.1 Binned graph

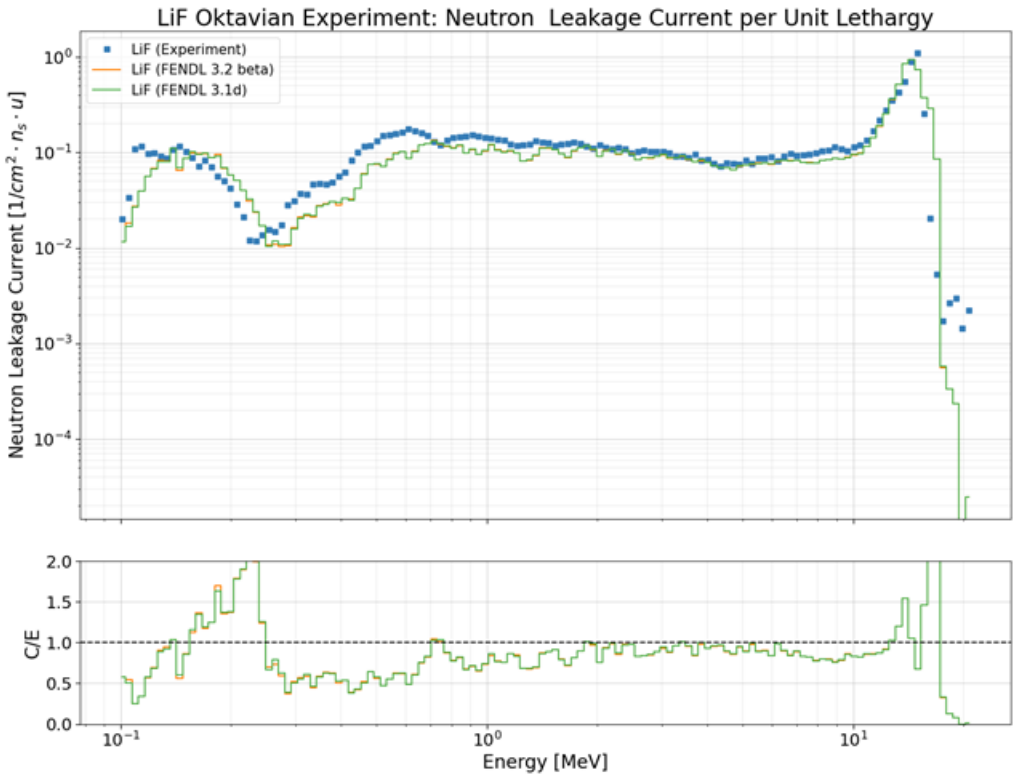


7.2.2 Ratio Graph

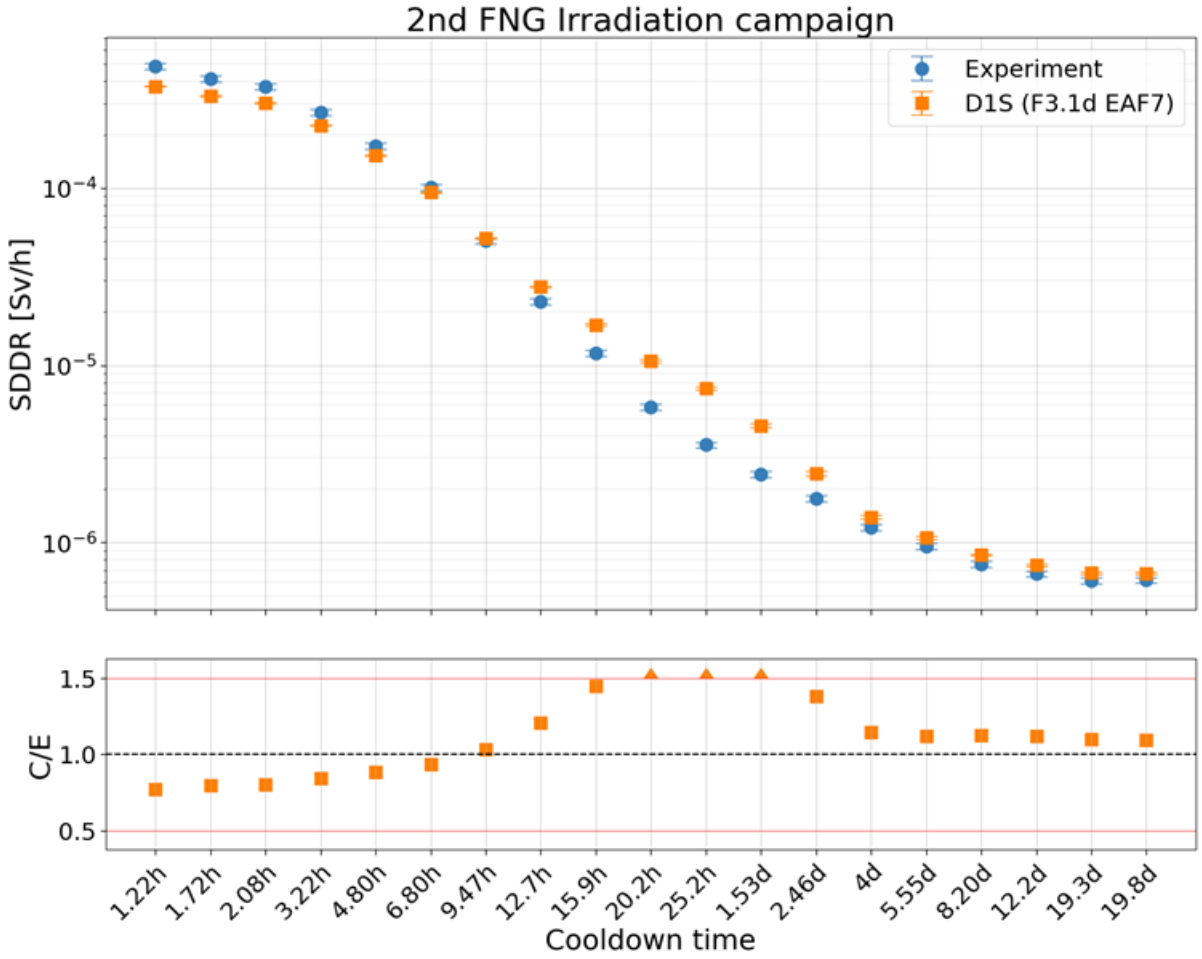




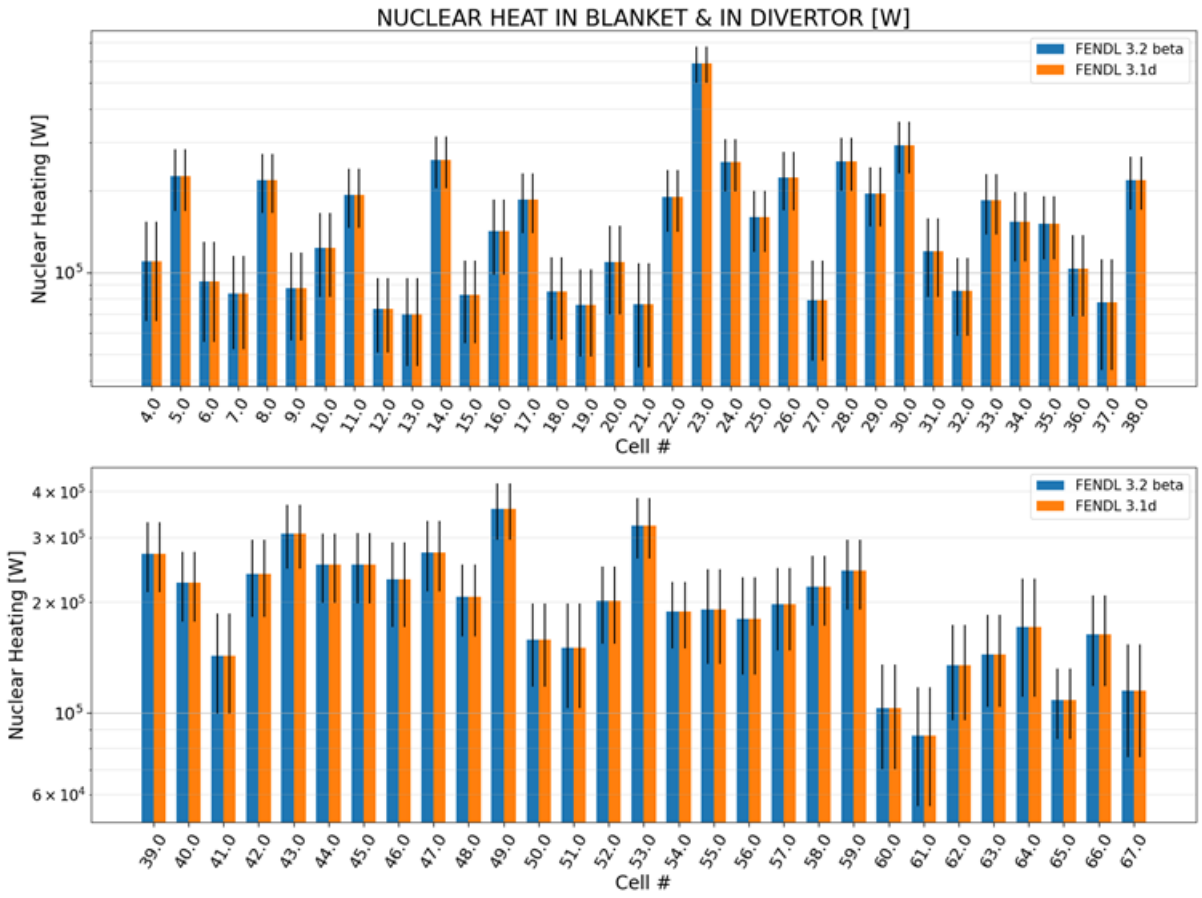
7.2.3 Experimental points



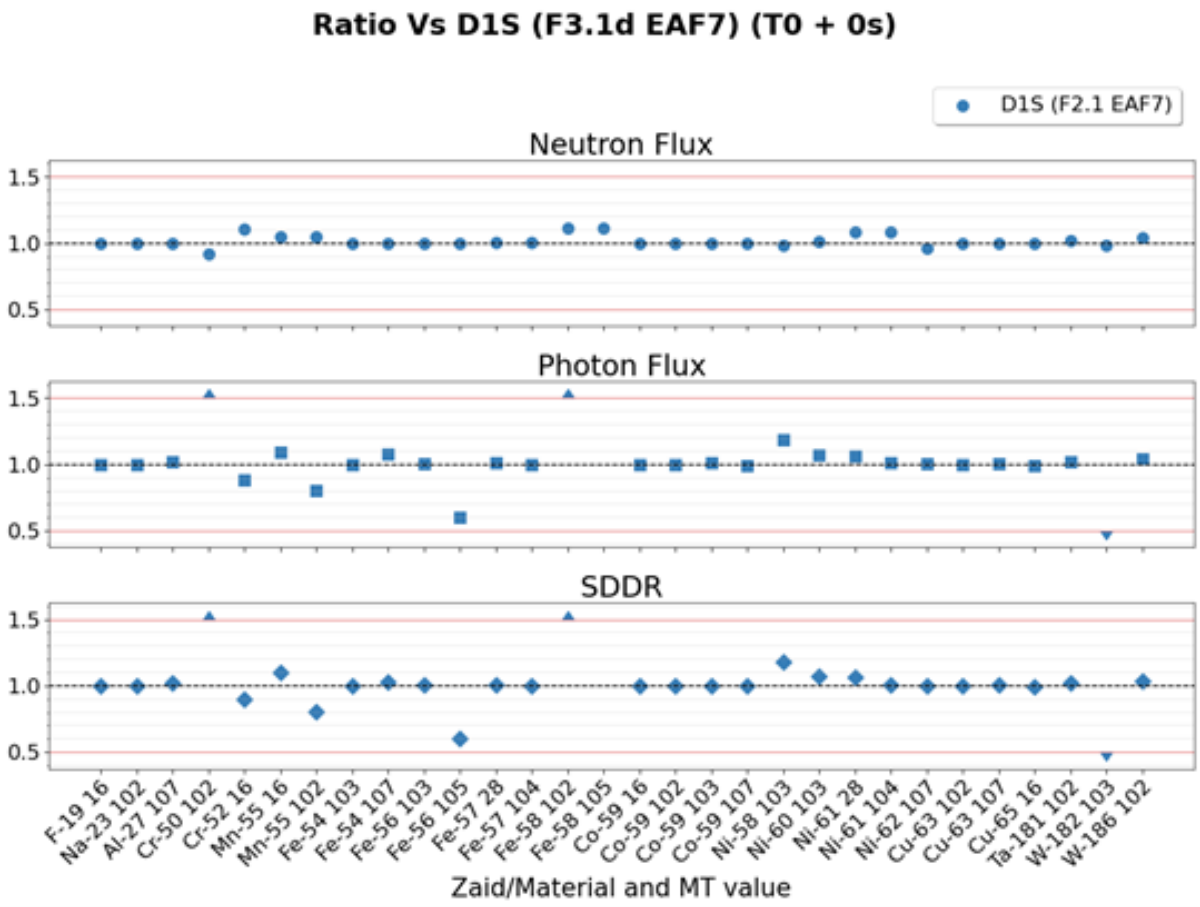
7.2.4 Discreet Experimental points



7.2.5 Grouped bars



7.2.6 Waves


Ratio Vs D1S (F3.1d EAF7) (T0 + 0s)
D1S (F2.1 EAF7)
Neutron Flux
Photon Flux
SDDR
1.5
1.0
0.5
F-19 16
Na-23 102
Al-27 107
Cr-50 102
Cr-52 16
Mn-55 16
Mn-55 102
Fe-54 103
Fe-54 107
Fe-56 103
Fe-56 105
Fe-57 28
Fe-57 104
Fe-58 102
Fe-58 105
Co-59 16
Co-59 102
Co-59 103
Co-59 107
Ni-58 103
Ni-60 103
Ni-61 28
Ni-61 104
Ni-62 107
Cu-63 102
Cu-63 107
Cu-65 16
Ta-181 102
W-182 103
W-186 102
Zaid/Material and MT value

CHAPTER

EIGHT

UTILITIES

During the development of JADE, many useful classes and methods were developed which could be used for small stand-alone tools, mostly operating on MCNP inputs.

A description of these *utilities*, accessible from the JADE main menu, is here provided.

The outputs (if generated) of these utilities can be found in specific subfolders of the `<JADE root>\Utilities` directory.

8.1 Print available libraries

`printlib`

This function allows to print to video all libraries (suffixes) that are available in xsdir file indicated in the main configuration file.

```
 Enter action: printlib
['03c', '01c', '02c', '03c', '04c', '05c', '06c', '21c', '30c', '31c',
'32c', '90c', '91c', '92c', '93c', '94c', '95c', '96c', '80c', '81c', '
82c', '83c', '84c', '85c', '86c', '70c', '71c', '72c', '73c', '74c', '6
2c', '66c', '60c', '50c', '42c', '53c', '24c', '50d', '30y', '50m', '71
h', '70h', '24h', '01g', '84p', '63p', '04p', '03p', '02p', '01p', '14p
', '12p', '03e', '01e']
 Enter action:
```

Fig. 1: Screenshot of the execution of the `printlib` command

8.2 Restore default configurations

`restore`

This function allows to restore the JADE configuration default settings. In other words, the content of the `<JADE root>\Configuration` directory is restored to "factory installation" and all user modifications to the configuration files are lost.

Note: When the the restoration is completed, the application will be terminated. The main configuration file ambient variable will need to be reconfigured before running another JADE instance.

See also:

Main Configuration

8.3 Translate an MCNP input

`trans`

This function allows to translate a material section of an MCNP input to a whatever nuclear data library available in the xsdir file.

The translation is carried out basically by the **convertZaid()** method of the **LibManager** class and by the **translate()** method of the **SubMaterial** class. The **convertZaid** method:

1. asks for a zaid (to translate) and for a library (to translate to);
2. checks if the library selected for the translation is available in the xsdir of the user;
3. **select the type of translation:**
    a. zaid not available in library: the default lib is used, no other changes applied;
    b. zaid available in library: the zaid is converted to the selected library, no other changes applied;
    c. the zaid is natural (i.e. it ends with 000).

For case c, at first, the selected library is checked for exact correspondence, i.e., it is checked if also in the selected library the zaid is expressed as natural. In this case, the behavior is identical to case b. If this is not true, the zaid needs to be expanded: all zaids of the same elements are returned with their atomic mass (m) and natural abundance (NA).

At this point, the **translate()** method completes the translation. No particular actions are required if there is no zaid expansion. In case of expansion, if the original natural zaid fraction is an atomic one (x_N^A), the new zaids deriving from the expansion will have as fraction their natural abundance (NA) multiplied for the original natural zaid fraction:

$$x_{zaid}^A = \text{NA}_{zaid} \cdot x_N^A$$

If, instead, the original natural zaid fraction is a mass one (x_N^M), the *equivalent mass* m_N of the natural zaid can be computed as:

$$m_N = \sum_{zaids} \text{NA}_{zaid} \cdot m_{zaid}$$

and then the mass fraction of each expanded new zaid (x_{zaid}^M) can be calculated as:

$$x_{\text{zaid}}^M = x_N^M \cdot (\text{NA}_{zaid} \cdot m_{zaid})/M_N$$

where $(\text{NA}_{zaid} \cdot m_{zaid})/M_N$ is basically the natural abundance in mass of the zaid.

The new input will be dumped in the `<JADE root>\Utilities\Translation` folder. The following scheme summarizes the JADE translation logic.

There are few format that can be used to request a translation:

Default (e.g. `31c`) Only one library is provided and the above described translation procedure is followed

ExactMode (e.g. `{99c: [1001, 1002], 31c: [8016]}`) This mode is used in the autoatic translation of D1S inputs where the mix between zaids to be used for transport and zaids to be used for activation result in additional complexity during the translation. The use of such mode is discourauged in the `trans` utility.

1to1Mode (e.g. `{99c: 98c, 31c: 21c}`) In case more than one library have been used in the original input the user can provide a dictionary which specifies for each original library (e.g. 99c and 31c) to which new library the zaids should be translated to (e.g. 31c and 21c).

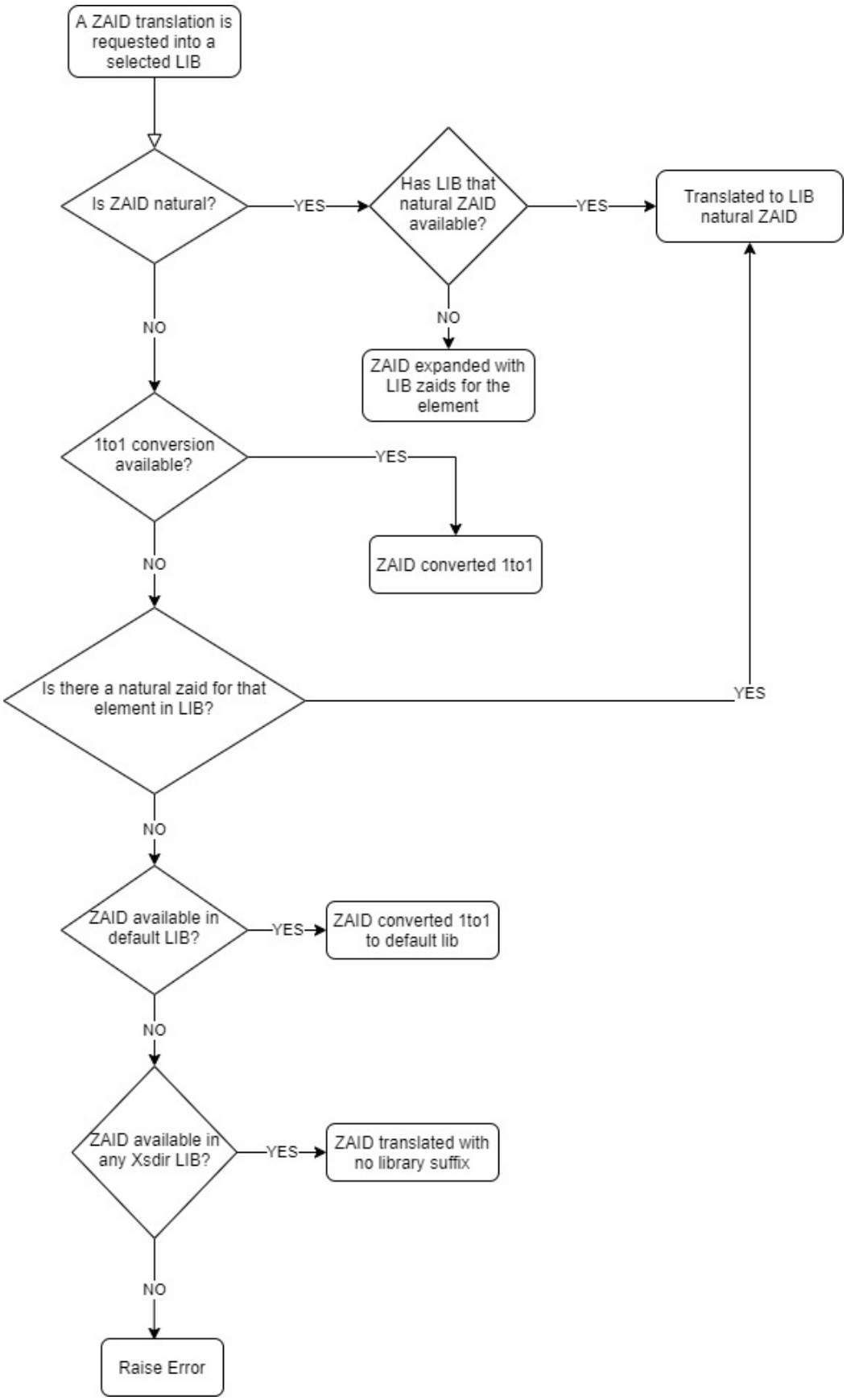


Fig. 2: Zaid translation logic

8.4 Print materials info

`printmat`

This function is used to print a summary of an MCNP input material section. The information is contained in two sheets of an Excel file dumped into the `<JADE root>\Utilities\Materials Infos` folder. The first sheet summarizes information at the single isotope level. Here both the atom and mass fraction for each zaid is reported divided by material and submaterial. It may happen that the original fraction appearing in the MCNP input is not normalized. JADE prints this fraction as it is and only the alternative fraction is normalized during its calculation.

Material	Submaterial	Element	Isotope	Atom Fraction	Mass Fraction
m1	1	H	H-1 [1001]	-1.90E-02	2.16E-02
			H-2 [1002]	-4.38E-06	2.49E-06
	2	C	C-12 [6012]	-1.96E-01	1.87E-02
			C-13 [6013]	-2.30E-03	2.02E-04
	3	N	N-14 [7014]	-2.51E-02	2.05E-03
			N-15 [7015]	-9.83E-05	7.50E-06
	4	O	O-16 [8016]	-3.77E-01	2.70E-02
			O-17 [8017]	-1.53E-04	1.03E-05
			O-18 [8018]	-8.72E-04	5.55E-05
	5	Mg	Mg-24 [12024]	-1.97E-02	9.40E-04
			Mg-25 [12025]	-2.60E-03	1.19E-04
			Mg-26 [12026]	-2.97E-03	1.31E-04
		Al	Al-27 [13027]	-9.26E-02	3.93E-03
		Si	Si-28 [14028]	-1.80E-01	7.38E-03
			Si-29 [14029]	-9.48E-03	3.75E-04
			Si-30 [14030]	-6.47E-03	2.47E-04
	6	S	S-32 [16032]	-1.35E-02	4.85E-04
			S-33 [16033]	-1.10E-04	3.83E-06
			S-34 [16034]	-6.37E-04	2.15E-05
			S-36 [16036]	-3.20E-06	1.02E-07
		Cu	Cu-63 [29063]	-3.46E-02	6.29E-04
			Cu-65 [29065]	-1.59E-02	2.81E-04
m2	1	Cu	Cu-63 [29063]	-6.85E-01	5.73E-02
			Cu-65 [29065]	-3.15E-01	2.56E-02
m3	1	Nb	Nb-93 [41093]	-7.01E-01	4.09E-02
	2	Sn	Sn-112 [50112]	-2.73E-03	1.32E-04
			Sn-114 [50114]	-1.89E-03	9.00E-05

Fig. 3: Extract of the isotope sheet. In the example, the material card was expressed in mass fraction and not normalized.

The second sheet summarizes information at the element level. Three fractions are here listed for each element: * the MCNP fraction of the element in the material; * the normalized fraction of the element in the submaterial; * the normalized fraction of the element in the material.

Depending on the orginal MCNP input, these three fraction need to be interpreted as either *mass* or *atom* fraction.

Material	Submaterial	Element	Fraction	Sub-Material Fraction	Material Fraction
m1	1	H	2.16E-02	1.00E+00	2.57E-01
	2	C	1.89E-02	1.00E+00	2.25E-01
	3	N	2.06E-03	1.00E+00	2.45E-02
	4	O	2.71E-02	1.00E+00	3.21E-01
	5	Al	3.93E-03	3.00E-01	4.67E-02
		Mg	1.19E-03	9.07E-02	1.41E-02
		Si	8.00E-03	6.10E-01	9.50E-02
	6	Cu	9.10E-04	6.41E-01	1.08E-02
		S	5.10E-04	3.59E-01	6.06E-03
m10	1	Cu	7.67E-02	1.00E+00	9.56E-01
	2	Sn	3.57E-03	1.00E+00	4.45E-02
m11	1	B	4.38E-06	1.00E+00	5.13E-05
	2	C	7.09E-05	1.00E+00	8.31E-04
	3	N	2.36E-04	1.00E+00	2.77E-03
	4	Al	5.26E-04	3.91E-01	6.17E-03
		O	5.90E-06	4.38E-03	6.92E-05

Fig. 4: Extract of the element sheet. In the example, the material card was expressed in mass fraction and not normalized.

8.5 Generate material mixture

`generate`

This function is used to generate a material mixture starting from two or more materials contained in a single MCNP input. The user will be asked for:

- absolute path to the MCNP input;
- if the zaids need to have a mass or atom fraction;
- material names (e.g. m1) to be used in the mixture;
- percentages to be used in the mixture for each material;
- nuclear data library to use for the new material mixture.

Each material will be transformed in a submaterial of the newly generated mixture retaining its header if present. The new material will be dumped in the `<JADE root>\Utilities\Generated Materials` folder.

8.6 Switch material fractions

`switch`

This function can be used to switch an MCNP input from having atom fractions to mass fractions and viceversa. The new input will be dumped in the `<JADE root>\Utilities\Fraction switch` folder.

8.7 Change .ace libraries suffix

New in version v1.3.0: `acelib`

This function asks for a directory absolute path and for a new library suffix (e.g. `98c`). All .ace files contained in the folder will have their original suffix changed to the new one. This function operates in a non-destructive way, that is, the switch is not implemented on the original file but on copies of them instead.

8.8 Produce D1S-UNED reaction files

New in version v1.3.0: `react`

This function, given a D1S input file, produce a correspondent reaction file where all possible reactions that can originate from the input materials are listed. The complete list of available reactions for each D1S activation library is provided (and may be modified) in the `<JADE_root>\Configuration\Activation.xlsx` file.

The generated reaction files are dumped in the `<JADE root>\Utilities\Reactions` folder

CHAPTER

NINE

TIPS & TRICKS

This section reunites a series of tips and tricks that can be used to *unlock* JADE additional capabilities.

9.1 External Run of a benchmark

It may be useful for particularly computational-intensive benchmark to be run on a separate hardware (e.g. a server) with respect to the one used for JADE. This can be achieved quite easily with the following steps:

1. set the `OnlyInput` option in the `<JADE root>\Configuration\Conf.xlsx` file to `True` for the benchmark that needs to be run externally. This will generate the MCNP input file of the benchmark that can be found in `<JADE root>\Tests\MCNP simulation\<lib suffix>\<Benchmark name>` without running it;
2. copy the generated input file into the hardware selected for the run and start the MCNP simulation. The only requirement is to use the MCNP keyword `name=` when launching the simulation in order to obtain consistently named outputs;
3. once the simulation is completed, copy all MCNP outputs to the same `<JADE root>\Tests\MCNP simulation\<lib suffix>\<Benchmark name>` folder;
4. normally run the post-processing.

9.2 Change the plots fontsizes

Font size in plots is hardcoded in JADE. Nevertheless to change these value globally for all plots it is quite easy since they are all defined at the beginning of the `<JADE root>\Code\plotter.py` file trough the matplotlib.pyplot.rc attribute:

```
import matplotlib.pyplot as plt
# ============================================================================
#                   Specify parameters for plots
# ============================================================================
SMALL_SIZE = 22
MEDIUM_SIZE = 26
BIGGER_SIZE = 30

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=BIGGER_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
```

(continues on next page)

(continued from previous page)

```
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
plt.rc('lines', markersize=12)           # Marker default size
```

CHAPTER

TEN

INSERT CUSTOM BENCHMARKS

This section of the guide describes how to add custom benchmarks to the JADE suite. The procedures necessary to implement new computational and experimental benchmarks are different and are described respectively in *Insert Custom Computational Benchmark* and *Insert Custom Experimental Benchmark*.

10.1 Insert Custom Computational Benchmark

Implementing a new computational benchmark is relatively easy and, theoretically, no additional code is required. The procedure is composed by the following steps:

1. Once the benchmark input has been finalized, save it as `<JADE_root>\Benchmark inputs\<name>.i`.
2. Add the benchmark to the main configuration file in the computational sheet. See *Computational benchmarks* for additional information on this.
3. [OPTIONAL] if external weight windows (WW) are used, the WW file must be named *wwinp* and inserted in `<JADE_root>\Benchmark inputs\VRT\<name>\`.
4. Create a custom post-processing configuration file as described in *Benchmark post-processing configuration* and save it in `<JADE_root>\Configuration\Benchmarks Configuration\<name>.xlsx`

Note: The benchmark input should not contain any STOP paramaters or NPS card (this is regulated by the main configuration file).

Note: It is recommended to provide a comment card (FC) for each tally. These comments are considered the extended tally names and are used during post-processing.

Warning: benchmark input file name cannot end with 'o' or 'm'.

10.2 Insert Custom Experimental Benchmark

Inserting a custom experimental benchmark is slightly more complex, but a significant higher order of customization is guaranteed. Steps 1) and 2) of the computational benchmarks procedure still need to be followed but then some additional coding needs to be performed, specifically, a new child of the *ExperimentalOutput* class needs to be defined inside `<JADE_root>\Code\expoutput.py`. In order to do that, at least the three abstract methods `_processMCNPdata()`, `_pp_excel_comparison()` and `_build_atlas()` need to be implemented in the new class. Once this has been done, a few other adjustments need to be done to the code.

10.2.1 Call the right Output class

In `<JADE_root>\Code\postprocess.py`, the function `_get_output()` controls the creation of the benchmark object during post-processing depending on the benchmark. Here an *elif* statement needs to be added to ensure that the newly created custom class is called when generating the output for the custom added experimental benchmark. Here is an example of how the FNG benchmark was added:

```
...

elif testname == 'FNG':
    if action == 'compare':
        out = expo.FNGOutput(lib, testname, session, multiplerun=True)
    elif action == 'pp':
        print(exp_pp_message)
        return False

...
```

The user should just substitute `FNG` with the name of the benchmark input and `FNGOutput` with the newly created class. Attention should be paid also to the `multiplerun` keyword, set True if the benchmark is actually composed by more than one input (i.e. multiple MCNP/D1S runs).

10.2.2 Additional actions for multi-run benchmarks

> **Warning:** these next actions need to be performed **only** if the benchmark is composed by more than one input.

In `<JADE_root>\Code\status.py` the name of the benchmark input needs to be added to the MULTI_TEST

```
MULTI_TEST = ['Sphere', 'Oktavian', 'SphereSDDR', 'FNG']
```

In `<JADE_root>\Code\computational.py` the function `executeBenchmarksRoutines` is responsible for the generation and run of the benchmarks during a JADE session. The modification here is to be performed in the part that is responsible for choosing the Test object to be used depending on the benchmark. Here is the code snippet of interest:

```
...

# Handle special cases
if testname == 'Sphere Leakage Test':
    test = testrun.SphereTest(*args)

elif testname == 'Sphere SDDR':
```

(continues on next page)

(continued from previous page)

```
    test = testrun.SphereTestSDDR(*args)

elif fname == 'Oktavian':
    test = testrun.MultipleTest(*args)

elif fname == 'FNG':
    test = testrun.MultipleTest(*args, TestOb=testrun.FNGTest)

else:
    test = testrun.Test(*args)

...
```

The default option is to simply create a `Test` object. Clearly, if a children was defined specifically for the new experimental benchmark, an option would need to be added here. If the benchmark is a multirun one, an additional *elif* statement needs to be added similarly to what has been done for the FNG benchmark.

See also:

see also *testrun module* for a better description of the Test object and his children

CHAPTER

ELEVEN

MODIFY DOCUMENTATION

This documentation is written with Sphynx using a template provided by Read The Docs. Before attempting to modify the documentation, the developer should familiarize with these tools and with the RST language that is used to actually write the doc.

Inside `<JADE root>\Code\docs` are located the *source* and *build* directories of the documentation. To apply a modification, the user must simply modify/add one or more files in the *source* tree and in the *docs* folder execute from terminal the `make html` command.

CHAPTER

TWELVE

GENERAL OOP SCHEME

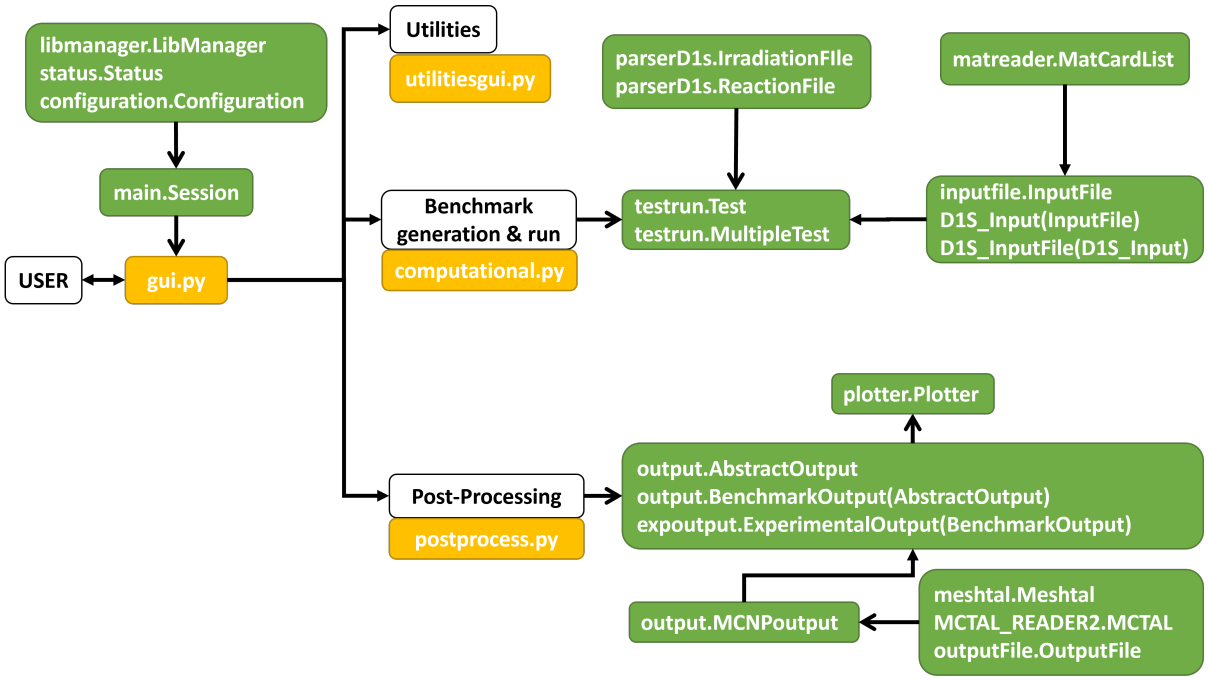


Fig. 1: General OOP scheme for the JADE code

All user interactions that happen through the command console are handled by the `gui.py module`. When JADE is started, a *Session* object is intialized which is a container for a series of information and tools that many parts of the code may need to access. In particular it contains:

Paths through *Session* it is possible to recover many paths to the different folders that constitutes the JADE tree (see also *Folder Structure*).

Status the *Status* object has informations on which libraries have been assessed or post-processed.

Configuration the *Configuration* object is the one that handles the parsing of the *Main Configuration* file.

Library Manager the *LibManager* is responsible for all operations related to nuclear data libraries. These include for instance checking the availability of a library, or handling the translation of a single isotope.

Generally speaking, the user can request three types of thing to the gui:

- use one of the utilities, that will trigger the call of the `utilitiesgui.py`;
- run the benchmark suite on a library through the `computational.py` module;
- perform the pst-processing on one or more libraries calling the `postprocess.py` module.

12.1 Utilities

The `utilitiesgui.py` simply contains a number of functions associated to the different utilities available in JADE.

See also:

Utilities

12.2 Benchmarks generation and run

Operations for the benchmarks generation and run are handled by the `computational.py` module. In JADE the object representing a benchmark is the *Test* (or *MultipleTest* in case the benchmark is composed by more than one run). This object is responsible for the creation of the MCNP input and for its run. A vital attribute of the benchmark is its associated *InputFile* or one of its children. In case the benchmark is run with d1s code, an *IrradiationFile* and a *ReactionFile* are also associated with the test. A fundamental attribute of inputs is the *MatCardsList* which handles all operations related to the materials (including library translations).

12.3 Post-processing

Operations for the post-processing of benchmark run results are handled by the `postprocessing.py` module. All objects representing outputs of a benchmark run must be a child of the abstract class *AbstractOutput*. These classes always include an *MCNPOutput* which collect the results coming from the parsers of the different MCNP outputs.

CHAPTER

THIRTEEN

JADE SESSION

13.1 main module

13.1.1 Session

class main.**Session**

Bases: object

This object represent a JADE session. All "environment" variables are initialized

Returns

Return type None.

check_active_tests(*action*, *exp=False*)

Check the configuration file for active benchmarks to perform or post-process

Parameters

- **session** (Session) – JADE session
- **action** (*str*) – either 'Post-Processing' or 'Run' (as in Configuration file)
- **exp** (*boolean*) – if True checks the experimental benchmarks. Default is False

Returns **to_perform** – list of active test names

Return type list

initialize()

Initialize JADE session:

- folders structure is created if absent
- Configuration file is read and correspondent object is created
- Libmanager is created
- Logfile is created

Returns

Return type None.

restore_default_settings()

Reset the configuration files to installation default. The session is re-initialized.

Returns

Return type None.

13.2 libmanager module

13.2.1 LibManager

class libmanager.**LibManager**(*xsdir_file*, *defaultlib='81c'*, *activationfile=None*)
Bases: object

Object dealing with all complex operations that involves nuclear data

Parameters

- **xsdir_file** (*str or path*) – path to the MCNP xsdir reference file.
- **defaultlib** (*str, optional*) – lib suffix to be used as default in translation operations. The default is '81c'.
- **activationfile** (*str or path, optional*) – path to the ocnfig file containing the reactions data for activation libraries. The default is None.

Returns

Return type None.

check4zaid(*zaid*)
Check which libraries are available for the selected zaid and return it

Parameters **zaid** (*str*) – zaid string (e.g. 1001).

Returns **libraries** – list of libraries available for the zaid.

Return type list

convertZaid(*zaid*, *lib*)
This methods will convert a zaid into the requested library

modes:

- 1to1: there is one to one correspondence for the zaid
- natural: the zaids will be expanded using the natural abundance
- absent: the zaid is not available in the library, a default one will be used

Parameters

- **zaid** (*str*) – zaid name (ex. 1001).
- **lib** (*str*) – library suffix (ex. 21c).

Raises **ValueError** – if the library is not available in the xsdir file or if there is no valid translation for the zaid.

Returns **translation** – {zaidname:(lib,nat_abundance,Atomic mass)}.

Return type dic

get_libzaids(*lib*)
Given a library, returns all zaids available

Parameters **lib** (*str*) – suffix of the library.

Returns **zaids** – list of zaid names available in the library.

Return type list

get_reactions(*lib*, *parent*)
get the reactions available for a specific zaid and parent nuclide

Parameters

- **lib** (*str*) – library suffix as in sheet name of the activation file.
- **parent** (*str*) – zaid number of the parent (e.g. 1001).

Returns **reactions** – contains tuple of (MT, daughter).

Return type list

get_zaid_mass(*zaid*)
Get the atomic mass of one zaid

Parameters **zaid** (matreader.Zaid) – Zaid to examinate.

Returns **m** – atomic mass.

Return type float

get_zaidname(*zaid*)
Given a zaid, its element name and formula are returned. E.g., hydrogen, H1

Parameters **zaid** (*str*) – zaid number (e.g. 1001 for H1).

Returns

- **name** (*str*) – element name (e.g. hydrogen).
- **formula** (*str*) – isotope name (e.g. H1).

get_zaidnum(*zaidformula*)
Given a zaid formula return the correct number

Parameters **zaidformula** (*str*) – name of the zaid, e.g., H1.

Returns **zaidnum** – number of the zaid ZZZAA

Return type str

select_lib()
Prompt an library input selection with Xsdir availabilty check

Returns **lib** – Library to assess.

Return type str

13.3 state module

13.3.1 Status

class status.**Status**(*session*)
Bases: object

Stores the state of the JADE runs and post-processing.

Parameters **session** (main.Session) – JADE session.

Returns

Return type None.

check_lib_run(*lib*, *session*, *config_option='Run'*, *exp=False*)

Check if a library has been run. To be considered run a meshtally or meshtal have to be produced. Only active benchmarks (specified in the configuration file) are checked.

Parameters

- **lib** (*str*) – Library to check.
- **session** (Session) – Jade Session.
- **config_option** (*str*) – Specifies the configuration option onto which the check for tests "to perform" are registered.
- **exp** (*boolean*) – if True checks the experimental benchmarks. Default is False

Returns **test_runned** – True if all benchmark have been run for the library.

Return type Bool

check_override_pp(*session*, *exp=False*)

Asks for the library/ies to post-process and checks which tests have already been performed and would be overidden according to the configuration file. This can be used also to check if in a post processing of multiple libraries wich single library post processing is missing.

Parameters **session** (Session) – JADE session

Returns

- **lib** (*str*) – Library/ies to post process
- **ans** (*Boolean*) – True if the PP can begin (Possible override has been accepted).
- **exp** (*boolean*) – if True checks the experimental benchmarks. Default is False

check_override_run(*lib*, *session*, *exp=False*)

Check status of the requested run. If overridden is required permission is requested to the user

Parameters

- **lib** (*str*) – Library to run.
- **session** (Session) – Jade Session.
- **exp** (*False*) – if True checks the experimental benchmarks. Default is False

Returns **ans** – if True proceed with the run, if False, abort.

Return type Boolean

check_pp_single(*lib*, *session*, *tree='single'*, *exp=False*)

Check if the post processing of a single library or a comparison has been already done. To consider it done, all benchmarks must have been post-processed

Parameters

- **lib** (*string*) – library/ies to check.
- **session** (Session) – JADE session.
- **tree** (*string, optional*) – Either 'single' to check in the single pp tree or 'comparison' to check into the comparison one. The default is 'single'.
- **exp** (*boolean*) – if True checks the experimental benchmarks. Default is False

Returns True if PP has been done.

Return type Boolean

static check_test_run(*files*)
Check if a test has been run

Parameters **files** (*list*) – file names inside test folder.

Returns **flag_test_run** – True if test has been run.

Return type Bool

get_path(*tree*, *itinerary*)
Get the resulting path of an itinery on one tree

Parameters

- **tree** (*str*) – Either 'comparison', 'single', or 'run'.
- **itinerary** (*list*) – list of strings representing the step to take inside the tree.

Raises **KeyError** – if var tree is not among possible strings.

Returns **cp** – path to final step.

Return type str/path

get_unfinished_zaids(*lib*)
Identify zaids to run for rerun or continuation purposes

Parameters **lib** (*str*) – library to check.

Returns **unfinished** – zaids/typical materials not run.

Return type list

update_pp_status()
Read/Update the post processing tree status. All files produced by post processing registered

Returns

- **comparison_tree** (*dic*) – Dictionary registering all test post processed for each comparison of libraries.
- **single_tree** (*dic*) – Dictionary registering all test post processed performed for single libraries.

update_run_status()
Read/Update the run tree status. All files produced by runs are registered

Returns **libraries** – dictionary of dictionaries representing the run tree

Return type dic

13.4 configuration module

13.4.1 Configuration

class configuration.**Configuration**(*conf_file*)
Bases: object

Parser of the main configuration file

Parameters **conf_file** (*path like object*) – path to configuration file.

Returns

Return type None.

get_lib_name(*suffix*)
Get the name of the library from its suffix. If a name was not specified in the configuration file the same suffix is returned

Parameters **suffix** (*str*) – e.g. 21c .

Returns Name of the library.

Return type str

read_settings()
Parse the configuration file

Returns

Return type None.

CHAPTER

FOURTEEN

INPUT GENERATION

In this section the most useful classes that can be used during benchmark input preparation are described.

14.1 matreader module

14.1.1 MatCardsList

class matreader.**MatCardsList**(*materials*)

Bases: collections.abc.Sequence

Object representing the list of materials included in an MCNP input. This class is a child of the Sequence base class.

Parameters **materials** (*list[Material]*) – list of materials.

Returns

Return type None.

classmethod from_input(*inputfile*)

This method use the numjuggler parser to help identify the mcards in the input. Then the mcards are parsed using the classes defined in this module

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **inputfile** (inputfile.InputFile) – MCNP input file containing the material section.

Returns new material card list generated.

Return type *MatCardsList*

get_info(*lib_manager*, *zaids=False*, *complete=False*)

Get the material informations in terms of fraction and composition of the material card

Parameters

- **lib_manager** (libmanager.LibManager) – To handle element name recovering.
- **zaids** (*bool, optional*) – Consider or not the zaid level. The default is False.
- **complete** (*bool, optional*) – If True both the atom and mass fraction are given in the raw table. The default is False.

Returns

- **df** (*pd.DataFrame*) – Raw infos on the fractions.

- **df_elem** (*pd.DataFrame*) – processed info for the element: normalized fraction added both for material and submaterial.

to_text()
return text of the material cards in order

Returns material card list MCNP formatted text.

Return type str

translate(*newlib*, *lib_manager*)
This method allows to translate the material cards to another library. The zaid are collapsed again to get the new elements

Parameters

- **newlib** (*dict or str*) – There are a few ways that newlib can be provided:

  1) str (e.g. 31c), the new library to translate to will be the one indicated;

  2) dic (e.g. {'98c' : '99c', '31c: 32c'}), the new library is determined based on the old library of the zaid

  3) dic (e.g. {'98c': [list of zaids], '31c': [list of zaids]}), the new library to be used is explicitly stated depending on the zaidnum.

- **lib_manager** (libmanager.LibManager) – Library manager for the conversion.

Returns

Return type None.

update_info(*lib_manager*)
This methods allows to update the in-line comments for every zaids containing additional information

Parameters **lib_manager** (*libmanager.Libmanager*) – Library manager for the conversion.

Returns

Return type None.

14.1.2 Material

class matreader.**Material**(*zaids*, *elem*, *name*, *submaterials=None*, *mx_cards=[]*, *header=None*)
Bases: object

Object representing an MCNP material

Parameters

- **zaids** (*list[zaids]*) – zaids composing the material.
- **elem** (*list[elem]*) – elements composing the material.
- **name** (*str*) – name of the material (e.g. m1).
- **submaterials** (*list[Submaterials], optional*) – list of submaterials composing the material. The default is None.
- **mx_cards** (*list, optional*) – list of mx_cards in the material if present. The default is [].
- **header** (*str, optional*) – material header. The default is None.

Returns

Return type None.

add_mx(*mx_cards*)
Add a list of mx_cards to the material

classmethod from_text(*text*)
Create a material from MCNP formatted text

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **text** (*list[str]*) – MCNP formatted text.

Returns material object created.

Return type *matreader.Material*

get_tot_fraction()
Returns the total material fraction

switch_fraction(*ftype*, *lib_manager*, *inplace=True*)
Switch between atom or mass fraction for the material card. If the material is already switched the command is ignored.

Parameters

- **ftype** (*str*) – Either 'mass' or 'atom' to chose the type of switch.
- **lib_manager** (libmanager.LibManager) – Handles zaid data.
- **inplace** (*bool*) – if True the densities of the isotopes are changed inplace, otherwise a copy of the material is provided. DEFAULT is True

Raises **KeyError** – if ftype is not either 'atom' or 'mass'.

Returns **submaterials** – list of the submaterials where fraction have been switched

Return type list

to_text()
Write the material to MCNP formatted text

Returns MCNP formatte text representing the material.

Return type str

translate(*newlib*, *lib_manager*, *update=True*)
This method allows to translate all submaterials to another library

Parameters

- **newlib** (*dict or str*) – There are a few ways that newlib can be provided:

  1) str (e.g. 31c), the new library to translate to will be the one indicated;

  2) dic (e.g. {'98c' : '99c', '31c: 32c'}), the new library is determined based on the old library of the zaid

  3) dic (e.g. {'98c': [list of zaids], '31c': [list of zaids]}), the new library to be used is explicitly stated depending on the zaidnum.
- **lib_manager** (libmanager.LibManager) – object handling all libraries operations.
- **update** (*bool, optional*) – if True, material infos are updated. The default is True.

Returns

Return type None.

update_info(*lib_manager*)
This methods allows to update the in-line comments for every zaids containing additional information

lib_manager: (LibManager) Library manager for the conversion

14.1.3 Submaterial

class matreader.**SubMaterial**(*name*, *zaidList*, *elemList=None*, *header=None*, *additional_keys=[]*)
Bases: object

Generate a SubMaterial Object starting from a list of Zaid and eventually Elements list

Parameters

- **name** (*str*) – if the first submaterial, the name is the name of the material (e.g. m1).
- **zaidList** (*list[Zaid]*) – list of zaids composing the submaterial.
- **elemList** (*list[Element], optional*) – list of elements composing the submaterial. The default is None.
- **header** (*str, optional*) – Header of the submaterial. The default is None.
- **additional_keys** (*list[str], optional*) – list of additional keywords in the submaterial. The default is [].

Returns

Return type None.

collapse_zaids()
Organize zaids into their elements and collapse mutiple istances

Returns

Return type None.

classmethod from_text(*text*)
Generate a submaterial from MCNP input text

Parameters

- **cls** (SubMaterial) – submaterial to be generated.
- **text** (*list[str]*) – Original text of the MCNP input.

Returns generated submaterial.

Return type *SubMaterial*

get_info(*lib_manager*)
Returns DataFrame containing the different fractions of the elements and zaids

Parameters **lib_manager** (libmanager.LibManager) – Library manager for the conversion.

Returns

- **df_el** (*pd.DataFrame*) – table of information of the submaterial on an elemental level.
- **df_zaids** (*pd.DataFrame*) – table of information of the submaterial on a zaid level.

scale_fractions(*norm_factor*)
Scale the zaids fractions using a normalizing factor

Parameters **norm_factor** (*float*) – scaling factor.

Returns

Return type None.

to_text()
Write to text in MNCP format the submaterial

Returns formatted submaterial text.

Return type str

translate(*newlib*, *lib_manager*)
This method implements the translation logic of JADE. All zaids are translated accordingly to the newlib specified.

Parameters

- **newlib** (*dict or str*) – There are a few ways that newlib can be provided:

  1) str (e.g. 31c), the new library to translate to will be the one indicated;

  2) dic (e.g. {'98c' : '99c', '31c: 32c'}), the new library is determined based on the old library of the zaid

  3) dic (e.g. {'98c': [list of zaids], '31c': [list of zaids]}), the new library to be used is explicitly stated depending on the zaidnum.

- **lib_manager** (LibManager) – Object handling libraries operation.

Returns

Return type None.

update_info(*lib_manager*)
This methods allows to update the in-line comments for every zaids containing additional information

Parameters **lib_manager** (libmanager.LibManager) – Library manager for the conversion.

Returns

Return type None.

14.1.4 Element

class matreader.**Element**(*zaidList*)
Bases: object

Generate an Element object starting from a list of zaids. It will collapse multiple instance of a zaid into a single one

Parameters **zaidList** (*list*) – list of zaids constituing the element.

Returns

Return type None.

get_fraction()
Get the sum of the fraction of the zaids composing the element

Returns **fraction** – element fraction.

Return type float

update_zaidinfo(*libmanager*)
Update zaids infos through a libmanager. Info are the formula name and the abundance in the material.

Parameters **libmanager** (libmanager.LibManager) – libmanager handling the libraries operations.

Returns

Return type None.

14.1.5 Zaid

class matreader.**Zaid**(*fraction*, *element*, *isotope*, *library*, *ab=''*, *fullname=''*)
Bases: object

Object representing a Zaid

Parameters

- **fraction** (*str/float*) – fraction of the zaid.
- **element** (*str*) – element part of the zaid (AA).
- **isotope** (*str*) – isotope part of the zaid (ZZZ).
- **library** (*str*) – library suffix (e.g. 99c).
- **ab** (*str, optional*) – abundance of the zaid in the material. The default is ‘’.
- **fullname** (*str, optional*) – formula name (e.g. H1). The default is ‘’.

Returns

Return type None.

classmethod from_string(*string*)
Generate a zaid object from an MCNP string

Parameters

- **cls** (matreader.Zaid) – zaid to be created.
- **string** (*str*) – original MCNP string.

Returns created zaid.

Return type *Zaid*

get_fullname(*libmanager*)
Get the formula name of the zaid (e.g. H1)

Parameters **libmanager** (libmanager.LibManager) – libmanager handling the libraries operations.

Returns **formula** – zaid formula name.

Return type str

to_text()

Get the zaid string ready for MCNP material card

Returns zaid string.

Return type str

14.2 testrun module

14.2.1 Test

class testrun.**Test**(*inp*, *lib*, *config*, *log*, *VRTpath*, *confpath*)
Bases: object

Class representing a general test. This class will have to be extended for specific tests.

Parameters

- **inp** (*str*) – path to inputfile blueprint.
- **lib** (*str*) – library suffix to use (e.g. 31c).
- **config** (*pd.DataFrame (single row)*) – configuration options for the test.
- **log** (*Log*) – Jade log file access.
- **VRTpath** (*path like object*) – path to the variance reduction folder.
- **confpath** (*path like object*) – path to the test configuration folder.

Raises **ValueError** – if the code specified in config is not admissible.

Returns

Return type None.

custom_inp_modifications()
Perform additional operation on the input before generation. In this parent object actually does nothing

Returns

Return type None.

generate_test(*lib_directory*, *libmanager*, *MCNP_dir=None*)
Generate the test input files

Parameters

- **lib_directory** (*path or string*) – Path to lib benchmarks input folders.
- **libmanager** (libmanager.LibManager) – Manager dealing with libraries operations.
- **MCNPdir** (*str or path*) – allows to ovewrite the MCNP dir if needed. The default is None

Returns

Return type None.

run(*cpu=1*, *timeout=None*)
run the input

Parameters

- **cpu** (*int, optional*) – number of CPU to be used. The default is 1.
- **timeout** (*int, optional*) – number of seconds after the simulation should be killed. The default is None.

Returns

Return type None.

14.2.2 MultipleTest

class testrun.**MultipleTest**(*inpsfolder*, *lib*, *config*, *log*, *VRTpath*, *confpath*, *TestOb=<class 'testrun.Test'>*)

Bases: object

A collection of Tests

Parameters

- **inpsfolder** (*path-like object*) – folder that contains all inputs of the tests.
- **lib** (*str*) – library suffix to use (e.g. 31c).
- **config** (*pd.DataFrame (single row)*) – configuration options for the test.
- **log** (*Log*) – Jade log file access.
- **VRTpath** (*path like object*) – path to the variance reduction folder.
- **confpath** (*path like object*) – path to the test configuration folder.
- **TestOb** (testrun.Test, *optional*) – type of test object to be used. The default is Test.

Returns

Return type None.

generate_test(*lib_directory*, *libmanager*)

Generate all the tests of the collection

Parameters

- **lib_directory** (*path-like*) – output directory where to generate the tests.
- **libmanager** (libmanager.LibManager) – object handling libraries operations.

Returns

Return type None.

run(*cpu=1*, *timeout=None*)

Run all the tests

Parameters

- **cpu** (*int, optional*) – number of CPU to be used. The default is 1.
- **timeout** (*int, optional*) – number of seconds after each simulation is killed. The default is None.

Returns

Return type None.

14.3 inputfile module

14.3.1 InputFile

class inputfile.**InputFile**(*cards*, *matlist*, *name=None*)
Bases: object

Object representing an MCNP input file

Parameters

- **cards** (*dic*) – contains the cells, surfaces, settings and title cards.
- **matlist** (*matreader.MatCardList*) – material list in the input.
- **name** (*str, optional*) – name associated with the file. The default is None.

Returns

Return type None.

add_edits(*edits_file*)
Add weight windows and source bias resulted from ADVANTG analysis

Parameters **edits_file** (*path like object*) – file containing the edits.

Returns

Return type None.

add_stopCard(*nps*, *ctme*, *precision*)
Add STOP card

Parameters

- **nps** (*int*) – number of particles to simulate.
- **ctme** (*int*) – copmuter time.
- **precision** (*(str, float)*) – tally number, precision.

Returns

Return type None.

addlines2card(*lines*, *blockID*, *cardID*, *offset_all=True*)
Append some lines to one of the cards in the input

Parameters

- **lines** (*list or str*) – list of lines to be appended to the card. If a string is given, this is wrapped
- **blockID** (*str*) – either 'cell', 'surf', 'settings' or 'title'.
- **cardID** (*str or int*) – card ID (e.g. FC22).
- **offset_all** (*bool, optional*) – if True all the lines are off-setted with whitespace. If False the first line will have no offset (new card)

Returns

- *bool*
- *if lines have been successfully added return True, otherwise False.*

change_density(*density*, *cellidx=1*)
Change the density of the sphere according to the selected zaid

Parameters

- **density** (*str/float*) – density to apply.
- **cellidx** (*int, optional*) – cell index where to modify the density. The default is 1.

Returns

Return type None.

classmethod from_text(*inputfile*)
This method use the numjuggler parser to help identify the mcards in the input which will usually undergo special treatments in the input creation

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **inputfile** (*path like object*) – path to the MCNP input file.

Returns

Return type None.

get_card_byID(*blockID*, *cardID*)
Get a card of the input based on its ID

Parameters

- **blockID** (*str*) – either 'cell', 'surf', 'settings' or 'title'.
- **cardID** (*str or int*) – card ID (e.g. FC22).

Raises **ValueError** – if blockID is not allowed.

Returns **card** – Selected card. If the card is not found, None is returned

Return type numjuggler.parser.Card

static mcnp_wrap(*text*, *maxchars=80*, *whitespace=' '*, *offset_all=True*)
Wrap the text of a card in MCNP style

Parameters

- **text** (*str*) – text of the card to be formatted.
- **maxchars** (*int, optional*) – max limit of chars in one line. The default is 80.
- **whitespace** (*str, optional*) – whitespace to be put in front of newlines. The default is ' '.
- **offset_all** (*bool, optional*) – if True all the lines are off-setted with whitespace. If False the first line will have no offset (new card)

Returns list of correctly wrapped line of a card expressing the initial text.

Return type list

translate(*newlib*, *libmanager*)
Translate the input to another library

Parameters

- **newlib** (*str*) – suffix of the new lib to translate to.

- **libmanager** (libmanager.LibManager) – Library manager for the conversion.

Returns

Return type None.

update_zaidinfo(*lib_manager*)
This methods allows to update the in-line comments for every zaids containing additional information

Parameters **lib_manager** (libmanager.LibManager) – Library manager for the conversion.

Returns

Return type None.

write(*out*)
Write the input to a file

Parameters **out** (*str*) – path to the output file.

Returns

Return type None.

14.3.2 D1S_Input

class inputfile.**D1S_Input**(*cards*, *matlist*, *name=None*)
Bases: *inputfile.InputFile*

Object representing an MCNP input file

Parameters

- **cards** (*dic*) – contains the cells, surfaces, settings and title cards.
- **matlist** (*matreader.MatCardList*) – material list in the input.
- **name** (*str, optional*) – name associated with the file. The default is None.

Returns

Return type None.

add_PIKMT_card(*parent_list*)
Add a PIKMT card to the input file

Parameters **parent_list** (*list*) – list of parent zaids.

Returns

Return type None.

add_track_contribution(*tallyID*, *zaids*, *who='parent'*)
Given a list of zaid add the FU bin in the requested tallies in order to collect the contribution of them to the tally.

Parameters

- **tallyID** (*str*) – ID of the tally onto which to operate (e.g. F4:p).
- **zaids** (*str*) – zaid number of the parent/daughter (e.g. 1001).
- **who** (*str, optional*) – either 'parent' or 'daughter' specifies the types of zaids to be tracked. The default is 'parent'.

Raises **ValueError** – check for admissible who parameter.

Returns return True if lines were added correctly

Return type bool

get_reaction_file(*libmanager*, *lib*)
Collect all the possible reactions that are allowed amnong all the materials in the input

Parameters

- **libmanager** (LibManager) – Object handling all cross-sections related operations.
- **lib** (*str*) – library suffix to be used.

Returns Object representing the react file for D1S.

Return type *ReactionFile*

translate(*newlib*, *libmanager*, *original_irradfile=None*, *original_reacfile=None*)
Translate the input to another library. This methods ovverride the parent one since often two different libraries must be considered in a D1S input: a tranport and an activation library

Parameters

- **newlib** (*str*) – suffix of the new lib to translate to.
- **libmanager** (LibManager) – Library manager for the conversion.
- **original_irradfile** (*d1s_parser.IrradiationFile, optional*) – original irradiation file. The default is None.
- **original_reacfile** (*d1s_parser.ReactionFile, optional*) – original reaction file. The default is None.

Returns

- **newirradiations** (*list*) – list of Irradiation objects coming from the translation.
- **newreactions** (*list*) – list of Reactions objects coming from the translation.

14.3.3 D1S5_InputFile

class inputfile.**D1S5_InputFile**(*cards*, *matlist*, *name=None*)
Bases: *inputfile.D1S_Input*

Object representing an MCNP input file

Parameters

- **cards** (*dic*) – contains the cells, surfaces, settings and title cards.
- **matlist** (*matreader.MatCardList*) – material list in the input.
- **name** (*str, optional*) – name associated with the file. The default is None.

Returns

Return type None.

add_stopCard(*nps*, *ctme*, *precision*)
STOP card is not supported in MCNP 5. This simply is translated to a nps card. Warnings are prompt to the user if ctme or precision are specified.

Parameters

- **nps** (*int*) – number of particles to simulate
- **= int** (*ctme*) – computer time

- **= (str** (*precision*) – tuple indicating the tally number and the precision requested
- **float)** – tuple indicating the tally number and the precision requested

Returns

Return type None.

14.4 parsersD1S module

14.4.1 IrradiationFile

class parsersD1S.**IrradiationFile**(*nsc*, *irr_schedules*, *header=None*, *formatting=[8, 14, 13, 9]*, *name='irrad'*)

Bases: object

Object representing an irradiation D1S file

Parameters

- **nsc** (*int*) – number of irradiation schedule.
- **irr_schedules** (*list of Irradiation object*) – contains all irradiation objects.
- **header** (*str, optional*) – Header of the file. The default is None.
- **formatting** (*list of int, optional*) – fwf values for the output columns. The default is [8, 14, 13, 9].
- **name** (*str, optional*) – name of the file. The default is 'irrad'.

Returns

Return type None.

classmethod from_text(*filepath*)

Parse irradiation file

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **filepath** (*str/path*) – path to the irradiation file.

Returns

Return type None.

get_daughters()

Get a list of all daughters among all irradiation files

Returns list of daughters.

Return type list

get_irrad(*daughter*)

Return the irradiation correspondent to the daughter

Parameters **daughter** (*TYPE*) – DESCRIPTION.

Returns Returns the irradiation corresponding to the daughter. If no irradiation is found returns None.

Return type *Irradiation*

write (*path*)
Write the D1S irradiation file

Parameters **path** (*str or path*) – output path where to save the file (only directory).

Returns

Return type None.

14.4.2 Irradiation

class parsersD1S.**Irradiation** (*daughter*, *lambd*, *times*, *comment=None*)
Bases: object

Irradiation object

Parameters

- **daughter** (*str*) – daughter nuclide (e.g. 24051).
- **lambd** (*str*) – disintegration constant [1/s].
- **times** (*list of strings*) – time correction factors.
- **comment** (*str, optional*) – comment to the irradiation. The default is None.

Returns

Return type None.

classmethod from_text (*text*, *nsc*)
Parse a single irradiation

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **text** (*str*) – text to be parsed.
- **nsc** (*int*) – number of irradiation schedule.

Returns DESCRIPTION.

Return type TYPE

14.4.3 ReactionFile

class parsersD1S.**ReactionFile** (*reactions*, *name='react'*)
Bases: object

Reaction file object

Parameters

- **reactions** (*list*) – contains all reaction objects contained in the file.
- **name** (*name, optional*) – file name. The default is 'react'.

Returns

Return type None.

change_lib (*newlib*, *libmanager=None*)
change the parent library tag of the reactions. If no libmanager is provided, the check on the availability of the parent in the xsdir file will be not performed.

Parameters

- **newlib** (*str*) – (e.g. 31c).
- **libmanager** (LibManager, *optional*) – Object managing library operations. The default is None.

Returns

Return type None.

classmethod from_text(*filepath*)
Generate a reaction file directly from text file

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **filepath** (*str or path*) – file to read.

Returns Reaction File Object.

Return type *ReactionFile*

get_parents()
Get a list of all parents

Returns

Return type None.

write(*path*)
write formatted reaction file

Parameters **path** (*str/path*) – path to the output file (only dir).

Returns

Return type None.

14.4.4 Reaction

class parsersD1S.**Reaction**(*parent*, *MT*, *daughter*, *comment=None*)
Bases: object

Represents a single reaction of the reaction file

Parameters

- **parent** (*str*) – parent nuclide ZZAAA.XXc representing stable isotope to be activated. ZZ and AAA represent the atomic and mass number and extension XX, is the extension number of the modified D1S library.
- **MT** (*str*) – integer, reaction type (ENDF definition).
- **daughter** (*str*) – integer, tag of the daughter nuclide. The value could be defined as ZZAAA of daughter nuclide, but any other identification type (with integer value) can be used.
- **comment** (*str, optional*) – comment to the reaction. The default is None.

Returns

Return type None.

change_lib(*newlib*)
Change the library tag

Parameters **newlib** (*str*) – (e.g. 52c).

Returns

Return type None.

classmethod from_text(*text*)
Create a reaction object from text

Parameters

- **cls** (*TYPE*) – DESCRIPTION.
- **text** (*str*) – formatted text describing the reaction.

Returns Reaction object.

Return type *Reaction*

write()
Generate the reaction text

Returns **text** – reaction text for D1S input.

Return type str

CHAPTER

FIFTEEN

POST-PROCESSING

In this section the most useful classes that can be used during benchmark post-processing are described.

15.1 output module

15.1.1 AbstractOutput

class output.**AbstractOutput**
Bases: abc.ABC

static _get_output_files(*results_path*)
Recover the meshtal and outp file from a directory

Parameters **results_path** (*str or path*) – path where the MCNP results are contained.

Raises **FileNotFoundError** – if either meshtal or outp are not found.

Returns

- **mfile** (*path*) – path to the meshtal file
- **ofile** (*path*) – path to the outp file

abstract compare()
To be executed when a comparison is requested

abstract single_postprocess()
To be executed when a single pp is requested

15.1.2 BenchmarkOutput

class output.**BenchmarkOutput**(*lib*, *testname*, *session*)
Bases: *output.AbstractOutput*

General class for a Benchmark output

Parameters

- **lib** (*str*) – library to post-process
- **testname** (*str*) – Name of the benchmark
- **session** (Session) – Jade Session
- **exp** (*str*) – the benchmark is an experimental one

Returns

Return type None.

compare()
Generates the full comparison post-processing (excel and atlas)

Returns

Return type None.

single_postprocess()
Execute the full post-processing of a single library (i.e. excel, raw data and atlas)

Returns

Return type None.

15.1.3 MCNPOutput

class output.**MCNPoutput**(*mctal_file*, *output_file*, *meshtal_file=None*)
Bases: object

Class representing all outputs coming from and MCNP run

Parameters

- **mctal_file** (*path like object*) – path to the mctal file.
- **output_file** (*path like object*) – path to the outp file.
- **meshtal_file** (*path like object, optional*) – path to the meshtal file. The default is None.

Returns

Return type None.

organize_mctal()
Retrieve and organize mctal data into a DataFrame.

Returns

- **tallydata** (*pd.DataFrame*) – organized tally data.
- **totalbin** (*pd.DataFrame*) – organized tally data (only total bins).

15.1.4 ExcelOutputSheet

class output.**ExcelOutputSheet**(*template*, *outpath*)
Bases: object

Excel workbook containing the post-processed results

Parameters

- **template** (*path like object*) – path to the sheet template.
- **outpath** (*path like object*) – dump path for the excel.

Returns

Return type None.

copy_sheets(*wb_origin_path*)
Copy all sheets of the selected excel file into the current one

Parameters **wb_origin_path** (*str/path*) – Path to excel file containing sheets to add.

Returns

Return type None.

insert_cutted_df(*startcolumn*, *df*, *ws*, *ylim*, *startrow=None*, *header=None*, *index_name=None*, *cols_name=None*, *index_num_format='0'*, *values_format=None*)

Insert a DataFrame in the excel cutting its columns

Parameters

- **startcolumn** (*str/int*) – Excel column where to put the first DF column.
- **df** (*pd.DataFrame*) – global DF to insert.
- **ws** (*str*) – Excel worksheet where to insert the DF.
- **ylim** (*int*) – limit of columns to use to cut the DF.
- **startrow** (*int, optional*) – initial Excel row. The default is None, the first available is used.
- **header** (*tuple (str, value)*) – contains the tag of the header and the header value. DEAFAULT is None
- **index_name** (*str*) – Name of the Index. DEAFAULT is None
- **cols_name** (*str*) – Name of the columns. DEFAULT is None
- **index_num_format** (*str*) – format of index numbers
- **values_format** (*str*) – how to format the values. DEAFAULT is None

Returns

Return type None.

insert_df(*startcolumn*, *df*, *ws*, *startrow=None*, *header=None*, *print_index=True*, *idx_format='0'*, *cols_head_size=12*, *values_format=None*)

Insert a DataFrame (df) into a Worksheet (ws) using xlwings.

Parameters

- **startcolumn** (*int or str*) – Starting column where to insert the DataFrame. It can be expressed both as an integer as a letter in Excel fashion.
- **df** (*pandas.DataFrame*) – DataFrame to insert in the excel sheet
- **ws** (*str*) – name of the Excel worksheet where to put the DataFrame.
- **startrow** (*int*) – starting row where to put the DataFrame. Default is None that triggers the use of the memorized first free row in the excel sheet
- **header** (*tuple (str, value)*) – contains the tag of the header and the header value. DEAFAULT is None
- **print_index** (*bool*) – if True the DataFrame index is printed. DEAFAULT is True.
- **idx_format** (*str*) – how to format the index values. DEAFAULT is '0' (integer)
- **cols_head_size** (*int*) – Font size for columns header. DEAFAULT is 12
- **values_format** (*str*) – how to format the values. DEAFAULT is None

Returns

Return type None

save()
Save Excel

15.2 expoutput module

15.2.1 ExperimentalOutput

class expoutput.**ExperimentalOutput**(**args*, ***kwargs*)
Bases: *output.BenchmarkOutput*

This extends the Benchmark Output and creates an abstract class for all experimental outputs.

Parameters

- ***args** (*TYPE*) – see BenchmarkOutput doc.
- ****kwargs** (*TYPE*) – see BenchmarkOutput doc.
- **multiplerun** (*bool*) – this additional keyword specifies if the benchmark is composed by more than one MCNP run. It defaults to False.

Returns

Return type None.

abstract _build_atlas(*tmp_path*, *atlas*)
Fill the atlas with the customized plots. Creation and saving of the atlas are handled elsewhere.

Parameters

- **tmp_path** (*path*) – path to the temporary folder where to dump images.
- **atlas** (*Atlas*) – Object representing the plot Atlas.

Returns **atlas** – After being filled the atlas is returned.

Return type Atlas

_extract_outputs()
Extract, organize and store the results coming from the MCNP runs

Returns

Return type None.

abstract _pp_excel_comparison()
Responsible for producing excel outputs

_print_raw()
Dump all the raw data

Returns

Return type None.

abstract _processMCNPdata(*mctal*)
Given an mctal file object return the meaningful data extracted. Some post-processing on the data may be foreseen at this stage.

Parameters **mctal** (*MCTAL*) – object representing an MCTAL file.

Returns the type of item can vary based on what the user intends to do whith it. It will be stored in an organized way in the self.results dictionary

Return type item

static _read_exp_file(*filepath*)
Default way of reading a csv file

Parameters **filepath** (*path/str*) – experimental file results to be read.

Returns Contain the data read.

Return type pd.DataFrame

_read_exp_results()
Read all experimental results and organize it in the self.exp_results dictionary. If multirun is set to true the first layer of the dictionary will consist in the different folders and the second layer will be the different files. If it is not multirun, insetead, only one layer of the different files will be generated. All files need to be in .csv format. If a more complex format is provided, the user should ovveride the _read_exp_file method.

Returns

Return type None.

build_atlas()
Creation and saving of the atlas are handled by this function while the actual filling of the atlas is left to _build_atlas which needs to be implemented for each child class.

Returns

Return type None.

compare()
Complete the routines that perform the comparison of one or more libraries results with the experimental ones.

Returns

Return type None.

pp_excel_comparison()
At the moment everything is handled by _pp_excel_comparison that needs to be implemented in each child class. Some standard procedures may be added in the feature in order to reduce the amount of ex-novo coding necessary to implement a new experimental benchmark.

Returns

Return type None.

single_postprocess()
Always raise an Attribute Error since no single post-processing is foreseen for experimental benchmarks

Raises **AttributeError** – DESCRIPTION.

Returns

Return type None.

15.3 plotter module

15.3.1 Plotter

class plotter.**Plotter**(*data*, *title*, *outpath*, *outname*, *quantity*, *unit*, *xlabel*, *testname*, *ext='.png'*)

Bases: object

Object Handling plots

Parameters

- **data** (*list*) – data = [data1, data2, ...] data1 = {'x': x data, 'y': y data, 'err': error data, 'ylabel': data label}
- **title** (*str*) – plot title
- **outpath** (*str/path*) – path to save image
- **outname** (*str*) – name of the image file
- **quantity** (*str*) – quantity of the y axis
- **unit** (*str*) – unit of the y axis
- **xlabel** (*str*) – name of the x axis
- **testname** (*str*) – name of the benchmark
- **ext** (*str*) – extension of the image to save. Default is '.png'

Returns

Return type None.

__init__(*data*, *title*, *outpath*, *outname*, *quantity*, *unit*, *xlabel*, *testname*, *ext='.png'*)

Object Handling plots

Parameters

- **data** (*list*) – data = [data1, data2, ...] data1 = {'x': x data, 'y': y data, 'err': error data, 'ylabel': data label}
- **title** (*str*) – plot title
- **outpath** (*str/path*) – path to save image
- **outname** (*str*) – name of the image file
- **quantity** (*str*) – quantity of the y axis
- **unit** (*str*) – unit of the y axis
- **xlabel** (*str*) – name of the x axis
- **testname** (*str*) – name of the benchmark
- **ext** (*str*) – extension of the image to save. Default is '.png'

Returns

Return type None.

plot(*plot_type*)

Function to be called to actually perform the plot

Parameters **plot_type** (*str*) – plot type. The current available ones are ['Binned graph', 'Ratio graph', 'Experimental points', 'Discreet Experimental points', 'Grouped bars', 'Waves'].

Raises **ValueError** – if plot type is not among the available ones.

Returns **outp** – path to the saved image.

Return type path like object

CHAPTER

SIXTEEN

JADE TESTING

TBD

CHAPTER

SEVENTEEN

LICENSE

JADE software is licensed under the *GNU GPLv3 License*.

The following external python modules are re-distributed toghether with JADE and their licenses needs to be propagated:

- **MCTAL_READER2.py**, licensed under the *GNU GPLv3 License*;
- **xsdirpyne.py**, licensed under the *Pyne License*.

17.1 GNU GPLv3 License

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
```

(continues on next page)

(continued from previous page)

```
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.
```

(continues on next page)

(continued from previous page)

```
  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.
```

(continues on next page)

(continued from previous page)

```
  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
```

(continues on next page)

(continued from previous page)

```
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
```

(continues on next page)

(continued from previous page)

```
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.
```

(continues on next page)

(continued from previous page)

```
  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
```

(continues on next page)

(continued from previous page)

```
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
```

(continues on next page)

(continued from previous page)

```
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
```

(continues on next page)

(continued from previous page)

```
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
```

(continues on next page)

(continued from previous page)

```
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.
```

(continues on next page)

(continued from previous page)

```
  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

(continues on next page)

(continued from previous page)

```
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```

17.2 Pyne License

```
Copyright 2011-2020, the PyNE Development Team. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:

   1. Redistributions of source code must retain the above copyright notice, this␣
↪list of
      conditions and the following disclaimer.

   2. Redistributions in binary form must reproduce the above copyright notice, this␣
↪list
      of conditions and the following disclaimer in the documentation and/or other␣
↪materials
      provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE PYNE DEVELOPMENT TEAM ``AS IS'' AND ANY EXPRESS OR␣
↪IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY␣
↪AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER>␣
↪OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS␣
↪OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND␣
↪ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the
authors and should not be interpreted as representing official policies, either␣
↪expressed
or implied, of the stakeholders of the PyNE project or the employers of PyNE␣
↪developers.
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------
The files cpp/measure.cpp and cpp/measure.hpp are covered by:

Copyright 2004 Sandia Corporation.  Under the terms of Contract
DE-AC04-94AL85000 with Sandia Coroporation, the U.S. Government
retains certain rights in this software.

https://press3.mcs.anl.gov/sigma/moab-library

--------------------------------------------------------------------------------
The files in fortranformat/ are covered by:

The MIT License. Copyright (c) 2011 Brendan Arnold

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

https://bitbucket.org/brendanarnold/py-fortranformat/src/
```

CHAPTER

EIGHTEEN

CONTRIBUTORS

JADE is the results of a joint effort between NIER ingegneria, Università di Bologna (UNIBO) and Fusion For Energy (F4E).

NIER

MAKING CHANGE HAPPEN. MAKING LIFE BETTER.

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA A.D. 1088

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

FUSION FOR ENERGY

Key People:

Name	Contribution	Institution/Company	Contacts
Davide Laghi	Main developer	NIER and UNIBO	d.laghi@nier.it
Marco Fabbri	Project manager and expert	F4E	marco.fabbri@f4e.europa.eu
Lorenzo Isolan	Tester	UNIBO	lorenzo.isolan2@unibo.it
Marco Sumini	Expert	UNIBO	marco.sumini@unibo.it

CHAPTER

NINETEEN

LIST OF PUBLICATIONS AND CONTRIBUTIONS

19.1 Publications featuring JADE

- D. Laghi, M. Fabbri, L. Isolan, R. Pampin, M. Sumini, A. Portone and A. Trkov, 2020, "JADE, a new software tool for nuclear fusion data libraries verification & validation", *Fusion Engineering and Design*, **161** 112075. doi: https://doi.org/10.1016/j.fusengdes.2020.112075
- D. Laghi, M. Fabbri, L. Isolan, M. Sumini, G. Shnabel and A. Trkov, 2021, "Application Of JADE V&V Capabilities To The New FENDL v3.2 Beta Release", *Nuclear Fusion*, **61** 116073. doi: https://doi.org/10.1088/1741-4326/ac121a

19.2 Benchmarks Related Publications

- A. Milocco, A. Trkov and I. A. Kodeli, 2010, "The OKTAVIAN TOF experiments in SINBAD: Evaluation of the experimental uncertainties", *Annals of Nuclear Energy*, **37** 443-449
- I.Kodeli, E. Sartori and B. Kirk, "SINBAD - Shielding Benchmark Experiments - Status and Planned Activities", *Proceedings of the ANS 14th Biennial Topical Meeting of Radiation Protection and Shielding Division*, Carlsbad, New Mexico (April 3-6, 2006)
- D. Leichtle, B. Colling, M. Fabbri, R. Juarez, M. Loughlin, R. Pampin, E. Polunovskiy, A. Serikov, A. Turner and L. Bertalot, 2018, "The ITER tokamak neutronics reference model C-Model", *Fusion Engineering and Design*, **136** 742-746
- M. Sawan, 1994, "FENDL Neutronics Benchmark: Specifications for the calculational and shielding benchmark", (Vienna: INDC(NDS)-316)
- M. Martone, M. Angelone, and M. Pillon. "The 14 MeV Frascati neutrongenerator". In:Journal of Nuclear Materials 212-215 (1994). Fusion ReactorMaterials, pp. 1661–1664
- P. Batistoni, M. Angelone, L. Petrizzi, and M. Pillon. "Benchmark Experimentfor the Validation of Shut Down Activation and Dose Rate in a Fusion Device".In: Journal of Nuclear Science and Technology 39.sup2 (2002), pp. 974–977.
- K. Seidel, Y. Chen, U. Fischer, H. Freiesleben, D. Richter, and S. Unholzer."Measurement and analysis of dose rates and gamma-ray fluxes in an ITERshut-down dose rate experiment". In:Fusion Engineering and Design 63-64 (2002), pp. 211–215.
- R. Pampin, A. Davis, R.A. Forrest, D.A. Barnett, I. Davis, and M.Z. Youssef."Status of novel tools for estimation of activation dose". In:Fusion Engineeringand Design 85.10 (2010). Proceedings of the Ninth International Symposiumon Fusion Nuclear Technology, pp. 2080–2085.
- J. Sanz, O. Cabellos, and N. Garcia-Herranz. Inventory Code for Nuclear Applications: User's Manual V. 2008. RSICC. 2008.

19.3 Miscellaneous

CHAPTER

TWENTY

INDICES AND TABLES

- genindex
- modindex
- search