FISPACT-II Exercises

Thomas Stainer, UKAEA
Andrew Davis, UKAEA
Mark Gilbert, UKAEA

June 20, 2018

1 Setup

For the sake of these exercises we must set an environment variable named FISPACT which points to the FISPACT-II 4.0 binary. For linux users this is done via

```
[1206][user1@fispact:~/] export FISPACT=/path/to/fispact-4.0.0
```

Then we can simply run fispact via

```
[1206][user1@fispact:~/] $FISPACT
```

The above commands are relevant for Linux and MacOS systems. If you are running on a Windows machine, the alternative commands are needed.

```
C:\> set FISPACT=/path/to/fispact-4.0.0
```

And running fispact with

```
[1206][user1@fispact:~/] %FISPACT%
```

The reader is reminded that these exercises complement the FISPACT-II manual and the manual should be consulted first before attempting these exercises. All of the setup is explained in the manual.

To run fispact it requires knowledge of your nuclear data, this is governed by the *files* file. Each exercise comes with at least one *files* file and is required as input when running the application. It is assumed that the nuclear data is in text format and already downloaded onto the machine intended to run these examples. If you do not have the nuclear data on your machine, this is a prerequisite and should be done before attempting to run any examples. The nuclear data is publicly available from the fispact website.

The paths in the files file are set to point to the base directory */opt/fispact/nuclear_data* by default. If your fispact nuclear data installation is not set here then you need to change the paths in each of your files files. It is always good practice to check your files file is valid before running any fispact inventory simulation. If not properly checked then it is likely fispact will throw an error.

All exercises come with the reference outputs, these have *_ref* prepended before the file extension and are bundled in a subdirectory named *ref* for each example. For example the output and log files for *ex1.i* are named *ex1_ref.out* and *ex1_ref.log* respectively. These are not required to run the exercises but act as a useful reference to compare against when coming into issues with running the examples.

2 Exercise 1

This is the introductory example for running FISPACT-II. It aims to introduce you how to run fispact from the command line. No knowledge is assumed about the input and files file at this point.

Exercise 1 is very basic example which simply performs the collapse and condense for a given flux. This exercise requires 3 input files, which are provided in the exercise 1 directory.

```
[1206][user1@fispact:~/exercises/1] ls
    ex1.i   files   fluxes
```

The input file is *ex1.i*, the files file is *files* and the fluxes file is *fluxes*. To run the exercise we must be in the same directory as the input files and then supply the input file as the first argument, without the *.i* extension, as below.

```
[1206][user1@fispact:~/exercises/1] $FISPACT ex1
```

In this case the files file is picked up by default since fispact looks for it in the current directory, if not specified. However, it is also possible to explicitly pass the files file as below.

```
[1206][user1@fispact:~/exercises/1] $FISPACT ex1 files
```

Run both commands and show that they give the same result. Check that no fatal errors occurred. You should end up with four additional files after a successful run.

```
[1210][user1@fispact:~/exercises/1] ls
    ex1.i   ex1.log    ex1.out    files   fluxes     ARRAYX    COLLAPX
```

If you did get errors, it is likely that you did not change your files file to match the paths for your local machine, or the FISPACT environment variable has not been set correctly. Please check these and revise section 1 for more details on this.

3 Exercise 2

This exercise builds on exercise 1 by printing out the collapsed cross section data for each reaction.

Exercise 2 uses the same nuclear data as exercise 1, but it consists of two runs to compare two different fluxes. We make use of the keyword **PRINTLIB**, for which we use the option **4**, to print the collapsed cross section data to the output file.

```
[1206][user1@fispact:~/exercises/2] ls
    ex2.i   files1   files2   fluxes1    fluxes2
```

The input file is *ex2.i*, the files file are *files1* and *files2*, and the two different fluxes files are *fluxes1* and *fluxes2*. We require two flies files because we want to compare two different fluxes which must be specified in the files file. Files file 1 points to *fluxes1*, files file 2 points to *fluxes2*. Do a diff to compare these files files to show this is the case.
First we run with *fluxes1* using *files1*.

```
[1206][user1@fispact:~/exercises/2] $FISPACT ex2 files1
```

Again check that no fatal errors occurred. If no errors occurred you should have the output file, *ex2.out*, which should contain the collapsed cross section data. Rename this to *ex2_1.out*. Do the same for *files2* and rename this output as *ex2_2.out*.

Q. What value of Al 24 (n,p) do you get for fluxes1?
A. $\mathbf{1.63927 \times 10^5}$ **barns**

Hint: you can use the following grep command to get this.

```
grep "Al 24  (n,p " ex2_1.out
```

Q. What value of Al 24 (n,p) do you get for fluxes2?
A. $\mathbf{1.58635 \times 10^1}$ **barns**

Hint: you can use the following grep command to get this.

```
grep "Al 24  (n,p " ex2_2.out
```

4 Exercise 3

This exercise is taken directly from the getting started examples, specifically for the FNS Inconel 600 simulation.

FNS Inconel uses the TENDL 2017 dataset, but otherwise the nuclear data is the same as previous exercises. The aim of this exercise is to do a full inventory calculation and plot the total heat output during the cooling phase. This example consists of 4 sections: collapse, condense, printlib and inventory. The former three have been somewhat covered in the previous exercise, but the last step for the inventory will show some of the FISPACT-II functionality has to offer.

Exercise 3 should contain four input files.

```
[1206][user1@fispact:~/exercises/3] ls
    collapse.i    condense.i     print_lib.i     inventory.i
```

Compared to the previous exercises, we have broken up the collapse and condense steps so that we can perform different analysis and simulations without having to rerun the computationally expensive parts of reading the nuclear data and processing it. The *collapse.i* input performs the **GETXS** step, the *condense.i* then performs the **GETDECAY** step, producing the **COLLAPX** and **ARRAYX** files respectively. The *print_lib.i* outputs the cross section data, decay data, fission yields, branching ratios, and more. The *inventory.i* input file then defines our initial setup and irradiation and cooling schedule. This will then be used to produce a total heat graph as a function of cooling time.

Run the collapse, condense, print_lib and inventory inputs and check the logs and outputs. Note that collapse must be ran first, followed by condense, otherwise errors will occur.
Run the *inventory.plt* script with gnuplot, as below, and check you get the output ps file *inventory.gra.ps*.

```
[1206][user1@fispact:~/exercises/3] gnuplot inventory.plt
[1206][user1@fispact:~/exercises/3] ls
    collapse.i         condense.i          print_lib.i
    inventory.i        collapse.out          condense.out
    print_lib.out      inventory.out         inventory.plt
    collapse.log       condense.log        print_lib.log
    inventory.log      inventory.json      inventory.gra
    inventory.gra.ps     files      fluxes
```

Check that the graph looks like the one in figure 1.

5 Exercise 4

This exercise builds on from exercise 3, FNS Inconel 600 but focuses on using the JSON output file to perform analysis. Instead of having 4 input files: collapse, condense, printlib and inventory, this has been condensed to one input file *inventory.i*, ignoring the printlib step.

Exercise 4 should contain one input file along with the files file and the fluxes file from exercise 3. In addition to these files there should exist three other files with extension *.py*, these are Python 3 scripts that will read the JSON output and make some plots. If you are unfamiliar with Python then you can choose

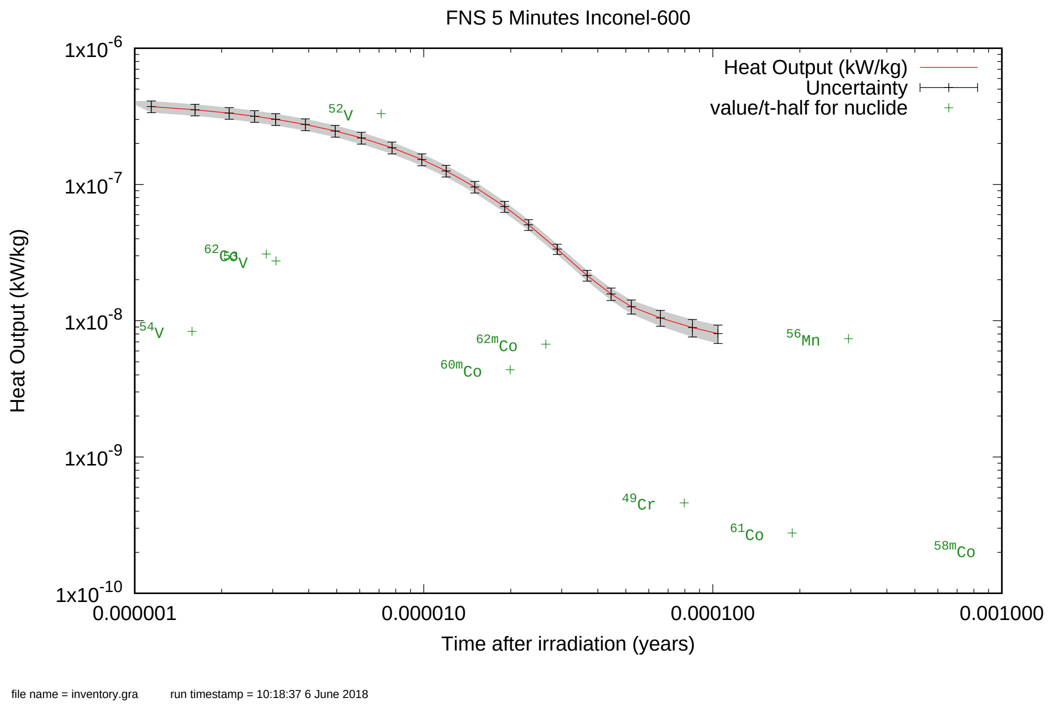


Figure 1: The total heat output (kW/kg) after the irradiation time (years) for FNS Inconel-600.

a different programming language to parse and analyse the JSON output or you can skip this exercise altogether. Python is chosen because it is a simple scripting language that has powerful plotting and charting libraries available.

```
[1206][user1@fispact:~/exercises/4] ls
       inventory.i         inventory1.py        inventory2.py        inventory3.py
```

Run the inventory input file and check that a valid JSON file is written, namely *inventory.json*. We will now use this to perform our analysis.

```
[1206][user1@fispact:~/exercises/4] python3 inventory1.py
```

Run the python3 script *inventory1.py* and check that an image file *pie.pdf* is now created. The script uses the JSON output to select the first time step, which shows the initial inventory, and then prints the elemental composition. Check that the printed values match with that specified in the input file. The pie chart should also reflect this composition, as shown in figure 2.

```
[1206][user1@fispact:~/exercises/4] python3 inventory.py
Cr 15.97%
Mn 0.39%
Fe 7.82%
Ni 75.82%
```

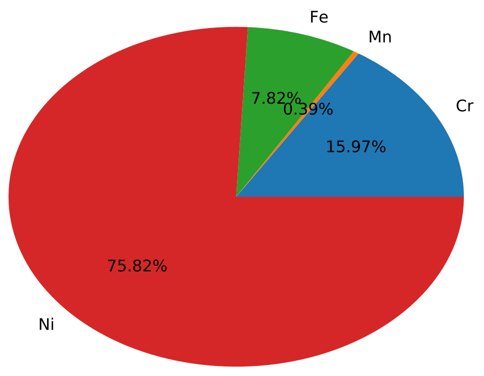


Figure 2: A pie chart showing the elemental composition specified in the initial inventory.

Can you now modify this script to do the same plot for the final time step, ignoring the initial elements. The *inventory2.py* file shows you how to do this, run this and compare with your example.

What do you get?
What is the most dominant element?

Figure 3 shows the answer.

```
[1206][user1@fispact:~/exercises/4] python3 inventory.py
H 1.94%
He 0.92%
Ti 1.09%
V 2.96%
Co 93.08%
Cu 0.00%
```

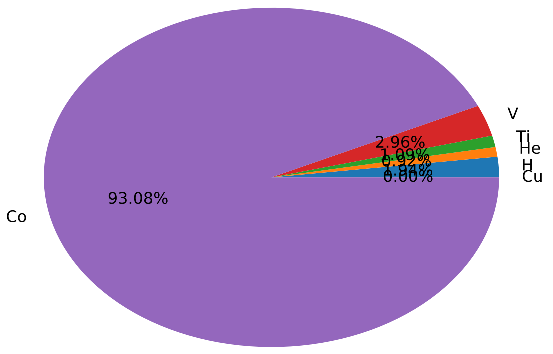


Figure 3: A pie chart showing the elemental composition specified in the final inventory, excluding the initial elements.

Can you take this further by getting the composition after the first irradiation phase?
Can you plot the elemental composition as a function of time? Again, the *inventory3.py* file shows you how to do this, run this and compare with your example.

6 Exercise 5

This exercise introduces the user to the open source Python3 tool, known as pypact. In addition to the JSON output, pypact can parse the existing standard FISPACT-II output file (*.out) and will therefore work for versions prior to

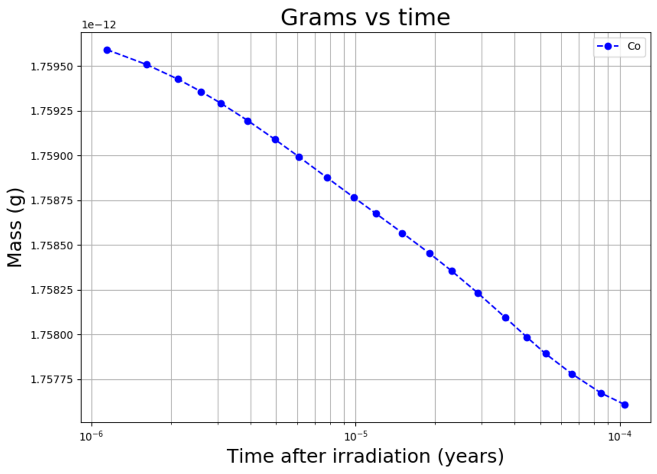


Figure 4: The mass, in grams, of Co as a function of time, post irradiation.

version 4.0. It is a Python3 tool, but is easy to install if you are familiar with the python package manager, pip. If you are not Python literate, you are welcome to ignore this exercise.

Exercise 5 illustrates that the same analysis can be performed using either the JSON output or the standard output and parsing it with pypact. Both output files will be used in this exercise and will produce the same chart as that which was produced in exercise 3, which was made with gnuplot. It shows an alternative and more bespoke approach to performing analysis.
Exercise 5 should contain one input file along with the files file and the fluxes file from exercise 3. In addition to these files there should exist one python file, *inventory.i*. The input file should be the same as the previous input file from exercise 4.

```
[1206][user1@fispact:~/exercises/4] ls
       inventory.i        inventory.py       fluxes         files
```

Again run the inventory input file and check that a valid JSON file and out file is written, namely *inventory.json* and *inventory.out*. We will now use this to perform our analysis. Run the python3 script *inventory1.py*.

```
[1206][user1@fispact:~/exercises/4] python3 inventory.py
```

After running the script, two identical plots should be dumped to screen. If not it may be that *matplotlib* was not installed. The two plots should match that of figure 5, one was produced using the JSON output data, while the other was produced from the standard .out file.

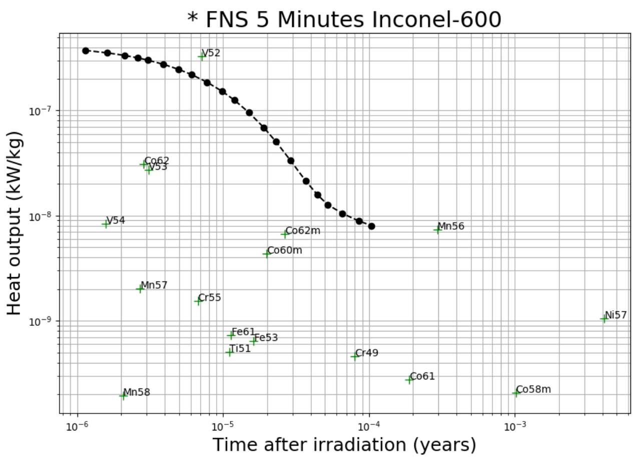


Figure 5: The total heat output for FNS Inconel 600 run as a function of cooling time. Dominate nuclide half lives are plotted with green crosses.

7 Exercise 6

This exercise examines the use of the PULSES keyword. The PULSES keyword is a compact way of representing repetitive irradiation scenarios, for example ignoring the early startup phase of H and DD irradiation, we expect 2 shifts of 8 hours each, running pulses of 300 seconds duration followed by 1500 seconds of dwell (flux off), followed by 8 hours of decay, 5 days a week with weekends off, for say 10 years. Using the PULSES keyword, we can model this using:

```
PULSES 10
  PULSES 5
    PULSES 2
      PULSES 8
        FLUX 1.0E14
        TIME 300 ATOMS
        FLUX 0.0
        TIME 1500 ATOMS
      ENDPULSES
      FLUX 0.0
    ENDPULSES
    FLUX 0.0
    TIME 8 HOURS ATOMS
  ENDPULSES
ENDPULSES
```

Note the above example is more verbose than needed for clarity, one could do this more compactly but perhaps less clearly:

```
PULSES 50
  PULSES 16
    FLUX 1.0E14
    TIME 300 ATOMS
    FLUX 0.0
    TIME 1500 ATOMS
  ENDPULSES
  FLUX 0.0
  TIME 8 HOURS ATOMS
ENDPULSES
```

7.1 Example A

In order to convince yourself that PULSES works, look at the at the example pulses.i example and introduce a multilayer irradiation schedule problem. So, lets say that we want to do 10 days of irradation, where in a single day, we have 1 hour of irradiation, followed by 11 hours of decay, twice.

1. Construct your example using the pulses keyword
2. Do the same as (1) but without the pulses keyword

7.2 Example B

1. MAST Upgrade - a device at CCFE is expected to run 52 weeks a year, 5 days a week (with weekends off), there are pulses 8 hours per day, and in any given hour there will be 2 pulses of 5 seconds duration, with $\sim$ 30 mins between pulses. Using the file included complete the irradiation section that defines this irradiation schedule - compute the activity after 1 year of operation, at shutdown 1 hour and 10 years decay time
2. Compute the same as in (1) but ensure that the activity is calculated immediately following the last pulse
3. Compute the activity as in (2) but at the following decay times, 5 mins, 1 hour, 1 day, 30 days, 60 days, 90 days, 180 days, 1 year, 2 year, 3 year, 5 year, 10 year, 20 year, 30 year, 50 year and 100 year

8 Exercise NUCGRAPH

This exercise examines the use of the `NUCGRAPH` keyword.

- Nuclide contributions to radiological quantities can be output in a separate file for plotting via, for example,:

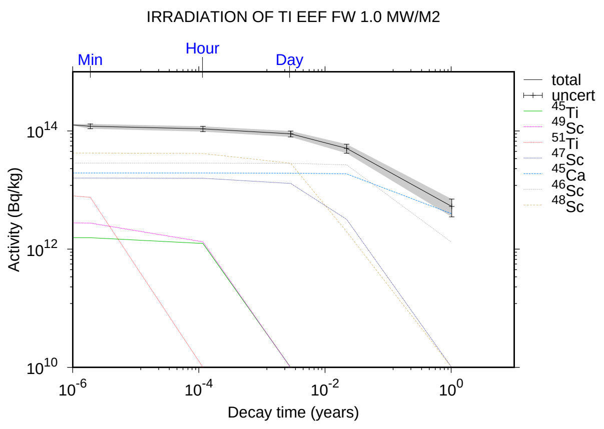
```
NUCGRAPH 1 1.0 1 1
```

Details of the various options of the code word are given at http://fispact.ukaea.uk/wiki/Keyword:NUCGRAPH

- the example in the nucgraph example folder requires (in order) the running of `collapse.i`, `condens.i`, and `test1.i`
- then to plot the example (of activation contributions to irradiated Ti) type:

```
gnuplot test1.pln
```

The result should be



- Additional: can you extend this example to print out other quantities?
  - E.g. decay heat
- Check http://fispact.ukaea.uk/wiki/Keyword:NUCGRAPH

9 Exercise multiflux

This exercise examines the ability of FISPACT-II to perform sequential irradiations under different particle spectra, in a single simulation.

- FISPACT-II can irradiate the same material with two different spectra in sequence (as demonstrated in W self-shielding case study)
- Two alternative approaches
  - One (via multiflux_full.i):

```
GETXS 1 709
GETDECAY 1
...
```

```
FISPACT
* run example
...
<< some irradiation>>
GETXS 1 709K
...
```

Here two collapses are performed in the same simulation and the required files file contains two blocks:

```
# collapsed cross section data input: plasma spectrum
fluxes fluxes.1
collapxo COLLAPX.1
collapxi COLLAPX.1
xs_endf ...
```

  - and then

```
# collapsed cross section data input: plasma spectrum
fluxes fluxes.2
collapxo COLLAPX.2
collapxi COLLAPX.2
xs_endf ...
```

The fluxes files, collapx files, etc. are “queued-up” for use in the order they are read.

- Alternatively, this same simulation can be performed via:
  - Two separate collapses (+condense) and then the inventory run

```
$FISPACT collapse1 files_collapse1
$FISPACT collapse2 files_collapse2
$FISPACT condense
$FISPACT multiflux_stepwise
```

Collapse files files just contain:

```
# collapsed cross section data input: plasma spectrum
fluxes fluxes.1
collapxo COLLAPX.1
collapxi COLLAPX.1
```

  - and in `multiflux_stepwise.i`

```
GETXS 0
GETDECAY 0
...
FISPACT
```

```
* run example
...
<< some irradiation>>
GETXS 0
...
```

The folder example is set up for this second choice, but can you reproduce the results using the one file approach?
Via `multiflux_full.i`