FISPACT-II Basic exercises

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

June 19, 2019

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 simple

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 'simple' is very basic example which simply performs the collapse and condense for a given flux but gives no output. This exercise requires 3 input files, which are provided in the basic directory.

```
[1206][user1@fispact:~/exercises/basic/simple] ls
    run.i   files   fluxes    ref/
```

The input file is *simple.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/basic/simple] $FISPACT simple
```

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/basic/simple]  $FISPACT simple 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.

```
[1206][user1@fispact:~/exercises/basic/simple] ls
    simple.i   simple.log    simple.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.
The four additional files that were created are the result of a FISPACT-II run. Every run will produce an output file (*.out) and log file (*.log) regardless of

input options and settings you set. It is always good practice to check and review the log file before analysing results. The additional files, ARRAYX and COLLAPX, are binary files that are produced due to decay and reaction data being read and processed. These binary files are platform and version specific and only relevant for simulations using the same nuclear data and fluxes file. If you do repeated calculations with static nuclear data and incident particle spectra then using these binary files can speed up simulation time (more on this later).

3 Exercise printlib

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

Exercise 'printlib' uses the same nuclear data as the previous exercise, 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/basic/printlib] ls
    printlib.i   files1   files2   fluxes1    fluxes2
```

The input file is *printlib.i*, the files file are *files1* and *files2*, and the two different fluxes files are *fluxes1* and *fluxes2*. One is a thermal neutron and the other is a 14 MeV D-T fusion source. We require two files 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. A plot of the incident neutron energy spectra is shown in figure 1. Can you plot them?

First we run with *fluxes1* using *files1*.

```
[1206][user1@fispact:~/exercises/basic/printlib] $FISPACT printlib files1
```

Again check that no fatal errors occurred. If no errors occurred you should have the output file, *printlib.out*, which should contain the collapsed cross section data. Rename this to *printlib_1.out*. Do the same for *files2* and rename this output as *printlib_2.out*.

Q. What value of Al 24 (n,p) do you get for fluxes1?
A. **1.63927×10^5 barns**

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

```
grep "Al 24  (n,p " printlib_1.out
```

Q. What value of Al 24 (n,p) do you get for fluxes2?

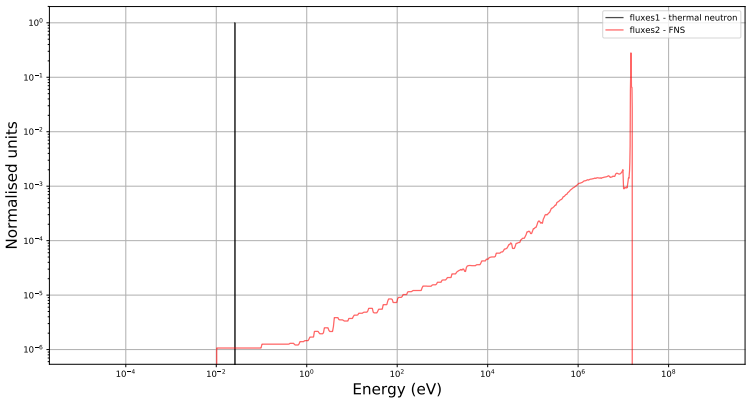

Figure 1: The incident particle spectra for a thermal neutron (black) and the FNS spectra (red).

A. **1.58635 × 10^1 barns**

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

```
grep "Al 24  (n,p " printlib_2.out
```

Why are the values so different? We need to look at the cross sections for Al 24 (n,p), this is shown in figure 2, with the corresponding spectra overlaid. Integrating the cross section with each flux should yield identical results to that found from the printlib. A later example will show how to extract the group wise cross section data for a given reaction.

4 Exercise inventory example

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/basic/inventory] ls
    collapse.i    condense.i     print_lib.i     inventory.i
```

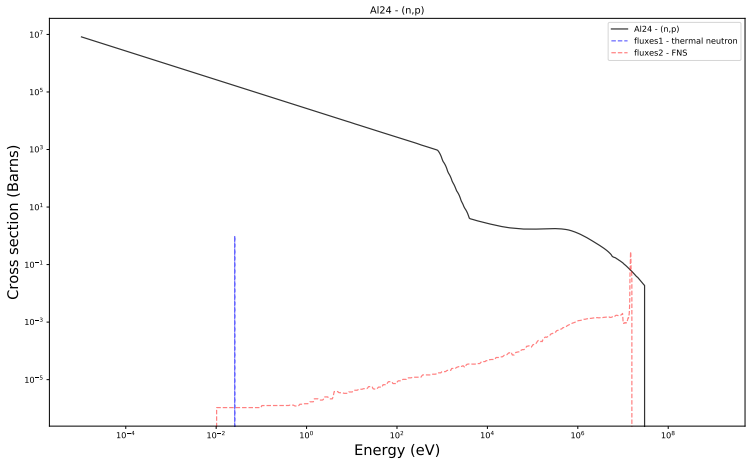

Figure 2: The cross section, in barns, for the (n,p) reaction on Al^{24}.

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/basic/inventory] gnuplot inventory.plt
[1206][user1@fispact:~/exercises/basic/inventory] 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 3.
Are you able to do the same analysis in just one file?

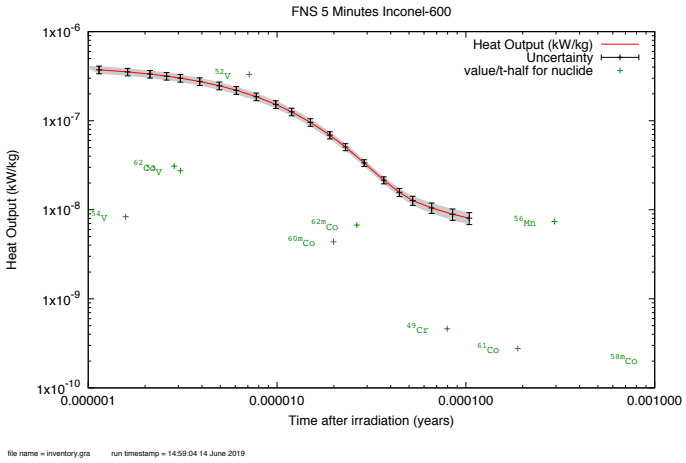

Figure 3: The total heat output (kW/kg) after the irradiation time (years) for FNS Inconel-600.

Can you produce a graph of the total activity as a function of time instead? What are the benefits of breaking the simulation up into different parts? What if you want to change the material? Which option is better?

The total activity (per unit mass) should match figure 4.

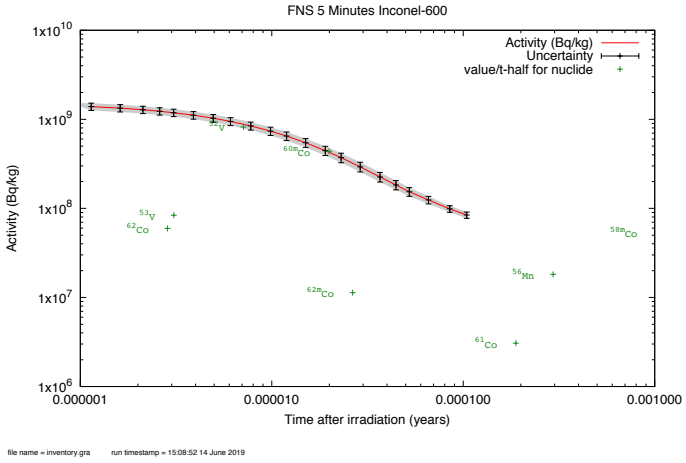

Figure 4: The total activity per unit mass (Bq/kg) after the irradiation time (years) for FNS Inconel-600.