FISPACT-II Basic exercises

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

November 23, 2020

1 Exercise compressxs

Before we start using FISPACT-II it is first useful to compress our nuclear data libraries into binary format to save us disk space and improve our simulation run times. We will use the *compress_xs_endf* tool to do this. The effect can be dramatic when processing large nuclear data libraries, such as TENDL2017.

We will first do a simple inventory calculation using TENDL 2017, without compress. Run the input file *'simpleinventorywithoutcompress.i'*.

How long did it take to run? On my machine it took roughly 106 seconds. Most of the time was actual spent reading the TENDL 2017 cross section data.

Now we will run *compress_xs_endf* to compress the cross sectional data. The tool is a binary executable program, like FISPACT-II, and should reside in the same directory as the main FISPACT-II program. Again, you can set an environment variable to point to the binary.

```
[$] export COMPRESS=/path/to/compress_xs_endf
```

To run the tool, it requires a files file pointing to corresponding index and cross section data. It only requires these two paths, using the keys *ind_nuc* and *xs_endf*. No other entries should be required in the files file.
Can you construct this files file for TENDL2017 using neutron incident data? Name it files.tendl17_n.

No input file is required for the tool and just requires command line arguments. The command line arguments are then:

```
[$] $COMPRESS <name> <particle> <group> <option> <filesfile>
```

Here the name is the name of the binary file produced without the *.bin* extension, particle is the character indicating the incident particle (n, g, a, d, t, or p), group is the group structure (typically 709), option indicates whether to include

uncertainties and covariances (option 5 does everything), and the files file should be the file we just created. Running *compress_xs_endf* will take some time, but it only needs to be performed once per host machine. A useful name would be *tal2017-n*. Try and run it and check the log to ensure no errors occurred.

In order to run the same inventory simulation but now with the newly compressed nuclear data library, we need to make a few changes to the input file and files file. The first step is to add an additional key and path to the files file. The key *xs_endfb* must now be added and point to our new binary file *tal2017-n.bin*. Secondly, in order for this path to be registered, the corresponding input file must be altered such that the *GETXS* keyword should have the 1 changed to -1 (this indicates to read from binary format). Note that without this, the standard text format will be used from the *xs_endf* path. Make these changes and create a new input file *simpleinventorywithcompress.i* based on the *simpleinventorywithoutcompress.i* but reading binary cross section data.

After running it what percent speed up do you get?
Hint: Use 'time' to precede the compress command to get the time taken in bash. On my machine it took roughly 64 seconds, almost 40% faster!

Check your output files have the same results.

We will now perform a more complex simulation, to highlight that the benefit of compress is diminished when enabling fission and using large irradiation schedules, as more time is spent solving the rate equation matrices and reading fission yield data, which is not included in the binary version. The input file *inventorywithoutcompress.i* reflects this added complexity using a fission case. Follow the same procedure as before and compare the runtimes and outputs. Again, check your output files have the same results, regardless of cross section format.

My runtime without compress was 181 seconds and with compress was 128 seconds - now only 30% speed up. The runtimes are very much machine specific.

A bash script exists which allows you to compress all versions of TENDL (14, 15, and 17) for all incident particles in one go (it will take a while). The script is aptly named *compress_all.sh*. Use this to compress all TENDL nuclear data libraries. Remember if we want to use them we need to use the files file key *xs_endfb* instead of *xs_endf*.

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?

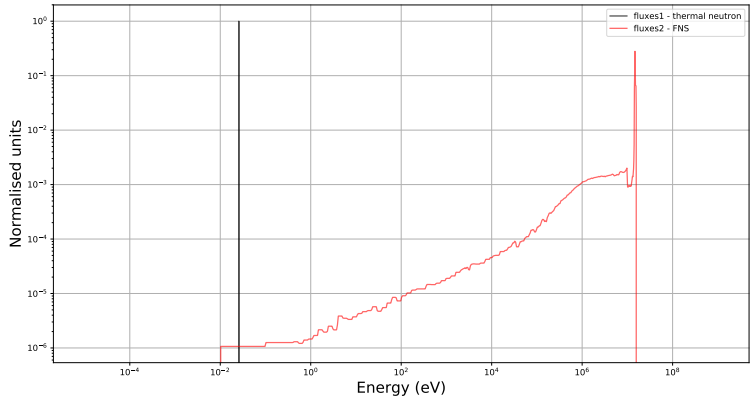

Figure 1: The incident particle spectra for a thermal neutron (black) and the FNS spectra (red).

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?
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.

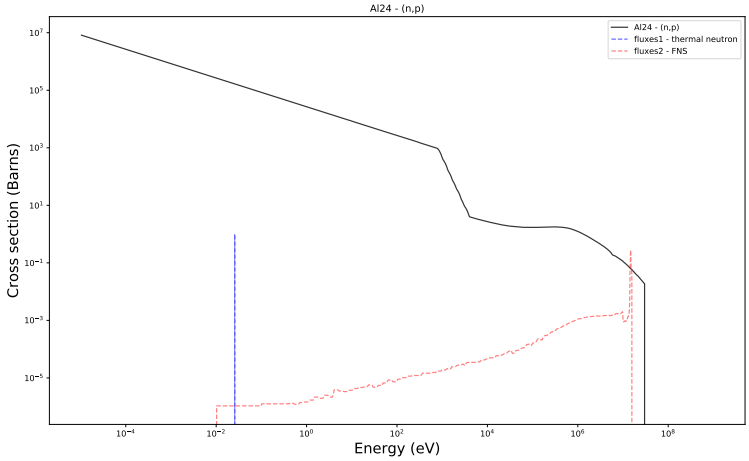

Figure 2: The cross section, in barns, for the (n,p) reaction on Al^{24}.

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
```

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?
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.

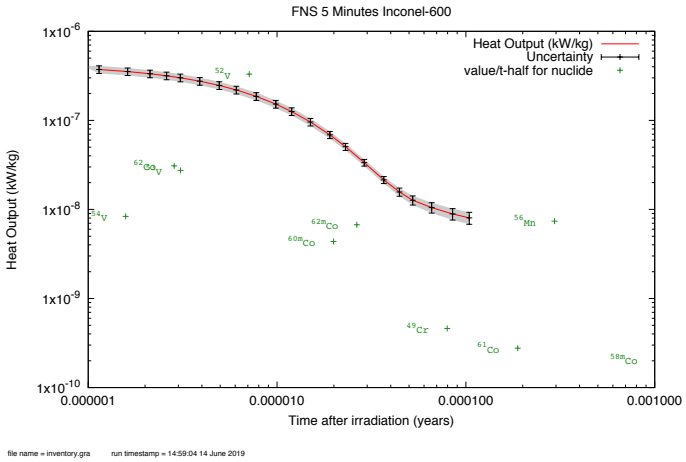

Figure 3: The total heat output (kW/kg) after the irradiation time (years) for FNS Inconel-600.

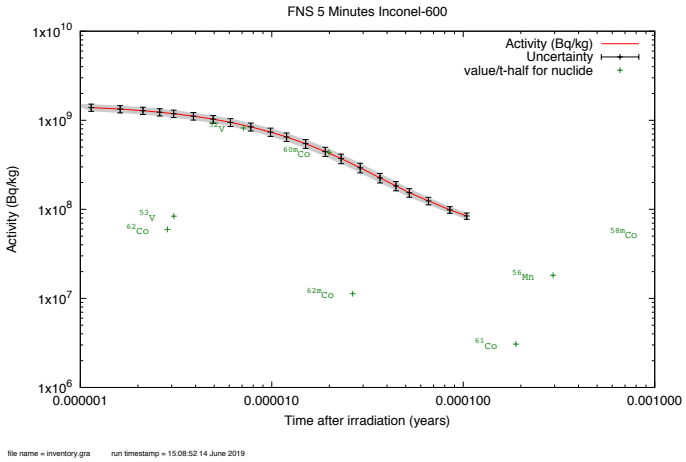

Figure 4: The total activity per unit mass (Bq/kg) after the irradiation time (years) for FNS Inconel-600.