

# **FISPACT-II**

## **Advanced Usage**

**Thomas Stainer, Mark Gilbert, Greg Bailey**  
**United Kingdom Atomic Energy Authority**

**FISPACT-II workshop**

**June 19-21, 2019, OECD/NEA, Paris**



This work was funded by the RCUK Energy Programme  
[Grant number EP/P012450/1]



# Additional tools

Not just FISPACT-II, additional tools included:

- **compress\_xs\_endf** – Turn nuclear data into binary format for quicker processing
- **extract\_xs\_endf** – Get the energy dependent cross section per reaction
- **listreactions** – Prints information about reaction data in ENDF-6 file
- **makenuclideindex** – Create a new index file given nuclide names

Other tools online and open source:

- **Pypact** – python3 package to parse output file and other functions
  - **GitHub** - <https://github.com/fispact>
- **Docker** – premade container environments with nuclear data included
  - **DockerHub** - <https://hub.docker.com/u/fispact/>

# GitHub



# Compress\_xs\_endf

Large nuclear data libraries can take a while to read...

- TENDL 2017 can take up to 2 minutes to read and process in it's entirety
- This can be improved by up to **60%** when using compress\_xs\_endf

Command line tool just like FISPACT-II

- Needs a lightweight files file
- No input file, just command line args
- Only works on ENDF-6 format, not EAF
- Only compresses reaction data, not decay data
- Binary files are platform and compiler dependent (non-transferable)
- **Reduce the size from 9GB to > 2GB!**

**Note: if we use binary files, we use GETXS -1 in input file and key xs\_endfb must point to path of binary file**

**Try the exercise in 'extended' folder, to understand how to use it and create the libraries.**

# Extract\_xs\_endf

Previously we compared different incident particle spectra and how this caused different collapsed cross section values. We showed the energy dependent cross section for Al 24 (n,p), but I did not show how to get this data.

FISPACT-II has a tool – `extract_xs_endf`, which will allow you to extract this data.

- Needs a lightweight files file
- No input file, just command line args
- **Does need a fluxes file** – even if you just want the XS

Try the exercise in ‘extended’ folder, to understand how to use it and plot cross sections.

# Listreactions

FISPACT-II has a tool – listreactions, which will allow you to print available channels in ENDF-6 files.

- No input files, just two args:
  1. “n”, “p”, “d”, “g”, “t”, “a”
  2. ENDF-6 file path

First argument is really redundant, since you can give it a proton file and input “g”, but for correctness use the correct particle for that file.

```
[tom@fispact:~/dev/]$ $LISTREACTIONS n /opt/nuclear_data/TENDL2017data/tal2017-n/gxs-709/H003g.asc
```

```

filename      = /Volumes/Pegasus2/Code/FISPACT-II/nuclear_data/TENDL2017data/tal2017-n/gxs-709/H003g.asc
parent name   = H   3
parent zai    = 10030
projectile    = neutron
data type     = TENDL

```

reaction	daughter	zai_d	mat	mf	mt	ns	lfs	lmf	elfs
file 3 data									
(n,total)		-5	131	3	1	2			
(n,E )	H 3	10030	131	3	2	2			
(n,2n )	H 2	10020	131	3	16	2			
(n,Xd )		-20	131	3	204	2			



# Group Convert

FISPACT-II uses binned (group-wise) data

A major problem with this:

- Group structure of F-II nuclear data is usually 709 for most libraries
- Whilst 709 is generally pretty good, your flux must match this structure
- If you have a spectra in another structure how do you use it with F-II?

Two solutions:

- Make your own nuclear data for that group – use NJOY, PREPRO (not recommended)
- Use **group convert** – F-II keyword (GRPCONVERT)



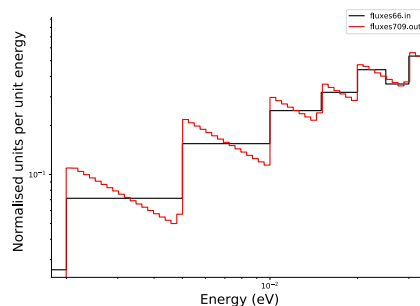
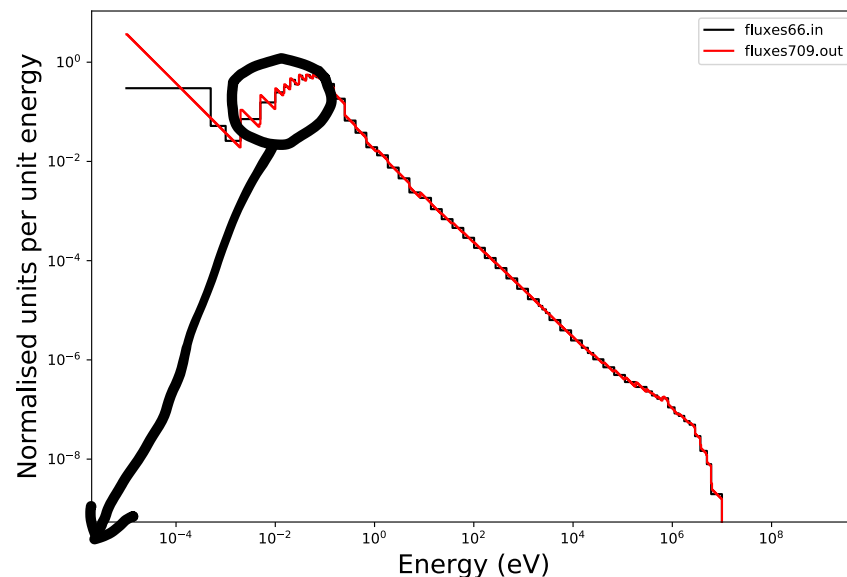
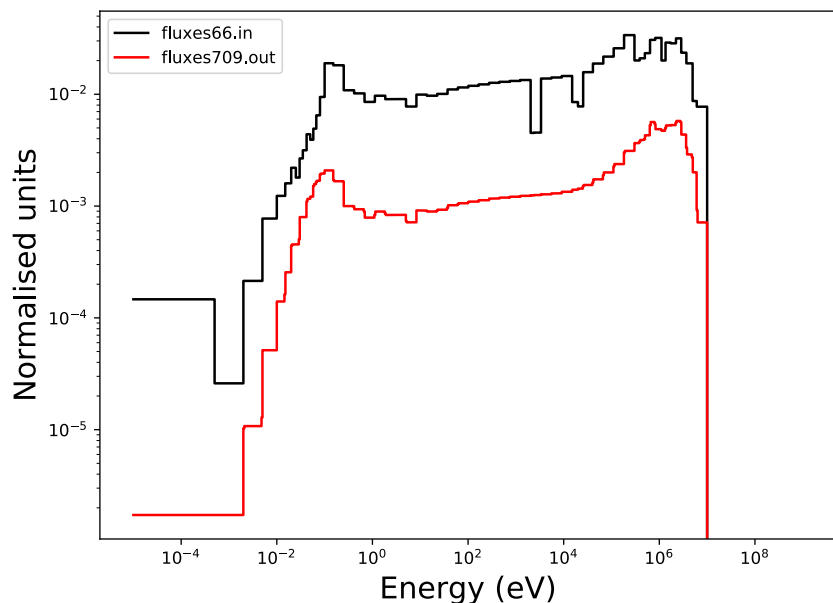
# Group Convert

- Input arbitrary group structure
  - Output F-II known group structure
  - Conversion can be done using **equal lethargy** or **equal energy** per bin method (CNVTYPE)
  - Splits groups based on weighting each input and output energy group
- $$\text{Lethargy} = u = \ln(E_0/E)$$
- Not a separate tool – use GRPCONVERT keyword in input file
  - Requires a files file with:
    - **ind\_nuc** – needed to run but not used
    - **arb\_flux** – input flux file
    - **fluxes** – output flux file

Recommended to use groups "close" to each other. But is possible to do 66 -> 1102 if desired. Results depend on how groups "line" up.

# Group Convert

Exercise 'grpconvert' shows conversion from group 66 to 709. Must compare using bin width (per unit energy).



Try the exercise in 'extended' folder.



# FISPACT-II Output Format

- Fairly easy to look inside and read a value
- Hard to script and parse values
- Output files can be quite large (often > 5MB)
- Grep will only take you so far

## Solutions:

- Write your own parser (not recommended)
- Use **TAB** files for easier parsing (limited output, limited precision)
- Use **JSON** output (standardised format, high precision, since v4 only)
- Use **pypact** (python3 package designed for parsing output)

Exercise on each in advanced section

- In most practical scenarios, energy spectra changes with time and space
- Important for burn-up calculations
- Need to allow for different spectra in inventory calculations, but...
  - One simulation/run = one collapse

⇒ When the spectra is changing, we must re-collapse

- Chain input1, output1 => input2, output2 =>...
- Can be difficult to parse output file
  - As mentioned before, a few options - **TAB**, **JSON**, **PYPACT**
- We will use the **TAB1** file to help automate this, since the input just needs to know the number of atoms

**Exercise multiflux shows how to do this**

FISPACT-II 4.0 now has a JSON output

```
{  
  "name" : "json",  
  "description" : "I am JSON",  
  "id" : 4,  
  "value" : 4.3e+9  
}
```

Simply use **JSON** keyword in control section

We did this in FNS Inconel example

```
<< -----set initial switches and get nuclear  
data----- >>  
  
CLOBBER  
JSON  
GETXS 0  
GETDECAY 0  
FISPACT  
* FNS 5 Minutes Inconel-600
```

## Why JSON?

- Human readable
- Light weight
- Easy to parse
- Very popular format
- Supported by many languages – Python, C++, Fortran, js, ...
- Web friendly (more on this later)
- Full double precision

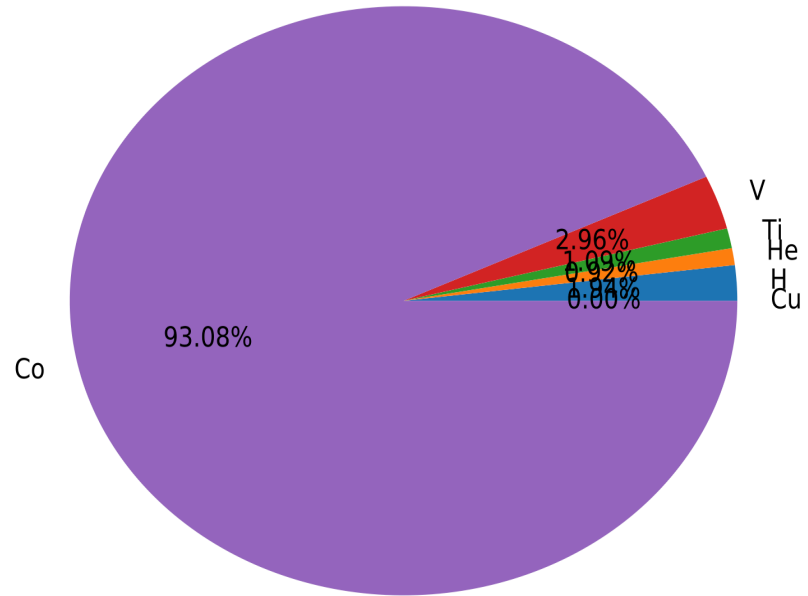
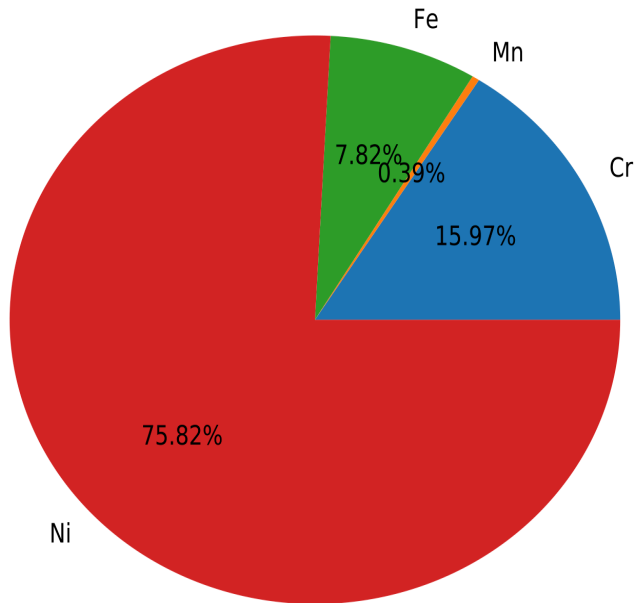
# FISPACT-II JSON

Supports most common output data

- Run data - timestamp, run name, flux name
- Inventory data - list of timesteps, each containing
  - Irradiation and cooling time
  - Flux amplitude
  - Heat - alpha, beta, gamma
  - Ingestion and inhalation
  - Dose
  - **Nuclides** → the inventory (activity, grams, heats,...)

**Exercise JSON shows how to do this**

# Exercise JSON



Left: initial

Right: final (excluding initial)



# Alternative options

Could alternatively have used **TAB1** keyword to get this data  
Contains columns of nuclide with grams only!

Alternatively, we can use `pypact`

Python package to parse fispact output files

Advisable to use JSON - but pypact works on standard output file (\*.out) also

Will work on FISPACT-II files pre version 4.0

Makes scripting easier and analyse FISPACT-II outputs

Easy to install via pip (python package management system)

```
pip3 install pypact
```

First let's do a very simple example with pypact

```
# import the library
import pypact as pp

# read the output file
with pp.Reader('inventory.out') as output:
    print(output.run_data.timestamp)
    print(output.run_data.run_name)
```

## Another more interesting example

```
import pypact as pp
import matplotlib.pyplot as plt

time = []
data = []

with pp.Reader('inventory.out') as output:
    for t in output.inventory_data:
        if not t.isirradiation():
            time.append(t.currenttime)
            data.append(t.gamma_heat)

plt.plot(time, data)
plt.show()
```

Can also use the JSON file

```
import pypact as pp
import matplotlib.pyplot as plt

time = []
data = []

output = pp.Output()
with open('inventory.json') as f:
    output.json_deserialize(f.read())
    for t in output.inventory_data:
        if not t.isirradiation():
            time.append(t.currenttime)
            data.append(t.gamma_heat)

plt.plot(time, data)
plt.show()
```

Use pypact to get all the output times from the output file  
It can be done in 3 lines!!

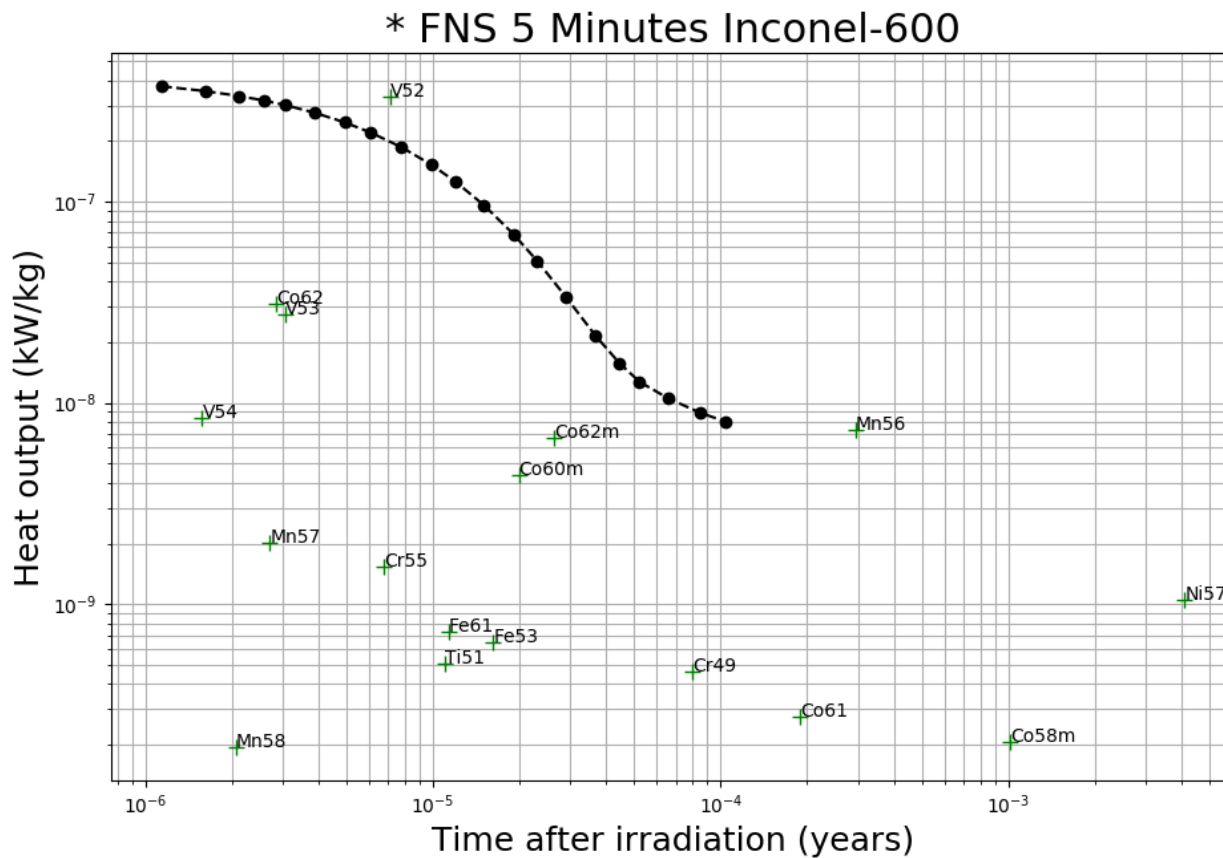
```
import pypact as pp

with pp.Reader('inventory.out') as o:
    [print("Time {} (secs)".format(t.currenttime)) for t in
o.inventory_data]
```

We will do another analysis using pypact to produce a similar plot  
as in FNS Inconel for gnuplot



# Exercise Pypact



# Pypact Summary

## Status

- Handle legacy FISPACT-II output files and JSON files
- Process nuclides, doses, ingestion, burnup, ..
- Create and read files files
- Contains all fispact keywords and group structures built in - 66, 69, ....., 709, 1102
- Simple plotting with analysis extension

## Future

- Pathways and uncertainties
- Gamma spectrum
- Optimizations and API improvements

Feedback welcome!

Engine for running containers...

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it

Can be thought of as a lightweight Virtual Machine.



# FISPACT-II with Docker

We utilise Docker to bundle all dependencies in a single image

Including nuclear data (in binary format) - faster!

To read binary data we use GETXS -1

Docker not installed on your machines, but you can run it with

```
docker run -it -v  
/opt/fispact/bin/ubuntu/16.04/fort18/fispact:/fispact  
fispact/ubuntu:16.04_gfortran_5_data bash
```

# FISPACT-II with Docker

Then we have ubuntu with nuclear data!

```
root@580a028b793c:/# ./fispact
FATAL ERROR: unable to find FILES file
                also unable to find input file
RUN TERMINATED
Either provide the FILES name as a second argument, or
provide a file named 'files', 'Files' or 'FILES'
in the present working directory, or
provide input file .i
in the present working directory.
```

# FISPACT-II Advanced Summary

- Open source tools – in active development
- Many ways to analyse FISPACT-II output
  - GNU plotting built in
  - TAB files can make life easier but difficult hard to script
  - JSON output has many parsers ready - C++, Fortran, Python, js, ...
- Pypact
  - Supports both JSON and standard output file formats
  - Very easy to script plots and analysis
  - Can read and plot flux file
  - Contains tool to create files file
  - Contains all keywords for FISPACT-II
- Docker
  - Many images ready made with nuclear data inside
  - Self contained environment to run fispact easily
  - Nuclear data in binary format – performance improvement