https://doi.org/10.61092/iaea.2sjj-rwa4

INTERNATIONAL ATOMIC ENERGY AGENCY

NUCLEAR DATA SERVICES

DOCUMENTATION SERIES OF THE IAEA NUCLEAR DATA SECTION

IAEA-NDS-0236
December 2020

Tools for TALYS
Autotalys, T6, TENDL, libraries and more

Arjan Koning
International Atomic Energy Agency
P.O. Box 100, A-1400 Vienna
AUSTRIA

Nuclear Data Section
International Atomic Energy Agency
P.O. Box 100
A-1400 Vienna
Austria

E-mail: NDS.Contact-Point@iaea.org
Fax: (43-1)26007
Telephone: (43-1)2600-21725
Web: http://www-nds.iaea.org

Tools for TALYS

Autotalys, T6, TENDL, libraries and more

Arjan Koning

IAEA NDS Document Series IAEA-NDS-236, December 2020

s

NDS.IAEA.ORG/TALYS

Typesetting: The Legrand Orange Book, LaTeX Template, Version 2.1.1 (14/2/16), downloaded from: www.LaTeXTemplates.com. Original author: Mathias Legrand (legrand.mathias@gmail.com) with modifications by: Vel (vel@latextemplates.com). License: CC BY-NC-SA 3.0 creativecommons.org/licenses/by-nc-sa/3.0/.

First printing, December 2020

About the author

Arjan Koning is a nuclear physicist with a Masters Degree in Theoretical physics at the Univ. of Amsterdam, a PhD in the Natural Sciences on Multi-step direct reactions at the Univ. of Groningen, and a professorship at the Univ. of Uppsala on theoretical nuclear reactions.
Arjan is currently Head of the Nuclear Data Section at the IAEA in Vienna. Before that, he has worked at ECN/NRG Petten, the Netherlands, on nuclear reaction data for science and technology, and as guest scientist at CEA/Bruyeres-le-Chatel and Los Alamos National Laboratory on the development of nuclear reaction models. He has led various students to PhD degrees, has coordinated and chaired various international nuclear data projects such as the OECD/NEA JEFF and WPEC projects, and has advised governments and international organisations on nuclear research and development. Among his scientific accomplishments are innovations in nuclear reaction physics, especially the optical model and pre-equilibrium reactions, the TALYS nuclear model code, Total Monte Carlo uncertainty propagation and the TENDL nuclear data library. As of end-2020, his h-index is 52 with more than 14 000 citations.
Although Arjan has finally accepted leadership offers that he could no longer refuse, he aims to keep his scientific creativity alive by maintaining and extending TALYS + all products that emerge from that. Pleas from his friends to also spend time on other things are sometimes honored.

Preface

TOOLS for TALYS is a collection of libraries, scripts and other software packages to steer TALYS and the main codes around it, towards applications that go far beyond these codes individually. In my view, it is important to have well-documented software packages which have a clear task and are well designed and tested, preferably by as many people as possible. This is certainly the case for TALYS, which enjoys a user database of thousands, and also other software, like TASMAN for uncertainty quantification and TEFAL for ENDF-6 formatting are now moving into that direction. Now that the documentation and validation of such codes has been established, it is useful to also document what you can do with a combination of these codes.

The aforementioned codes can produce a huge amount of nuclear data libraries, especially when they are connected together in scripts. This is all explained in the current tutorial. I must say that this tutorial is not only for you, but also for myself. When producing TENDL or other collections of nuclear data libraries, it is important to have the procedure well documented, especially if one revisits certain tasks only once per year or so. This means that the current tutorial contains the order of steps to produce certain libraries such as TENDL. An essential part in all this is, I think, **autotalys**: a bash script which automatically runs TALYS and associated software to easily perform an otherwise complicated task, all the way from fundamental nuclear data to an applied result. Examples are the production of an isotopic ENDF nuclear data file, including uncertainties, the entire TENDL library, but also automatic optimization of nuclear model parameters to experimental data for a whole range of projectile-nuclide combinations, and complete sets of TALYS outputs for intercomparison of models. In addition, other scripts such as **libmaker** produce complete tabulated nuclear data libraries, including plots of these data vs EXFOR, which are of general use, also for non-TALYS or non-TENDL users.

License, contact and reference

As mentioned on the first page, in the source code and detailed in the last Appendix, TOOLS for TALYS falls in the category of GNU General Public License software.

In addition to the GNU GPL *terms* I have a *request*:

- When TOOLS for TALYS is used for your reports, publications, etc., please make a proper reference to the code. At the moment this is:
  When you refer to the application of this software:
  A.J. Koning, D. Rochman, J.-Ch. Sublet, N. Dzysiuk, M. Fleming, and S. van der Marck, TENDL: Complete Nuclear Data Library for innovative Nuclear Science and Technology, Nuclear Data Sheets 155,1 (2019).
  When you refer to something particular of this tutorial:
  A.J. Koning, TOOLS for TALYS, IAEA NDS Document Series IAEA(NDS)-236, December 2020
- Please inform me about, or send, extensions you have built into TOOLS for TALYS. Of course, proper credit will be given to the authors of such extensions in future versions of the code.
- Informing me about your use of TOOLS for TALYS in reports and publications will help me to maintain the TOOLS for TALYS-bibliography.

Although I have invested a lot of effort in the validation of our code, I will not make the mistake to guarantee perfection. Therefore, in exchange for the free use of TOOLS for TALYS: If you find any errors, or in general have any comments, corrections, extensions, questions or advice, I would like to hear about it at **A.Koning@iaea.org**. The webpage for TOOLS for TALYS is **nds.iaea.org/talys**.

Acknowledgements

I wish to thank a few persons who have contributed to the overall system:

- Dimitri Rochman for developing several scripts and databases and testing many of the TENDL files and other results from **autotalys** and T6
- Jean-Christophe Sublet for his feedback on the results and overall structure of T6, especially for the production of TENDL.
- A few beta users of the T6 systems:
  - Henrik Sjoestrand, Erwin Alhassan and Georg Schnabel at Uppsala University,
  - Satoshi Chiba and co-workers at Tokyo Institute of Technology.
  - Shin Okumura at IAEA

Arjan Koning

Contents

1. Introduction

Since the first official release of TALYS [1], several tools have been developed to come to a complete, automated system for the production of nuclear reaction data. Examples of such satellite codes are TASMAN, to generate uncertainty/covariance data via random TALYS input files, and TEFAL which writes the TALYS output into ENDF-6 format. Now that these and other software have been released as individual packages it is timely to document what you can do with a combination of all these codes.

This "Tools for TALYS" tutorial is a cookbook for performing the tasks that have led to many publications on TALYS, TENDL [2] and Total Monte Carlo [3] in recent years. You will be able to do this yourself now. Also, several other codes which have not been released and documented before are outlined here. Central in all this is the **autotalys** command. Analogous to software development tools as 'automake' and 'autoconf' used on Linux systems, **autotalys** automates a whole sequence of nuclear data generation steps into one command. A mimimal example for **autotalys** is

```
autotalys -proj n -element Fe -mass 56
```

while e.g.

```
autotalys -proj n -element Am -mass 241 -high -bins 60 -njoy -residual
-isomer -ntalys 100 -recoil -covar -binsrand 60 -plot -subfission -nomcnp
-tasmanfile /Users/koning/tasman/aux/tasman.tendl2019
-sdefault -s20 -s60 -acf -eaf -mt
```

is the command that produces the TENDL file for neutrons on ^{241}Am, including covariance data.

"Tools for TALYS" and **autotalys** is not only for ENDFista's however. Understandably, many nuclear physicists want to stay away from the ENDF format as far as possible, and we would like to note here that the tools described in this tutorial can also be used for automatic search of nuclear model parameters, production of astrophysical reaction rates, comparisons with EXFOR[4], plotting of your latest TALYS results etc.

Central in our approach is that *all* relevant nuclear data needs to be available, in consistent form and instantaneously. It should be clear that a system which produces TENDL and Total Monte Carlo does not work via click-by-click retrievals from a GUI for one particular reaction channel for one nuclide, while this is how most of nuclear data has been organized in the past decade. This tutorial describes what needs to be set up to make nuclear data evaluation completely reproducible, efficient, systematic and automated.

As specific components and features of "Tools for TALYS" we mention

- the **autotalys** script, with which you can automatically perform complex nuclear data tasks with a simple command.
- ENDFTABLES: a Fortran-95 program to transform an ENDF-6 formatted nuclear data library into x-y data tables with descriptive filenames, so that the evaluated data can be easily used for numerical comparisons, adoption in new libraries, plotting etc.
- *libraries/*: a directory structured database with all major ENDF nuclear data libraries and EXFOR in easy accessible x-y or x-y-dy data files. (needed until a versatile ENDF API is released)
- the T6 software system [2] to produce nuclear data libraries
- collection of scripts to automatically run, and analyze the output of, codes like PREPRO[5], NJOYKahler2010, FUDGE[6], CHECKR, FIZCON[7] etc.
- step-by-step outline of the production of TENDL [2],
- plot scripts for direct comparison of a new evaluation versus existing nuclear data libraries and EXFOR

The entire system described in this tutorial revolves around software for which separate tutorials are available:

- TALYS: a code for the simulation of nuclear reactions, which produces all physical quantities needed in a complete nuclear data library, with the exception of those produced by TARES, TANES and TAFIS (see below).
- TEFAL [2]: a code for producing nuclear data libraries in ENDF format. TEFAL relies on the output of TALYS and processes all the TALYS output files into an ENDF library.
- TASMAN [2]: a code for covariances, optimization, sensitivities and other statistical information for TALYS. TASMAN works on the input and output of TALYS: it creates random input files for TALYS and collects all the output of those random runs to produce covariance matrices. Additionally, TEFAL can be included in that loop for Total Monte Carlo error propagation.
- EXFORTABLES[8]: A code to produce an experimental nuclear reaction database based on EXFOR (needed until a versatile EXFOR API is released)
- RESONANCETABLES[9]: A code to produce a database for thermal cross sections, MACS and average resonance parameters
- TARES[10]: a code by Dimitri Rochman for neutron resonance parameters and its uncertainty quantification
- TAFIS [2]: a code by Dimitri Rochman for the average number of fission neutrons and its uncertainty quantification
- TANES [2]: a code by Dimitri Rochman for prompt fission neutron spectra and its uncertainty quantification

A collection of scripts and other small programs exist to connect the above software, and this will be outlined here.

1.1 This tutorial

This tutorial is set up as follows

Chapter 2: a manual for ENDFTABLES, a tool to transform ENDF-6 libraries into ordinary x-y data tables,
Chapter 3: the *libraries/* system which contains all historic evaluated and experimental data in a consistent form,
Chapter 4: the AUTONORM tool, to normalize TALYS results automatically to evaluated or experimental data,
Chapter 5: The T6 system, including how to create TENDL,
Chapter 6: The **autotalys** script,
Chapter 7: Conclusions and ideas for future work.

2. ENDFTABLES: from ENDF-6 libraries to tables

2.1 Introduction

Historical evaluated nuclear reaction data has generally been stored with the Evaluated Nuclear Data Format (ENDF), in ENDF data libraries or files. There are cynical people who claim the 'E' stands for Encrypted. Indeed it is not only a challenge to get data into the ENDF format, but also to get it out again, especially when compared to modern database formats which are keyword-based (XML, JSON, YAML, etc.). This holds especially for the very complete ENDF-6 files which are made nowadays (which run from from MF1 to MF40). The only released software that I am aware of that produces complete ENDF-6 libraries is EMPEND [11], DECE [12] and TEFAL[2]. There are more codes which can *read* ENDF-6 format, but they are often integrated in all-in-one tools. Examples are ENDVER [13] (directly coupled with ZVVIEW [14]), JANIS [15], FUDGE [6], BNL checking codes [7], PREPRO [5] and NJOY [16]. ENDFTABLES differs from the above in that it performs this task as a stand alone code and is not necessarily integrated with plotting etc. tools. It provides a cut in the middle. I expect that ENDFTABLES will be replaced in the future by an API which can retrieve data on the basis of specific user requests.

The ENDFTABLES code, written in Fortran-95, reads one ENDF formatted data library and produces a large collection of data tables, with one data file per reaction channel, in x-y or x-y-dy format and a small header containing the basic information about the reaction. Hence, ENDFTABLES can be regarded as the inverse of TEFAL, which *produces* data from TALYS and other sources in ENDF format.

2.2 Using ENDFTABLES and installation

ENDFTABLES has, at the moment, no input parameters. It just runs as follows

```
endftables <endf file>
```

for example

```
endftables n-U238.endfb8.0
```

and that is it. Below follow the installation steps.

2.2.1 Installation of ENDFTABLES

ENDFTABLES is a Fortran-95 code which is part of the T6 package. However, it can also be obtained as one tar file *endftables.tar*. After

```
tar zxf endftables.tar
```

there is a directory *endftables/source*. Go into that directory and do

```
gfortran -c *f90
gfortran *.o -o endftables
mv endftables ~/bin
```

assuming you have a *bin/* directory with all your executables.

ENDFTABLES comes with one sample case, in *endftables/sample*. In *sample/org* we have the file n-U238.endfb8.0 and all the output files obtained from running the code on this file. In *sample/new*, you find only the ENDF file so you can test whether you get the same result.

2.3 Output of ENDFTABLES

After successful installation, you may run ENDFTABLES on an ENDF file, e.g. goto *sample/new* and do

```
endftables n-U238.endfb8.0
```

In a few seconds, hundreds to thousands of files are created in your current directory and these are classified below. These files have the so-called MT numbers in their name. Table 2.1 explains the relation between an MT number and a reaction channel.

2.3.1 Cross sections

There is a long list of cross section files

```
n-U238-MT001.endfb8.0
n-U238-MT002.endfb8.0
n-U238-MT004.endfb8.0
n-U238-MT005.endfb8.0
n-U238-MT016.endfb8.0
n-U238-MT017.endfb8.0
n-U238-MT018.endfb8.0
........
n-U238-MT649.endfb8.0
n-U238-MT800.endfb8.0
```

which shows that each reaction channel has its own file, whereby the filename is fully descriptive. Note that this is the output for an original ENDF file. For resonance reactions like total, elastic, fission, capture and exothermic (n,p) and (n,alpha) reactions we provide, in addition to the pointwise data, the data also in three different group structures, of 69, 171 and 700 groups respectively, provided that first a pointwise data library has been made. Hence, if we would have run

```
endftables n-U238.endfb8.0.pendf
```

(obtained after resonance reconstruction) there would be additional output files like e.g.

```
n-U238-MT102-G069.endfb8.0
n-U238-MT102-G171.endfb8.0
n-U238-MT102-G700.endfb8.0
```

Files like *n-U238.endfb8.0* and *n-U238.endfb8.0.pendf* are present in our *libraries* database which will be discussed later.

As an example of a single cross section output file of ENDFTABLES, *n-U238-MT016.endfb8.0* looks as follows

```
# n + U 238 : (n,2n)
# endfb8.0   IAEA Consortium                    EVAL-DEC14
# uncertainties: y
# # energies =    37
#   E(MeV)       xs(mb)       xslow(mb)      xsupp(mb)
 6.17907E+00    0.00000E+00    0.00000E+00    0.00000E+00
 6.30000E+00    5.87750E+00    5.43962E+00    6.31538E+00
 6.40000E+00    3.03702E+01    2.81076E+01    3.26328E+01
 6.50000E+00    7.81910E+01    7.23656E+01    8.40163E+01
 7.00000E+00    5.07335E+02    4.80232E+02    5.34438E+02
...............
```

where we have reproduced as much information as possible from the original data library in the header of the file. Note that if covariance information is available, as in this example, we also provide the 1-sigma uncertainty band.

For low-energy cross sections the decomposition is rather simple, however for high-energy cross sections and quantities like angular distributions, spectra, isomeric cross sections etc, a combination between the various parts of the ENDF-6 data library need to be used to produce physical quantities that can be compared with measurement.

Fission quantities

The general data for fission and average number of fission neutrons are in

```
n-U238-MT452.endfb8.0
n-U238-MT455.endfb8.0
n-U238-MT456.endfb8.0
n-U238-MT458.endfb8.0
```

which contain the total, delayed and prompt nubar, and the average fission energies, respectively. As an example, the file for the total nubar, *n-U238-MT452.endfb8.0*, looks as follows,

```
# n + U 238 : total    nubar
# endfb8.0   IAEA Consortium                    EVAL-DEC14
#
# # energies =    45
#   E(MeV)       nubar
 1.00000E-11   2.44304E+00
 2.53000E-08   2.44304E+00
 5.50000E-03   2.44313E+00
 6.00000E-03   2.44412E+00
.....................
 1.60000E+01   4.74600E+00
 3.00000E+01   6.41400E+00
```

2.3.2 General information and resonance parameters

We store the general information of the evaluation in

`n-U238-MF01-MT451.endfb8.0`

which is merely a direct copy of the MF1/MT451 information section of the data file.

Basic information of the resonance parameters are stored in

`n-U238-MF02-MT151.endfb8.0`

which is merely a direct copy of the MF2/MT151 section of the data file. If covariance data of the resonance parameters exist, also a MF32 file will be available.

2.3.3 Angular distributions

The elastic angular distributions per incident energy are given in

```
.............
n-U238-MT002-Eang003.000.endfb8.0
n-U238-MT002-Eang003.100.endfb8.0
n-U238-MT002-Eang003.400.endfb8.0
n-U238-MT002-Eang003.600.endfb8.0
n-U238-MT002-Eang004.000.endfb8.0
n-U238-MT002-Eang004.250.endfb8.0
.............
```

while the inelastic angular distributions per incident energy, in the case of ENDFB8.0 for the first 39 discrete levels, are given in

```
.............
n-U238-MT051-Eang004.000.endfb8.0
n-U238-MT051-Eang004.250.endfb8.0
.............
n-U238-MT089-Eang030.000.endfb8.0
```

2.3.4 Residual production cross sections

Residual production cross sections are reconstructed from the exclusive channels, at low energies, or from the MF3/MF6/MT5 combination using non-elastic cross sections and residual nuclide yields at higher energies. If we would run

`endftables n-Fe054.tendl.2019`

for neutrons on ^{54}Fe of TENDL-2019, this looks as follows

```
n-Fe054-rp016030.tendl.2019
n-Fe054-rp016031.tendl.2019
n-Fe054-rp016032.tendl.2019
.....
n-Fe054-rp026052.tendl.2019
n-Fe054-rp026052g.tendl.2019
n-Fe054-rp026052m.tendl.2019
n-Fe054-rp026053.tendl.2019
n-Fe054-rp026054.tendl.2019
n-Fe054-rp026055.tendl.2019
```

where e.g. *rp026052* means the residual production of the Z=26, A=52 (^{52}Fe) nuclide. Note that for this residual product also the distinction between ground state *rp026052g* (^{52g}Fe) and isomer *rp026052m* (^{52m}Fe) is available.

2.3.5 Damage cross sections and DPA

For this case, the libraries should first have been processed with NJOY, via the GASPR routine, to a derived PENDF file. Then the various damage quantities are also available, e.g.

```
n-Fe054-MT301-G069.tendl.2019
n-Fe054-MT301-G171.tendl.2019
n-Fe054-MT301-G700.tendl.2019
n-Fe054-MT301.tendl.2019
n-Fe054-MT302-G069.tendl.2019 etc.
.......................
```

where again group-averaged damage cross sections are provided next to the pointwise data.

MT	Reaction	MT	Reaction	MT	Reaction
1	Total	34	(n,nh)	113	(n,t2α)
2	Elastic	35	(n,nd2α)	114	(n,d2α)
3	Non-elastic	36	(n,nt2α)	115	(n,pd)
4	Total (n,n')	37	(n,4n)	116	(n,pt)
5	(n,x)	38	4th-chance (n,f)	117	(n,dα)
11	(n,2nd)	41	(n,2np)	201	(n,xn)
16	(n,2n)	42	(n,3np)	202	(n,xγ)
17	(n,3n)	44	(n,n2p)	203	(n,xp)
18	Total (n,f)	45	(n,npα)	204	(n,xd)
19	1st-chance (n,f)	51-90	(n,n'$_1$) - (n,n'$_{40}$)	205	(n,xt)
20	2nd-chance (n,f)	91	Continuum (n,n')	206	(n,xh)
21	3rd-chance (n,f)	102	(n,γ)	207	(n,xα)
22	(n,nα)	103	(n,p)	600-640	(n,p$_0$) - (n,p$_{40}$)
23	(n,n3α)	104	(n,d)	649	Continuum (n,p)
24	(n,2nα)	105	(n,t)	650-690	(n,d$_0$) - (n,d$_{40}$)
25	(n,3nα)	106	(n,h)	699	Continuum (n,d)
28	(n,np)	107	(n,α)	700-740	(n,t$_0$) - (n,t$_{40}$)
29	(n,n2α)	108	(n,2α)	749	Continuum (n,t)
30	(n,2n2α)	109	(n,3α)	750-790	(n,h$_0$) - (n,h$_{40}$)
32	(n,nd)	111	(n,2p)	799	Continuum (n,h)
33	(n,nt)	112	(n,pα)	800-840	(n,α_0) - (n,α_{40})
				849	Continuum (n,α)

Table 2.1: The ENDF-6 MT numbers and corresponding reaction channels.

3. Libraries

3.1 Introduction

This chapter concerns the **direct** access of all important nuclear data libraries in the world, and the experimental reaction database EXFOR, in a universal tabular format, as simple x-y or x-y-dy data tables per reaction channel. For TALYS, TENDL and Total Monte Carlo development work, but also other future projects that depend on automation like Machine Learning, this is needed. Since it did not exist, I had to build it myself. Nowadays, one would like to have an ENDF API which retrieves nuclear data at command line level for user-specified options, but this is not available yet. The various nuclear data library projects in the world have not agreed on a way to release large data libraries, and certainly not on filename conventions, and at most a tar file with all isotopic files of one nuclear data library can be retrieved (and then again, the individual files are named with a convention different from that of the next library). What is required is a global agreement on what the GND community calls a **protare**, for PROjectile-TARget-Evaluation, which uniquely defines a data library. We note that the current *libraries* described here takes away this complication and can be useful to anyone, not only to TALYS/TENDL users.

Efficient development of nuclear model codes and evaluated nuclear data libraries can only take place when all related nuclear data information generated up to the present moment is available in an easy accessible way. For nuclear data this means the direct access to all evaluated nuclear data libraries of interest and the EXFOR database of experimental data. This has already been recognized by many nuclear data developers, and consequently various software packages exist which can read an ENDF formatted nuclear data library and, often below the surface of a graphical user interface, find the associated experimental data after which the data can be plotted. Examples of such ENDF + EXFOR plotting and formatting packages are ZVVIEW[14], JANIS[15], TEFAL[2], EMPEND[11], ENDVER[13] and of course also checking + processing software such as CHECKR, FIZCON, PREPRO, FUDGE and NJOY can interpret full ENDF data libraries. The problem is that a nuclear data user is always restricted to the capabilities of these codes and what the author chose to provide as output. And surely we want to do more with available data than plotting! We

now provide an alternative route to the use of ENDF and EXFOR data: We first decompose these libraries into x-y or x-y-dy data tables (files) per reaction channel, and store the results in a large database with a logical directory and filename convention. Next we, and in fact any user with or without deep nuclear data formatting knowledge, can start using nuclear data from that new starting point, without having to do the ENDF-6 format interpretation first.

There are various reasons for doing this, both for TALYS and TENDL development but also for general use by others:

- In general, it provides a much needed cut in the middle. The entire process of reading and interpreting an ENDF-6 data library has been done in a separate step. The resulting database provides a fresh starting point for nuclear physicists who do not want to be bothered with the ENDF-6 format, and I think this is the majority. All aforementioned packages have specific objectives and often do not do all you want with the contents of an ENDF-6 library.
- The *libraries/* database provides a new starting point for plotting, without the overhead and limitations of interpreting the ENDF or EXFOR format of a library first. Hence, a much larger pool of graphical software developers can potentially be reached now that barrier is removed. It should now be much easier to plot for e.g. one reaction channel all existing experimental data sets and nuclear data libraries, or to easily plot e.g. the neutron capture cross sections for all Fe isotopes in one plot. There is always something an all-in-one plotting package does not provide and having all data in handy tables provides much more flexibility.
- Having all nuclear reaction results in x-y data tables makes it also easy to adopt the data of certain well-evaluated reaction channels into *new* nuclear data libraries in ENDF-6 formatting software as included in TEFAL. The TENDL concept relies heavily on this easy availability.
- The TASMAN code can automatically optimize nuclear model parameters to evaluated and experimental data. For this, the libraries structure is essential. This also holds for users outside the TALYS community.
- It is more efficient to perform statistical analyses like validation of the entire EXFOR data library, or any other study where results from EXFOR and nuclear data libraries are needed, or nuclear data libraries are intercompared.

The whole 'libraries' project heavily relies on the ENDFTABLES code, which reads an ENDF-6 formatted library and produces a complete output, for all reactions. Ideally a more specific API, probably written in Python or something similar, should be made which extracts exactly a user-defined query from a data library (e.g. the (n,2n) cross section), so that not everything needs to be extracted. Putting that module into a loop over all reactions would then produce the same as ENDFTABLES. At the moment, ENDFTABLES is doing the job for us.

3.2 The 'mother' database

The nuclear data libraries which have been processed are

```
n-cendl3.1
n-eaf.2010
n-endfb8.0
n-iaea.med
n-irdff2.0
n-jeff3.3
n-jendl.2007.he
n-jendl.ad
n-jendl4.0
n-jendl4.0.he
n-tendl.2019
```

```
p-endfb8.0
p-iaea
p-iaea.med
p-ibandl
p-jendl.2007.he
p-tendl.2019
g-endfb8.0
g-iaea
g-iaea.med
g-jendl.2016
g-tendl.2019
d-endfb8.0
d-iaea
d-iaea.med
d-ibandl
d-tendl.2019
a-iaea
a-iaea.med
a-ibandl
a-jendl.2005
a-tendl.2019
h-endfb8.0
h-iaea
h-iaea.med
h-tendl.2019
t-endfb8.0
t-tendl.2019
```

where we hope that the combination of projectile and library specifies the contents clearly enough. Quite a bit of effort was needed to unify the contents of each of these libraries after their retrieval from the nuclear data centers, i.e. to make the filenames consistent. For example the n-endfb8.0/ directory looks as follows,

```
n-Ac225.endfb8.0.gz
n-Ac226.endfb8.0.gz
n-Ac227.endfb8.0.gz
n-Ag107.endfb8.0.gz
n-Ag108.endfb8.0.gz
.................
```

This convention is followed in all 'mother' libraries. More interesting are the data files derived from this and that is discussed below.

3.3 Data library and filename conventions

We use one consistent filename convention throughout the entire TALYS system. The base of every filename should be the projectile-target combination. Projectile is essential, since though many users may only be interested in neutrons (e.g. resonance data), the applications involving the other particles are rapidly growing.

The basic convention for a projectile-target combination is projectile-ElementMass[possible isomer].extension, where

- projectile is one of n, g, p, d, t, h, or a.
- Element is one or two characters with the first character in uppercase. Up to Z=118 this is officially defined, while we use B9 for Z=119, C0 for Z=120 up to C9 for Z=129, D0 for Z=130, etc. (TALYS is restricted to a maximum of Z=124 (C4) at the moment)
- Mass is in the i3.3 (Fortran) format, i.e. has leading zeroes.
- Isomer is m, n, o, p, etc. and directly follows the mass number.

Hence, the part of the filename which designates the projectile-target combination looks for example as follows: n-Nb093, p-F019, n-Am242m, a-Be009.

To the above, the extension can be added, for example to designate the particular evaluation file of a library, which is the aforementioned 'protare'. An example is n-Nb093.endfb8.0. In this way all isotopic nuclear data files from the world are uniquely defined. In *libraries* they are stored in the appropriate subdirectories. As you will see below, derived data files from these 'protare's' will get further consistent extensions.

3.4 Processing the mother database

A Fortran-95 software package, ENDFTABLES has been written which processes an ENDF-6 formatted nuclear data library into tabular format. A full description of ENDFTABLES is available in Chapter 2. ENDFTABLES produces files with a x-y or x-y-dy structure (if covariance matrices are available). Hence, for this database all isotopic nuclear data files from the major world libraries can be completely dismantled using ENDFTABLES and put into single files per reaction channel.

3.4.1 Libmaker

A *libmaker* bash script has been made to loop ENDFTABLES over all nuclear data libraries and to store all results in appropriate subdirectories.

For example, the command

```
libmaker Cu 065
```

will do this for all n, g, p, etc libraries of Cu-65 in the mother database. A few variants are possible, e.g.

```
libmaker Cu 065 p
```

makes tables for ^{65}Cu for incident protons only, and

```
libmaker Am 242m n tendl.2019
```

only makes tables for the neutron file for ^{242m}Am of TENDL-2019, and nothing else.

For each individual data file in the mother database, *libmaker* will make sure that ENDFTABLES is executed, e.g. the first *libmaker* command above will run *endftables n-Cu065.endfb8.0*, *endftables n-Cu065.jeff3.3*, etc. This produces a huge amount of ASCII x-y tables for cross sections, angular distributions etc. for this evaluated file, and in the process also various checks, with BNL checking codes, and processing with PREPRO and (optional) NJOY are performed on the mother library. Next *libmaker* stores all these results in appropriate subdirectories. The script also copies the experimental data from the *exfortables/* database (x-y-dy structured version of EXFOR) into *libraries/*. In addition the *libmaker* script is also run for all natural isotopes, using the NATURALTABLES code, described below, which averages all isotopic tables into tables for the natural target element. Even though none of the original nuclear data libraries contain any reaction data for natural targets, this is important for e.g. comparison with experimental data for natural elements in EXFOR. Finally, when all data from the nuclear data libraries and EXFOR are in place, *libmaker* runs the *plotall*

plotting script, described later, to produce plots of cross sections of the nuclear data libraries vs. EXFOR.

The user has to make the sure that the appropriate paths for the mother data libraries and EXFOR are set in the script. The entire data structure which *libmaker* produces is discussed below.

3.4.2 Naturaltables

3.4.3 loopnuc

A *loopnuc* bash script has been made to prepare scripts for the production of *libraries*, TENDL, etc. for all nuclides. As the name suggests *loopnuc* reads in a file e.g. 'nuclides' and then prepares a script for each nuclide in that list. This can be done to run *libmaker* for each nuclide but also to produce the resonance data of *resbase/* or to produce the collection of scripts that create the entire TENDL library.

3.5 Library structure

The entire library contains about 200 Gb of data. However, for practical use we often disseminate heavily trimmed versions which are e.g. enough to use in T6 to produce the TENDL library. Locally, we often have a *libraries.all/* database containing the full 200 Gb of data, on which we run the *libfilter* script, described later, to produce a much smaller database. The description below is for the full version.

3.5.1 Particles

The highest structure level contains the incident particles. Hence, under *libraries/* we find

```
n/
g/
p/
d/
t/
h/
a/
```

3.5.2 Nuclides

The next level contains the nuclides. This deviates from usual directory structures of various nuclear data libraries, where (obviously) the nuclides are stored under a certain nuclear data library, since each file project only disseminates its own list of isotopic evaluations (The 'mother' database that *libraries/* was constructed from has this structure.) Here, we however store all information that is available under the nuclide, since that is usually what we want when we are going to 'work on a nuclide'. Hence, we have

```
....
n/Fe054/
n/Fe055/
n/Fe056/
etc.
```

After this, the next level contains the nuclear data libraries, EXFOR and plots. For example, for neutrons on ^{54}Fe we have

```
n/Fe054/cendl3.1/
n/Fe054/eaf.2010/
n/Fe054/endfb8.0/
n/Fe054/exfor/
n/Fe054/irdff2.0/
n/Fe054/jeff3.3/
n/Fe054/jendl.2007.he/
n/Fe054/jendl4.0/
n/Fe054/jendl4.0.he/
n/Fe054/plots/
n/Fe054/tendl.2019/
```

At this level, there are three different types of subdirectories, each with their own structure, so they are discussed separately.

3.5.3 EXFOR

All EXFOR data that is available in computational XC4 format is stored in the *exfor/* directory. There, different quantities such as cross sections, ratio's, angular distributions etc. are stored in subdirectories as well. This directory comes straight from another software package, EXFORTABLES [8], which translates the entire EXFOR database in a more descriptive directory structure, and does goodness-of-fit comparisons with world libraries, outlier detection and statistics on EXFOR data in the process. For neutrons on ^{54}Fe, the subdirectories of *exfor/* look as follows

```
n/Fe054/exfor/angle/
n/Fe054/exfor/ddx/
n/Fe054/exfor/ratio/
n/Fe054/exfor/residual/
n/Fe054/exfor/resonance/
n/Fe054/exfor/spectrum/
n/Fe054/exfor/xs/
```

while e.g. the cross section directory *xs/* is subdivided into MT numbers, i.e.

```
n/Fe054/exfor/xs/001/
....
n/Fe054/exfor/xs/102/
n/Fe054/exfor/xs/103/
etc.
```

and zooming in on e.g. the (n,p) cross sections in directory *n/Fe054/exfor/xs/103/* we find the following files

```
n-Fe054-MT103-Allan-20961010.1959
n-Fe054-MT103-Artemev-41321006.1980
n-Fe054-MT103-Bahal-21936015.1985
.....
n-Fe054-MT103-VanLoef-30221006.1961
n-Fe054-MT103-VenugopalaRao-11722003.1967
n-Fe054-MT103-Viennot-30644005.1982
n-Fe054-MT103-Viennot-30978021.1991
```

where the filename contains projectile, target, reaction channel (MT number), first author name, EXFOR subentry number and year of publication. while e.g. *n-Fe054-MT103-VanLoef-30221006.1961* looks as follows

```
# Target Z    :  26
# Target A    :  54
# Target state:
# Projectile  : n
# Reaction    : (n,p)
# Final state :
# Quantity    : Cross section
# Frame       : L
# MF          :   3
# MT          :  103
# X4 ID       : 30221006
# X4 code     : 26-FE-54(N,P)25-MN-54,,SIG
# Author      : VanLoef
# Year        : 1961
# Data points :      3
#    E(MeV)       xs(mb)      dxs(mb)     dE(MeV)
  2.60000E+00  1.50000E+02  2.00000E+01  2.00000E-01
  3.30000E+00  1.90000E+02  2.00000E+01  1.00000E-01
  3.60000E+00  2.10000E+02  1.50000E+01  1.00000E-01
# Full reference:
#J.J.Van Loef
#Jour. Nuclear Physics Vol.24, p.340, 1961
#Activation cross sections for (n,p) reactions in some medium weight ...
```

The capabilities of EXFORTABLES are not yet complete: while quantities like cross sections, nubar, angular distributions etc. are now translated into tables, other quantities like spectra, DDX, etc. are not yet translated. An extensive tutorial of EXFORTABLES exists[8].

3.5.4 Nuclear data libraries

Most data resides in the world's evaluated nuclear data libraries. We have stored as much information as possible of a nuclear data library. There are generally 3 subdirectories. As an example, for the JENDL-4.0 neutron data file of ^{54}Fe we have

```
n/Fe054/jendl4.0/check/
n/Fe054/jendl4.0/files/
n/Fe054/jendl4.0/tables/
```

All information below is obtained from the ENDFTABLES code, and with the *libmaker* script this is sorted into the various subdirectories.

Tables

This directory contains the very reason for creating the entire *libraries/* structure. It has all evaluated data tabulated per reaction channel in simple tables. The directory structure below *tables/* is, for the JENDL example,

```
n/Fe054/jendl4.0/tables/angle/
```

```
n/Fe054/jendl4.0/tables/damage/
n/Fe054/jendl4.0/tables/info/
n/Fe054/jendl4.0/tables/resonance/
n/Fe054/jendl4.0/tables/xs/
```

Cross sections

If we take the *xs/* directory as the first example, we see the subdirectory looks as follows

```
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT001-G069.jendl4.0
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT001-G171.jendl4.0
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT001-G700.jendl4.0
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT001.jendl4.0
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT002-G069.jendl4.0
.............
n/Fe054/jendl4.0/tables/xs/n-Fe054-MT207.jendl4.0
```

which contains the files that have already been discussed in the outline of ENDFTABLES.

Angular distributions

The elastic angular distributions per incident energy are given in

```
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT002-Eang000.001.jendl4.0
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT002-Eang000.010.jendl4.0
.....
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT002-Eang018.000.jendl4.0
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT002-Eang020.000.jendl4.0
```

while the inelastic angular distributions per incident energy, in the case of JENDL4.0 for the first 19 discrete levels, are given in

```
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT051-Eang001.435.jendl4.0
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT051-Eang002.000.jendl4.0
.....
n/Fe054/jendl4.0/tables/angle/n-Fe054-MT069-Eang020.000.jendl4.0
```

General info

We store the general information of the evaluation in

```
n/Fe054/jendl4.0/tables/info/n-Fe054-MF01-MT451.jendl4.0
```

Resonance parameters

For neutrons on ^{54}Fe for JENDL4.0, this looks as follows

```
n/Fe054/jendl4.0/tables/resonance/n-Fe054-MF02-MT151.jendl4.0
```

For TENDL-2019 also the covariance data in the resonance range are available, so in that case the directory is

```
n/Fe054/tendl.2019/tables/resonance/n-Fe054-MF02-MT151.tendl.2019
n/Fe054/tendl.2019/tables/resonance/n-Fe054-MF32-MT151.tendl.2019
```

We have not (yet) decomposed the resonance information into human readable format.

Residual production cross sections

In the *residual/* directory the residual production cross sections are stored. For neutrons on ^{54}Fe for TENDL-2019, this looks as follows

```
n/Fe054/tendl.2019/tables/residual/n-Fe054-rp016030.tendl.2019
n/Fe054/tendl.2019/tables/residual/n-Fe054-rp016031.tendl.2019
.....
n/Fe054/tendl.2019/tables/residual/n-Fe054-rp026055.tendl.2019
```

Damage cross sections and DPA

Since all libraries have been processed with NJOY, damage quantities are available in the *damage/* directory,

```
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT301-G069.jendl4.0
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT301-G171.jendl4.0
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT301-G700.jendl4.0
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT301.jendl4.0
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT302-G069.jendl4.0 etc.
.........................
n/Fe054/jendl4.0/tables/damage/n-Fe054-MT447.jendl4.0
```

Fission quantities

In the *fission/* subdirectory, the average fission quantities are stored, e.g.

```
n/Pu240/endfb8.0/tables/fission/n-Pu240-MT452.endfb8.0
n/Pu240/endfb8.0/tables/fission/n-Pu240-MT455.endfb8.0
n/Pu240/endfb8.0/tables/fission/n-Pu240-MT456.endfb8.0
n/Pu240/endfb8.0/tables/fission/n-Pu240-MT458.endfb8.0
```

Files

The *files/* subdirectory contains the original ENDF-6 library as well as three processed files, a PENDF file (with pointwise data) a GENDF file (with groupwise data) and an ACE+xsdir file (for use in MCNP). These files are produced by PREPRO and NJOY respectively. For our JENDL example, this n/Fe054/jendl4.0/files/ directory looks as follows

```
n/Fe054/jendl4.0/files/n-Fe054.jendl4.0
n/Fe054/jendl4.0/files/n-Fe054.jendl4.0.ace.gz
n/Fe054/jendl4.0/files/n-Fe054.jendl4.0.gendf.gz
n/Fe054/jendl4.0/files/n-Fe054.jendl4.0.pendf.gz
n/Fe054/jendl4.0/files/n-Fe054.jendl4.0.xsdir
```

Check

The *check/* subdirectory contains the diagnosis from checking and processing codes (BNL codes, PREPRO, NJOY), and plots from those processing codes. It also contains files with integral quantities, such as that delivered by the INTER code[7], which can be compared with compilations of average resonance data. It also contains the input files used for these codes, to generate the processed data.

3.5.5 Plots

The *plots/* directory contains plots of differential data of all world data libraries and EXFOR, if available. Clearly, these plots are only as good as an automated plotting procedure can provide. A future project may be to establish, and save, plotting parameters such as minimum and maximum

energy and cross section, and placement of legenda per reaction channel. In other words, sometimes the data may interfere with the legenda, and not always a beautiful plot is obtained.

For neutrons on ^{54}Fe, this looks as follows

```
n/Fe054/plots/n-Fe054-MT001.eps
n/Fe054/plots/n-Fe054-MT001.plt
n/Fe054/plots/n-Fe054-MT001.png
n/Fe054/plots/n-Fe054-MT001-lin.eps
n/Fe054/plots/n-Fe054-MT001-lin.plt
n/Fe054/plots/n-Fe054-MT001-lin.png
n/Fe054/plots/n-Fe054-MT001-low.eps
n/Fe054/plots/n-Fe054-MT001-low.plt
n/Fe054/plots/n-Fe054-MT001-low.png
n/Fe054/plots/n-Fe054-MT002.eps
n/Fe054/plots/n-Fe054-MT103.eps
n/Fe054/plots/n-Fe054-MT103.plt
n/Fe054/plots/n-Fe054-MT103.png
.................
n/Fe054/plots/n-Fe054.pdf
```

i.e. we provide one encapsulated postscript plot per reaction channel, although for the neutron total and capture cross sections two extra zoomed-in plots are produced. For completeness, also the .png files and the Gnuplot style files .plt are provided. The file *n-Fe054.pdf* is the entire collection of plots for this projectile-target combination in one pdf file. Some examples of these plots are given in Figs. 3.1. There are about 15 000 plots like this, of which close to 7000 for incident neutrons.

3.6 libfilter

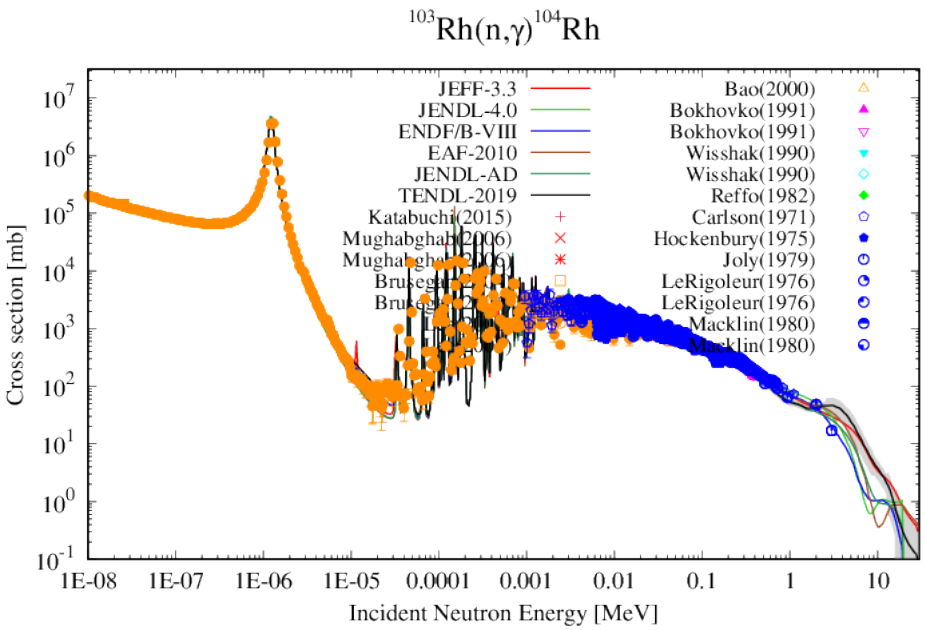
In the process of analyzing and decomposing the data, we also process the files with PREPRO and NJOY into PENDF and ACE libraries and the resulting libraries directory is huge. Therefore, for more practical applications, e.g. direct use for TENDL production in T6, a *libfilter* script has been made, which produces a smaller *libraries/* database with several types of data and data libraries left out. An example of the use of *libfilter* is

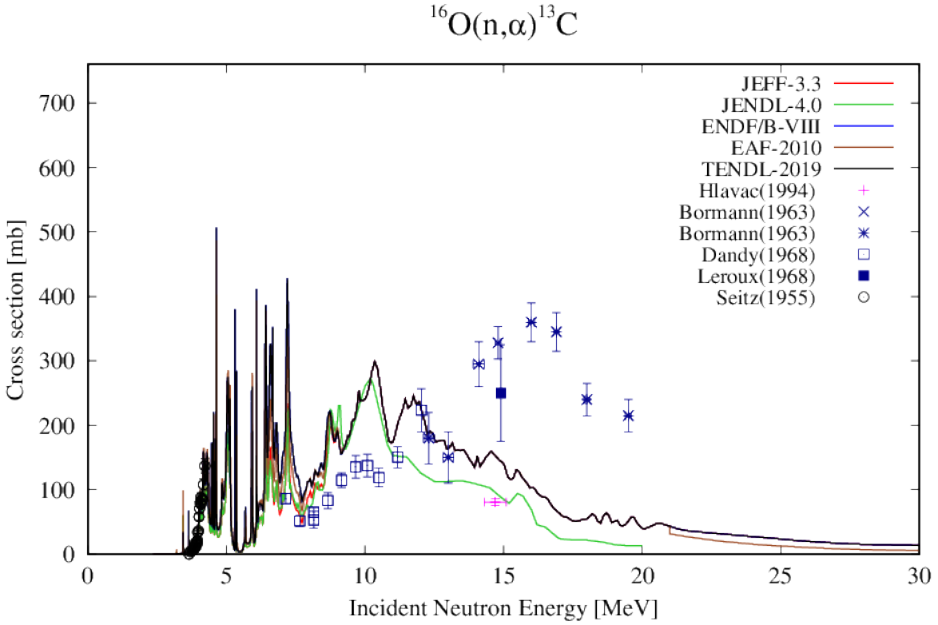
```
libfilter -year -noeaf
```

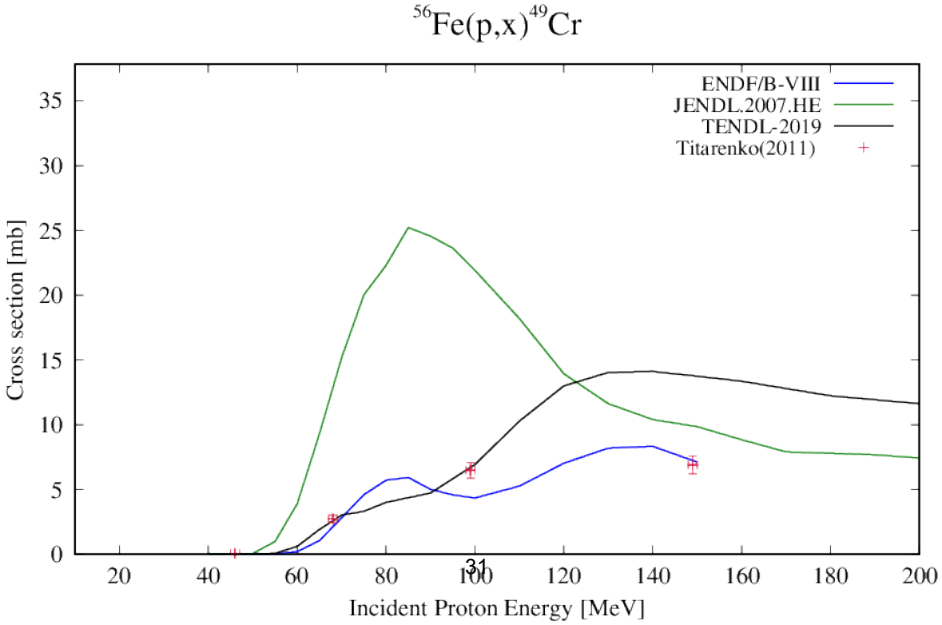
which deletes all data for nuclides with a half life below one year and that of the EAF library. The *libfilter* command to go from the full *libraries.all* to the *libraries* that is needed for T6 production the following command was used

```
libfilter -year -nogendf -noace -nocendl -nojendlhe -noeaf -noibandl
          -tendl -nocheck -residual -noangle -nodamage -nat
```

since for TENDL we do not need to adopt data from the nuclear data libraries that have been left out, and we also do not adopt angular, damage etc information from other libraries. For most users, this smaller *libraries/* database is enough, but of course special *libraries/* packages can be made on request out of the full *libraries.all/* database. A full description of all the options of *libfilter* is given in the appendix.


103Rh(n,γ)104Rh
JEFF-3.3
JENDL-4.0
ENDF/B-VIII
EAF-2010
JENDL-AD
TENDL-2019
Katabuchi(2015)
Mughabghab(2006)
Mughabghab(2006)
Bruseg
Bruseg
Bao(2000)
Bokhovko(1991)
Bokhovko(1991)
Wisshak(1990)
Wisshak(1990)
Reffo(1982)
Carlson(1971)
Hockenbury(1975)
Joly(1979)
LeRigoleur(1976)
LeRigoleur(1976)
Macklin(1980)
Macklin(1980)
Cross section [mb]
Incident Neutron Energy [MeV]


16O(n,α)13C
JEFF-3.3
JENDL-4.0
ENDF/B-VIII
EAF-2010
TENDL-2019
Hlavac(1994)
Bormann(1963)
Bormann(1963)
Dandy(1968)
Leroux(1968)
Seitz(1955)
Cross section [mb]
Incident Neutron Energy [MeV]


56Fe(p,x)49Cr
ENDF/B-VIII
JENDL.2007.HE
TENDL-2019
Titarenko(2011)
Cross section [mb]
Incident Proton Energy [MeV]

3.7 Databases for resonance parameters, PFNS and nubar

T6 contains the codes TARES[10] (resonance parameters), TANES[2] (PFNS) and TAFIS[2] (nubar) which can be run on the fly to produce its output. An alternative approach, now used as the default, is to first make an entire database of these parameters and then use them as static data in an analysis. These database are called *resbase/, fnsbase/* and *nubarbase/* respectively and are also stored in *libraries/*. This avoids the issue of the machine portability of TARES, TANES and TAFIS.

3.8 Creating libraries

For completeness, this section contains an outline on how the entire *libraries* database is made.

- the tabulated EXFOR database needs to exist in the *exfortables/* directory and the path to this directory needs to be set in *libmaker*.
- All results will be stored in a directory *libraries.all/*
- Create a working directory where we are going to store and run all the *libmaker* scripts, in my case e.g. *tendl/libraries/*
- do

  ```
  ~/tools/tendl/loopnuc libraries > lib.all
  ```

  This will create 2800+ scripts, one per isotope, in which libmaker is executed. All scripts are listed in *lib.all*, which I split with *split* into 26 batch scripts, one per leading character of the nuclear symbol, which can all be run (I have 18 processors on my machine).
- After about 16 hours *libraries.all/* is filled with all data
- If not yet present, I add *resbase/, fnsbase/* and *nubarbase/* to *libraries.all/*. To create these make a directory e.g. *tendl/tares* and do

  ```
  ~/tendl/tools/loopnuc tares > n.all
  ```

  and similar for TANES and TAFIS. This will create *resbase/* (or *fnsbase/* and *nubarbase*) in the current directory, which can be copied into *libraries.all*.
- Update the medical isotope data library *isotopia.libs* with its local scripts (this taps data from *libraries.all*), which is a separate project needed for the ISOTOPIA code]citeISOTOPIA.
- go into *libraries.all/* and tar the directories for nuclides with a half life shorter than a day by giving *libday tar*.
- still in *libraries.all/* do *plotlibs* to copy all the plots from the database into one *plots/* library
- go into *plots/* and do *plotcat* to make one pdf file per reaction type, e.g. *n-MT102.pdf* for all plots of capture cross sections
- go into *stat/* and run *globallibCE* and *globallibplot* to produce C/E tables and plots of all world libraries compared with average resonance data and MACS.
- Create the 'final' directory, e.g. *libraries/*, which will be the trimmed down version of *libraries.all/* and go into this new directory. Run the script *lib2t6*: this is basically the execution of *libfilter*, and the current command for this is

  ```
  libfilter -year -nogendf -noace -nocendl -nojendlhe -noeaf -noibandl
            -tendl -nocheck -residual -noangle -nodamage -nat
  ```

  Also the *resbase/, fnsbase/, nubarbase/, plots/,* and *stat/* directories are copied into *libraries*.
- tar all libraries with the *tarlibs* script, this stores all tar files in the directory *libraries.tar* for easy dissemination

3.9 Summary

A *libraries/* database has been made which contains all world nuclear data libraries and the EXFOR database in convenient tabulated x-y-dy format using consistent filenames. This has various advantages for applied use:

- Easy plotting: once this decomposition step is done you "only" need a plotting script that takes the relevant data as starting point without having to do the ENDF-6 decomposition first. ENDFTABLES and *libmaker* did that for you.
- Easy adoption (something which TENDL evaluators call "autonorming") of a well evaluated, or experimental, reaction channel from other data libraries (like IRDFF-2.0) in TALYS or TENDL, if we feel TALYS by itself can not do better than the evaluated data that already exists.
- Statistical analyses like validating the entire EXFOR data library, Bayesian inference for uncertainty propagation etc.

We are not yet at the stage that we can decompose any ENDF-6 file, including interpreting, and implementing in ENDFTABLES, all specific procedures that evaluators have chosen to use. Also double-differential cross sections, PFNS, and gamma-ray cross sections from existing libraries have not yet been processed.

On request you can obtain a series of gzipped tar files named libraries*.tar. If you untar the whole collection for neutrons, protons etc. you will end up with the directory *libraries/*.

4. AUTONORM: Normalizing TALYS calculations

4.1 Introduction

Ideally, TALYS would be so good that one only needs to make a TALYS input file with the right model parameters after which the perfect data library comes out at the end. For TENDL, this is at least attempted for many nuclides; a "best" TALYS input file is stored in our system, which contains improved (relative to the default) optical model, level density, pre-equilibrium, fission, discrete level etc. parameters which bring the TALYS results closer to experimental data than a default calculation would.

But TALYS is not *that* good. Often, we would like to include a particular evaluation for one or a few reaction channels, especially from the neutron standards library or IRDFF which contains several reaction channels which are so well evaluated that their performance can not easily be matched a model code. There are 3 ways to include such high quality data in a new data library. One is to introduce so much parameter freedom into TALYS, such as incident energy dependent variation of nuclear model parameters that TALYS will always fit the target data, no matter the inconsistency of the physics. A second option is to include the cross sections of one or a few reaction channels into the new evaluation at the ENDF level. Actually, TEFAL allows for this. The problem is that as soon as an ENDF (TENDL) file is not completely produced with TALYS results, one may run into trouble with non-matching energy grids and violation of sum rules, which means one should correct for the externally included reaction channel in some other reaction channel. Also, experienced evaluators know that combining different energy grids into one ENDF-6 evaluation is a nightmare.

The third option uses AUTONORM. For this option TALYS is run twice. The first TALYS run proceeds as normal. Next you decide which reaction channels you want to normalize exactly to e.g. standards, IRDFF or ENDFB/VIII evaluations, possibly in a restricted energy range. Then AUTONORM will calculate the ratio between the TALYS cross sections and the evaluation to be adopted and store the results in so-called 'rescue' files (the naming comes from the idea that we have no other option than to do something this drastic) After that, TALYS is run a second time,

but this time the TALYS input file has a few extra lines, containing the names of the rescue files in combination with the **rescue** keyword. Internally in TALYS, several reaction channels are then normalized using the energy-dependent ratios and the output files of that second run then produce 'perfect' results.

4.2 AUTONORM in practice

In the input file for AUTONORM, the reaction channels for which normalization to other libraries needs to take place are given. Next, the procedure is the following:

- Run TALYS. This will produce for each reaction channel c, and incident energy E, the original TALYS cross section $\sigma_c^T(E)$.
- We create a specific input file for AUTONORM for the normalization of the TALYS results to the cross sections of the nuclear data library, e.g. IRDFF-2.0 for some channels and ENDF/B-VIII for some other channels, in a certain energy range that we want to restrict the normalization to. This defines for a few channels the library cross sections $\sigma_c^L(E)$ Then, AUTONORM reads both the TALYS and the library cross sections and produces cross section ratios $R_c(E) = \sigma_c^L(E)/\sigma_c^T(E)$ and stores these in a so-called 'rescue' file, i.e. a file to be used when anything else fails to get the perfect TALYS result.
- Run TALYS for a second time, but now using 'rescue' keywords which make TALYS read in the rescue files with normalization factors. Hence, all channels for which normalization takes place now have their own 'rescue' keyword in the second TALYS input file. TALYS then multiplies the 'original' TALYS cross sections, $\sigma_c^{Tnew}(E) = R_c(E)\sigma_c^T(E)$, so that after the second TALYS run, the results for the specified channels will be exactly equal to that of the nuclear data library to which it was normalized. The difference $\sigma_c^{Tnew}(E) - \sigma_c^T(E)$ is added to, or subtracted from, the elastic cross section, which also means that this approach becomes dangerous if the original TALYS results are too far from the data one wants to normalize to. In a future version of AUTONORM, this redistribution will be made according to the cross-channel covariance matrix that is available from our nuclear models. This is physically better justified and also less risky than accounting for the difference in the elastic cross section.

4.3 Input description

For the communication between AUTONORM and its users, we use the same method as for TALYS. An input file of AUTONORM consists of keywords and their associated values, and you may consult the TALYS tutorial about the rules for giving this input. An example of an input file is

```
projectile n
element V
mass 051
library irdff2.0
norm mt=2 width=0.05 lib=jendl4.0 emin=0. emax=10. ebeg=0. eend=8.
norm mt=107 width=0.05 emin=0. emax=30. ebeg=0. eend=27.
```

You can run AUTONORM by e.g. **autonorm** < **autonorm.inp**, and this should be done in the working directory with all the TALYS output files.

4.3.1 AUTONORM keywords

projectile

Seven different symbols can be given as **projectile**, namely **n, p, d, t, h, a, g** representing neutron, proton, deuteron, triton, ^{3}He, alpha and gamma, respectively.

Examples:

projectile n
projectile d

Range: **projectile** must be equal to **n, p, d, t, h, a, g**.
Default: None.

element

Either the nuclear symbol or the charge number Z of the target nucleus can be given. Possible values for **element** range from Li (3) to C4 (124). To accommodate target nuclides with $Z > 110$ the element names are defined as follows: Rg(111), Cn(112), Nh(113), Fl(114), Mc(115), Lv(116), Ts(117), Og(118), B9(119), C0-4(120-124). Obviously the symbols for Z above 118 will be changed as soon as official names are assigned to them.
Examples:

element pu
element 41
element V
element B9

Range: **3** $\leq$ **element** $\leq$ **124** or **Li** $\leq$ **element** $\leq$ **C4**.
Default: None.

mass

The target mass number A.
Examples:

mass 239

Range: **5** $<$ **mass** $\leq$ **339**.
Default: None.

isomer

The target isomer.
Examples:

isomer m

Range: **isomer** should be equal to **m** or **n**.
Default: None.

library

The nuclear data library to be normalized to.
Examples:

library irdff2.0

Range: **library** should be available in the *libraries* database.
Default: None.

hfnorm

Flag to produce files for the normalization of the transmission coefficients in the Hauser-Feshbach calculation.
Examples:

hfnorm y
hfnorm n

Range: **y** or **n**
Default: **hfnorm n**

iterate

Flag for iteration. Only active if **hfnorm y**.
Examples:

iterate y
iterate n

Range: **y** or **n**
Default: **iterate n**

norm

Normalization to be carried out.
Examples:

- norm mt=2 width=0.05 lib=jendl4.0 emin=0. emax=10. ebeg=0. eend=8.
- norm mt=107 width=0.05 emin=0. emax=30. ebeg=0. eend=27.

Range:
Default: none

5. The T6 system

5.1 Introduction

This chapter contains:

- a description of the various tools which are needed for the production of TENDL and other nuclear data libraries and analyses that can be produced and performed with T6, the software system built around TALYS.
- a strategy for TENDL production,
- a description of the production of the TENDL library, perhaps the best known output of the system, but also other outputs of T6 such as sensitivity tables, tables for EXFOR comparison, astrophysical reaction rates, parameter optimization etc.
- a listing of problems encountered in TENDL, which should be worked on.

This Chapter should cover what is not yet included in other tutorials. Hence, for TALYS, TASMAN, TEFAL, TARES, TAFIS, TANES, and EXFORTABLES complete tutorials exist. The most important tool, **autotalys**, will be discussed in the next Chapter.

5.2 The T6 software system

T6 is named after the 6 core codes which are needed to produce a complete nuclear data library:

- TALYS[1]: a code for the simulation of nuclear reactions, which produces all physical quantities needed in a complete nuclear data library, with the exception of those produced by TARES, TANES and TAFIS.
- TEFAL[2]: a code for producing nuclear data libraries in ENDF format. TEFAL relies on the *output* of TALYS and processes all the TALYS output files into an ENDF library.
- TASMAN[2]: a code for covariances, optimization, sensitivities and other statistical information for TALYS. TASMAN works on the input and output of TALYS: it creates random input files for TALYS and collects all the output of those random runs to produce covariance matrices. Additionally, TEFAL can be included in that loop for Total Monte Carlo error

propagation.
- TARES[10]: a code for resonance parameters including covariances, based on the Atlas of Neutron Resonances and other data libraries.
- TAFIS[2]: a code for fission neutron quantities, based on Wahl systematics among others.
- TANES[2]: a code for prompt fission neutron spectra, based on the Los Alamos model among others.

However, inspection of the *bin/* directory of T6 reveals no less than 51 codes, much more than the aforementioned 6 codes. We will explain, or at least shortly mention, them all. A future clean up of T6 also means getting rid of some of these codes, or merging some of them into one code.

Besides the 6 'T' codes above, other important tools are:
- **autotalys**: a bash shell script that drives the entire T6 system. With **autotalys**, TENDL is produced, random files are made, TALYS parameters are optimized, etc. Later in this document, the **autotalys** command for the production of TENDL is given, as an example.
- ENDFTABLES: a Fortran code to produce a directory-structured x-y or x-y-dy database (if covariance matrices are available), entirely derived from an existing nuclear data library. Hence, all isotopic nuclear data files from the major world libraries are completely dismantled using ENDFTABLES and put into single files per reaction channel.
- *plot*: a plotting script which plots the cross section for one particular nuclear reaction comparing TALYS, all nuclear data libraries and EXFOR data, JENDL-4.0, CENDL-3.1, EAF-2010 and IRDFF-1.05. Use: e.g. plot n Nb 93 n2n. This is described in Section B.
- *plotall*: a plotting script which runs the aforementioned 'plot' for all important reaction channels per target nuclide. Use: e.g. plotall n Nb 093. This is described in Section B.
- *autonorm*: A Fortran code to normalize TALYS against other nuclear data libraries, described in Section 4/
- *driplist*: A code to produce a list of nuclides that **autotalys** will loop over. This can be the full TENDL range or other ranges of nuclides. driplist may be called from **autotalys**.
- *mf1maker*: a bash shell script that produces a full MF1 documentation file. mf1maker can be steered by various flags and is called from the **autotalys** script.
- *extrema*: a very short fortran program that determines the maximum and minimum of a cross section, for plotting purposes.
- *ZAres*: code to calculate the Z and A of the residual product given the MT number. This is needed for, and called from, 'plot'.
- *select*: bash script to randomize selected parts of an ENDF-6 file or to make covariance files during the random runs. This script is called from the TASMAN code.
- *run-plots*: script to run NJOY and produce plots from ACER, called from **autotalys**.
- *run-errorj*: script to run NJOY and produce covariance plots from NJOY, called from **autotalys**.

5.2.1 Checking and processing codes

Various well-known codes to check ENDF files and to process them for applications are included in T6
- The BNL checking codes for ENDF files. Not all BNL codes are needed, for T6 we use:
  - CHECKR: Format checks
  - FIZCON: Basic physics checks
  - PSYCHE: Advanced physics checks
  - INTER: Calculation of integral cross sections
- NJOY-2016.47[16]: NJOY processing code
- *njoycovx_ld*: special version of NJOY for covariances
- PREPRO-2018: suite of ENDF processing codes (we only list the PREPRO codes which are

needed)
  - *recent*: pointwise cross sections
  - *sigma1*: broadening cross sections
  - *sixpak*: transform MF6 into MF4 and MF5
  - *groupie*: groupwise cross sections
  - *evalhard*: plot cross sections
  - *evalplot*: plot cross sections
  - *activate*: create MF10 out of MF3 and MF9
  - *legend*: transform Legendre coefficients into tables
  - *merger*: combine evaluated data
  - *complot*: comparison of cross sections
  - *comhard*: comparison of cross sections
  - *dictin*: sum rules for cross sections
  - *linear*: linearize cross sections
- *xcalendf*: CALENDF processing code which is used in TARES
- *tafis-pfns*: Fortran code for TAFIS
- *tanes-pfns*: Fortran code for TANES

5.2.2 Scripts for running and diagnosis of checking and processing codes

To run all the processing codes, scripts have been written which produce, for the ENDF file under consideration, an input file for these codes and runs it. These bash shell scripts are described below

run.bnl

This script creates input files and runs the BNL checking codes CHECKR, FIZCON, PSYCHE and (optional) INTER. The default usage is e.g.

```
run.bnl -file n-Fe056.jendl4.0
```

and you will get the output

```
       run.bnl-1.0 (Version March 8 2019) (C) Copyright 2019 Arjan Koning,
             All Rights Reserved


 * Start BNL codes
run checkr
STOP     CHECKR - Tests completed successfully
run fizcon
STOP     FIZCON - Tests completed successfully
run psyche
STOP     PSYCHE - Tests completed successfully
run inter
STOP     INTER - Tests completed successfully
run.bnl is done
```

while your current directory contains the files checkr.inp(out) fizcon.inp(out), psyche.inp(out) and inter.inp(out). The output files can be consulted to see whether there are any problems with the nuclear data file, though we often do that automatically via the *diag.bnl* script.

It is possible to disable certain codes and give other options, e.g.

```
run.bnl -file n-Fe056.jendl4.0 -nopsyche -nointer
```

Appendix C gives all the possible options.

run.prepro

This script creates input files and runs the PREPRO processing codes mentioned before. The default usage is e.g.

```
run.prepro -file n-Fe056.jendl4.0
```

and you will get the output

```
       run.prepro-1.0 (Version May 23 2019) (C) Copyright 2019 Arjan Koning,
             All Rights Reserved

/Users/koning/bin/run.prepro -file n-Fe054.jendl4.0

 Directory with results:  /Users/koning/k/libraries/n/Fe054/jendl4.0/files/
 * Start PREPRO
run linear
run recent
run sigma1
run sixpak
run activate
run dictin
run groupie
run evalhard
run.prepro is done
```

and your current directory contains the many input and output files of which all output from the various PREPRO codes have been accumulated in the file *prepro.out*. This output file can be consulted to see whether there are any problems with the nuclear data file, though we often do that automatically via the *diag.prepro* script. It is possible to disable certain codes and give other options, e.g.

```
run.prepro -file n-Fe056.jendl4.0 -nogroupie
```

Appendix D gives all the possible options.

run.njoy

This script creates input files and runs the NJOY processing code mentioned before. The default usage is e.g.

```
run.njoy -file n-Fe056.jendl4.0
```

and you will get the output

```
       run.njoy-1.0 (Version May 28 2019) (C) Copyright 2019 Arjan Koning,
             All Rights Reserved

/Users/koning/bin/run.njoy -file n-Fe054.jendl4.0

       Starting time: 11-28-2019, 03:28:05 PM

MAT: 2625 AWI: 1.000000+0 element: Fe mass: 054 iso: projectile: neutron
 Directory with results:  /Users/koning/k/libraries/n/Fe054/jendl4.0/files/b/
 * Start NJOY for  n-Fe054.jendl4.0
```

```
 * NJOY done


  ...  It took 27 seconds
  ...  Ending time: 11-28-2019, 03:28:32 PM

run.njoy is done
```

and your current directory contains the njoy.inp file and various output files such as the standard NJOY output *njoy.out* but also and ACE file, a xsdir file and a PENDF file, plus postscript files with NJOY plots. The output file can be consulted to see whether there are any problems with the nuclear data file, though we often do that automatically via the *diag.njoy* script. It is possible to disable certain modules and give other options, e.g.

```
run.njoy -file n-Fe056.jendl4.0 -noacer
```

Appendix E gives all the possible options.

run.fudge

This script creates input files and runs the FUDGE processing code mentioned before. The default usage is e.g.

```
run.fudge -file n-Fe056.jendl4.0
```

Diagnosis scripts

Diagnosis scripts have been written which filter the warning and error messages from the output of the most important checking and processing codes. These are needed since TENDL contains thousands of output files for each of these codes, so a clever method to discover errors must be used. The available scripts are:

- diag.checkr <CHECKR output file>
- diag.fizcon <FIZCON output file>
- diag.psyche <PSYCHE output file>
- diag.prepro <PREPRO output file>
- diag.njoy <NJOY output file>
- diag.fudge <FUDGE output file>

Each script has been constructed in the same way. The source files of the checking or processing code have been scanned for all possible warning and error messages. These messages have been listed in each 'diag' script and the output files of these codes are then systematically checked for any of these messages by the script. For example the NJOY output file is scanned for terms like 'not allowed', 'illegal', 'insufficient' etc. since those terms have been programmed as warning or error messages in the NJOY source code. If the output from all 'diag' scripts is zero we probably have an ENDF file of, at least formal, good quality. With these scripts, the TENDL creation procedures have been cleaned until all errors were gone.

5.3 Libraries with data not provided by TALYS

5.3.1 RESBASE: Database with resonance parameters

For the first versions of TENDL, up to TENDL-2017, the production of the nuclear data libraries required running the TARES, TAFIS and TANES codes on the fly. At some point in the **autotalys** process TARES was called to produce MF2/MF32 formatted resonance parameters and their covariances, and similarly for the fission data of TAFIS and TANES. It was then decided that it may be more flexible, robust, and making T6 more portable, if resonance parameters are not produced

on the fly by TARES but if they can be imported in TALYS, TEFAL and TASMAN as prepared tables. A resonance parameter database is made, using **loopnuc**, when a new update of TARES is available and the resulting database is then used as input to T6. This will make TENDL production also more efficient. This database uses the filename convention of T6 and TALYS, using projectile, isotope and filetype in the name. Taking ^{93}Nb as an example, we have for each isotope:

- n-Nb093.mf1.tendl: MF1 file describing the resonance range
- n-Nb093.mf2.tendl: MF2 file
- n-Nb093.mf2.lrf7.tendl: MF2 file in LRF7 format
- n-Nb093.mf4.tendl: MF4 file for the resonance range
- n-Nb093.mf32.tendl: MF32 file
- n-Nb093.mf33.tendl: MF33 file in resonance range
- n-Nb093.res.tendl: A human-readable file of resonance parameters + uncertainties, so that TASMAN can sample from that for TMC files. This can later be upgraded to correlated sampling, but in order to go to the new system as soon as possible we may start with diagonal sampling. Alternatively, TASMAN would have to read the diagonal from MF32 file.
- n-Nb093.res.mlbw.tendl, n-Nb093.res.rm.tendl: proposal to have different files for different resonance representations

For other world libraries, TARES can produce e.g.

- n-Nb093.mf1.endfb8.0: MF1 file describing the resonance range
- n-Nb093.mf2.endfb8.0: MF2 file

which will require some future (boring) work of reading through the MF1 of each isotope of each world library to find the first and last line in MF1 that gives the resonance documentation, which can then be automatically adopted in the MF1 of TENDL.

Related issues are:

- The production of all the above files using HFR[17] for other (microscopic) TALYS input options, for astrophysical purposes.
- The consistent update of the TARES database for HFR calculation using the latest version of TALYS. In other words, produce this URR database now with TALYS-1.8, but if TALYS-2.0 is released the database needs to be reproduced so that the TARES outputs are consistent with TALYS.

5.3.2 NUBARBASE: Database with average number of fission neutrons

Similar to RESBASE, a database with all nubar etc. values has been produced by **loopnuc** running TAFIS for all actinides.

5.3.3 FNSBASE: Database with fission neutron spectra

Similar to RESBASE, a database with all prompt and delayed fission neutron spectra has been produced by **loopnuc** running TANES for all actinides.

5.4 Automatic plots

There are two scripts for automated plotting of TALYS results, all existing nuclear data libraries and the EXFOR data. One is **plot** to plot a particular reaction, and the other is **plotall** to plot all reactions for a projectile-nuclide combination. These scripts have been used to produce a complete collection of plots as described in the Chapter about libraries. Simply typing **plot** will give all the options of this bash script. Assuming the *libraries* database is installed, one can obtain a fairly complete plot with e.g.

```
plot n Nb 93 n2n
```

which you can directly write to an *eps* file with

```
plot n Nb 93 n2n print
```

Appendix B gives all the possible options. In addition, *plotall* calls *plot* for all reactions and is used as follows

```
plotall n Nb 93
```

after which your screen will fill up with plots of all reaction channels. You can obtain all plots on *eps* files with

```
plotall n Nb 93 'print'
```

5.5 MAT numbers

It should not be a surprise that the large number of nuclides covered in TENDL causes problems with the official definition of MAT numbers. We here outline the scheme we have used. As a basis, we start from the standard ENDF assignment

$$MAT = 100.Z + 25 + 3.(A - A_{\text{light}}) + L \tag{5.1}$$

where A_{light} is the mass number of the lightest stable isotope of the element and L is the isomeric number. These are generally well defined, e.g. Fe-54 is the lightest stable isotope of Fe, so its MAT number is 2625, making 2631 the MAT number of Fe-56. For elements without a stable isotope, i.e. Tc, Pm and all nuclides above Bi, with exception of Th and U, we follow the A_{light} definition of BNL's CHECKR code, (does any official ENDF documentation exist?) which makes sure that at least for nuclides present in ENDF/B-VII the definition is equal. For very light isotopes of an element, we continue counting down from the basis number 25 even if we cross the zero, i.e. the MAT number of Fe-46 is 2601 and Fe-45 is 2598. There is only a possibility of double MAT number definition in the case of an isomer of a very heavy or very light nucleus, since the MAT numbers of the lightest isotopes of Z run out of phase with the heaviest isotopes of Z-1.

5.6 Driplist and nuclide lists

To create TENDL and *libraries*, but also for other purposes, we need a list of nuclides to loop over. **driplist** may be called from **autotalys**. Go to a new directory and do

```
autotalys -Zbeg 3 -Zend 124 -lifetime 1. -noloop
```

which, as you may have guessed, loops over all nuclides between Z=3-124 with a half life equal to or longer than 1 second, plus the stable nuclides.

After about 30 seconds, this has created a file 'nuclides' with 2808 nuclides, sorted first by natural abundance and then by half life. Examples:

```
...
Al_027_000_0_1_100_300_E
...
Xe_132_000_0_1_026_060_E
...
Eu_154_000_0_0_99999992_020
...
Te_123_002_1_0_10300000_010
...
Hf_179_046_2_0_02164000_010
```

with on each line, separated by underscores, element, mass number, excited target level, isomeric number, identifier for stable (1) or radioactive (0) nuclide, abundance for stable or half life in sec for radioactive, number of random runs for the case of uncertainty quantification, designator 'E' if EXFOR data exists.

In *t6/tools/tendl*, we have stored several nuclide lists, e.g. *nuclides.light, nuclides.dat, nuclides.sec, nuclides.year*

5.7 TENDL (to be cleaned up)

5.7.1 Creating TENDL

In Chapter 3, the **loopnuc** script was introduced, which reads the *nuclides* file to produce executable scripts. It can be found in *t6/tools/tendl/*. If we do 'loopnuc n > n.all', we have 2808 executable **autotalys** scripts, each with a projectile and nuclide name, e.g. n-Fe056. The file 'n.all' contains all these 2808 scripts, one per line. Next, it depends on the computer cluster how these scripts are distributed over the various processors.

When all runs are finished, **allnuccheck n** gives a diagnosis of the finished nuclides. The .diag files contain the errors and warnings.

The script **nuclistcheck n** loops over nuclides to list all data files which are and are not created.

E.g. the script n-Pd106 looks as follows

```
#!/bin/bash
/Users/koning/t6/bin/autotalys -element Pd -mass 106 -Ltarget 000 -Liso 0
-ntalys 060 -proj n -bins 60 -high -scratch -sdefault -acf -eaf -njoy
-residual -isomer -recoil -delete -noclean -binsrand 60 -plot -subfission
-gzip  -tasmanfile /Users//koning/tasman/aux/tasman.tendl2019 -nomcnp
-covar -s20 -s60
```

5.7.2 Production of TENDL files

Some future options and considerations for TENDL are

- Continue working on the so called 'best' files which are stored in *talys/structure/best*,giving the best possible fits to experiment. Ideally 'best' files are made for all projectiles and target nuclides. These best files are not only for TALYS, but also for TASMAN, TEFAL and other codes.
- Improve the neutron and proton OMP above 200 MeV: there is a nasty bump in the proton reaction c.s. above 200 MeV. After this, data files up to 1 GeV could be made.
- Make a general ATLAS of cross sections, with on one page:
  - One or more plots of the reaction, a different scales
  - A table with all EXFOR sets and all libraries and a C/E plus chi-2
  - For capture: comparison of all libraries with thermal Res Int and Macs
  - Comparison with EASY database, again for all world libraries.

5.7.3 Testing and Processing

Various tests can be done to assess the basic and more advanced quality of TENDL.

- Grep on Inf and NaN in all produced ENDF-6 files.
- Investigate the quality of fit with the differential data plots which have been automatically produced.
- Test TENDL against thermal cross section database
- Test TENDL against 30 keV MACS database
- Test TENDL against Resonance integral database

- Automate integral activation (EASY) testing of TENDL
- Test against ICSBEP
- More extensive MCNP(X) testing suite, e.g. standard Oktavian.

Ideally integral validation should be so automated that they can be used DURING TALYS or resonance optimization. This is probably the largest challenge for T6.

5.7.4 Strategy for TENDL

For neutrons:

- Adopt the big 5 from ENDF/B-VIII: 1H, 16O, 235U, 238U, 239Pu. In first instance, a direct copy (no autonorming), this will give us a base to test ALL other materials in TENDL, knowing that the big 5 are (reasonably) good. When TENDL is integrally tested for ICSBEP, this will give a direct difference of all the isotopic evaluations of ENDFB/VIII compared to TENDL.
- Later: Adopt the big 5 from CIELO using autonorming. This will give the possibility to complete the files with missing information provided by the more complete TENDL approach, and for doing TMC and the Petten method.
- Be conservative with ENDF format. The standard file has: MT5 cut at 30 MeV, MF33 in the resonance range instead of MF32, and no LRF7. The other files, e.g. 's20' can be used for MF32 and LRF7. ENDF/B testing before we adopt LFR7.
- Adopt everything up to and including F-19 from ENDF/B-VIII
- Later: Autonorming the light nuclides
- Maintain a script with all light nuclides and actinides to be copied from other nuclear data libraries.
- adopt EGAF (thermal gamma production data)

5.7.5 Creating latest TALYS results for development

For development of the latest library, it may be helpful to compare the EXFOR data and data from official libraries with the latest results of TALYS, while improvements to the TALYS input files are made. The input files of these can then be adjusted until the results are satisfactory, and the 'best' TALYS input files can be stored in *talys/structure/best/*. Often it is then efficient to go back and forth between e.g. directories of different isotopes of the same element until an optimum is reached. For this one can again run **loopnuc** though now with an **autotalys** flag which produce only quick TALYS results and no ENDF libraries, e.g.

```
autotalys -element Fe -mass 056 -proj n -norescue -noendf -nocovar -E30 -thin
```

The TALYS outputs are stored in directory drip/ in subdirectories g/ n/ p/ d/ t/ h/ and a/. Run e.g. 'loopnuc p > tendl.all' to produce scripts for every isotope for incident protons. There is also a subdirectory plots. This can be filled with latex documents containing plots using the script nucplot, which itself makes use of the code texmaker.f.

5.7.6 Other

talcmp

To be done.

nucplot

To be done.

texmaker.f

To be done.

allnuccheck

To be done.

drip/ or other name

To be done.

autorochman

To be done.

6. Nuclear data automation with autotalys

As described in the Chapter on the T6 system, the total nuclear data evaluation system consists of a whole suite of codes built around TALYS, uncertainty quantification tools, ENDF formatting and checking codes, processing codes and other software. To make the nuclear data evaluation process as traceable, systematic and reproducible as possible, a single script has been created which gives us all the flexibility we need, and is also powerful for the mass production of nuclear data libraries of all sorts. This chapter describes this **autotalys** script as well as some examples.

6.1 autotalys

As its name suggests, **autotalys** automatically performs a wide variety of tasks in which TALYS is involved. When running **autotalys**, it is assumed that all codes it calls are compiled and available, and all specific input files for these codes are stored at the right place. This is basically what the T6 system involves. By giving a simple **autotalys** command like e.g.

```
autotalys -element Zr -mass 90
```

TALYS and other software will run and eventually a complete ENDF nuclear data library including covariance matrices will be produced, as well as all possible checks, plotting (versus EXFOR and other nuclear data libraries) and processing steps for this ENDF file will be done. Let us have a closer look at the simple example given above. It is not explicit about the fact that there are close to 100 flags for **autotalys** for setting various parameters, enabling or disabling options, etc. Appendix 6 gives an exhaustive list of all the options of **autotalys**.

As usual, all these flags have defaults and although **autotalys** can be used for very different tasks which may have nothing to do with the creation of an ENDF-6 library, we have chosen that as default task (setting **-noendf** gets rid of ENDF files in the final results). The simple command above has the following important defaults:

- the projectile is a neutron
- the incident energy grid runs up to 20 MeV only

- uncertainty quantification is enabled and a modest number of 10 random TALYS runs is used to come to the final answer
- an ENDF-6 file including covariance data is produced
- 10 random ENDF-6 files for use in Total Monte Carlo are produced
- the ENDF-6 file is checked and processed by the BNL checking codes, PREPRO and NJOY and the resulting diagnosis and processed PENDF, ACE etc files are stored.
- plots of the results versus EXFOR and other evaluated data libraries are automatically produced.

At this point it may be good to repeat the **autotalys** command mentioned in the Introduction, namely the one that that produces the TENDL-2019 file for neutrons on ^{241}Am, including covariance data.

```
autotalys -proj n -element Am -mass 241 -high -bins 60 -njoy -residual
-isomer -ntalys 100 -recoil -covar -binsrand 60 -plot -subfission -nomcnp
-tasmanfile /Users/koning/tasman/aux/tasman.tendl2019
-sdefault -s20 -s60 -acf -eaf -mt
```

Here we have been somewhat more explicit with the keywords, even though some could have been left out since they are set to their default value. In particular for a neutron TENDL file we set:

- the projectile, just to be explicit,
- the files runs up to 200 MeV though the keyword **-high**,
- We use a higher number of continuum bins in TALYS for both the central value file, with **-bins** and the random files, with **-binsrand**, for more precise results.
- As TALYS output, residual production cross sections (**-residual**), isomeric production (**-isomer**) and recoils (**-recoil**) are included,
- Though not relevant for ^{241}Am (it would be for e.g. Pb isotopes), subactinide fission is enabled with **-subfission**,
- We use 100 random TALYS runs to get a well-converted covariance matrix, with **-ntalys 100**,
- Extra keywords for TASMAN are included by the TASMAN input file given by **-tasmanfile**,
- We run NJOY for processing (**-njoy**) but do not run MCNP for testing (**-nomcnp**),
- We make plots of the final library with **-plot**,
- 6 different structured ENDF files are made with **-sdefault -s20 -s60 -acf -eaf -mt**.

For incident protons, the command to make the TENDL-2019 file looks only slightly different,

```
autotalys -proj p -element Ni -mass 058 -bins 60 -high -njoy -residual
-isomer -ntalys 60 -recoil -covar -binsrand 60 -plot -subfission -nomcnp
-tasmanfile /Users/koning/tasman/aux/tasman.tendl2019
-sdefault -s0 -s60 -acf -eaf -mt
```

6.2 Evaluation of all actinides

We consider the following interesting challenge: Can we build a systematic evaluation approach which gives a simultaneous good description for all actinides for which experimental data exist, i.e. from ^{227}Ac to ^{252}Cf, and clearly beyond (for which no other reliable libraries exist anyway). The objective is the same as for nuclides other than actinides in TENDL: can we systematically improve the nuclear data libraries over a broad mass range, such that for more and more nuclides the superior evaluation comes from our system instead of from the other world nuclear data libraries. Hence, for this case, we should not expect (yet) that the best possible evaluation for the "big 3" ^{235,238}U and ^{239}Pu comes from this approach, but perhaps for several minor actinides the current data libraries start to approach the quality of the others.

6.2.1 Setting up the autotalys scripts

Analogous to the **loopnuc** script described before, we have written a **loopsearch** script which loops over all actinides for which experimental (n,f) cross sections exist, so we get a list of executable scripts

```
n-Th232
n-U238
n-U235
n-Pu239
n-U233
n-U234
n-U236
n-U237
n-U232
n-Pu240
n-Pu242
n-Am241
n-Am243
n-Np237
n-Cm243
n-Cm244
n-Cm245
n-Cm246
n-Cm247
n-Cm248
n-Th230
n-Th228
n-Pu241
n-Pu244
n-Pu238
n-Pa231
n-Np236
n-Cf249
n-Cf250
n-Cf252
n-Bk247
n-Bk248
```

Each of this scripts contains exactly the same **autotalys** command, with exception of element and mass of course. The example for ^{233}U is

```
#!/bin/bash
/Users/koning/bin/autotalys -element U -mass 233 -Ltarget 000 -Liso 0
-proj n -energyfile /Users/koning/search/energies -search -noautosearch
-talysfile /Users/koning/search/act.talys
-tasmanfile /Users/koning/search/act.tasman
-noendf -nocovar -binsrand 40 -nobest -norescue  >& n-U233.log &
```

Let us describe the most important flags here. Clearly, there are a few files in */Users/koning/search/* which are equal for all cases. Indeed, since we first want to optimize first-chance fission only, the *energies* file is

```
0.01
0.02
0.04
0.07
0.1
0.2
0.4
0.7
1.000
1.200
1.400
1.600
1.800
2.000
2.200
2.400
2.600
2.800
3.000
3.200
3.400
3.600
3.800
4.000
4.200
4.400
4.600
4.800
5.000
```

6.2.2 Physics

With the flag **-talysfile**, the TALYS keywords from */Users/koning/search/act.talys* are included. This file looks as follows

```
maxrot 4
strength 8
strengthm1 8
upbend y
ldmodel 5
fismodel 5
fispartdamp y
hbstate n
class2 n
filechannels y
channels y
ecissave y
eciscalc y
inccalc y
outdiscrete y
riplrisk y
```

First of all, we must stress here that the latest version of TALYS can include all the OMP's from the RIPL database and has as default the dispersive OMP of Soukhovitskii et al, RIPL OMP 2408, which gives a good description of all OMP related observables over the whole actinide range. The keyword **riplrisk y** allows TALYS to go beyond the original mass range for this OMP. The other global models we set are a rotational band up to the 4th excited state, microscopic PSF including an M1-upbend, microscopic HFB based level densities with **ldmodel 5**, and HFB based fission paths using the WKB approach **fismodel 5**.

6.2.3 Parameter search

Assuming a good OMP, the remaining available experimental data to be fitted to for actinides consists of (n,f), (n,γ) and for only a few important actinides (n,n') and (n,2n) cross sections. We first will fit the first-chance (n,f) cross sections only. With the flag **-tasmanfile**, the TASMAN keywords from */Users/koning/search/act.tasman* are included. This file looks as follows

```
#mode 3
#chi2 2
#ecis n
#save y
#deltaE y
#parameters y
##expinclude 18
#libinclude 18 endfb8.0
#fracmax 0.02
#cwidth 2.
#Zdeep 0
#Adeep 0
#minbar 1
#searchmode 2
##keyvary bdampadjust
#keyvary ctable
#keyvary ptable
#keyvary vfiscor
#keyvary betafiscor
```

The TASMAN tutorial gives a full description of all these options. Here it is important that we have chosen to optimize to the (n,f) cross sections of ENDF/B-VIII instead of to experimental data, and that we *only* want to optimize parameters for the fission barriers (and not the ground state) for the initial compound nucleus. This is enforced by **Zdeep, Adeep, minbar**. The parameters which are optimized are **ctable, ptable** for the level densities on top of each barrier and the global adjustable factors **vfisvor, betafiscor** for the WKB fission paths. Hence, there are usually 6 parameters to be optimized.

6.2.4 Results

The 32 autotalys scripts can be started, and after several hours, the search process starts to converge. Using the *plot* script, plots of the results can be made, and here the results after 2 days of searching is plotted.

If we look at the results we can perhaps classify the quality of fit, **as compared to ENDF/B-VIII** as follows.

Excellent fit

- Th228

- Np237
- U232
- U233
- Pu240
- Cm244
- Cm245

Good fit, though structure missing

- Th232
- Th230
- Np236

Local deviations, overall good normalization

- U234
- U235
- U236
- U238
- Pu238
- Pu239
- Pu242
- Cm243
- Cm246
- Cm247
- Am241
- Cf250

Larger deviations

- Pu241
- Pu244
- Am243

Missing threshold by more than 500 keV

- Pa231
- Cm248
- Bk247
- Bk248
- Cf252

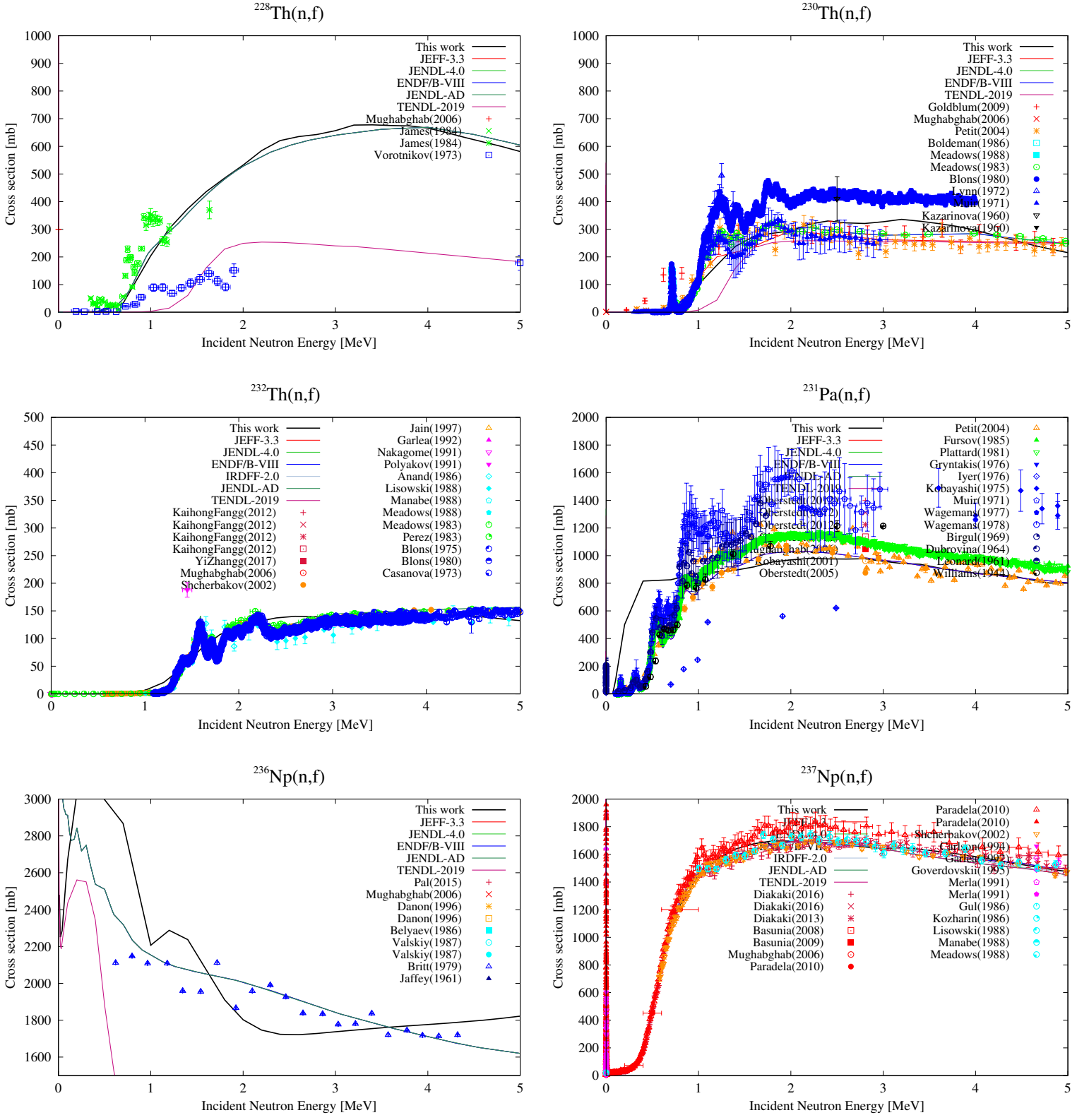


Figure 6.1: Fission cross sections for Th, Pa and Np isotopes.

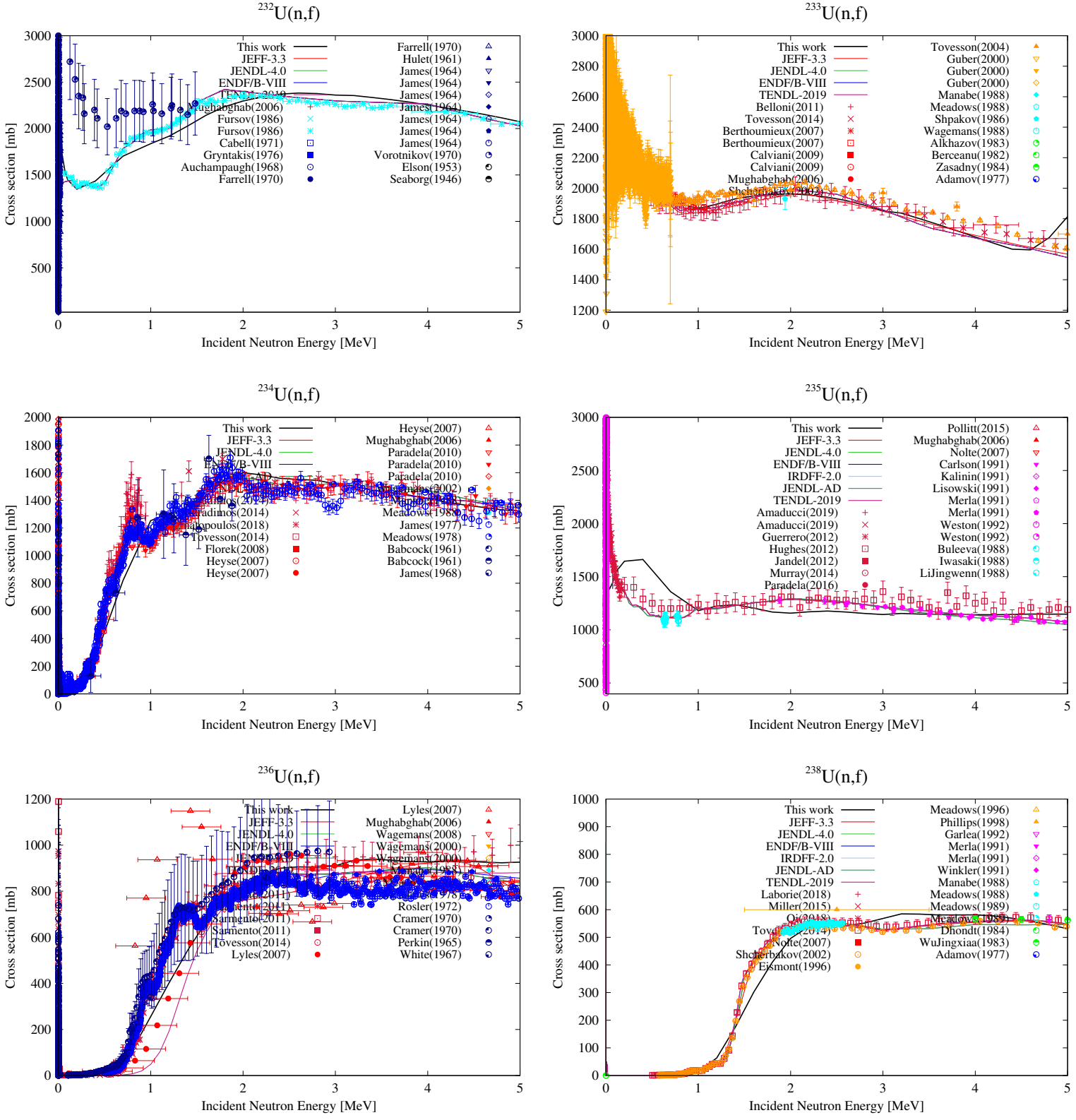


Figure 6.2: Fission cross sections for U isotopes.

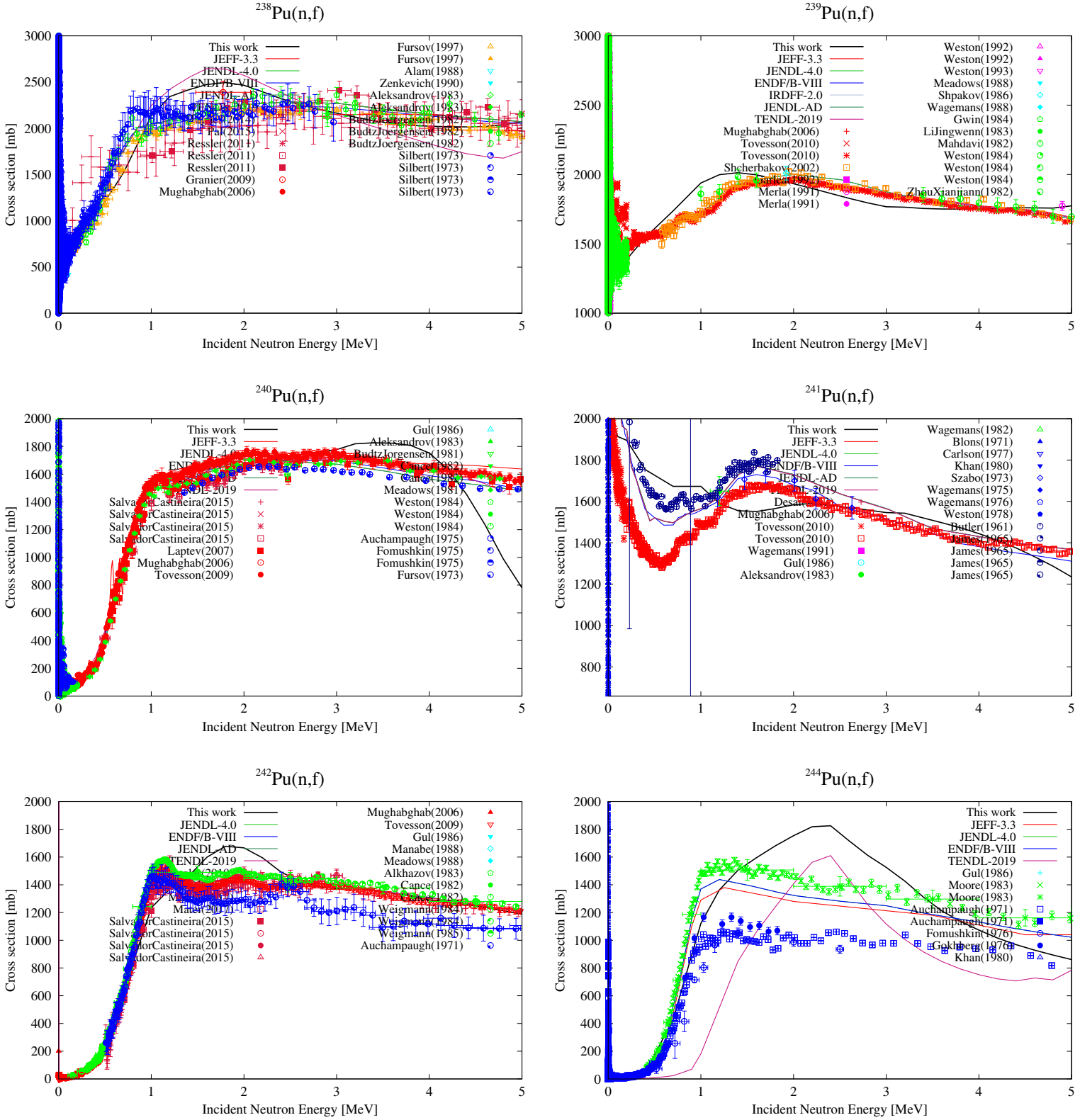


Figure 6.3: Fission cross sections for Pu isotopes.

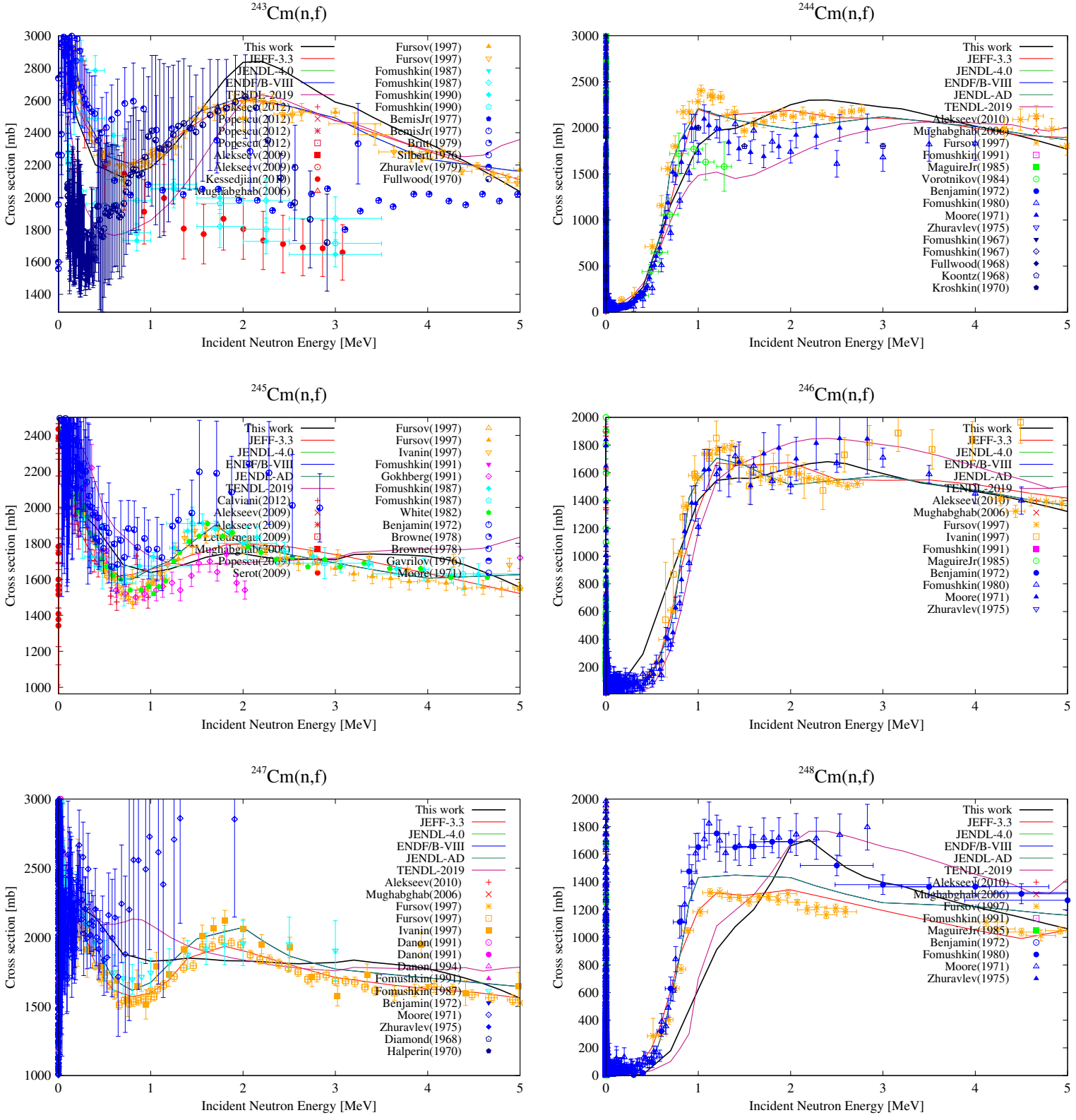


Figure 6.4: Fission cross sections for Cm isotopes.

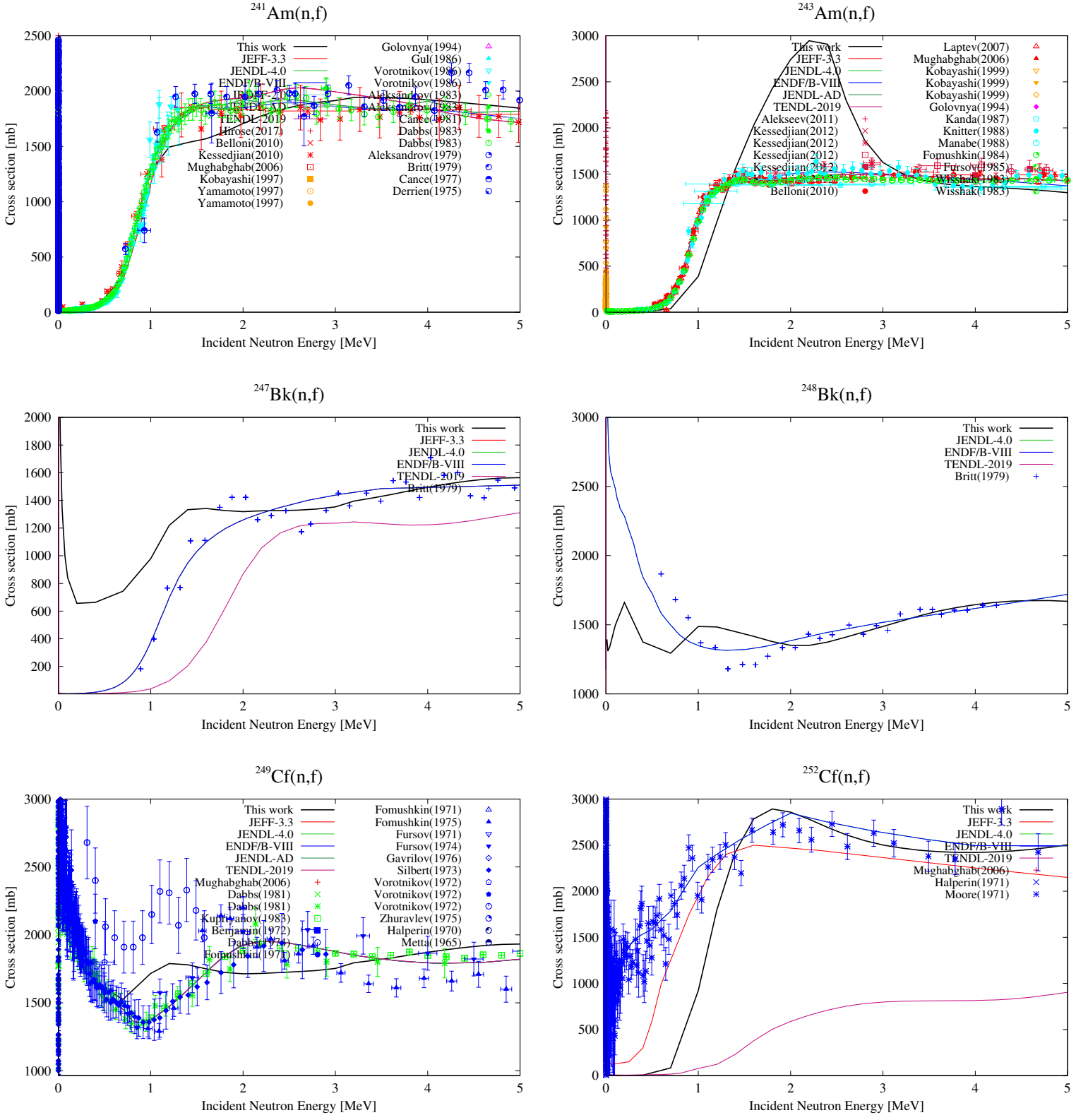


Figure 6.5: Fission cross sections for Am, Bk and Cf isotopes.

7. Conclusions and outlook

It requires some discipline to document close to everything you have ever produced in terms of software. This "Tools for TALYS" tutorial is an attempt in that direction. All scripts that drive TALYS and its satellite codes have been outlined and with this tutorial you should be able to "make your own TENDL", and perform a variety of other tasks. Nuclear data evaluation is by many regarded as a nuclide-by-nuclide or even channel-by-channel activity. Examples of this are e.g. the CIELO initiative of the past decade where all effort was focused to maximize the quality and minimize the uncertainty of a few reaction channels.

The output of such evaluations, though often impressive in quality, are not computationally reproducible, even though the “core knowledge” of such evaluations is much more compact than the final nuclear data library. The T6 approach built around TALYS relies on computational reproducibility and a systematic approach. At some point, computational methods such as Machine Learning may also arrive in the nuclear field, and then it will become even more clear that this is needed.

Bibliography

[1] A.J. Koning and D. Rochman. “Modern nuclear data evaluation with the TALYS code system”. In: *Nuclear Data Sheets* 113 (2012), p. 2841.

[2] A.J. Koning et al. “TENDL: Complete Nuclear Data Library for innovative Nuclear Science and Technology”. In: *Nuclear Data Sheets* 155 (2019), p. 1.

[3] A.J. Koning and D. Rochman. “Towards sustainable nuclear energy: Putting nuclear physics to work”. In: *Annals of Nuclear Energy* 35.11 (2008), pp. 2024–2030. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2008.06.004. URL: http://www.sciencedirect.com/science/article/pii/S0306454908001813.

[4] N. Otuka et al. “Towards a more complete and accurate experimental nuclear reaction data library (EXFOR): International collaboration between nuclear reaction data centres (NRDC)”. In: *Nuclear Data Sheets* 120 (2014), pp. 272–276.

[5] D. E. Cullen. *PREPRO 2018 - 2018 ENDF-6 Pre-processing codes*. Tech. rep. Technical report IAEA-NDS-39 (Rev.18), IAEA (2018). 2017.

[6] C.M. Mattoon et al. “Generalized Nuclear Data: A New Structure (with Supporting Infrastructure) for Handling Nuclear Data”. In: *Nuclear Data Sheets* 113 (2012), pp. 2145–3171.

[7] C. Dunford. *ENDF Utility Codes Release 6.10*. Tech. rep. IAEA report, IAEA-NDS-29. Brookhaven National Lab., 1995.

[8] A.J. Koning. *EXFORTABLES, Bayesian Monte Carlo method for nuclear data evaluation, Eur. Phys. Journ. A51(12) 1 (2015)*. 2020.

[9] A.J. Koning. *RESONANCETABLES-1.0: Database for thermal cross sections, MACS and average resonance parameters*. 2020.

[10] J.-Ch. Sublet D. Rochman A.J. Koning. “A statistical analysis of evaluated neutron resonances with TARES for JEFF-3.3, JENDL-4.0, ENDF/B-VIII.0 and TENDL-2019”. In: *Nuclear Data Sheets* 163 (2020), p. 163.

[11] M. Herman et al. “EMPIRE: Nuclear Reaction Model Code System for Data Evaluation”. In: *Nuclear Data Sheets* 108.12 (2007). Special Issue on Evaluations of Neutron Cross Sections, pp. 2655–2715. ISSN: 0090-3752. DOI: DOI:10.1016/j.nds.2007.11.003. URL: http://www.sciencedirect.com/science/article/B6WNV-4RBYC24-3/2/c48a3c57f47635bdf34f020814f75bc0.

[12] T. Kawano. “DeCE: the ENDF-6 data interface and nuclear data evaluation assist code”. In: *Journal of Nuclear Science and Technology* 56 (2019), pp. 1029–1035.

[13] A. Trkov. *ENDVER ENDF-6 File Verification Support Package*. https://nds.iaea.org/public/endf/endver.

[14] V. Zerkin. *ZVVIEW*. https://www.iaea.org/resources/databases/zvview.

[15] N. Soppera. *JANIS*. https://www.oecd-nea.org/jcms/pl_39910/janis.

[16] A.C. Kahler and R.E.MacFarlane. “Methods for Processing ENDF/B-VII with NJOY”. In: *Nuclear Data Sheets* 111 (2010), pp. 2739–2890.

[17] D. Rochman et al. “From average parameters to statistical resolved resonances”. In: *Annals of Nuclear Energy* 51 (2013), pp. 60–68.

A. All options for autotalys

Giving simply *autotalys* will give you all the options. For completeness we give all the options below

```
      Autotalys-1.92 (Version October 31 2019) (C) Copyright 2019 Arjan Koning
          All Rights Reserved

/Users/koning/bin/autotalys

### A Input and Initialization
  # A.1 Output of all options
AUTOTALYS                                                       AUTOTALYS



NAME
      autotalys - Script to run programs of the TALYS nuclear data system and
                  to move all results into appropriate directories

SYNOPSIS
      autotalys [option] [value]

DESCRIPTION
      With Autotalys, various tasks of the TALYS system can be automated:
        - For one nuclide
        - For a user-specified nuclide range
        - For the entire nuclide chart (TENDL)

      The tasks which can be performed are
        DATA LIBRARY PRODUCTION
```

```
      - Creation of an input file for TALYS
      - Run TALYS, with or without automatic normalization
      - Run TEFAL to produce an ENDF-6 data library
      - Run TASMAN to randomize inputs and create random data libraries or
          covariance matrices
      - Run TARES to produce resonance parameters + uncertainties
      - Run TAFIS to produce fission parameters + uncertainties
      - Run TANES to produce fission neutron spectra + uncertainties
      - Run CHECKR, FIZCON, PSYCHE and INTER to check data libraries
      - Run PREPRO codes to check and process data libraries
      - Run NJOY to process data library and run MCNP test calculation
      - Store all x-y tables, ENDF-6 data libraries and other results in
          appropriate directories

      SENSITIVITY ANALYSIS
      - With TASMAN, nuclear model parameter sensitivities can be calculated,
          by linearly
        changing the input parameters one by one. With Autotalys, this can be
            looped over nuclides.

      SEARCH (automatic optimization to experimental or evaluated data)
      - This is under development

    Consult the Autotalys manual for a complete description


Operation mode:

    Usage: autotalys [option] [value]

    Some option flags should be accompanied by a value

    Examples: autotalys -element Nb -mass 93 (single-target case)
              autotalys -Zbeg 10 -Zend 20 -noendf (multi-target case)

General options: projectile, nuclide (range) and energies

    -proj [projectile]
         Projectile: n,g,p,d,t,h or a [default: -proj n]

    -element [element]
            Element [default: -element Nb]
            Use either the '-element -mass' or the '-Zbeg -Zend' combination

    -mass [mass]
         Mass number [default: -mass 93]
         Use either the '-element -mass' or the '-Zbeg -Zend' combination

    -Ltarget [Ltarget]
            Level number of target [default: -Ltarget 0]

    -Liso [Liso]
         Isomeric number of target [default: -Liso 0]

    -Zbeg [Z value]
         First Z number in range, e.g. 9 [default: -Zbeg 0]
```

```
      Use either the '-element -mass' or the '-Zbeg -Zend' combination

-Zend [Z value]
      Last Z number in range, e.g. 15 [default: -Zend 0]
      Use either the '-element -mass' or the '-Zbeg -Zend' combination

-nlevels [number]
        Maximum level number to include [default: -nlevels 30]

-nuclides [name]
         File with list of nuclides [default: none]

-lifetime [time]
         Lifetime (s) above which nuclides are included [default: none]

-abun [abundance]
      Abundance (%) above which nuclides are included [default: none]

-lifeskip [time]
         Lifetime (s) above which nuclides are skipped [default: none]

-abskip [%]
       Abundance (%) above which nuclides are skipped [default: none]

-high
      List of incident energies up to 200 MeV [default: -low]

-low
     List of incident energies up to 20 MeV [default: -low]

-thin
     Thin energy grid [default: -nothin]

-E30
     List of incident energies up to 30 MeV [default: -low]

-E60
     List of incident energies up to 60 MeV [default: -low]

-E1000
     List of incident energies up to 1000 MeV [default: -low]

-E [E value]
   Individual incident energy in MeV [default: not used]

-energyfile [name]
           File with incident energies [default: not used]

-prefix [name]
           Prefix for name of nuclide directories [default: not used]
-[no]loop
         Only create nuclide list, do not loop over nuclides [default: -
            loop]


TALYS keywords and version
```

```
    -bins [bins]
         Number of bins for TALYS calculation [default: -bins 40]

    -version [name]
           Specific TALYS executable in bin/ [default: -version talys]

    -talysfile [name]
              File with TALYS keywords [default: not used]

    -[no]recoil
              Include recoil information [default: -norecoil]

    -[no]macs
              Calculate MACS [default: -nomacs]

    -[no]levels
             Calculate discrete level cross sections [default: -nolevels]

    -[no]micro
             Use microscopic TALYS options [default: -nomicro]

    -[no]best
            Use best files if present [default: -best]

    -[no]rescue
              Use rescue files if present [default: -rescue]

    -[no]autonorm
              Automatically normalize data [default: -autonorm]

    -[no]omponly
            Execute ONLY an optical model calculation [default: -noomponly]

    -[no]subfission
            Include high energy fission for subactinides [default: -
                nosubfission]

    -ldmodel
            Level density model [default: -ldmodel 1]

Covariance or optimization calculation

    -[no]covar
             Total Monte Carlo and/or covariances [default: covar]

    -binsrand [bins]
            Number of bins for random TALYS runs [default: -binsrand 20]

    -ntalys [runs]
           Number of random TALYS runs [default: -ntalys 10]

    -nburn [runs]
           Number of burn in runs [default: -nburn ntalys]

    -[no]isomer
```

```
            Covariance data for isomers [default: -noisomer]

     -[no]residual
            Covariance data for residual cross sections [default: -residual]

     -seed [number]
          Seed for random number generator [default: mass number]

     -offset [number]
            Offset for TALYS run counter [default: -offset 0]

     -tasmanfile [name]
               File with TASMAN keywords [default: not used]

     -[no]allrandom
              Store random files for all ENDF options [default: -noallrandom]

     -[no]allprocess
              Process files for all ENDF options [default: -noallprocess]

     -[no]weight
              Use weighted random runs [default: -noweight]

     -[no]ompvar
               Vary optical model parameters [default: -ompvar]

     -[no]parauto
               Vary automatically many TALYS parameters [default: -parauto]

     -[no]sens
             No linear sensitivity analysis [default: -nosens]

     -[no]search
               Perform automatic optimization [default: -nosearch]

  Output: x-y tables, ENDF files and options

     -[no]talys
              TALYS and/or TASMAN run [default: -talys]

     -tefalfile [name]
               File with TEFAL keywords [default: not used]

     -libname  [name]
               Name of library for ENDF MF1 [default: TENDL-2019]

     -libext   [name]
               Name of library extension [default: tendl]

     -[no]tables
               Create x-y tables for results [default: -tables]

     -[no]plot
               Create plots with differential data [default: -plot]

     -[no]endf
```

```
         Create ENDF-6 files [default: -endf]

-[no]gpf
        Create ENDF-6 General Purpose File (GPF) [default: -gpf]

-[no]sdefault
        GP file with MT5 switch at the default energy:
        30 MeV [sdefault: -sdefault]

-[no]s0
        GP file with MT5 switch at 0 MeV [default: -nos0]

-[no]s20
        GP file with MT5 switch at 20 MeV [default: -nos20]

-[no]s60
        GP file with MT5 switch at 60 MeV [default: -nos60]

-[no]mt
        GP file without MT5 cut [default: -nomt]

-[no]mtextra
           Use ENDF-6 MT numbers MT151-200 [default: -nomtextra]

-[no]lrf7
           Use LRF7 (R-matrix limited format) for MF2 [default: -nolr7]

-[no]select
          Data files with selected random parts [default: -noselect]

-[no]covrand
           Data files with updated covariances [default: -nocovrand]

-[no]nubar
         Fission (nubar) values with TAFIS [default: -nubar]

-[no]nubarbase
           Take nubar from database or online TAFIS calculation [default:
               -nubarbase]

-[no]fns
        Fission neutron spectrum with TANES [default: -fns]

-[no]fnsbase
           Take FNS from database or online TANES calculation [default: -
               fnsbase]

-[no]acf
        No ENDF-6 activation file [default: -noacf]

-[no]eaf
        EAF activation file [default: -noeaf]

-[no]resbase
           Take resonances from database or online TARES calculation [
               default: -resbase]
```

```
    -[no]compact
               Compact resonance covariance matrix [default: -compact]

    -[no]talysurr
               Adopt URR parameters from TALYS [default: -notalysurr]

    -[no]mf33res
               Resonance covariances in MF33 [default: -nomf33res]

    -[no]bnl
              Check ENDF-6 file with BNL codes [default: -bnl]

    -[no]prepro
              Check ENDF-6 file with PREPRO system [default:-prepro]

    -[no]fudge
              Check ENDF-6 file with FUDGE system [default:-nofudge]

    -[no]njoy
             Produce ACE file with NJOY [default: -njoy]

    -[no]purr
             Use PURR in NJOY [default: -nopurr]

    -[no]mcnp
             Run MCNP with ACE file [default: -nomcnp]

    -[no]integral
             Calculation of integral effective cross section [default: -
                nointegral]


System options

    -[no]gzip
             Gzip working directory [default: -nogzip]

    -[no]tar
             Tar final directory [default: -notar]

    -[no]tarwork
             Tar work directory [default: -notarwork]

    -[no]scratch
               /scratch directory for results [default: -noscratch]

    -[no]clean
              Clean up working directories [default: -noclean]

    -copydir [name]
              Directory to which results are copied [default: not used]

    -movedir [name]
              Directory to which results are moved [default: not used]
```

```
      -[no]delete
                Delete previous scratch/ directory [default: -delete]

      -[no]replace
                Replace previous nuclide directory [default: -replace]

      -extension [extension]
               Use an extension (e.g. C, 77) for TALYS, TEFAL
               and TASMAN executables [default: blank]

      -bestlib [bestlib]
               Name of the best library used by TALYS [default: best]

      Extra options can be given to TALYS, TEFAL and TASMAN by providing
      files talys.add, tefal.add and tasman.add respectively, in the
      working directory, containing the usual keywords for these codes.
      A safer option is to use -talysfile, -tefalfile and -tasmanfile,
          respectively,

      Consult the Autotalys manual for several sample cases

REQUIRED
      Codes and scripts needed to run Autotalys:

      TALYS    : nuclear reaction program
      TASMAN   : TALYS input randomizer, optimization and covariance program
      TEFAL    : ENDF-6 formatting program
      TARES    : program for ENDF-6 resonance and covariance data
      TAFIS    : program for ENDF-6 fission (nubar) parameters and covariances
      TANES    : program for ENDF-6 fission neutron spectrum and covariances
      driplist : program to produce a sorted list of nuclides for Autotalys
      autonorm : program to normalize TALYS results to evaluated or experimental
          data
      run.bnl  : script to run the checking codes CHECKR, FIZCON, PSYCHE and
          INTER
      CHECKR   : BNL code to check ENDF-6 formats
      FIZCON   : BNL code to check procedures in ENDF-6 data libraries
      PSYCHE   : BNL code to perform physics checks of ENDF-6 data libraries
      run.prepro: script to run the PREPRO codes
      PREPRO   : Lawrence Livermore pre-processing codes LINEAR, FIXUP, etc.
      FUDGE    : Lawrence Livermore re-formatting code
      run.njoy : script to run NJOY
      NJOY     : Los Alamos processing program
      run.fudge : script to run the FUDGE codes
      run.mcnp : script to run MCNP
      MCNP     : Los Alamos Monte Carlo transport program (not implemented)

      The above requirement list depends of course on the particular use of
          Autotalys,
      e.g. whether ENDF-6 data library production, checking and processing is
          included or not

AUTHOR
      Autotalys was written by Arjan Koning.
```

B. All options for plot and plotall

Giving simply *plot* will give you all the options for plotting. For completeness we give all the options below

```
Usage: /Users/koning/bin/plot <projectile> <element> <A> <channel> <optional:
    isomer> <options>

projectile: n g p d t h a
element   : element symbol, e.g. Nb, NB, nb, or nB
A         : mass number, e.g. 93 or 093
channel   : reaction channels: tot totlow totlin el non nn nn1...nn9
            n2n n3n ng nglow nghigh nf nflow np nplog nd nt nh na nalog nnp nna
            nxg nxn nxp nxd nxt nxh nxa or ZZZAAA of residual
            For other projectiles, use e.g. pn p2n, etc.
isomer    : g or m
display   : plot    - screen display with Gnuplot [default]
            print   - postscript file with Gnuplot
exp flag  : Eonly   - plot/print only if experimental data available [default]
            Ealways - plot/print always
lib flag  : Lonly   - plot/print only if library data available
            Lalways - plot/print always [default]
log flag  : log     - logarithmic scale
            nolog   - linear scale [default]
maximum   : maximum - include largest experimental data point
          : nomaximum- use reasonable scales [default]
-Ebeg     : begin energy in MeV
-Eend     : end energy in MeV
-xsbeg    : begin cross section in mb
-xsend    : end cross section in mb
-calc     : directory with calculated cross sections
-libs     : directory with libraries
```

```
-newpath : directory with new nuclear data library files
-num     : number of random run
-maxexp  : maximum number of experimental data sets
png      : png     - Make png file
         : nopng   - do not make png file [default]
quality  : quality - include quality flag in legend
         : noquality- do not include quality flag in legend [default]
unc      : unc     - include uncertainty band [default]
           nounc   - exclude uncertainty band
talys    : talys   - include TALYS calculation
           notalys - exclude TALYS calculation [default]
group    : group   - use groupwise data for resonance channels [default]
         : nogroup - do not use groupwise data for resonance channels
high     : high    - compare with high energy data libraries
         : nohigh  - do not compare with high energy data libraries [default]
large    : large   - large fonts [default]
         : small   - small fonts
clean    : clean   - clean up plot parameter file [default]
         : noclean - do not clean up plot parameter file
tendl    : tendl   - include TENDL data in plot
         : notendl - do not include TENDL data in plot
endfb    : endfb   - include ENDFB data in plot [default]
         : noendfb - do not include ENDFB data in plot
jendl    : jendl   - include JENDL data in plot [default]
         : nojendl - do not include JENDL data in plot
jeff     : jeff    - include JEFF data in plot [default]
         : nojeff  - do not include JEFF data in plot
cendl    : cendl   - include CENDL data in plot [default]
         : nocendl - do not include CENDL data in plot
eaf      : eaf     - include EAF data in plot [default]
         : noeaf   - do not include EAF data in plot
irdff    : irdff   - include IRDFF data in plot [default]
         : noirdff - do not include IRDFF data in plot
ibandl   : ibandl  - include IBANDL data in plot [default]
         : noibandl - include IBANDL data in plot
jendlad  : jendlad - include JENDL/AD data in plot [default]
         : nojendlad- do not include JENDL/AD data in plot
iaea     : iaea    - include IAEA data in plot [default]
         : noiaea  - do not include IAEA data in plot
new      : new     - include new data in plot [default]
         : nonew   - include new data in plot
subentry : legend with subentry instead of author [default:not used]

 The first 4 arguments are in fixed order: projectile, element, mass, reaction.
 After that the order is arbitrary

Examples : plot n Nb 93 nn
           plot N Nb 93 non
           plot p Nb 93 pn
           plot p Nb 93 040090
           plot n Nb 93 nn m
           plot n Nb 93 nn m print
           plot n Nb 93 nn m Plot Eonly
           plot n Nb 93 nna Eonly Lonly
```

C. All options for run.bnl

Giving simply *run.bnl* will give you all the options. For completeness we give all the options below

```
      run.bnl-1.0 (Version September 19 2019) (C) Copyright 2019 Arjan Koning,
          All Rights Reserved

run.bnl                                                              run.bnl

NAME
      run.bnl - Script to make input file for BNL checking codes and running them

SYNOPSIS
      run.bnl [option] [value]

DESCRIPTION
      With run.bnl, various modules of the BNL checking codes can be invoked in,
          or left out of,
      an input file. Specific input options can be set.

   Operation mode:

      Usage: run.bnl [option] [value]

      Some option flags should be accompanied by a value

      Examples: run.bnl -file Nb093-n.tendl (obligatory: ENDF-6 file)
                run.bnl -file Nb093-n.tendl -bin /home/raynal/bin/ -nopsyche
                        (special BNL bin directory, skip PSYCHE)

   General options: file to be processed, BNL code version, pathnames, etc.
```

```
      -file [ENDF file]
           ENDF filename [no default]

      -bin [path name]
          Full pathname for BNL binaries [default: none]

      -ext [name]
          Extension to executables, e.g. C [default: blank]

      -outmode [value]
              mode for output filename, 1: checkr.out 2: 'ENDF file'.checkr [
                  default: 1]

      -[no]clean
                Clean up working directories [default: -noclean]

  Specific BNL modules

       -[no]checkr
                 Run CHECKR [default: -checkr]

       -[no]fizcon
                 Run FIZCON [default: -fizcon]

       -[no]psyche
                 Run PSYCHE [default: -psyche]

       -[no]inter
                 Run INTER [default: -inter]

  BNL input parameters

       -[no]deviant
                 Deviant point test for FIZCON [default: -nodeviant]

       -[no]sumup
                 Sumup test for FIZCON [default: -nosumup]

       -err [error]
            Allowable fractional error for FIZCON [default: 0.01]

AUTHOR
      run.bnl was written by Arjan Koning.
```

D. All options for run.prepro

Giving simply *run.prepro* will give you all the options. For completeness we give all the options below

```
      run.prepro-1.0 (Version September 30 2019) (C) Copyright 2019 Arjan Koning,
           All Rights Reserved

/Users/koning/bin/run.prepro

run.prepro                                                          run.prepro

NAME
      run.prepro - Script to make input file for PREPRO, run PREPRO and post-
          process the results

SYNOPSIS
      run.prepro [option] [value]

DESCRIPTION
      With run.prepro, various modules of PREPRO can be invoked in, or left out
          of,
      a PREPRO input file. Specific input options can be set.

   Operation mode:

      Usage: run.prepro [option] [value]

      Some option flags should be accompanied by a value

      Examples: run.prepro -file Nb093-n.tendl (obligatory: ENDF-6 file)
```

```
            run.prepro -file Nb093-n.tendl -bin /home/raynal/bin/ -nogroupie -
                xsmin 1.e-12 -mergefile Nb093-n.pendfN
                    (special PREPRO bin directory, skip GROUPIE, minimal
                        cross section of interest: 1.e-12 b, merge with other
                        file)

General options: file to be processed, PREPRO version, pathnames, etc.

    -file [ENDF file]
         ENDF filename [no default]

    -mergefile [ENDF file]
              Second ENDF filename to merge with [default: none]

    -bin [path name]
        Full pathname for PREPRO binaries [default: none]

    -ext [name]
        Extension to executables, e.g. C [default: blank]

    -[no]here
             Run PREPRO here or in other (automatically created) directory [
                 default: -here]

    -preprodir [path name]
            Full pathname for PREPRO results [default: none]

    -[no]preprorun
              Run PREPRO or just produce the input file [default: -preprorun]

    -[no]clean
              Clean up working directories [default: -noclean]

General PREPRO parameters

     -temp [temperature]
          Temperature [default: 293.16]

Specific PREPRO modules

     -[no]linear
              Run LINEAR [default: -linear]

     -[no]recent
              Run RECENT [default: -recent]

     -[no]sigma1
              Run SIGMA1 [default: -sigma1]

     -[no]sixpak
              Run SIXPAK [default: -sixpak]

     -[no]merger
              Run MERGER [default: -merger]

     -[no]activate
```

```
                Run ACTIVATE [default: -activate]

       -[no]dictin
                Run DICTIN [default: -dictin]

       -[no]groupie
                Run GROUPIE [default: -groupie]

       -[no]evalplot
                Run EVALPLOT [default: -evalplot]

       -[no]complot
                Run COMPLOT [default: -nocomplot]

      -file2 [ENDF file]
           ENDF filename for COMPLOT [no default]

      -name1 [name]
           Name of first library for COMPLOT [no default]

      -name2 [name]
           Name of second library for COMPLOT [no default]

  PREPRO input parameters

       -[no]keep
                Keep original data points in LINEAR [default: -keep]

       -errmax [error]
              Allowable fractional error for error law [default: 1.e-4]

       -xsmin [value]
                Minimum cross section of interest (barns) for LINEAR [default: 1.
                    e-10]

       -[no]urrbroad
                Broaden URR [default: -nourrbroad]

       -group [groupie number]
              Integer to set group structure for groupie [default: 12]

AUTHOR
      run.prepro was written by Arjan Koning.
```

E. All options for run.njoy

Giving simply *run.njoy* will give you all the options. For completeness we give all the options below

```
      run.njoy-1.0 (Version June 26 2019) (C) Copyright 2019 Arjan Koning, All
          Rights Reserved

/Users/koning/bin/run.njoy

      Starting time: 11-28-2019, 04:43:47 PM

run.njoy                                                          run.njoy

NAME
      run.njoy - Script to make input file for NJOY, run NJOY and post-process
          the results

SYNOPSIS
      run.njoy [option] [value]

DESCRIPTION
      With run.njoy, various modules of NJOY can be invoked in, or left out of,
      an NJOY input file. The NJOY executable and specific input options can be
          set.

   Operation mode:

      Usage: run.njoy -file <endf file> [option] [value]

      Some option flags should be accompanied by a value
```

```
    Examples: run.njoy -file Nb093-n.tendl (obligatory: ENDF-6 file)
              run.njoy -file Nb093-n.tendl -bin /home/raynal/bin/ -version
                  njoy99.mine -nogaspr -bins 24
                       (special NJOY executable, skip GASPR, use 24 probability
                          bins for PURR)

General options: file to be processed, NJOY version, pathnames, etc.

    -file [ENDF file]
         ENDF filename [no default]

    -bin [path name]
        Full pathname for NJOY binary [default: none]

    -version [NJOY name]
            Name of NJOY binary [default: njoy-2016.47]

    -[no]here
             Run NJOY here or in other (automatically created) directory [
                 default: -here]

    -njoydir [path name]
            Full pathname for NJOY results [default: none]

    -[no]njoyrun
               Run NJOY or just produce the input file [default: -njoyrun]

    -outname [name]
            Name of output file, 1: nuclide-proj.ace
                                 2: endffile.ace etc. [default: 1]

    -[no]clean
              Clean up files [default: -clean]

General NJOY parameters

    -[no]binary
             Use binary files [default: -nobinary]

     -temp [temperature]
          Temperature [default: 293.16]

     -libid [identifier]
           Library indentifier [default: TENDL-2019]

     -mcnpext [ext. code ]
            Extension for ACE file [default: .00]

Specific NJOY modules

     -[no]pointwise
              Do NJOY run for pointwise data only [default: -nopointwise]

     -[no]moder
               Run MODER [default: -moder]
```

```
     -[no]reconr
               Run RECONR [default: -reconr]

     -[no]broadr
               Run BROADR [default: -broadr]

     -[no]unresr
               Run UNRESR [default: -unresr]

     -[no]heatr
               Run HEATR [default: -heatr]

     -[no]gaspr
               Run GASPR [default: -gaspr]

     -[no]viewr
               Run VIEWR [default: -viewr]

     -[no]purr
               Run PURR [default: -purr]

     -[no]acer
               Run ACER [default: -acer]

NJOY input parameters

     -iprint [value]
            Print option (0=min, 1=max) [default: 0]

     -tempr [temperature]
         Reconstruction temperature for RECONR [default: 0]

     -errr [error]
         Fractional tolerance for RECONR [default: 0.001]

     -errmaxr [error]
         Fractional tolerance with RI criterion for RECONR [default: 10*errr]

     -errintr [error]
         Maximum RI error for RECONR [default: errr/20000]

     -errmult [value]
         Multiplier for NJOY defaults of errmaxr and errintr [default: 1]

     -thnmax [value]
             Maximum energy for broadening and thinning for BROADR [default:
                 1.]

     -errthnb [error]
             Fractional tolerance for thinning for BROADR [default: 0.001]

     -errmaxb [error]
         Fractional tolerance with RI criterion for BROADR [default: 10*
             errthnb]
```

```
       -errintb [error]
           Maximum RI error for BROADR [default: errthnb/20000]

       -sigz [value]
            sigma0 value for unresolved [default: 1.e10]

       -nbin [value]
            Number of probability bins for PURR [default: 20]

       -nladr [value]
             Number of resonance ladders for PURR [default: 64]

       -[no]allkerma
                   partial kermas for all reactions or for 442, 443 and 444 only
                        [default: -noallkerma]

       -[no]local
                 gamma rays transported or deposited locally [default: -local]

AUTHOR
      run.njoy was written by Arjan Koning.
```

F. From Fortran-77 to Fortran-95

To make TALYS and its satellite codes more portable, readable, and so memory-efficient that even more capabilities could be included, it was decided to rewrite the code into Fortran-95. In 2020, the total number of programming lines in TALYS (version 1.96) was close to 120000, and that of its main satellite codes, the ENDF formatting code TEFAL, 18000 lines, and the uncertainty code TASMAN, also 18000 lines. Overall, this is more than 150000 lines of Fortran-77 source code. Fortunately, most of the code has been programmed in a systematic way, one of the advantages of having a very limited number of programmers. This made it possible to write a Fortran-77 to Fortran-95 converter to transform one readable subroutine into another readable subroutine. This had to be applied with care. An automated step should be used to do the "dirty" work, but every subroutine, function etc had to be checked and cleaned up here and there to obtain the desired results. Also, the translation took place systematically for a restricted, but important, set of Fortran-95 features. The most essential step is the possibility to use dynamic memory allocation, and the simultaneous riddance of common blocks, which allows TASMAN and TEFAL to be included in the main TALYS code in the future, but also other clean up steps have been taken. We consider this first translation as the major one, at least in terms of look and feel of the individual subroutines and functions of the code. In the following sections we list the systematic changes from Fortran-77 to Fortran-95 that were applied to TALYS. We emphasize that this has only been done to the subroutines for which this was possible (in practice, those written by me). We have not attempted to translate legacy codes, which are included in TALYS because they are indispensable, since that would result in source code that is even more unreadable that the original source. The description below takes TALYS as an example, but has also been applied to other software such as TASMAN and TEFAL.

F.1 From common block to module

The backbone of the TALYS-1.x source codes is one large common block file, talys.cmb, which contains all variables that are shared between the different subroutines of the entire code. As a first

step of modernization, we have changed this into one large module, A1_talys_mod.f90. "True" modularity may then be obtained in a next step where this large module is replaced by various smaller modules which only contain the data needed for a particular part of the program. A main advantage of the module is that it allows for defensive programming. In every subroutine which requires global data, one can use the 'use only' construct in every subroutine. This means that explicitly **only** those variables of the global modules that appear in the subroutine are **use**d, instead of including the whole module (we used to do this with 'include "talys.cmb"' in TALYS-1.96). As an additional readability feature, since every variable in A1_talys_mod.f90 is explained, the description is copied as well to the subroutine. Next comes the local data. After the obligatory 'implicit none' comes the declaration of each local variable, line by line, and including a description. In basically every subroutine, the first included variables are 'sgl' and 'dbl' from the A1_kinds_mod.f90 module specifying single and double precision. As an example, this is the declaration part of comptarget.f90,

```
..Global data...
  use A0_kinds_mod, only: & ! Definition of single and double precision
              sgl, &          ! single precision kind
              dbl             ! double precision kind
  use A1_talys_mod, only: & ! All global variables
              numfact, & ! number of terms for factorial logarithm
              numhill, & ! maximum number of Hill - Wheeler points
              numin, &   ! maximum number of neutrons in channel descr.
...
..Local data...
  implicit none
  logical   :: elastic   ! designator for elastic channel
  integer   :: parspin2i ! 2 * particle spin for incident channel
  integer   :: pspin2i   ! 2 * spin of particle (usually) for incident ch.
...
```

All this means that before the first executable statement in a subroutine is encountered, which may be well beyond half of the subroutine, **all** variables have been declared and provided with a comment explaining their meaning, using one declaration per line. In my view, this improves readability and minimizes errors. Note that we have used *A0...*, *A1...*, etc. as module names so that all modules and subroutines are compiled in the required order.

F.2 Dynamic memory allocation

As a first step towards the use of full dynamic memory allocation, we have made every array of A1_talys_mod.f90 allocatable. As the first step in TALYS we call alloc.f90 which allocates all arrays to its static sizes, with values taken directly from the TALYS-1.96 common blocks. At the end of TALYS, we call dealloc.f90 which deallocates them all. This means that there is in practice no change (yet) compared to TALYS-1.96, apart from the fact that all arrays are allocated at runtime. In the future, two essential refinements will follow:

- dynamic declaration of arrays on the basis of the values given on input, instead of the maximum possible value for each dimension, as is done now
- allocation of the array just before its first use in the program, and deallocation just after its last use, i.e. returning memory for new purposes as soon as possible.

Both the above actions will lead to more efficient memory management. Note that dynamic memory allocation is known to slow down the code, but I think this is less important than the gained efficiency, portability and readability.

F.3 Do loops

We consistently changed the 'do continue' loop construct by the 'do enddo' loop construct. In the process, this leads to

- no more use of numbered labels in continue statements,
- use of 'cycle' to jump to the next case in the loop instead of a goto statement,
- use 'exit' to jump out of the loop instead of a goto statement.

Example:

```
      do 310 type=9,11
        if (type.ne.10.and..not.flagurrnjoy) goto 310
...
  310 continue
```

becomes

```
  do type = 9, 11
    if (type /= 10 .and. .not. flagurrnjoy) cycle
...
  enddo
```

Most numeric labels are now gone. There are still a few goto statements left, since at he moment they are considered as more efficient or readable as the alternative. The other use of labels are the 'err=' and 'end=' statements in read operations. They have been replaced by the more versatile 'iostat='. The numbered format statement is only used in legacy subroutines (like ECIS).

F.4 Arrays

The only systematic change we make for arrays at the moment are those for array initializations.

```
      do 130 k=0,numlev
        do 130 i=0,numlev
          do 130 Nix=0,numN
            do 130 Zix=0,numZ
              bassign(Zix,Nix,i,k)=' '
              conv(Zix,Nix,i,k)=0.
  130 continue
```

is translated by our translation code into

```
do k = 0, numlev
  do i = 0, numlev
    do Nix = 0, numN
      do Zix = 0, numZ
        bassign(Zix, Nix, i, k) = ' '
        conv(Zix, Nix, i, k) = 0.
      enddo
    enddo
  enddo
enddo
```

but such loops have all been replaced by the more compact

```
bassign = ' '
conv = 0.
```

where the Fortran-95 compiler knows that this concerns all elements of these multidimensional arrays.

Other array operations which are allowed by Fortran-95 will have to be considered case by case. At some point, this will be done for one such case, systematically throughout the whole source code. This will have to be done manually.

F.5 Free format and other cosmetics

The following lines are from comptarget.f90,

```
Wab = 1.
parspin2i = int(2. * parspin(k0))
pspin2i = spin2(k0)
```

showing that we use more spacing in our programming. Also we no longer use the first 6 blank characters which were reserved in Fortran-77. Similar spaces are used in do loops and multi-dimensional arrays:

```
do updown = - 1, 1
  do l = 0, lmaxinc
    Tjlnex(k0, Ltarget, updown, l) = Tjlinc(updown, l)
  enddo
enddo
```

In this piece of coding,

```
! If the parity of the target nucleus is equal (unequal) to the parity
! of compound nucleus, i.e. pardif=0(1), the l-value must be even (odd).
!
          if (flagwidth .or. flagcompang .or. flagurr) then
            if (mod(l, 2) /= pardif) cycle
            updown = (jj2 - l2) / pspin2i
            Tinc = Tjlinc(updown, l)
            tnumi = tnumi + 1
          endif
```

we also see the spacing around '.or.' and that we use the more mathematical notation for (in)equalities such as '/=' instead of '.ne.'. Note also that we consistently use '!' instead of 'c' for comments, which is mandatory for free format compilation of .f90 files.

Since we can now use a line length of 132 characters, several continuation characters are no longer necessary, i.e.

```
          if (flagfission.and.nfisbar(Zcomp,Ncomp).ne.0)
     +      call tfission(Zcomp,Ncomp,nex,J2,parity)
```

becomes

```
    if (flagfission .and. nfisbar(Zcomp, Ncomp) /= 0) call tfission....
```

and if it goes beyond 132 characters we use the continuation character as in

```
angfac = (J2 + 1) * (jj2prime + 1) * (jj2 + 1) * (l2 + 1) * &
  (l2prime + 1) / fourpi
```

We also use the scientific notation for numbers in exponential notation, which gets rid of the '1p', '0p' prefixes, which we used to the less readable exponential notation with leading zeroes. Hence, '1p,e12.5,0p' now becomes 'es12.5' throughout the code.

F.6 Quality classes

The core of TALYS has been programmed in a systematic way, and is considered by many users as quite readable. To extend the capabilities of TALYS, also subroutines by other authors, and much older subroutines, have been adopted. Often, they are put in a safe spot inside the TALYS structure and do not interfere with the "core" subroutines and modules. Examples of these "other" subroutines are the ECIS-code (Raynal) for optical model and coupled-channels calculations, PREPRO (Cullen) for reconstruction of pointwise data out of resonance parameters, RACAP (Goriely and Xi) for direct capture, MOM (Bauge) for the microscopic optical model, WKB (Capote) for fission transmission coefficients, and FOLDALPHA (Goriely) for the microscopic alpha optical model potential. Such codes have not been translated from Fortran-77 to Fortran-95.

There are also differences between the various core subroutines of TALYS. Depending on the complexity of the task, the subroutines or functions are more or less nicely programmed. Ideally, TALYS should consists of a large collection of "pure" subroutines and functions, with a very clear communication between them. What we have done is to give, totally subjective, a quality assignment to each subroutine and function. If a task of a function is very simple and clear, e.g. a Fermi Gas distribution, one would expect that function to be pure, i.e. have no unintended side effects, and to be constructed in a style which directly makes the task of the function clear, as if the equations are directly implemented from the nuclear physics book. One would expect and hope that such functions and subroutines never need to be touched anymore, and that they could be adopted in any other code with a simple copy action.

At the other extreme we have the legacy codes programmed in uppercase, of which every 5th line seems to contain a GOTO statement. For reference to the user and the author, at the top of each subroutine a quality class is given. The classification is:

- C black-box 60's, 70's, 80's style programming, of which it is known that the output is indispensable to TALYS, but for which the programming details are only known to the original author,
- B More readable programming, probably from fellow TALYS developers, but not in the general style of the other TALYS subroutines
- A Standard TALYS style subroutine, but not yet modularized to the extreme, probably containing still too many tasks in one and the same subroutine
- AA More structure TALYS subroutine with very clearly identified and logically structured tasks,
- AAA Pure subroutines or functions
- AAAA Possible future quality class that is even higher
- AAAAA ....

Obviously, the idea is that in the future upgrades to higher classes can take place.

G. GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

G.1 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 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.

G.2 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.
   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.

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 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 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.

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 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 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 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 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.
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