AUTOTALYS

Tools for TALYS: systematic nuclear data evaluation, TENDL nuclear data library, optimization to experimental data and more

Arjan Koning

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/.

October 2025

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/Bruyères-le-Châtel and Los Alamos National Laboratory on the development and computational implementation of nuclear reaction models. He has led several 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 for the optical model and pre-equilibrium reactions, and initiating the '3 T's': the TALYS nuclear model code, Total Monte Carlo uncertainty propagation and the TENDL nuclear data library. As of 2025, his h-index is 62 with more than 25 000 citations.
Although Arjan is currently in a managerial role, he aims to keep his scientific creativity alive by maintaining and extending TALYS plus all software that emerges from that. Pleas from his friends to also spend time on other things are sometimes honoured.

Preface

AUTOTALYS 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 from EXFOR for a whole range of projectile-nuclide combinations, astrophysical reaction databases 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 aimed at general use, also for non-TALYS or non-TENDL users.

License, contact and reference

As mentioned on the first page and in the source code, AUTOTALYS falls in the category of MIT License software.

In addition to the MIT *terms* I have a *request*:

- When AUTOTALYS 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, AUTOTALYS, IAEA NDS Document Series IAEA(NDS)-xxx, April 2023

The webpage for AUTOTALYS 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**,
- Jean-Christophe Sublet for his feedback on the results and overall structure of autotalys, especially for the production of TENDL,
- A few beta users of the autotalys systems:
  - Henrik Sjoestrand and Erwin Alhassan at Uppsala University,
  - Satoshi Chiba and co-workers at Tokyo Institute of Technology,
  - Shin Okumura and Georg Schnabel at IAEA,
  - Bret Beck and Caleb Mattoon at LLNL.

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 AUTOTALYS tutorial is a cookbook for performing the tasks that have led to many publications on TALYS, TENDL [2, 3] and Total Monte Carlo [4] 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 and MacOS systems, **autotalys** automates a whole sequence of nuclear data generation steps into one command. A mimimal example 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.tendl2023
-sdefault -s20 -s60 -acf -eaf -mt
```

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

We stress **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 the optimal nuclear model parameters, production of astrophysical reaction rates, comparisons with EXFOR[5],

plotting of your latest TALYS results and production of nuclear data libraries in more modern or easier formats.

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 or manual evaluation of one particular reaction channel for one nuclide, while this is how most of nuclear data evaluation has been organized in the past decades. This tutorial describes what needs to be set up to make nuclear data evaluation completely reproducible, efficient, systematic and automated. We say "needs to be" because this standardization and automation is still ongoing.

As specific components and features of AUTOTALYS 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 or x-y-dy 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 someone releases a versatile ENDF API which does the same)
- the entire **autotalys** software system [3] to produce nuclear data libraries
- collection of scripts to automatically run, and analyze the output of, codes like PREPRO[6], NJOY[7], FUDGE[8], CHECKR, FIZCON, PSYCHE and INTER[9].
- step-by-step outline of the production of the TENDL nuclear data library[3],
- 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 [3]: 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 [3]: 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 the random TALYS runs to produce covariance matrices. Additionally, TEFAL can be included in that loop for Total Monte Carlo uncertainty propagation.
- EXFORTABLES[10]: A code to produce an experimental nuclear reaction database based on EXFOR (needed until a versatile EXFOR API is released)
- RESONANCETABLES[11]: A code to produce a database for thermal cross sections, MACS and average resonance parameters
- AUTOENDF[3]: A collection of bash shell scripts for the automatic testing and processing of ENDF-6 formatted nuclear data libraries with the BNL checking codes CHECKR, FIZCON, PSYCHE and INTER, the PREPRO checking and processing codes, and the NJOY processing code.
- TARES[12]: a code by Dimitri Rochman for neutron resonance parameters and its uncertainty quantification
- TAFIS [3]: a code by Dimitri Rochman for the average number of fission neutrons and its uncertainty quantification
- TANES [3]: a code by Dimitri Rochman for prompt fission neutron spectra and its uncertainty

quantification

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

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 directory-structured form,

Chapter 4: the AUTONORM tool, to normalize TALYS results automatically to evaluated or experimental data,

Chapter 5: The **autotalys** system, including how to create TENDL,

Chapter 6: The **autotalys** script, and several examples for automated parameter adjustment,

Chapter 7: Conclusions and ideas for future work.

Several appendices outlining the options for the scripts.

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 key-value 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 [13], DECE [14] and TEFAL[3]. There are more codes which can *read* ENDF-6 format. Examples are ENDVER [15] (directly coupled with ZVVIEW [16]), JANIS [17], FUDGE [8], ENDFtk, BNL checking codes [9], PREPRO [6] and NJOY [7]. ENDFTABLES performs this task as a stand alone code to provides a cut in the middle: it makes a large database with observables, i.e. turns an ENDF-6 file into data that can be directly compared with TALYS or EXFOR. I expect that ENDFTABLES will be replaced in the future by an API (ENDFtk may be a good candidate) 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 header containing the basic information about the reaction. The header is written in the so-called YANDF format as detailed in an Appendix. 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 autotalys 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

```
# header:
#   title: U238(n,2n)U237 cross section
#   source: ENDF
#   creator: Arjan Koning
#   date: 2023-10-27
#   format: YANDF-0.1
# endf:
#   library: endfb8.0
#   author: IAEA Consortium
#   year: 2014
# target:
#   Z: 92
#   A: 238
#   nuclide: U238
# reaction:
#   type: (n,2n)
#   Q-value [MeV]: -6.153000E+00
#   E-threshold [MeV]:  6.179071E+00
#   ENDF_MF: 3
#   ENDF_MT: 16
# residual:
#   Z: 92
#   A: 237
#   nuclide: U237
# datablock:
#   quantity: cross section
#   columns: 4
#   entries: 37
##       E             xs            xslow          xsup
##     [MeV]          [mb]           [mb]           [mb]
   6.179071E+00   0.000000E+00   0.000000E+00   0.000000E+00
   6.300000E+00   5.877499E+00   5.439617E+00   6.315382E+00
   6.400000E+00   3.037020E+01   2.810757E+01   3.263282E+01
   6.500000E+00   7.819099E+01   7.236565E+01   8.401634E+01
   7.000000E+00   5.073350E+02   4.802318E+02   5.344382E+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,

```
# header:
#   title: U238(n,f) total nubar
#   source: ENDF
#   creator: Arjan Koning
#   date: 2023-10-27
#   format: YANDF-0.1
# endf:
#   library: endfb8.0
#   author: IAEA Consortium
#   year: 2014
# target:
#   Z: 92
#   A: 238
#   nuclide: U238
# reaction:
#   type: (n,f)
#   ENDF_MF: 1
#   ENDF_MT: 452
# datablock:
#   quantity: total nubar
#   columns: 2
#   entries: 44
##       E            nubar
##     [MeV]            []
   1.000000E-11   2.443040E+00
   5.500000E-03   2.443130E+00
   6.000000E-03   2.444120E+00
   1.500000E-02   2.444280E+00
   2.000000E-02   2.445340E+00
   3.000000E-02   2.445510E+00
.....................
   1.600000E+01   4.746000E+00
   3.000000E+01   6.414000E+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[16], JANIS[17], TEFAL[3], EMPEND[13], ENDVER[15] 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

```
FY
a-iaea.2019
a-ibandl
a-jendl5.0
a-tendl.2023
d-iaea.2019
d-ibandl
d-jendl5.0
d-tendl.2023
g-endfb8.0
g-iaea.2019
```

```
g-iaea.pd
g-jendl5.0
g-tendl.2023
h-endfb8.0
h-iaea.2019
h-tendl.2023
n-cendl3.2
n-eaf.2010
n-endfb8.0
n-iaea.2019
n-irdff2.0
n-jeff3.3
n-jendl5.0
n-tendl.2023
p-endfb8.0
p-iaea.2019
p-ibandl
p-jendl5.0
p-tendl.2023
t-endfb8.0
t-tendl.2023
```

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.2023
```

only makes tables for the neutron file for ^{242m}Am of TENDL-2023, 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 autotalys 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/jendl5.0/
n/Fe054/plots/
n/Fe054/tendl.2023/
```

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 [10], 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

```
# header:
#   title: "Fe54(n,p)Mn54 cross section"
#   source: EXFORtables
```

```
#   user: Arjan Koning
#   date: 2023-07-14
#   format: YANDF-0.1
# exfor:
#   author: Vanloef
#   year: 1961
#   subentry: 30221006
#   X4 reaction: 26-FE-54(N,P)25-MN-54,,SIG
#   X4 source: database version 2023-04-29, C5 file by Viktor Zerkin
# target:
#   Z: 26
#   A: 54
#   nuclide: Fe54
# reaction:
#   type: "(n,p)"
# residual:
#   Z: 25
#   A: 54
#   nuclide: Mn54
# datablock:
#   quantity: "cross section"
#   columns: 4
#   entries: 3
##       E             dE             xs             dxs
##     [MeV]          [MeV]          [mb]           [mb]
   2.600000E+00   2.000000E-01   1.500000E+02   2.000000E+01
   3.300000E+00   1.000000E-01   1.900000E+02   2.000000E+01
   3.600000E+00   1.000000E-01   2.100000E+02   1.500000E+01
# reference:
#   author: J.J.Van Loef
#   title: Activation cross sections for (n,p) reactions in some medium weight nuclei with
#   journal: Jour: Nuclear Physics, Vol.24, p.340 (1961)
```

where the data are again in YANDF format, see the Appendix. 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[10].

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-5.0 neutron data file of ^{54}Fe we have

```
n/Fe054/jendl5.0/check/
n/Fe054/jendl5.0/files/
n/Fe054/jendl5.0/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/jendl5.0/tables/angle/
n/Fe054/jendl5.0/tables/damage/
n/Fe054/jendl5.0/tables/info/
n/Fe054/jendl5.0/tables/resonance/
n/Fe054/jendl5.0/tables/xs/
```

Cross sections

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

```
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT001-G069.jendl5.0
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT001-G171.jendl5.0
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT001-G700.jendl5.0
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT001.jendl5.0
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT002-G069.jendl5.0
.............
n/Fe054/jendl5.0/tables/xs/n-Fe054-MT207.jendl5.0
```

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/jendl5.0/tables/angle/n-Fe054-MT002-Eang000.001.jendl5.0
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT002-Eang000.010.jendl5.0
.....
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT002-Eang018.000.jendl5.0
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT002-Eang020.000.jendl5.0
```

while the inelastic angular distributions per incident energy, in the case of JENDL5.0 for the first 19 discrete levels, are given in

```
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT051-Eang001.435.jendl5.0
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT051-Eang002.000.jendl5.0
.....
n/Fe054/jendl5.0/tables/angle/n-Fe054-MT069-Eang020.000.jendl5.0
```

General info

We store the general information of the evaluation in

```
n/Fe054/jendl5.0/tables/info/n-Fe054-MF01-MT451.jendl5.0
```

Resonance parameters

For neutrons on ^{54}Fe for JENDL5.0, this looks as follows

```
n/Fe054/jendl5.0/tables/resonance/n-Fe054-MF02-MT151.jendl5.0
```

For TENDL-2023 also the covariance data in the resonance range are available, so in that case the directory is

```
n/Fe054/tendl.2023/tables/resonance/n-Fe054-MF02-MT151.tendl.2023
n/Fe054/tendl.2023/tables/resonance/n-Fe054-MF32-MT151.tendl.2023
```

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-2023, this looks as follows

```
n/Fe054/tendl.2023/tables/residual/n-Fe054-rp016030.tendl.2023
n/Fe054/tendl.2023/tables/residual/n-Fe054-rp016031.tendl.2023
.....
n/Fe054/tendl.2023/tables/residual/n-Fe054-rp026055.tendl.2023
```

Damage cross sections and DPA

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

```
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT301-G069.jendl5.0
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT301-G171.jendl5.0
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT301-G700.jendl5.0
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT301.jendl5.0
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT302-G069.jendl5.0 etc.
.........................
n/Fe054/jendl5.0/tables/damage/n-Fe054-MT447.jendl5.0
```

Fission neutron observables

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
```

Fission yields

A separate directory, call *FY/* with all info related to fission yields is available.

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/jendl5.0/files/ directory looks as follows

```
n/Fe054/jendl5.0/files/n-Fe054.jendl5.0
n/Fe054/jendl5.0/files/n-Fe054.jendl5.0.ace.gz
n/Fe054/jendl5.0/files/n-Fe054.jendl5.0.gendf.gz
n/Fe054/jendl5.0/files/n-Fe054.jendl5.0.pendf.gz
n/Fe054/jendl5.0/files/n-Fe054.jendl5.0.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[9], 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

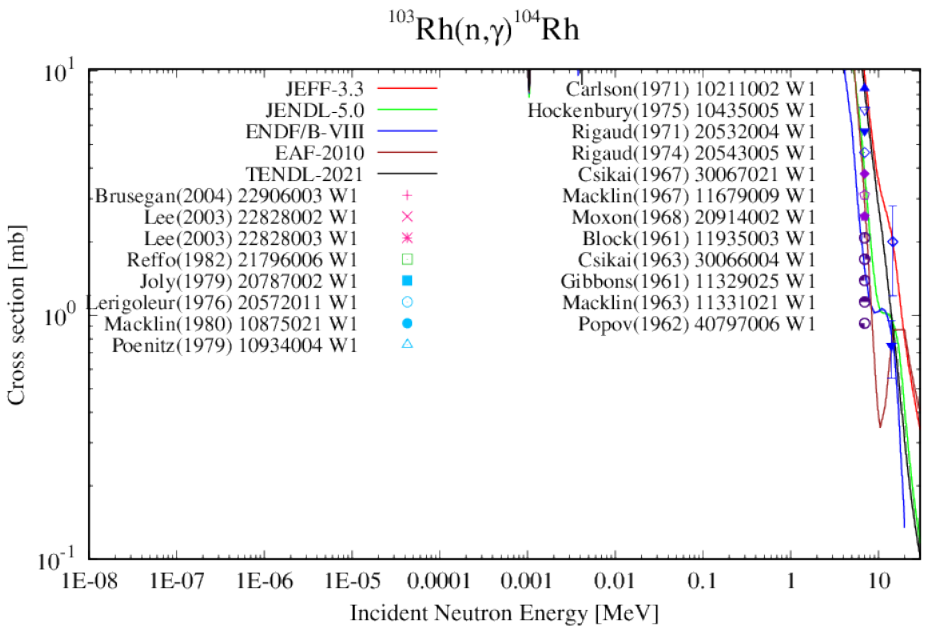
In the process of analyzing and decomposing the data, we also process the files with PREPRO and NJOY into PENDF and ACE libraries and the resulting libraries directory is huge. Therefore, for more practical applications, e.g. direct use for TENDL production in autotalys, a *libfilter* script has been 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

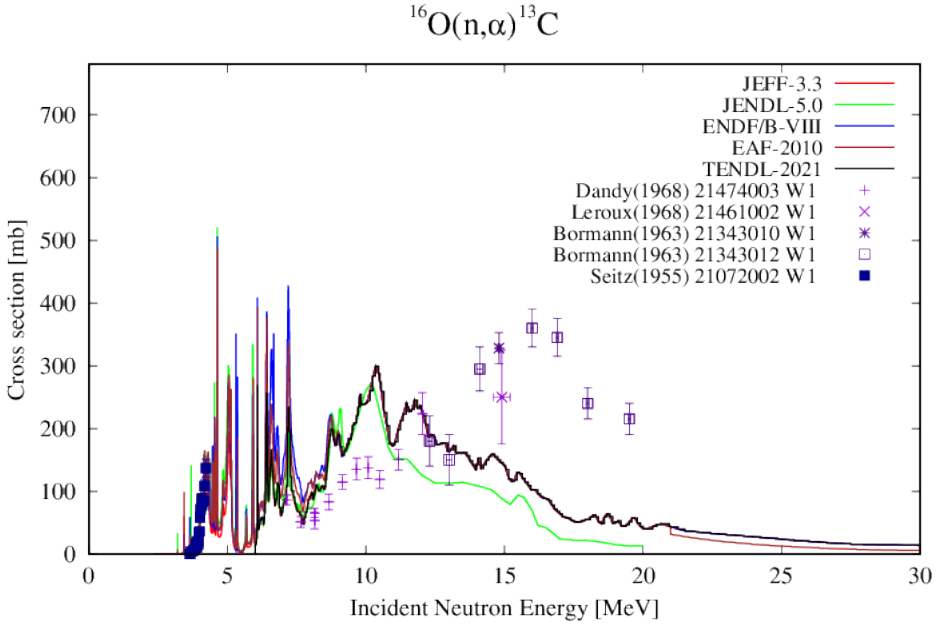
```
libfilter -year -noeaf
```

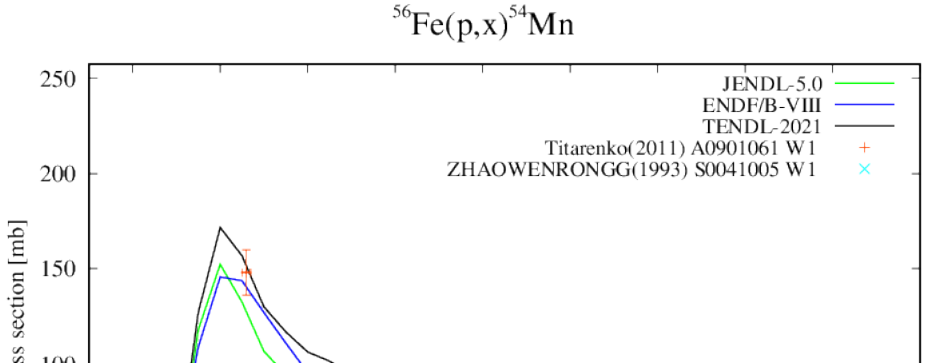
which deletes all data for nuclides with a half life below one year and that of the EAF library. The *libfilter* command to go from the full *libraries.all* to the *libraries* that is needed for autotalys production the following command was used

```
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-5.0
ENDF/B-VIII
EAF-2010
TENDL-2021
Brusegan(2004) 22906003 W1
Lee(2003) 22828002 W1
Lee(2003) 22828003 W1
Reffo(1982) 21796006 W1
Joly(1979) 20787002 W1
Lerigoleur(1976) 20572011 W1
Macklin(1980) 10875021 W1
Poenitz(1979) 10934004 W1
Carlson(1971) 10211002 W1
Hockenbury(1975) 10435005 W1
Rigaud(1971) 20532004 W1
Rigaud(1974) 20543005 W1
Csikai(1967) 30067021 W1
Macklin(1967) 11679009 W1
Moxon(1968) 20914002 W1
Block(1961) 11935003 W1
Csikai(1963) 30066004 W1
Gibbons(1961) 11329025 W1
Macklin(1963) 11331021 W1
Popov(1962) 40797006 W1
Cross section [mb]
Incident Neutron Energy [MeV]


16O(n,α)13C
JEFF-3.3
JENDL-5.0
ENDF/B-VIII
EAF-2010
TENDL-2021
Dandy(1968) 21474003 W1
Leroux(1968) 21461002 W1
Bormann(1963) 21343010 W1
Bormann(1963) 21343012 W1
Seitz(1955) 21072002 W1
Cross section [mb]
Incident Neutron Energy [MeV]


56Fe(p,x)54Mn
JENDL-5.0
ENDF/B-VIII
TENDL-2021
Titarenko(2011) A0901061 W1
ZHAOWENRONGG(1993) S0041005 W1

3.7 Databases for resonance parameters, PFNS and nubar

autotalys contains the codes TARES[12] (resonance parameters), TANES[3] (PFNS) and TAFIS[3] (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 30-40 batch scripts, driven by the leading character of the nuclear symbol, which can all be run (I have 18 processors on my machine) with *run*.
- 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

  ```
  ~/tools/tendl/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 [18].
- 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 *plotlibcopy* to copy all the plots from the database into one *plots/* library. This script also runs *plotMT* 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
- Create a trimmed down libraries directory for the Data explorer, which does not require several libraries and subdirectories of *libraries*: run *lib2plot*.

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=jendl5.0 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=jendl5.0 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 autotalys 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 autotalys, 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 autotalys 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 autotalys software system

Originally the system was called T6, 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[3]: 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[3]: 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[12]: a code for resonance parameters including covariances, based on the Atlas of Neutron Resonances and other data libraries.
- TAFIS[3]: a code for fission neutron quantities, based on Wahl systematics among others.
- TANES[3]: a code for prompt fission neutron spectra, based on the Los Alamos model among others.

In 2017, we decided that a more robust system would be obtained if we would use the latter 3 codes separately and produce ready to use tables for the autotalys system. Even then, inspection of the *bin/* directory of autotalys reveals no less than 35 codes, much more than the aforementioned codes. We will explain, or at least shortly mention, them all. A future clean up of autotalys also means getting rid of some of these codes, or merging some of them into one code.

Besides the 'T' codes above, other important tools are:

- **autotalys**: a bash shell script that drives the entire autotalys 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-5.0, 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 autotalys

- The BNL checking codes for ENDF files. Not all BNL codes are needed, for autotalys we use:
  - CHECKR: Format checks
  - FIZCON: Basic physics checks
  - PSYCHE: Advanced physics checks

  - INTER: Calculation of integral cross sections
- NJOY-2016.47[7]: 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

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

autobnl

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

```
autobnl -file n-Fe056.jendl5.0
```

and you will get the output

```
       autobnl-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All Rights Rese

 * Start BNL codes
run CHECKR
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG
STOP     CHECKR - Tests completed successfully
run FIZCON
STOP     FIZCON - Tests completed successfully
run PSYCHE
Note: The following floating-point exceptions are signalling: IEEE_UNDERFLOW_FLAG IEEE_DENO
STOP     PSYCHE - Tests completed successfully
run INTER
STOP     INTER - Tests completed successfully
Diagnosis of CHECKR output in checkr.ers
Diagnosis of FIZCON output in fizcon.ers
Diagnosis of PSYCHE output in psyche.ers
autobnl is done
```

while your current directory contains the files checkr.inp(out,ers) fizcon.inp(out,ers), psyche.inp(out,ers) and inter.inp(out,ers). The error and output files can be consulted to see whether there are any problems with the nuclear data file.

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

```
autobnl -file n-Fe056.jendl5.0 -nopsyche -nointer
```

Appendix D gives all the possible options.

autoprepro

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

```
autoprepro -file n-Fe056.jendl5.0
```

and you will get the output

```
       autoprepro-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All Rights R

../bin/autoprepro -file n-Nb093.tendl.2021

 Directory with results:  /Users/koning/tools/autoendf/samples/new/
 * Start PREPRO
run linear
run recent
run sigma1
run sixpak
run activate
run dictin
run groupie
Note: The following floating-point exceptions are signalling: IEEE_INVALID_FLAG
run evalhard
Diagnosis of PREPRO output in prepro.ers
autoprepro 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 *prepro.ers* file. It is possible to disable certain codes and give other options, e.g.

```
autoprepro -file n-Fe056.jendl5.0 -nogroupie
```

Appendix E gives all the possible options.

autonjoy

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

```
autonjoy -file n-Fe056.jendl5.0
```

and you will get the output

```
       autonjoy-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All Rights Res

../bin/autonjoy -file n-Nb093.tendl.2021

       Starting time: ,

MAT: 4125 AWI: 1.000000+0 element: Nb mass: 093 iso: projectile: neutron
 Directory with results:  /Users/koning/tools/autoendf/samples/new/
 * Start NJOY for  n-Nb093.tendl.2021
 * NJOY done
Diagnosis of NJOY output in njoy.ers
autonjoy 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 *njoy.ers* files. It is possible to disable certain modules and give other options, e.g.

```
autonjoy -file n-Fe056.jendl5.0 -noacer
```

Appendix F gives all the possible options.

autofudge

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

```
autofudge -file n-Fe056.jendl5.0
```

Diagnosis files

Diagnosis files are produced by the above scripts 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. 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 'auto' 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 '.ers' files are of zero length 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 autotalys 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 autotalys. This will make TENDL production also more efficient. This database uses the filename convention of autotalys 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[19] 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 2853 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 *autotalys/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 *autotalys/tools/tendl/*. If we do 'loopnuc n > n.all', we have 2853 executable **autotalys** scripts, each with a projectile and nuclide name, e.g. n-Fe056. The file 'n.all' contains all these 2853 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/autotalys/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 autotalys.

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 autotalys 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 autotalys 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 thus produce data with some predictive power beyond that (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
0.6
0.8
1.000
1.200
1.400
1.600
1.800
2.000
2.200
2.400
2.600
2.800
3.000
3.500
4.000
4.500
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

```
best n
maxrot 5
strength 9
strengthm1 3
upbend y
bins 40
#micro
fismodel 5
ldmodel 5
fispartdamp y
hbstate n
class2 n
#pheno
#fismodel 5
#ldmodel 1
#fispartdamp n
#hbstate y
#class2 y
#
ngfit y
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 5th excited state, a SMLO PSF model including an M1-upbend of which the parameters have been adjusted first to then (n,γ) cross sections,, 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
#Fmax 1.000001 4
#Fmax 1.000001 16
#Fmax 1.000001 102
#Fmax 1.000001 103
#Fmax 1.000001 107
##expinclude 18
#libinclude 18 endfb8.0
#fracmax 0.10
#cwidth 2.
#Zdeep 0
#Adeep 0
#minbar 1
#maxbar 2
#searchmode 4
#keyvary ctable
#keyvary ptable
#keyvary betafiscor
#keyvary vfiscor
```

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 (which gives a more stable search procedure), 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, maxbar. 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 6 parameters to be optimized.

6.2.4 Results for 0-5 MeV

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 3 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

- Th230
- Th232
- Np236

Local deviations, overall good normalization

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

Larger deviations

- Pa231
- U235
- Pu241
- Pu244
- Am243
- Cm248
- Bk247
- Bk248
- Cf252

6.2.5 Results for 0-20 MeV: no further fitting

In theory, if the models for energies up to about 5 MeV would be perfect, we would not need any further adjustment for an extension to 20 MeV, because the parameters for multi-chance fission have been optimized for fist-chance fission for that fissioning nuclide. Note that this would only

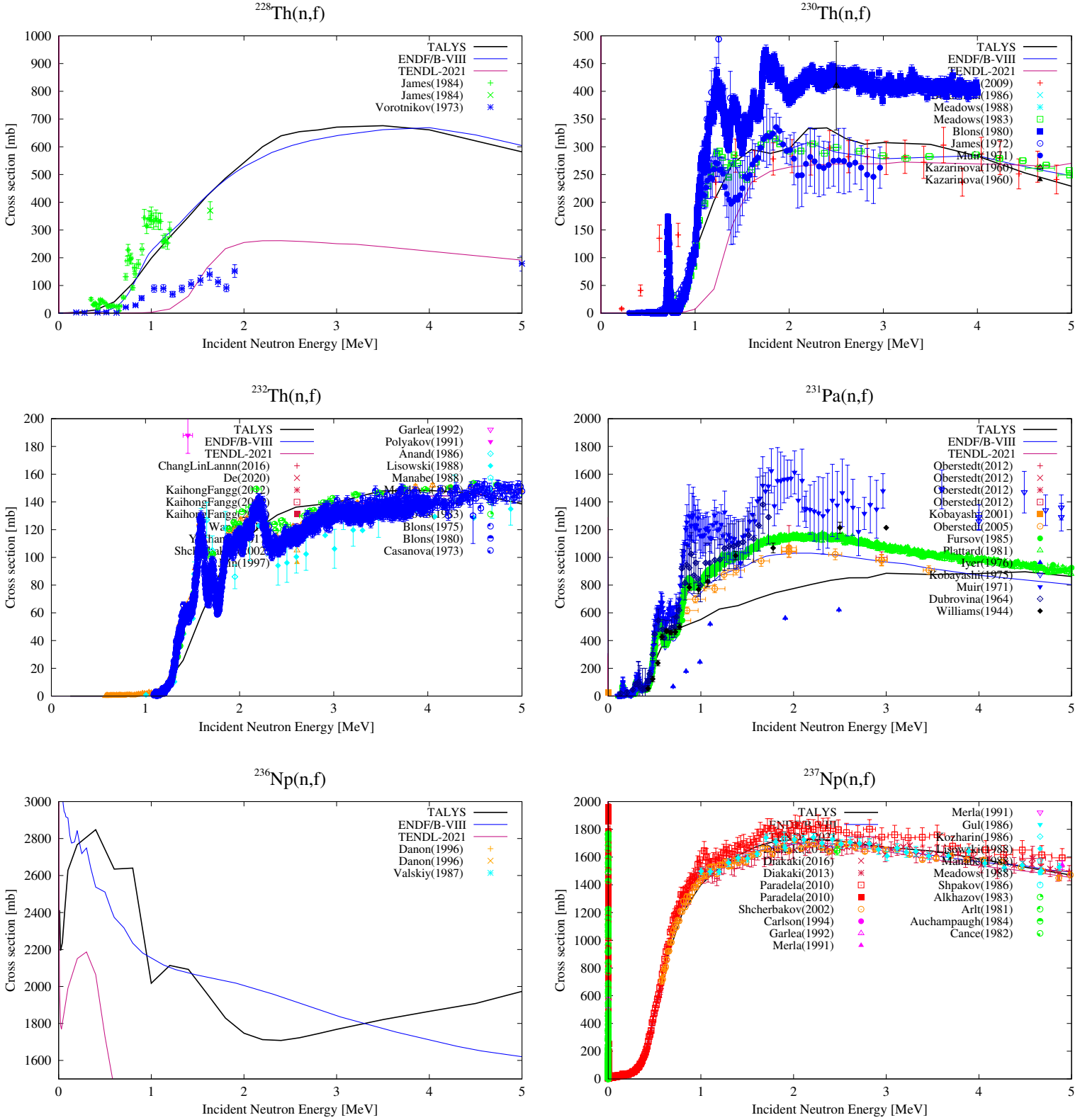


Figure 6.1: Fission cross sections for Th, Pa and Np isotopes.

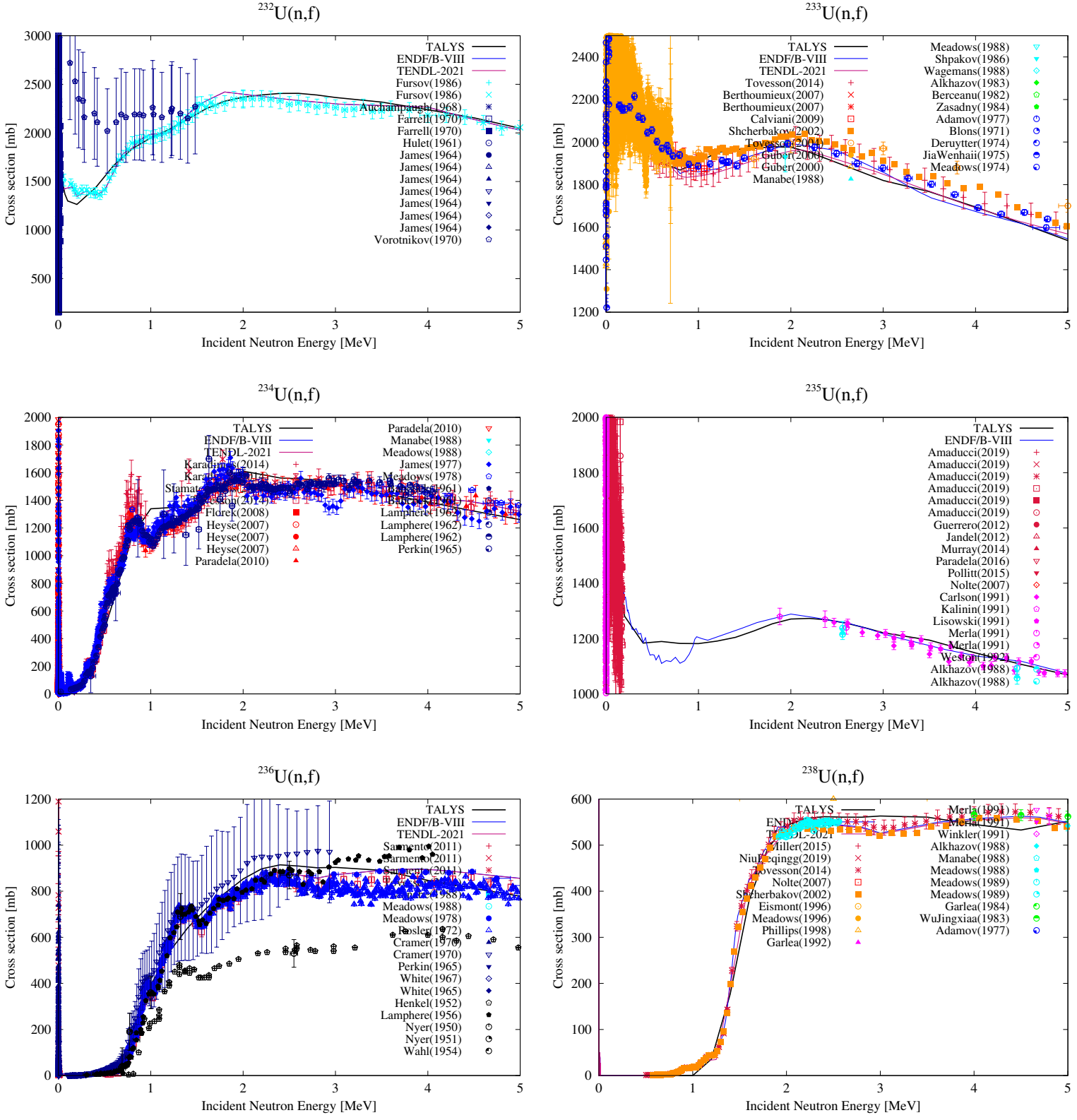


Figure 6.2: Fission cross sections for U isotopes.

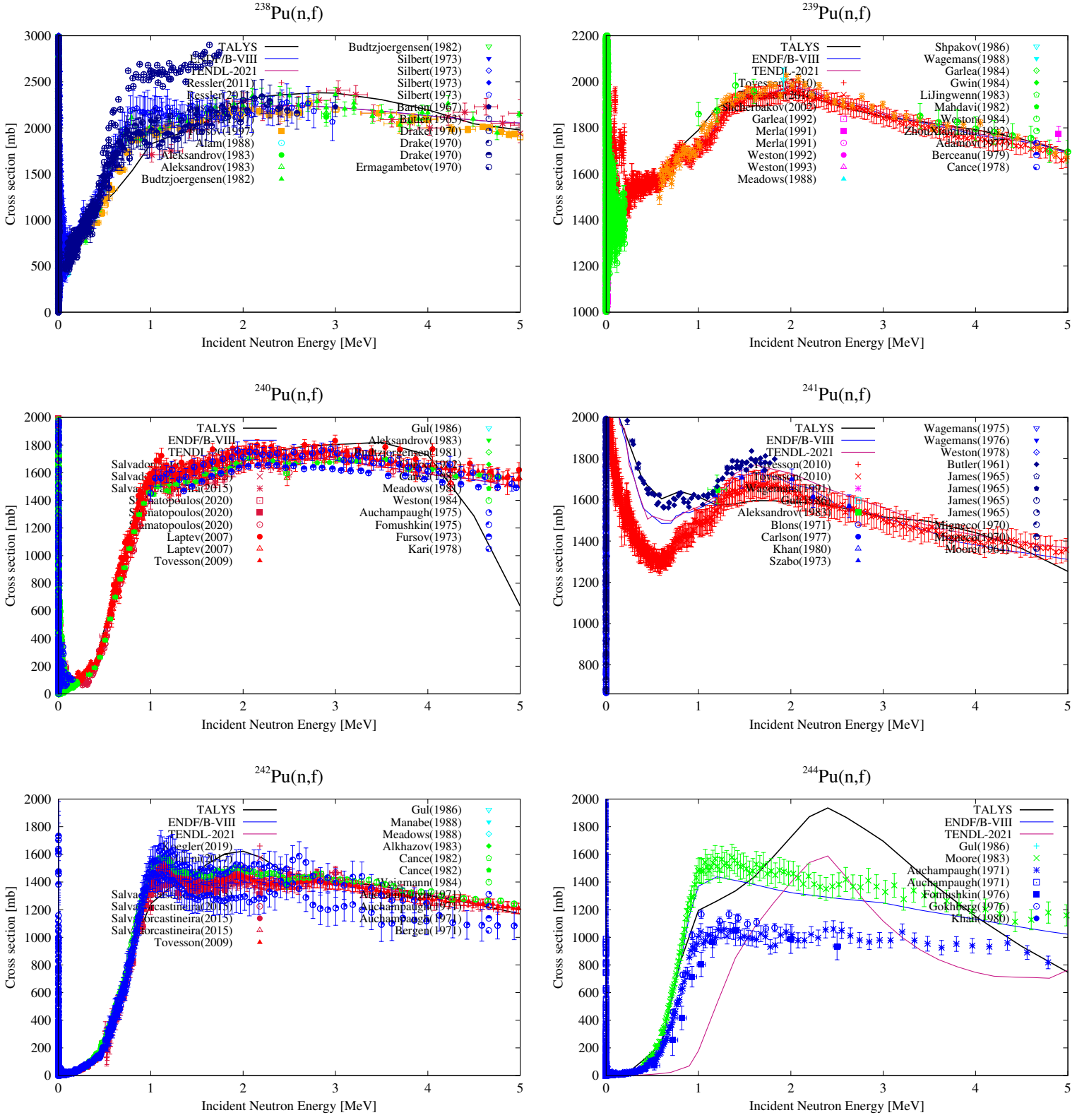


Figure 6.3: Fission cross sections for Pu isotopes.

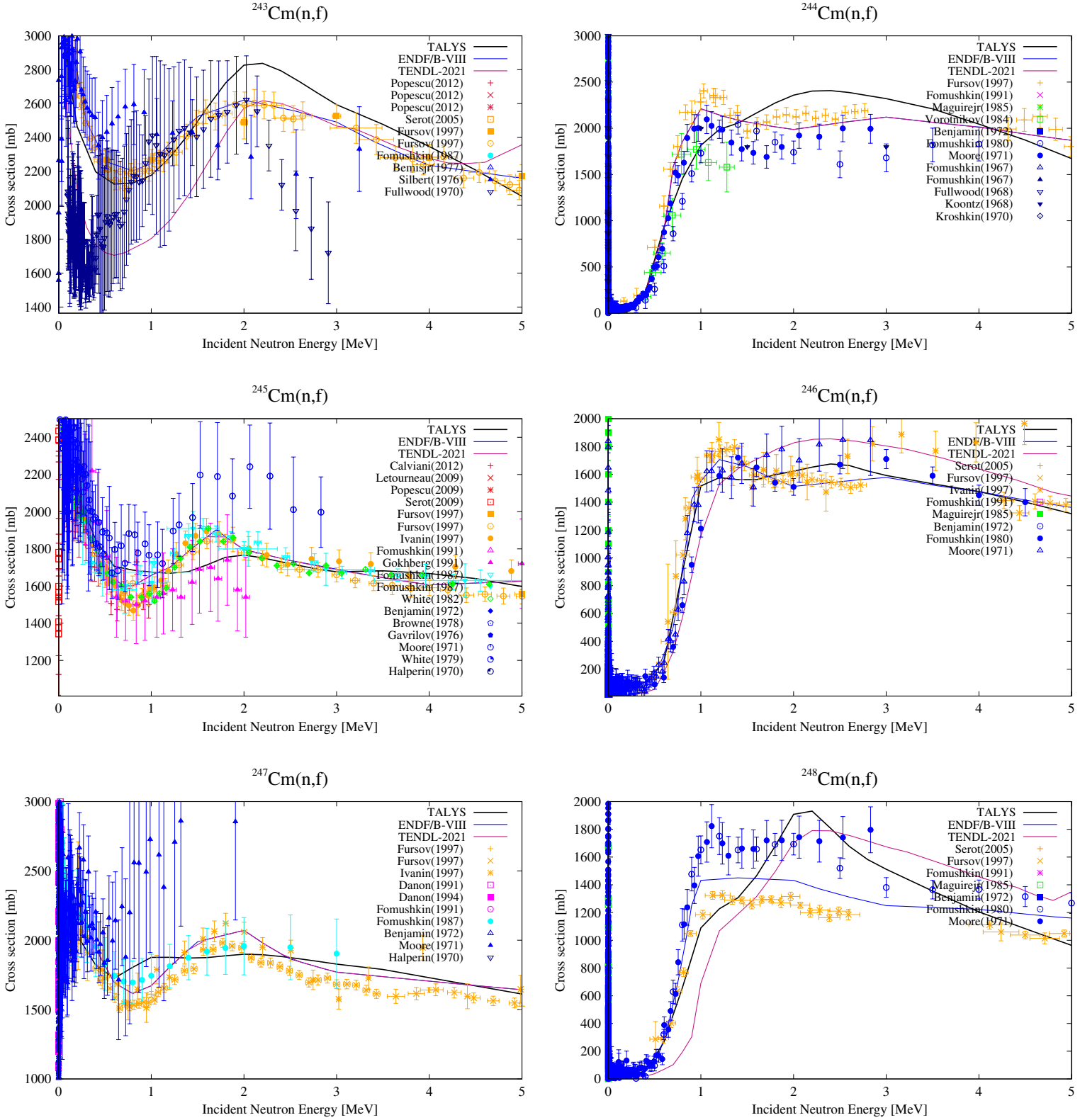


Figure 6.4: Fission cross sections for Cm isotopes.

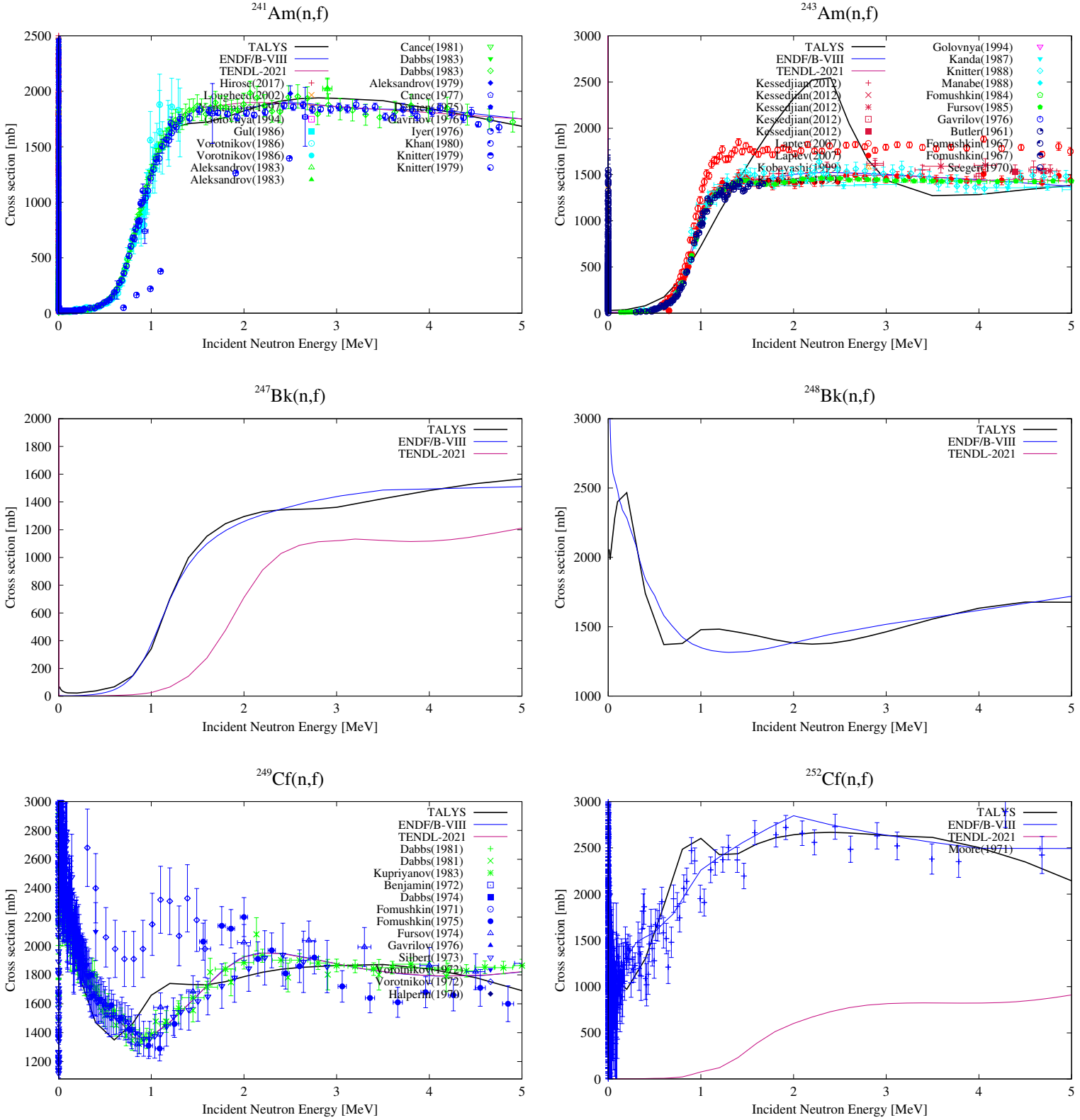


Figure 6.5: Fission cross sections for Am, Bk and Cf isotopes.

hold for the heavier isotopes of an actinide, like e.g. ^{238}U, since there is experimental first chance fission data for all the residual nuclides in the multi-chance fission chain. For e.g. ^{239}Pu this would only work for second chance fission since for ^{238}Pu there is, but for ^{237}Pu there is no experimental data. One of the reasons why expect this 'no-fitting' approach to fail is that the first-chance fission parameters were only fitted until the opening of second chance fission i.e. about 5 MeV, while the fall-off of first chance fission for the next several MeV plays an important role for the shape of the total fission cross section beyond 5 MeV.

6.2.6 Results for 0-20 MeV: further fitting

Finally, we can abandon the restriction of consistent multi-chance fission components and use the 6 adjustable parameters per fissioning nuclide and see how far we get with the best possible fit up to 20 MeV. As starting point in the parameter search we use the parameters found by adjusting the first chance fission, i.e. the results of the previous section. We search for parameters until 3rd chance fission, i.e. 18 parameters in total per projectile + target combination. This is not yet done, since we first want to get a better agreement of multi-chance fission in the previous step.

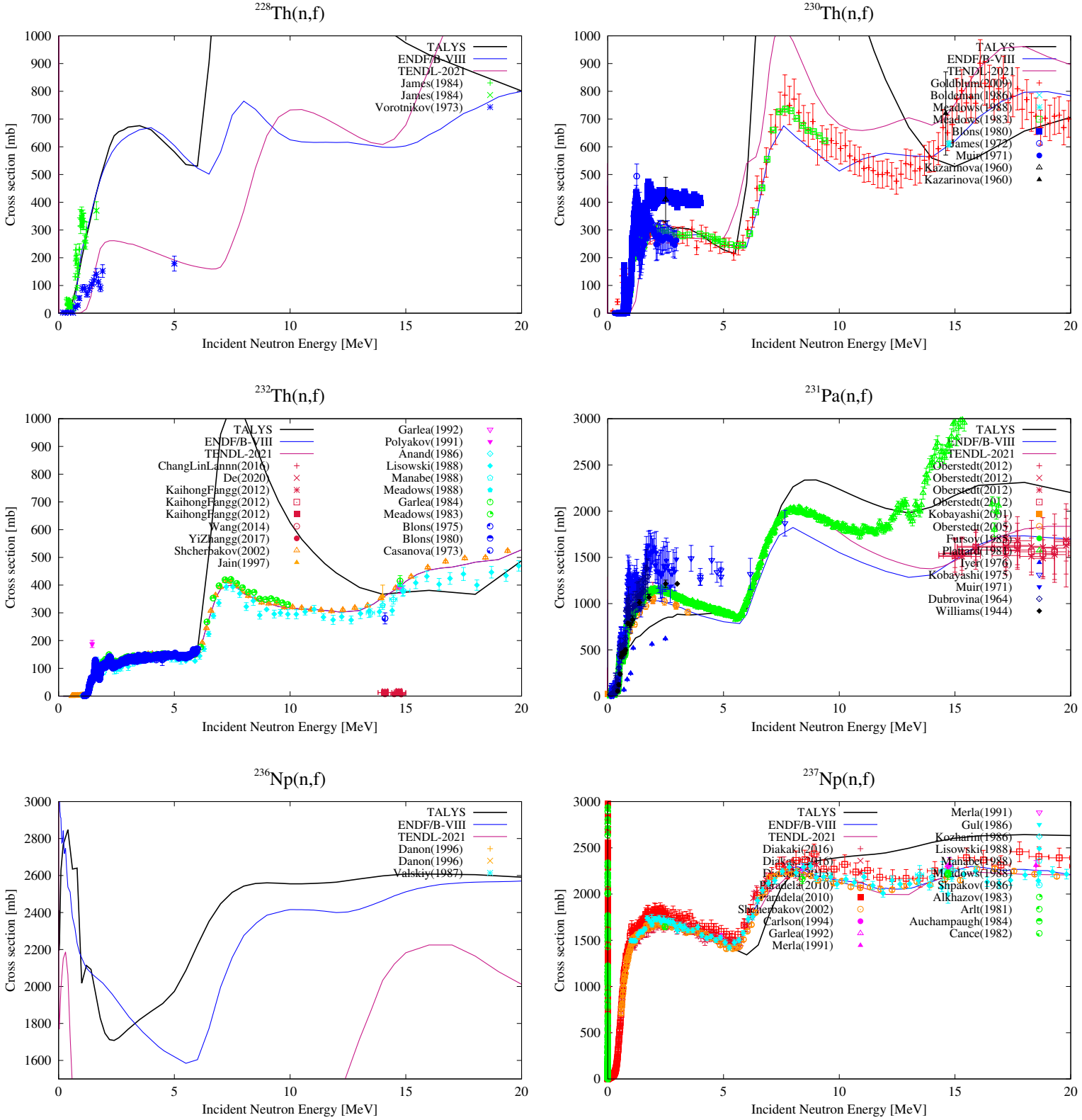


Figure 6.6: Fission cross sections for Th, Pa and Np isotopes.

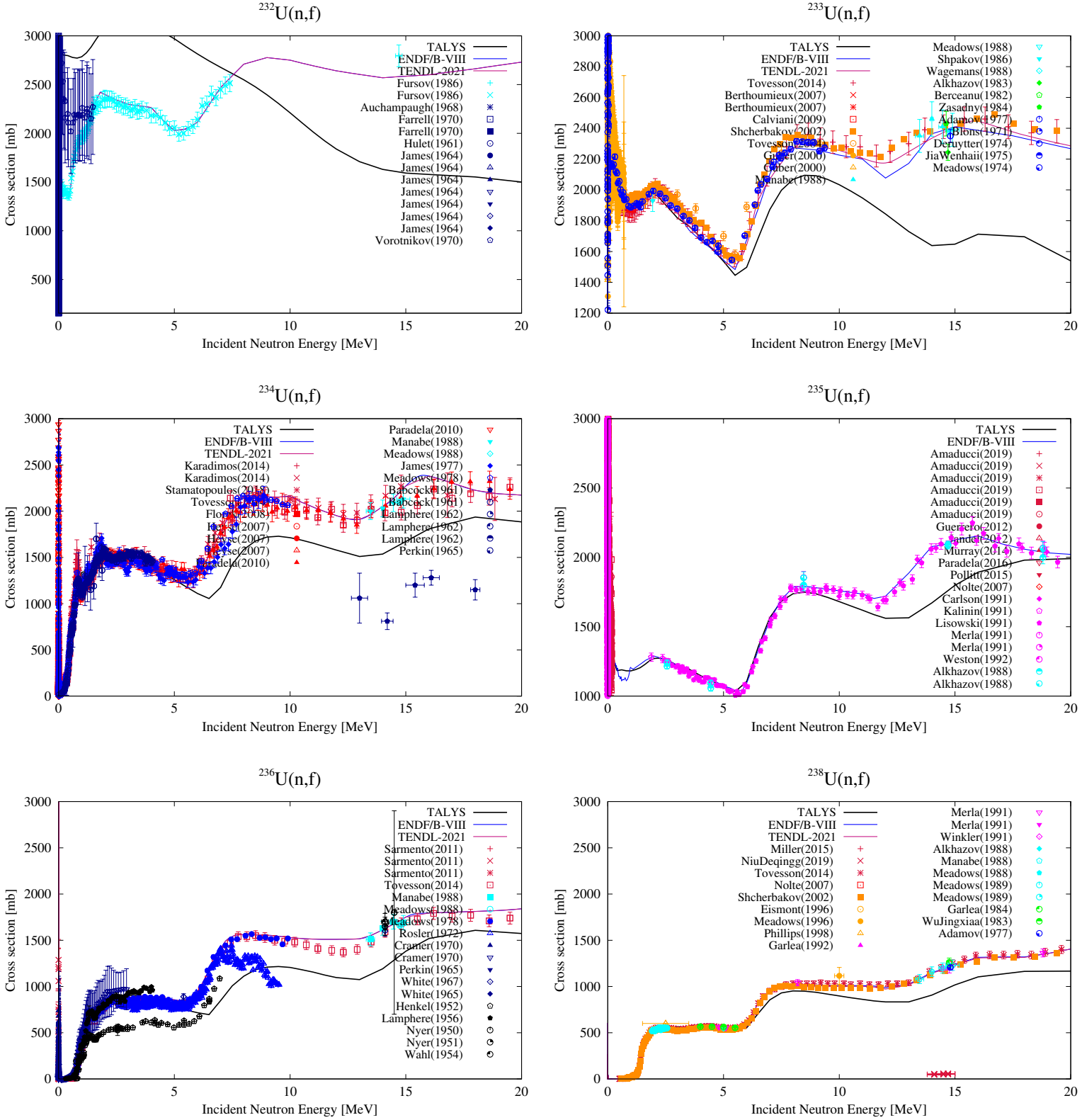


Figure 6.7: Fission cross sections for U isotopes.

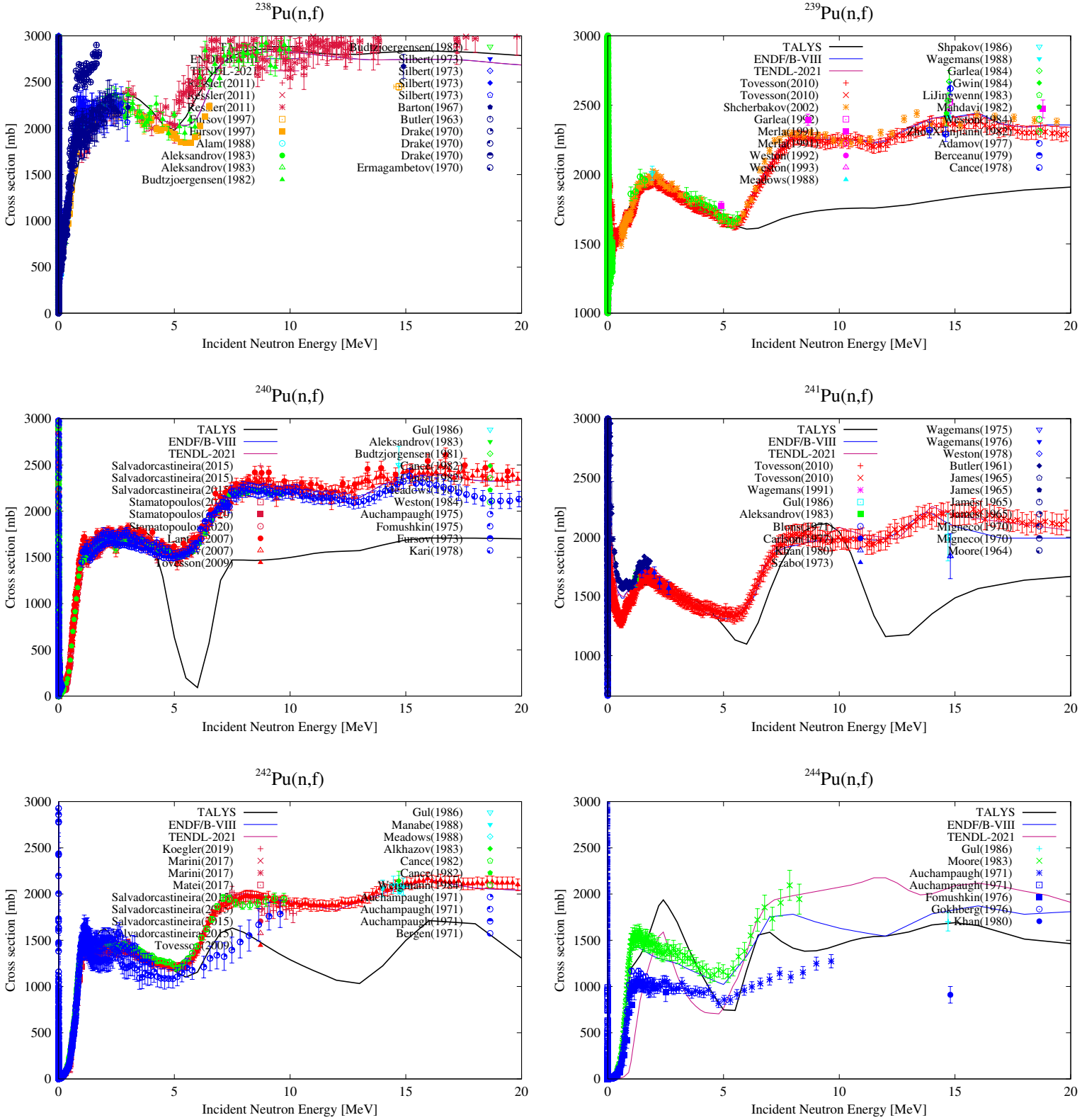


Figure 6.8: Fission cross sections for Pu isotopes.

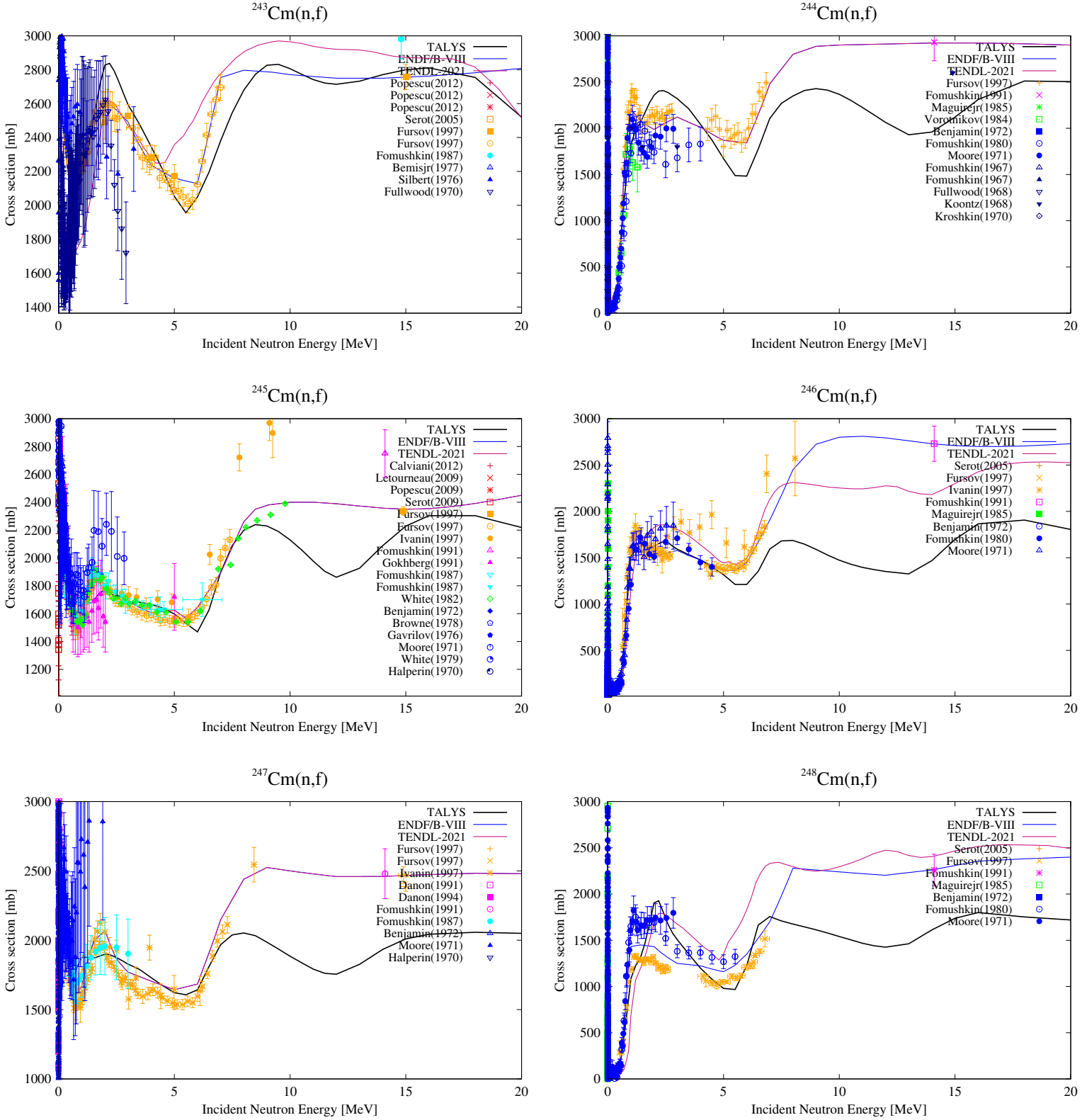


Figure 6.9: Fission cross sections for Cm isotopes.

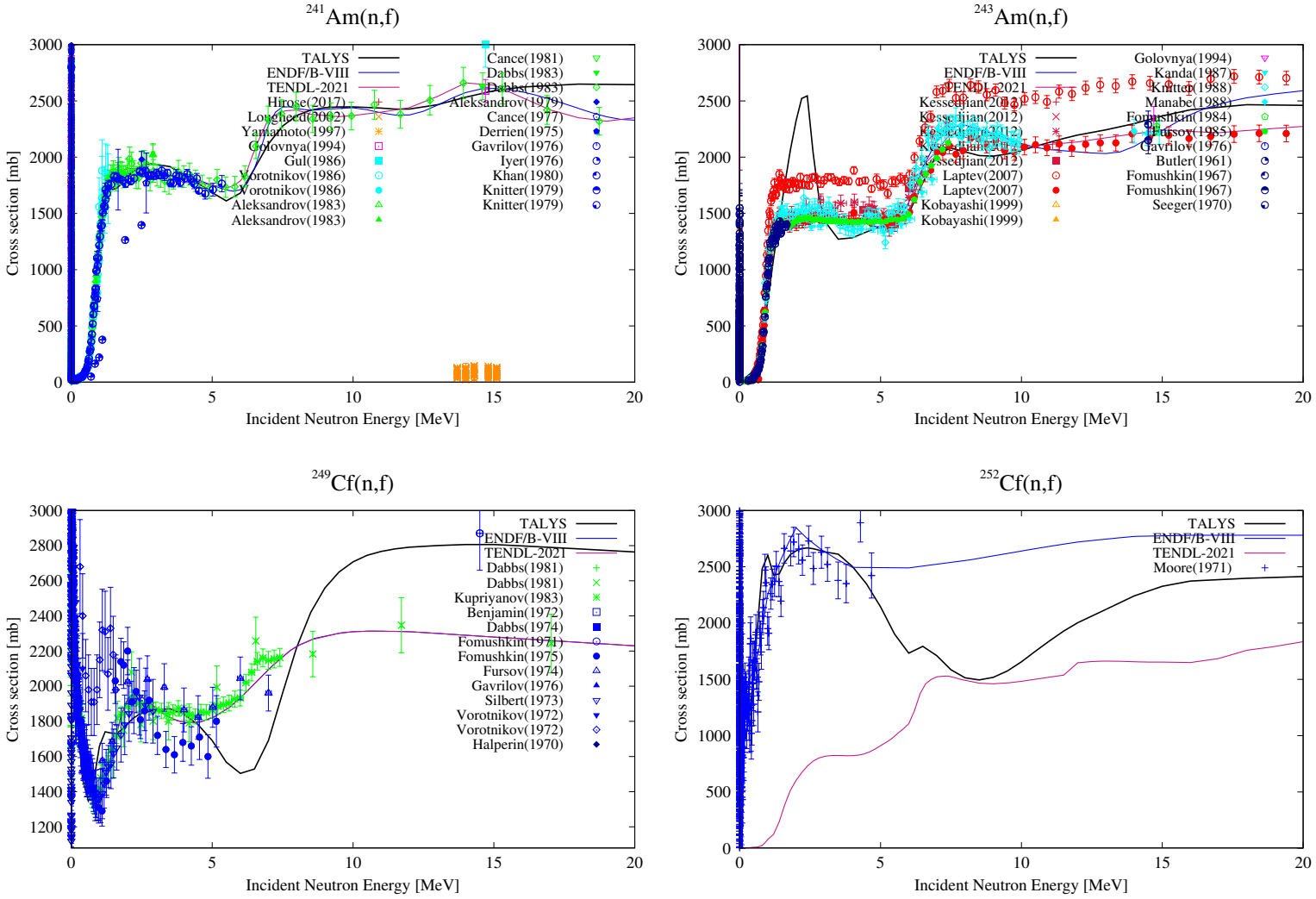


Figure 6.10: Fission cross sections for Am, Bk and Cf isotopes.

6.3 autosearch: script for optimization with autotalys for many nuclides

We wish to automate the optimization of TALYS model parameters to experimental data from EXFOR or other (evaluated) experimental databases such as the Atlas or RIPL. Since autotalys has a '-search' flag, we can make another script which loops such a search over many nuclides. This script is called **autosearch**. Here is an example which optimizes the nuclear model parameters, using one of the optimization techniques built in TASMAN, for all nuclides in the *batch.nn011* file for (n,n'), (n,2n) etc parameters,

```
autosearch -mode nn -isomer -ldmodel 7 -fit
-nuclist /Users/koning/search/nn011/batch.nn011 -ntalys 160 -plotonly -plotter
```

6.3.1 Optimization of level density parameters

For optimization of level density parameters we use a script like this

```
#!/bin/bash
/Users/koning/bin/autotalys  -element Y -mass 089  -proj g -bins 10 -low -ld
-ldmodel 7 -noendf -nonjoy -noresidual -noisomer -nolevels -norecoil -search
-binsrand 10 -noplot -subfission -nomcnp
-talysfile /Users/koning/tendl/ld/ld7/talys.ld
-tasmanfile /Users/koning/tendl/ld/ld7/tasman.ld
-tarwork -nofit -nobest -noautonorm -ntalys  400
```

which is created by a script **loopld** which loops over all 2854 nuclides. **autotalys** automatically recognizes whether there are discrete levels and/or D0 values to be fitted. This optimization is then done for each level density model. Use **tendl/ld/allld** to make all the plots and tables.

6.4 Nuclear model parameter adjustment for TENDL

For the production of TENDL, we attempt to get the best agreement with experimental data by adjusting the TALYS nuclear model parameters. This optimal set of input parameters is then stored in *talys/structure/best/* so that the optimal calculation can always be reproduced. If the fit is not good enough, we always have the option to *autonorm* to other nuclear data libraries as explained in this tutorial. Especially in the years 2013-2015 I spent a lot of effort to produce a 'best' TALYS input file for each isotope for neutron-induced reactions. An automated search for the best parameters still does not exist, due to varying experimental data and insufficient knowledge on the sensitivity of parameters in various energy ranges, and other biases which makes automatic optimization difficult. In the meantime, TALYS has evolved, some better nuclear models + parameters are on the market, notably the SMLO model for gamma-ray strength functions, the RIPL discrete level database has changed, including revised decay branchings which affects isomeric production cross sections. Also, it is inevitable that some isotopes get more attention than other isotopes. Recently, new evaluations for all isotopes of Ag, Pm, Eu and Lu have been performed. Improved resonance parameters were available and these have been included in the latest version of TARES. To have a consistent quality level between the resonance and fast range, the fast range has also been re-evaluated. For future reference, also to myself, I note some parameter adjustments which have the desired effect:

- In the rare earth region, but also for other deformed nuclides like Ag isotopes, if there is no nuclide-specific OMP available, the default in TALYS is to subtract 15 % from **d1adjust** which compensates the W_D component of the OMP for the inclusion of strongly coupled levels. For the recent fits, we have in addition applied the work of Nobre et al. on the OMP in the rare-earth region. Conservation of the volume potential dictates that for deformed

nuclides we may reduce the real volume radius by about 1%, i.e. you may find *rvadjust n 0.99* in some of the best files. However, this adjustment alone is often not enough, and instead of resorting to a dispersive OMP, which we only have for spherical nuclides, we apply some low-energy adjustment of the geometry as e.g. **rvadjust n 1.018 -0.2 0.1 0. 0.98** which is described in the TALYS manual. In short, this means, the default r_v radius is multiplied by 1.018 over the entire energy range. In addition to that multiplication, **rvadjust** is subjected to another multiplication factor which is equal 1. at and outside the range -0.2 and 0.1 MeV and reaches the maximum of 0.98 at 0 MeV. This enables a smooth transition between all values of **rvadjust**.

- Often the isomeric ratio needs to be changed. This was especially clear for the Eu153(n,2n) to the g.s. and m and n isomers. There are two ways to change this. First, there may be branching ratios in the discrete level file which are unknown, in which case they have a 'B' at the discrete level and the level decays by default for 50% to two levels. Those can be changed to force more or less decay to the isomer. Adjust branchings are stored in *talys/structure/level/branch*. Another method is to change the width of the spin distribution of the nucleus which has the isomer, e.g. *s2adjust 63 152 1.6*.
- The (n,gamma) cross section may fall off too fast above 1 MeV. One remedy is to change the level density of the target nucleus, e.g. **Pshiftadjust 63 151 0.50**
- Simultaneously adjusting (n,2n) and (n,p). These channels are often in competition and the exciton model partial level density parameters are used to adjust. This is done for the compound nucleus, so e.g. **gnadjust 61 152 1.08**and **gpadjust 61 152 1.08** simultaneously will increase the (n,2n) and decrease the (n,p) compared to the default. Reducing the (n,p) cross section with less effect on the other channels can be done by e.g. **rvadjust p 0.95**.
- Adjusting the (n,alpha) cross sections. At threshold, the (n,a) curve can be increased by e.g. *rvadjust a 1.10* while at higher energies, near the peak, also e.g. bf Cknock a 1.30 may work. The latter is less sensitive near threshold so one can play with a combination of **rvadjust a** and **Cknock a**

7. Conclusions and outlook

It requires some discipline to document close to everything you have ever produced in terms of software. This AUTOTALYS 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 autotalys 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, S. Hilaire, and S. Goriely. “TALYS: modeling of nuclear reactions”. In: *European Physical Journal A* 59 (2023), p. 131.

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

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

[4] 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.

[5] 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.

[6] D.E. Cullen. “PREPRO 2021 - ENDF/B6 Pre-processing codes”. In: *Technical report IAEA-NDS-0238, IAEA* (2021).

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

[8] 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.

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

[10] A.J. Koning. *EXFORTABLES-1.0: An experimental nuclear reaction database based on EXFOR*. Vol. IAEA-NDS-0235. 2020.

[11] A.J. Koning. *RESONANCETABLES-1.0: Database for thermal cross sections, MACS and average resonance parameters*. Vol. IAEA-NDS-0234. 2020.

[12] 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.

[13] M. Herman et al. “EMPIRE: Nuclear Reaction Model Code System for Data Evaluation”. In: *Nucl. Data Sheets* 108 (2007), p. 2655.

[14] T. Kawano et al. “IAEA Photonuclear Data Library 2019”. In: *Nuclear Data Sheets* 163 (2020), pp. 109–162. ISSN: 0090-3752. DOI: https://doi.org/10.1016/j.nds.2019.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0090375219300699.

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

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

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

[18] A.J. Koning. *ISOTOPIA: Simulation of medical isotope production with accelerators*. Vol. IAEA-NDS-xxxx. 2020.

[19] 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-1.97 (Version December 29 2023) (C) Copyright 2023 Arjan Koning
          All Rights Reserved

/Users/koning/autotalys/bin/autotalys

________Script [/Users/koning/autotalys/bin/autotalys] starting on: Arjans-
    MacBook-Pro.local at: 2023-12-29,12:42:48

### 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)

    Consult the Autotalys manual in doc/autotalys.pdf 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]

-list
       Print only list of nuclides, no TALYS etc. runs [default: -nolist]

-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]

-E300
    List of incident energies up to 300 MeV [default: -low]

-E600
    List of incident energies up to 600 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]

    -fy
        Fission yield calculation [default: -nofy]

    -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]

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

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

    -talysdir [name]
              Directory with files for TALYS run [default: not used]

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

    -[no]resonance
              Include resonances in TALYS [default: -noresonance]

    -[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]fit
            Use fitted parameters if present [default: -nofit]

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

    -[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]

    -[no]ld
             TALYS runs for level density only [default: -nold]

    -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
           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]

    -tasmanversion [name]
            Specific TASMAN executable in bin/ [default: -tasmanversion tasman
               ]

    -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]

    -[no]autosearch
              Automatically insert parameters to be searched [default: -
                  autosearch]

Output: x-y tables, ENDF files and options

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

    -tefalversion [name]
            Specific TEFAL executable in bin/ [default: -tefalversion tefal]

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

    -libname  [name]
              Name of library for ENDF MF1 [default: TENDL-2023]

    -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 doc/autotalys.pdf 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
      autobnl  : 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
      autoprepro: script to run the PREPRO codes
      PREPRO   : Lawrence Livermore pre-processing codes LINEAR, FIXUP, etc.
      FUDGE    : Lawrence Livermore re-formatting code
      autonjoy : script to run NJOY
      NJOY     : Los Alamos processing program
      autofudge : script to run the FUDGE codes
      automcnp : 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. Yet Another Nuclear Data Format: YANDF

YANDF stands for 'Yet Another Nuclear Data Format'. As it name suggests, it is inspired by the YAML markup language, as YAML is the serialization format which in my view is closest to being human and computer readable at the same time. YANDF is an attempt to unify the nuclear data structure for data sets which come from either TALYS, ENDF nuclear data libraries, EXFOR or other basic nuclear data sources. The format is aimed to be relatively simple. Once data are stored in YANDF, processing can be done independently from the particular source of the data into JSON, ENDF, GNDS etc. data formats. This may be useful for processing of data from any of these categories for numerical operations, plotting, ML applications etc. The main reason for its construction was a consistent output for the TALYS nuclear model code to enable easier processing towards various applications, but at the same time we took ENDF and EXFOR along in the process. Also, it is used for a compilation of all resonance parameters. The serialization of YANDF has a key-value schema which is not as non-descriptive as ENDF and not as heavy as GNDS. It aims to give a compact description of a nuclear reaction in terms of metadata. The source of the data, TALYS, ENDF or EXFOR may give rise to some keywords which are different, but in general the structure is the same for each of these categories. The metadata and associated keywords are supposed to be a direct classification of a nuclear reaction as defined by nuclear physics, such as found in textbooks or journal articles. This means that nuclear reaction observables are leading in the description, and not the ENDF format with its MF/MT numbers and neither the EXFOR format with its emphasis on experimental methods and details. As the role of a nuclear model code is to provide an estimate of all nuclear reaction observables as commonly defined in nuclear physics, we have taken TALYS as the basis for the schema. As EXFOR stores experimental observables, the step from EXFOR to YANDF is almost only a format change: the main keywords are the same as the one of TALYS and all experimental details are stored in their original format for the moment. We note that we only focus on the actual data, not the complete metadata of EXFOR, which can be obtained from the original EXFOR entries. (The main challenge for EXFOR is to determine *which* EXFOR entries correspond to the reaction channels as defined in ENDF or in TALYS output

files. Several different EXFOR categories of data may have to be included or excluded.) In ENDF libraries, only some data are observables (cross sections, nubar), but most of the data need to be processed into observables using operations on data in different parts of the data file.

Before we describe the format in more detail we give 3 examples of a typical YANDF file, for the same nuclear reaction ^{235}U(n,f). This is the file coming from TALYS

```
# header:
#   title: U235(n,f) cross section
#   source: TALYS-2.0
#   user: Arjan Koning
#   date: 2023-11-24
#   format: YANDF-0.1
# target:
#   Z: 92
#   A: 235
#   nuclide: U235
# reaction:
#   type: (n,f)
#   ENDF_MF: 3
#   ENDF_MT: 18
# datablock:
#   quantity: cross section
#   columns: 2
#   entries: 24
##       E             xs
##     [MeV]          [mb]
   1.000000E-11   0.000000E+00
   2.530000E-08   0.000000E+00
   2.000000E-07   0.000000E+00
   1.000000E-06   3.049857E+05
   1.000000E-05   9.592149E+04
.....
```

This is the file for the ENDF-B/VIII.0 data library

```
# header:
#   title: U235(n,f) cross section
#   source: ENDF
#   user: Arjan Koning
#   date: 2023-11-26
#   format: YANDF-0.1
# endf:
#   library: endfb8.0
#   author: IAEA CIELO Collaboration
#   year: 2017
# target:
#   Z: 92
#   A: 235
#   nuclide: U235
# reaction:
#   type: (n,f)
```

```
#   Q-value [MeV]:  1.934054E+02
#   E-threshold [MeV]:  1.000000E-11
#   ENDF_MF: 3
#   ENDF_MT: 18
# datablock:
#   quantity: cross section
#   columns: 4
#   entries: 333
##       E             xs            xslow          xsup
##     [MeV]          [mb]           [mb]           [mb]
   1.000000E-11   0.000000E+00   0.000000E+00   0.000000E+00
   2.250000E-03   0.000000E+00   0.000000E+00   0.000000E+00
   2.250000E-03   2.634378E+03   2.397892E+03   2.870864E+03
   2.250014E-03   2.668097E+03   2.428584E+03   2.907610E+03
   2.250056E-03   2.769988E+03   2.521328E+03   3.018648E+03
....
```

This is the file for one of the experimental data sets in EXFOR

```
# header:
#   title: U235(n,f) cross section
#   source: EXFOR
#   user: Arjan Koning
#   date: 2023-09-18
#   format: YANDF-0.1
# exfor:
#   author: Moore
#   year: 1978
#   subentry: 10629004
#   X4 reaction: 92-U-235(N,F),,SIG
#   X4 source: IAEA-NDS C5 file, database version 2023-07-18
#   X4 link: https://nds.iaea.org/EXFOR/10629004
# target:
#   Z: 92
#   A: 235
#   nuclide: U235
# reaction:
#   type: (n,f)
#   ENDF_MF: 3
#   ENDF_MT: 18
# datablock:
#   quantity: cross section
#   columns: 5
#   entries: 3777
##       E             dE             xs             dxs       Normalization
##     [MeV]          [MeV]          [mb]           [mb]            []
   1.625000E-06   0.000000E+00   1.304000E+04   1.539000E+02   1.000000E+00
   1.675000E-06   0.000000E+00   1.256000E+04   1.510000E+02   1.000000E+00
   1.725000E-06   0.000000E+00   1.270000E+04   1.549000E+02   1.000000E+00
   1.775000E-06   0.000000E+00   1.226000E+04   1.527000E+02   1.000000E+00
.....
```

Obviously, for EXFOR we have several files which in metadata only differ from TALYS or ENDF in the **exfor** keyword. The metadata in the above files completely defines the U235(n,f) reaction.

C.1 Format

The YANDF format is almost equal to the well-known YAML format. If the '#' is removed from the first columns of the metadata header of the above file, we almost have a YAML file. The difference is that we do not quote strings and that the data can be given in multi-column format. This means that indentation of the key-value pairs is essential, which is the price that YAML pays for not having to include computational symbols such as '', '[', ',' as in e.g. JSON The above files show the most general keywords without any indentation, while sub-keywords are indented by two spaces, subsub-keywords by 4 spaces, etc. As there are many users who want to use numerical data directly from the file, as in gnuplot or other software, we have chosen to use a '#' at the start of every metadata line. It is not too difficult to remove the '#' and parse the above file to JSON with either your own script or, as we already verified, with an AI assistant.

A YANDF file only contains what is relevant. For example, in the above case there is no specification of any isomeric level in either the target or residual nucleus. Hence, we have decided to leave all 'inactive' metadata out. Parsers will have to take this feature into account.

C.2 Keywords and values

The main keywords should be general enough to describe nuclear reaction observables from at least TALYS, ENDF or EXFOR, but also for additional quantities such as e.g. level densities, photon strength functions, radioisotope yields, etc. as written by TALYS to output files. Also, compiled data for e.g. thermal neutron cross sections, resonance parameters, Maxwellian-averaged cross sections etc. fit well into this schema.

C.2.1 header

All YANDF files start with the same keywords:

- **header**:
  - **title**: the title, generally constructed from the other metadata, enabling the user to see directly which nuclear reaction this concerns
  - **source**: the source of the datafile, this is often TALYS, ENDF or EXFOR. If calculated uncertainties and covariance data are available, this can also be another source, like e.g. TASMAN
  - **user**: the name of the person who produced this file (e.g. in TALYS you can change the hard-wired name into your own)
  - **date**: the date of the production of this file in yyyy-mm-dd format
  - **format**: version of the YANDF format

C.2.2 endf

When the source is an ENDF library, we have the keywords

- **endf**:
  - **library**: one of the NDL's such as ENDFB8.1, JENDL5.0, JEFF3.3, TENDL-2023, CENDL3.2 etc,
  - **author**: the author of the evaluation as extracted from the ENDF file
  - **year**: the year of the evaluation as extracted from the ENDF file

C.2.3 exfor

When the source is EXFOR, we have the keywords

- **exfor**:
  - **author**: first author of the experimental work
  - **year**: the year of the publication of the measurement
  - **subentry**: the EXFOR subentry number
  - **X4 reaction**: the particular EXFOR reaction code as extracted from EXFOR, for checking purposes
  - **X4 source**: the version of EXFOR, and the specific computational form of starting database
  - **X4 link**: the https link to the EXFOR subentry, for all experimental details
  - **level energy [MeV]**: only for discrete levels: the level energy as given by EXFOR

C.2.4 target

The first part of a nuclear reaction specification is the target nucleus.

- **target**:
  - **Z**: the charge number
  - **A**: the mass number
  - **nuclide**: the nuclide name

The above keywords are always present. In addition, isomeric level information can be provided by the **level** keywords described below.

C.2.5 reaction

The nuclear reaction may have several keywords for a complete description.

- **reaction**:
  - **type**: the nuclear reaction channel
  - **Q-value [MeV]**: the Q-value (only specified when appropriate)
  - **E-threshold [MeV]**: the incident energy threshold (only specified when appropriate)
  - **ENDF_MF**: the ENDF MF number for specification of the type of data
  - **ENDF_MT**: the ENDF MT number for specification of the reaction channel

C.2.6 residual

Often, but not always, a nuclear reaction leads to a well-defined residual nucleus.

- **residual**:
  - **Z**: the charge number
  - **A**: the mass number
  - **nuclide**: the nuclide name

The above keywords are always present when **residual** is present. In addition, isomeric level information can be provided by the **level** keyword described below.

C.2.7 datablock

Before we read the data, we need to know what we are reading and in what format.

- **datablock**: Description of the data block that follows below.
  - **quantity**: the physical quantity that we are reading
  - **columns**: the number of columns
  - **entries**: the number of entries

Below these keywords always follow 2 lines starting with '##', one with the quantities and the other one with the units.

C.2.8 Keyword: level

The **level** keyword describes the data of a discrete level. It can appear as an keyword under

- **target**, when the target is in an isomeric state
- **reaction**, for scattering off a discrete level
- **residual**, when the residual nuclide is in an isomeric state, or for gamma-ray transitions between discrete states

It is described by

- **level**: Description of discrete level
  - **isomer**: the isomeric number
  - **number**: the level number
  - **energy [MeV]**: the level energy
  - **spin**: the level spin
  - **parity**: the level parity
  - **half-life [sec]**: the half life

In general, the indentation for **level** is 2 spaces, i.e. one below the main keyword, but for discrete level gamma-ray transitions the final level is specified at 4 spaces.

C.2.9 parameters

This is a TALYS-specific keyword. It contains the nuclear models and parameters used in the calculation. It starts with

- **parameters**:

after which various parameters can be given. Here is an example for a level density output file

```
# parameters:
#   ldmodel keyword: 5
#   level density model: Hilaire-Goriely tables
#   Nlow: 8
#   Ntop: 17
#   ctable: -1.621100E-01
#   ptable: -5.763700E-01
```

C.2.10 observables

This is a TALYS-specific keyword. It contains the observables estimated by TALYS used in the calculation. It starts with

- **observables**:

after which various observables can be given. Here is an example for a level density output file

```
# observables:
#   experimental D0 [eV]:  1.200000E+01
#   experimental D0 unc. [eV]:  1.300000E+00
#   theoretical D0 [eV]:  1.245919E+01
#   Chi-2 D0:  1.247688E-01
#   C/E D0:  1.038266E+00
```

D. All options for autobnl

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

```
      autobnl-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All
          Rights Reserved

autobnl                                                          autobnl

NAME
      autobnl - Script to make input file for BNL checking codes and running them

SYNOPSIS
      autobnl [option] [value]

DESCRIPTION
      With autobnl, 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: autobnl [option] [value]

      Some option flags should be accompanied by a value

      Examples: autobnl -file Nb093-n.tendl (obligatory: ENDF-6 file)
                autobnl -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]diag
                Diagnosis of output [default: -diag]

      -[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
      autobnl was written by Arjan Koning.
```

E. All options for autoprepro

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

```
      autoprepro-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All
           Rights Reserved

/Users/koning/bin/autoprepro

autoprepro                                                          autoprepro

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

SYNOPSIS
      autoprepro [option] [value]

DESCRIPTION
      With autoprepro, 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: autoprepro [option] [value]

      Some option flags should be accompanied by a value

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

```
              autoprepro -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]diag
                Diagnosis of output [default: -diag]

      -[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
      autoprepro was written by Arjan Koning.
```

F. All options for autonjoy

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

```
      autonjoy-1.0 (Version March 16 2023) (C) Copyright 2023 Arjan Koning, All
          Rights Reserved

/Users/koning/bin/autonjoy

      Starting time: ,

autonjoy                                                          autonjoy

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

SYNOPSIS
      autonjoy [option] [value]

DESCRIPTION
      With autonjoy, 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: autonjoy -file <endf file> [option] [value]

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

```
    Examples: autonjoy -file Nb093-n.tendl (obligatory: ENDF-6 file)
              autonjoy -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-2016.68]

    -[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]diag
              Diagnosis of output [default: -diag]

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

General NJOY parameters

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

     -temp [temperature]
          Temperature [default: 293.6]

     -libid [identifier]
           Library indentifier [default: TENDL-2023]

     -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]

     -[no]groupr
               Run ACER [default: -nogroupr]

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]

       -ign [value]
            Number of group structure [default: 20, ECCO 1968]

       -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
      autonjoy was written by Arjan Koning.
```

G. 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.

G.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.

G.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.

G.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).

G.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.

G.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.

G.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.