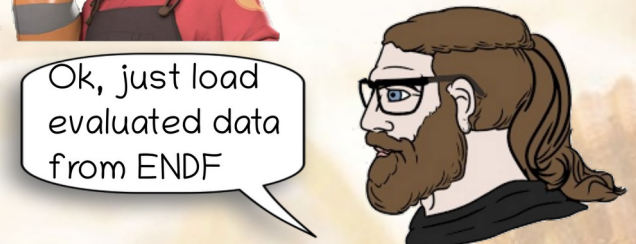
TalysLib – an easy way to get nuclear data

N.A. Fedorov,
FLNP JINR
na.fedorov@physics.msu.ru

Alushta-2022

Why do we need nuclear data?

- Nuclear facilities modeling



- Theory testing and parameters estimation

- Planning of new experiments

Sources of nuclear data

Evaluated:

- ENDF (characteristics of nuclear reactions)
- ENSDF (nuclear structure)
- AME (nuclear masses)
- **RIPL (ENSDF+AME+model parameters)**
- **<u>TALYS program</u>**

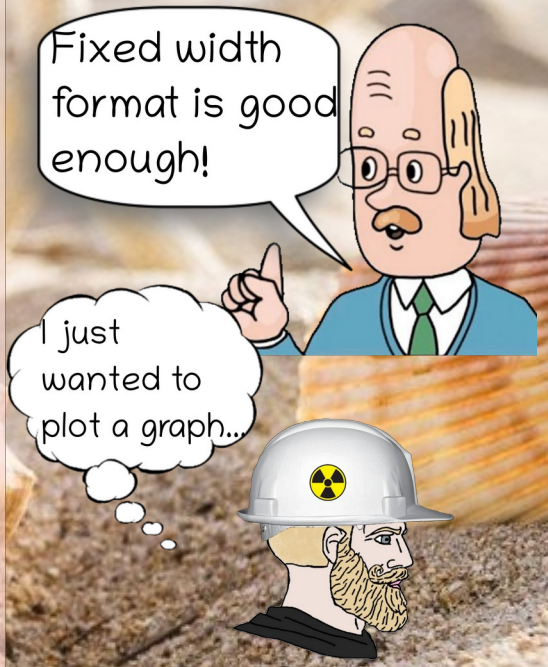
etc...

Measured:

- EXFOR (“Raw experimental data”)
- Pre-processed EXFOR (C4, T4, EXFORTABLES)

Problems

- There is no C++/Python parser for this data
- The EXFOR and ENDF data format is quite complex to read
- One have to perform data search/plotting/processing by hand
- It is interesting to compare your data with other experiments/estimations and calculations. *In automatic mode*

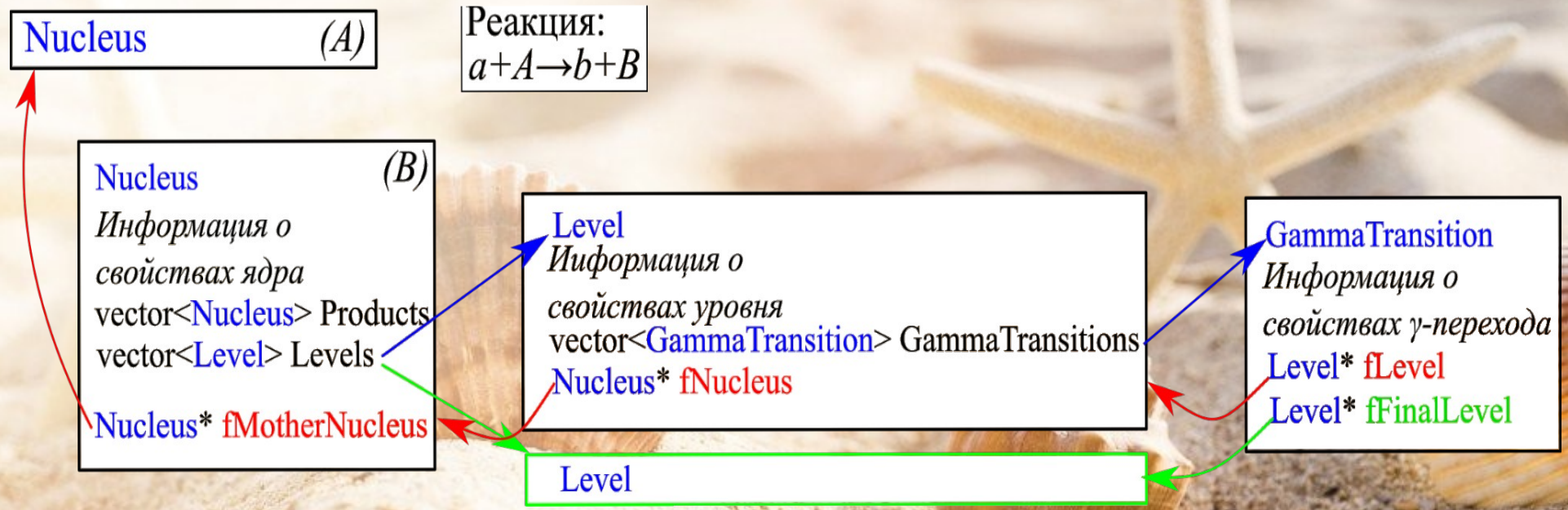


TalysLib

An object-oriented C++ library for nuclear data access

- TALYS is a powerful nuclear reaction calculation program which uses RIPL-3 database
- ROOT is a data analysis framework used by high energy physics and others
- TalysLib automates work with TALYS and its database
- TalysLib contains parser for ENDF and EXFOR (EXFORTABLES)

TalysLib structure



- The TalysLib structure groups data to related objects.
- Each object has a pointer to parent object.

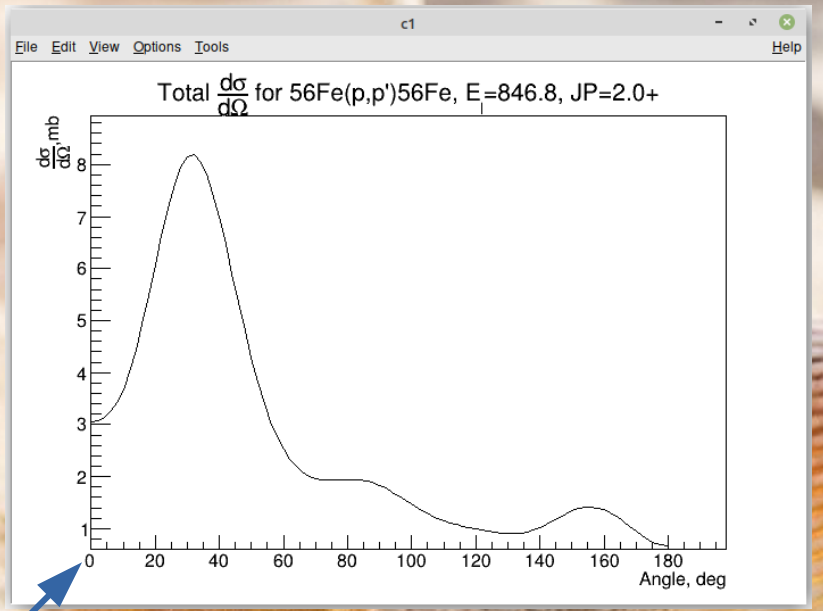
Usage examples

- Calculation of Q value for d(t,n)α reaction

```
user@jinr:~$ root -l
root [0] Nucleus d("2H"), t("3H"), n("n"), a("4He");
root [1] double Q=d.Mass+t.Mass-n.Mass-a.Mass
(double) 17.589895
root [2]
```

- Calculation and plotting 56Fe(p,p'$_1$) angular distribution

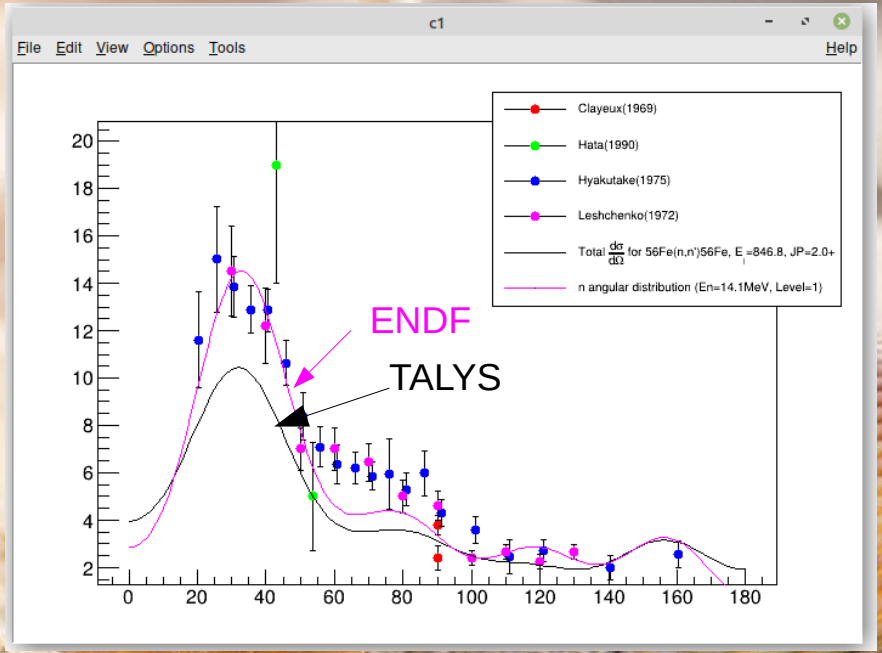
```
user@jinr:~$ root -l
root [0] Nucleus Fe("56Fe");
root [1] Fe.SetProjectileEnergy(20) //In MeV
root [2] Fe.GenerateProducts("p")
root [3] Fe.FindProductsByReaction("(p,p')")->Levels[1].GetAngularDistribution()->Draw("al")
```



Usage examples

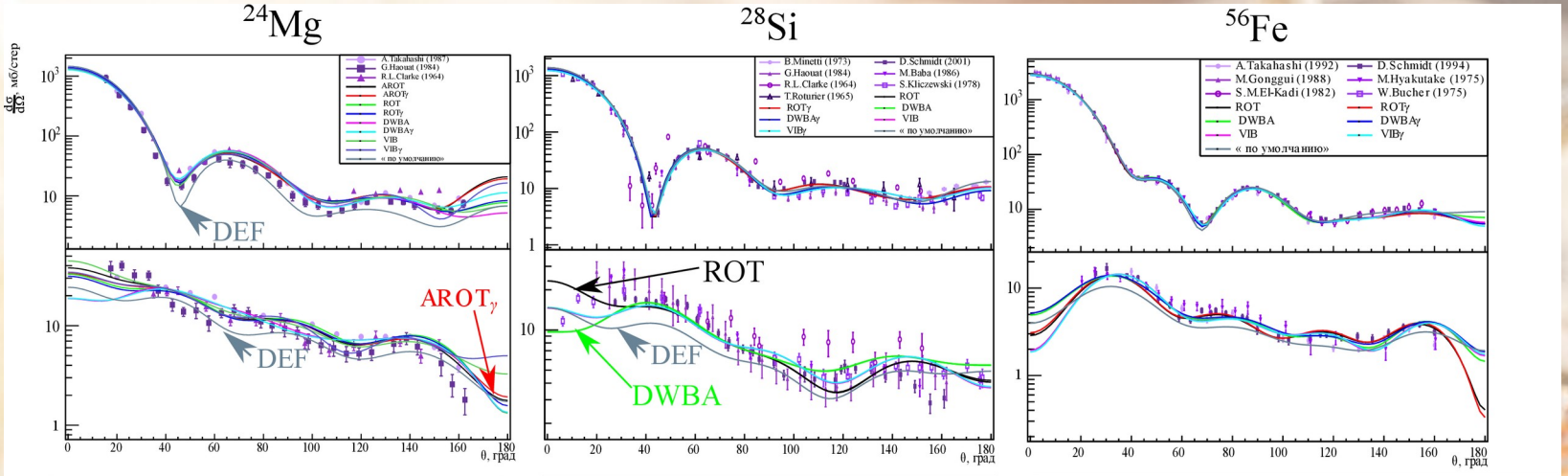
- Calculation and plotting 56Fe(n,n'$_1$) angular distribution with ENDF and EXFOR

```
user@jinr:~$ root -l
root [0] Nucleus Fe("56Fe");
root [1] Fe.GenerateProducts()//14.1 MeV n by default
root [2] Nucleus* Fe2=Fe.FindProductByReaction("(n,n')")
root [3] g1=Fe2->Levels[1].
GetEXFORTMultiGraphForAngularDistributions(13,15)
//Find data in 13-15 MeV range
root [3]  g2=Fe2->Levels[1].GetAngularDistribution()
root [4] g3=Fe2→Levels[1].
GetAngularDistribution("ENDF")
root [5] g1->Add(g2,"l"); g1->Add(g3,"l");
g1->Draw("ap");
```



Usage examples

Optical model fit



	AROT	AROT$_\gamma$	DEF(AROT)
χ^2/N	8,37	8,35	26,51
χ^2_n/N	7,16	7,19	28,52
χ^2_γ/N	17,47	17,11	11,39

DWBA	DWBA$_\gamma$	ROT	DEF(ROT)
8,975	9,028	9,436	18,017
8,818	8,873	9,276	18,229
13,553	13,531	14,080	11,845

ROT	ROT$_\gamma$	VIB	VIB$_\gamma$	DEF(DWBA)
6,13	6,08	6,10	6,29	10,05
5,20	5,15	5,18	5,40	8,89
13,62	13,52	13,49	13,50	19,32

AROT – асимметричный ротатор
ROT – ротатор
DWBA – Борновское приближение искаженных волн
VIB – осциллятор
DEF(...) – параметры "по умолчанию"
Индекс γ показывает, что в выборку включены γ-выходы

$$\chi^2/N = \frac{1}{N}\sum_{i=1}^{N}\frac{(x_i^{exp}-x_i^{th})^2}{(\sigma_i^{exp})^2},$$

χ^2/N – отклонение для всей выборки
χ^2_n/N – для ***нейтронной*** части
χ^2_γ/N – для ***γ-выходов***

Conclusion

- There is a new *automated* way to get nuclear data
- EXFOR processing now is not perfect but we are working on it
- If somebody asks, we will add new features

  https://github.com/terawatt93/TalysLib

  http://159.93.100.133:85/TalysLib.zip