Argonne NATIONAL LABORATORY

Filip G. Kondev
kondev@anl.gov

pyRULER - stand-alone version

- ✓ part of an IAEA-NDS-led project – huge credit to V. Dimitriou (IAEA)

Ruler & the meaning of lifetime …

- calculates the strength (in Weisskopf units) of a particular gamma-ray transition and compare to RUL

- links the lifetime of a particular level with the matrix element connecting the initial and final state:

$$P_\gamma(XL: I_i \rightarrow I_f) = \frac{\ln 2}{T^\gamma_{1/2}} = \frac{8\pi(L+1)}{L[(2L+1)!!]^2}\left(\frac{E_\gamma}{\hbar c}\right)^{2L+1} B(XL: I_i \rightarrow I_f)$$

Partial γ−ray Transition Probability   Reduced Transition Probability

$$B(XL: I_i \rightarrow I_f) = \frac{\left|\langle I_i | M(XL) | I_f \rangle\right|^2}{2I_i + 1}$$

contains the nuclear structure information

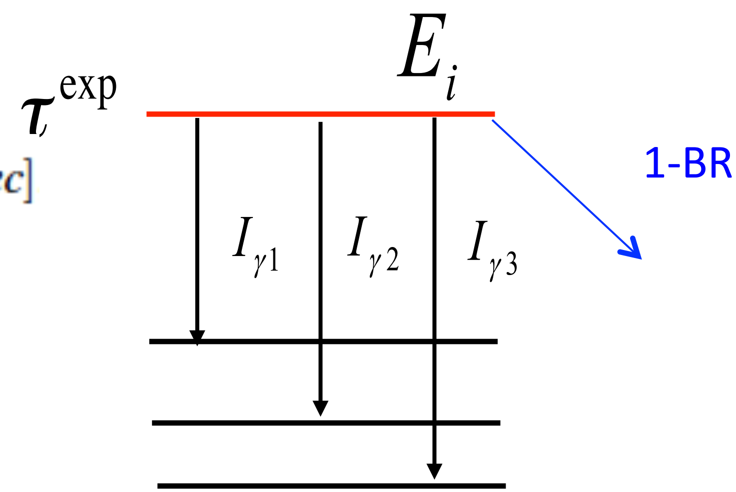
- defines the nuclear shape: quadrupole (BE2) – octupole (BE3) deformations, forbiddenness, etc. – implications for J^π

General formulae

$$\tau_\gamma^j = \frac{\tau^{exp}}{BR} \times \frac{\sum_{k=1}^{N} I_\gamma^k \times (1+\alpha_T^k)}{I_\gamma^j}$$

$$\Gamma \times \tau = \hbar = 0.6582 \times 10^{-15} [eV \cdot sec]$$

$$\Gamma_\gamma^j = \frac{\hbar}{\tau_\gamma^j} = \frac{\hbar \times BR}{\tau^{exp}} \times \frac{I_\gamma^j}{\sum_{k=1}^{N} I_\gamma^k \times (1+\alpha_T^k)}$$



$$\delta^2(\sigma'\lambda'/\sigma\lambda) = \frac{I_\gamma(\sigma'\lambda')}{I_\gamma(\sigma\lambda)} = \frac{\Gamma_\gamma(\sigma'\lambda')}{\Gamma_\gamma(\sigma\lambda)}.$$

$$\tau_\gamma(\sigma\lambda) = \tau_\gamma^j \times (1+\delta^2) \quad \text{and} \quad \Gamma_\gamma(\sigma\lambda) = \Gamma_\gamma^j \times \frac{1}{1+\delta^2},$$

$$\tau_\gamma(\sigma'\lambda') = \tau_\gamma^j \times \frac{1+\delta^2}{\delta^2} \quad \text{and} \quad \Gamma_\gamma(\sigma'\lambda') = \Gamma_\gamma^j \times \frac{\delta^2}{1+\delta^2},$$

$$\alpha_T^k = \frac{\alpha_T(\sigma\lambda) + \delta^2 \times \alpha_T(\sigma'\lambda')}{1+\delta^2}$$

- need $T_{1/2}$, BR, Eγ, Iγ, Mult., δ and α_T and their uncertainties

General formulae - cont.

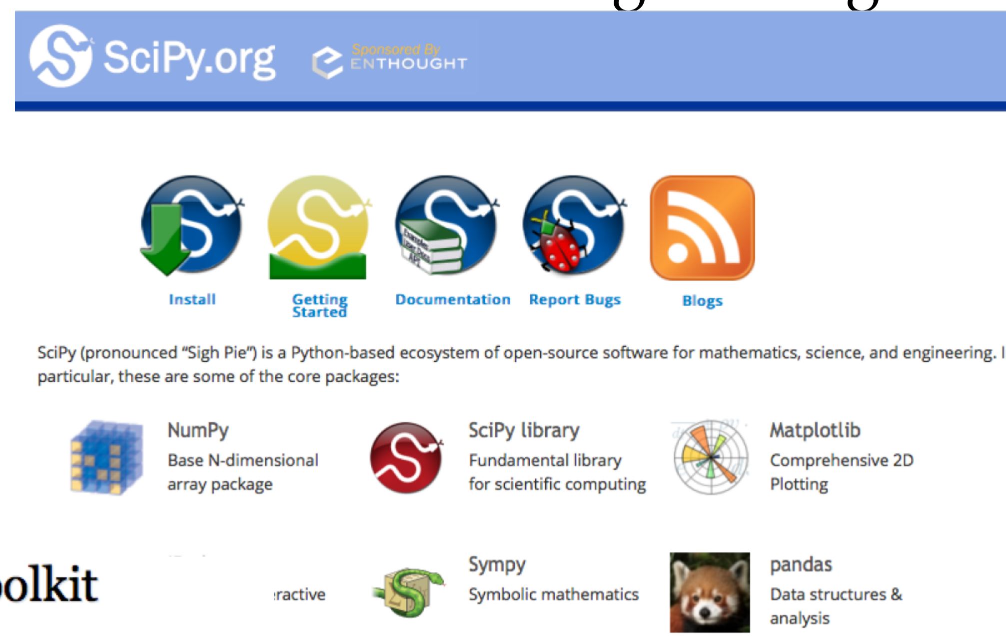
$$|M|^2\,(W.u.) = \Gamma_\gamma/\Gamma_W = \tau_W/\tau_\gamma = B_\gamma\downarrow/B_{sp}\downarrow = 1/F_W.$$

$$\tau_W(XL) = \frac{\hbar \cdot L[(2L+1)!!]^2}{2(L+1)\cdot R^{2L}}\left(\frac{3+L}{3}\right)^2\left(\frac{\hbar c}{E_\gamma}\right)^{2L+1}\cdot\begin{cases}1/e^2 & \text{for X} \equiv \text{E.}\\ R^2/(40\cdot\mu_N^2) & \text{for X} \equiv \text{M.}\end{cases}$$

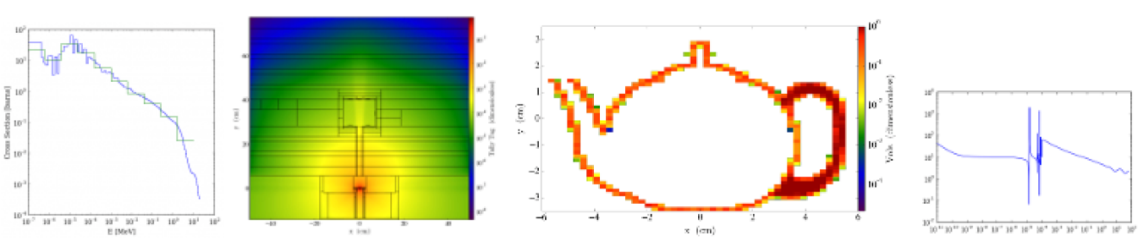
- if all quantities have symmetric uncertainties or ~ or no uncertainties – it is a piece of cake
- but what about the following scenarios:
  - ✓ **T1/2** +a/-b and/or **BR** +a/-b?
  - ✓ **Mult.** is given as *M1,E2* or *M1+E2* with no MR or *E0+M1+E2* – note that BrICC is giving you automatically an ICC value with assumed MR, which is not folded into calculation of the transition strength by ruler?
  - ✓ **MR** is missing for a mixed transition, **MR** +a/-b, or MR is a limit, e.g. **MR** > -2 or MR < 0.5
- what about if all of the above happen simultaneously …?

pyruler - why python?

- object-oriented – just like C, C++,Java, modern Fortran, etc.
- multi-platform – Mac OS X, Linux, Windows …
- a lot of stuff already developed for science and engineering …



PyNE: The Nuclear Engineering Toolkit



What about uncertainties?

http://pythonhosted.org/uncertainties/

uncertainties

Overview User Guide Uncertainties in arrays Technical Guide

```
>>> x = ufloat(0.20, 0.01)  # x = 0.20+/-0.01
```

```
>>> from uncertainties import ufloat_fromstr
>>> x = ufloat_fromstr("0.20+/-0.01")
>>> x = ufloat_fromstr("(2+/-0.1)e-01")  # Factored exponent
>>> x = ufloat_fromstr("0.20(1)")  # Short-hand notation
>>> x = ufloat_fromstr("20(1)e-2")  # Exponent notation
>>> x = ufloat_fromstr(u"0.20±0.01")  # Pretty-print form
>>> x = ufloat_fromstr("0.20")  # Automatic uncertainty of +/-1 on last digit
```

pyRuler (stand-alone)

- interative input

```
kondev — bash — 86×35
Last login: Wed Sep 30 15:39:17 on ttys000
c347962:~ kondev$ pyruler


                   Gamma-ray Transition Probbility
                    F.G. Kondev - ANL, April 2015
                          2015-10-01 10:35


Input the Nuclide (e.g. 176Ta or Ta176): 176Ta

Input the level energy (E,dE) in keV (e.g. 1000.0 5 or 1000.0+X 5): 1000 5

Input the Half-life of the level in ENSDF format (value unit unc): 10 ns 1

Input %IT of the level in ENSDF format (value unc) [100 %]:

Input the number of gamma rays that depopulate the level: 1

             Gamma ray # 1

Input gamma-ray energy in keV (e.g. 100.0 5): 100 1
Input gamma-ray intensity (e.g. 100.0 5) [100.0]: 100
Input gamma-ray multipolarity (e.g. M1, E2, M1+E2 ...): E2
Input gamma-ray ICC (e.g. 0.5 or 0.5 1):
```

← silent version of BrICC

pyRuler (stand-alone)-cont.

```
                    Nuclide= 176TA

      T1/2    =  (1.00+/-0.10)e-08  -> level T1/2 [s]
                 %IT    =  100.0   [%]


   Eg = 100.0 (10) keV;  Mult.= E2 ; Ef= 900.0(5) keV : final level energy

   Eg(100g)      =  100.0 (10)        : g-ray energy [keV]
   Ig(100g)      =  100.0             : g-ray intensity [rel.u.]
   ICC(100g)     =  3.71 (16)         : total ICC
   BR(100g)      =  100.0             : brachning intensity [%]
   T1/2g(100g)   =  4.7 (5)E-8        : partial g-ray T1/2 [s]
   T1/2W(100g)   =  9.7 (5)E-7        : Weiskopf T1/2 [s]
   WIDTH(100g)   =  9.7 (10)E-9       : partial g-ray width [eV]
   FW(100g)      =  4.9 (6)E-2        : hindrance factor
   BXL(100g)     =  20.5 (24)         : reduced probability in [W.u.]

c347962:~ kondev$
```

- automatically check for deviations from RUL and give warnings

pyRuler (stand-alone)-cont.

kondev — bash — 86×35

```
c347962:~ kondev$ pyruler


                    Gamma-ray Transition Probbility
                     F.G. Kondev - ANL, April 2015
                           2015-10-01 10:40


Input the Nuclide (e.g. 176Ta or Ta176): 176Ta

Input the level energy (E,dE) in keV (e.g. 1000.0 5 or 1000.0+X 5): 1000 1

Input the Half-life of the level in ENSDF format (value unit unc): 10 ns 1

Input %IT of the level in ENSDF format (value unc) [100 %]:

Input the number of gamma rays that depopulate the level: 1

             Gamma ray # 1

Input gamma-ray energy in keV (e.g. 100.0 5): 100.0 5
Input gamma-ray intensity (e.g. 100.0 5) [100.0]:
Input gamma-ray multipolarity (e.g. M1, E2, M1+E2 ...): m1+e2
Input gamma-ray mixing ratios (MR) for Mult:  M1+E2
       MR (e.g -0.5 or -0.5 2 or -0.5 +2-3): 0.7 1
Input gamma-ray ICC (e.g. 0.5 or 0.5 1):
```

pyRuler (stand-alone)-cont.

```
                    Nuclide= 176TA

     T1/2    =  (1.00+/-0.10)e-08  -> level T1/2 [s]
                %IT    =  100.0   [%]


Eg = 100.0 (5) keV;  Mult.= M1+E2 ; Ef= 900.0(11) keV : final level energy

                       Mult.= M1

Eg(100g)      =  100.0 (5)         : g-ray energy [keV]
Ig(100g)      =  100.0             : g-ray intensity [rel.u.]
ICC(100g)     =  4.15 (10)         : total ICC
BR(100g)      =  100.0             : brachning intensity [%]
MR(100g)      =  7.0 (10)E-1       : mixing ratio
T1/2g(100g)   =  7.7 (11)E-8       : partial g-ray T1/2 [s]
T1/2W(100g)   =  2.20 (3)E-11      : Weiskopf T1/2 [s]
WIDTH(100g)   =  5.9 (8)E-9        : partial g-ray width [eV]
FW(100g)      =  3.5 (5)E+3        : hindrance factor
BXL(100g)     =  2.9 (4)E-4        : reduced probability in [W.u.]

                       Mult.= E2

Eg(100g)      =  100.0 (5)         : g-ray energy [keV]
Ig(100g)      =  100.0             : g-ray intensity [rel.u.]
ICC(100g)     =  4.15 (10)         : total ICC
BR(100g)      =  100.0             : brachning intensity [%]
MR(100g)      =  7.0 (10)E-1       : mixing ratio
T1/2g(100g)   =  1.6 (3)E-7        : partial g-ray T1/2 [s]
T1/2W(100g)   =  9.66 (24)E-7      : Weiskopf T1/2 [s]
WIDTH(100g)   =  2.9 (6)E-9        : partial g-ray width [eV]
FW(100g)      =  1.6 (4)E-1        : hindrance factor
BXL(100g)     =  6.2 (13)          : reduced probability in [W.u.]
```

pyRuler - status

- close to be distributed – early 2016
  - ✓ implement non-symmetric uncertainties for T1/2 and BR
  - ✓ testing & documentation

future

- implement MC approach for the treatment of uncertainties
- program version that will work on an ENSDF file – developed by T. Kibedi (ANU) - https://www-nds.iaea.org/index-meeting-crp/CodesTM2/