Argonne NATIONAL LABORATORY

Member of the US Nuclear Data Program

RULER - stand-alone version

kondev@anl.gov

Technical Meeting on Codes for Nuclear Structure and Decay Data Evaluation,
3-7 December 2018, IAEA, Vienna

U.S. DEPARTMENT OF ENERGY

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

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

Partial γ-ray Transition Probability — Reduced Transition Probability

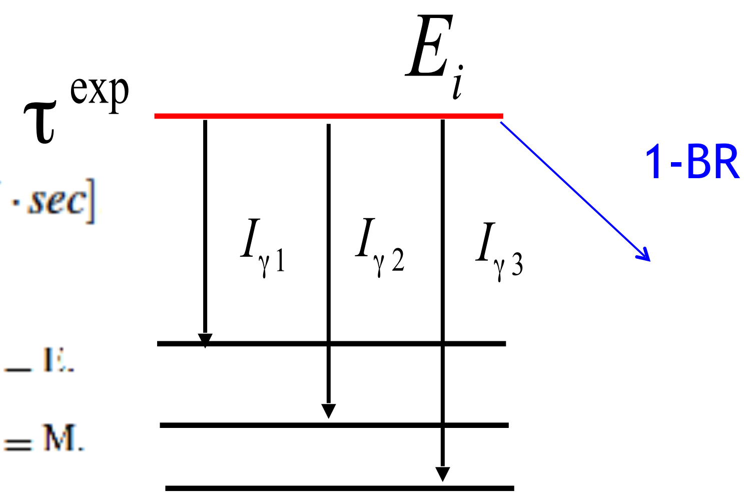
$$B(XL: I_i \rightarrow I_f) = \frac{\left|\langle I_i | M(XL) | I_f \rangle\right|^2}{2I_i+1}$$

contains the nuclear structure information

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

General formulae

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



$$\tau_\gamma^j = \frac{\tau^{exp}}{BR} \times \frac{\sum_{k=1}^{N} I_\gamma^k \times (1+\alpha_T^k)}{I_\gamma^j} \qquad \Gamma \times \tau = \hbar = 0.6582 \times 10^{-15}[eV \cdot sec]$$

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

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

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

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

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

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

ruler code (ENSDF analysis code)

https://www-nds.iaea.org/public/ensdf_pgm/index.htm

1.1 : February 18, 1984 — First exportable version
3.0 : October 20, 2004 — C.L.Dunford (Converted to Fortran 95)
3.2a: August 6, 2007 — T.W.Burrows (with input from Balraj Singh)
3.2d: January 20, 2009 — Jun-ichi Katakura (Fixed bug in TI initialization)

```
! ------------------------------------------------------------ original RULER code
!          TI(i) = 100.*TI(i)/tot
! FGK 11/17/2017 - this is where the wrong uncertainty, DTI, is calculated
!          IF(x.NE.0.)DTI(i) = TI(i)*DSQRT((DTI(i)/x)**2+(dtot/tot)**2)
! ------------ START FGK modification, using E.Browne - NIM A268 (1988) 541
         TGI(i) = RI(i)
         IF(x.NE.0.) THEN
            Rerr = (DTI(i)/TI(i))**2*(1.0 - 2.0*TI(i)/tot) + dtot1/tot**2
            TI(i) = 100.*TI(i)/tot
            DTI(i) = TI(i)*DSQRT(Rerr)

            D2 = sum1(i)/tot**2
            C2 = ((sum(i)/tot)**2)*(DRI(i)/RI(i))**2
            Rerr1 = D2 + C2 + (DCC(i)*RI(i)/tot)**2
            TGI(i) = 100.*TGI(i)/tot
            DTGI(i) = TGI(i)*DSQRT(Rerr1)
!              write(*,*) TI(i),DTI(i),TGI(i),DTGI(i)
         ELSE
            DTI(i) = 0.0
            DTGI(i) = 0.0
         END IF
! ---------- END FGK modification
```

Changes to the original code (as of 11/27/2017)

- recent fundamental constants
- Weisskopf lifetimes
- logic with error propagations and other "cosmetic" changes

works OK when symmetric uncertainties are involved

pyruler - stand-alone version

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 - cont.

- following our K-isomer horizontal evaluation
- interactive 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

comparison pyruler with jruler (JC)

uncertainties in $T_{1/2}$ & RI ~1-5%

	jruler	pyruler
100 keV, E1, RI=100	1.58 (8)E-6	1.58(8)E-6
100 keV, M1+E2, RI=100	M1: 3.4(6)E-5	M1: 3.4(6)E-5
δ=0.5(2)	E2: 0.37(24)	E2: 0.37(24)
71.64, E1+M2, RI=1.58	E1: 1.24(5)E-5	E1: 1.24(5)E-5
δ=0.018(9)	M2: 4 (4)	M2: 4 (4)
208.36, E1+M2, RI=89.3	E1: 3.17(8)E-5	E1: 3.17(8)E-5
δ=0.076(19)	M2: 19 (10)	M2: 19 (10)
321.31, E1+M2, RI=1.88	E1: 1.77(6)E-5	E1: 1.77(6)E-5
δ=0.175(10)	M2: 0.24 (3)	M2: 0.24 (3)

essentially identical results

mcerp
monte carlo error propagation

http://pythonhosted.org/mcerp/

A Simple Example

Let's start with the main import:

```
>>> from mcerp import *  # N, U, Gamma, Beta, correlate, etc.
```

Table of contents

Neglecting correlations or assuming independent variables yields a common formula among engineers and experimental scientists to calculate error propagation, the variance formula:[3]

$$s_f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 s_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 s_y^2 + \left(\frac{\partial f}{\partial z}\right)^2 s_z^2 + \cdots}$$

where s_f represents the standard deviation of the function f, s_x represents the standard deviation of x, s_y represents the standard deviation of y, and so forth. One practical application of this formula in an engineering context is the evaluation of relative uncertainty of the insertion loss for power measurements of random fields.[4]

It is important to note that this formula is based on the linear characteristics of the gradient of f and therefore it is a good estimation for the standard deviation of f as long as $s_x, s_y, s_z, \ldots$ are small compared to the partial derivatives[clarification needed].[5]

0.25-0.40%

```
>>> a=N(40.0,0.1)
>>> b=N(22.0,0.1)
>>> x=a/b
>>> x.mean
1.818219446201891
>>> x.std
0.009416826331139003
>>> a1=ufloat(40.0,0.1)
>>> b1=ufloat(22.0,0.1)
>>> a1/b1
1.8181818181818181+/-0.009431993562407709
>>>
```

12.5-20.0%

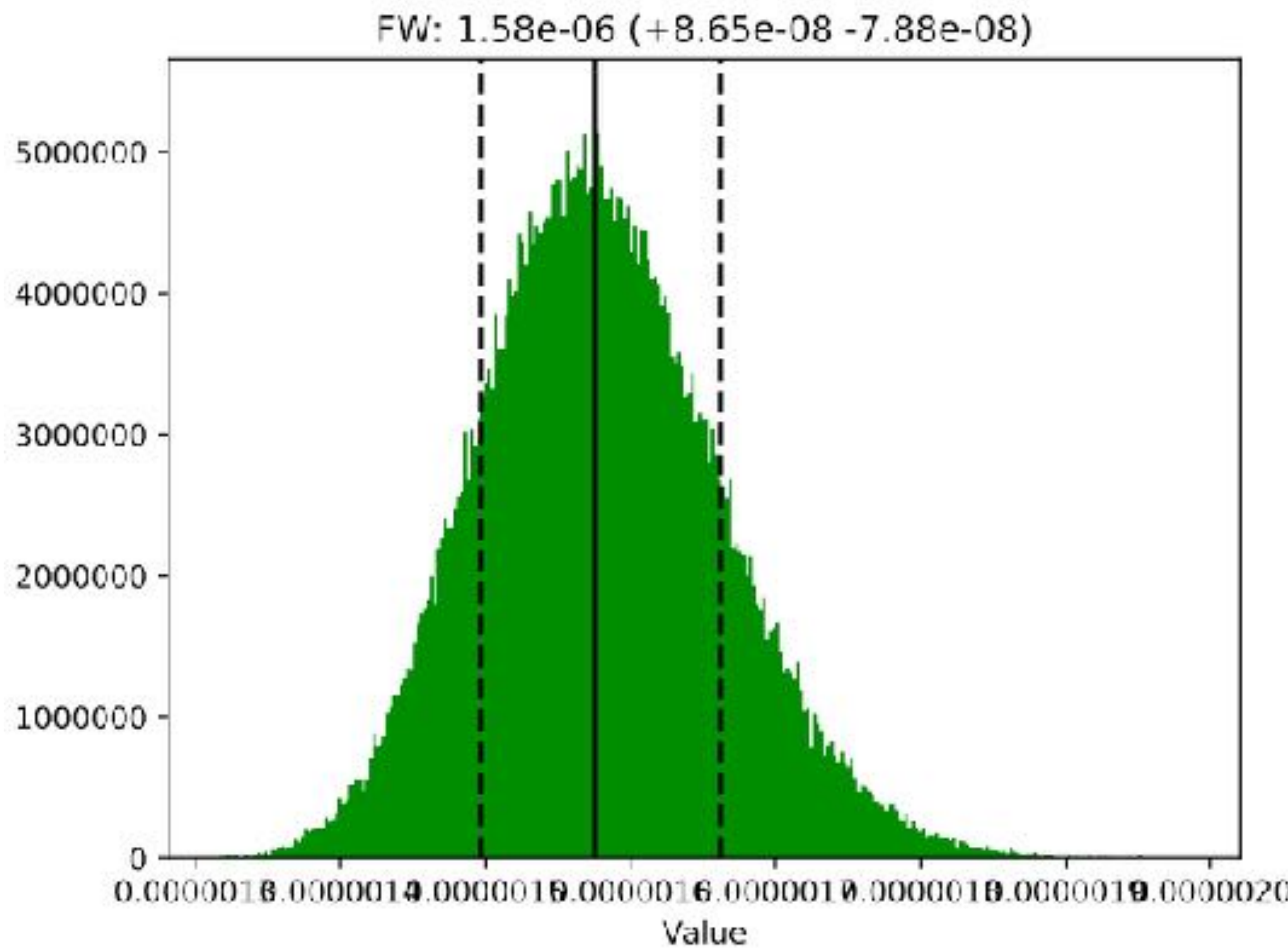
```
>>> a=N(40.0,5.0)
>>> b=N(22.0,5.0)
>>> x=a/b
>>> x.mean
1.934067377649564 9
>>> x.std
0.623875598763767 08
>>> a1=ufloat(40.0,5.0)
>>> b1=ufloat(22.0,5.0)
>>> a1/b1
1.8181818181818181+/-0.4715996781203854 5
>>>
```

pyruler_mc (stand alone version)

x_programs-benchmark — ruler_alone1.py

```
                    Gamma-ray Transition Probbility
                  Monte-Carlo Error Propagation (MCERP)
                   F.G. Kondev - ANL, November 2018
                          2018-12-03 22:40


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):

Input the Half-life of the level in ENSDF format (value unit unc): 100 ns 20

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): 200 1
Input gamma-ray intensity (e.g. 100.0 5) [100.0]:
Input gamma-ray multipolarity (e.g. M1, E2, M1+E2 ...): m1
Input gamma-ray ICC (e.g. 0.5 or 0.5 1):
```

pyruler_mc (stand alone version)



comparison pyruler with jruler (JC)

uncertainties in $T_{1/2}$ & RI ~20%

	jruler	pyruler_MC
71.64, E1+M2, RI=1.58 (5)	E1: 1.2(+10-5)E-5	E1: 1.2(+5-4)E-5
d=0.018(9)	M2: 4 (+11-3)	M2: 4(+5-3)
208.36, E1+M2, RI=100.0 (14)	E1: 3.1(+9-6)E-5	E1: 3.2(+13-9)E-5
d=0.076(19)	M2: 19 (+19-10)	M2: 19(+15-9)
321.31, E1+M2, RI=2.10 (4)	E1: 1.8(+15-8)E-5	E1: 1.8(+7-5)E-5
d=0.175(10)	M2: 0.24 (+25-12)	M2: 0.24 (+11-7)