Running TALYS with Oslo-guided inputs

Written by Cecilie, May 31, 2015

1. Level density

First, one needs to make sure that the level density used in TALYS corresponds as well as possible to the Oslo data. Normally, the constant-temperature model fits very well to the data above ≈ 2 MeV or so.

In the normalization of the level density with the 'counting' code, use the option for the constant-temperature formula:

```
100  11336.88       0       0.000e+00
101  11459.66       5       4.072e+01
Mass number A             < 89>:
Neutron or proton binding energy (Bn or Bp) (MeV)  <11.482>:

Choose constant temperature CT (1) or Fermi gas FG (2) formula <1>:
```

Then, choose the spin-cutoff model:

```
You may choose between 4 spin cut-off formulas:
1 The rigid moment of inertia formula (RMI) (E&B2006)
2 The Gilbert and Cameron formula (G&C) Can. J. Phys 43(1965) 1446
3 The constant temperature (CT) formula (E&B2009) and NPA 481 (1988) 189
4 The Fermi gas formula with appropriate cut-off parameter (E&B2009)
Type 1 for RMI: sig**2=0.0146*(A**(5/3))*T for FG+CT (E&B2006)
Type 2 for G&C: sig**2=0.0888*(A**(2/3))*a*T for FG+CT
Type 3 for E&B: sig**2=(0.98*(A**(0.29)))**2 for CT
Type 4 for E&B: sig**2=0.391*A**0.675*(E-0.5*Pa_prime)**0.312 for FG+CT


Choose RMI(FG+CT) (1), G&C(FG+CT) (2), E&B(CT) (3) or E&B(FG+CT) (4) <4>:
```

Probably, the RMI(FG+CT) (1) is best for heavy nuclei (heavier than Mo or so). Make sure to use the same model as you used to calculate the total level density and the neutron separation energy in the code 'd2rho'. Furthermore, get the proper Fermi gas parameters from the code 'robin' (unless you are using Goriely's calculations for normalization – but that's another story ☺).

```
You need to run the program Robin to get the Fermi-gas parameters a and E1:
Level density parameters a (1/MeV) <  8.856>:
Fermi-gas shift parameter E1 (MeV) <  0.277>:

You need to run the program Robin to get the constant temperature parameters
T:
Temperature parameter T (MeV) <  1.000>:
```

Now, 'counting' calculates the proper shift to make the level density with temperature T go straight through the rho(Sn) entered previously:

```
The level density goes through Rho(Bn), thus determining the
const. temp. shift parameter to be E0 =   0.617 MeV
```

These two parameters are then used in the code I attached (one for even spins: ct_nld_talysform_90Y, and one for odd spins: ct_nld_talysform_89Y.cpp).

The trick now is to generate a table in the format of the level–density model of Goriely (keyword ldmodel 4). The table in TALYS is located at /Applications/talys/structure/density/ground/goriely
The file with the tabulated level densities is called, for yttrium for example, z039.tab. Open the file, and make a copy of the first two columns, like this (hold the ‘alt’ button in and click to enwrap two columns):

U[MeV]	T[MeV
0.25	0.001
0.50	0.001
0.75	0.359
1.00	0.382
1.25	0.402
1.50	0.419
1.75	0.435
2.00	0.450
2.25	0.464
2.50	0.477
2.75	0.489
3.00	0.505
3.25	0.524
3.50	0.541
3.75	0.559
4.00	0.576
4.25	0.592
4.50	0.609
4.75	0.625
5.00	0.641
5.50	0.672
6.00	0.703
6.50	0.733
7.00	0.763
7.50	0.793
8.00	0.822
8.50	0.850
9.00	0.879
9.50	0.906
10.00	0.934
11.00	0.988
12.00	1.040
13.00	1.090
14.00	1.139
15.00	1.186
16.00	1.232
17.00	1.276
18.00	1.319
19.00	1.361
20.00	1.401
22.50	1.496
25.00	1.586
30.00	1.749
40.00	2.030
50.00	2.272
60.00	2.489
70.00	2.687
80.00	2.872
90.00	3.046
100.00	3.213
110.00	3.374
120.00	3.530
130.00	3.681
140.00	3.829
150.00	3.975

Now, make a folder where you have the code to generate the tabulated level densities, make a file with the two columns in (in my example: ex_and_T_90Y.txt), which is input for the code. Then modify the input parameters according to your nucleus and which spincut model you used (RMI in this example):

```
/* Declare parameters */
const double T_ct  = 0.850000;      // temperature parameter
const double E0    =-0.763974;  // shift for the CT level density

const double a_par = 9.050;     // level density parameter
const double A     = 90;      // mass A
const double E1    = -1.231;     // Fermi-gas shift for the spin cutoff
parameter, E&B 2005/2006
//const double Pa = 1.590; //  deuteron pairing shift Pa' for the spin
cutoff parameter, E&B 2009
```

Also make sure that the spin distribution and the very end of the code is the one that you used in the normalization of the level density and gamma strength, in this example, the RMI from von Egidy and Bucurescu, PRC (2005/2006):

```
/* Spin function, von Egidy & Bucurescu PRC 2005/2006 */
double spin_distribution(double Ex, double I){
    double g_Ex = 0.;
    double U = Ex - E1;
    if (U<0.) U=0.;
    double sigma2 = 0.;

    sigma2 = 0.0146*pow(A,(5./3.))*(1. + sqrt(1. + 4.*a_par*U))/(2.*a_par);
    g_Ex = (2.*I+1.)*exp(-pow((I+0.5),2.)/(2.*sigma2))/(2.*sigma2);
    return g_Ex;
}
```

Then, compile:

```
talys_nld_format> g++ -lm -o ct_nld_talysform_89Y ct_nld_talysform_89Y.cpp
```

and run the code:

```
talys_nld_format> ./ct_nld_talysform_89Y
```

Now, you should get an output file with a long table in the form of the goriely tables. Copy now the first lines, up to U = 20 MeV, and paste into the zXXX.tab for your specific nucleus. In this way you “trick” TALYS into using the data-compatible model.

NB!!! In your input file for TALYS, make sure that you have the following:

```
ctable 39 89 0
ptable 39 89 0
```

If not, TALYS will potentially take a shift and a slope correction meant for the original calculations of Goriely and apply them on the new level density.

Now you can run TALYS, and check that the output D0 matches the one that you used, wit a reasonable precision. For example, for 89Y, I used D0 = 72.6 eV. TALYS gives me in the output:

```
########## GAMMA STRENGTH FUNCTIONS, TRANSMISSION COEFFICIENTS AND CROSS
SECTIONS ##########

 Gamma-ray information for Z= 39 N= 50 ( 89Y )
 S-wave strength function parameters:
```

```
 Exp. total radiative width=   0.17000 eV +/- 0.00000 Theor. total radiative
width=        0.18881 eV
 Exp. D0                          =         0.00 eV +/-      0.00 Theor. D0
=          72.62 eV
```

2. Gamma strength function

For the gamma strength, I don't do anything nearly as fancy as Stephane does. I basically just fit models to the data points, and put these models into TALYS.

First, I decide upon a proper E1 model for the GDR, and compare with (gamma,n) data. The easiest is often to use the GLO model with a constant temperature. Then I put the constant temperature into the source code in TALYS, into the subroutine called 'fstrength.f', which is found in /Applications/talys/source
like this:

```
c
c 1. Kopecky-Uhl generalized Lorentzian.
c
c Efs      : fast particle energy for gamma ray strength function
c Tnuc     : nuclear temperature
c k0       : index of incident particle
c Einc     : incident energy
c delta    : energy shift
c S        : separation energy per particle
c alev     : level density parameter
c ggredep0 : energy dependent damping width at zero gamma energy
c ggredep  : energy dependent damping width
c twopi    : 2.*pi
c enum     : enumerator of Lorentzian
c denom    : denominator of Lorentzian
c factor1-2: help variables
c qrpaexist: flag for existence of tabulated QRPA strength functions
c
        Tnuc=0.
        if (strength.eq.1.and.l.eq.1.and.irad.eq.1) then
          if (k0.gt.0.or.Egamma.ne.Einc) then
            e=min(Efs,20.)+S(Zcomp,Ncomp,k0)-delta(Zcomp,Ncomp,0)-Egamma
            if (e.gt.0..and.alev(Zcomp,Ncomp).gt.0.)
     +        Tnuc=sqrt(e/alev(Zcomp,Ncomp))
          endif
c          Tnuc=0.3 !FOR RECOMMENDED 89,90Y (HF08)
          Tnuc=0.43 !FOR UPPER LIMIT, 89Y (FG05)
c          Tnuc=0.26 !FOR UPPER LIMIT, 90Y (FG05)
c          Tnuc=0.42 !FOR LOWER LIMIT, 89Y (FG09)
c          Tnuc=0.25 !FOR LOWER LIMIT, 90Y (FG09)
ggredep0=ggr1*twopi**2*Tnuc**2/egr2
          ggredep=ggredep0+ggr1*Egam2/egr2
          enum=ggredep*Egamma
          denom=(Egam2-egr2)**2+Egam2*ggredep**2
          factor1=enum/denom
          factor2=0.7*ggredep0/(egr1**3)
          fstrength=fstrength+kgr1*sgr1*ggr1*(factor1+factor2)
        endif
```

The GDR parameters should go into the input file for TALYS like this:

```
ggr 39 89 7. E1
egr 39 89 17.80 E1
sgr 39 89 233.0 E1
```

If you have two GDR bumps you can do

```
ggr 39 89 7. E1 1
```

…
for the first (just add the number ‘1’ behind all the 3 Lorentzian parameters), and

```
ggr 39 89 5. E1 2
```
…
for the second.

For the M1 spin flip, you can do the same thing with the parameters:

```
# M1 SPIN-FLIP
ggr 39 89 2.7 M1
egr 39 89 9.5 M1
sgr 39 89 4.12 M1
```

If you need additional resonances, they can be added as pygmy resonances like this (one for an E1 and one for an M1 resonance):

```
# EXTRA M1 STRENGTH AS LOW-LYING PYGMY
gpr 39 89 2.20 M1
epr 39 89 2.80 M1
spr 39 89 0.13 M1

# EXTRA E1 STRENGTH AS LOW-LYING PYGMY TO REPRODUCE DATA
gpr 39 90 2.5 E1
epr 39 90 7.0 E1
spr 39 90 3.5 E1
```

Finally, if you have an upbend, I put it as an M1 into the ‘fstrength.f’ like this:

```
c
c 2. Brink-Axel standard Lorentzian.
c
        if (strength.eq.2.or.((strength.eq.3.or.strength.eq.4).and.
     +    .not.qrpaexist(Zcomp,Ncomp)).or.l.ne.1.or.irad.ne.1) then
          enum=0.
          if (Egamma.gt.0.001) then
            enum=ggr2*Egamma**(3-2*l)
            denom=(Egam2-egr2)**2+Egam2*ggr2
            fstrength=fstrength+kgr1*sgr1*enum/denom
            ! Implementation of upbend as part of the M1 strength function,
            ! using an exponential form as recommended in
            ! Schwengner et al., PRL 111, 232504 (2013)
            ! upbend parameterization for 73Ge, CT normalization
            if(irad.eq.0.and.l.eq.1) then
              fstrength=fstrength+
c              ! IN ACCORDANCE WITH RONALD'S CALCS FOR 89Y
               ! UPPER LIMIT 89Y, FG05
     +        (5.0*10.**(-08)*exp(-2.5*Egamma)) ! UPPER 89Y
c     +        (4.0*10.**(-08)*exp(-2.5*Egamma)) ! LOWER 89Y
c     +        (5.0*10.**(-08)*exp(-2.6*Egamma)) ! UPPER 90Y
c     +        (4.0*10.**(-08)*exp(-2.6*Egamma)) ! LOWER 90Y

            endif
          endif
        endif
```

NB!! When ‘fstrength.f’ is changed, it needs to be re-compiled, so you need to run the talys.setup in the TALYS folder each time you either change the GLO temperature or do other modifications to the source code. This is not

necessary when putting in the level density tables as they are read “on the fly”, same with the input file of course.

Now, you can again run TALYS and check that the output <Gamma_gamma> is reasonable compared with what you used for normalizing the gamma strength function. I used 188 meV and the TALYS output gives me:

```
########## GAMMA STRENGTH FUNCTIONS, TRANSMISSION COEFFICIENTS AND CROSS
SECTIONS ##########

 Gamma-ray information for Z= 39 N= 50 ( 89Y )
 S-wave strength function parameters:
 Exp. total radiative width=   0.17000 eV +/- 0.00000 Theor. total radiative
width=        0.18881 eV
 Exp. D0                           =        0.00 eV +/-      0.00 Theor. D0
=          72.62 eV
```

So, not exact, but good enough (less than 1% deviation). If it doesn’t match, check that the extrapolations to the transmission coefficient are in accordance with the models that you use. This is done by reading in the transext.nrm file and transform transmission coefficient to gamma strength:

```
    ifstream transfile_90Y("y90_FG05_pwave/transext.nrm");
...

    i=0;
     while(transfile_90Y){
       transfile_90Y >> x;
        trans_90Y[i] = x/(2.*3.14159265359*pow(energy_90Y[i],3.));
       i++;
     }
    transfile_90Y.close();
```

Then this can be put in a TGraph in a Root-script and plotted together with the model inputs. Finally, plot the output gamma strength from TALYS (found just below the `########## GAMMA STRENGTH FUNCTIONS, TRANSMISSION COEFFICIENTS AND CROSS SECTIONS ########## in the output file)` and compare with your data to make sure the visual agreement is OK too.

Then, you can look at, for example, the rp… files with the (n,g) final nucleus, and get the total (n,g) cross section, of the astrorate.g to see the astrophysical (n,g) rates. These are now the ones as predicted with the models compatible with the data.

Good luck ☺