PyNE ENSDF Example

November 6, 2014

1 ENSDF decay data

There is a basic ENSDF data interface (excluded here for previtity), and there is also an extended interface that provides access to a large portion of the decay data in the ENSDF dataset. For example, lets look at Cs-137:

In [10]:
```
from pyne import data

# we can look up most decay data by parent or by decay energy
# returns a list of gamma ray level pairs from ENSDF decay data
# using a given parent's state id.
decay_pairs = data.gamma_from_to_byparent(551370000)

print "\nDecay pairs:", decay_pairs, "\n"

# this resturns a list of gamma ray energies (and errors) given a parent
energies = data.gamma_energy(551370000)

print "Energies:", energies, "\n"
```

```
Decay pairs: [(561370001, 561370000), (561370002, 561370000)]

Energies: [(283.5, 0.1), (661.657, 0.003)]
```

In [11]:
```
# these are just relative intensities (and errors)
p_intensities = data.gamma_photon_intensity(551370000)

print "\nPhoton intensities:", p_intensities, "\n"

# This converts the relative intensities to decays per 100 decays of the parent
photon_br, photon_br_error = data.decay_photon_branch_ratio(551370000,561370000)
p_final_intensities = []

print "Photon branch ratio:", photon_br, "\n"

for item in p_intensities:
    # compute the intensities ignore the errors (for now)
    p_final_intensities.append(photon_br*item[0])

print "Photon final intensities:", p_final_intensities, "\n"


# Alpha intensities are a bit trickier
from pyne import rxname
```

```
        # This is just intensity per 100 alpha decays of U-229 with no errors (yet)
        a_intensities = data.alpha_intensity(922290000)

        print "\nAlpha intensities:", a_intensities, "\n"

        # Find the alpha branch intensity
        a_br = data.branch_ratio(922290000,rxname.child(922290000,'a','decay'))

        print "Alpha branch ratio:", a_br, "\n"

        # Calculate the intensities of each alpha per 100 decays of the parent
        a_final_intensities = []
        for item in a_intensities:
            a_final_intensities.append(a_br*item)

        print "Alpha final intensities:", a_final_intensities, "\n"
```

```
Photon intensities: [(0.00058, 8e-05), (85.1, 0.2)]

Photon branch ratio: 1.0

Photon final intensities: [0.00058, 85.1]


Alpha intensities: [1.0, 3.0, 1.0, 11.0, 20.0, 64.0]

Alpha branch ratio: 0.2

Alpha final intensities: [0.2, 0.6000000000000001, 0.2, 2.2, 4.0, 12.8]
```

1.1 Another possible use is the search for candidate gamma rays

In [8]:
```
# The default range is +- 1 keV, which returns a lot of stuff
possible_parents = data.gamma_parent(661.657)
print "\nThere are", len(possible_parents), "possible parents\n"

# this returns gamma ray level pairs based on gamma ray energy
from_to = data.gamma_from_to_byen(661.657)

# a list of gamma ray intensities from a given gamma ray energy
intensity_list = data.gamma_photon_intensity_byen(661.657)
```

```
There are 865 possible parents
```

In [12]:
```
# we can downselect this
import numpy as np
from pyne import nucname
half_life_s = []
final_ints = []
parents = []

for i, item in enumerate(possible_parents):
    # gammas that don't have from-to pairs get lost here
    p_br = data.decay_photon_branch_ratio(item,nucname.groundstate(from_to[i][0]))
```

```
            # Select items that have non-zero branch ratios + intensities over 5% of all
            # decays + half lives over 10 days
            if p_br[0] > 0.0 and intensity_list[i][0]*p_br[0] > 5.0:
                if nucname.groundstate(from_to[i][0]) != item and \
                data.half_life(item,False) > 60.*60.*24.*10.:
                    parents.append(item)
                    half_life_s.append(data.half_life(item,False))
                    final_ints.append(intensity_list[i][0]*p_br[0])

        # now we have a managably short list
        for i, item in enumerate(parents):
            print("possible candidate: {0}".format(nucname.name(item)))
            print("half life: {0} s/ {1} days/ {2} years".
                  format(half_life_s[i],half_life_s[i]/(60.*60.*24.),\
                         half_life_s[i]/(60.*60.*24.*365.25)))
            print("intensity: {0} %".format(final_ints[i]))
            print("######################################################################")
```

```
possible candidate: Cs137
half life: 949252608.0 s/ 10986.72 days/ 30.08 years
intensity: 85.1 %
######################################################################
possible candidate: I126
half life: 1117152.0 s/ 12.93 days/ 0.0354004106776 years
intensity: 32.8848 %
######################################################################
possible candidate: Sb126
half life: 1067040.0 s/ 12.35 days/ 0.0338124572211 years
intensity: 85.6732 %
######################################################################
```

In []: