Fission implementation

C.M. Mattoon

October 30, 2014

Current implementation for fission reactions (in both ENDF-6 and GND) is similar to other reactions:

- Reaction (fission)
  - Cross section
    - includes resolved/unresolved resonance contributions
  - output channel
    - contains both an average 'Q-value' and energy-dependent Q-values for products, neutrons, gammas, betas, neutrinos (MT 458)
    - List of products: prompt and delayed neutrons, gammas

Current implementation for fission reactions (in both ENDF-6 and GND) is similar to other reactions:

- Inside the list of products:
  - prompt neutron product
    - energy-dependent multiplicity (prompt nubar)
    - outgoing distributions (PFNS, angular term assumed isotropic)
  - delayed neutron products (divided into groups by decay time)
    - decay rate
    - energy-dependent multiplicity (delayed nubar)
    - outgoing distributions, often stored as 'general evaporation'
  - gammas
    - multiplicity & distributions
  - fission products
    - yields (independent and cumulative) given in a separate sub-library
  - other products (betas, neutrinos)
    - only present in MT 458

1st, 2nd, 3rd, 4th-chance fission are stored in separate 'fissionComponent' reactions

- Each has its own cross section, but links back to the total fission cross section for product multiplicities and distributions

Treating fission like all other reactions (almost) has pros and cons

- Advantages:
  - fission is complex, but still ‘just another open reaction channel’. Current layout emphasizes that

- Disadvantages:
  - Reaction naming convention breaks down for fission
  - multi-chance cross sections aren’t ‘derived’ or ‘summed’, so they require special markup

- Fission products are treated very differently. Unify them with other <product> elements inside fission reaction?

Naming conventions

- Most reactions are named based on list of products
  - e.g., n + Pu239 → n[multiplicity:’2’] + Pu238
- This gets unwieldy for fission (even if only neutrons and gammas are listed):
  - n + Pu239 → n[multiplicity:'energyDependent', emissionMode:'prompt'] + n[emissionMode:'6 delayed'] + gamma [total fission]
- What about just ‘total fission’?

example of fissionEnergyRelease section

```xml
<outputChannel genre="NBody" Q="1.98902e8 eV">
  <fissionEnergyReleased nativeData="polynomial">
    <polynomial order="2" energyUnit="eV" hasUncertainties="true">
      <promptProductKE> 175550000.0 400000.0 -0.4566 0.04566 0.0 0.0</promptProductKE>
      <promptNeutronKE> 6128000.0 100000.0 0.3428 0.03428 0.0 0.0</promptNeutronKE>
      <delayedNeutronKE> 2990.0 440.0 0.0 0.0 0.0 0.0</delayedNeutronKE>
      <promptGammaEnergy> 6741000.0 470000.0 0.1165 0.01165 -0.0017 0.00017</promptGammaEnergy>
      <delayedGammaEnergy> 5170000.0 60000.0 -0.075 0.0075 0.0 0.0</delayedGammaEnergy>
      <delayedBetaEnergy> 5310000.0 60000.0 -0.075 0.0075 0.0 0.0</delayedBetaEnergy>
      <neutrinoEnergy> 7140000.0 90000.0 -0.1 0.01 0.0 0.0</neutrinoEnergy>
      <nonNeutrinoEnergy> 198902000.0 1090440.0 -0.1473 0.01473 -0.0017 0.00017</nonNeutrinoEnergy>
      <totalEnergy> 206042000.0 1180440.0 -0.2473 0.02473 -0.0017 0.00017</totalEnergy></polynomial>
```

Trouble: mixing coefficients, uncertainties

- Add support for pointwise energy release
- Associate energy release with <product> element? Would require having explicit ‘neutrino’, ‘beta’, ‘fissionProduct’ as <product>

Adding support for P(multiplicity | E)

- Similar to other probability distributions like P(E’ | E), except multiplicity should be an integer
- What about outgoing neutron spectra? Should those also depend on multiplicity, as in P(E’ | E, multiplicityt)?

R. Vogt suggestion: support storing 'realizations' of fission events?

- For each incident neutron energy, store multiple realizations of all products
  - prompt neutrons + energy, angle
  - fission fragments
  - plus delayed neutrons, betas, gammas
- Useful for Monte Carlo codes (preserves correlations)
  - However, how to interpolate between incident energies?

How should we handle fission product yields?

- For ENDF-6 compatibility, just need a table like

Product	Yield (thermal)	dYield (thermal)	Yield (0.5 MeV)	dYield (0.5 MeV)
V66	2.05032e-19	1.3122e-19	4.48456e-18	2.87012e-18
...				

- Two of these tables (independent and cumulative yields)

- Not integrated into <reaction> yet.
  - Special child element for <independentProductYields> and <cumulativeProductYields>?
  - In future, should these be stored instead as separate products with incident-energy-dependent multiplicity?

Q-matrix

- The `Q-matrix’ is constructed from decay data + time cutoff, transforms independent to cumulative product yields. Generally a sparse array
  - Support (optionally) storing this matrix

- What about spontaneous fission? Should probably use similar format inside the particle database