Angular Correlations in Gamma Deexcitation

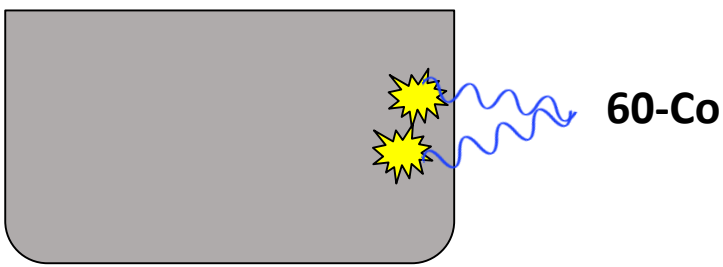
Micah Buuck, Jason Detwiler, **Ian Guinn**, Aobo Li

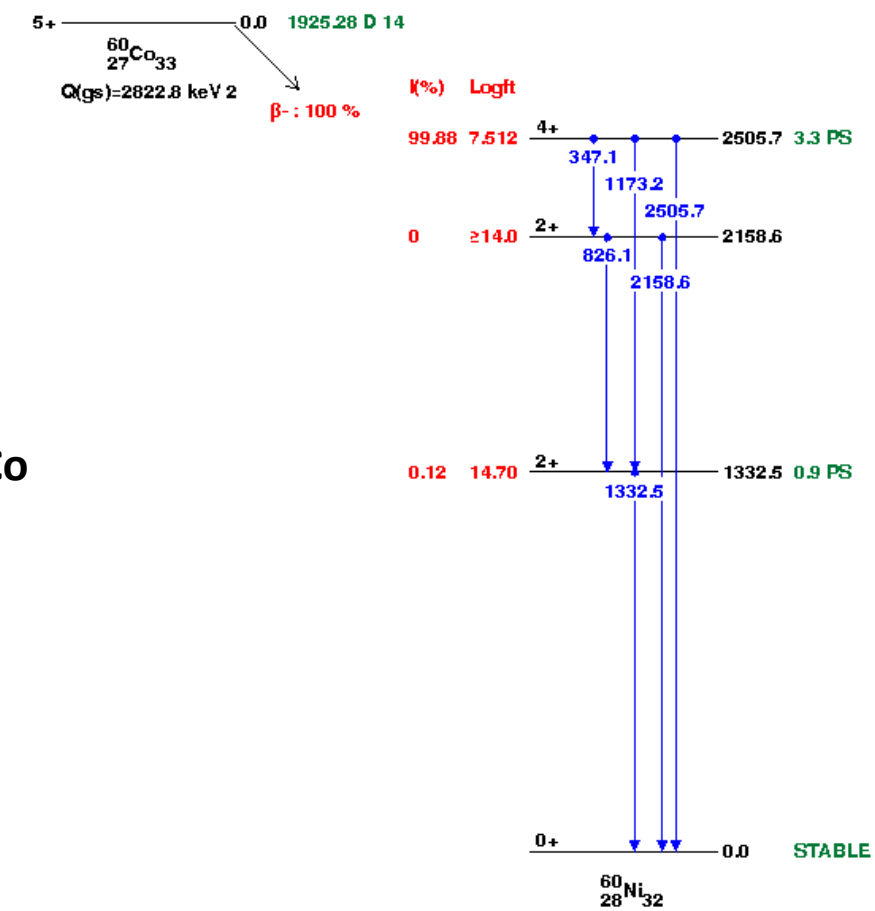
Geant 4 RDM Mini Workshop 2017

April 5, 2017

Why angular correlations?

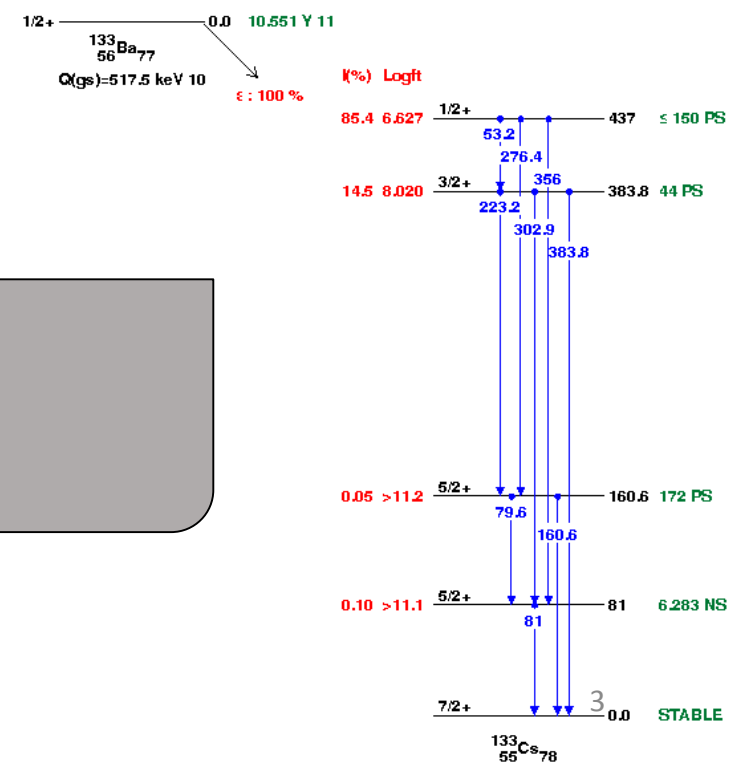
- Coincident and sum gammas important to model in low BG experiments





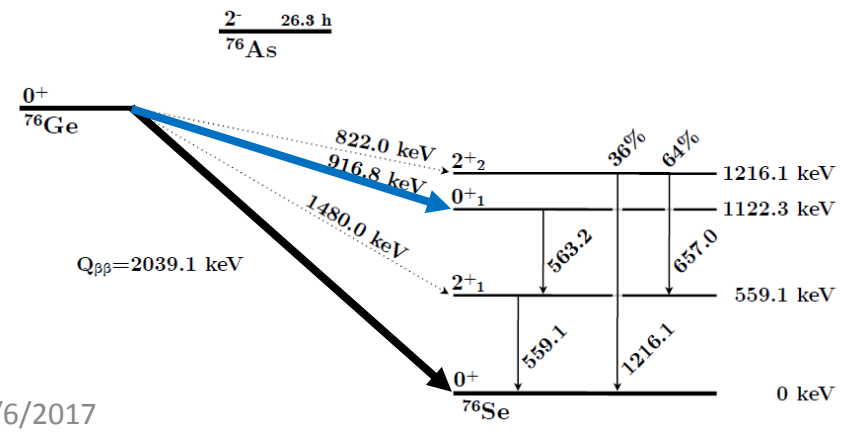
Why angular correlations?

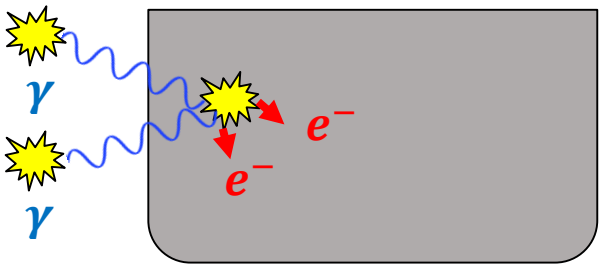
- Coincident and sum gammas important to model in low BG experiments
- Back-to-back gammas can be useful for calibrations and detector measurements



Why angular correlations?

- Coincident and sum gammas important to model in low BG experiments
- Back-to-back gammas can be useful for calibrations and detector measurements
- Coincident gammas useful for identifying some rare processes (e.g. double-β decay to excited states)





IT Multipole Expansion

- Consider the transition:

$$|J_1^{\pi}, M_1\rangle \rightarrow |J_2^{\pi'}, M_2\rangle + |L, M\rangle$$

- In this transition, the amplitude for photon emission in direction ***k*** is

$$\text{Amplitude}(\mathbf{k}) = \sum_{M_2,L,M} A_{J_1,M_1,\pi,J_2,M_2,\pi',L,M}\, T_{J_1,J_2,L}\, D_{L,M}(\mathbf{k})$$

Clebsch-Gordannery

Nuclear Data

Spherical Harmonics

- Sum over M_1s in density matrix describing polarization state

Sampling Gamma Emission

Typical calculation for an excited nucleus with J=J_1 that is going to de-excite to levels with J = J_2, J_3, … down to the ground state:

1. Start unpolarized: the "statistical tensor" representing the entangled nuclear state is trivial (rank 1, equal to 1).
2. Sample $\boldsymbol{k}$ based on J_1^{π}, $J_2^{\pi'}$, and L (and sometimes also L' and δ).
3. Update the statistical tensor based on the sampled value of $\boldsymbol{k}$: the statistical tensor now represents a non-trivial entanglement of M_2 states.
4. Repeat from step 2 for $J_2 \rightarrow J_3$, $J_3 \rightarrow J_4$, etc. until you reach the ground state.

Required elements

$$\text{Amplitude}(\mathbf{k}) = \sum_{M_2,L,M} A_{J_1,M_1,\pi,J_2,M_2,\pi',L,M}\, T_{J_1,J_2,L}\, D_{L,M}(\mathbf{k})$$

- Classes to represent the nuclear polarization (**G4NuclearPolarization**) and manipulate it (**G4PolarizationTransition**) in gamma cascades
- Utility classes
  - **G4Clebsch** (Wigner6J() and Wigner9J() functions, etc.)
  - **G4LegendrePolynomial** (N^{th} coefficient, associated Legendre polynomial evaluation, etc.)
  - **G4PolynomialPDF** (set coefficients, randomly sample)
- Multipolarity data into PhotonEvaporation data files
- Code in photo_evaporation that reads in the data and uses the classes to generate the angular distributions
- **All required pieces are now in place!** Just working out a few final bugs.

Geant4 Integration

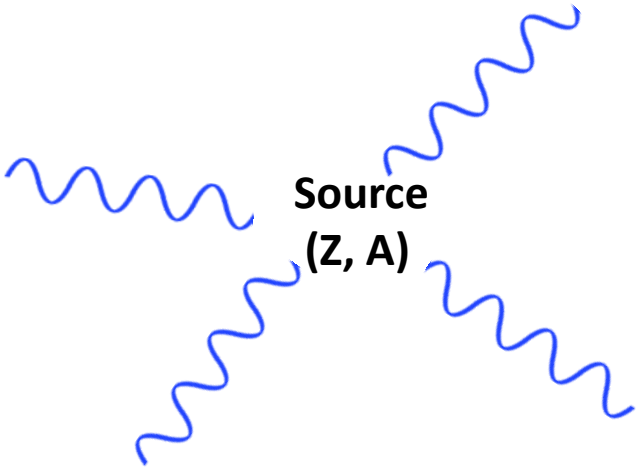
- G4Fragment (represents the decaying nucleus) has a G4NuclearPolarization pointer, NULL by default
- G4GammaTransition uses an available nuclear polarization to generate the gamma angular distribution and updates it to the required polarization state after gamma decay.
- G4ITDecay manages passing daughter fragments with non-trivial nuclear polarization as parent fragment in subsequent transitions

Testing: low level

- Low level classes have been extensively tested, as Jason showed previously
  - Retested with current implementation
- PhotoEvaporation4.3.2/correlated_gamma appears to have all the right data
  - Minor bug in reading of mixing ratio, workaround identified
- Verbose output shows that all the right calculations appear to be being performed.

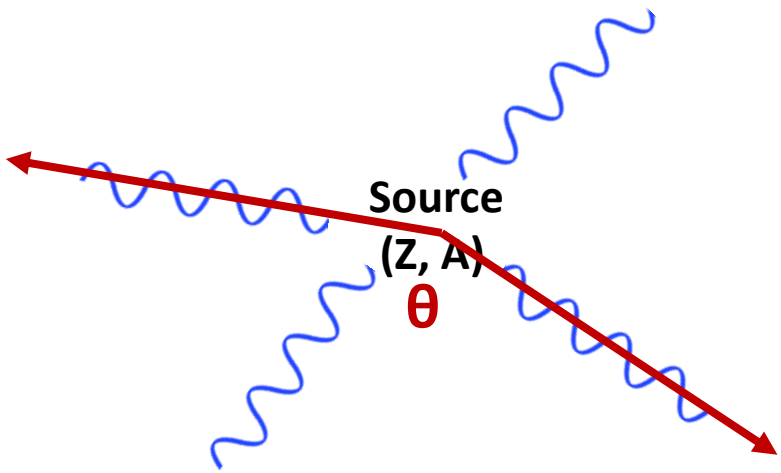
Testing: high level

- Aobo Li wrote a test program:
  - Input: Z, A, E1, E2, ΔE
  - Empty spherical volume with isotope generated at origin and non-interacting decay products



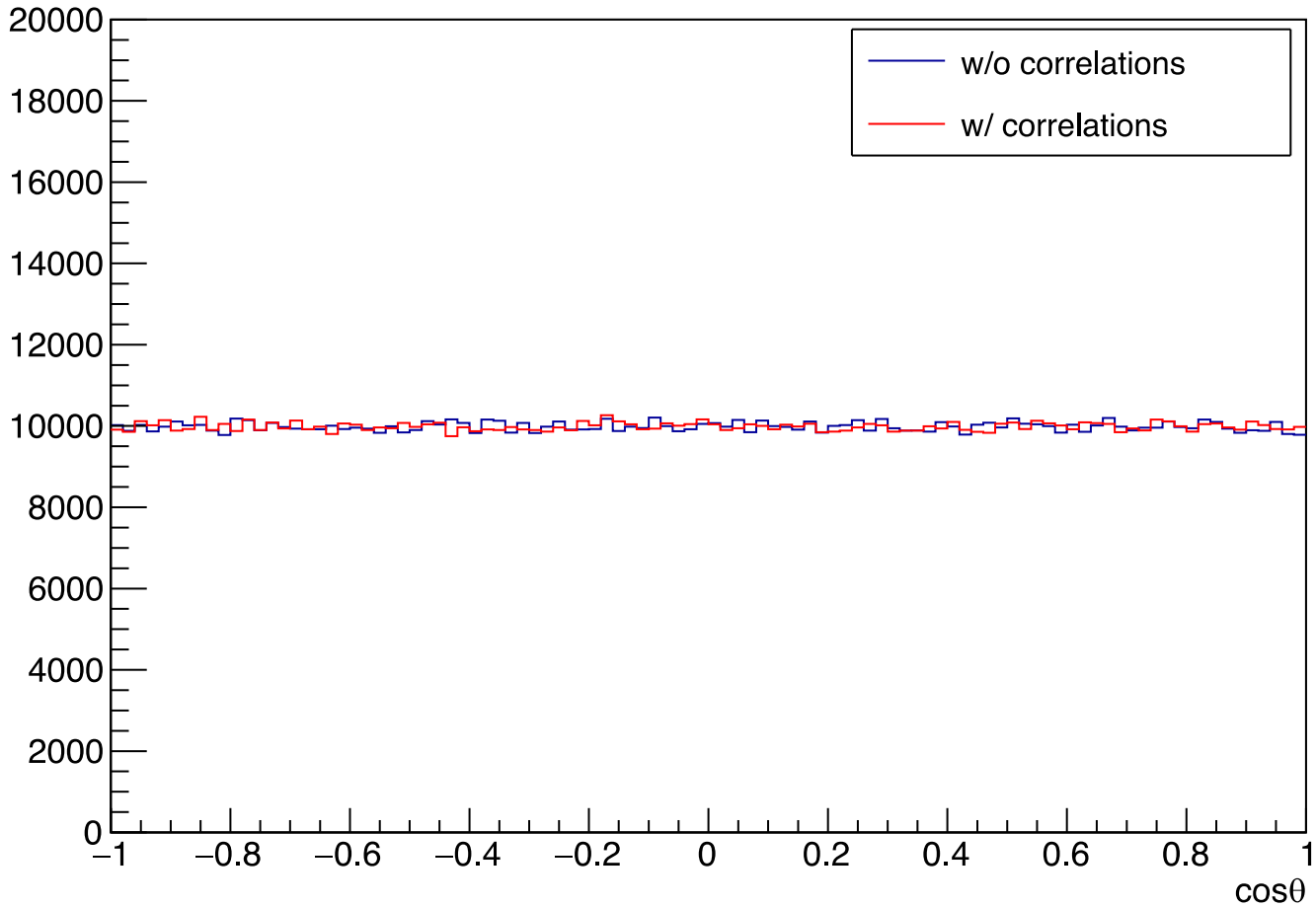
Testing: high level

- Aobo Li wrote a test program:
  - Input: Z, A, E1, E2, ΔE
  - Empty spherical volume with isotope generated at origin and non-interacting decay products



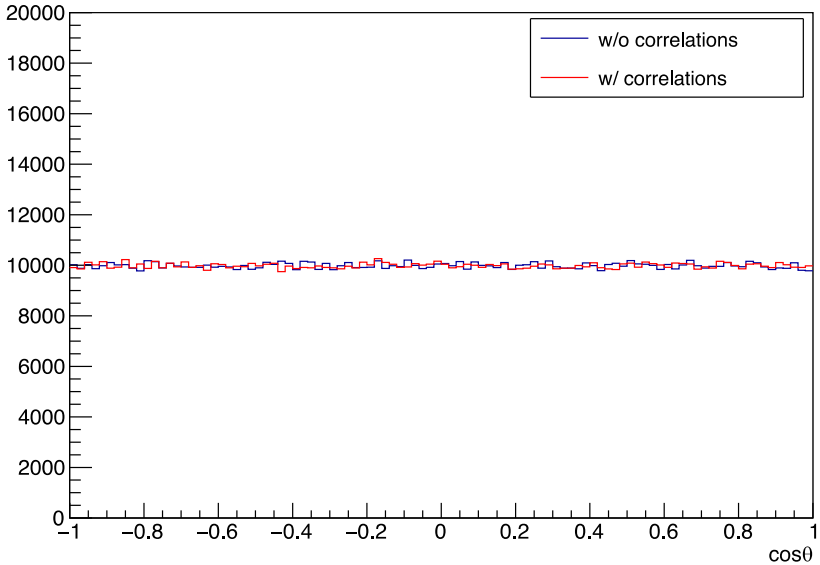
  - Find gammas with energies within ΔE of E1 and E2 and measure angle

Testing: high level



Problem in G4RadioactiveDecay

- A new G4ITDecay instantiated for each level
- Nuclear polarization is trapped in parent level's G4ITDecay when it is needed by daughter level.
- Need a solution compatible with multithreading

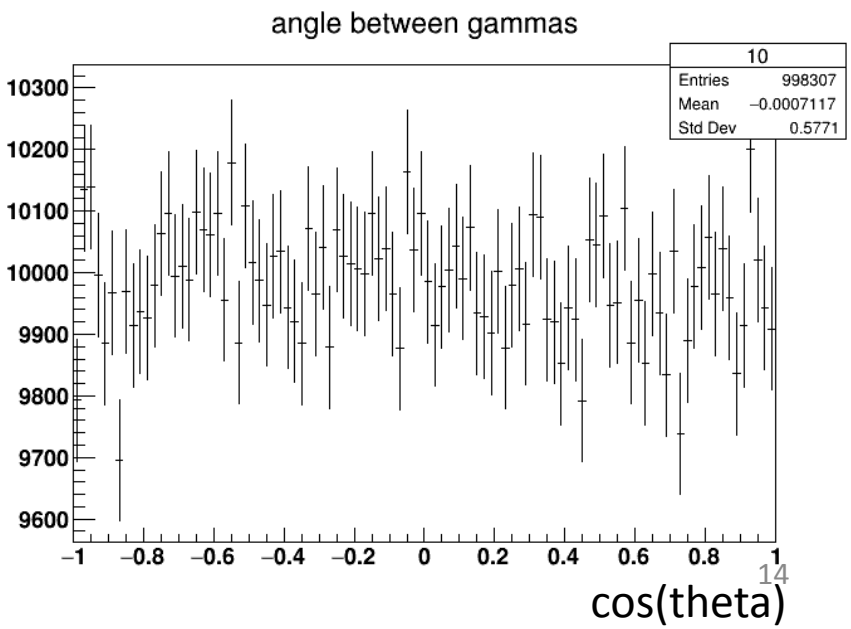


Radioactive decay examples

rdecay01: same geometry as Aobo's test code

Updated to include:

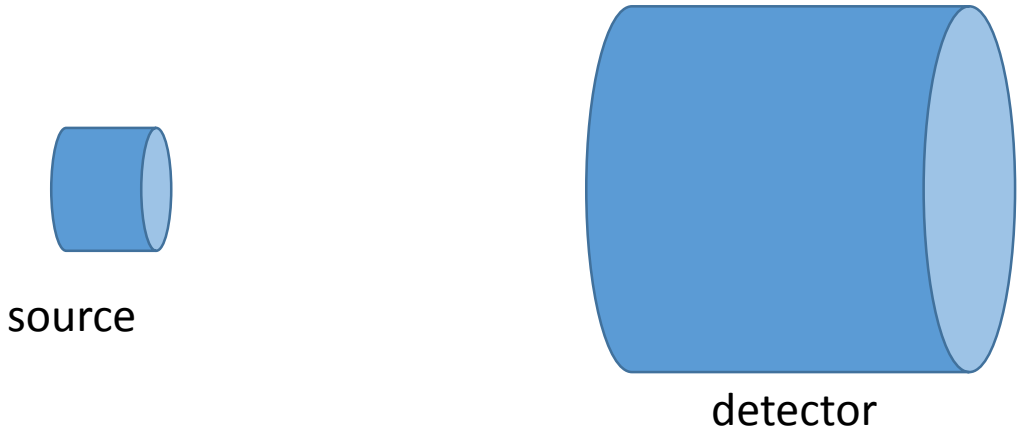
- /rdecay01/gammaAngularCorrelation to enable angular correlation
- Added histogram 10, the angle between two gammas, specified by:
- /rdecay01/gamma/AngularCorrelationEnergy E1 E2 Ewindow
- Updated Co60.mac to plot the angular correlation between 1.17 and 1.33 MeV gammas:



Radioactive decay examples

rdecay02:

- Simulate source and detector
- /rdecay01/gammaAngularCorrelation to enable angular correlation
- Added Co60 macro
  - Should see a difference in 2.5 MeV sum gamma line



Recommendations

- Macro command to turn angular correlations on and off
- Output info during initialization saying that it is turned on
- Macro command to set / adjust verbosity in the angular correlation code (Jason had to recompile G4 with G4PhotonEvaporation::fVerbosity initialized to 2)
- Do benchmarking with examples
  - 60-Co only has 2 gammas; try decay with a longer cascade
  - Consider turning on by default with performance degradation by <10% or so.
- Add changes to examples; update documentation

Backup
(Jason’s old slides)

G4NuclearPolarization

- Encapsulates / manages statistical tensor representing the entangled nuclear spin state throughout a gamma cascade

```
class G4NuclearPolarization
{
  private:
    G4int fTwoJ1;
    std::vector< std::vector<G4complex> > fPolarization;

  public:
    G4double GenerateGammaCosTheta(G4int twoJ2, G4int Lbar,
                                   G4double delta=0, G4int Lprime=1) const;

    G4double GenerateGammaPhi(G4double gammaCosTheta,
                              G4int twoJ2, G4int Lbar,
                              G4double delta=0, G4int Lprime=1) const;

    void UpdatePolarizationAfterGammaEmission(G4double gammaCosTheta, G4double gammaPhi,
                                              G4int twoJ2, G4int Lbar,
                                              G4double delta=0, G4int Lprime=1);
    ...
};
```

G4NuclearPolarization

- Encapsulates / manages statistical tensor representing the entangled nuclear spin state throughout a gamma cascade
- I ignore gamma polarization, although it could be easily added later if there is a need.
- Not specific / particular to gamma cascades! Could be used e.g. for beta decay as well (although I have not implemented it)

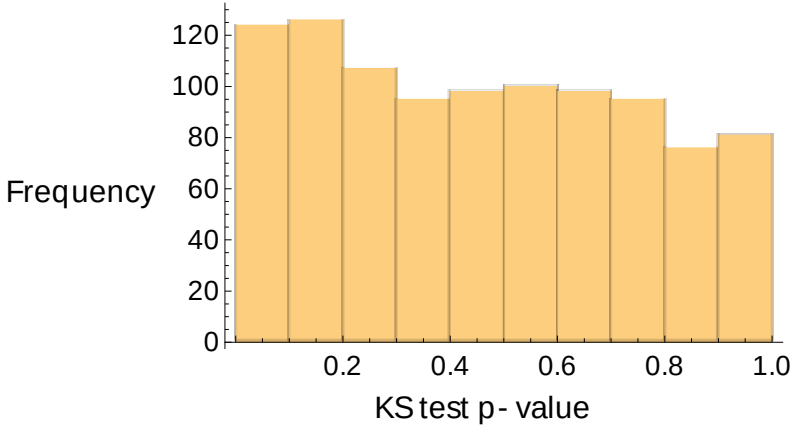
Nuclear Polarization

- Vladimir split my code into two pieces:
  - **G4NuclearPolarization**: the "statistical tensor" itself (data object)
  - **G4PolarizationTransition**: samples angles and modifies the G4NuclearPolarization to represent the final nuclear state after decay
- These classes were added to G4.10.02
  - Thank you Vladimir!

Nuclear Polarization Validation

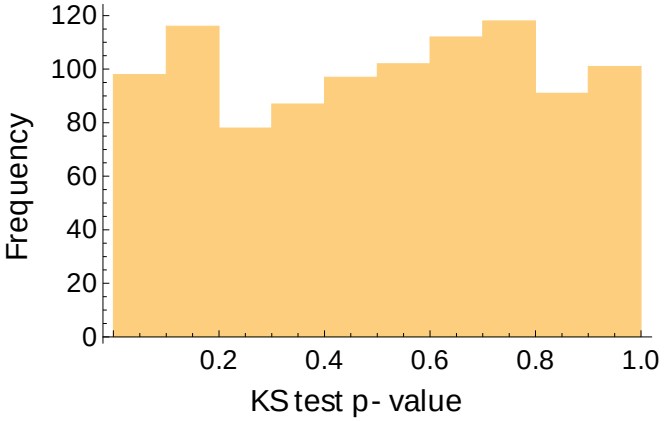
- In 2015 there was still some minor disagreement between my code and a brute-force Mathematica calculation
  - Identified as round-off error in Mathematica

G4NuclearPolarization vs. Mathematica



P(uniform) = 3.5×10^{-7}

G4NuclearPolarization vs. Theory



P(uniform) = 23%

G4NuclearPolarization Testing

- Rough test: verified that it generates correct polynomials for simple cases
- Deep low-level test: check that it generates correct PDFs for multi-level cascades with high J_1 / J_2 / L (highly non-trivial entangled polarization state)
- High-level test: generate ^{60}Co decay and “measure” the angular distribution produced by G4

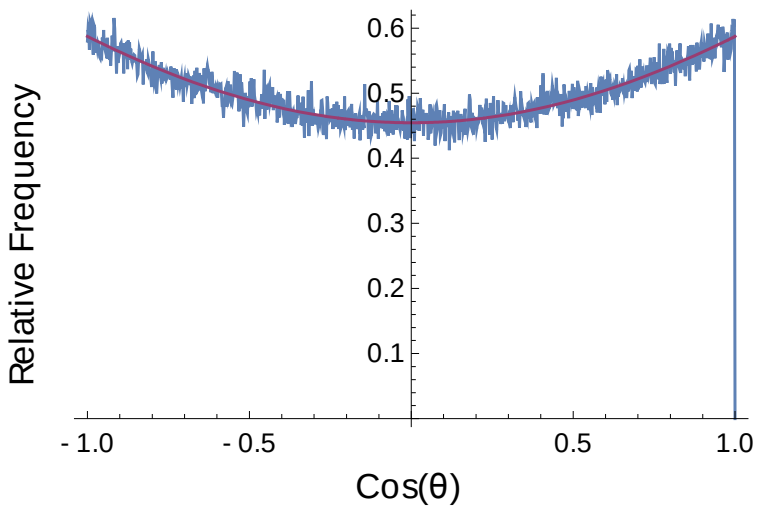
G4NuclearPolarization Low-Level Testing

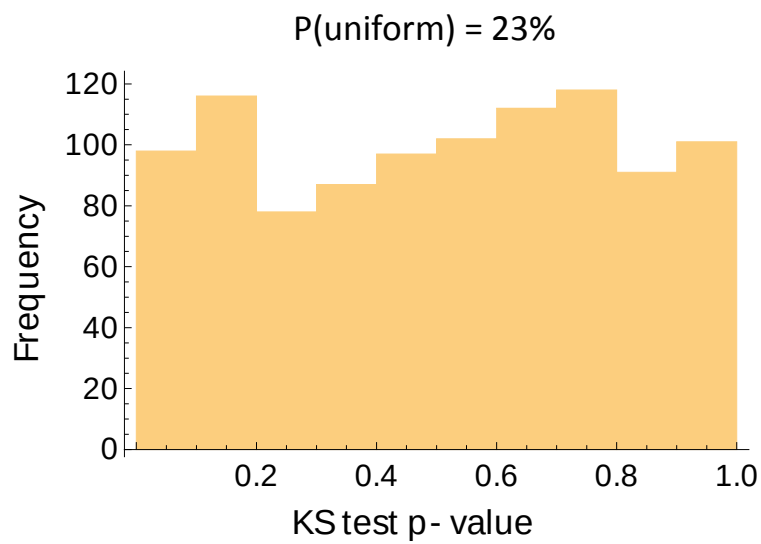
- Generate distributions for three successive gammas emitted by nucleus
  - First gamma establishes polar axis
  - Second gamma establishes azimuthal axis
  - Third gamma is generated from a highly nontrivial statistical tensor
- Distribution of angle between first and third gamma can be calculated by integrating out second gamma direction: get a simple closed form that can be compared to the simulated distribution.

G4NuclearPolarization Low-Level Testing

Use G4NuclearPolarization to generate 1000 gammas from a given transition 1000 times

- Compute 1000 KS statistics between G4 and the theoretical distributions





G4NuclearPolarization High-Level Testing

- Use G4RDM to generate nuclear decay at the center of a spherical vacuum.
- Tag gamma positions as they exit the sphere
- Histogram angular correlations and compare to theoretical expectations and/or measured distributions found in the literature

Utility Classes

- My students tested each extensively. All we found were bugs in Geant4, Mathematica, and gsl. ☺
  - Just kidding, there were 2 bugs in my code too, both associated with an incomplete refactoring of G4Clebsch.
- Vladimir kindly ported these into Geant4
  - Despite being generically useful, for now these are in processes/hadronics/models/util
  - Coverity reported some problems, but I don’t see these problems in the latest release?
  - Thank you Vladimir!
  - Will re-test with the as-integrated classes.

G4Clebsch Testing

- Generate 3 lists of parameters and solutions for each Wigner symbol (3J, 6J, 9J)
  - Exhaustively scan through all angular momenta up to J = 5 that satisfy selection rules
  - Generate random sets of parameters up to J = 10 that satisfy selection rules
  - Generate random sets of parameters up to J = 10 that do not need to satisfy selection rules
- Comparison to gsl:
  - Found bug in gsl: https://savannah.gnu.org/bugs/?29606
- Comparison to Mathematica:
  - For 9-j symbol, used angular.m notebook by Andrei Derevianko (University of Nevada, Reno)
  - Agreed to within a few parts in 10^{-12} for all values tests, and to within machine precision for all cases of physical relevance (lower J)

G4LegendrePolynomial Testing

1. Generate Associated Legendre Polynomials with *Mathematica* through *l*=30
   - Coefficients
   - Evaluations at random x-values
2. Compare to results from G4LegendrePolynomial
   - Coefficients and evaluations agree to within machine precision

G4PolynomialPDF Testing

1. Randomly generate 100 polynomials with *Mathematica* 10
   - Endpoints generated with Tan[RandomReal[{-Pi/2,Pi/2}]]
   - Random # of coefficients between 1 and 30 generated in same fashion
   - Only accept polynomials that are nonnegative between endpoints
     - Need well-defined PDFs
   - Normalize polynomials

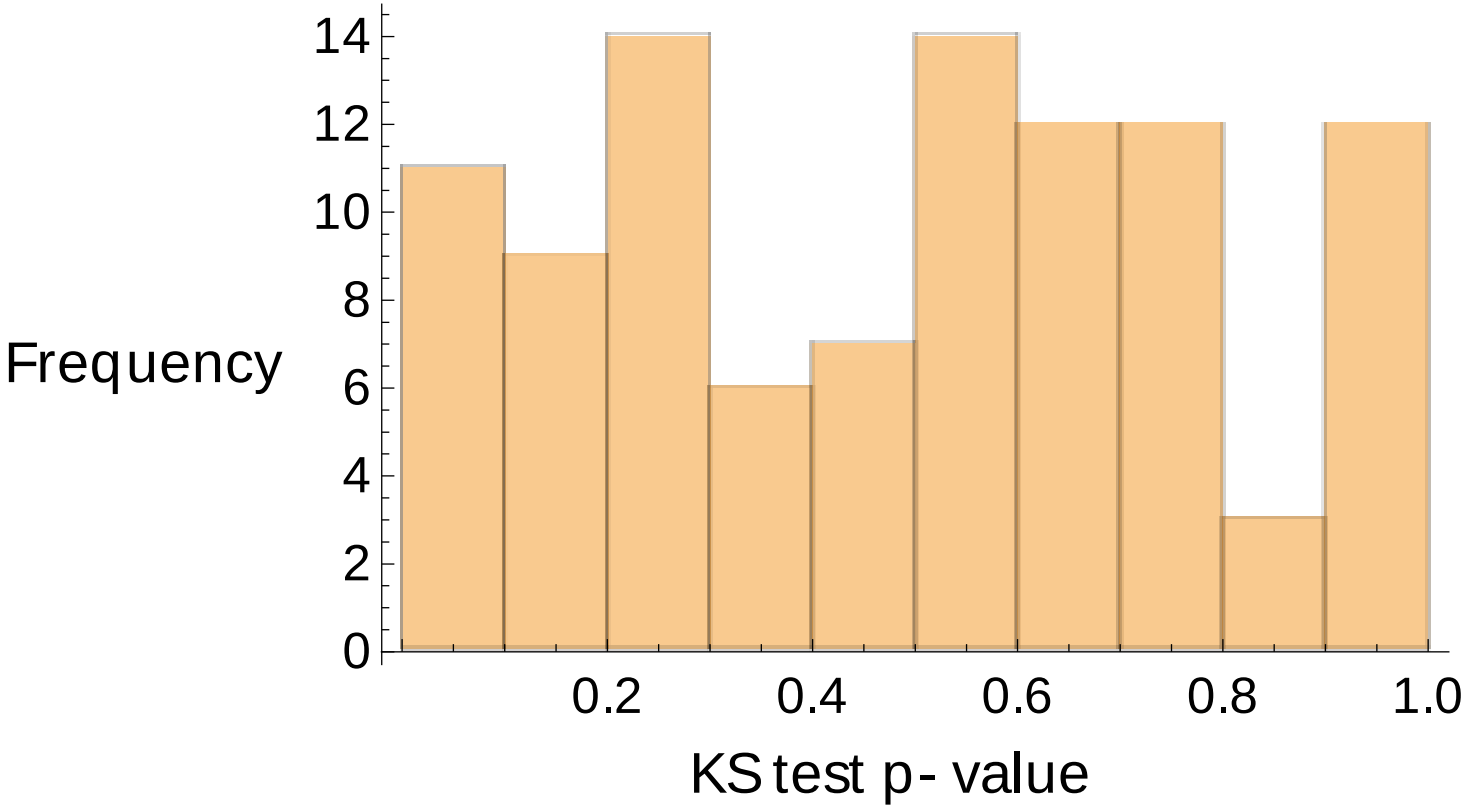
G4PolynomialPDF Testing

2. Generate 10000 random x-values for each polynomial and evaluate each polynomial at all 10000 points
3. Convert each polynomial to PDF and randomly sample 10000 times
4. Save polynomials, evaluations, random x-values, and random samples to disk
5. For each polynomial, compute max relative error of normalizing coefficients with G4PolynomialPDF.Normalize() compared to normalizing with *Mathematica*
   - We achieve agreement to machine precision

G4PolynomialPDF Testing

6. For each polynomial, evaluate at 10000 random x-values with G4PolynomialPDF.Evaluate() and find max error relative to *Mathematica*
   - We achieve agreement to within machine precision
7. Randomly sample each polynomial 10000 times with G4PolynomialPDF.RandomX()
8. For each PDF, compare random sampling to *Mathematica* with ROOT KolmogorovTest

G4PolynomialPDF Testing



KS test comparing above data to uniform distribution gives p-value of 0.73

Multipolarity Data

- Required Data
  - *J* of initial and final nuclear levels
  - Dominant multipolarity (*L*) of the transition
  - For some transitions, the sub-dominant multipolarity (*L*) and its relative strength (δ)
  - Nomenclature:

$$J_1^{\pi} \rightarrow J_2^{\pi'} + L$$

	L = 1	*L* = 2	*L* = 3	*L* = 4	*L* = …
π□= π	E1	M2	E3	M4	…
π□= π	M1	E2	M3	E4	…

- PhotonEvaporation3.2 contains all necessary *J* information, but *L* (and *L*, δ) was still missing.

Status: Multipolarity Data

- Once in PhotoEvaporation database, will need to be loaded into G4 classes
- Class: G4NucLevel
  - Holds information on possible gamma and CE transitions from a particular nuclear excited state
  - Needs access to *J* of initial level: add pointer to corresponding G4LevelManager?
  - Needs vectors holding *J*, *L*, *L*, $\bar{\delta}$ of each transition.
    - Fill during G4LevelReader::MakeLevelManager()
    - Can be left empty if users don't need angular correlations