Custom Physics Lists in larg4 and Updates

David Rivera

University of Pennsylvania

October 19, 2019

Table of Contents

Recap

- As mentioned in my presentation for the DUNE collaboration meeting, the StepLimit in the geometry was set too finely resulting in very large ouput files
- Other contributing factors:
  - Zero energy tracking cut
  - storing all MCParticle information for daughters from EM interactions
  - storing SimEnergyDeposits
  - zlib compression setting of 0 i.e. no compression [1]

[1] This was intentionally set to zero to favor faster output write speed.

Tests

- Used a corsika-generated event
- Used the same event for all tests
  - The number of primary particles for the various tests was the same: 686
  - the number of secondaries can vary, but for the most part, the number of hits was consistent between all tests (>800,000 hits)
- the Legacy standard is:
  - **KeepEMShowerDaughters:** false # minimal info will be stored for EM daughters
  - **EnergyCut:** 1e-5 # [GeV], below this kinetic energy, particles will not be tracked
  - **compressionLevel:** 1 # output file zlib compression level

Benchmarks

- Output file size (**Out Size**) in MB
- Peak virtual memory usage: (**Virtual**) in MB
- Peak resident memory usage: (**Resident**) in MB
- Time to write output: (**Write time**) in seconds

	Out Size (MB)	Virtual	Resident	Write time
Legacy	134 MB	3488.9 MB	2810.7 MB	7.35 s
Refactored	114 MB	2752.5 MB	2043.8 MB	5.75 s
% change	-14.9%	-21.1%	-27.3%	-21.8%

Conclusions

Refactored looks good so far in terms of output size, memory consumption and output write time, but with some **caveats:**

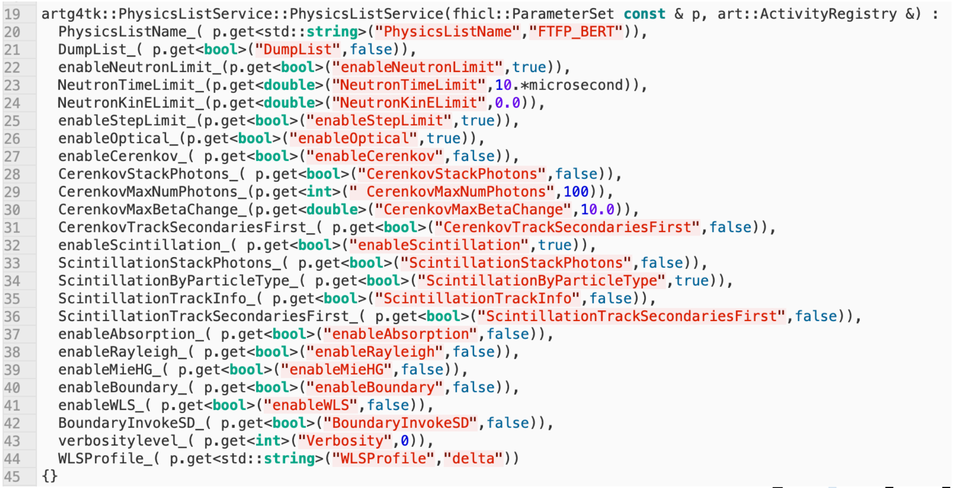
- the refactored larg4 is still missing some data products in my tests (photon and crt products)
- I used **QGSP_BERT_HP** in refactored

Will continue benchmark tests as things evolve to ensure that resource consumption remains reasonable

Table of Contents

Refactored Physics List Service Constructor

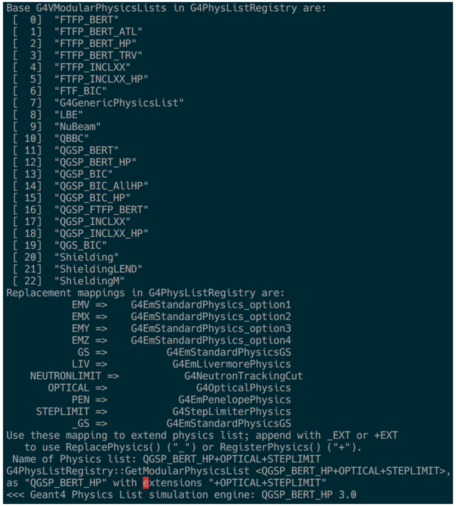
- the physics list along with other options that enable extensions (e.g. enableStepLimit) are selected in the g4 stage fhicl file



```
artg4tk::PhysicsListService::PhysicsListService(fhicl::ParameterSet const & p, art::ActivityRegistry &) :
  PhysicsListName_( p.get<std::string>("PhysicsListName","FTFP_BERT")),
  DumpList_( p.get<bool>("DumpList",false)),
  enableNeutronLimit_(p.get<bool>("enableNeutronLimit",true)),
  NeutronTimeLimit_(p.get<double>("NeutronTimeLimit",10.*microsecond)),
  NeutronKinELimit_(p.get<double>("NeutronKinELimit",0.0)),
  enableStepLimit_(p.get<bool>("enableStepLimit",true)),
  enableOptical_(p.get<bool>("enableOptical",true)),
  enableCerenkov_( p.get<bool>("enableCerenkov",false)),
  CerenkovStackPhotons_( p.get<bool>("CerenkovStackPhotons",false)),
  CerenkovMaxNumPhotons_(p.get<int>(" CerenkovMaxNumPhotons",100)),
  CerenkovMaxBetaChange_(p.get<double>("CerenkovMaxBetaChange",10.0)),
  CerenkovTrackSecondariesFirst_( p.get<bool>("CerenkovTrackSecondariesFirst",false)),
  enableScintillation_( p.get<bool>("enableScintillation",true)),
  ScintillationStackPhotons_( p.get<bool>("ScintillationStackPhotons",false)),
  ScintillationByParticleType_( p.get<bool>("ScintillationByParticleType",true)),
  ScintillationTrackInfo_( p.get<bool>("ScintillationTrackInfo",false)),
  ScintillationTrackSecondariesFirst_( p.get<bool>("ScintillationTrackSecondariesFirst",false)),
  enableAbsorption_( p.get<bool>("enableAbsorption",false)),
  enableRayleigh_( p.get<bool>("enableRayleigh",false)),
  enableMieHG_( p.get<bool>("enableMieHG",false)),
  enableBoundary_( p.get<bool>("enableBoundary",false)),
  enableWLS_( p.get<bool>("enableWLS",false)),
  BoundaryInvokeSD_( p.get<bool>("BoundaryInvokeSD",false)),
  verbositylevel_( p.get<int>("Verbosity",0)),
  WLSProfile_( p.get<std::string>("WLSProfile","delta"))
{}
```

Reference Physics Lists

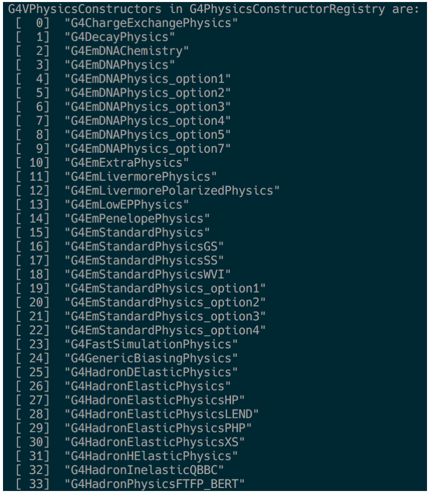
- larg4 takes advantage of the Extensible physics list factory class **G4PhysListFactoryAlt** written by R. Hatcher
- the extensible physics list factory retains a central registry of known types (e.g. reference lists) and allows it to be extended by registering new types with it.
- For example we can use one of the reference physics lists (QGSP_BERT) and extend/modify it with other available physics constructor factories
- The currently defined extensions include:
  - various EM options (EM_V, EM_X, ... etc)
  - Neutron tracking cut, the StepLimiter, and Optical physics



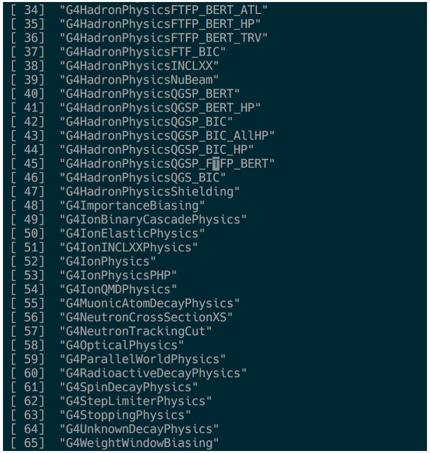
```
Base G4VModularPhysicsLists in G4PhysListRegistry are:
 [  0]  "FTFP_BERT"
 [  1]  "FTFP_BERT_ATL"
 [  2]  "FTFP_BERT_HP"
 [  3]  "FTFP_BERT_TRV"
 [  4]  "FTFP_INCLXX"
 [  5]  "FTFP_INCLXX_HP"
 [  6]  "FTF_BIC"
 [  7]  "G4GenericPhysicsList"
 [  8]  "LBE"
 [  9]  "NuBeam"
 [ 10]  "QBBC"
 [ 11]  "QGSP_BERT"
 [ 12]  "QGSP_BERT_HP"
 [ 13]  "QGSP_BIC"
 [ 14]  "QGSP_BIC_AllHP"
 [ 15]  "QGSP_BIC_HP"
 [ 16]  "QGSP_FTFP_BERT"
 [ 17]  "QGSP_INCLXX"
 [ 18]  "QGSP_INCLXX_HP"
 [ 19]  "QGS_BIC"
 [ 20]  "Shielding"
 [ 21]  "ShieldingLEND"
 [ 22]  "ShieldingM"
Replacement mappings in G4PhysListRegistry are:
           EMV =>    G4EmStandardPhysics_option1
           EMX =>    G4EmStandardPhysics_option2
           EMY =>    G4EmStandardPhysics_option3
           EMZ =>    G4EmStandardPhysics_option4
            GS =>          G4EmStandardPhysicsGS
           LIV =>           G4EmLivermorePhysics
    NEUTRONLIMIT =>           G4NeutronTrackingCut
       OPTICAL =>              G4OpticalPhysics
           PEN =>            G4EmPenelopePhysics
     STEPLIMIT =>           G4StepLimiterPhysics
           _GS =>          G4EmStandardPhysicsGS
Use these mapping to extend physics list; append with _EXT or +EXT
   to use ReplacePhysics() ("_") or RegisterPhysics() ("+").
 Name of Physics list: QGSP_BERT_HP+OPTICAL+STEPLIMIT
G4PhysListRegistry::GetModularPhysicsList <QGSP_BERT_HP+OPTICAL+STEPLIMIT>,
as "QGSP_BERT_HP" with extensions "+OPTICAL+STEPLIMIT"
<<< Geant4 Physics List simulation engine: QGSP_BERT_HP 3.0
```

Physics Constructors

- The physics constructor registry shows the various different “physics” that can be selected and used to replace the equivalent model in the base, reference list chosen



```
G4VPhysicsConstructors in G4PhysicsConstructorRegistry are:
 [  0]  "G4ChargeExchangePhysics"
 [  1]  "G4DecayPhysics"
 [  2]  "G4EmDNAChemistry"
 [  3]  "G4EmDNAPhysics"
 [  4]  "G4EmDNAPhysics_option1"
 [  5]  "G4EmDNAPhysics_option2"
 [  6]  "G4EmDNAPhysics_option3"
 [  7]  "G4EmDNAPhysics_option4"
 [  8]  "G4EmDNAPhysics_option5"
 [  9]  "G4EmDNAPhysics_option7"
 [ 10]  "G4EmExtraPhysics"
 [ 11]  "G4EmLivermorePhysics"
 [ 12]  "G4EmLivermorePolarizedPhysics"
 [ 13]  "G4EmLowEPPhysics"
 [ 14]  "G4EmPenelopePhysics"
 [ 15]  "G4EmStandardPhysics"
 [ 16]  "G4EmStandardPhysicsGS"
 [ 17]  "G4EmStandardPhysicsSS"
 [ 18]  "G4EmStandardPhysicsWVI"
 [ 19]  "G4EmStandardPhysics_option1"
 [ 20]  "G4EmStandardPhysics_option2"
 [ 21]  "G4EmStandardPhysics_option3"
 [ 22]  "G4EmStandardPhysics_option4"
 [ 23]  "G4FastSimulationPhysics"
 [ 24]  "G4GenericBiasingPhysics"
 [ 25]  "G4HadronDElasticPhysics"
 [ 26]  "G4HadronElasticPhysics"
 [ 27]  "G4HadronElasticPhysicsHP"
 [ 28]  "G4HadronElasticPhysicsLEND"
 [ 29]  "G4HadronElasticPhysicsPHP"
 [ 30]  "G4HadronElasticPhysicsXS"
 [ 31]  "G4HadronHElasticPhysics"
 [ 32]  "G4HadronInelasticQBBC"
 [ 33]  "G4HadronPhysicsFTFP_BERT"
```



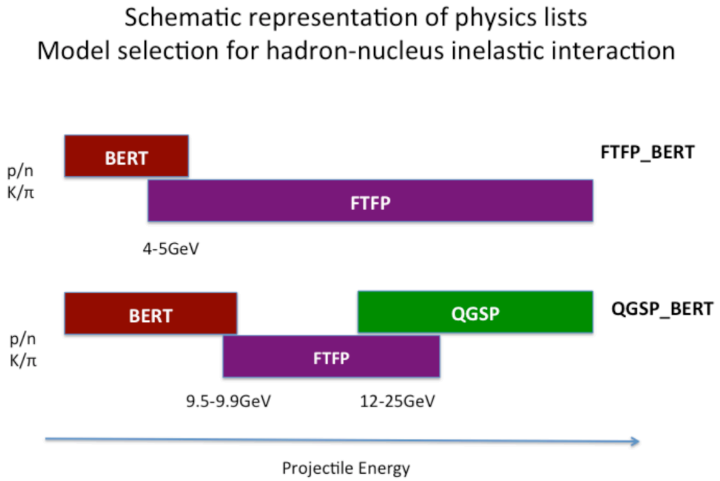
```
 [ 34]  "G4HadronPhysicsFTFP_BERT_ATL"
 [ 35]  "G4HadronPhysicsFTFP_BERT_HP"
 [ 36]  "G4HadronPhysicsFTFP_BERT_TRV"
 [ 37]  "G4HadronPhysicsFTF_BIC"
 [ 38]  "G4HadronPhysicsINCLXX"
 [ 39]  "G4HadronPhysicsNuBeam"
 [ 40]  "G4HadronPhysicsQGSP_BERT"
 [ 41]  "G4HadronPhysicsQGSP_BERT_HP"
 [ 42]  "G4HadronPhysicsQGSP_BIC"
 [ 43]  "G4HadronPhysicsQGSP_BIC_AllHP"
 [ 44]  "G4HadronPhysicsQGSP_BIC_HP"
 [ 45]  "G4HadronPhysicsQGSP_FTFP_BERT"
 [ 46]  "G4HadronPhysicsQGS_BIC"
 [ 47]  "G4HadronPhysicsShielding"
 [ 48]  "G4ImportanceBiasing"
 [ 49]  "G4IonBinaryCascadePhysics"
 [ 50]  "G4IonElasticPhysics"
 [ 51]  "G4IonINCLXXPhysics"
 [ 52]  "G4IonPhysics"
 [ 53]  "G4IonPhysicsPHP"
 [ 54]  "G4IonQMDPhysics"
 [ 55]  "G4MuonicAtomDecayPhysics"
 [ 56]  "G4NeutronCrossSectionXS"
 [ 57]  "G4NeutronTrackingCut"
 [ 58]  "G4OpticalPhysics"
 [ 59]  "G4ParallelWorldPhysics"
 [ 60]  "G4RadioactiveDecayPhysics"
 [ 61]  "G4SpinDecayPhysics"
 [ 62]  "G4StepLimiterPhysics"
 [ 63]  "G4StoppingPhysics"
 [ 64]  "G4UnknownDecayPhysics"
 [ 65]  "G4WeightWindowBiasing"
```

What you cannot do

1. substitute physics constructors containing models that don't cover the full particle-energy space spanned by the simulations
2. substitute physics constructors containing models that overlap in the valid particle-energy range spanned by the simulations
3. cannot redefine the validity range for two physics models at run time

For example: If I choose the reference physics list **QGSP_BERT_HP** which applies the Neutron High Precision models for elastic, inelastic, capture and fission processes from 0-20 MeV
and I would like to register a new physics constructor, **G4NeutronHPThermalScattering** which is valid between 0 and 4 eV in order to extend QGSP_BERT_HP, this would conflict with the Neutron HP model between 0 and 4eV

Physics Model


Schematic representation of physics lists
Model selection for hadron-nucleus inelastic interaction
p/n
K/π
BERT
FTFP
FTFP_BERT
4-5GeV
p/n
K/π
BERT
QGSP
FTFP
QGSP_BERT
9.5-9.9GeV
12-25GeV
Projectile Energy

Solution

However,

- It is fairly typical for physics lists to have two models that overlap over a small range of particle-energy space
- However the two models are invoked with a probability P in this inteface region such that P(Model 1) + P(Model 2) = 1
- E.g. :
  - The probability of invoking Model 1 goes from $1 \rightarrow 0$ linearly in the overlap region
  - The probability of invoking Model 2 is then 1-P(Model 1) in the overlap region

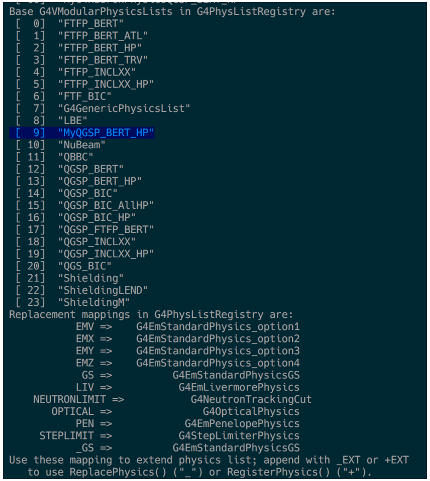
For the example in the previous slide, one would need to create a physics list which properly handles the interface between the Neutron Thermal Scattering model and the Neutron High Precision model to ensure unitarity

- It would not be sufficient to introduce the Neutron Thermal Scattering model as a simple extension.
- This is one motivator for having the ability to define a custom physics list in larg4.
- Other motivators include detailed cross-section studies (see later slides)

My Custom Physics List

As a proof of concept:

- Copied the reference physics list **QGSP_BERT_HP** headers and template class implementation
- Created a directory for them within larg4 (larg4/lists) and "re-branded" them as **MyQGSP_BERT_HP**
- Having access to my custom physics list required:
  1. source code to register it with the physics list factory registry
  2. compiling it into a shared object in larg4
  3. linking the physicsList_service in artg4tk to this library



```
Base G4VModularPhysicsLists in G4PhysListRegistry are:
 [  0]  "FTFP_BERT"
 [  1]  "FTFP_BERT_ATL"
 [  2]  "FTFP_BERT_HP"
 [  3]  "FTFP_BERT_TRV"
 [  4]  "FTFP_INCLXX"
 [  5]  "FTFP_INCLXX_HP"
 [  6]  "FTF_BIC"
 [  7]  "G4GenericPhysicsList"
 [  8]  "LBE"
 [  9]  "MyQGSP_BERT_HP"
 [ 10]  "NuBeam"
 [ 11]  "QBBC"
 [ 12]  "QGSP_BERT"
 [ 13]  "QGSP_BERT_HP"
 [ 14]  "QGSP_BIC"
 [ 15]  "QGSP_BIC_AllHP"
 [ 16]  "QGSP_BIC_HP"
 [ 17]  "QGSP_FTFP_BERT"
 [ 18]  "QGSP_INCLXX"
 [ 19]  "QGSP_INCLXX_HP"
 [ 20]  "QGS_BIC"
 [ 21]  "Shielding"
 [ 22]  "ShieldingLEND"
 [ 23]  "ShieldingM"
Replacement mappings in G4PhysListRegistry are:
           EMV =>    G4EmStandardPhysics_option1
           EMX =>    G4EmStandardPhysics_option2
           EMY =>    G4EmStandardPhysics_option3
           EMZ =>    G4EmStandardPhysics_option4
            GS =>          G4EmStandardPhysicsGS
           LIV =>           G4EmLivermorePhysics
    NEUTRONLIMIT =>           G4NeutronTrackingCut
       OPTICAL =>             G4OpticalPhysics
           PEN =>            G4EmPenelopePhysics
     STEPLIMIT =>           G4StepLimiterPhysics
           _GS =>          G4EmStandardPhysicsGS
Use these mapping to extend physics list; append with _EXT or +EXT
   to use ReplacePhysics() ("_") or RegisterPhysics() ("+").
```

MyQGSP_BERT_HP (highlighted in blue) available as an option within larg4.

Table of Contents

Proposal

- A. Higuera has proposed a study on pion quasi-elastic scattering cross sections that would require distinguising between outgoing particles from the QE vertex and outgoing particles resulting from the intranuclear cascade process at the ∼1GeV range
- One would ideally like to "turn off" the cascade process; however, it is not sufficient to push the energy range of validity for the cascade model as mentioned in item 1 of slide 11
- Would have to define an alternative model to apply to the hadrons from 0 to 1GeV
- Alternatively one can perhaps change the behavior of the Cascade model itself OR
- it may suffice to extract information about the interactions themselves

Since it's far easier to extract the information from the Cascade model, I have started with that

Bertini Cascade Model

- See this document for more details and for the figure shown on the right

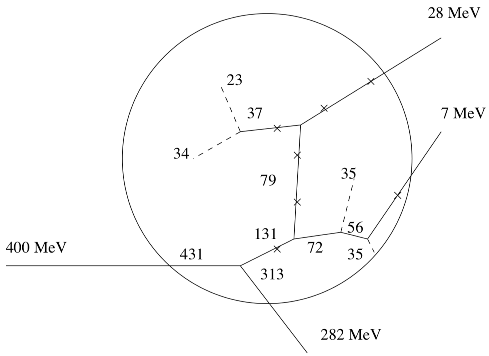
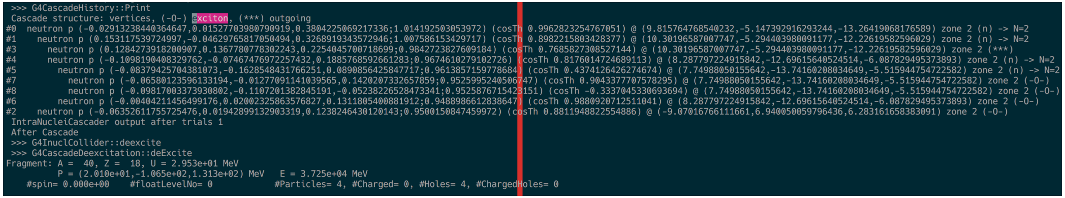


Figure 1: Schematic presentation of the intra-nuclear cascade. A hadron with 400 MeV energy is forming an INC history. Crosses present the Pauli exclusion principle in action. (The picture is a reproduction from original work of Bertini [4].)

First look

- In order to have full control over the cascade model I have copied and rebranded the Inelastic Physics constructor under the **QGSP_BERT_HP** physics list (namely G4HadronPhysicsQGSP_BERT_HP) and all associated headers and source code to the same area where I have my custom physics list to be declared and registered as a physics constructor
- The Pion builder class also had to be copied
- The Bertini Cascade model is itself implemented in the G4CascadeInterface which I have copied and rebranded as well

After many failed attempts:



```
 >>> G4CascadeHistory::Print
 Cascade structure: vertices, (-O-) exciton, (***) outgoing
#0  neutron p (-0.02913238440364647,0.01527703980790919,0.3804225069217336;1.014192503053972) (cosTh 0.9962823254767051) @ (9.815764768540232,-5.147392916293244,-13.26419068176589) zone 2 (n) -> N=2
#1    neutron p (0.153117539724997,-0.04629765817050494,0.3268919343572946;1.007586153429717) (cosTh 0.8982215803428377) @ (10.30196587007747,-5.294403980091177,-12.22619582596029) zone 2 (n) -> N=2
#3      neutron p (0.1284273918200907,0.1367780778302243,0.2254045700718699;0.9842723827609184) (cosTh 0.7685827308527144) @ (10.30196587007747,-5.294403980091177,-12.22619582596029) zone 2 (***)
#4      neutron p (-0.1098190408329762,-0.07467476972257432,0.1885768592661283;0.9674610279102726) (cosTh 0.8176014724689113) @ (8.287797224915842,-12.69615640524514,-6.087829495373893) zone 2 (n) -> N=2
#5        neutron p (-0.08379425704381073,-0.1628548431766251,0.0890856425847717;0.9613857159778684) (cosTh 0.4374126426274674) @ (7.74988050155642,-13.74160208034649,-5.515944754722582) zone 2 (n) -> N=2
#7          neutron p (-0.06580123596133194,-0.01277091141039565,0.1420207332657859;0.9525995240506747) (cosTh 0.9043377707578295) @ (7.74988050155642,-13.74160208034649,-5.515944754722582) zone 2 (-O-)
#8          neutron p (-0.09817003373930802,-0.1107201382845191,-0.05238226528473341;0.9525876715423151) (cosTh -0.3337045330693694) @ (7.74988050155642,-13.74160208034649,-5.515944754722582) zone 2 (-O-)
#6        neutron p (-0.004042114564991 76,0.02002325863576827,0.1311805400881912;0.9488986612838647) (cosTh 0.9880920712511041) @ (8.287797224915842,-12.69615640524514,-6.087829495373893) zone 2 (-O-)
#2    neutron p (-0.06352611755725476,0.01942899132903319,0.1238246430120143;0.9500150847459972) (cosTh 0.8811948822554886) @ (-9.07016766111661,6.940050059796436,6.283161658383091) zone 2 (-O-)
 IntraNucleiCascader output after trials 1
 After Cascade
 >>> G4InuclCollider::deexcite
 >>> G4CascadeDeexcitation::deExcite
Fragment: A =  40, Z =  18, U = 2.953e+01 MeV
         P = (2.010e+01,-1.065e+02,1.313e+02) MeV   E = 3.725e+04 MeV
    #spin= 0.000e+00    #floatLevelNo= 0            #Particles= 4, #Charged= 0, #Holes= 4, #ChargedHoles= 0
```

TO DOs

- Document the custom physics list
- Agree on a place to store the physics list example
- Quick study using the CascadeInterface history and other information

Table of Contents

Refactored LArG4

Standard – **larsim/LArG4** AKA **Legacy**

- depends on nug4
- ConfigurablePhysicsList.h
- Optical simulation in Legacy was taken out of Geant and adapted from the Peter Gumplinger's original G4 implementations
  - **TheScintillationProcess** $\rightarrow$ SetScintillationYield()
  - there can be only one scintillating material in the optical simulation (LAr)

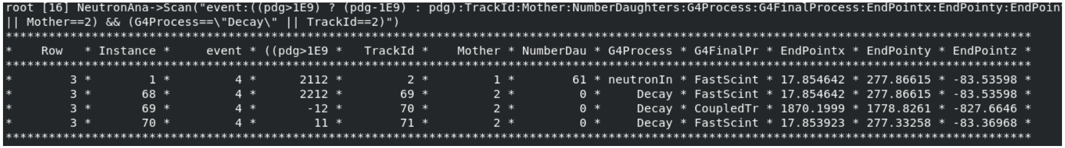
Refactored – **LArG4**

- depends on artg4tk (artg4 tool kit)
- Access to reference physics lists + extensions
- Updated OpticalPhysics in G4
  - scintillation properties are attached to the materials
  - can have any number of scintillating materials in the detector (e.g. LAr and plastic scintillator)

See Hans Wenzel's presentation from the DUNE collaboration meeting for a more comprehensive list of features and improvements of the refactored larg4 over Legacy: slides

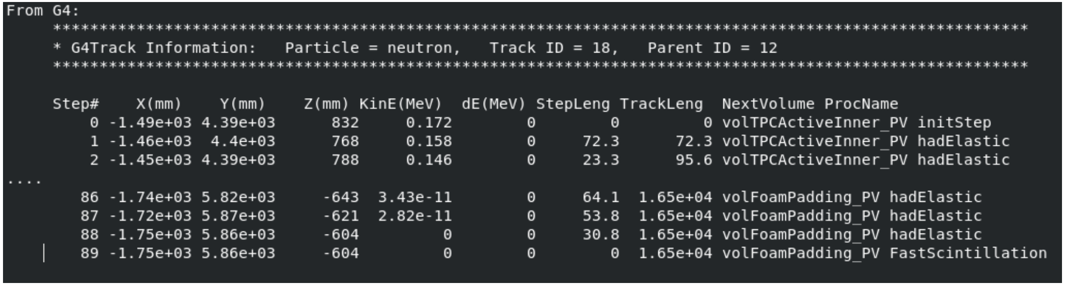
Neutron Study in Legacy

- Produced various samples of 10 MeV neutrons at the center of TPC1 (larsoft numbering, APA3-active)
- **Issue 1:** simb::MCParticle->EndProcess() for secondary neutrons often returns *FastScintillation*
- **Issue 2:** Some neutrons ending with FastScintillation processes come to rest in the ProtoDUNEFoam
- **Issue 3:** At rest neutrons subsequently decay... (n$\rightarrow$ p + e$^-$ + $\overline{\nu}_e$)
    - Neutron EndProcess is still marked as FastScintillation
    - simb::MCParticle$\rightarrow$Process() for proton, e$^-$, and $\overline{\nu}_e$ returns *Decay*



```
root [16] NeutronAna->Scan("event:((pdg>1E9) ? (pdg-1E9) : pdg):TrackId:Mother:NumberDaughters:G4Process:G4FinalProcess:EndPointx:EndPointy:EndPoin
|| Mother==2) && (G4Process==\"Decay\" || TrackId==2)")
*************************************************************************************************************************************************
*    Row   * Instance *     event * ((pdg>1E9 *   TrackId *    Mother * NumberDau * G4Process * G4FinalPr * EndPointx * EndPointy * EndPointz *
*************************************************************************************************************************************************
*        3 *        1 *         4 *      2112 *         2 *         1 *        61 * neutronIn * FastScint * 17.854642 * 277.86615 * -83.53598 *
*        3 *       68 *         4 *      2212 *        69 *         2 *         0 *     Decay * FastScint * 17.854642 * 277.86615 * -83.53598 *
*        3 *       69 *         4 *       -12 *        70 *         2 *         0 *     Decay * CoupledTr * 1870.1999 * 1778.8261 * -827.6646 *
*        3 *       70 *         4 *        11 *        71 *         2 *         0 *     Decay * FastScint * 17.853923 * 277.33258 * -83.36968 *
*************************************************************************************************************************************************
```

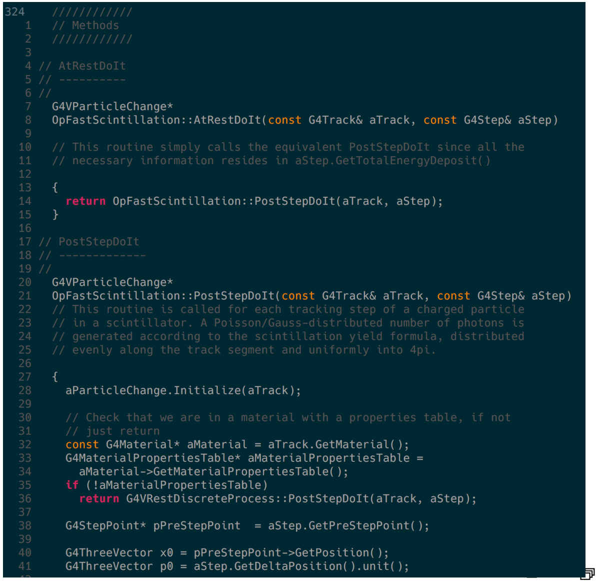
G4 Printout



```
From G4:
    *********************************************************************************************
    * G4Track Information:   Particle = neutron,   Track ID = 18,   Parent ID = 12
    *********************************************************************************************

    Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
        0 -1.49e+03 4.39e+03      832     0.172        0        0         0 volTPCActiveInner_PV initStep
        1 -1.46e+03  4.4e+03      768     0.158        0     72.3      72.3 volTPCActiveInner_PV hadElastic
        2 -1.45e+03 4.39e+03      788     0.146        0     23.3      95.6 volTPCActiveInner_PV hadElastic
....
       86 -1.74e+03 5.82e+03     -643  3.43e-11        0     64.1  1.65e+04 volFoamPadding_PV hadElastic
       87 -1.72e+03 5.87e+03     -621  2.82e-11        0     53.8  1.65e+04 volFoamPadding_PV hadElastic
       88 -1.75e+03 5.86e+03     -604         0        0     30.8  1.65e+04 volFoamPadding_PV hadElastic
   |   89 -1.75e+03 5.86e+03     -604         0        0        0  1.65e+04 volFoamPadding_PV FastScintillation
```

OpFastScintillation



```
  ////////////
  // Methods
  ////////////

// AtRestDoIt
// ----------
//
  G4VParticleChange*
  OpFastScintillation::AtRestDoIt(const G4Track& aTrack, const G4Step& aStep)

  // This routine simply calls the equivalent PostStepDoIt since all the
  // necessary information resides in aStep.GetTotalEnergyDeposit()

  {
    return OpFastScintillation::PostStepDoIt(aTrack, aStep);
  }

// PostStepDoIt
// -------------
//
  G4VParticleChange*
  OpFastScintillation::PostStepDoIt(const G4Track& aTrack, const G4Step& aStep)
  // This routine is called for each tracking step of a charged particle
  // in a scintillator. A Poisson/Gauss-distributed number of photons is
  // generated according to the scintillation yield formula, distributed
  // evenly along the track segment and uniformly into 4pi.

  {
    aParticleChange.Initialize(aTrack);

    // Check that we are in a material with a properties table, if not
    // just return
    const G4Material* aMaterial = aTrack.GetMaterial();
    G4MaterialPropertiesTable* aMaterialPropertiesTable =
      aMaterial->GetMaterialPropertiesTable();
    if (!aMaterialPropertiesTable)
      return G4VRestDiscreteProcess::PostStepDoIt(aTrack, aStep);

    G4StepPoint* pPreStepPoint  = aStep.GetPreStepPoint();

    G4ThreeVector x0 = pPreStepPoint->GetPosition();
    G4ThreeVector p0 = aStep.GetDeltaPosition().unit();
```