

Geant4 v11.4.2 Bugs & Issues

Jonathan Rositas

Overview of Bugs & Issues

- Incorrect Inverse Muon (& Tau) Decay Energy Threshold
- Neutrino Name Typo
- Muon Anti-Neutrino Charged Current Electron Scattering
- Energy Imbalances
- Momentum Imbalances
- Charge Imbalances

Charge Violation

```
Process: muNuNucleusCC , Model: NuMuNuclCcModel
Primary: nu_mu (14), E= 300000, target nucleus (8,16)
pass relative, limit 0.001, values E/T(0) = -7.98803e-05 p/p(0)= 0.000594612
fail absolute, limit (MeV) 10, values E / p (MeV) = -23.9641 / 178.384 3mom: (-35.4705,156.393,78.1263)
fail charge/baryon number balance 1 / 0
```

Process: muNuNucleusCC | Type: CC | Model: CCQE

Event ID: 14

```
Incident nu_mu Momentum: (-43.1786,-38.9871,294.305) GeV/c
-> Daughter: mu- | P: (-32.538,-35.4538,250.599) GeV/c
-> Daughter: neutron | P: (-10.7863,-3.57409,44.3021) GeV/c
-> Daughter: 015 | P: (0.175331,-0.116589,-0.677257) GeV/c
-> Daughter: gamma | P: (0.00583188,0.000920343,0.00376906) GeV/c
```

```
-----
Vector Sum Out:      (-43.1431,-39.1435,294.227) GeV/c
Missing Vector:      (-0.0354705,0.156393,0.0781263) GeV/c
Magnitude Missing: 0.178384 GeV/c
```

$$\nu_{\mu} + {}^{16}\text{O} \rightarrow {}^{15}\text{O} + \mu^{-} + n^0 + \gamma$$
$$\text{Charge: } 0 + 8 \rightarrow +8 - 1 + 0 + 0$$
$$8 \rightarrow 7$$

Characteristic terminal output for a 300 GeV muon neutrino CC nuclear scattering interaction. The top printout is G4 native with increased reporting enabled. The bottom print out is from my [SteppingAction.cc](#), giving more information, which is validated by the momentum value aligning with that reported above. Notice that there is a reported energy and momentum imbalance of -24 MeV and 178 MeV respectively. More importantly, note the self-reported charge imbalance, elaborated on in the provided interaction equation. This shows that G4 failed to generate a proton as appropriate and instead generated a neutron.

Q/E/P Violation

Process: eLNuNucleusCC | Type: CC | Model: CCQE

===== Q/E/P Conservation Check =====

Event ID: 31

[INITIAL STATE]

Incident nu_e | Q: 0 | E: 300 GeV | P: (-191.216,197.107,120.77) GeV/c

Target Z:13 A:27 | Q: 13 | E (mass): 25.1265 GeV

TOTAL IN | Q: 13 | E: 325.127 GeV | P: (-191.216,197.107,120.77) GeV/c

[FINAL STATE]

-> e- | Q: -1 | E: 299.763 GeV | P: (-191.216,197.092,120.206) GeV/c

-> proton | Q: 1 | E: 1.17613 GeV | P: (0.00427503,0.0202142,0.708872) GeV/c

-> Mg26 | Q: 12 | E: 24.199 GeV | P: (-0.331978,-0.0934838,-0.0623138) GeV/c

-> gamma | Q: 0 | E: 0.00302158 GeV | P: (-0.00160648,-5.98557e-05,0.00255844) GeV/c

-> gamma | Q: 0 | E: 0.00129153 GeV | P: (-0.00129011,5.09387e-05,3.27752e-05) GeV/c

-> gamma | Q: 0 | E: 0.00182881 GeV | P: (-0.00103874,-0.00150428,-5.20414e-05) GeV/c

TOTAL OUT | Q: 12 | E: 325.145 GeV | P: (-191.547,197.017,120.855) GeV/c

[Conservation Deltas]

Q Delta : 1 e

E Delta: -0.0182418 GeV (-18.2418 MeV)

P Vector Delta: (0.331725,0.0899472,-0.0849816) GeV/c

|P| Delta : 0.354053 GeV/c

=====

Process: eLNuNucleusCC | Type: CC | Model: CCRES

===== Q/E/P Conservation Check =====

Event ID: 46

[INITIAL STATE]

Incident nu_e | Q: 0 | E: 300 GeV | P: (67.4956,193.752,-218.871) GeV/c

Target Z:6 A:12 | Q: 6 | E (mass): 11.1749 GeV

TOTAL IN | Q: 6 | E: 311.175 GeV | P: (67.4956,193.752,-218.871) GeV/c

[FINAL STATE]

-> pi+ | Q: 1 | E: 3.79764 GeV | P: (-0.727323,-2.53769,2.72648) GeV/c

-> C12 | Q: 6 | E: 62.0074 GeV | P: (11.6871,40.7821,-43.8208) GeV/c

-> gamma | Q: 0 | E: 0.0520116 GeV | P: (0.0119666,0.0368169,-0.0347351) GeV/c

-> e- | Q: -1 | E: 241.526 GeV | P: (56.2769,154.609,-176.816) GeV/c

TOTAL OUT | Q: 6 | E: 307.383 GeV | P: (67.2486,192.89,-217.945) GeV/c

[Conservation Deltas]

Q Delta : 0 e

E Delta: 3.79159 GeV (3791.59 MeV)

P Vector Delta: (0.246985,0.861751,-0.925858) GeV/c

|P| Delta : 1.28873 GeV/c

=====

Some characteristic output for biased interaction models without any specific subclassing in place to handle the charge, energy, or momentum imbalances that occur. From my testing, it seems charge imbalance only occurs in QE and DIS scattering, while large energy imbalances (> 0.5 GeV) only occur in RES scattering.

NB: I have only observed the large Energy Violations with my own `G4VBiasingOperator` scheme. I have not seen it, namely, in `GDMLTargetPractice`. All of those energy violations seem to be within the 2% and <1 GeV thresholds.

Incorrect Inverse Muon (& Tau) Decay Formula

- IMD has a threshold energy defined by

$$E_{thresh} \geq \frac{m_{\mu}^2 - m_e^2}{2m_e} = (m_{\mu} - m_e)(m_{\mu} + m_e)/(2m_e)$$

- This formula is missing a factor of 2 in the implementation in `G4NeutrinoElectronCcXsc::IsElementApplicable()`, which causes IMD not to happen until ~22 GeV rather than ~11 GeV as is proper. This has been seen in simulation.

```
74  G4bool
75  G4NeutrinoElectronCcXsc::IsElementApplicable( const G4DynamicParticle* aPart, G4int, const G4Material*)
76  {
77      G4bool result = false;
78      G4String pName = aPart->GetDefinition()->GetParticleName();
79      G4double minEnergy = 0., energy = aPart->GetTotalEnergy();
80      G4double fmass, emass = electron_mass_c2;
81
82      if( pName == "anti_nu_e" || pName == "nu_mu" || pName == "anti_nu_mu" ) fmass = theMuonMinus->GetPDGMass();
83      else if( pName == "nu_tau" || pName == "anti_nu_tau" ) fmass = theTauMinus->GetPDGMass();
84      else fmass = emass;
85
86      minEnergy = (fmass-emass)*(fmass+emass)/emass;
87
88      if( ( pName == "nu_mu" || pName == "anti_nu_mu" ||
89          pName == "nu_tau" || pName == "anti_nu_tau" ) &&
90          energy > minEnergy )
91      {
92          result = true;
93      }
94      return result;
95  }
```

`G4NeutrinoElectronCcXsc.cc` excerpt. See line 86 for the typo.

Neutrino Name Typo

- In `G4NeutrinoElectronNcModel::IsApplicable()` there is a typo in the name of the `"anti_nu_mu"` particle that instead names it as `"anti_nu_nu"`, meaning that it is impossible for a muon anti neutrino to undergo neutral current electron scattering.

- **Nope!**
- **This bug actually does nothing!**
- **This *not breaking things* is actually a much larger problem...**

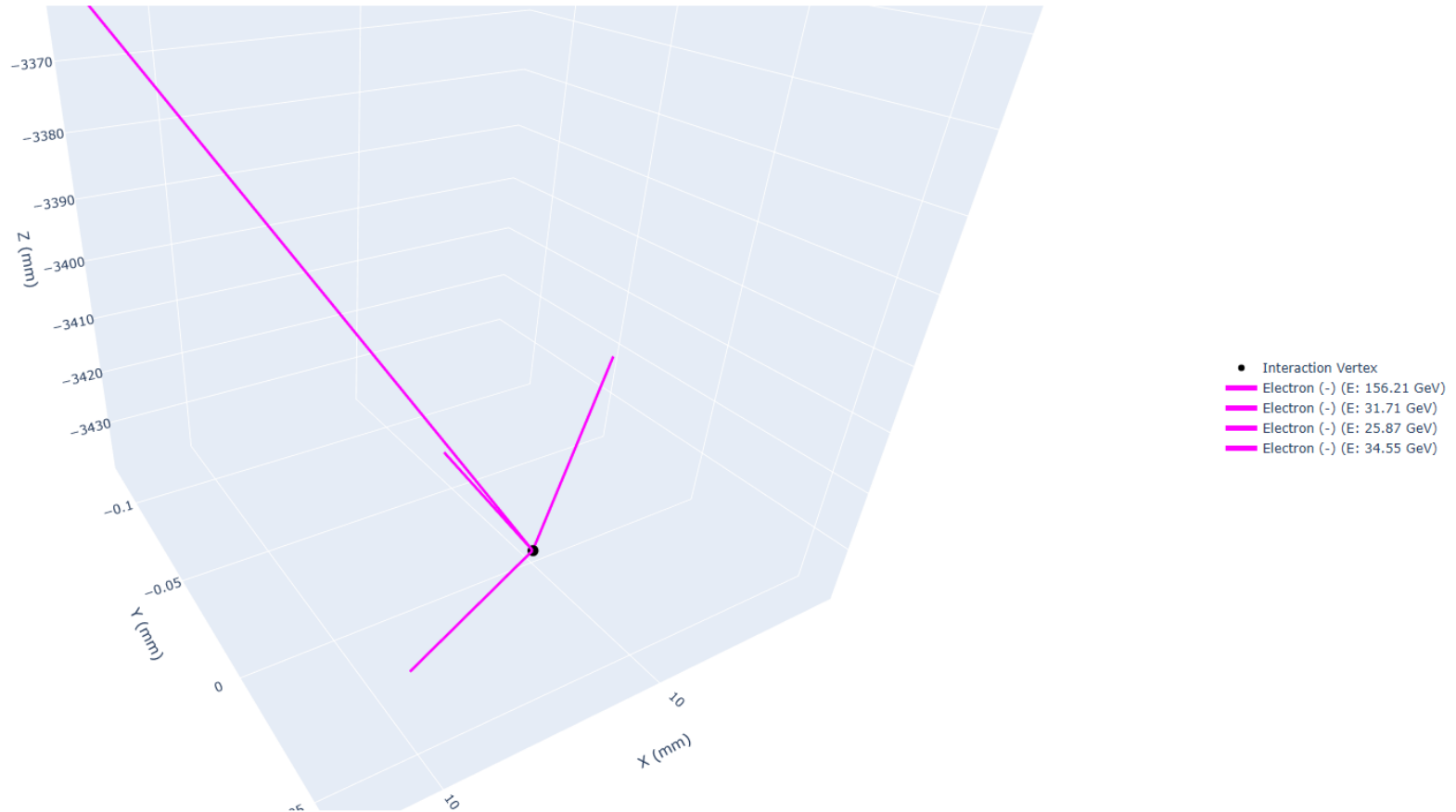
```
75  G4bool G4NeutrinoElectronNcModel::IsApplicable(const G4HadProjectile & aTrack, G4Nucleus&)
76  {
77      G4bool result = false;
78      G4String pName = aTrack.GetDefinition()->GetParticleName();
79      G4double minEnergy = 0.;
80      G4double energy = aTrack.GetTotalEnergy();
81
82      if( fCutEnergy > 0. ) // min detected recoil electron energy
83      {
84          minEnergy = 0.5*(fCutEnergy+sqrt(fCutEnergy*(fCutEnergy+2.*electron_mass_c2)));
85      }
86      if( ( pName == "nu_e"    || pName == "anti_nu_e"    ||
87          pName == "nu_mu"   || pName == "anti_nu_nu"   ||
88          pName == "nu_tau"  || pName == "anti_nu_tau"  ) &&
89          energy > minEnergy )
90      {
91          result = true;
92      }
93      return result;
94  }
```

`G4NeutrinoElectronNcModel.cc` excerpt. See line 87 for the typo.

$$\overline{\nu}_{\mu} + e^{-} \rightarrow e^{-} + e^{-} + e^{-} + e^{-}$$

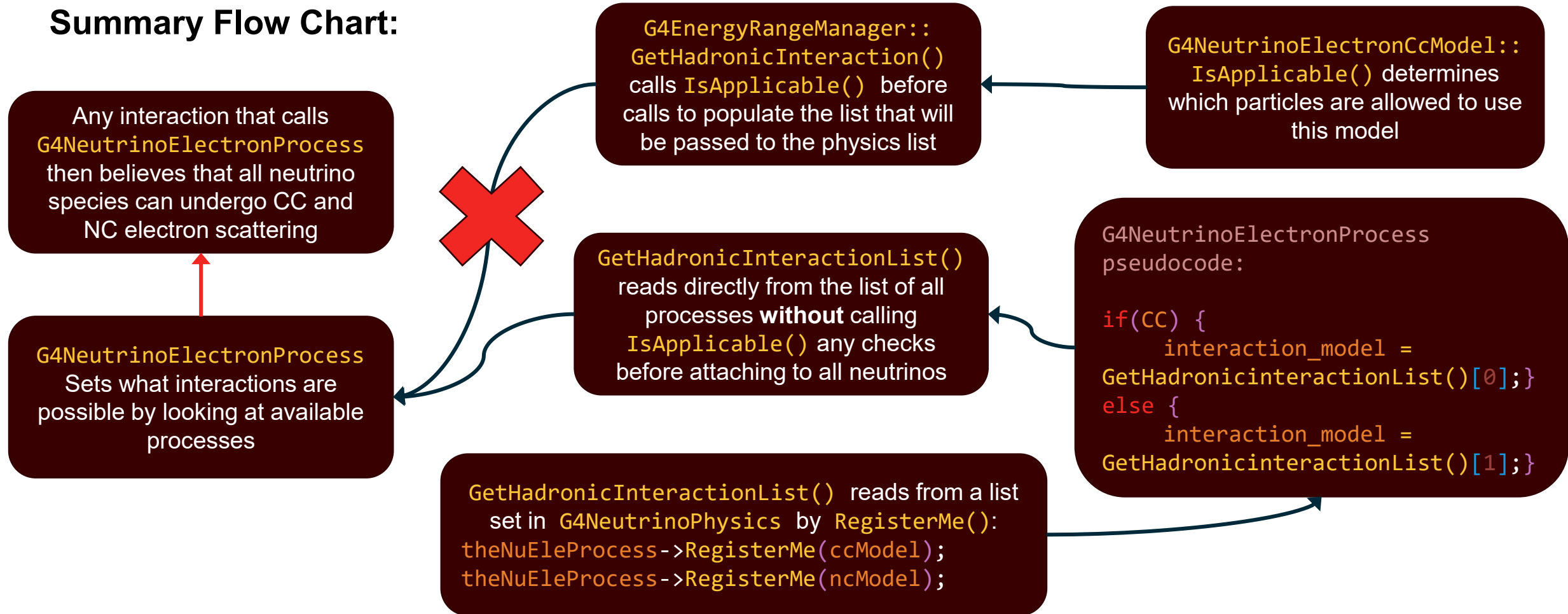
GEANT4 Vectors: 300.00 GeV Muon Antineutrino Interaction on Unknown Target
Sum of Secondary Total Energy: 248.34 GeV

- By simulating 300.00 GeV Muon Anti Neutrinos in G4TargetPractice with high neutrino electron CC biasing, we can see unphysical interactions
- This interaction was generated using G4TargetPractice, which uses the more standard physics list biasing scheme



Muon/Tau Anti Neutrino CC Electron Scattering

Summary Flow Chart:



Muon/Tau Anti Neutrino CC Electron Scattering

- Implications
 - This is an issue within the very structure of `G4NeutrinoElectronProcess`.
 - It is not a simple fix, it would require lots of delicate architectural changes, potentially necessitating an overhaul of Geant4 neutrino process handling logic across other processes to ensure parallel structure.
 - Because, as we have seen, this is ultimately a problem inherited from `G4NeutrinoElectronProcess` and `G4NeutrinoPhysics`, any simulation using the standard `G4NeutrinoPhysics` list will be vulnerable to these unphysical interactions, as evidenced by the output on slide 7 from G4TargetPractice.
 - Patching over the issue downstream may similarly be difficult, but is likely more feasible.

Summary of All Issues

- Neutrino Name Typo
- Incorrect IMD Formula
- Muon Anti Neutrino CC Electron Scattering
- Momentum Imbalances
- Charge Imbalances
- All Issues have bug reports submitted for them
- The typos have PRs made as well on the Geant4 GitHub
- The more complex issues point more deeply to issues with Geant4's neutrino event generation capabilities
- Geant4, seemingly, has never been rigorously validated in its neutrino region, especially in the highly complex regime of DIS at energies > 1 TeV

In light of the issues documented here and those yet undiscovered, Geant4 should likely be abandoned for studies as a neutrino event generator, with GENIE replacing it.

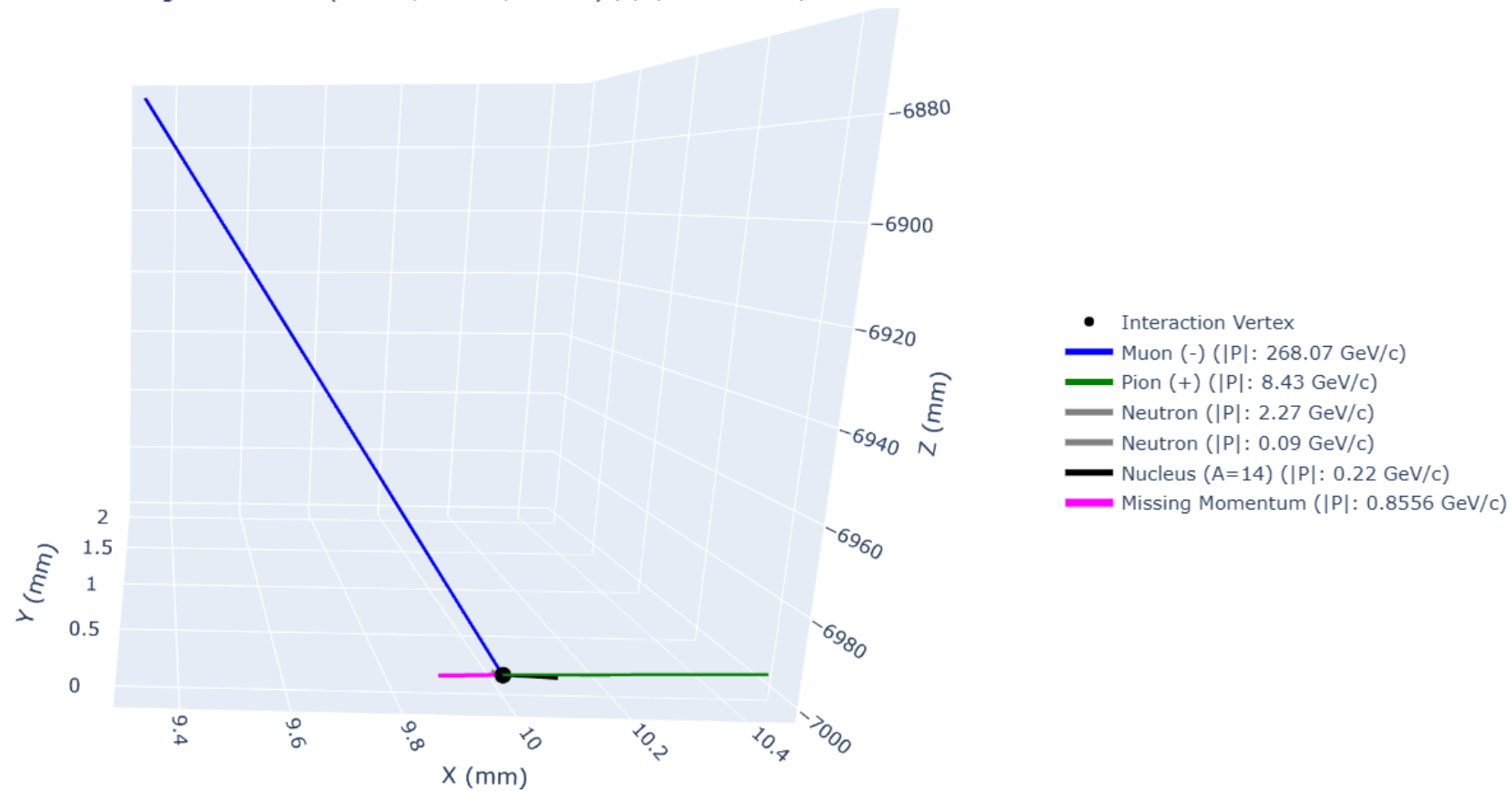
Backups/Verbose Explanations

Event 5 - 278.02 GeV Muon Neutrino on Tungsten-182 (Coherent Nucleus)

Incident Momentum: $(-0.164, 3.840, 277.992)$ | $|P|$: 278.018 GeV/c

Secondaries Sum: $(0.071, 3.843, 278.815)$ | $|P|$: 278.841 GeV/c

Missing Momentum: $(-0.235, -0.003, -0.823)$ | $|P|$: 0.856 GeV/c



Muon/Tau Anti Neutrino CC Electron Scattering

- In `G4NeutrinoElectronCcModel::IsApplicable()` we see that there does in fact exist a proper check that attempts to prevent these unphysical interactions in line 103.
- So we search for where `IsApplicable()` is actually called

```
98  G4bool G4NeutrinoElectronCcModel::IsApplicable(const G4HadProjectile & aPart,  
99  | | | | | | | G4Nucleus & )  
100 {  
101     G4bool result = false;  
102     G4String pName = aPart.GetDefinition()->GetParticleName();  
103     if(pName == "anti_nu_mu" || pName == "anti_nu_tau") return result; // no cc for anti_nu_(mu,tau)  
104     G4double minEnergy = 0., energy = aPart.GetTotalEnergy();  
105     G4double fmass, emass = electron_mass_c2;  
106  
107     if(      pName == "nu_mu" ) fmass = theMuonMinus->GetPDGMass();  
108     else if( pName == "nu_tau" ) fmass = theTauMinus->GetPDGMass();  
109     else  
110         fmass = emass;  
111  
112     minEnergy = (fmass-emass)*(fmass+emass)/emass;  
113     SetMinEnergy( minEnergy );  
114  
115     if( ( pName == "nu_mu" || pName == "nu_tau" || pName == "anti_nu_e" ) && energy > minEnergy )  
116     {  
117         result = true;  
118     }  
119     return result;  
120 }
```

`G4NeutrinoElectronNcModel.cc` excerpt. See line 87 for the typo.

Muon/Tau Anti Neutrino CC Electron Scattering

- We find `IsApplicable()` in `G4EnergyRangeManager.cc`, where it is called as shown
- This call *should* be running the check from the previous step
- In order to understand a potentially broken process, we first must define what a “working” process is

```
81     for (G4int i = 0; i<theHadronicInteractionCounter; ++i) {
82         if ( theHadronicInteraction[i]->IsApplicable( aHadProjectile, aTargetNucleus ) ) {
83             G4double low  = theHadronicInteraction[i]->GetMinEnergy( aMaterial, anElement );
84             G4double high = theHadronicInteraction[i]->GetMaxEnergy( aMaterial, anElement );
85             if (low <= kineticEnergy && high >= kineticEnergy) {
86                 ++cou;
87                 emi2 = emi1;
88                 ema2 = ema1;
89                 emi1 = low;
90                 ema1 = high;
91                 memor2 = memory;
92                 memory = i;
93             }
94         }
95     }
```

`G4EnergyRangeManager.cc` excerpt. The `IsApplicable()` check should be called in line 82, and the interaction should be skipped if it fails that check

Muon/Tau Anti Neutrino CC Electron Scattering

- So we look at a process that is known to operate correctly, `G4HadronElasticProcess`
- This process is highly general and uses the standard Geant4 architecture, so it serves as a perfect example.
- We see that it sets the `hadi` variable using a call to `ChooseHadronicInteraction()`, which just uses the member function `GetHadronicInteraction()` from `theEnergyRangeManager`
- And where do we recognize that from?

```
349 inline G4HadronicInteraction* G4HadronicProcess::
350 ChooseHadronicInteraction(const G4HadProjectile& aHadProjectile,
351                           G4Nucleus& aTargetNucleus,
352                           const G4Material* aMaterial,
353                           const G4Element* anElement)
354 {
355     return theEnergyRangeManager.GetHadronicInteraction(aHadProjectile,
356                                                         aTargetNucleus,
357                                                         aMaterial,anElement);
358 }
```

`G4HadronicProcess.hh` excerpt, showing that `ChooseHadronicInteraction()` is just a method of calling `GetHadronicInteraction()` from `theEnergyRangeManager`

```
148     hadi = ChooseHadronicInteraction( theProj, *targNucleus, material, elm );
```

`G4HadronElasticProcess.cc` line 148, showing how the `hadi` variable is supposed to be properly set. `hadi` is a `G4HadronicInteraction` pointer as set on line 103 of the same file.

Muon/Tau Anti Neutrino CC Electron Scattering

- From right here in `G4EnergyRangeManager.cc`!
- `G4EnergyRangeManager::GetHadronicInteraction()` is the function that contains the `IsApplicable()` check we needed!
- This has shown that a correctly implemented process sets its `hadi` variable through using a call to `G4EnergyRangeManager::GetHadronicInteraction()`, which runs the `IsApplicable()` check
- If the neutrino electron process does this, then the unphysical processes should be forbidden!

```
81     for (G4int i = 0; i<theHadronicInteractionCounter; ++i) {
82         if ( theHadronicInteraction[i]->IsApplicable( aHadProjectile, aTargetNucleus ) ) {
83             G4double low  = theHadronicInteraction[i]->GetMinEnergy( aMaterial, anElement );
84             G4double high = theHadronicInteraction[i]->GetMaxEnergy( aMaterial, anElement );
85             if (low <= kineticEnergy && high >= kineticEnergy) {
86                 ++cou;
87                 emi2 = emi1;
88                 ema2 = ema1;
89                 emi1 = low;
90                 ema1 = high;
91                 memor2 = memory;
92                 memory = i;
93             }
94         }
95     }
```

`G4EnergyRangeManager.cc` excerpt. The `IsApplicable()` check should be called in line 82, and the interaction should be skipped if it fails that check

Muon/Tau Anti Neutrino CC Electron Scattering

- But `G4NeutrinoElectronProcess` does not do this!
- Instead, it sets the `hadi` variable (which contains the information of the interaction model) by grabbing it directly from `GetHadronicInteractionList()`
- In the CC case, the hadronic model it uses is taken from index 0
- In the NC case, the hadronic model is taken from index 1
- But how is this list set?

```
246 G4double ccTotRatio = fTotXsc->GetCcRatio();
247
248 if( G4UniformRand() < ccTotRatio ) // Cc-model
249 {
250     // Initialize the hadronic projectile from the track
251     thePro.Initialise(track);
252
253     hadi = (GetHadronicInteractionList())[0];
254
255     result = hadi->ApplyYourself( thePro, *targNucleus);
256
257     result->SetTrafoToLab(thePro.GetTrafoToLab());
258
259     ClearNumberOfInteractionLengthLeft();
260
261     FillResult(result, track);
262 }
263 else // Nc-model, like 'elastic', 2->2 scattering
264 {
265
266     hadi = (GetHadronicInteractionList())[1];
```

`G4NeutrinoElectronProcess.cc` excerpt. Lines 253 & 266 show that the `hadi` variable is hardcoded from `GetHadronicInteractionList()` rather than set from a call to `ChooseHadronicInteraction()`.

Muon/Tau Anti Neutrino CC Electron Scattering

- `GetHadronicInteractionList()` is actually just a way of getting access to `theHadronicInteraction`, which is a private vector of pointers to available hadronic interactions that is populated by calling `G4EnergyRangeManager::RegisterMe()`

```
60 private:
61
62     G4EnergyRangeManager(const G4EnergyRangeManager& right);
63     G4EnergyRangeManager& operator=(const G4EnergyRangeManager& right);
64     G4bool operator==(const G4EnergyRangeManager& right) const;
65     G4bool operator!=(const G4EnergyRangeManager& right) const;
66
67     G4int theHadronicInteractionCounter;
68     std::vector<G4HadronicInteraction*> theHadronicInteraction;
69 };
```

`G4EnergyRangeManager.hh` excerpt, showing that `theHadronicInteraction` is a private variable that is a vector of `G4HadronicInteraction` pointers

```
46 void G4EnergyRangeManager::RegisterMe(G4HadronicInteraction* a)
47 {
48     if(!a) { return; }
49     if(0 < theHadronicInteractionCounter) {
50         for(G4int i=0; i<theHadronicInteractionCounter; ++i) {
51             if(a == theHadronicInteraction[i]) { return; }
52         }
53     }
54     theHadronicInteraction.push_back(a);
55     ++theHadronicInteractionCounter;
56 }
```

`G4EnergyRangeManager.cc` excerpt, showing where `theHadronicInteraction` is populated by `RegisterMe()`

```
155 std::vector<G4HadronicInteraction*>&
156 G4EnergyRangeManager::GetHadronicInteractionList()
157 {
158     return theHadronicInteraction;
159 }
```

`G4EnergyRangeManager.cc` excerpt, showing that `GetHadronicInteractionList()` is just a function that gives access to a variable within `G4EnergyRangeManager.cc` that stores hadronic interaction information

Muon/Tau Anti Neutrino CC Electron Scattering

- And finally, where is `RegisterMe()` called?
- `RegisterMe()` is called in the physics list itself!
- `G4NeutrinoPhysics.cc` populates `theHadronicInteraction` with `RegisterMe()` on the CC and NC electron scattering models, and it adds it to the 6 neutrinos denoted as p here.
- And notice that the CC interaction is at index 0, and the NC interaction is at index 1!
- That should remind us of something...

```
190  G4NeutrinoElectronCcModel* ccModel = new G4NeutrinoElectronCcModel();
191  G4NeutrinoElectronNcModel* ncModel = new G4NeutrinoElectronNcModel();
192  theNuEleProcess->RegisterMe(ccModel);
193  theNuEleProcess->RegisterMe(ncModel);
194
195  for (G4int i=0; i<6; ++i) {
196      p[i]->GetProcessManager()->AddDiscreteProcess(theNuEleProcess);
197  }
```

`G4NeutrinoPhysics.cc` excerpt, showing that, for the electron scattering process, `RegisterMe()` is filling `theHadronicInteraction` with the CC and NC models directly!

Muon/Tau Anti Neutrino CC Electron Scattering

- Recall that `G4NeutrinoElectronProcess` never went through the steps of running the `IsApplicable()` check
- It instead was hardcoded to just pick the interaction models at specific indices
 - CC at 0...
 - And NC at 1!
- So, we have now discovered exactly how `G4NeutrinoElectronProcess` sidesteps the checks that do exist to prevent unphysical interactions

```
246     G4double ccTotRatio = fTotXsc->GetCcRatio();
247
248     if( G4UniformRand() < ccTotRatio ) // Cc-model
249     {
250         // Initialize the hadronic projectile from the track
251         thePro.Initialise(track);
252
253         hadi = (GetHadronicInteractionList())[0];
254
255         result = hadi->ApplyYourself( thePro, *targNucleus);
256
257         result->SetTrafoToLab(thePro.GetTrafoToLab());
258
259         ClearNumberOfInteractionLengthLeft();
260
261         FillResult(result, track);
262     }
263     else // Nc-model, like 'elastic', 2->2 scattering
264     {
265
266         hadi = (GetHadronicInteractionList())[1];
```

`G4NeutrinoElectronProcess.cc` excerpt. Lines 253 & 266 show that the `hadi` variable is hardcoded from `GetHadronicInteractionList()` rather than set from a call to `ChooseHadronicInteraction()`.

Impossible to Change Energy Momentum Check Thresholds

```
87 // --- Muon Neutrinos: CC and NC ---
88 auto muCC = new G4HadronicProcess("muNuNucleusCC");
89 muCC->RegisterMe(new MyNuMuNucleusCcModel());
90 muCC->AddDataSet(new MyMuNeutrinoNucleusCcXsc());
91 muCC->SetEnergyMomentumCheckLevels(0.001, 10.0 * CLHEP::MeV);
92 muCC->SetEpReportLevel(3);
93
94 // =====
95 // Query the model to see if it actually accepted the new limits
96 std::pair<G4double, G4double> testLevels = ccModel->GetFatalEnergyCheckLevels();
97 G4cout << "\n=====
98 << G4endl;
99 G4cout << "TESTING ccModel LIMITS IN PHYSICS LIST:" << G4endl;
100 G4cout << "Expected Relative: 0.001 | Actual Relative: "
101 << testLevels.first << G4endl;
102 G4cout << "Expected Absolute: 10.0 | Actual Absolute: "
103 << testLevels.second / CLHEP::MeV << " MeV" << G4endl;
104 G4cout << "=====
105 << G4endl;
106 // =====
107
```

=====

TESTING ccModel LIMITS IN PHYSICS LIST:

Expected Relative: 0.001 | Actual Relative: 0.02

Expected Absolute: 10.0 | Actual Absolute: 1000 MeV

=====

Top: Code in my G4 simulation suite that should set and then immediately check the energy momentum check levels of the model.

Bottom: Terminal output showing that setting the energy & momentum thresholds this way accomplishes nothing, implying that, although G4 allows the user to set these values, it refuses to actually read them

```
652 std::pair<G4double, G4double> checkLevels =
653 | theModel->GetFatalEnergyCheckLevels(); // (relative, absolute)
654 if (std::abs(deltaE) > checkLevels.second &&
655 | std::abs(deltaE) > checkLevels.first*aPro.GetKineticEnergy()){
```

G4HadronicProcess.cc inside `CheckResult()`, showing that the `checkLevels` are exclusively set by a single call to `GetFatalEnergyCheckLevels()`. The remainder of the logic underneath is suppressed here, but the event is either resampled or killed if 100 resamples have been attempted

```
210 const std::pair<G4double, G4double> G4HadronicInteraction::GetFatalEnergyCheckLevels() const
211 {
212 | // default Level of Check
213 | return std::pair<G4double, G4double>(2.*perCent, 1. * GeV);
214 }
```

G4HadronicInteraction.cc, showing that the call to `GetFatalEnergyCheckLevels()` is hardcoded to return a default value, so the values set by `SetEnergyMomentumCheckLevels()` are never read anywhere.

THIS IS ACTUALLY FINE BECAUSE THE VALUE THAT GETS CHANGED IS NOT the one set by `GetFatalEnergyCheckLevels()`. It is set by ``param->GetEPRelativeLevel()``. Those values do get changed, as evidenced by

```
Process: muNuNucleusCC , Model: NuMuNuclCcModel
Primary: nu_mu (14), E= 300000, target nucleus (13,27)
pass relative, limit 0.001, values E/T(0) = -8.27427e-05 p/p(0)= 0.000410084
fail absolute, limit (MeV) 10, values E / p (MeV) = -24.8228 / 123.025 3mom:
(113.004,14.4631,46.4334)
fail charge/baryon number balance 1 / 0
```

Momentum Imbalance

- I first noticed that, when biasing the channels as individual processes classified as G4HadronicProcess, I was getting warnings about off-shell mass secondaries.
- Eventually, these warnings became crashes, especially as I increased the neutrino energies.
- The Geant4 output here tells us that, sometimes, Geant4 is seeing particles with impossible dynamic masses and attempts to resample to fix this.
- If it cannot generate an acceptable dynamic mass within 100 attempts, it crashes the program.

```

----- WWW ----- G4Exception-START ----- WWW -----
*** G4Exception : had012
      issued by : G4HadronicProcess::CheckResult()
Warning: Secondary with off-shell dynamic mass detected:
      pi0, PDG mass: 134.977, dynamic mass: 1521.41
re-sample the interaction
      Process / Model: muNuNucleusCC / NuMuNuclCcModel
      Primary: nu_mu (14), E= 300000, target nucleus (14, 28)

*** This is just a warning message. ***
----- WWW ----- G4Exception-END ----- WWW -----

----- EEEE ----- G4Exception-START ----- EEEE -----
*** G4Exception : had006
      issued by : G4HadronicProcess::PostStepDoIt
Call for NuMuNuclCcModel
Target element SI_elm Z= 14 A= 28
Unrecoverable error in the method ApplyYourself of muNuNucleusCC
TrackID= 1 ParentID= 0 nu_mu
Ekin(GeV)= 300; direction= (-0.289445,0.952581,-0.0938611)
Position(mm)= (-29.7968,98.0631,-9.66249); material Silicon
PhysicalVolume <av_7_impr_6_VertexBarrel_layer6_supp_pv_54>
ApplyYourself does not completed after 100 attempts

*** Fatal Exception *** core dump ***
G4Track (0x7fc4d878ea70) - track ID = 1, parent ID = 0
Particle type : nu_mu - creator process : not available
Kinetic energy : 300 GeV - Momentum direction : (-0.289445,0.952581,-0.0938611)
Step length : 128.304 um - total energy deposit : 0 eV
Pre-step point : (-29.7597,97.9409,-9.65045) - Physical volume :
av_7_impr_6_VertexBarrel_layer6_supp_pv_54 (Silicon)
- defined by : Transportation - step status : 1
Post-step point : (-29.7968,98.0631,-9.66249) - Physical volume :
av_7_impr_6_VertexBarrel_layer6_supp_pv_54 (Silicon)
- defined by : biasWrapper(muNuNucleusCC) - step status : 4
*** Note: Step information might not be properly updated.

----- EEEE ----- G4Exception-END ----- EEEE -----

*** G4Exception: Aborting execution ***
Aborted (core dumped)

```

Output while biasing neutrino interactions as HadronicProcess. Primary particle was a 300 GeV muon neutrino with biased nuclear charged current scattering Warnings about off-shell dynamic mass secondaries eventually crash the program.

Momentum Imbalance

- I investigated G4TargetPractice, where these errors do not appear during simulation, and I looked into its energy balancing. I found that it balances energy perfectly.
- But I then turned to the momentum balance, and I found that the secondaries never fully reconstructed the momentum of the incoming neutrino

```
*** NEUTRINO INTERACTION RECORDED ***
Process: muNuNucleusCC | Type: CC | Model: CCRES

===== RUNTIME KINEMATIC DIAGNOSTIC =====
Event ID: 35
Incident nu_mu Momentum: (129.699,-194.71,-187.793) GeV/c
-> Daughter: pi+ | P: (-3.30169,4.8372,5.59376) GeV/c
-> Daughter: A127 | P: (23.3559,-34.219,-39.5688) GeV/c
-> Daughter: gamma | P: (0.00695223,-0.00783824,-0.00748571)
GeV/c
-> Daughter: gamma | P: (0.00156612,-0.0033113,-0.00700661)
GeV/c
-> Daughter: gamma | P: (0.00489585,-0.00753664,-0.00928174)
GeV/c
-> Daughter: mu- | P: (109.319,-164.852,-153.266) GeV/c
-----
Vector Sum Out: (129.387,-194.252,-187.264) GeV/c
Missing Vector: (0.312173,-0.457355,-0.528888) GeV/c
Magnitude Missing: 0.765734 GeV/c
=====
```

Output with MyExceptionHandler silencing and handling warning and errors, respectively. Both are for 300 GeV, nu_mu

Momentum Imbalance

```
758 // Evaluate relative and absolute conservation
759 G4bool relPass = true;
760 G4String relResult = "pass";
761 if ( std::abs(relative) > checkLevels.first
762     || std::abs(relative_mom) > checkLevels.first ) {
763     relPass = false;
764     relResult = checkRelative ? "fail" : "N/A";
765 }
766
767 G4bool absPass = true;
768 G4String absResult = "pass";
769 if ( std::abs(absolute) > checkLevels.second
770     || std::abs(absolute_mom) > checkLevels.second ) {
771     absPass = false ;
772     absResult = "fail";
773 }
```

G4HadronicProcess.cc lines 761-762 & 769-770 show that checkLevels sets the levels that the energy and momentum checks are run against, so we check that variable

```
210 const std::pair<G4double, G4double> G4HadronicInteraction::GetFatalEnergyCheckLevels() const
211 {
212     // default level of Check
213     return std::pair<G4double, G4double>(2.*perCent, 1. * GeV);
214 }
```

G4HadronicInteraction.cc, definition of the fatal energy (and momentum) check level defaults

G4NeutrinoPhysics.cc excerpt, showing that, for the electron scattering process, `RegisterMe()` is filling `theHadronicInteraction` with the CC and NC models directly!

Momentum Imbalance

```
758 // Evaluate relative and absolute conservation
759 G4bool relPass = true;
760 G4String relResult = "pass";
761 if ( std::abs(relative) > checkLevels.first
762     || std::abs(relative_mom) > checkLevels.first ) {
763     relPass = false;
764     relResult = checkRelative ? "fail" : "N/A";
765 }
766
767 G4bool absPass = true;
768 G4String absResult = "pass";
769 if ( std::abs(absolute) > checkLevels.second
770     || std::abs(absolute_mom) > checkLevels.second ) {
771     absPass = false ;
772     absResult = "fail";
773 }
```

G4HadronicProcess.cc lines 761-762 & 769-770 show that `checkLevels` sets the levels that the energy and momentum checks are run against, so we check that variable

```
210 const std::pair<G4double, G4double> G4HadronicInteraction::GetFatalEnergyCheckLevels() const
211 {
212     // default level of Check
213     return std::pair<G4double, G4double>(2.*perCent, 1. * GeV);
214 }
```

G4HadronicInteraction.cc, definition of the fatal energy (and momentum) check level defaults from

`G4HadronicInteraction::GetFatalEnergyCheckLevels()`