Uniquness Test in DSelector

Benedikt Zihlmann

August 10, 2020

Introduction

Uniqueness test in DSelector with the use of `std::set` and `std::map`

Look at 3 different final states:

- $\gamma + p \rightarrow \rightarrow p + 6\gamma$
- $\gamma + p \rightarrow p + \pi^+ + \pi^- + 4\gamma$
- $\gamma + p \rightarrow p + \pi^+ + \pi^- + 6\gamma$

Kinematic fit all masses (η, π^0) are NOT constrained!

Example of `std::set`

Listing 1: C++ code Create set and map

```
1   map<Particle_t, set<Int_t> > locThisCombo;
2   set<Int_t> locCombo_Neutrals; //Final State
3   set<Int_t> locCombo_Proton; // Final State
4   set<Int_t> locCombo_BeamPhoton;// Initial State
5
6   locCombo_BeamPhoton.insert(locBeamID);
7
8   locCombo_Neutrals.insert(locPhoton1NeutralID);
9   locCombo_Neutrals.insert(locPhoton2NeutralID);
10  locCombo_Neutrals.insert(locPhoton3NeutralID);
11  locCombo_Neutrals.insert(locPhoton4NeutralID);
12  locCombo_Neutrals.insert(locPhoton5NeutralID);
13  locCombo_Neutrals.insert(locPhoton6NeutralID);
14
15  locCombo_Proton.insert(locProtonTrackID);
```

Example of `std::set`

Listing 2: C++ code Fill set and map

```
1     locThisCombo.insert(std::make_pair(Proton,locCombo_Proton));
2     locThisCombo.insert(std::make_pair(Gamma, locCombo_Neutrals));
3     locThisCombo.insert(std::make_pair(Unknown, locCombo_BeamPhoton));
4     // use "Unknown" for beam gamma
5
6     // alternatively one could do:
7     // locThisCombo[Proton] = locCombo_Proton;
8     // locThisCombo[Gamma] =  locCombo_Neutrals;
9
10    RetVal = locUsedSoFar_FS.insert(locThisCombo);
11    int ThisComboHasBeenSeenAlready = 0;
12    if (!RetVal.second){
13      // this combo already has been done.
14      ThisComboHasBeenSeenAlready = 1;
15    }
```

Example of `std::set`

Listing 3: C++ code Use set and map

```
1         if (ThisComboHasBeenSeenAlready){
2       char str1[148];
3       sprintf(str1, ,
4           locRunNumber, locEntry,
5           Weight, locBeamID, locProtonTrackID,
6           locPhoton1NeutralID, locPhoton2NeutralID, locPhoton3NeutralID, locPhoton4NeutralID,
7           locPhoton5NeutralID, locPhoton6NeutralID, KinFitchi2);
8       if( NNeutralShowers == 6) {
9         if (OCounter[0]<100000){
10          OUTFCe6<<str1<<endl;
11        }
12        OCounter[0]++;
13      } else{
14        if (OCounter[1]<100000){
15          OUTFCm6<<str1<<endl;
16        }
17        OCounter[1]++;
18      }
19      dComboWrapper->Set_IsComboCut(true);
20      continue;
21        }
```

Example of `std::set`

The resulting output:

```
51440 114 :  (W=-0.1) 0 3 0 5 4 3 6 1 chi2 = 5.712e+02
51440 123 :  (W=-0.1) 0 1 1 2 0 4 3 5 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 3 0 5 2 1 4 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 0 5 1 2 3 4 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 0 2 3 5 1 4 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 0 2 3 4 5 1 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 0 4 3 2 5 1 chi2 = 5.913e+02
51440 123 :  (W=-0.1) 0 1 3 0 2 4 5 1 chi2 = 5.913e+02
51440 126 :  (W=-0.1) 0 2 8 5 7 1 12 2 chi2 = 5.668e+02
51440 126 :  (W=-0.1) 0 2 5 1 7 2 8 12 chi2 = 5.668e+02
51440 126 :  (W=-0.1) 0 2 5 1 7 8 12 2 chi2 = 5.668e+02
51440 129 :  (W=-0.1) 0 1 2 5 8 1 4 6 chi2 = 3.316e+03
51440 129 :  (W=-0.1) 0 1 2 6 4 5 8 1 chi2 = 3.316e+03
51440 131 :  (W=-0.1) 0 1 7 5 6 0 2 1 chi2 = 1.529e+02
51440 131 :  (W=-0.1) 0 1 7 1 6 0 5 2 chi2 = 1.529e+02
51440 131 :  (W=-0.1) 0 1 7 5 0 1 6 2 chi2 = 1.529e+02
51440 134 :  (W= 1.0) 0 1 2 1 5 3 0 4 chi2 = 2.489e+02
51440 134 :  (W= 1.0) 0 1 2 1 5 0 3 4 chi2 = 2.489e+02
```

$\pi^+\pi^-4\gamma$ Final State

Results:

```
30680 34 :  (W=-0.1) 0 ( 1, 5, 4) [ 4, 6, 5, 0] chi2 = 1.090e+03
30680 86 :  (W=-0.1) 0 ( 5, 1, 4) [ 6, 1, 3, 0] chi2 = 1.589e+03
30680 97 :  (W=-0.1) 0 ( 4, 8, 3) [ 0, 5, 1, 3] chi2 = 2.747e+02
30680 100 :  (W=-0.1) 0 ( 7, 1, 6) [ 7, 1, 0, 2] chi2 = 7.502e+02
30680 112 :  (W=-0.1) 0 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 9.388e+01
30680 112 :  (W=-0.1) 1 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 7.726e+01
30680 112 :  (W=-0.1) 2 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 9.857e+01
30680 112 :  (W= 1.0) 3 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 6.311e+01
30680 112 :  (W= 1.0) 4 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 5.188e+01
30680 112 :  (W= 1.0) 5 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 5.616e+01
30680 112 :  (W=-0.1) 6 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 1.337e+02
30680 112 :  (W=-0.1) 7 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 5.445e+01
30680 112 :  (W=-0.1) 8 ( 1, 3, 2) [ 4, 1, 3, 2] chi2 = 5.145e+01
30680 124 :  (W=-0.1) 0 ( 2, 1, 3) [ 3, 1, 0, 2] chi2 = 2.648e+02
30680 124 :  (W=-0.1) 1 ( 2, 1, 3) [ 3, 1, 0, 2] chi2 = 4.086e+02
30680 124 :  (W=-0.1) 2 ( 2, 1, 3) [ 3, 1, 0, 2] chi2 = 1.031e+02
30680 124 :  (W= 1.0) 3 ( 2, 1, 3) [ 3, 1, 0, 2] chi2 = 6.527e+01
30680 135 :  (W=-0.1) 0 ( 3, 4, 1) [ 2, 4, 1, 0] chi2 = 7.126e+01
30680 135 :  (W=-0.1) 1 ( 3, 4, 1) [ 2, 4, 1, 0] chi2 = 3.218e+02
30680 135 :  (W= 1.0) 2 ( 3, 4, 1) [ 2, 4, 1, 0] chi2 = 1.249e+02
30680 135 :  (W=-0.1) 3 ( 3, 4, 1) [ 2, 4, 1, 0] chi2 = 2.382e+02
30680 135 :  (W=-0.1) 4 ( 3, 4, 1) [ 2, 4, 1, 0] chi2 = 4.428e+02
30680 135 :  (W=-0.1) 0 ( 3, 4, 1) [ 2, 0, 1, 4] chi2 = 7.126e+01
30680 135 :  (W=-0.1) 1 ( 3, 4, 1) [ 2, 0, 1, 4] chi2 = 3.218e+02
30680 135 :  (W= 1.0) 2 ( 3, 4, 1) [ 2, 0, 1, 4] chi2 = 1.249e+02
30680 135 :  (W=-0.1) 3 ( 3, 4, 1) [ 2, 0, 1, 4] chi2 = 2.382e+02
30680 135 :  (W=-0.1) 4 ( 3, 4, 1) [ 2, 0, 1, 4] chi2 = 4.428e+02
```

$\pi^+\pi^-6\gamma$ Final State

Results:

```
51020  4 :  (W=-0.1) 0 ( 5, 1, 3) [ 9, 3, 8, 6, 4, 2] chi2 = 3.334e+02
51020  4 :  (W=-0.1) 0 ( 1, 5, 3) [ 9, 3, 8, 6, 4, 2] chi2 = 3.680e+02
51020  5 :  (W= 1.0) 0 ( 5, 2, 4) [ 1, 2, 4, 6, 0, 5] chi2 = 1.325e+03
51020  5 :  (W= 1.0) 1 ( 5, 2, 4) [ 1, 2, 4, 6, 0, 5] chi2 = 1.325e+03
51020  5 :  (W=-0.1) 2 ( 5, 2, 4) [ 1, 2, 4, 6, 0, 5] chi2 = 1.323e+03
51020  5 :  (W=-0.1) 3 ( 5, 2, 4) [ 1, 2, 4, 6, 0, 5] chi2 = 1.322e+03
51020  7 :  (W=-0.1) 0 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 1.695e+01
51020  7 :  (W=-0.1) 1 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 1.244e+01
51020  7 :  (W= 1.0) 2 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 9.777e+00
51020  7 :  (W=-0.1) 3 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 1.030e+02
51020  7 :  (W=-0.1) 4 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 1.252e+02
51020  7 :  (W=-0.1) 5 ( 2, 3, 1) [ 0, 4, 5, 1, 2, 3] chi2 = 9.844e+01
51020 22 :  (W=-0.1) 0 ( 3, 1, 5) [ 0, 4, 1, 5, 3, 2] chi2 = 5.885e+02
51020 22 :  (W=-0.1) 1 ( 3, 1, 2) [ 0, 4, 1, 5, 3, 2] chi2 = 2.902e+02
51020 22 :  (W= 1.0) 2 ( 3, 1, 2) [ 0, 4, 1, 5, 3, 2] chi2 = 3.001e+02
51020 22 :  (W=-0.1) 0 ( 3, 1, 2) [ 0, 4, 1, 5, 3, 2] chi2 = 2.736e+02
51020 25 :  (W=-0.1) 0 ( 4, 2, 5) [ 6, 0, 4, 2, 5, 1] chi2 = 4.281e+02
51020 25 :  (W= 1.0) 1 ( 4, 2, 5) [ 6, 0, 4, 2, 5, 1] chi2 = 4.042e+02
51020 25 :  (W=-0.1) 2 ( 4, 2, 5) [ 6, 0, 4, 2, 5, 1] chi2 = 4.063e+02
```

Conclusions 1

`std::set` and `std::map` work for identifying unique combos with some caveats.

- Additional neutral showers. (unused energy)
- Additional charged tracks. (unused tracks)

Conclusions 1

`std::set` and `std::map` work for identifying unique combos with some caveats.

- Additional neutral showers. (unused energy)
- Additional charged tracks. (unused tracks)

1. Lead to possible multiple INDEPENDENT FS in same event!
2. Need to count them somehow!
3. Need to decide: select one(qualifier), select all(weight)

Conclusions 1

`std::set` and `std::map` work for identifying unique combos with some caveats.

- Additional neutral showers. (unused energy)
- Additional charged tracks. (unused tracks)

1. Lead to possible multiple INDEPENDENT FS in same event!
2. Need to count them somehow!
3. Need to decide: select one(qualifier), select all(weight)
4. FS $p + 6\gamma$ χ^2-Cut: =1(95.7%), >1(4.3%) with [1/6]2.1%, [2/X]0.07%
5. FS $p + \pi^+ + \pi^- + 4\gamma$ χ^2-Cut++: =1(85.8%), >1(14.2%) with [3/4]6.3%, [3/>4]6.3%, [4/N]1.6%
6. FS $p + \pi^+ + \pi^- + 6\gamma$ χ^2-Cut++: =1(56.9%), >1(43.1%) with [3/6]6%, [3/>6]34.2%, [4/N]1.2%

Conclusions 2

- If NO additional charged tracks and NO "unused"-energy is used, there is much less of a possibility for more than one solution:
  `std::set` with `std::map` will work quite fine caveat: multiple charged tracks in FS.

Conclusions 2

- If NO additional charged tracks and NO "unused"-energy is used, there is much less of a possibility for more than one solution:
  `std::set` with `std::map` will work quite fine caveat: multiple charged tracks in FS.
- IF additional charged tracks or "unused"-energy is allowed, there is a possiblity of more than one FS solution for the same beam photon.
  `std::set` with `std::map` will NOT work!

Conclusions 2

- If NO additional charged tracks and NO "unused"-energy is used, there is much less of a possibility for more than one solution:
  `std::set` with `std::map` will work quite fine caveat: multiple charged tracks in FS.
- IF additional charged tracks or "unused"-energy is allowed, there is a possiblity of more than one FS solution for the same beam photon.
  `std::set` with `std::map` will NOT work!
- Need a decission process!: qualifier like best χ^2 or weight=$1/N$

DSelector Example (Paul)

Example of code in DSelector:

Listing 4: C++ code Pauls example

```
1   double locMissingMassSquared = locMissingP4_Measured.M2();
2
3   map<Particle_t, set<Int_t> > locUsedThisCombo_MissingMass;
4   locUsedThisCombo_MissingMass[Unknown].insert(locBeamID); //beam
5   locUsedThisCombo_MissingMass[PiPlus].insert(locPiPlusTrackID);
6   locUsedThisCombo_MissingMass[PiMinus].insert(locPiMinusTrackID);
7   locUsedThisCombo_MissingMass[Proton].insert(locProtonTrackID);
8   locUsedThisCombo_MissingMass[Gamma].insert(locPhoton1NeutralID);
9   locUsedThisCombo_MissingMass[Gamma].insert(locPhoton2NeutralID);
10  locUsedThisCombo_MissingMass[Gamma].insert(locPhoton3NeutralID);
11  locUsedThisCombo_MissingMass[Gamma].insert(locPhoton4NeutralID);
12  locUsedThisCombo_MissingMass[Gamma].insert(locPhoton5NeutralID);
13  locUsedThisCombo_MissingMass[Gamma].insert(locPhoton6NeutralID);
14
15  //compare to what's been used so far
16  if(locUsedSoFar_MissingMass.find(locUsedThisCombo_MissingMass)
17        == locUsedSoFar_MissingMass.end())
18    {
19      dHist_MissingMassSquared->Fill(locMissingMassSquared);
20      locUsedSoFar_MissingMass.insert(locUsedThisCombo_MissingMass);
21
22    }
```