ee->mumu forward-backward asymmetry at CEPC

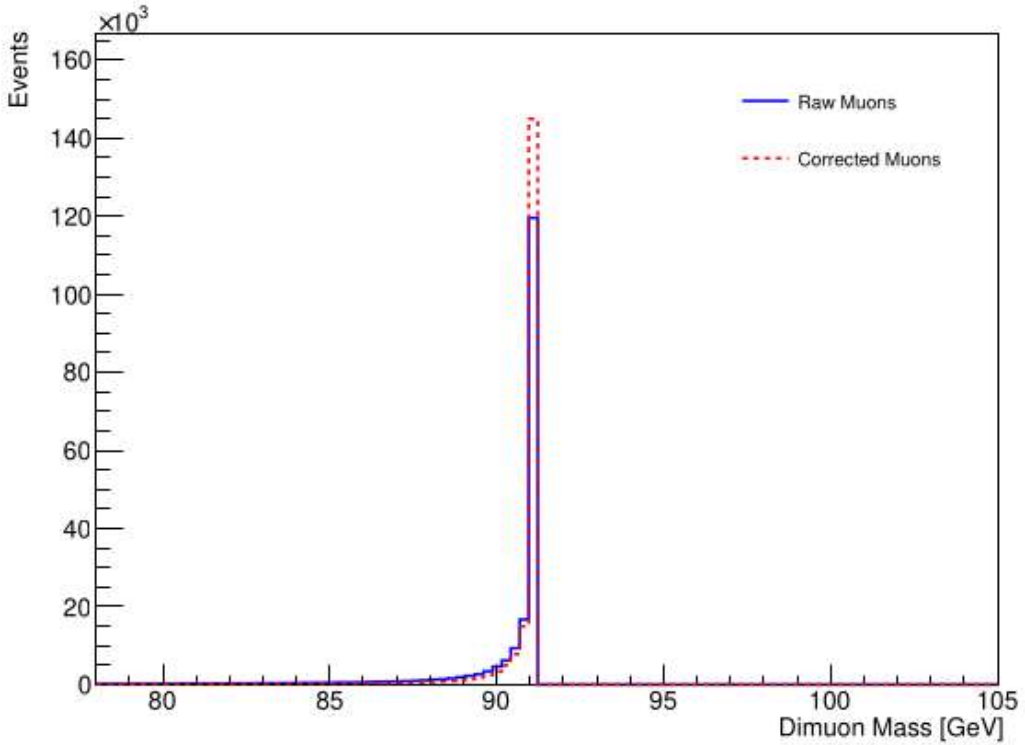
Jiawei wan , Shuo Han , Lei Zhang

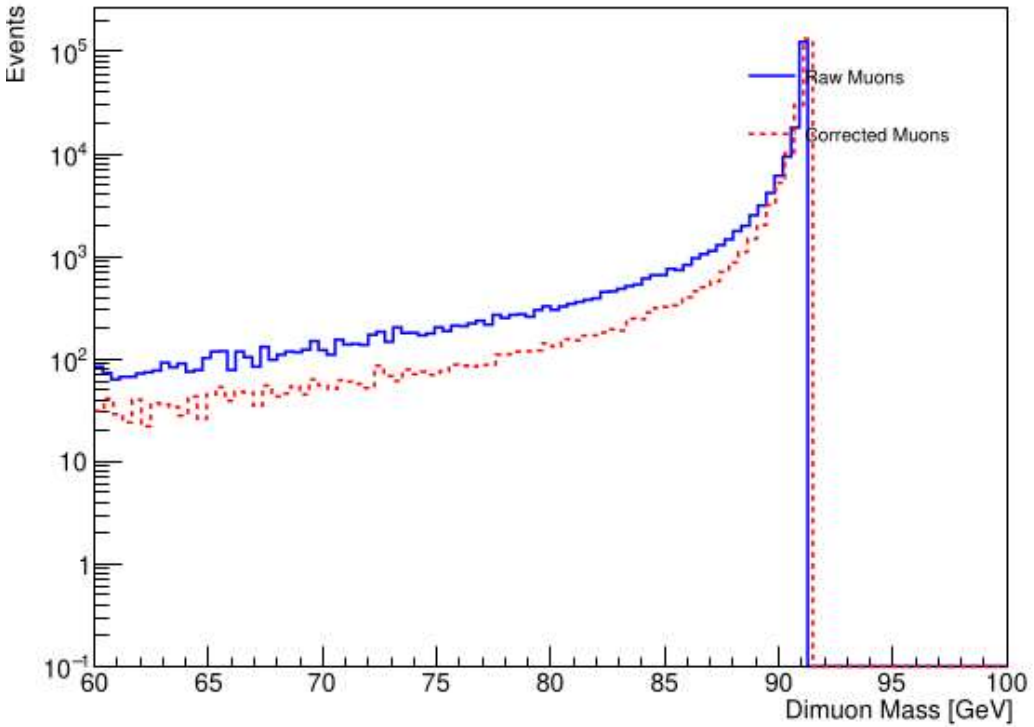
Introduction

- Check photons from final-state radiation and use them to correct the muon's four-momentum.
- Identify particles within $\Delta R < 0.3$ of the muon. If photons are found, add their four-momenta to the muon’s.
- Perform cuts and calculate $\cos\theta$ using the corrected muon four-momentum.
- This is expected to increase the number of events passing the Z mass cut.
- Approximately 20% of muons have associated particles within their vicinity, and all such particles are photons.

Corrected stdhep

collision energy : 91.1876
Total ： 200000
Original Z mass cut : 177644
Corrected Z mass cut :184239
Original AFB: 0.018798
Corrected AFB: 0.0188788





Problems

- Currently, neutral particle information is not stored in PFO.
- We examined the status of these photons in MCP; most have status 1, meaning they are detectable.
- The previous version of MissingET.cpp skipped neutral particles, while the new version no longer skips them. However, neutral particle information still does not appear in stored PFO.
- We attempted modifying PIDDumpAlg by removing the code section that skipped neutral particles to force storing neutral PFO information. Neutral particle information remains unstored.

PIDDumpAlg.cpp

```
if( PFO && fultrkparassCol) {
    for(const auto& pfo : *PFO) {
        if (DEBUG) info() << "Step4" <<endmsg;

        // 检查是否为中性粒子
        bool isNeutral = (pfo.tracks_size() == 0);
        IsNeutral.push_back(isNeutral ? 1 : 0);

        if (isNeutral) {

            int Pdgid_tmp = 0;
            bool decay = false;
            double genP = -1;
            double genTheta = -1;

            double energy = pfo.getEnergy();
            const auto& momentum = pfo.getMomentum();
            double px = momentum.x;
            double py = momentum.y;
            double pz = momentum.z;
            double p = std::sqrt(px*px + py*py + pz*pz);

            double theta = std::atan2(std::sqrt(px*px + py*py), pz) * 180.0 / M_PI;
            double phi = std::atan2(py, px) * 180.0 / M_PI;

            TLorentzVector p_pfo;
            p_pfo.SetPxPyPzE(px, py, pz, energy);

            int mcpidx = -999;
            double DR_min = 999;

            int i = 0;
            for(const auto& Gen : *MCParticlesGen) {
                if (Gen.getGeneratorStatus() != 1) continue;
                if (abs(Gen.getPDG()) == 12 || abs(Gen.getPDG()) == 14 || abs(Gen.getPDG()) == 16) continue;

                TLorentzVector p_mcp;
                p_mcp.SetPxPyPzE(Gen.getMomentum()[0], Gen.getMomentum()[1], Gen.getMomentum()[2], Gen.getEnergy());
                double deltaR = p_mcp.DeltaR(p_pfo);
                double deltaE = fabs(Gen.getEnergy() - energy);
                double e_thr = 3 * std::sqrt(0.00022201 / Gen.getEnergy() + 0.00000625);
```

```
// 存储基本信息
PDGID.push_back(Pdgid_tmp);
GenP.push_back(genP);
GenTheta.push_back(genTheta);
P.push_back(p);
Theta.push_back(theta);
Phi.push_back(phi);
Charge.push_back(0);

Chi2TPC.push_back(std::vector<double>(5, -999.0));
Chi2TOF.push_back(std::vector<double>(5, -999.0));
prob.push_back(std::vector<double>(5, -999.0));
Tof.push_back(-999.0);
Dndx.push_back(-999.0);

double eecal = 0.0;
double ehcal = 0.0;

if (pfo.clusters_size() > 0) {

    auto cluster = pfo.getClusters(0);
    eecal = cluster.getEnergy();

    if (pfo.clusters_size() > 1) {
        auto hcal_cluster = pfo.getClusters(1);
        ehcal = hcal_cluster.getEnergy();
    }
}

Eecal.push_back(eecal);
Eecalp.push_back(-999.0);
Ehcal.push_back(ehcal);
Ehcalp.push_back(-999.0);

Lecal.push_back(-999.0);
Lhcal.push_back(-999.0);
R90ecal.push_back(-999.0);
R90hcal.push_back(-999.0);
Weta2ecal.push_back(-999.0);
Weta2hcal.push_back(-999.0);
Wphi2ecal.push_back(-999.0);
Wphi2hcal.push_back(-999.0);

MindR.push_back(-999.0);
MindR1.push_back(-999.0);
MindR2.push_back(-999.0);
MindR_last.push_back(-999.0);

PID.push_back(0);
Nhad.push_back(-999);
Nmuon.push_back(-999);
```