Truth Parameter Smearing

Shyam Kumar, Annalisa Mastroserio, Domenico Elia
INFN Bari, Italy

Comment on my previous presentation there was a default cut of 1 KeV in epic 22.10.0 for MPGD detecter which was fixed in the later versions
This was causing the miss of hits in MPGD in some of the events

Truth Parameters in Fun4All

int PHG4TrackFastSim::process_event(PHCompositeNode* /*topNode*/)

int PseudoPatternRecognition(const PHG4Particle* particle,

std::vector<PHGenFit::Measurement*>& meas_out, SvtxTrack* track_out,

TVector3& seed_pos,

TVector3& seed_mom, TMatrixDSym& seed_cov, const bool do_smearing = true);

https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/simulation/g4simulation/g4trackfastsim/PHG4TrackFastSim.cc#L734

10% smearing in p_{true}

root [0] 3.0/180.*3.1416
(double) 0.052360000
0.05236 radian smearing in theta and Phi

```
int PHG4TrackFastSim::PseudoPatternRecognition(const PHG4Particle* particle,
                                               std::vector<PHGenFit::Measurement*>& meas_out,
                                               SvtxTrack* track_out,
                                               TVector3& seed_pos,
                                               TVector3& seed_mom, TMatrixDSym& seed_cov, const bool do_sm
{
  assert(track_out);

  seed_cov.ResizeTo(6, 6);        Initialization default

  seed_pos.SetXYZ(0, 0, 0);
  // reset the seed resolution to the approximate position resolution of the last detector
  seed_cov[0][0] = .1 * .1;
  seed_cov[1][1] = .1 * .1;
  seed_cov[2][2] = 30 * 30;
  //  for (int i = 0; i < 3; i++)
  //  {
  //    seed_cov[i][i] = _phi_resolution * _phi_resolution;
  //  }

  seed_mom.SetXYZ(0, 0, 10);
  for (int i = 3; i < 6; i++)
  {
    seed_cov[i][i] = 10;
  }

  if (particle)                   If there is a truth information
  {
    TVector3 True_mom(particle->get_px(), particle->get_py(),
                      particle->get_pz());

    seed_mom.SetXYZ(particle->get_px(), particle->get_py(),
                    particle->get_pz());
    if (do_smearing)              Option for smearing
    {
      const double momSmear = 3. / 180. * M_PI;  // rad
      const double momMagSmear = 0.1;            // relative

      seed_mom.SetMag(
          True_mom.Mag() + gsl_ran_gaussian(m_RandomGenerator,
                                            momMagSmear * True_mom.Mag()));
      seed_mom.SetTheta(True_mom.Theta() + gsl_ran_gaussian(m_RandomGenerator, momSmear));
      seed_mom.SetPhi(True_mom.Phi() + gsl_ran_gaussian(m_RandomGenerator, momSmear));
    }
  }
```

Truth Parameters in ACTS

```
  // require minimum momentum
  const auto& p = part->getMomentum();
  const auto pmag = std::hypot(p.x, p.y, p.z);
  if (pmag * GeV < m_cfg.m_minMomentum) {
      m_log->trace("ignoring particle with p = {} GeV ", pmag);
      return nullptr;
  }

  // require minimum pseudorapidity
  const auto phi   = std::atan2(p.y, p.x);
  const auto theta = std::atan2(std::hypot(p.x, p.y), p.z);
  const auto eta   = -std::log(std::tan(theta/2));
  if (eta > m_cfg.m_maxEtaForward || eta < -std::abs(m_cfg.m_maxEtaBackward)) {
      m_log->trace("ignoring particle with Eta = {}", eta);
      return nullptr;
  }

// modify initial momentum to avoid bleeding truth to results when fit fails
const auto pinit = pmag*(1.0 + m_cfg.m_momentumSmear * m_normDist(generator));

// build some track cov matrix
Acts::BoundSymMatrix cov                    = Acts::BoundSymMatrix::Zero();
cov(Acts::eBoundLoc0, Acts::eBoundLoc0)     = 1000*um*1000*um;
cov(Acts::eBoundLoc1, Acts::eBoundLoc1)     = 1000*um*1000*um;
cov(Acts::eBoundPhi, Acts::eBoundPhi)       = 0.05*0.05;
cov(Acts::eBoundTheta, Acts::eBoundTheta)   = 0.01*0.01;
cov(Acts::eBoundQOverP, Acts::eBoundQOverP) = (0.1*0.1) / (GeV*GeV);
cov(Acts::eBoundTime, Acts::eBoundTime)     = 10.0e9*ns*10.0e9*ns;

Acts::BoundVector  params;
params(Acts::eBoundLoc0)   = 0.0 * mm ;  // cylinder radius
params(Acts::eBoundLoc1)   = 0.0 * mm ;  // cylinder length
params(Acts::eBoundPhi)    = phi;
params(Acts::eBoundTheta)  = theta;
params(Acts::eBoundQOverP) = charge / (pinit * GeV);
params(Acts::eBoundTime)   = part->getTime() * ns;
```

$(l_0, l_1, \theta, \phi, q/p)$

↓

$(d_0, z_0, \theta, \phi, q/p)$

↓ Extrapolate to Vertex

$(DCA_{xy}, DCA_z, \theta, \phi, q/p)$

I smeared momentum by 10% (Gaussian)
Theta and Phi has to be smeared

https://github.com/eic/EICrecon/blob/main/src/algorithms/tracking/TrackParamTruthInit.cc

Track fitting with Realistic Parameters

$(l_0, l_1, \theta, \phi, q/p)$

- Simulate a Single Track (pi+, pi-, p, etc.)
- At the reconstruction just consider primary hits using quality flag
- Sort these hits w.r.t. time
- Initialize: l_0 , l_1, θ, ϕ from the last hit (If propagating backwards)
- Use last three hits (Sorted Hits) to estimate radius of track (p_T = 0.3*B*R)
- Initialize: p = p_TCoshη
- This will give the realistic estimation of single track performances (momentum, theta, phi, DCA resolutions) which can be compared with the results from truth seeding

It can be implemented very quickly but drawbacks: no background and no combinatorics which is ongoing work

Summary & Future Plan

- Smearing in Theta and Phi will be required the correct distribution
- Track fitting with realistic parameters: idea explained can be useful and can be compared with truth seeding
-

Backup

https://github.com/eic/EICrecon/issues/215

DraTeots commented on Oct 13, 2022 — Member

On behalf of **@Simple-Shyam**. Original message:

```
It's giving an error. Can you please change the line in the correct format, the below line is giving an error.

    const auto phi_smear = phi + 0.20*gRandom->Gaus(0.,phi);
    const auto theta_smear = theta + 0.20*gRandom->Gaus(0.,theta);
    const auto pmag_true = pmag * GeV
    const auto pmag_smear = pmag_true + 0.20*gRandom->Gaus(0.,pmag_true);

I want to pass them like below

    Acts::BoundVector  params;
    params(Acts::eBoundLoc0)   = 0.0 * mm ;  // cylinder radius
    params(Acts::eBoundLoc1)   = 0.0 * mm ;  // cylinder length
    params(Acts::eBoundPhi)    = phi_smear;
    params(Acts::eBoundTheta)  = theta_smear;
    params(Acts::eBoundQOverP) = charge / (pmag_smear);
    params(Acts::eBoundTime)   = part->getTime() * ns;
```

The issue is actually:

- ☐ update TrackParamTruthInitConfig and add smearing values (those 0.20)
- ☐ update TrackParamTruthInit_factory to add Smearing parameters so users can modify them

Simple-Shyam commented on Oct 13, 2022 — Member

Thanks, let me state the problem, the main issue is when we are fitting in ACTs, we need to pass parameters, which we are currently passing true parameters from MC. If you reconstruct, you will get true momentum (fitter is biased). We need to smear the initial true parameters so that fitter tries to minimize them. I put an example (just an example), we should smear with some numbers after discussion, how much smear we can put in the true parameters. We can follow similar as fun4all for smearing linked below.

https://github.com/sPHENIX-Collaboration/coresoftware/blob/master/simulation/g4simulation/g4trackfastsim/PHG4TrackFastSim.cc

I informed truth smearing issue in October