Propagator Bug or Unintended Use Case?

Joe Osborn

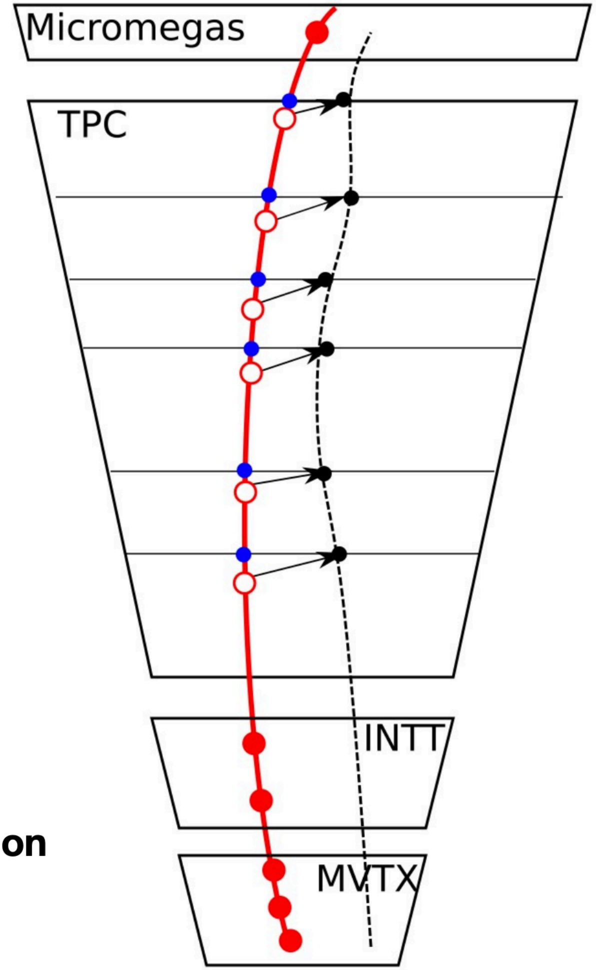
Oak Ridge National Laboratory

November 17, 2020

ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

Quick Overview

- Working on implementing space charge distortion calibrations into sPHENIX
- TPC suffers from space charge distortions which alter true measurement position by O(mm)
- To determine the (average) distortion, we fit silicon+micromegas measurements to get an estimate of the trajectory and then determine the TPC measurement residuals to that trajectory



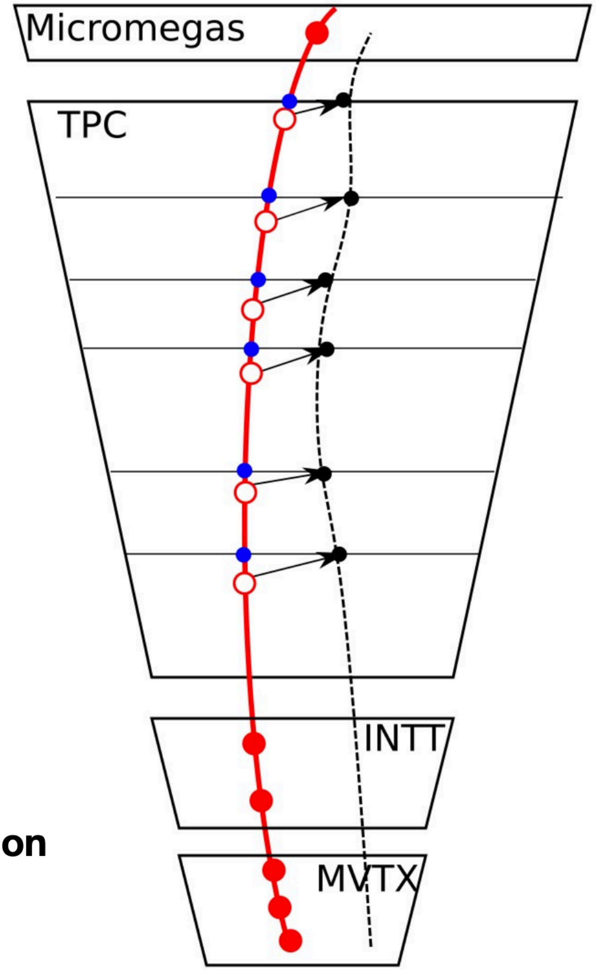
2 layers of micromegas

48 layers of TPC

5 layers of silicon

Quick Overview

- To determine the residuals of the TPC measurements with the trajectory, we fit the silicon+Micromegas with the Acts::DirectNavigator
- Take resulting track parameters and propagate them to the 48 TPC surfaces to determine where measurement "should" have been along trajectory
- Then calculate residuals between track propagated state and actual measurement



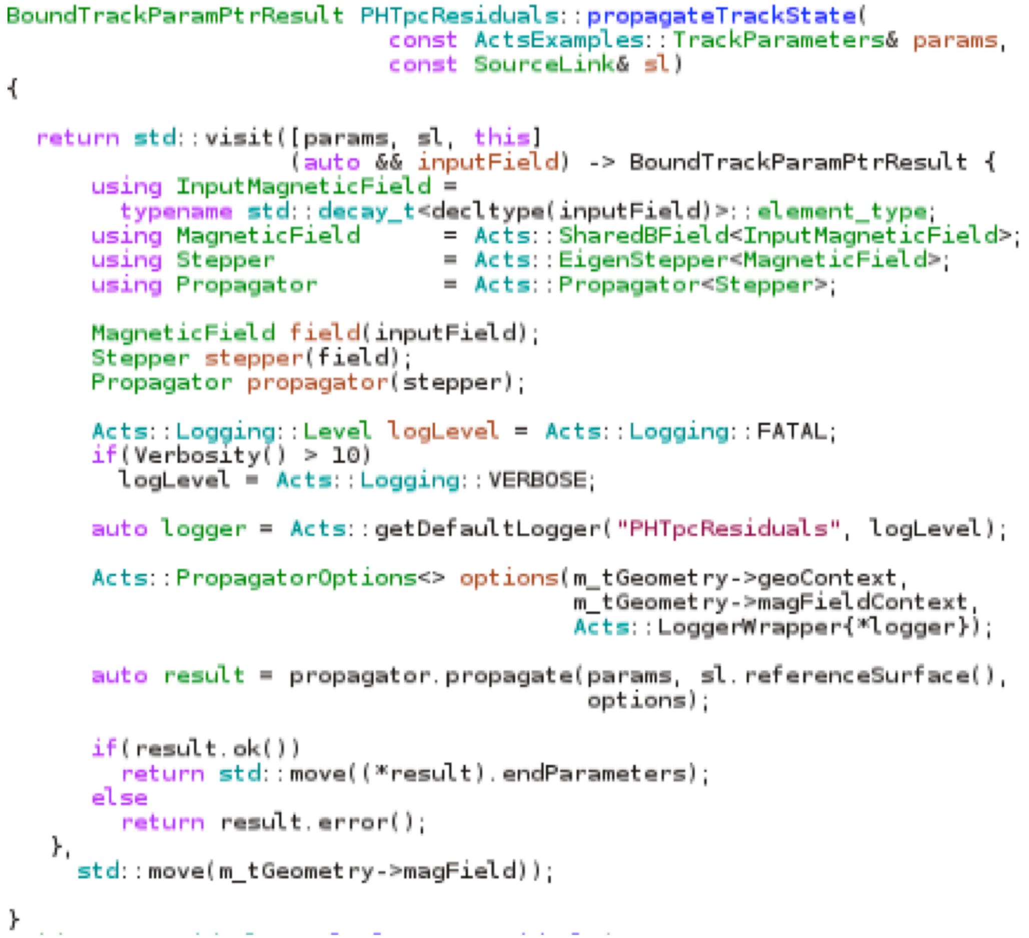
2 layers of micromegas

48 layers of TPC

5 layers of silicon

Bug?

- Loosely followed idea from ImpactPointEstimator
- Create basic propagator, take silicon+MM fit parameters to propagate to surface that measured SourceLink exists on to determine track parameters on that surface
- This generally works, but when running large jobs many fail with a fatal crash that stops the whole job

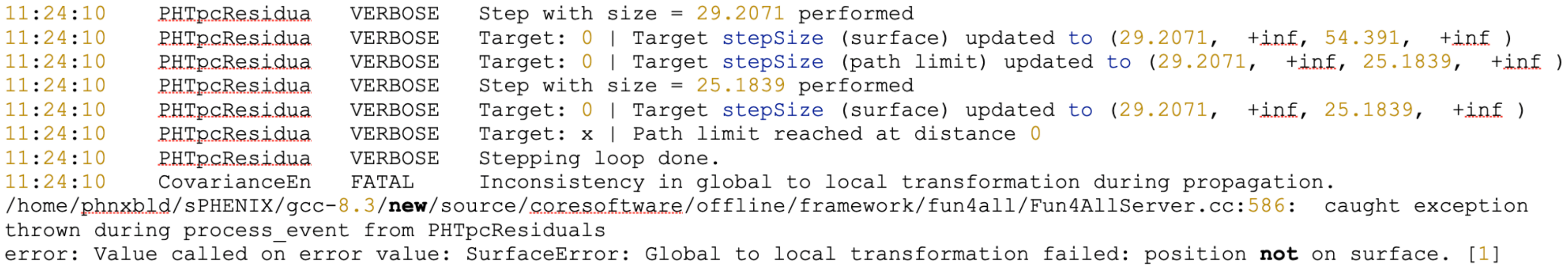


```
BoundTrackParamPtrResult PHTpcResiduals::propagateTrackState(
                              const ActsExamples::TrackParameters& params,
                              const SourceLink& sl)
{

  return std::visit([params, sl, this]
                    (auto && inputField) -> BoundTrackParamPtrResult {
      using InputMagneticField =
        typename std::decay_t<decltype(inputField)>::element_type;
      using MagneticField      = Acts::SharedBField<InputMagneticField>;
      using Stepper            = Acts::EigenStepper<MagneticField>;
      using Propagator         = Acts::Propagator<Stepper>;

      MagneticField field(inputField);
      Stepper stepper(field);
      Propagator propagator(stepper);

      Acts::Logging::Level logLevel = Acts::Logging::FATAL;
      if(Verbosity() > 10)
        logLevel = Acts::Logging::VERBOSE;

      auto logger = Acts::getDefaultLogger("PHTpcResiduals", logLevel);

      Acts::PropagatorOptions<> options(m_tGeometry->geoContext,
                                        m_tGeometry->magFieldContext,
                                        Acts::LoggerWrapper{*logger});

      auto result = propagator.propagate(params, sl.referenceSurface(),
                                         options);

      if(result.ok())
        return std::move((*result).endParameters);
      else
        return result.error();
    },
     std::move(m_tGeometry->magField));

}
```

Bug?

- Sometimes the propagator will propagate and never reach the surface, quitting after reaching the 1000 step limit
  - That is fine, as it returns an Acts::Result that is an error and we can just skip this one and go to the next propagation
  - This is typically related to a bad initial silicon+MM fit - e.g. if the starting track parameters are bad, then the surface may never be reached
- However, sometimes this leads to a fatal crash

Bug?



```
11:24:10     PHTpcResidua   VERBOSE   Step with size = 29.2071 performed
11:24:10     PHTpcResidua   VERBOSE   Target: 0 | Target stepSize (surface) updated to (29.2071,  +inf, 54.391,  +inf )
11:24:10     PHTpcResidua   VERBOSE   Target: 0 | Target stepSize (path limit) updated to (29.2071,  +inf, 25.1839,  +inf )
11:24:10     PHTpcResidua   VERBOSE   Step with size = 25.1839 performed
11:24:10     PHTpcResidua   VERBOSE   Target: 0 | Target stepSize (surface) updated to (29.2071,  +inf, 25.1839,  +inf )
11:24:10     PHTpcResidua   VERBOSE   Target: x | Path limit reached at distance 0
11:24:10     PHTpcResidua   VERBOSE   Stepping loop done.
11:24:10     CovarianceEn   FATAL     Inconsistency in global to local transformation during propagation.
/home/phnxbld/sPHENIX/gcc-8.3/new/source/coresoftware/offline/framework/fun4all/Fun4AllServer.cc:586:  caught exception
thrown during process_event from PHTpcResiduals
error: Value called on error value: SurfaceError: Global to local transformation failed: position not on surface. [1]
```

- Verbose output gives this
- Looking through the code, it appears that the aborter believes that the surface has been reached within a tolerance of 0
- However, when the surface tries to perform a global to local transformation, it fails as the position is not on the surface and then this exception is thrown

Bug?

- I see the same error when doing track projections to the calorimeters
- We project final track fits to the calorimeter radii to help with electron id etc.
- Do this by creating cylindrical surfaces at the calorimeter radii, then (essentially) running the same propagator code I showed before to the cylindrical surfaces
- Usually works, but there are instances where I get this same crash feature

Bug?

- Is this a bug, or is it a cause of an unintended use case that I am giving Acts?
  - Note - we are running v1.1.0 (need to update). However, current master branch aborter looks similar to v1.1.0
- Regardless, there are two solutions
  - Fix bug and/or my unintended use case
  - Force propagator to return an Acts::Result that contains an error message when this happens (maybe that was intended and I found an exceptional case), rather than allowing code to continue and ultimately crash in this logic scenario
    - This would allow user to catch the exception/error, and handle it themselves (e.g. in our case we would just skip it and move to the next propagation)