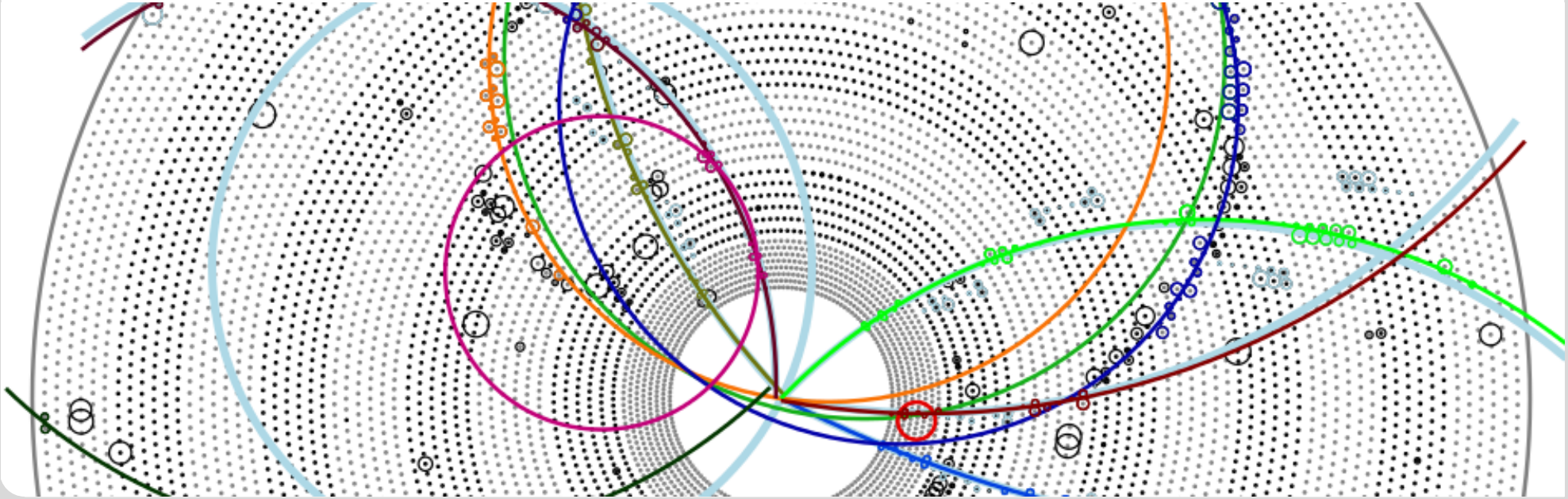
Karlsruher Institut für Technologie

T_0-Estimation using CDC Drift Circles.

Weekly Tracking Meeting.

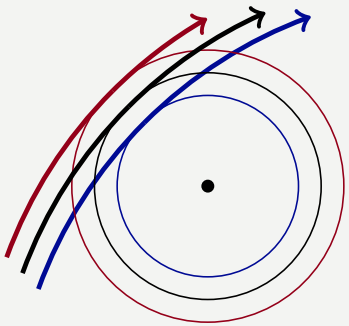
Nils Braun | DATE

IEKP - KIT



Recap from Tobias Schlüter

What Does a Drift Chamber Measure?



Measurement Procedure

- passing charged particle ionizes gas
- gas cloud collapses on wire
- difference
  T(passage of particle) – T(collapse)
  gives distance of passage

How do we know the passage time?

Usually:

- Starting time of the track is evaluated
- T(Passage) =Track Length / Velocity

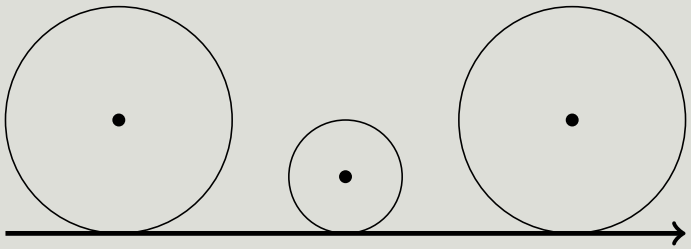
A drift chamber is a device that measures time, positions are inferred.

Recap from Tobias Schlüter

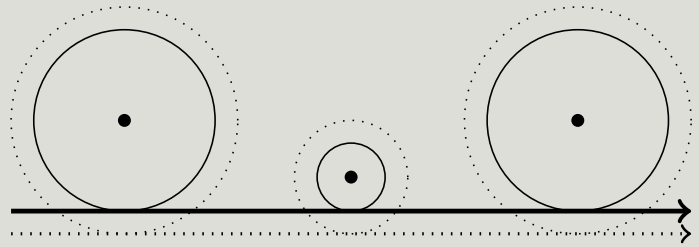
What Happens if the Time is Badly Known?

The simplest case, straight lines, all hits on one side.

Correct Timing



Passage Time Underestimated



- position measurement depends on the evaluated drift time
- In this simple case a bias in time leads to bias in position.

Work to be done

Work performed already by Tobias:

- Write out full covariance matrices in track fitting.
- Build module to calculate the extracted time (after arXiv:0810.2241).
- Some (preliminary) validation and tests.

What is still needed:

- Include the algorithm into the whole tracking data flow.
- Study the impact of a wrong track time onto track finding and fitting.
- Find reasonable settings for the track finders and the time extrapolation (e.g. only use some tracks for timing)
- Test the module in the CDC-Cosmics test end of this year.

Some first work (performed this week)

- Refactored the module to use `RecoTracks` (done already before).
- Extracted the functionality into own class; can be used from python.
- Written a python harvester-based validation and some jupyter notebooks for analysis.
- Included a loop in the module to fit and extract times until a certain optimization condition is fulfilled.
- Write a simulation steering file to include time shift during simulation (see next slide)

How to shift the simulation in time

Just for reference: Include this module after `FullSim` but before `CDCDigitizer`

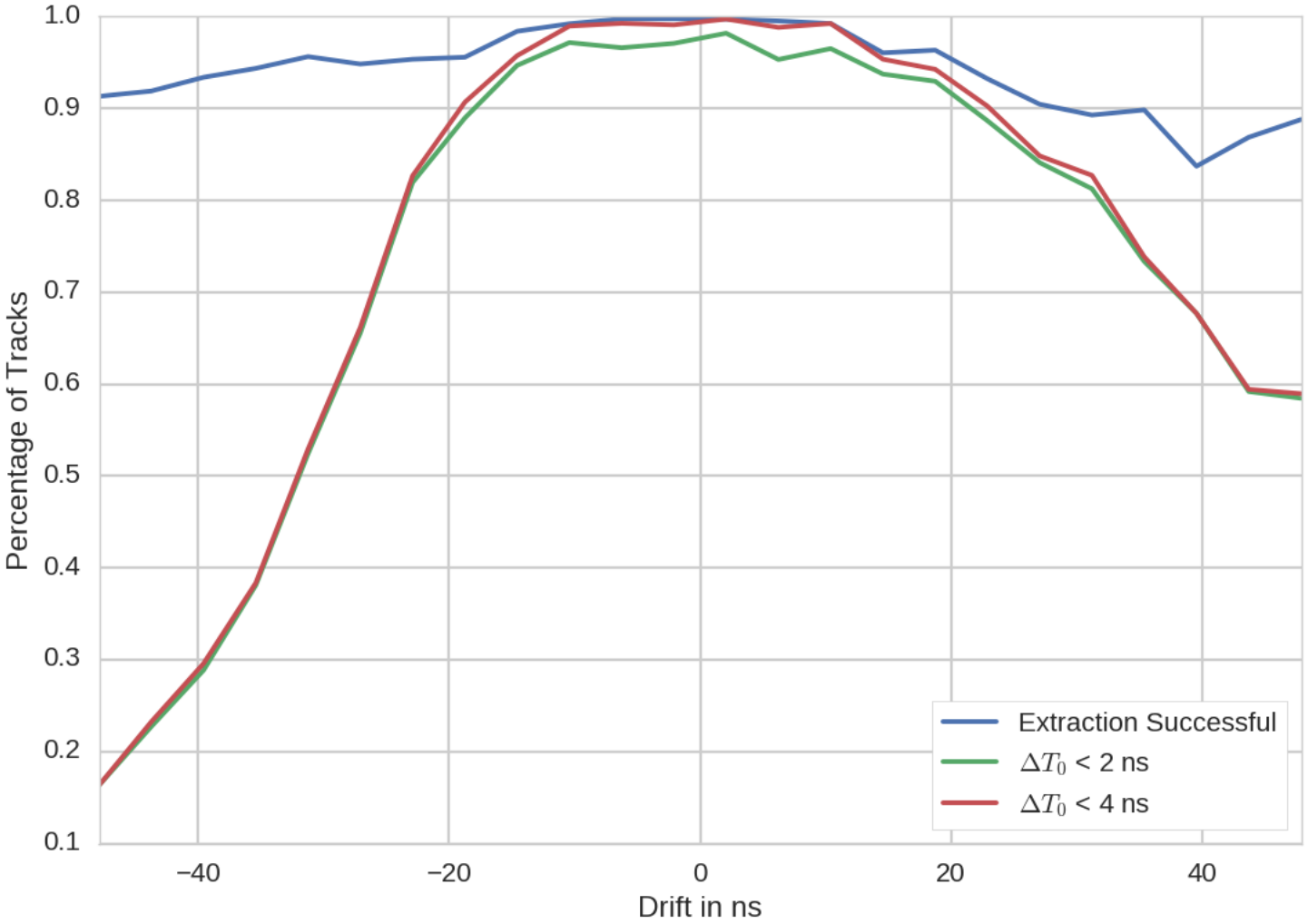
```
class RandomizeTrackTimeModule(basf2.Module):
  def __init__(self, mc_drift):
      self.mc_drift = mc_drift
      basf2.Module.__init__(self)
  def event(self):
      cdc_sim_hits = Belle2.PyStoreArray("CDCSimHits")

      # Special mode: random drift
      if self.mc_drift == -999:
          mc_drift = np.random.randint(-50, 50)
      # Normal mode: shift as module parameter
      else:
          mc_drift = self.mc_drift

      for cdc_sim_hit in cdc_sim_hits:
          cdc_sim_hit.setFlightTime(cdc_sim_hit.getFlightTime() + float(mc_drift))

      # For later reference: add the shift to the mc production time
      mc_particles = Belle2.PyStoreArray("MCParticles")
      for mc_particle in mc_particles:
          mc_particle.setProductionTime(mc_particle.getProductionTime() +
                                        mc_drift)
```

First results: MC Track Finder on Particle Gun (one high-p_T Track)



First results: MC Track Finder on Particle Gun (one high-p_T Track)

- Adjusted the `RealisticTDCCountTranslator`:

```
double driftL_old = (driftTime >= 0.) ?
    m_cdcp.getDriftLength(driftTime, layer, leftRight,
                          alpha, theta) :
    m_vFactor * driftTime;
double driftL_new = std::copysign(
    m_cdcp.getDriftLength(fabs(driftTime), layer,
                          leftRight, alpha, theta),
                          driftTime);
```

- Median (and mean) stay around 0 ns (a bit under 0 ns) for most of the drift values.
- Standard deviation is not a good measure because of outliers; interquartil range below 1 ns in $[-20\,\mathrm{ns}, 40\,\mathrm{ns}]$.
- L1-Time jitter is expected to be around 10 ns for hadronic events and 20 ns for low-multiplicity events.