Update on Track Reconstruction for LDMX trackers

PF, Omar, Tom
with help and support by Paul Gessinger-Befurt

25/01/2022

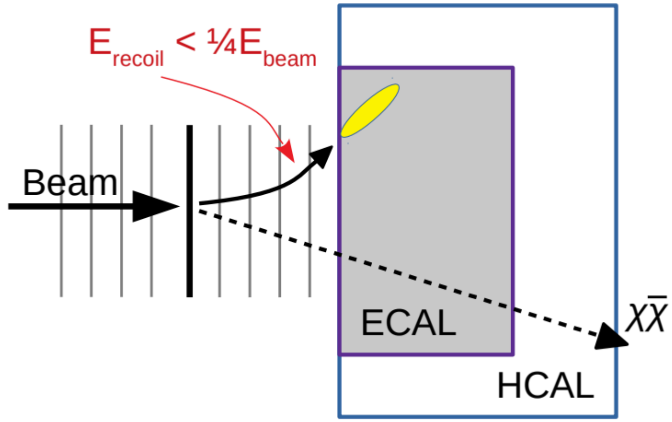
- not as bad as Tim's talk -

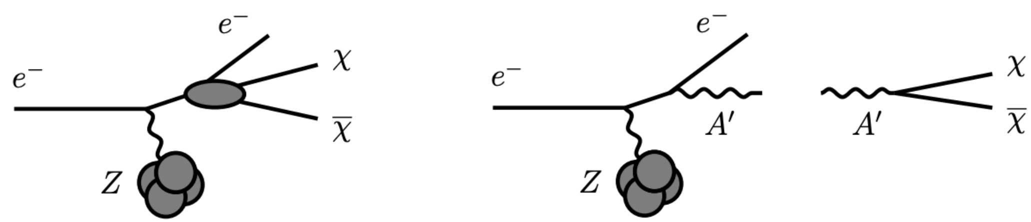
U.S. DEPARTMENT OF ENERGY | Stanford University

SLAC NATIONAL ACCELERATOR LABORATORY

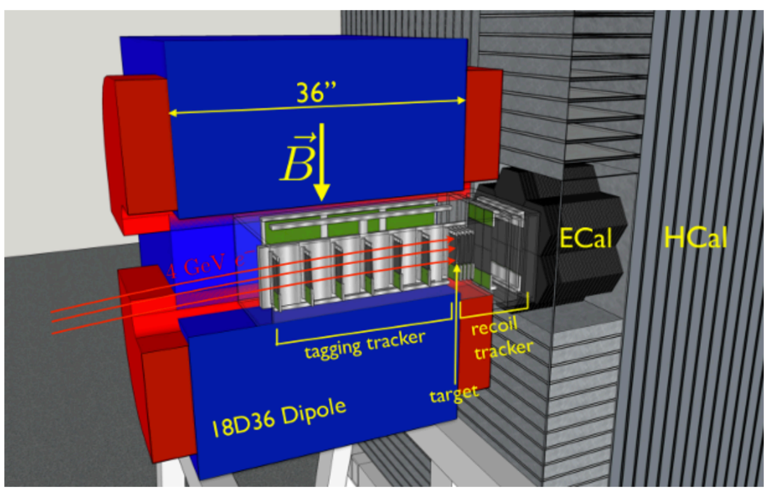
The LDMX experiment

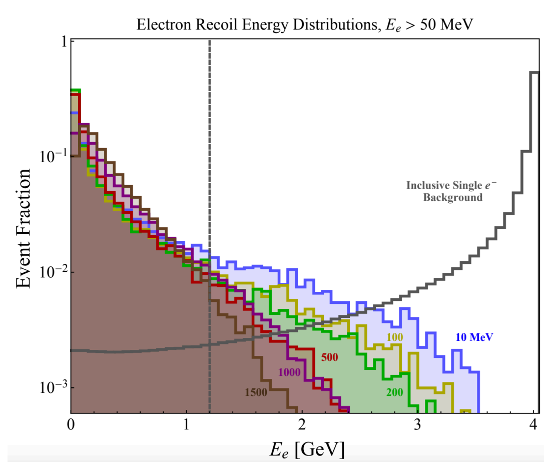
- High-luminosity measurement of missing momentum in multi-GeV electron fixed target collisions, through both direct dark matter and mediator particle production.
- This measurement would provide broad sensitivity to dark matter interactions over the entire sub-GeV mass range or visibly decaying dark photons, axions, dark higgs…

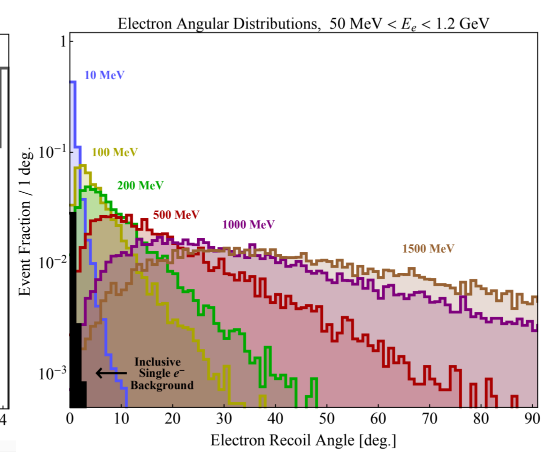




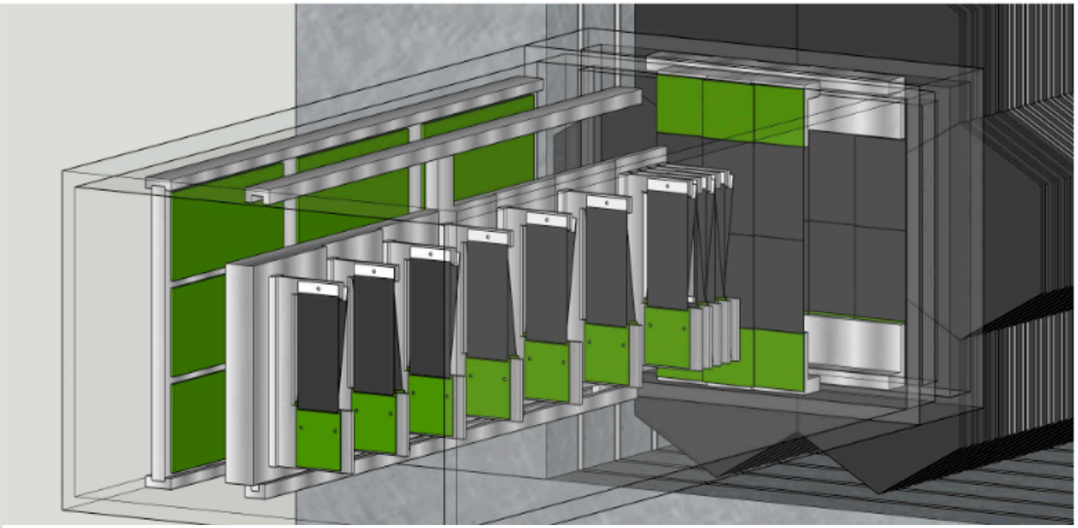
At the level of 10^16 incident e-, large number of e with low E, are consistent with signal recoils
Tagger tracker measures the trajectories to veto recoils originating from beam impurities







The LDMX Tracking systems



Tagger Tracker: 7 layers SCT-like double strip-sensors. Alternating stereo-angle orientations. Placed at different offsets wrt beam-axis to follow bent electron trajectory

Recoil Tracker: 4 SCT-like double strip-sensors and 2 single side strip-sensors. 10 sensors per module glued back to back. Placed in the dipole fringe field

TABLE I: The layout and resolution of the tagging tracker.

Layer	1	2	3	4	5	6	7
z-position, relative to target (mm)	-607.5	-507.5	-407.5	-307.5	-207.5	-107.5	-7.5
Stereo Angle (mrad)	-100	100	-100	100	-100	100	-100
Bend plane (horizontal) resolution (μm)	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6
Non-bend (vertical) resolution (μm)	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60

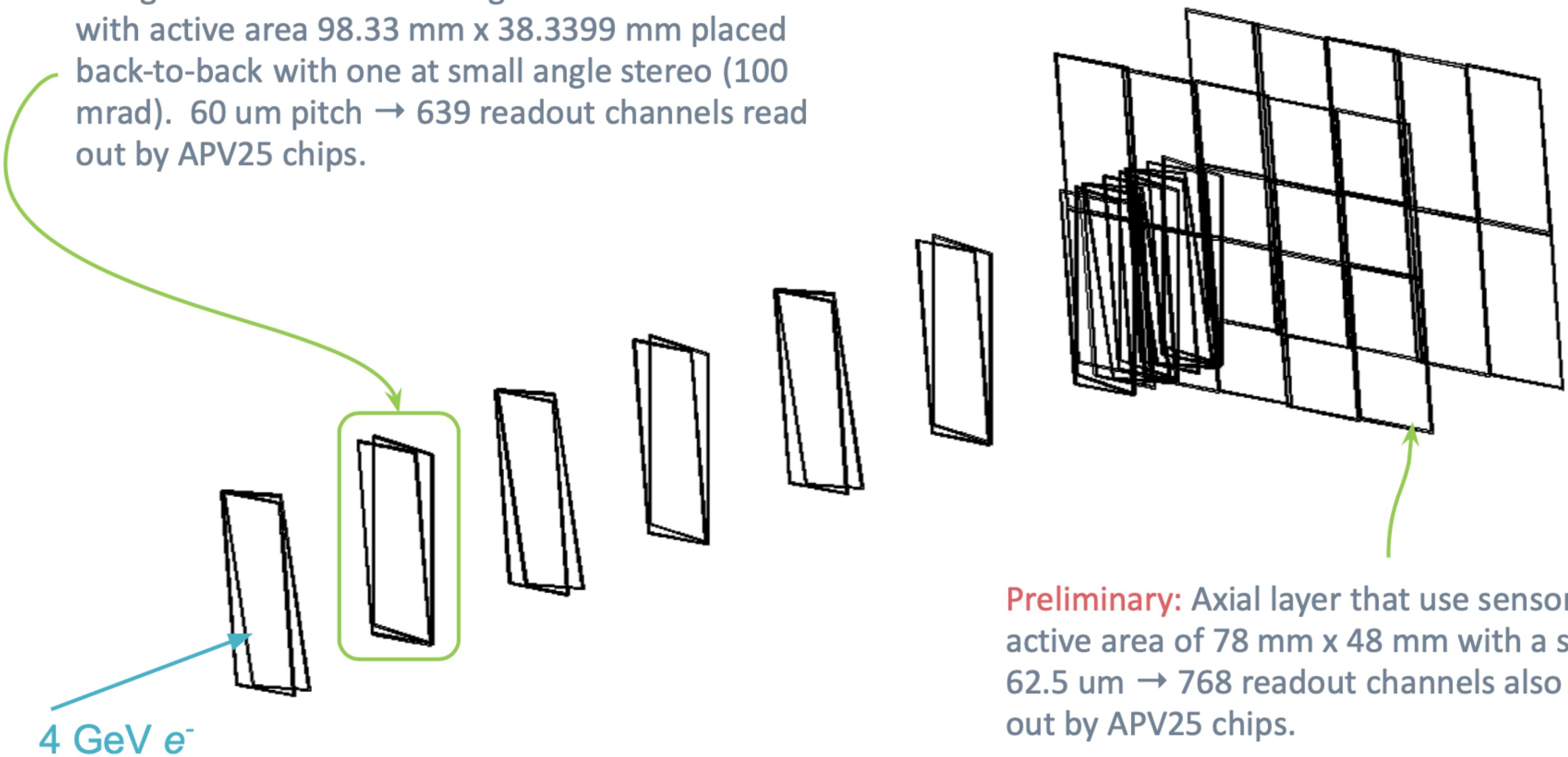
TABLE II: The layout and resolution of the recoil tracker.

Layer	1	2	3	4	5	6
z-position, relative to target (mm)	+7.5	+22.5	+37.5	+52.5	+90	+180
Stereo Angle (mrad)	100	-100	100	-100	-	-
Bend plane (horizontal) resolution (μm)	$\approx$6	$\approx$6	$\approx$6	$\approx$6	$\approx$6	$\approx$6
Non-bend (vertical) resolution (μm)	$\approx$60	$\approx$60	$\approx$60	$\approx$60	-	-

The LDMX Tracker system

Tagger and Recoil Tracker Geometry

Using the same module design as HPS: 2 sensors with active area 98.33 mm x 38.3399 mm placed back-to-back with one at small angle stereo (100 mrad). 60 um pitch → 639 readout channels read out by APV25 chips.



Preliminary: Axial layer that use sensors with an active area of 78 mm x 48 mm with a strip pitch of 62.5 um → 768 readout channels also being read out by APV25 chips.

Outline

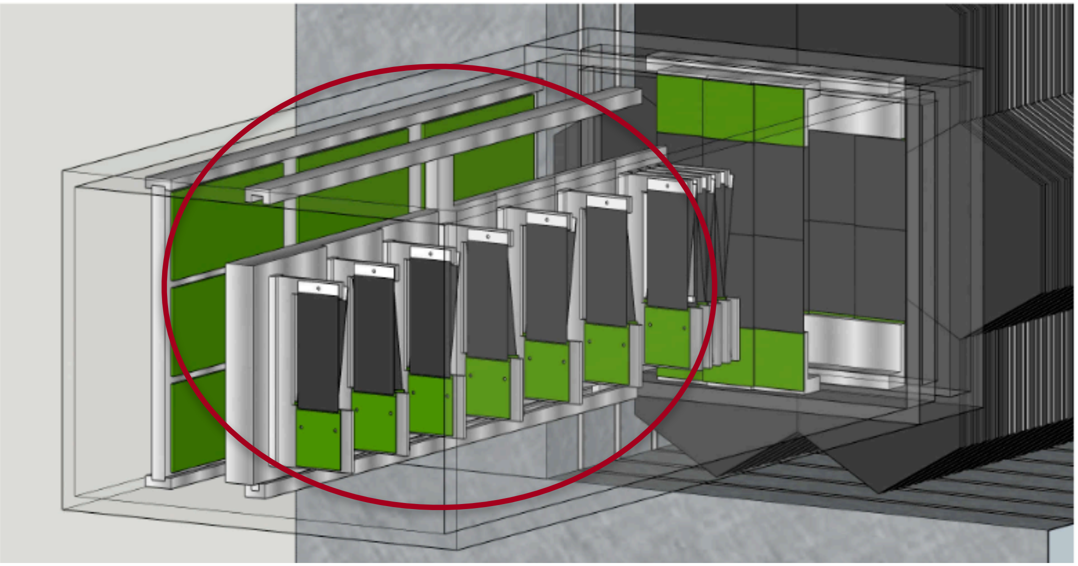
- **Charged particles trajectory reconstruction in the Tagger and Recoil trackers** is an important missing piece of the ldmx-sw framework
- **Today:**
  - General intro to ACTS tracking framework and integration in ldmx-sw
  - First applications to LDMX:
    - **Seed finding** from 3D simulated space points
    - Implementation of **tracking geometry** from DD4hep detectors
    - **Track propagation** in non uniform B field
    - **Combinatorial Kalman Filter track finding and fitting**

Basic tracking procedure

- Outlined here are the steps of track reconstruction for LDMX.
- Necessary ingredients are
  - **Write the tracking geometry** as a support for track propagation, track finding and fitting
  - Form **external 3D (2D) space points** from raw data clusters.
  - **Provide the 3D (2D) space points to the seed finder tools** in order to find track seeds.
  - Provide the hit collection, the initial seed track parameters and the tracking geometry to the **combinatorial kalman filter** for track finding and fitting.

- Space point formation
- Seed finding
- Track candidates (Combinatorial Kalman Filter)
- Ambiguity solver
- Tagger - Recoil matching

The LDMX Tracking systems

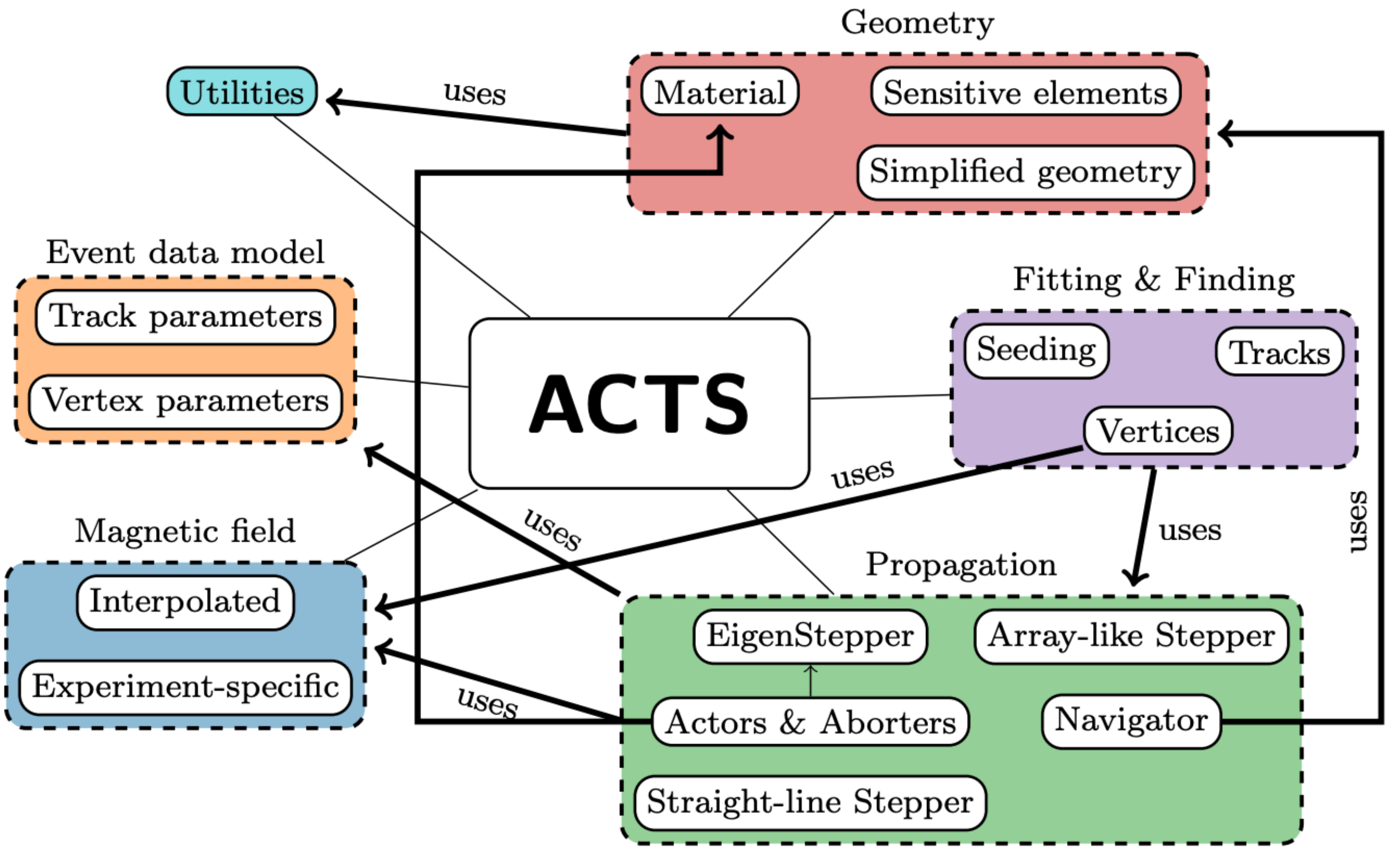


CKF tracking tests are performed in the tagger tracker as recoil tracking is still under testing and devel

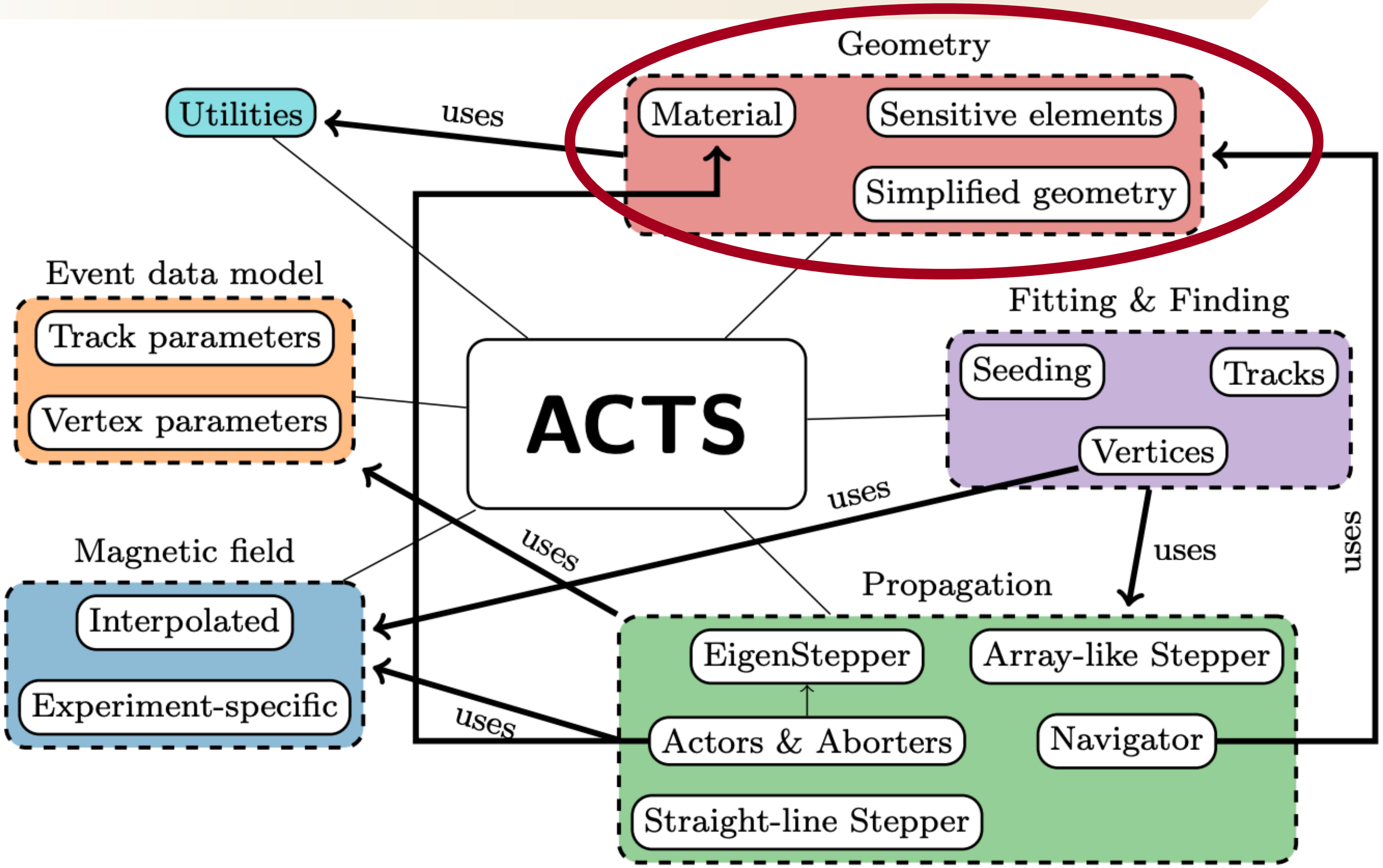
TABLE I: The layout and resolution of the tagging tracker.

Layer	1	2	3	4	5	6	7
z-position, relative to target (mm)	-607.5	-507.5	-407.5	-307.5	-207.5	-107.5	-7.5
Stereo Angle (mrad)	-100	100	-100	100	-100	100	-100
Bend plane (horizontal) resolution (μm)	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6	$\sim$6
Non-bend (vertical) resolution (μm)	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60	$\sim$60

ACTS For LDMX - Library features

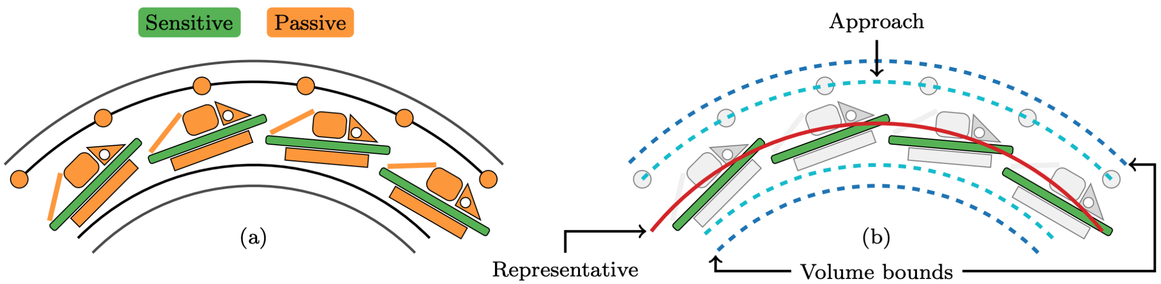


ACTS For LDMX - Library features



ACTS For LDMX - Tracking Geometry Maker

arxiv 2106.13593



- TGeo, DD4Hep or GeoModel can be used to generate the tracking Geometry
- Tagger Tracker, Target and Recoil Tracker host a small amount of sensors
  - ACTS can be configured to generate the needed geometry holding the transformations
- Initialized place holders for
  - Time dependent and contextual geometry, calibration and bField information

Collect the DD4Hep World volume from detector instance

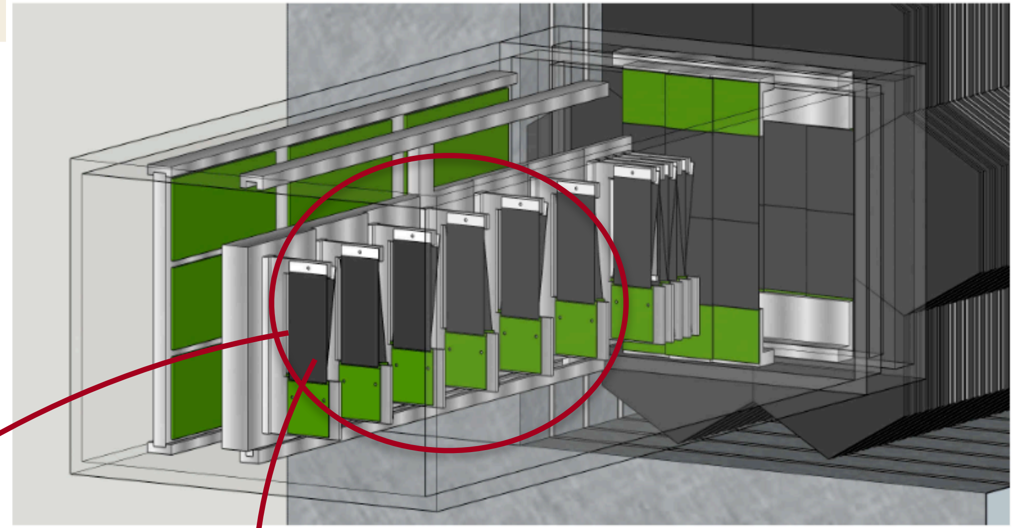
Retrieve Tagger/Recoil sub detectors

Form sensitive surfaces with material information

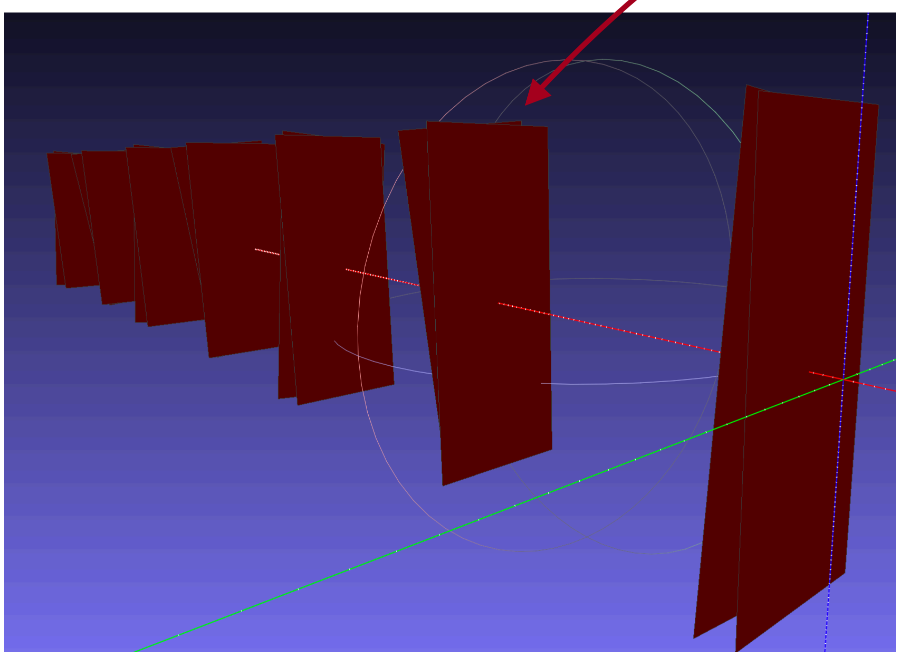
Assign transformations from the dd4hep detector elements

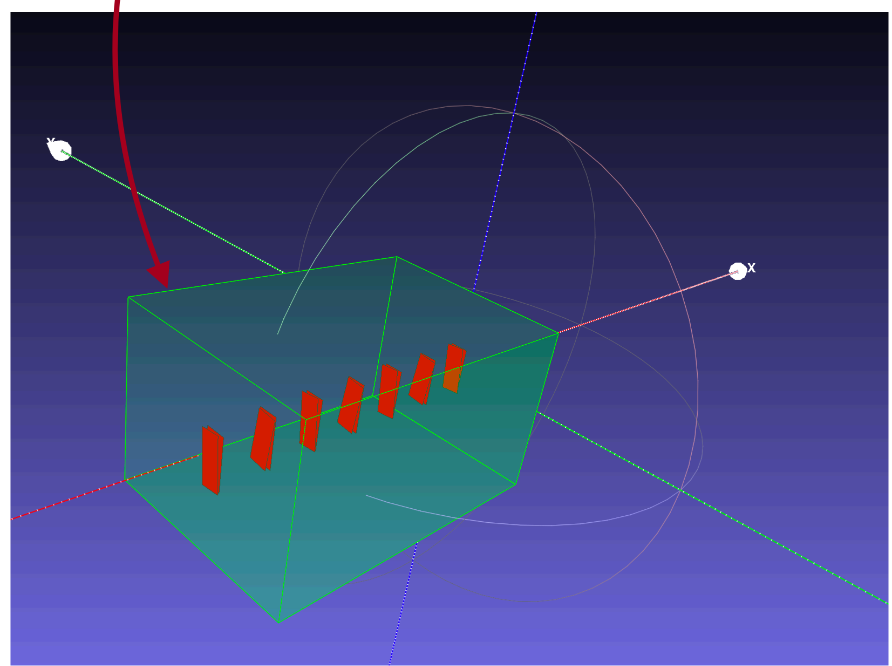
Pass the information to ACTS tracking geometry builder

ACTS For LDMX - Tracking Geometry



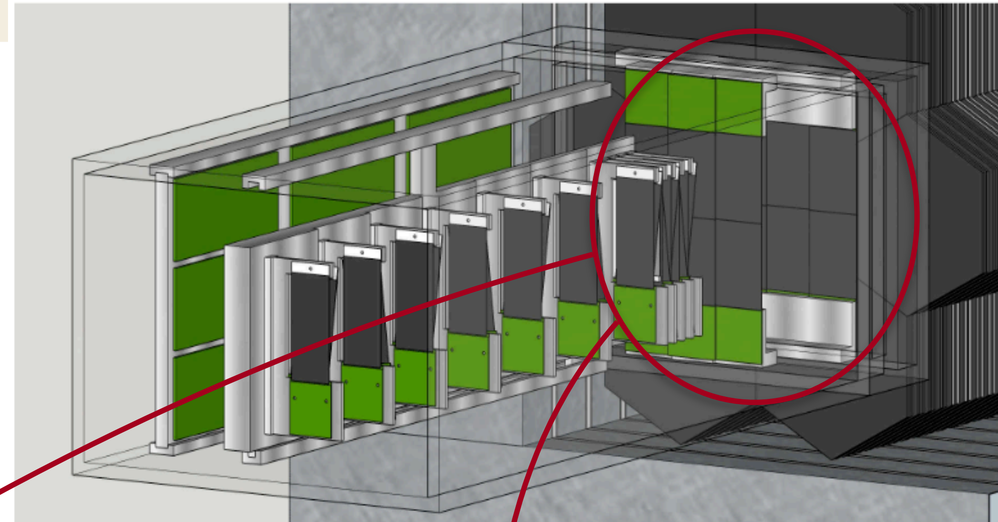
- Successfully constructed the tracking geometry for the Tagger detector
  - Used CuboidVolumeBuilder factory (some more in next slide)
- Axes are rotated to have bField along Z
  - Seems un-avoidable due to θ angle definition even if CKF algorithms can use the BField Vector
- Tracking tests are done in the Tagger tracker only so far

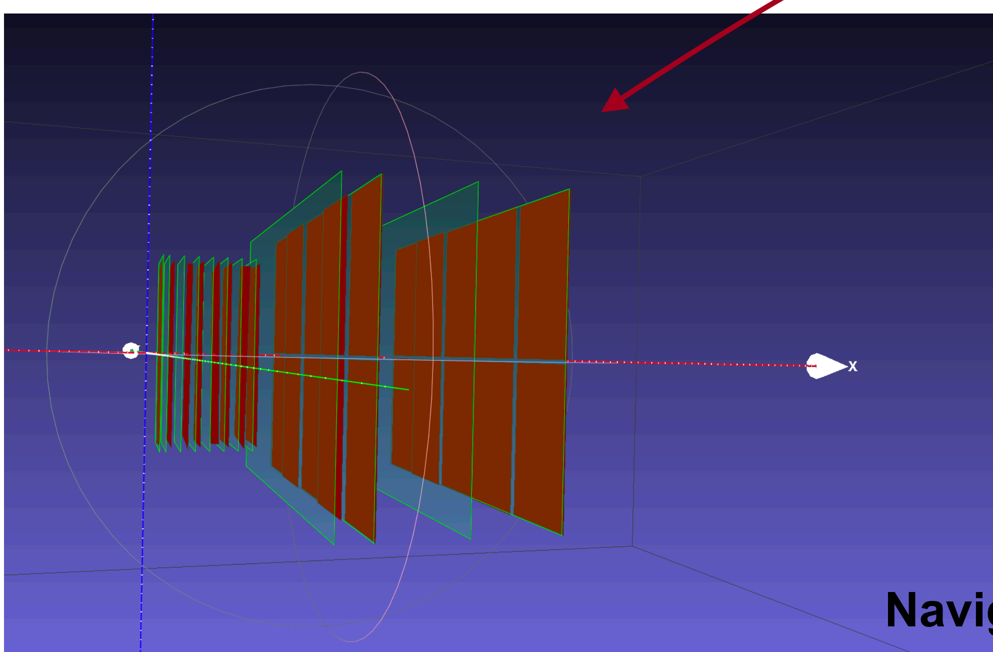


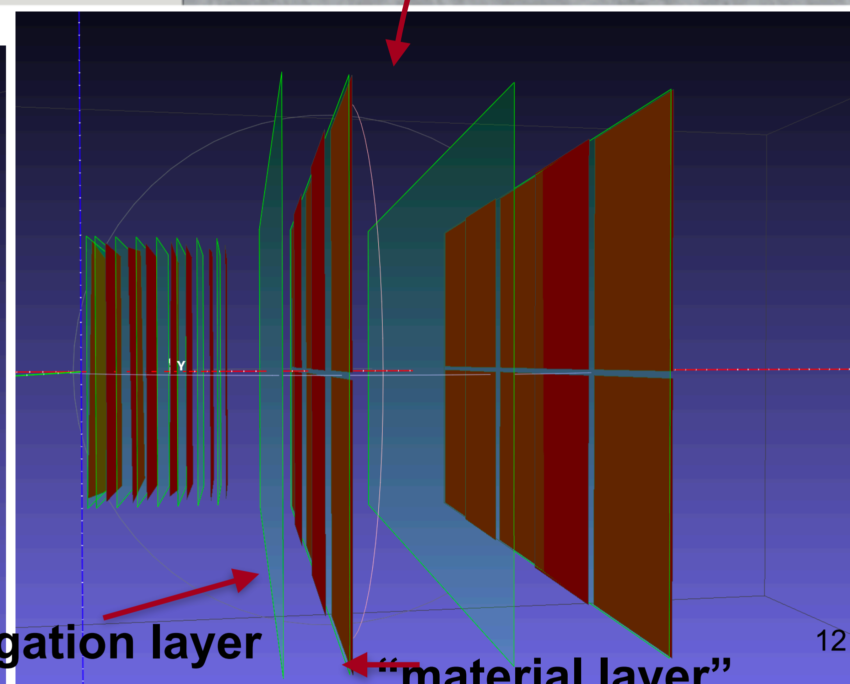


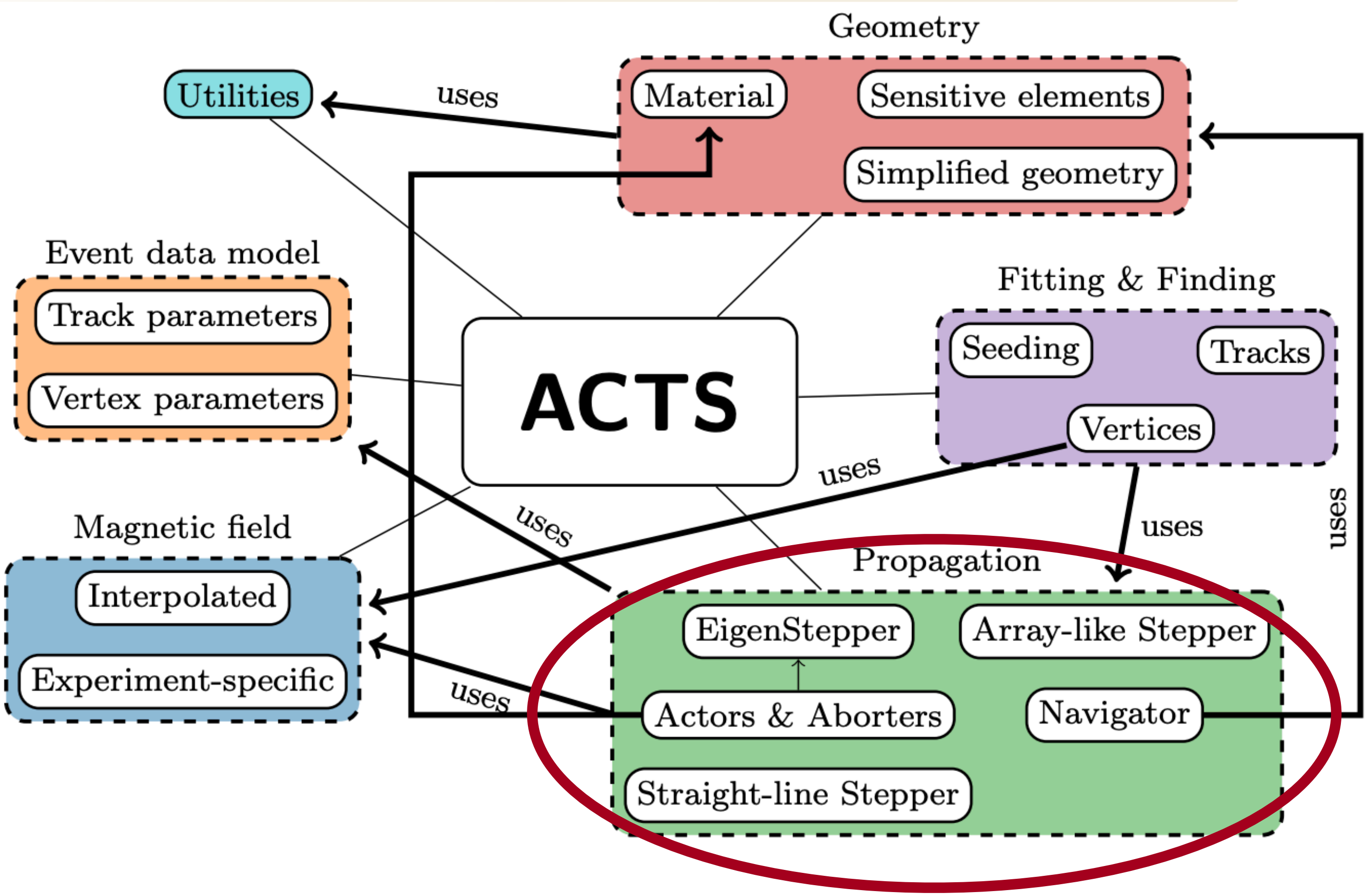
ACTS For LDMX - Tracking Geometry

- Modified version of acts CuboidVolumeBuilder (PR12 for Paul https://github.com/paulgessinger/acts/pull/12) that allows to build telescope geometries specifying more surfaces per layer (in the case of LDMX, recoil the multiple-sensor single-sided module was breaking the tracking-geometry checks)
- Navigation layers are interleaved between “material layers”. Change the configuration to allow the user to decide if the material layer is wanted or not?








Geometry
Material
Sensitive elements
Simplified geometry
Utilities
uses
Event data model
Track parameters
Vertex parameters
ACTS
Fitting & Finding
Seeding
Tracks
Vertices
uses
uses
uses
uses
Magnetic field
Interpolated
Experiment-specific
Propagation
EigenStepper
Array-like Stepper
Actors & Aborters
Navigator
Straight-line Stepper
uses

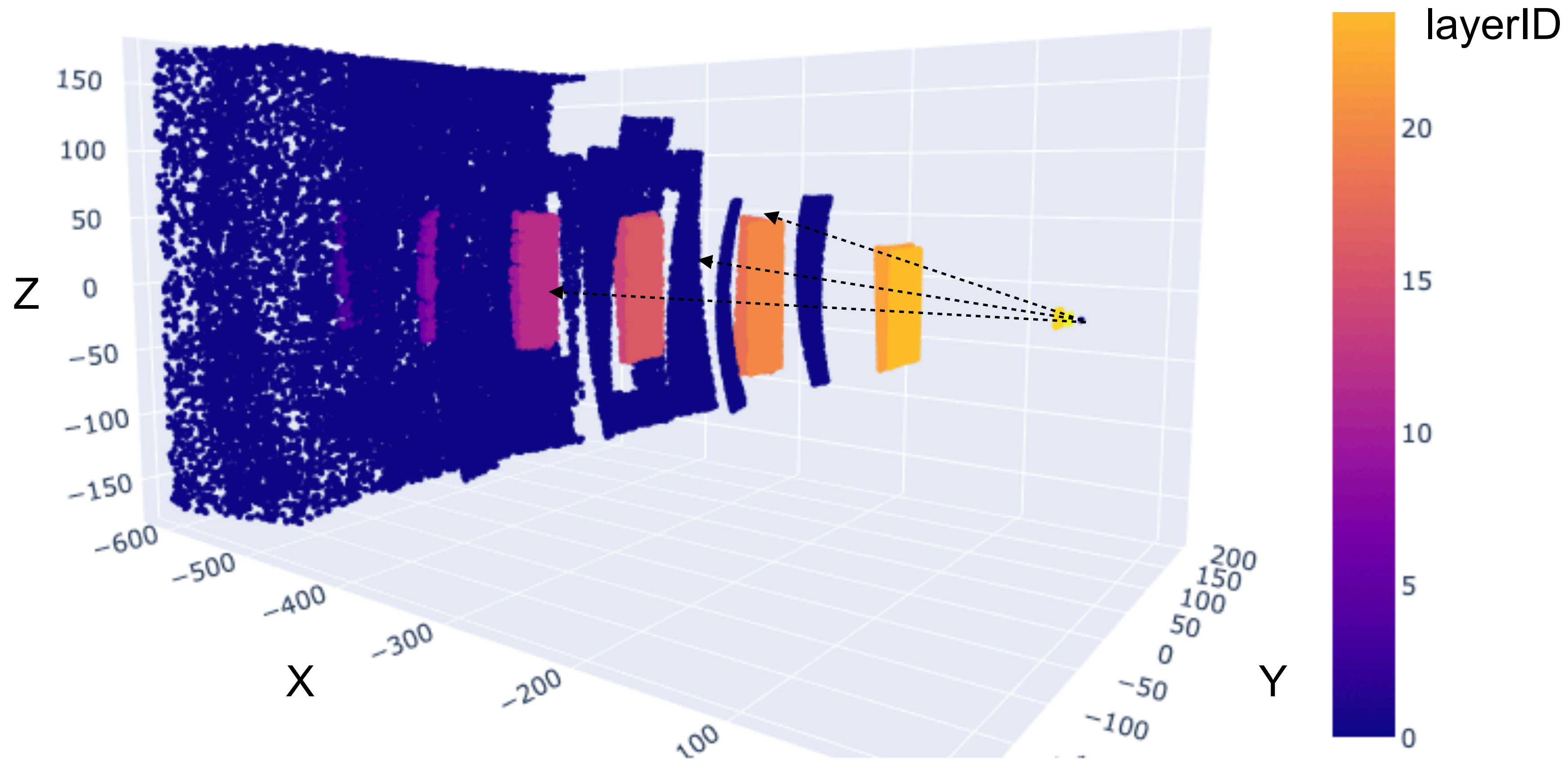
ACTS For LDMX - Tracks propagation

- The tracking geometry is at the basis for track states propagation using RK integrator
- Two components form the track state propagation:
  - **Navigator**: which **resolves the various sensitive/ approach layers** defined in the tracking geometry
  - **Stepper**: the actual **mathematical implementation of the transport of the track parameters and covariance matrix** from a point to another, taking into account traversed material
- Both component ported and interfaced to ldmx-sw

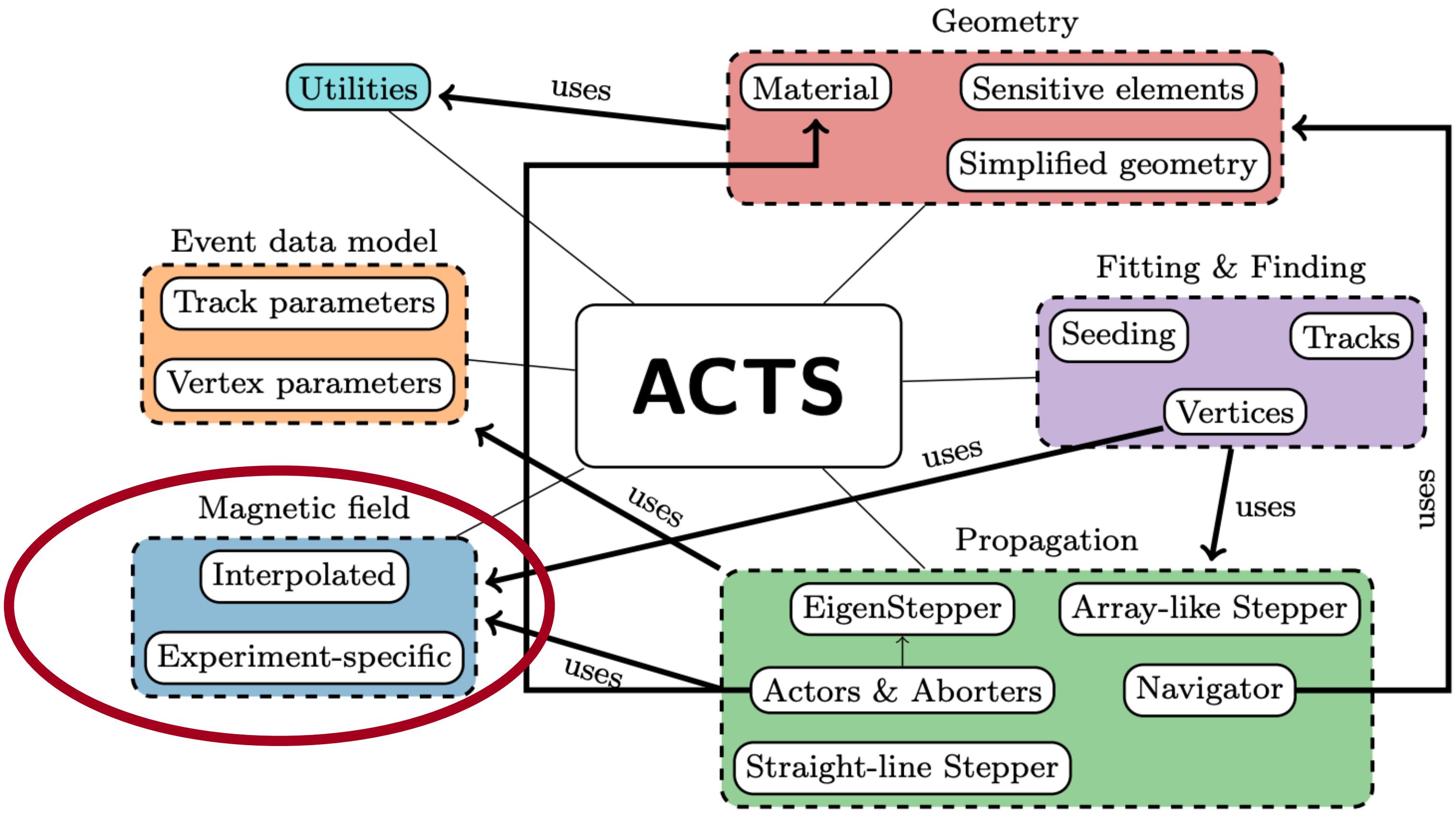
ACTS For LDMX - Tracking Geometry

SLAC

- 3D view shows the sensitive layers in space (coloured points)
- Blue points are intermediate states if no intersections are found (just for visualisation)

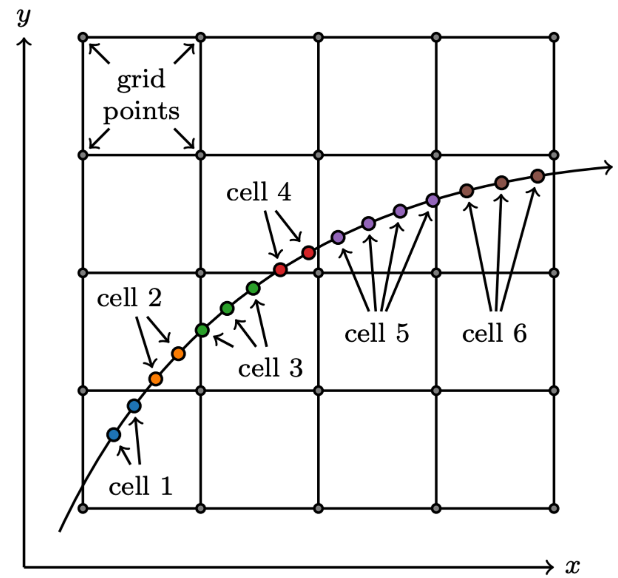


ACTS For LDMX - Library features



ACTS For LDMX - Interpolated Magnetic Field map

- Magnetic field is loaded into ldmx-sw tracking via dedicated ACTS interpolated provider:
  - Magnetic field is **retrieved multiple times** from cached cell along the propagation
  - Possibility to get both field and gradient to adapt the RK step
- Field map is loaded in global LDMX coordinates: **two** functions are defined to **transforms from tracking to map space and vice versa**




```
101 29 601
1 X(mm)
2 Y(mm)
3 Z(mm)
4 BX(1000T)
5 BY(1000T)
6 BZ(1000T)
0 End of Header. Data follows in above format
-250.0 -70.0 -1500.0 2.447E-07 -4.322E-06 9.861E-07
-250.0 -70.0 -1495.0 2.509E-07 -4.390E-06 1.031E-06
-250.0 -70.0 -1490.0 2.571E-07 -4.457E-06 1.076E-06
…
```

tracking $(z, y, x) \longrightarrow$ transformPos $\longrightarrow$ B-Map $(x, y, z + 400mm)$

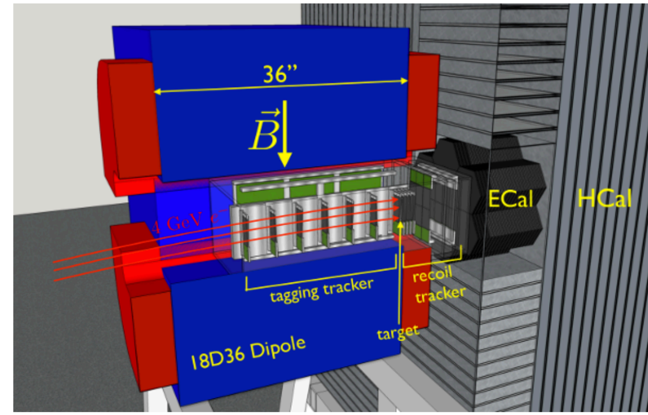
B-Map $(Bx, By, Bz) \longrightarrow$ transformField $\longrightarrow$ B-Map (Bz, Bx, By)

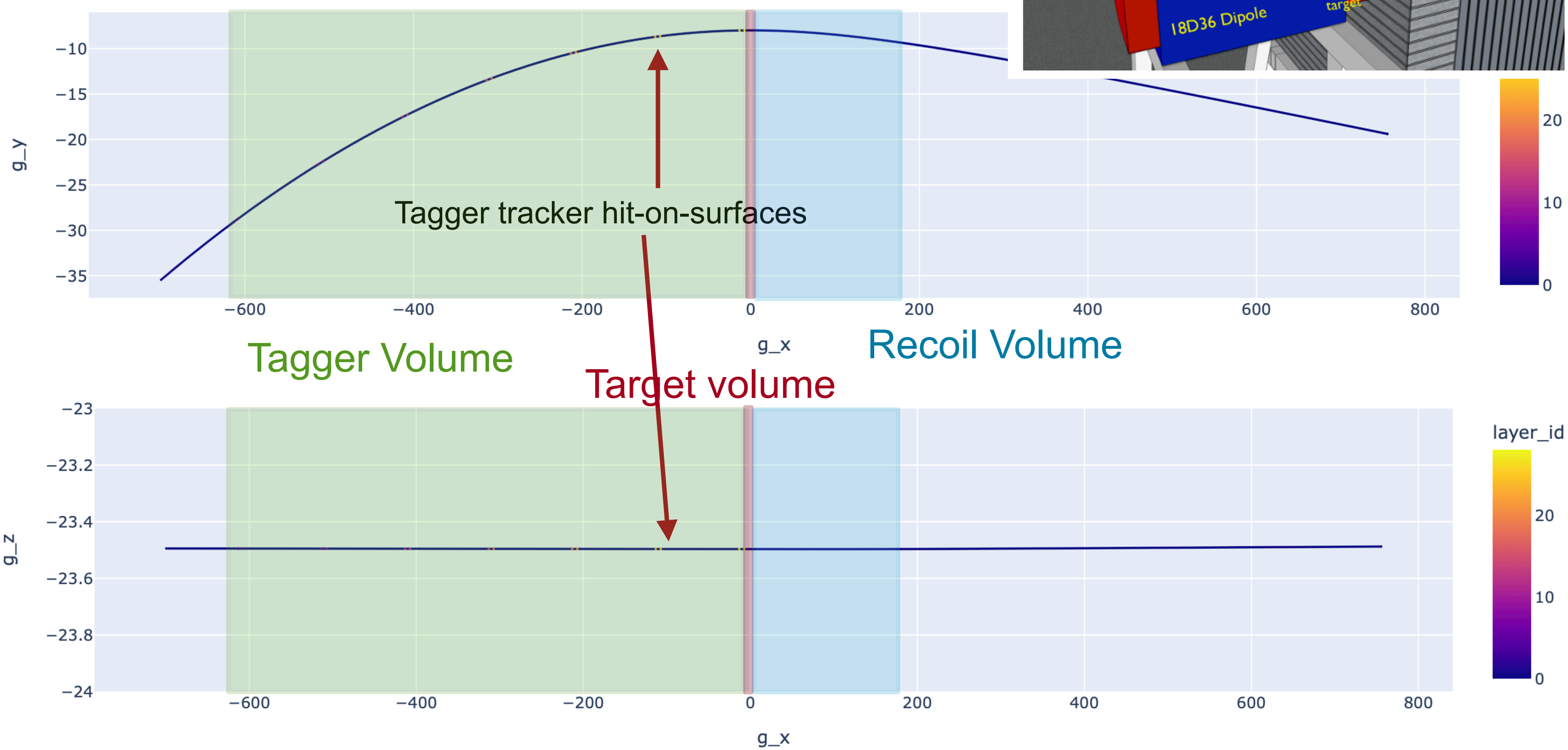
Origin in the center of the dipole

Origin on the target, rotated frame

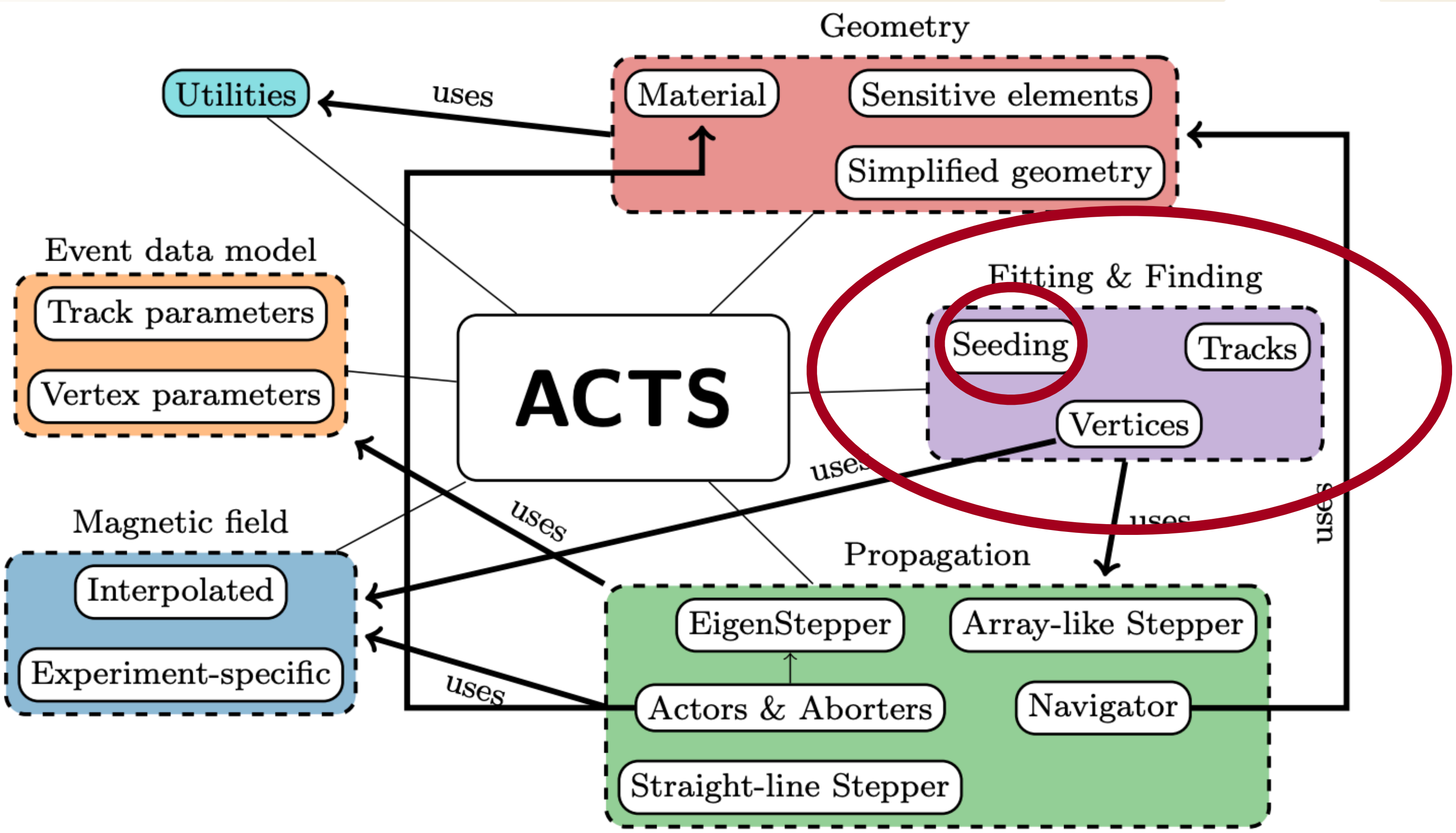
ACTS For LDMX - Interpolated Magnetic Field map

- Example of 1 electron, 4 GeV propagated from -35,0,-700 through the trackers
- Track straightens exiting the Bfield
- No target surface yet.





ACTS For LDMX - Library features



Tagger tracker seeding for ACTS

Space point formation

- No Digitisation available at the moment of making these studies
- 3D points for seeding are taken from Simulated hits

Seed finding

- Bin the available hits in middle-bottom and top points
- Form 3D-space points triplets that are compatible (various algorithms could be used here in principle)
- Fit for track parameters

The following slides on seed finding will show a recap of the work shown at the 2020 collab meeting

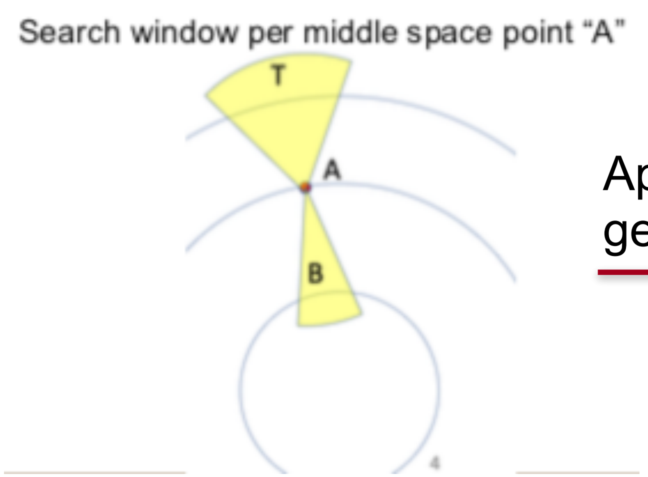
ACTS For LDMX - Seed Finder - Configuration

- I've used the nominal seed finder algorithm provided by ACTS configured for LDMX geometry
- Started by opening up a lot the cuts

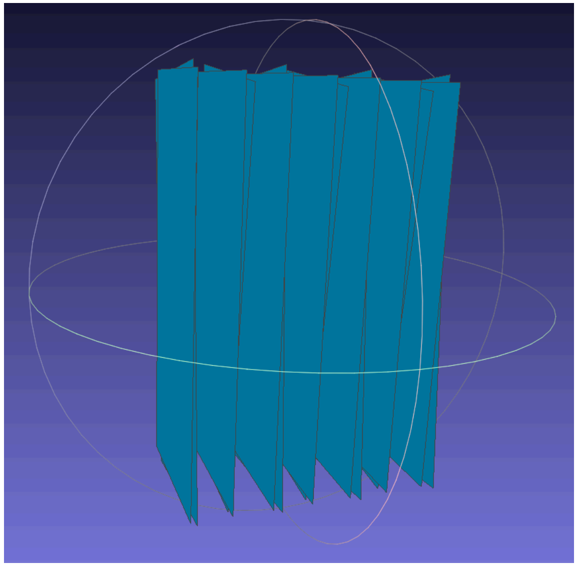
  - Notice wide cuts on $\tan(\lambda)$ and $p_T \geq 500$MeV.

  - Assume constant bField of 1.5 Tesla

  - Beamspot in (0,0), wide cuts in z0 (along global Y axis)



Apply to telescope geometry →

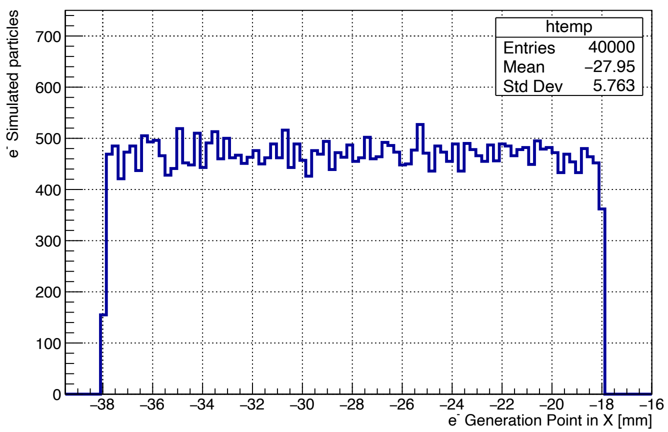


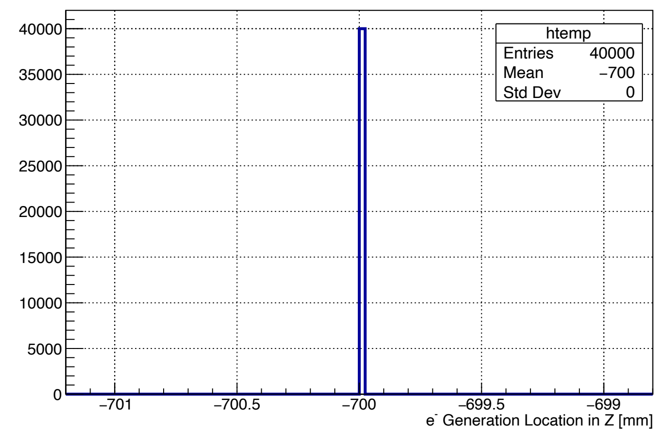
ACTS For LDMX - Seed Finder - Inputs

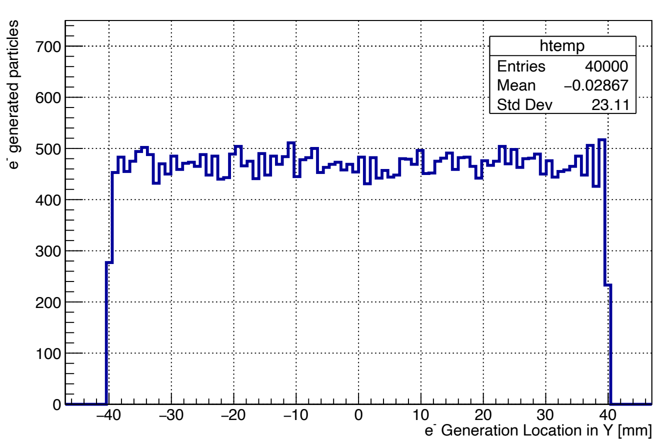
- The data used for the seed finder tests are single electrons from particle gun.
- Electrons are shot from (x=-27.926, y=0, z=-700)

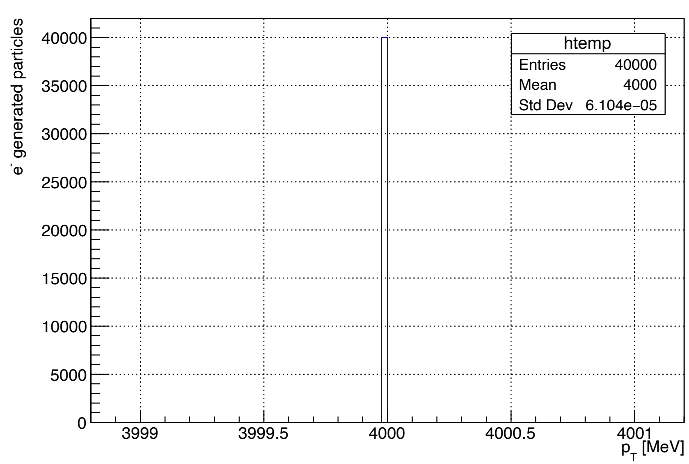
  - Parallel to X-Z plane, so $\tan(\lambda) = 0, \;\; |p| = |p_T| = \sqrt{p_x^2 + p_z^2} = 4$ GeV

  - **Such that they hit the target in a 80x20 region, nominally [-40,40] in Y and [-10,10] in X**

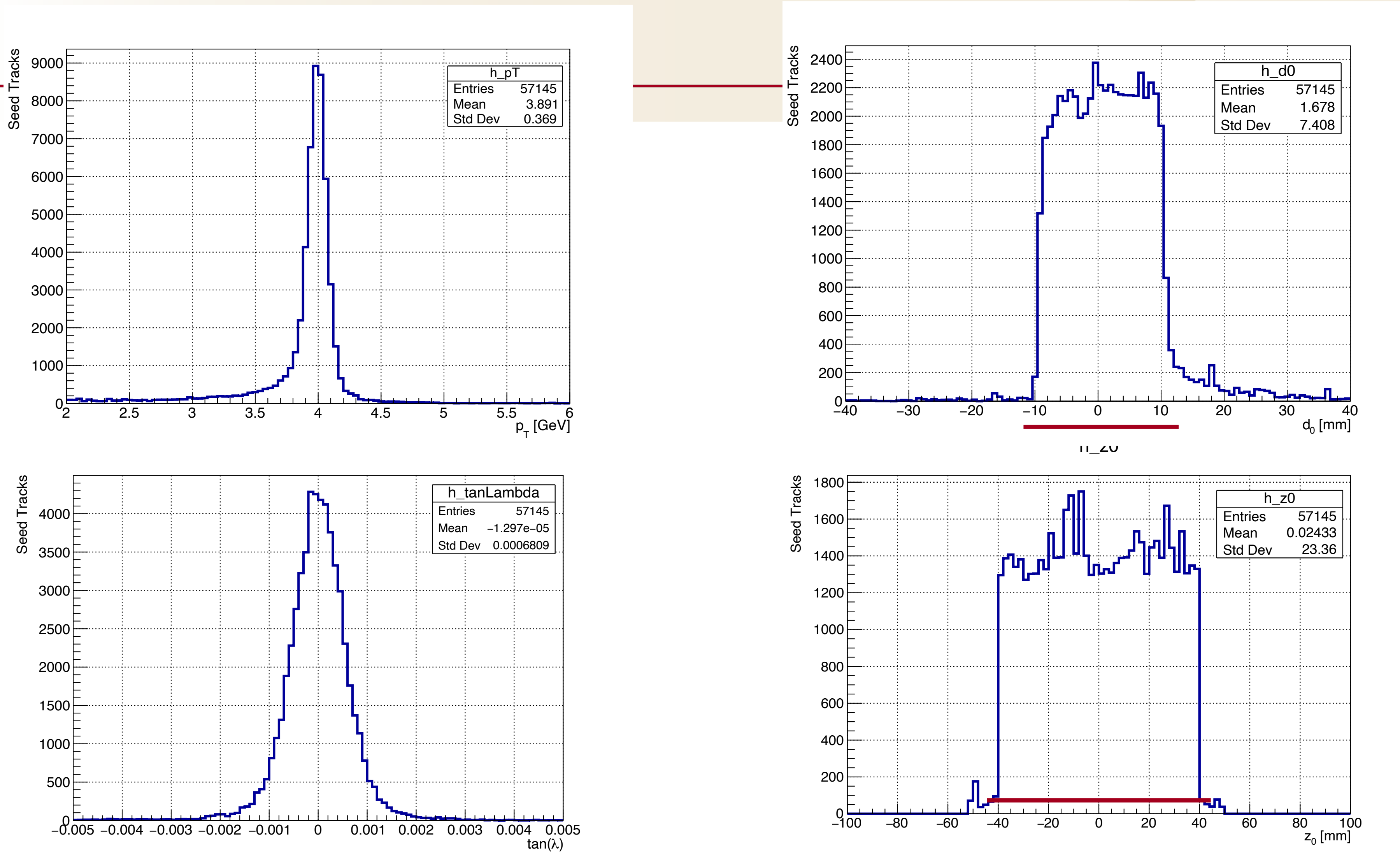






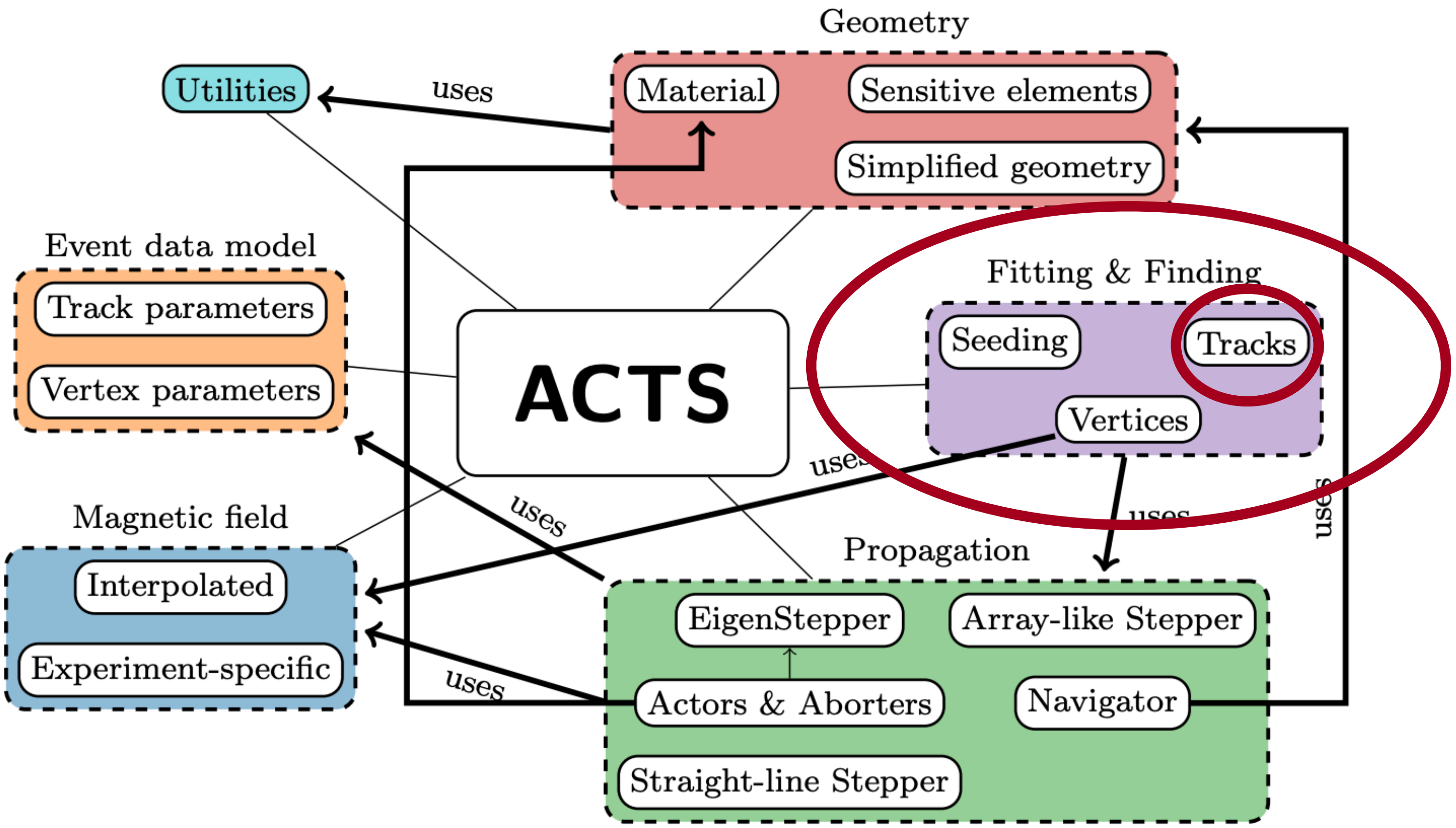


ACTS For LDMX - Seed Finder - Results



- Acts implementation of SiTrackMakerTool from ATHENA
- Seed finder correctly fits tracks to **pT~4 GeV**, **Impact parameters are within the expected limits from simulated production**.

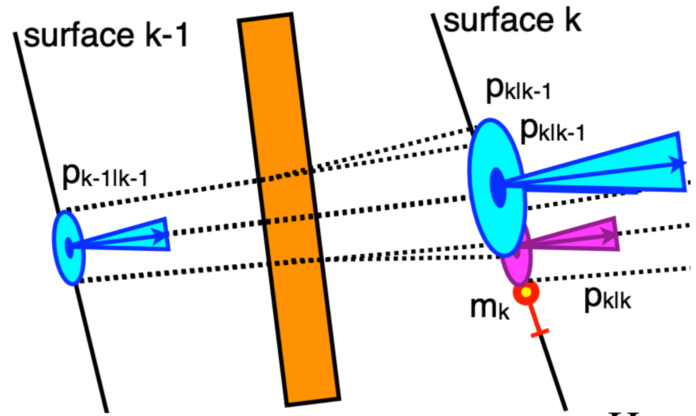
ACTS For LDMX - Library features



ACTS For LDMX - Track finding

SLAC

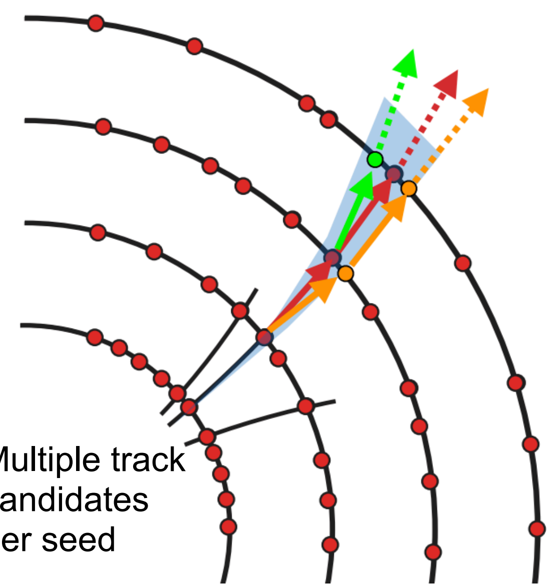
- Implemented **local** track finding algorithm using **Combinatorial Kalman Filter** (CKF)
  - Full combinatorials exploration to find all possible track candidates
  - Dedicated **measurement selectors** can be defined to accept/reject measurements. Currently **only** the hit with best χ^2 contribution is kept: **single track per seed**
  - Layer-wise **measurement calibrators** can be used to modify/correct single layer clusters during filtering
  - **Material effects and energy loss incorporated** during parameter propagation and filtering stage
  - **Non-uniform magnetic** field



$$r_k = m_k - Hp_{k|k-1}$$

H : projection matrix

- As digitization still not implemented, **currently using geant4 3D space points projected on relative sensor surfaces** (this mean 2D pixel-like local measurements, trivial to update to 1D-strips local measurements)



Multiple track candidates per seed

Details on the ldmx-sw to acts interface - Source Links

- First a map of the surfaces to layerId is formed (*current single uint to map the surface to layer, won't work with multiple surfaces per layer*)
- Used the **ActsExample::IndexSourceLink** class to interface the CKF to the ldmx-sw internally simulated points.
    - I don't want to depend on ActsExample, so I ported over the classes necessary for the compilation:
      - GeometryContainers.h
      - GroupBy.h
      - Range.h
- Loop over the simulated hits and associated them to the IndexSourceLink multi map

```
//Create a mapping between the layers
//and the Acts::Surface
  makeLayerSurfacesMap(tGeometry);
//Layers from 1 to 14
    unsigned int layerId = (surface->geometryId().layer() / 2);
    layer_surface_map_[layerId] = surface;

//Step 1 - Form the source links

  std::vector<ActsExamples::IndexSourceLink> sourceLinks;

//The mapping between the geometry identifier
//and the IndexSourceLink that points to the hit
  std::unordered_multimap<Acts::GeometryIdentifier,
                          ActsExamples::IndexSourceLink> geoId_sl_mmap_;

ActsExamples::IndexSourceLink idx_sl(hit_surface->geometryId(),i_ldmx_hit);
geoId_sl_mmap_.insert(std::make_pair(hit_surface->geometryId(), idx_sl));
 for (unsigned int i_ldmx_hit = 0;
 i_ldmx_hit < ldmxsps.size(); i_ldmx_hit++)


 const Acts::Surface* hit_surface = layer_surface_map_[layerid];
     if (hit_surface) {

  ActsExamples::IndexSourceLink idx_sl(hit_surface->geometryId(),
  i_ldmx_hit);
  geoId_sl_mmap_.insert(std::make_pair(hit_surface->geometryId(), idx_sl));

  }
```

```
using LdmxSourceLinkAccessor = GeneralContainerAccessor<
std::unordered_multimap<Acts::GeometryIdentifier, ActsExamples::IndexSourceLink>
>  ;

  Acts::CombinatorialKalmanFilterOptions<LdmxSourceLinkAccessor> kfOptions(
      gctx_,bctx_,cctx_,
      LdmxSourceLinkAccessor(), ckf_extensions, Acts::LoggerWrapper{logger()},
      //propagator_options,&(*perigee_surface));
      propagator_options,&(*gen_surface));
  // run the CKF for all initial track states
  auto results = ckf.findTracks(geoId_sl_mmap_, startParameters, kfOptions);
```

Details on the ldmx-sw to acts interface - Source Links

- The LdmxSourceLinkAccessor is used to loop over the hits on each collected and mapped surface and interface to the CKF algorithm.
- It's an example of implementation for the accessor over a multi map that maps a geometry ID to a vector of SourceLinks.
- The code is basically ported from the Acts CKFTests

```cpp
template <typename T>
using GeometrySourceLinkMap = std::unordered_map<Acts::GeometryIdentifier, std::vector<T> >;

template <typename T>
struct LdmxSourceLinkAccessor {

  using Container = GeometrySourceLinkMap<T>;
  using Key       = Acts::GeometryIdentifier;
  //using Value     = typename GeometrySourceLinkMap<T>::value_type;
  using Value     = T;
  //using Iterator  = typename GeometrySourceLinkMap<T>::const_iterator;
  using Iterator  = typename std::vector<T>::const_iterator;

  const Container* container = nullptr;

  size_t count(const Acts::GeometryIdentifier& geoId) const {
    assert(container != nullptr);
    //return container->count(geoId);
    return container->at(geoId).size();
  }

  std::pair<Iterator, Iterator> range (
      const Acts::GeometryIdentifier& geoId) const {
    assert(container != nullptr);
    return std::pair<Iterator,Iterator>(container->at(geoId).begin(), container->at(geoId).end());
  }

  //const Value& at(const Iterator& it) const {return *it;}
  const T& at(const Iterator& it) const {return *it;}

};
```

```cpp
template <typename container_t>
struct GeneralContainerAccessor {
  using Container = container_t;
  using Key = typename container_t::key_type;
  using Value = typename container_t::mapped_type;
  using Iterator = typename container_t::const_iterator;

  // pointer to the container
  const Container* container = nullptr;

  // count the number of elements with requested key
  size_t count(const Key& key) const {
    assert(container != nullptr);
    return container->count(key);
  }

  // get the range of elements with requested key
  std::pair<Iterator, Iterator> range(const Key& key) const {
    assert(container != nullptr);
    return container->equal_range(key);
  }

  // get the element using the iterator
  const Value& at(const Iterator& it) const { return (*it).second; }
};
```

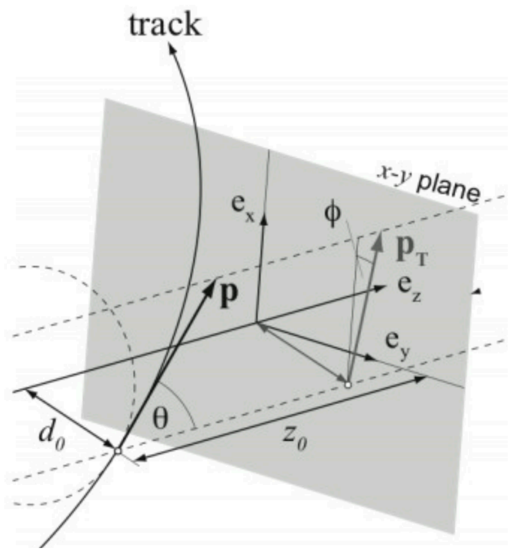
Details on the ldmx-sw to acts interface - Measurement Calibrator

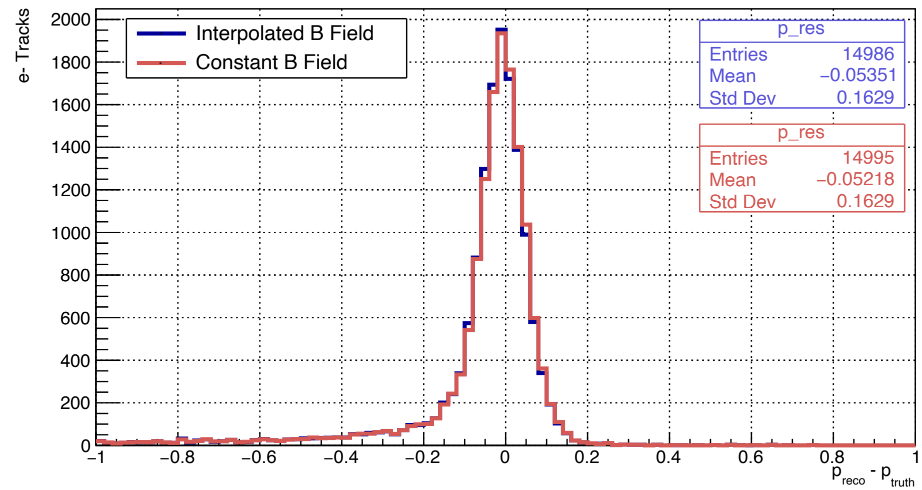
- Last piece is an implementation of a Measurement Calibrator
- In this part I explicitly define the number of measurements I want to use (easy to pass from a 3D space point (axial-stereo) cross to a 2D space point (single strip-sensor cluster)) by redefining the H matrix, measurement dimension and cov matrix.

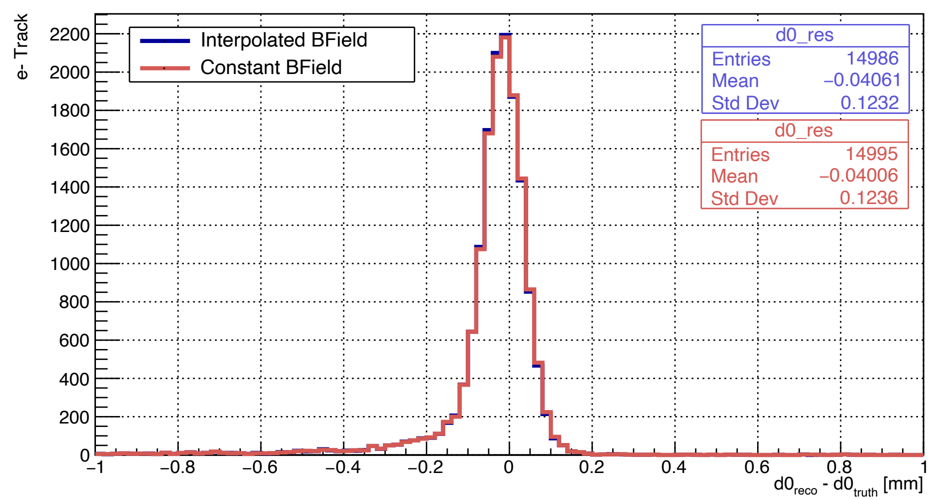
```
//The calibrator needs to access the sim hit container
LdmxMeasurementCalibrator(const std::vector<ldmx::LdmxSpacePoint*>& measurements) {
  m_measurements = &measurements;
}

/// Find the measurement corresponding to the source link.
///
/// @tparam parameters_t Track parameters type
/// @param gctx The geometry context (unused)
/// @param trackState The track state to calibrate
void calibrate(const Acts::GeometryContext& /*gctx*/,
               Acts::MultiTrajectory::TrackStateProxy trackState) const {
  const auto& sourceLink =
      static_cast<const ActsExamples::IndexSourceLink&>(trackState.uncalibrated());

  assert(m_measurements and
         "Undefined measurement container in LdmxMeasurementCalibrator");
  assert((sourceLink.index() < m_measurements->size()) and
         "Source link index is outside the container bounds in LdmxMeasurementCalibrator");

  auto meas = m_measurements->at(sourceLink.index());

  trackState.calibrated().setZero();
  trackState.calibrated().head<2>() = meas->local_pos_;
  trackState.data().measdim = 2;
  trackState.calibratedCovariance().setZero();
  trackState.calibratedCovariance().block<2,2>(0,0) = meas->local_cov_;
  trackState.setProjector(meas->projector_);

}
```

ACTS For LDMX - Track finding

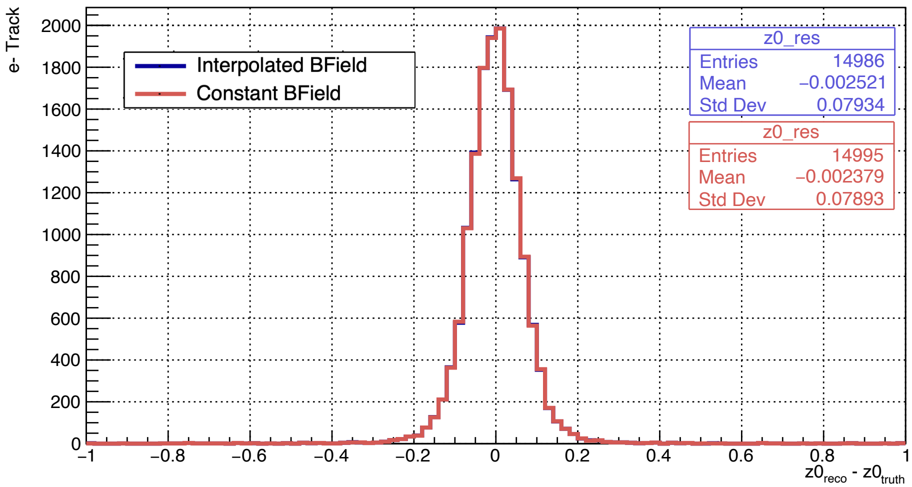
- Track finding/fitting performance is checked using 15k 4 GeV propagated electrons from (-35,0,-700) through the trackers
- **Track parameters** $(d_0, z_0, \theta, \phi, q/p, t)$ are **estimated back at the generation vertex**, where the perigee point is defined (so reco-truth residuals make sense)

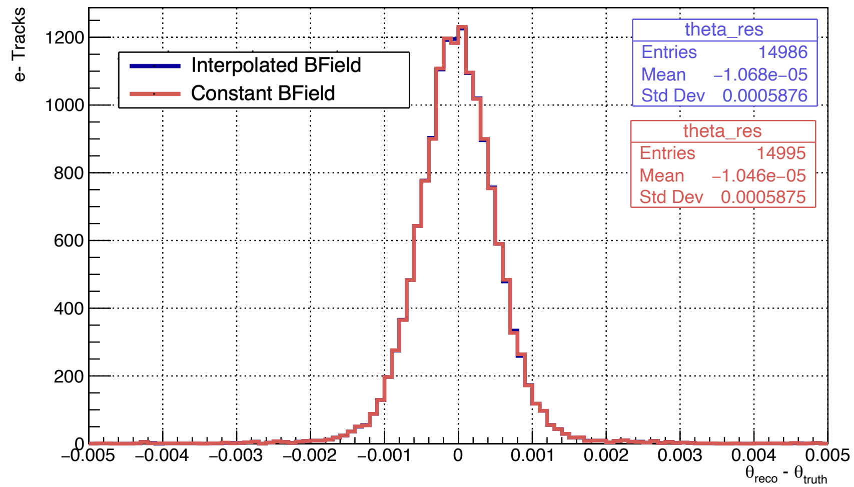


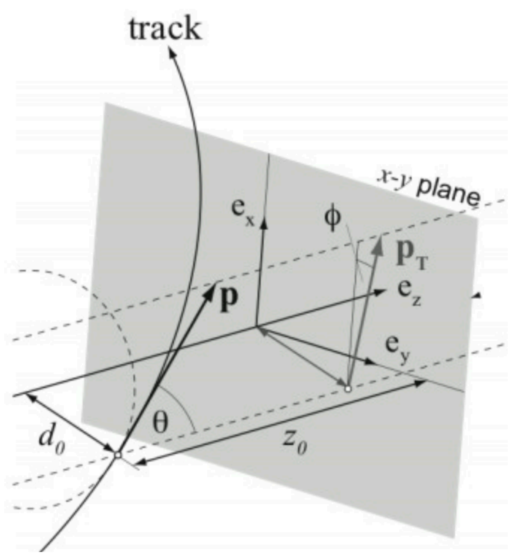


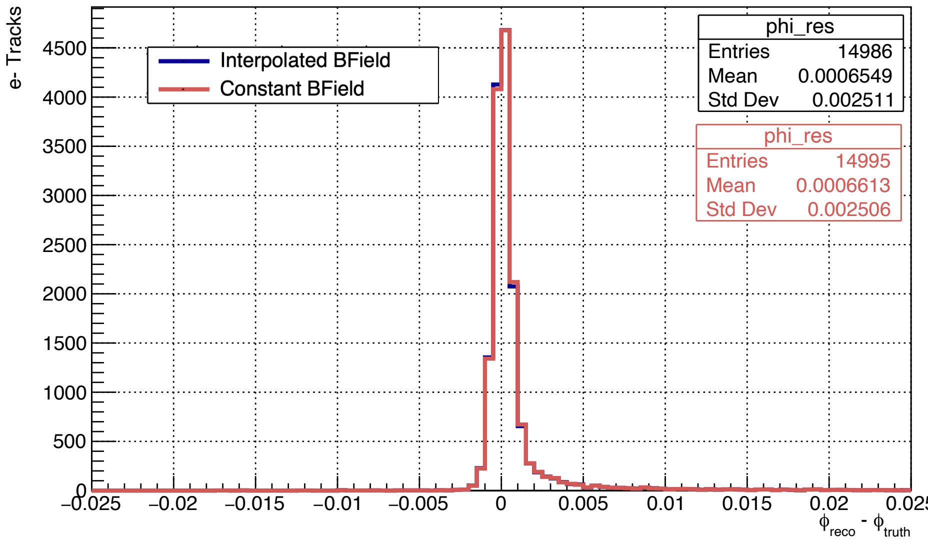


ACTS For LDMX - Track finding









Summary

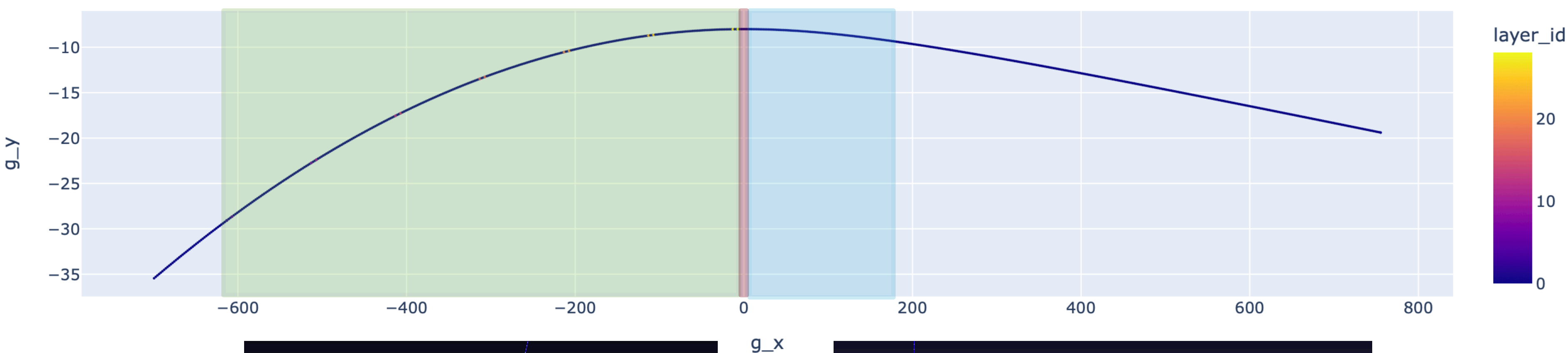
- ACTS library has been integrated into ldmx-sw
  - **Tom** is the author of a docker container with the acts/ dd4hep libraries included (see PR 37)
  - **Omar** provided an implementation of the Tagger Tracker and Recoil Tracker based on DD4hep which can be read in and parsed into the ACTS tracking geometry
- A **seeding algorithm** has been included in ldmx-sw which takes in input smeared 3D-space points and provides tracks states on an arbitrary surface for starting the KF-based pattern recognition
- **CKF-based track finding and fitting** based has been implemented and tested in constant and **non-uniform B-Field**

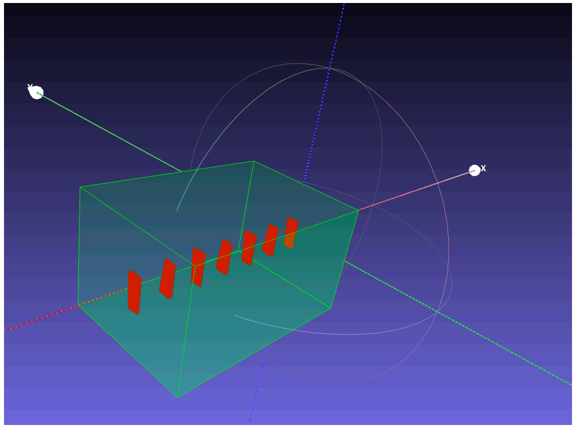
Questions for the experts

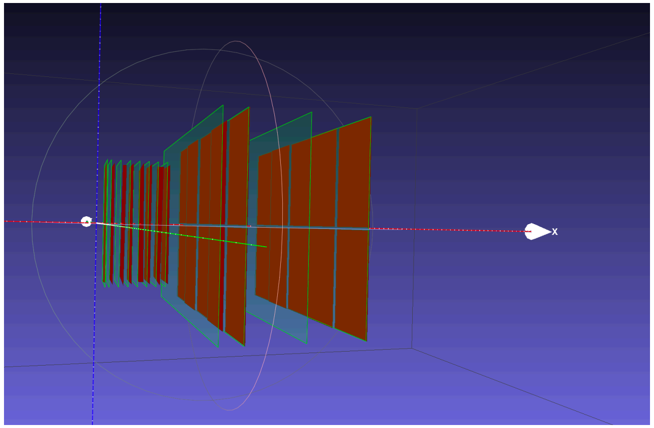
- We are working on a new version of our reconstruction software that would abandon ROOT. This means we would like to use GeoModel for the geometry representation: is there a decoder for building the tracking Geometry?
- Do we still need Navigation/Material layers in the tracker trackingGeometry? Is the orientation/definition something I should revisit?

Questions for the experts

- What is the best way to run track reconstruction in the two tracker systems?
  - We probably want to reconstruct the two tracklets separately: do we need two disjoint tracking volumes? Propagator should stop at the target in this case
  - Most of the particles will pass through the target: what is the best way to refit a track with hits in both systems?
  - What if we want to vertex tracks from the tagger and the recoil at the target?
- Is there a conceptual analogy in ATLAS, maybe hadronic interactions on material layers that use different sub-system tracks?



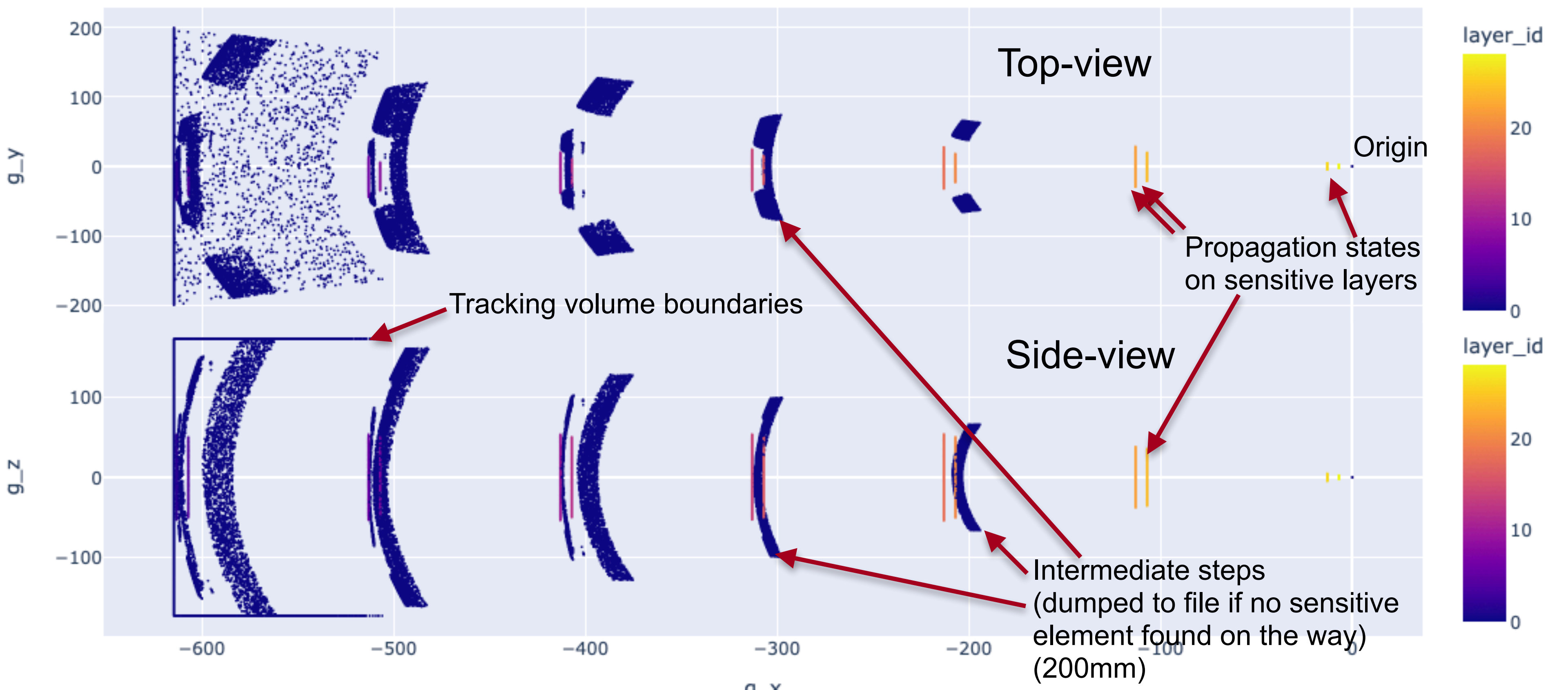




BACKUP

ACTS For LDMX - Tracking Geometry

- Implemented a propagator test in ldmx-sw: Check RK extrapolation and tracking geometry implementation
- Can be applied to other tracking geometries, both for the recoil or calorimeters (if wanted)
- Figures shows 10k states propagated from (0.,0.,0) for 1GeV particles with randomised ϕ and θ angles in **BField OFF** case (straight-lines). Intersections with sensitive elements are shown (coloured ones) as well as intermediate propagator steps or boundary ones (blue)



ACTS For LDMX - Seed Finder - Results

- Look for seeds with hits on **ly 3,7 and 9** (in 1-14) numbering **(for example)**
  - Single iteration, no recursive search, duplicates only present if multiple hits on those layers.
- **After seed is found a 3-hit fit is perfomed**
  - Implemented few fitting procedures:
    - SiTrackMakerTool from ATHENA (ATLAS)
    - Karimaki Circle Fit (NIM 305 1 1991 (187-191))
    - Parabolic fit in conformal space (WIP)