Acts overview

Paul Gessinger

CERN / JGU Mainz

01.10.2019 - ErUM Data IDT Collaboration Meeting

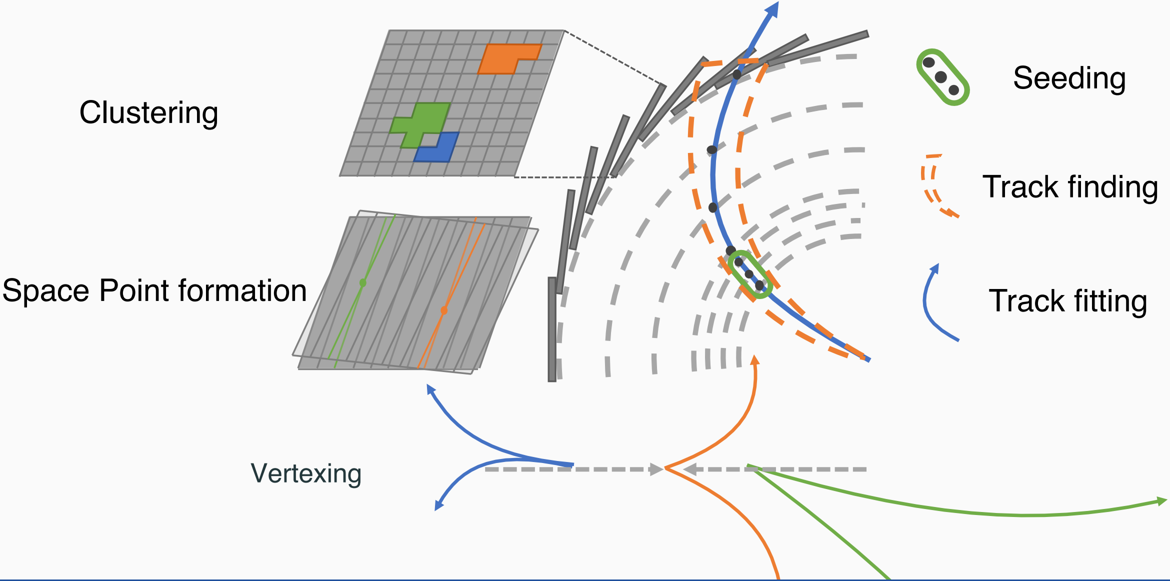
This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research

Outline

- Track reconstruction
  - What is it?, Challenges
- The Acts project
  - Design goals and basic principles, project structure
- Components:
  - Geometry description
  - Numerical integration / propagation
  - Multiple events in flight and context handling
  - Track fitting: the Acts Kalman Filter and Event Data Model
- Status and outlook

Track reconstruction

Track reconstruction

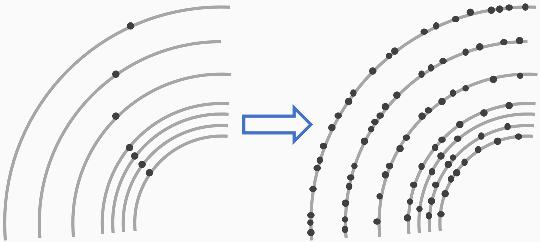


Track reconstruction

- Turn hits on sensors into trajectories of particles that produced them
- Multi-Stage process:
  - Pattern recognition to reduce combinatorics
  - Exploration of all compatible measurements
  - Selection of best candidates and *precise* fit $\rightarrow$ **best estimate** of trajectory
- Remove overlap between different solutions: *ambiguity resolution* (crucial for performance)

Challenges

- This is routinely the largest CPU consumer in event reconstruction
- Pileup affects performance significantly:



- Looking at *HL-LHC* era and beyond: tracking needs to **improve**!

The Acts project

cern.ch/acts

The Acts project

- Based on ATLAS tracking software[3]
- Goal: speed up development cycles, test out concepts in parallelism
- Contributions mostly from ATLAS so far, but interest/contributions from outside, too
- Open-source (MPLv2): Gitlab

Project structure

- `acts-core`: main library
  - Contains tools and components
  - Doesn't assume anything about event-processing framework
- `acts-framework`: small `GaudiHive`-inspired event processing framework
  - Event-level parallelism for testing
  - Has *generic* geometry, `TGeo` and `DD4hep` plugins
- `acts-fatras`: Acts-based fast track simulation
  - Can be used to create scenarios for testing and validation

Tool interface design

- Constant configuration as config struct at construction
- *Options* struct for configuration at invokation
- Thread-local state as arguments: no **mutable** members

```
struct MyTool {
  struct Config { double value{42}; };
  struct Options { double optval{1337}; };
  struct State{};
  MyTool(Config cfg)
    : m_cfg(std::move(cfg)) {}
  void doSomething(State& state,
                   const Options& opt)
    const { /* ... */ };
  Config m_cfg;
};
```

```
int main() {
  MyTool::Config cfg;
  MyTool tool{std::move(cfg)};
  MyTool::Options opt;
  MyTool::State state;
  tool.doSomething(state, opt);
}
```

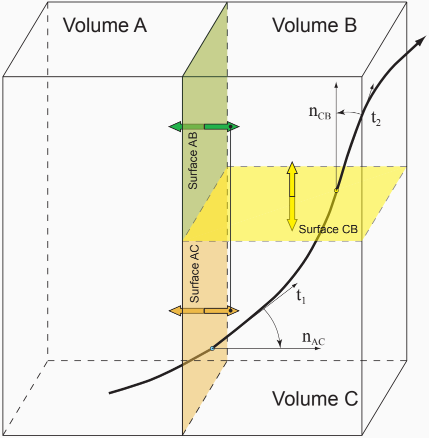
Geometry and navigation

Geometry modeling and navigation

- Concepts from ATLAS: fully detailed geometry for precise simulation, simplified **tracking geometry** with only sensitive sensors for faster navigation and propagation
- Individual sensors are grouped into **layers**
- Layers are binned to allow fast retrieval of compatible surfaces
- Layers are grouped into **volumes**

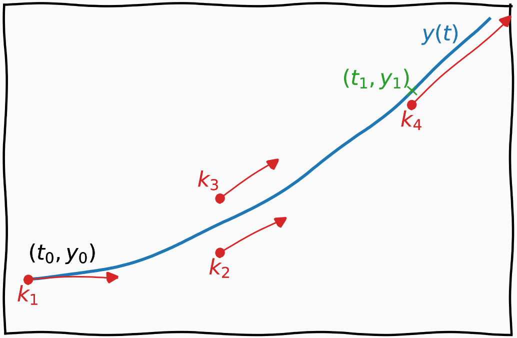
Navigation

- Optimize navigation for **speed**
- Idea: pre-resolve transitions as much as possible
- Volumes are *glued* together using *boundary* surfaces
- Navigation works volume to volume, then layer to layer

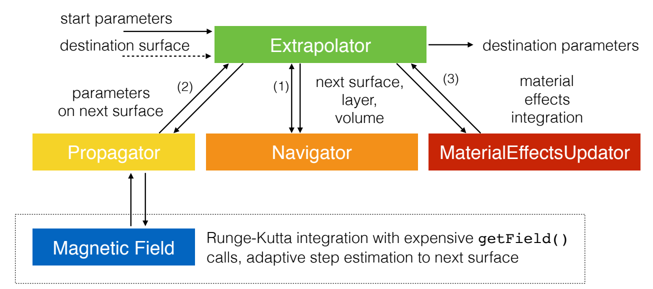


[8]

Numerical integration



Propagation in ATLAS



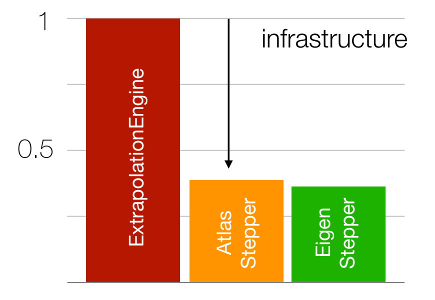
- Uses Runge-Kutta-Nyström integration[2, 4]
- Lots of **virtual method calls** and **dynamic memory allocation**
- Was packaged in Acts as `ExtrapolationEngine` as baseline (now gone)

Propagation in Acts

- Generalize *optional* components into **Actors** and **Aborters** (e.g. `MaterialActor`)
- Keep component structure for *required* components: **Stepper** and **Navigator**
- All components are template parameters: **no virtual calls**
- ATLAS' different propagators $\rightarrow$ *integration-term extensions* to our main integrator **EigenStepper**

Propagation

- **EigenStepper**: primary Acts integrator
- **AtlasStepper**: transcription of ATLAS numerical integrator
- Infrastructure changes enable significant speedup:
- Covariance transport and *STEP* mechanism[5, 6] implemented
- Possibility for alternative / specialized implementations



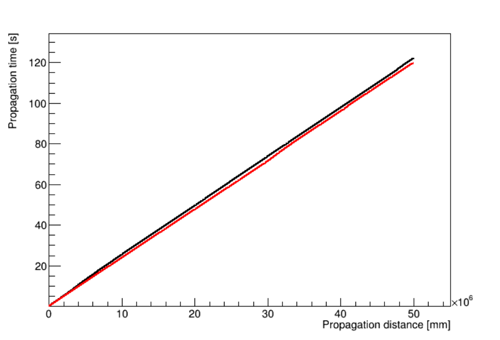
[7]

Timing information

- Timing is supported in numerical integration
- Data structures allow for time measurements naturally (see KF later)
- Tests indicate no negative performance impact

$$\vec{x} = (x, y, z, t, T_x, T_y, T_z, q/p)$$

with time, without time



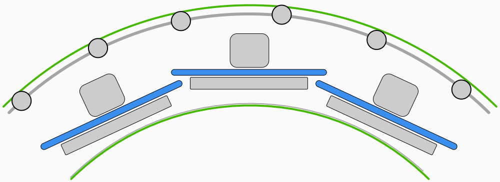
MR (merged) by Fabian Klimpel

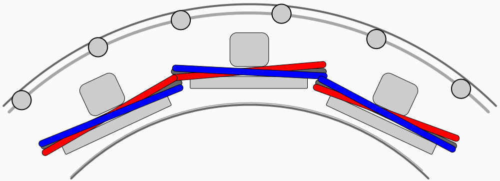
Multiple events in flight and context handling

Handling of context

- Some aspects of the detector change over time: e.g. magnetic field, temperatures, calibration and **alignment**
- Especially non-trivial when multiple events in flight
- Need to be able to communicate what the current context is

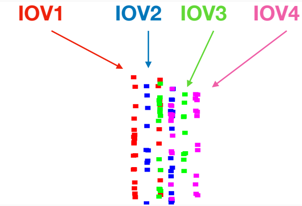
Example: alignment handling

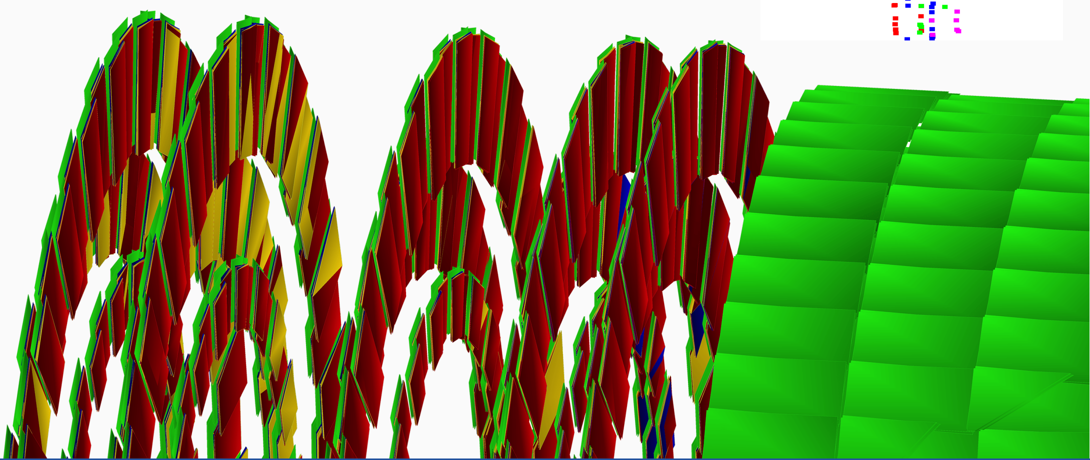




- Use **context objects**: geometry context, calibration context, magnetic field context
- Create at event level, pass down to where needed
- Current ATLAS code: **Synchronization by `std::mutex`**
- Acts implementation in Athena:
  - Acts implementation: lock free, no synchronization needed!
  - Flexible, doesn't change for different sources
  - Want to extend tests to magnetic field and calibration

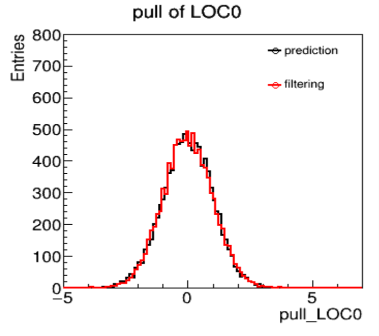
Example: alignment handling

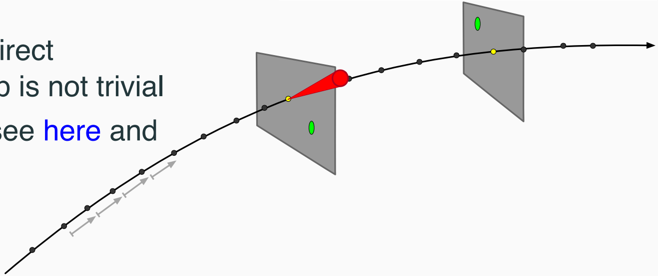




Kalman Filter in Acts

- Kalman Filter is implemented as an extension to the propagator[1]
- Gets called automatically during regular propagation
- Can update direction, uncertainties after filtering step
- Aim to minimize heap allocation
- Runtime performance: So far no direct comparison, comparable test setup is not trivial
- Study of numerical performance (see here and here by Xiaocong Ai)

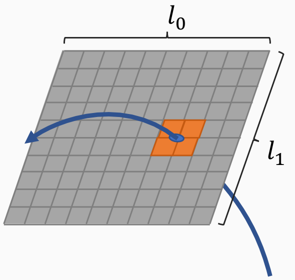




[1] *Actor*

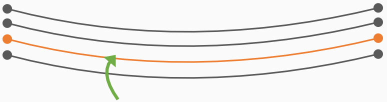
Kalman Filter EDM: Measurements

- Local sensor frame is same as measurement frame
- Measurement mapping function H is a projection matrix
- Input to KF: lightweight `SourceLink` object
- Is turned into Acts measurement by **calibrator** (context aware)
- Knows dimensions at compile-time: **fixed-size** matrix operations
- Projector is known from type
- Dispatch on concrete type at runtime



$$\vec{r} = (l_0, l_1, \phi, \theta, q/p, t)^T$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Multiple events in flight and context handling

- Current EDM still uses dynamic allocation in a number of places
- Goal: further minimize heap allocation as much as possible
- Take inspiration from ATLAS' xAOD (column wise storage, collection based)
- Decouple access code from actual storage/memory model (ideally: switchable memory backend)
- Currently ongoing: explore EDM replacement for Kalman Filter
  - Investigating if that can be used for other fitters as well
  - Investigating compatibility for interfacing to other execution environments
  - Look into performance characteristics

Status and outlook

Status

- Propagation, single Kalman Filter, Seed Finder developed and usable
- Time propagation implemented and tested
- So far: tests mostly standalone, real validation pending
- Tests of ATLAS geometry (ID + first tests for calorimeter modeling and navigation)
- Demonstrated multi-threaded execution with alignment

Work in Progress

- **Multi-component propagation** to be used for Gaussian Sum Filter (WIP MRs: !582, !657 by Jin Zhang)
- Components for **vertex fitting**, finding implemented and being deployed (see this and more by Bastian Schlag)
- R&D into **machine learning for vertex finding** (Bastian Schlag) **ambiguity resolution** (Nicholas Cinko)
- Implement **hashing based *bucketing*** in Acts (Sabrina Amrouche, Moritz Kiehn, see here)
- Integration testing in **Belle-2** (Nils Braun, following talk)
- Three promising solutions to **TrackML challenge** being implemented ([1], WIP MR !152, Sabrina Amrouche, Moritz Kiehn, Sharad Chitlangia (GSoC))

Plans

- Detailed comparison of Acts and ATLAS propagation (Noemi Calace)
- Compare Acts Kalman Filter to ATLAS Kalman Filter (Xiaocang Ai)
- Rewrite navigation to make it more adaptable to various navigation approaches (Andreas Salzburger, Paul Gessinger)
- Start work on a Combinatorial Kalman Filter implementation (some preliminary tests exists, Xiaocong Ai, Paul Gessinger)
- Complete contextual setup with magnetic field, alignment, calibration (demonstrate on OpenDataDetector, deploy in ATLAS)
- Get to the point where we can demonstrate a full track reconstruction chain

Backup

Numerical integration

FORTRAN

```fortran
CALL THEFLSP ( IPR(3)    )  ! track propagation through
                            ! precision detectors
CALL THEFILP (   1       )  ! Fill output track bank without TRT
CALL THEFLST ( IPR(4)    )  ! Track propagation through TRT
CALL THEFILT                ! Fill output track bank with TRT
CALL THEBRE  ( IBREM     )  ! Brem. fit possibility investigation
IF ( IBREM.NE.0 ) GO TO 20  ! Repeat fit with brem. conditions
                  GO TO 10  ! Go to next track candidate
CALL THERRO  (   IER     )  ! Test errors list
CALL THELOOK                ! Tracks comparison
```

Numerical integration

FORTRAN++

```
double H1[3] = {f[0]*PS2,f[1]*PS2,f[2]*PS2};
double A3    = (A[0]+B2*H1[2])-C2*H1[1]     ;
double B3    = (A[1]+C2*H1[0])-A2*H1[2]     ;
double C3    = (A[2]+A2*H1[1])-B2*H1[0]     ;
double A4    = (A[0]+B3*H1[2])-C3*H1[1]     ;
double B4    = (A[1]+C3*H1[0])-A3*H1[2]     ;
double C4    = (A[2]+A3*H1[1])-B3*H1[0]     ;
double A5    = 2.*A4-A[0]                   ;
double B5    = 2.*B4-A[1]                   ;
double C5    = 2.*C4-A[2]                   ;
```

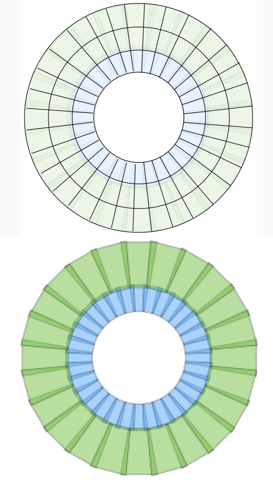
Numerical integration

Modern C++ & Eigen

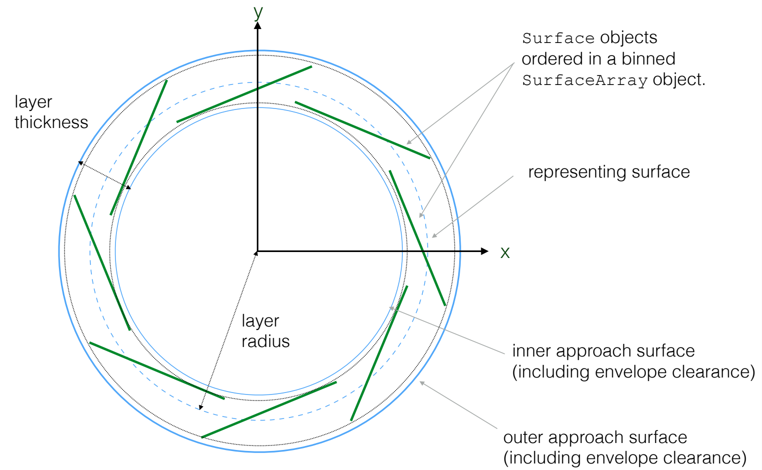
```
// Update the track parameters according to the equations of motion
state.stepping.pos +=
    h * state.stepping.dir + h2 / 6. * (sd.k1 + sd.k2 + sd.k3);
state.stepping.dir += h / 6. * (sd.k1 + 2. * (sd.k2 + sd.k3) + sd.k4);
state.stepping.dir /= state.stepping.dir.norm();
```

- Matrix operations abstracted into operators
- Eigen should produce near-optimal code for operations
- Readable → more maintainable

Geometry modeling and navigation

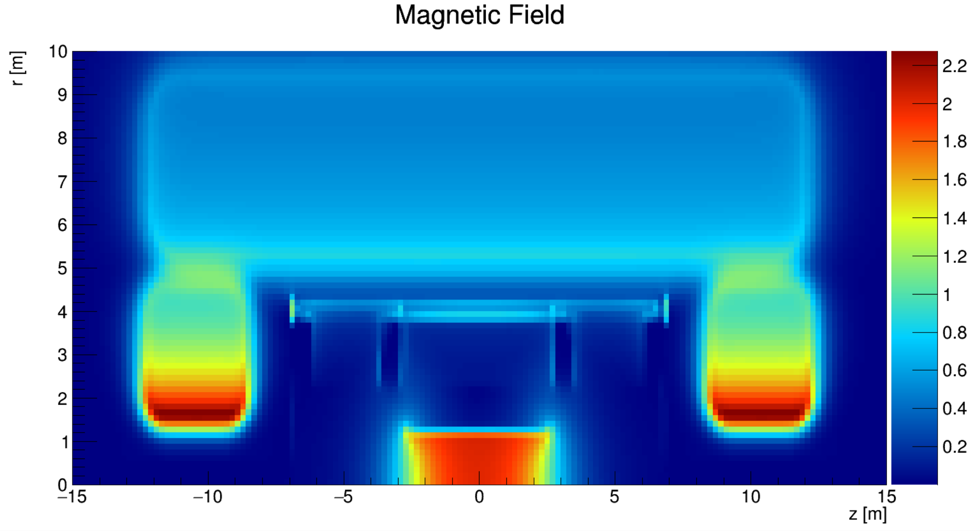


A. Salzburger



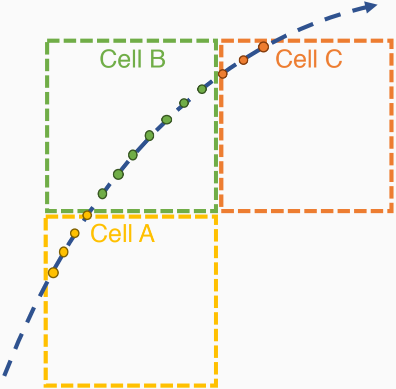
x-y view of a `CylinderLayer`
(with planar detection elements)

Magnetic field access



Magnetic field access

- Observation: magnetic field is queried **very frequently** during propagation
- Most of the time: **very little distance** between queries
- Mitigation: keep field cell in thread-local cache, interpolate linearly from corners



Example: alignment handling

Current ATLAS code:

```
const Amg::Transform3D &
SiDetectorElement::transform() const
{
  std::lock_guard<std::recursive_mutex> lock(m_mutex);
  if (!m_cacheValid) updateCache();
  return m_transform;
}
```

Example: alignment handling

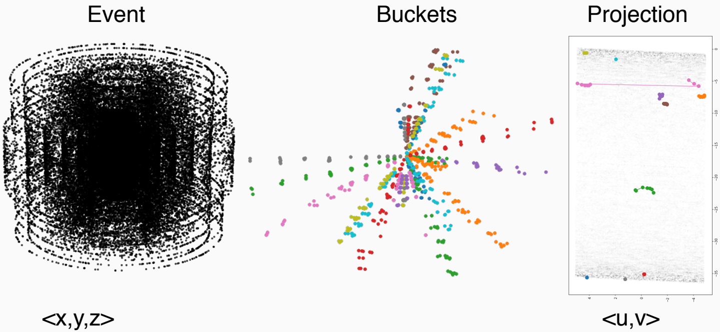
Acts implementation in Athena:

```
const Acts::Transform3D&
ActsDetectorElement::transform(
  const Acts::GeometryContext& anygctx) const
{
  const ActsGeometryContext* gctx
     = std::any_cast<const ActsGeometryContext*>(anygctx);
  const ActsAlignmentStore* alignmentStore = gctx->alignmentStore;
  const Transform3D* trf = alignmentStore->getTransform(this);
  return *trf;
}
```

Kalman Filter EDM: Measurements iii

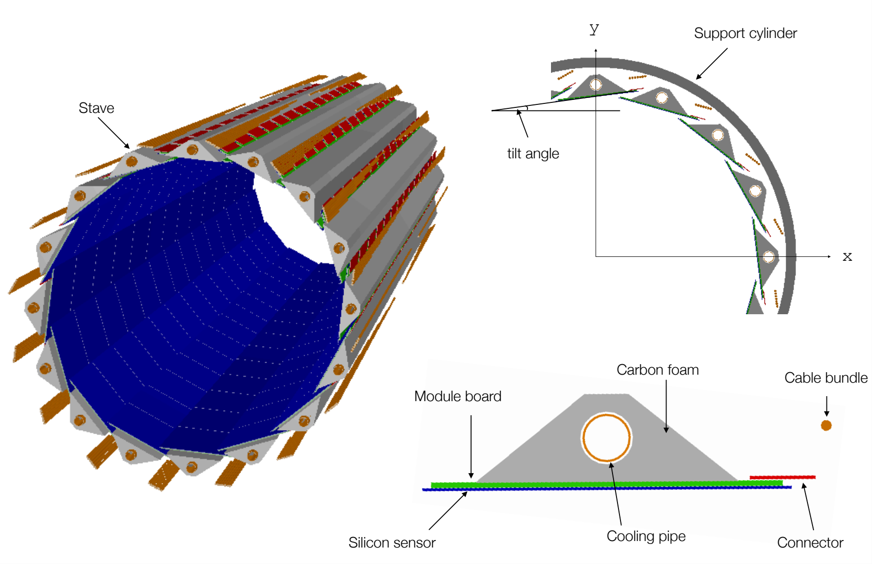
```
std::visit([&](const auto& calib) {
  const projection_t& H = calib.projector();
  // calculate gain matrix
  gain_matrix_t K = pred_covariance * H.transpose() *
    (H*pred_covariance*H.transpose() + calib.covariance()).inverse();
  // update parameters and covariance
  filt_parameters = pred.parameters() + K * calib.residual(pred);
  filt_covariance =
      (CovMatrix_t::Identity() - K * H) * pred_covariance;
  parameters_t filt(/* ... */);
  meas_par_t res = calib.resisual(filt);
  // calculate filtered chi2
  ts.parameter.chi2 = (res.transpose() * ((meas_cov_t::Identity()
    - H * K) * calib.covariance()).inverse() * res).value();
  ts.parameter.filt = std::move(filt);
}, *ts.measurement.calibrated);
```

Hashing based bucketing

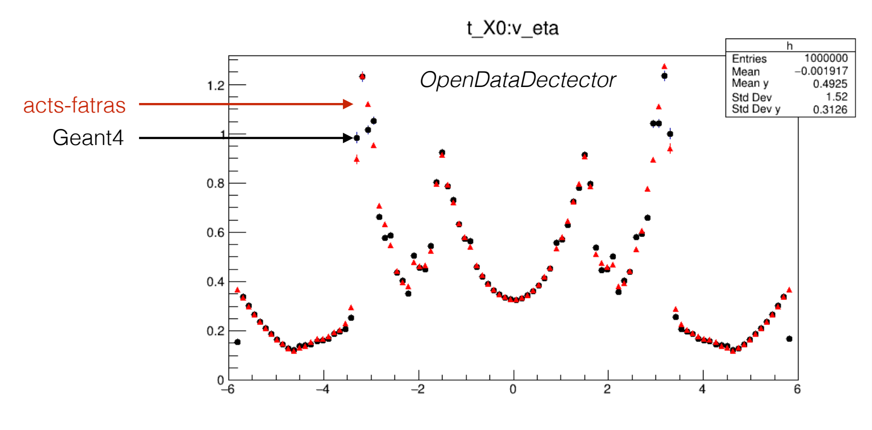


Sabrina's talk at the TrackML Grand Finale

Open Data Detector I



Open Data Detector II



References I

[1] S. Amrouche et al. “The Tracking Machine Learning challenge : Accuracy phase”. In: (2019). arXiv: 1904.06778 [hep-ex].

[2] L. Bugge and J. Myrheim. “Tracking and Track Fitting”. In: *Nuclear Instruments and Methods* 179.2 (1981), pp. 365–381. ISSN: 0029-554X. DOI: https://doi.org/10.1016/0029-554X(81)90063-X.

[3] T. G. Cornelissen et al. “The New ATLAS Track Reconstruction (NEWT)”. In: *J. Phys. Conf. Ser.* 119 (2008), p. 032014. DOI: 10.1088/1742-6596/119/3/032014.

[4] E. Lund et al. “Track Parameter Propagation through the Application of a New Adaptive Runge-Kutta-Nystroem Method in the ATLAS Experiment”. In: *JINST* 4 (2009), P04001. DOI: 10.1088/1748-0221/4/04/P04001.

[5] E. Lund et al. “Transport of Covariance Matrices in the Inhomogeneous Magnetic Field of the ATLAS Experiment by the Application of a Semi-Analytical Method”. In: *Journal of Instrumentation* 4.04 (Apr. 24, 2009), P04016–P04016. ISSN: 1748-0221. DOI: 10.1088/1748-0221/4/04/P04016.

References II

[6] E. Lund et al. *Treatment of Energy Loss and Multiple Scattering in the Context of Track Parameter and Covariance Matrix Propagation in Continuous Material in the ATLAS Experiment*. ATL-SOFT-PUB-2008-003, ATL-COM-SOFT-2008-008. Jan. 23, 2019. URL: https://cds.cern.ch/record/1114577.

[7] A. Salzburger. "Optimisation of the ATLAS Track Reconstruction Software for Run-2". CHEP 2015 (Okinawa). Apr. 13, 2019. URL: https://indico.cern.ch/event/304944/contributions/1672482/.

[8] A. Salzburger, Š. Todorova, and M. W. Wolter. *The ATLAS Tracking Geometry Description*. ATL-SOFT-PUB-2007-004. ATL-COM-SOFT-2007-009. Geneva: CERN, June 2007. URL: http://cds.cern.ch/record/1038098.