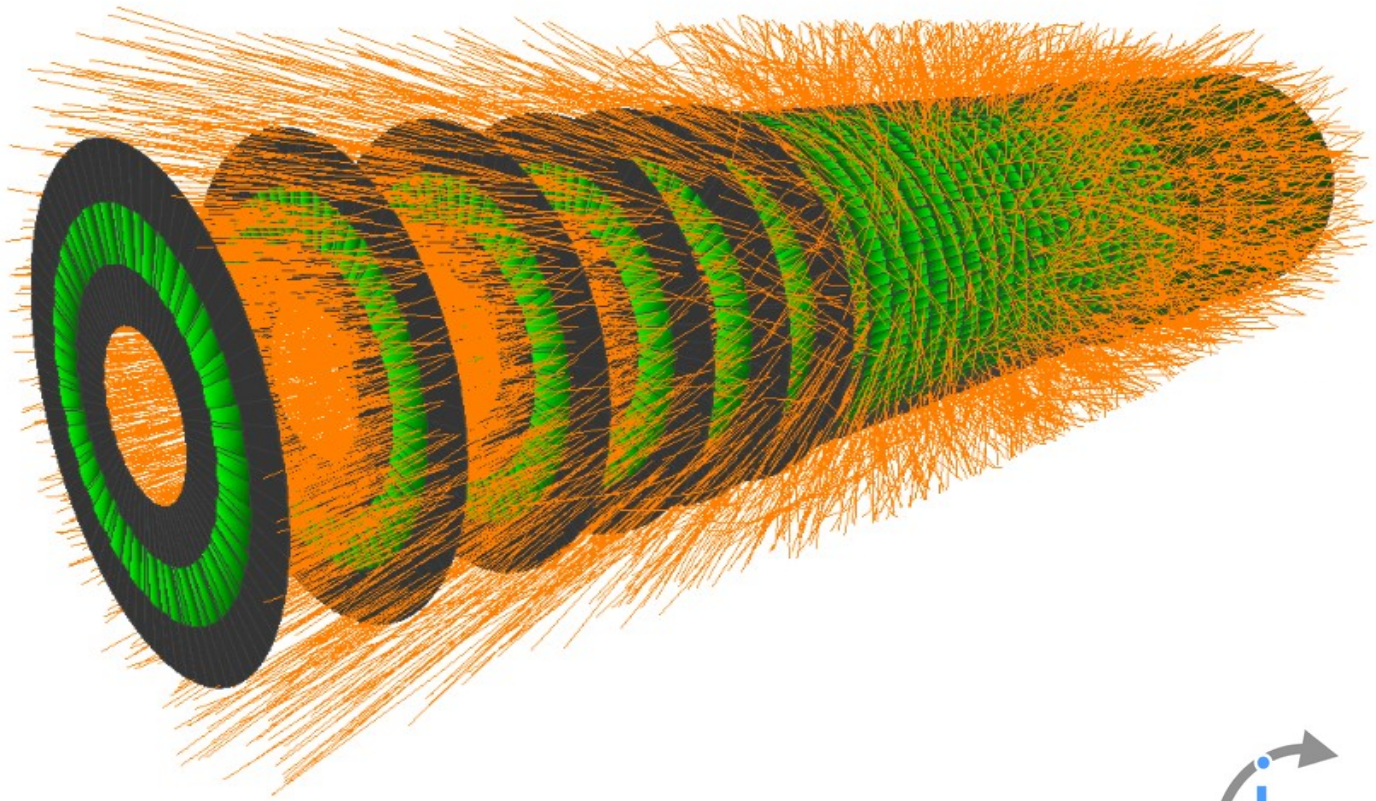


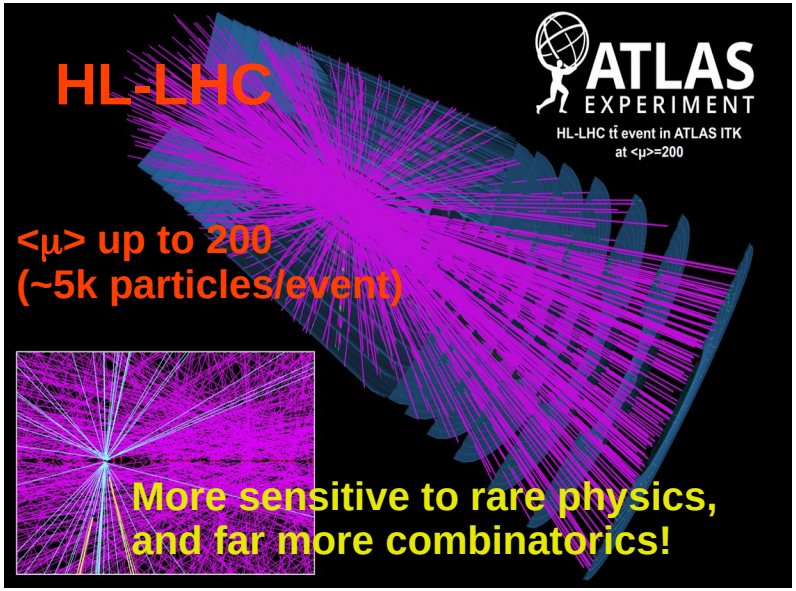
Track fitting/finding with acts

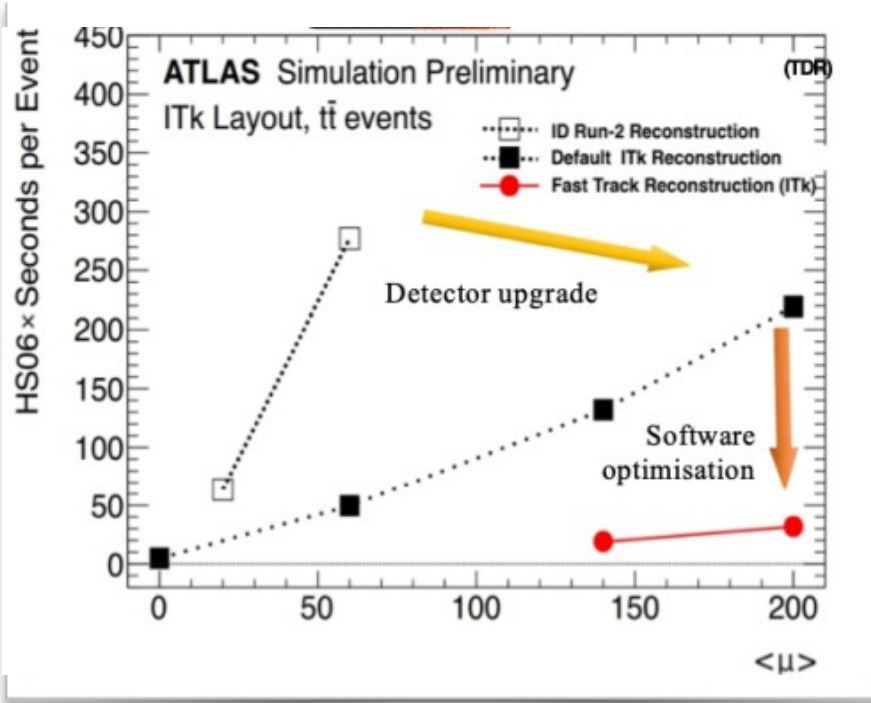
Xiaocong Ai, Heather Gray for the ACTS developers
UC Berkeley

IRIS-HEP topical meeting, Feb 19, 2020

The tracking challenge

- Much increased combinatorics at future hadron colliders
  - Great strain on CPU
  - More complex tracking environment
- All possible efforts to reduce memory footprint and speed up track reconstruction
  - Thread safety for concurrency
  - Performant software technology (data storage memory access)
  - Fast tracking strategy (ML techniques)
  - Benefit from multi-core and accelerators

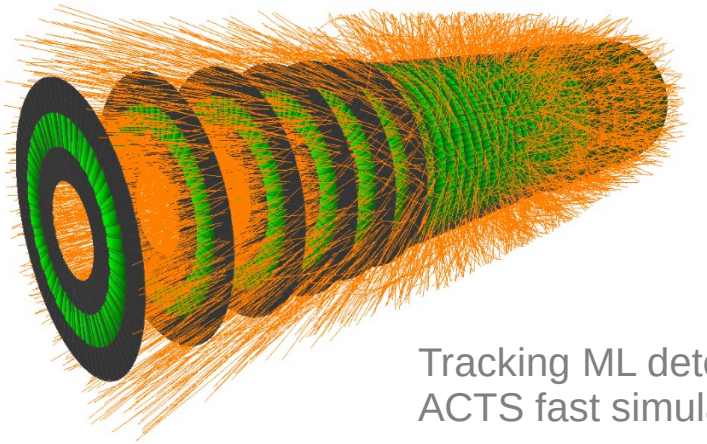




ACTS recap

- A tracking toolkit for future detectors and modern computing infrastructure
  - Based on ATLAS tracking code (well-tested, but difficult maintenance, thread-unsafe design etc.)
  - Improved performance
    - Modern C++ 17 concepts
    - Thread-safe design
    - Minimal dependencies for ease of integration
    - Ease of use and maintenance

http://acts.web.cern.ch/ACTS/



Tracking ML detector with ACTS fast simulation

- A testbed for new algorithms (ML techniques) and hardware architectures

ACTS components and functionalities

acts-core

- CI
- Core
- Fatras
- Plugins
- Tests

- EventData
- Fitter
- Geometry
- MagneticField
- Material
- Propagator
- Seeding
- Surfaces
- TrackFinder
- Utilities
- Vertexing

acts-fatras

Fast simulation engine, integrated into acts-core

- Geometry
  - Tracking geometry following ATLAS concepts
  - Plugins: DD4hep, Tgeo, GeoModel
- Event Data Model
  - Eigen-based tracking EDM
- Propagator
  - Integrate transport and navigation
  - Highly templated to avoid deep look-up
- Tracking features
  - Track seeding and track following (CombinatorialKalmanFilter)
  - Track fitting (KalmanFilter, Gaussian Sum Filter)
  - Vertex Finding&Fitting

Focus of this talk

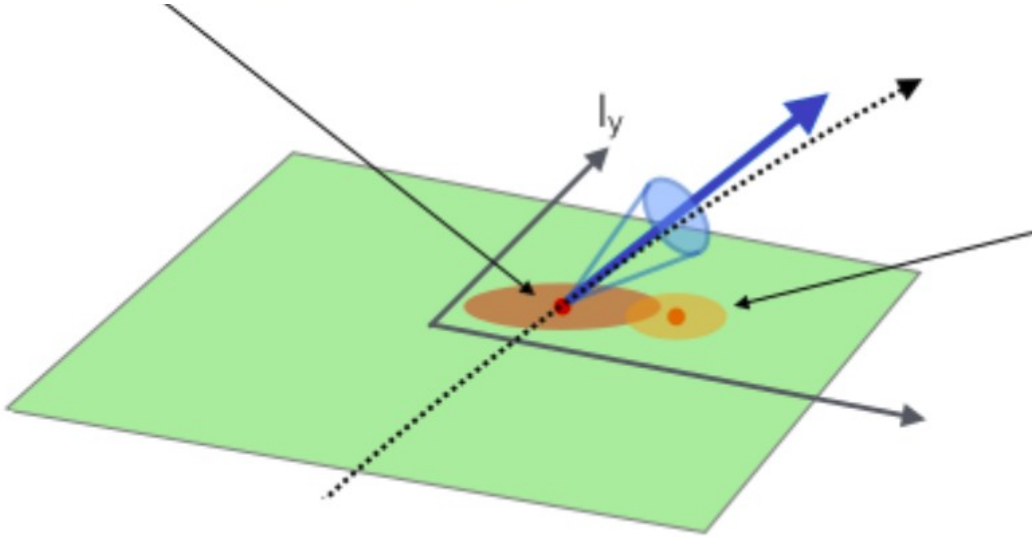
acts-framework

A light-weight Gaudi style framework for event processing, integration and concurrency test

TrackState EDM in acts

```
/// The parameter part
/// This is all the information that concerns the
/// the track parameterisation and the jacobian
/// It is enough to to run the track smoothing
struct {
  /// The predicted state
  std::optional<Parameters> predicted{std::nullopt};
  /// The filtered state
  std::optional<Parameters> filtered{std::nullopt};
  /// The smoothed state
  std::optional<Parameters> smoothed{std::nullopt};
  /// The transport jacobian matrix
  std::optional<Jacobian> jacobian{std::nullopt};
  /// The path length along the track - will help sorting
  double pathLength = 0.;
  /// chisquare
  double chi2 = 0;
} parameter;
```

- Design supports concept of KalmanFilter
- A container of possible information:
  - track parameters (predicted, filtered, smoothed)
  - measurements (uncalibrated, calibrated)
  - fitting quality
  - path Length
  - typeFlags (Parameter, Measurement, Outlier, Hole, Material etc.)




```
/// @brief Nested measurement part
/// This is the uncalibrated and calibrated measurement
/// (in case the latter is different)
struct {
  /// The optional (uncalibrated) measurement
  std::optional<SourceLink> uncalibrated{std::nullopt};
  /// The optional calibrabed measurement
  std::optional<FittableMeasurement<SourceLink>> calibrated{std::nullopt};
} measurement;
```

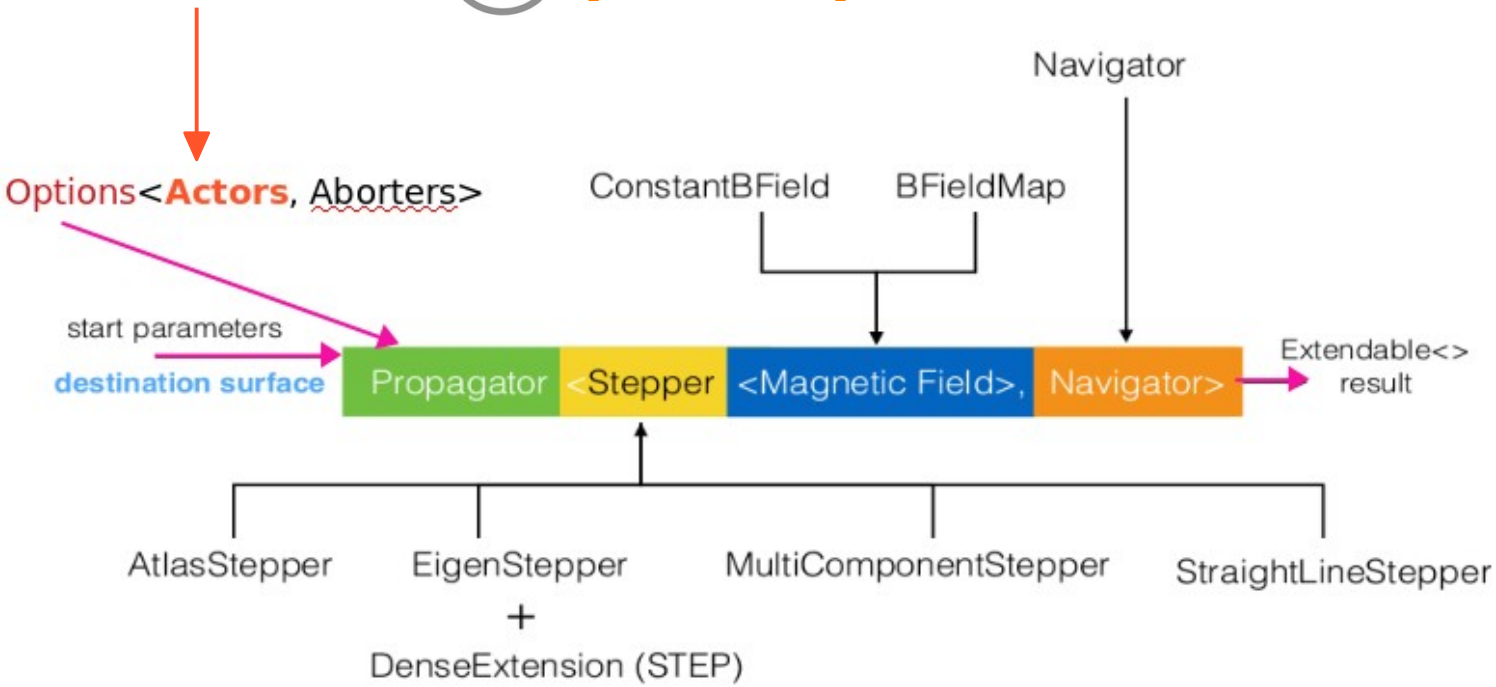
Track EDM in acts

- Eigen::Array based container (Acts::MultiTrajectory) of track states for efficient storage/ look-up
  - Keep track of capacity and size, provide read-write views into storage with index-tuples
    - Allows branching of tracks. Facilitate extension to Combinatorial Kalman Filter
    - Avoid storage duplication for shared measurements and parameters

```
// single items
using Coefficients = Eigen::Matrix<Scalar, Size, 1, Flags>;
using Covariance = Eigen::Matrix<Scalar, Size, Size, Flags>;
using CoefficientsMap = Eigen::Map<ConstIf<Coefficients, ReadOnlyMaps>>;
using CovarianceMap = Eigen::Map<ConstIf<Covariance, ReadOnlyMaps>>;
// storage of multiple items in flat arrays
using StorageCoefficients =
    GrowableColumns<Eigen::Array<Scalar, Size, Eigen::Dynamic, Flags>,
                    SizeIncrement>;
using StorageCovariance =
    GrowableColumns<Eigen::Array<Scalar, Size * Size, Eigen::Dynamic, Flags>,
                    SizeIncrement>;
```

```
struct IndexData {
  using IndexType = uint16_t;

  static constexpr IndexType kInvalid = UINT16_MAX;

  IndexType irefsurface = kInvalid;
  IndexType iprevious = kInvalid;
  IndexType ipredicted = kInvalid;
  IndexType ifiltered = kInvalid;
  IndexType ismoothed = kInvalid;
  IndexType ijacobian = kInvalid;
  IndexType iprojector = kInvalid;

  double chi2;
  double pathLength;
  TrackStateType typeFlags;

  IndexType iuncalibrated = kInvalid;
  IndexType icalibrated = kInvalid;
  IndexType icalibratedsourcelink = kInvalid;
  IndexType measdim = 0;
};
```

KalmanFilter in acts (Cont'd)



- Used as an Actor in propagator
- Templated on propagator, KalmanUpdater, KalmanSmoother, outlier finder, measurement calibrator etc.
- Support hole search and outlier rejection during the fitting

```
template <typename propagator_t, typename updater_t = VoidKalmanUpdater,
          typename smoother_t = VoidKalmanSmoother,
          typename outlier_finder_t = VoidOutlierFinder,
          typename calibrator_t = VoidMeasurementCalibrator,
          typename input_converter_t = VoidKalmanComponents,
          typename output_converter_t = VoidKalmanComponents>
class KalmanFitter {
```

KalmanFilter in acts

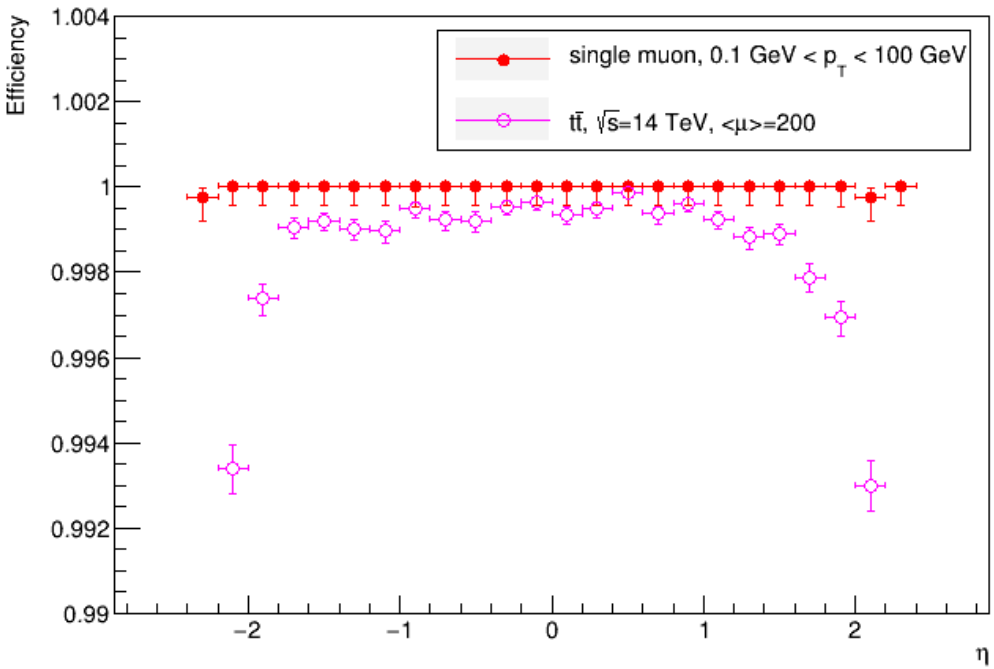
- One fit invocation takes:
  - Starting parameter (with large uncertainty)
  - Measurements (unsorted source links, e.g. clusters) on a known trajectory
  - User-defined fitting options
    - Algorithm context (GeometryContext, MagneticFieldContext, CalibrationContext)
    - Target surface
    - Outlier search criteria
    - Material effects
    - Smoothing mode
- Fitting result:
  - An Acts::MultiTrajectory object with single track entry point
  - Fitted parameter

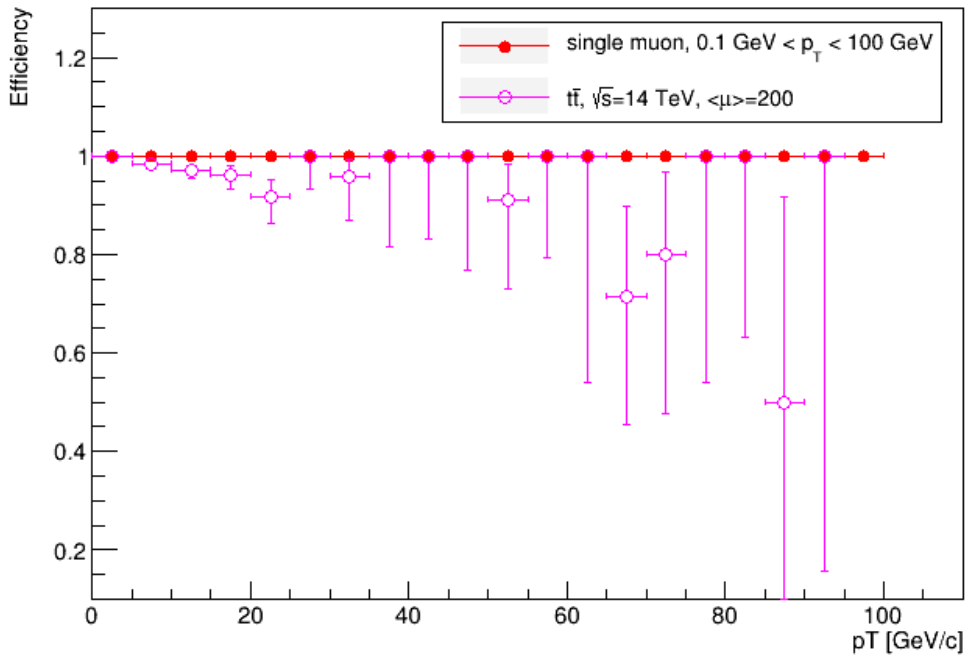
```
/// PropagatorOptions with context
///
/// @param gctx The goemetry context for this fit
/// @param mctx The magnetic context for this fit
/// @param cctx The calibration context for this fit
/// @param olCfg The config for the outlier finder
/// @param rSurface The reference surface for the fit to be expressed at
/// @param mScattering Whether to include multiple scattering
/// @param eLoss Whether to include energy loss
/// @param bwdFiltering Whether to run backward filtering as smoothing
KalmanFitterOptions(std::reference_wrapper<const GeometryContext> gctx,
                    std::reference_wrapper<const MagneticFieldContext> mctx,
                    std::reference_wrapper<const CalibrationContext> cctx,
                    const OutlierFinderConfig& ofCfg,
                    const Surface* rSurface = nullptr,
                    bool mScattering = true, bool eLoss = true,
                    bool bwdFiltering = false)
    : geoContext(gctx),
      magFieldContext(mctx),
      calibrationContext(cctx),
      outlierFinderConfig(ofCfg),
      referenceSurface(rSurface),
      multipleScattering(mScattering),
      energyLoss(eLoss),
      backwardFiltering(bwdFiltering) {}
```

KF fitting efficiency

- Fitting efficiency is defined as: $\frac{N_{reco}}{N_{truth}}$
  - ~100% fitting efficiency with |η|<1.5

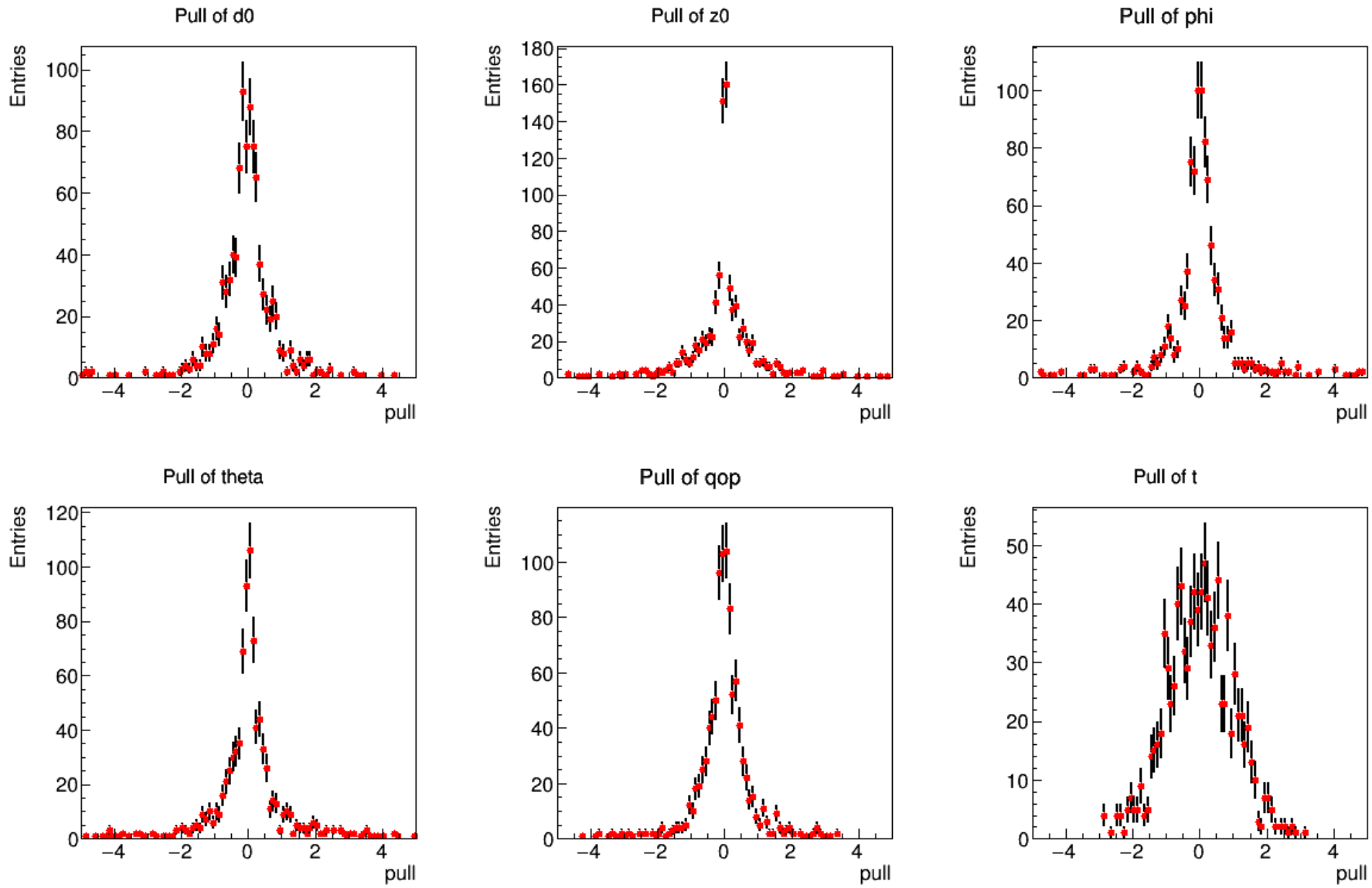
TrackML detector, ATLAS B field





KF fitting resolution

- Single muon, $0.1<p_T<100$ GeV, $|\eta|<2.4$
- TrackML detector, ATLAS B field

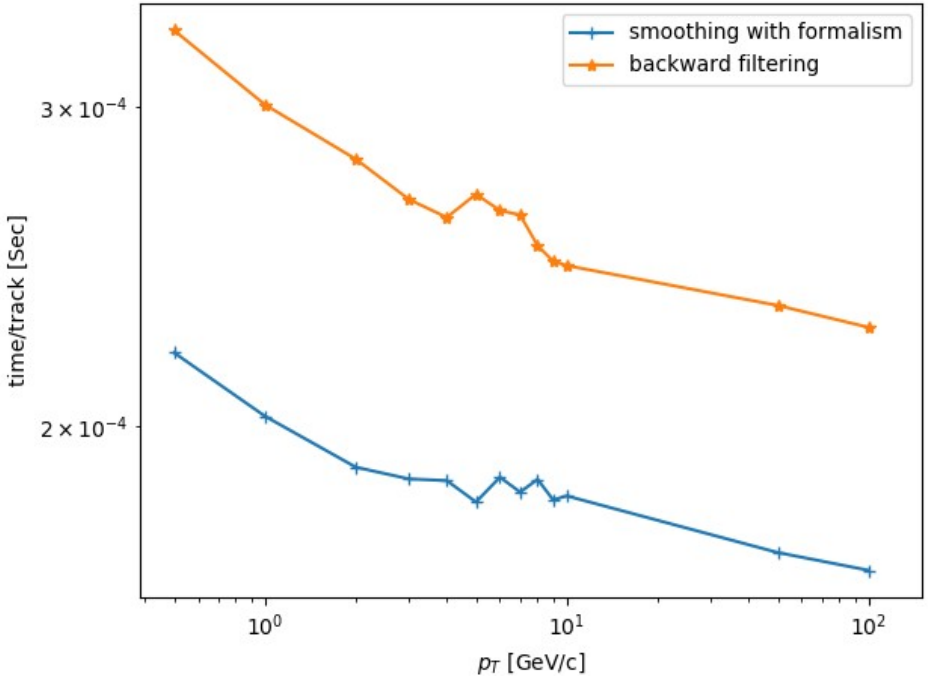


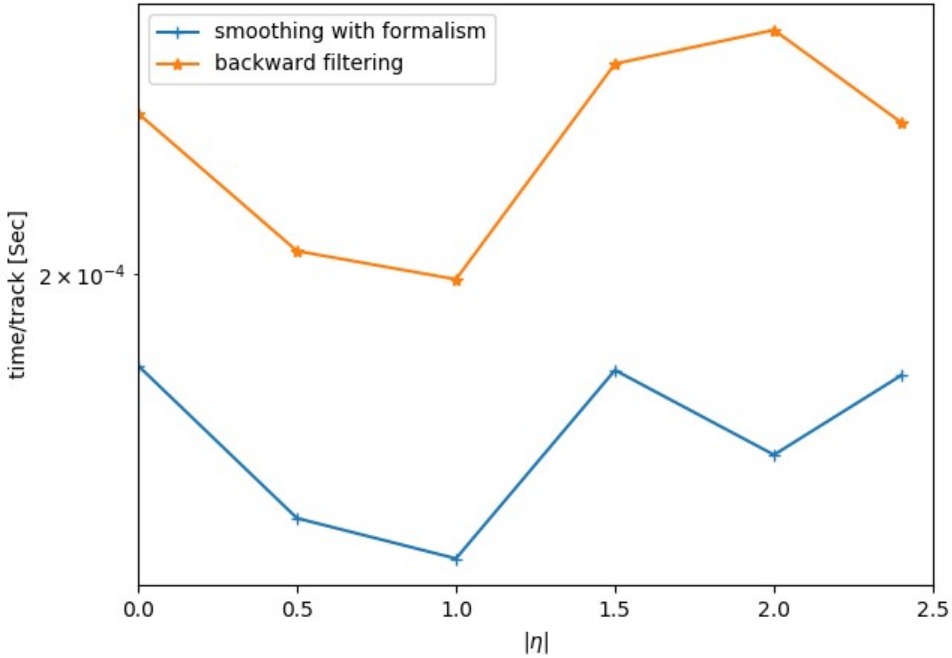
Little material for TrackML detector, hence no great justification for gaussian pulls

KF timing test

- Running with two different approach for smoothing
  - So-called 'smoothing-matrix' formalism
    - Faster, relies on surface-wise Jacobian in forward filtering
  - Run another Kalman filter in backward direction
    - Slower, but more stable

- Single muon, $0.1<p_T<100$ GeV, $|\eta|<2.4$
- TrackML detector, ATLAS B field

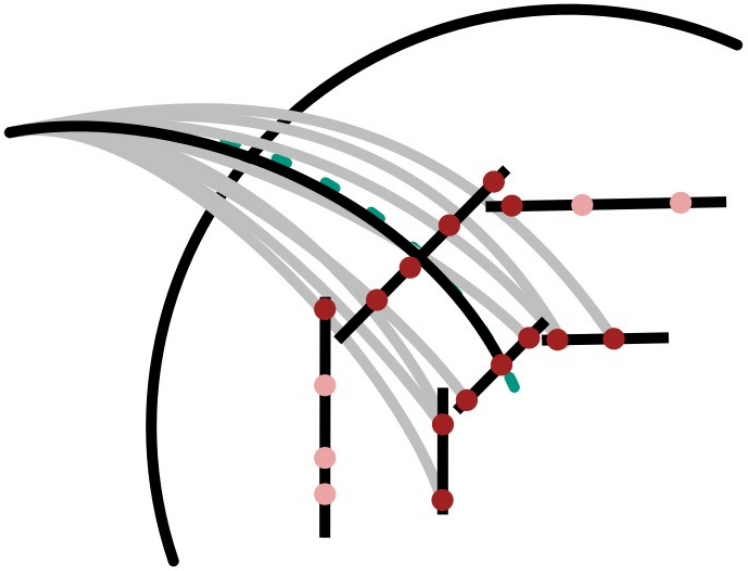




CombinatorialKalmanFilter in acts (Cont'd)

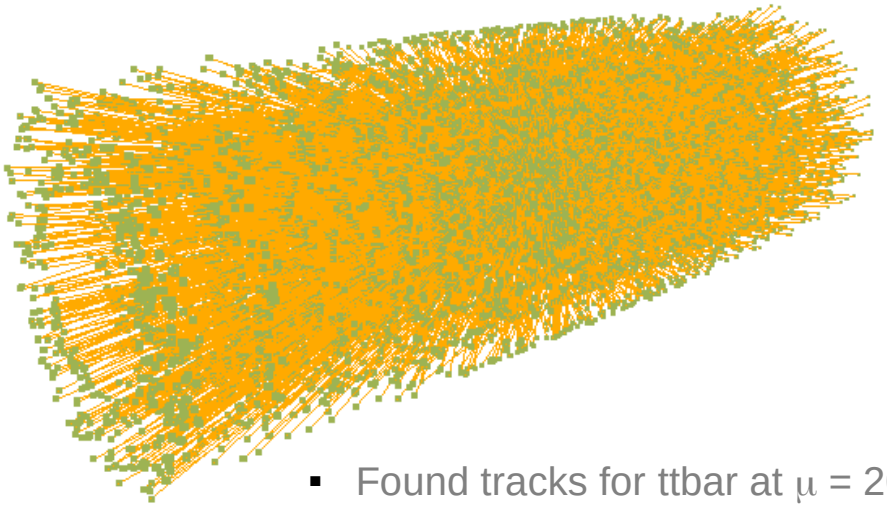
- KalmanFilter-extended track following algorithm
  - Simultaneous tracking fitting and finding
  - Allow track branching if more than one compatible measurement found
    - Support user-defined measurement search and branching strategy (e.g. detector-specific chi2 cutoff and maximum branches)
  - Allow stopping of bad quality branch

```
/// @brief CombinatorialKalmanFilter implementation of Acts as a plugin
///
/// to the Propgator
///
/// @tparam propagator_t Type of the propagation class
/// @tparam updater_t Type of the kalman updater class
/// @tparam smoother_t Type of the kalman smoother class
/// @tparam source_link_selector_t Type of the source link selector class
/// @tparam branch_stopper_t Type of the branch stopper class
/// @tparam calibrator_t Type of the calibrator class
/// @tparam input_converter_t Type of the input converter class
/// @tparam output_converter_t Type of the output converter class
...
template <typename propagator_t, typename updater_t = VoidKalmanUpdater,
          typename smoother_t = VoidKalmanSmoother,
          typename source_link_selector_t = VoidSourceLinkSelector,
          typename branch_stopper_t = VoidBranchStopper,
          typename calibrator_t = VoidMeasurementCalibrator,
          typename input_converter_t = VoidKalmanComponents,
          typename output_converter_t = VoidKalmanComponents>
class CombinatorialKalmanFilter {
  ...
```



CombinatorialKalmanFilter in acts

- One track finding invocation takes:
  - Starting parameter (currently using smeared truth parameter, will be taken from seeding)
  - All measurements in a single event
  - User-defined track finding options
    - Algorithm context
    - Target surface
    - Source link selection criteria
    - Material effects
    - Option for smoothing



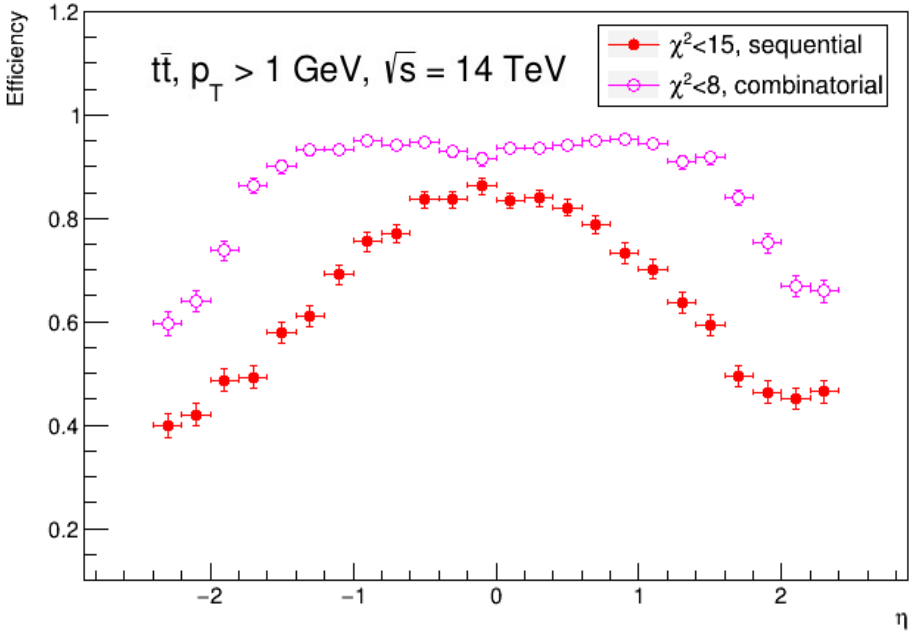
- Found tracks for ttbar at μ = 200 (~6k truth particles, ~70k truth hits)
- TrackML detector, ATLAS B fileld

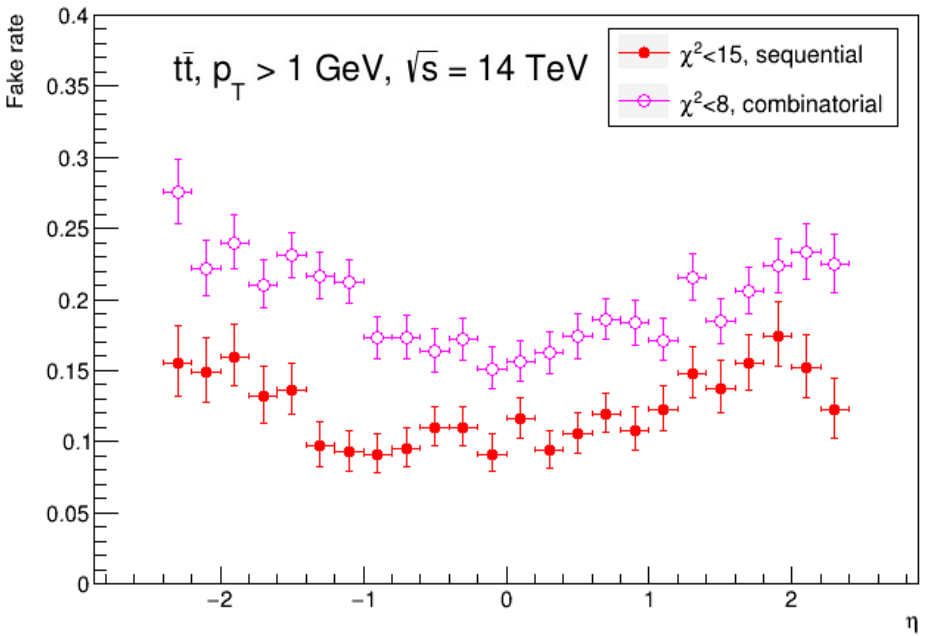
- Track finding result:
  - An Acts::MultiTrajectory object with a list of track entry points for all found tracks
  - (Optional) fitted parameter

CKF efficiency&fake rate

- Measurement selection based on χ^2 window:
  - Sequential: branching with only the best measurement
  - Combinatorial: branching with all compatible measurements
- Track selection: $n_{Hits}>5$, $|\eta|<2.4$
- Reco-truth matching criteria: $\frac{N_{hits}(Majority)}{N_{hits}(Total)}>0.5$
- Track finding efficiency is defined as: $\frac{N_{reco}(selected, matched)}{N_{truth}(selected)}$
- Fake rate is defined as: $\frac{N_{reco}(selected, unmatched)}{N_{reco}(selected)}$

TrackML detector, ATLAS B fileld

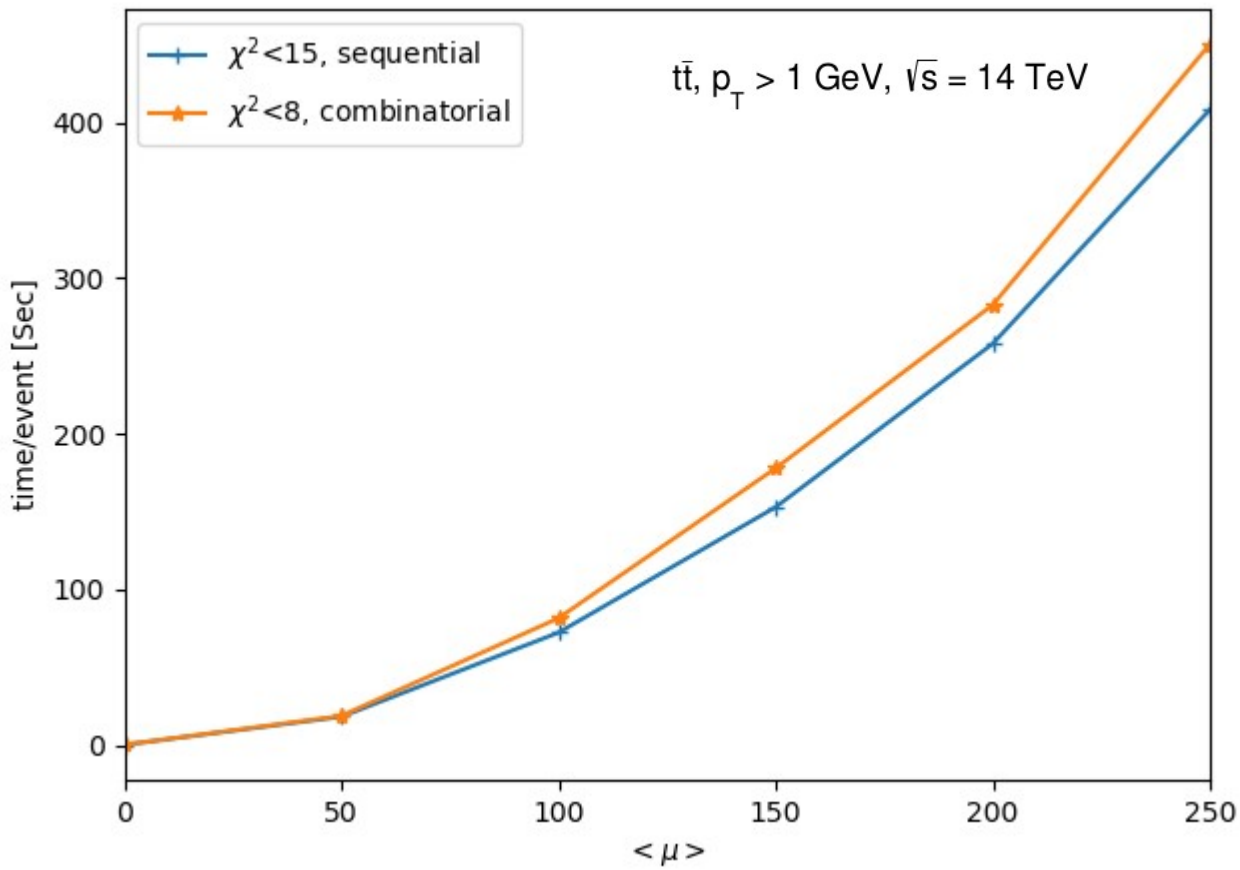




CKF timing test

- Timing varies with measurement selection strategy (which requires detector-specific tuning)
  - Planned tests with more realistic detectors (eg. OpenData, ATLAS ITk)
- Investigate possible speed-up with fast tracking geometry

TrackML detector, ATLAS B fileld



Contextual alignment and calibration in acts

- ACTS uses an AlgorithmContext object to support on-the-fly event-dependent changes of alignment, calibration and magnetic field

```
size_t                  algorithmNumber;  ///< Unique algorithm identifier
size_t                  eventNumber;      ///< Unique event identifier
WhiteBoard&             eventStore;       ///< Per-event data store
Acts::GeometryContext geoContext;         ///< Per-event geometry context
Acts::MagneticFieldContext
                          magFieldContext;  ///< Per-event magnetic Field context
Acts::CalibrationContext calibContext;     ///< Per-event calbiration context
```

- Validated concept of contextual alignment and calibration

Propagation tests with contextual **alignment**

(Different alignment every single event, $n_{threads}$ = 4)

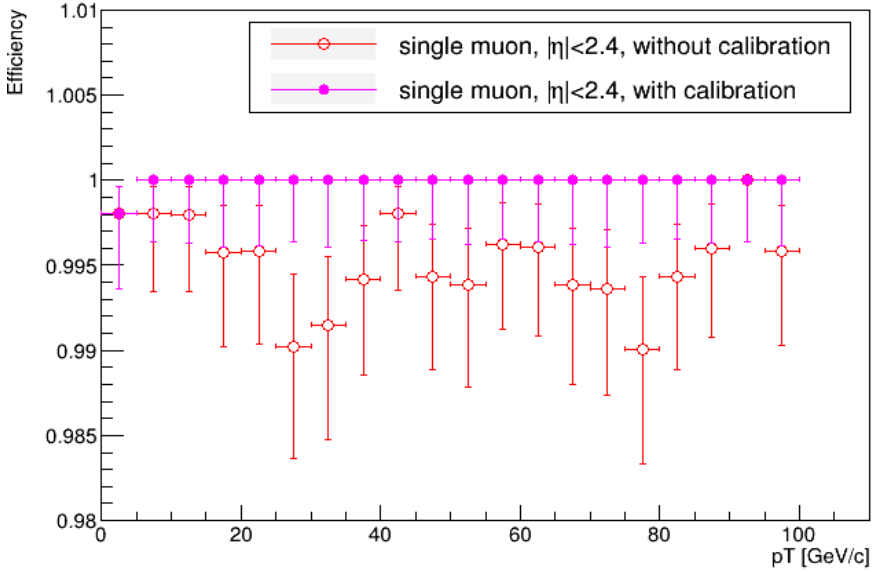
```
salzburg$ export ACTSFW_NUM_THREADS=1
salzburg$ ./ACTFWAlignablePropagationExample -n10 --prop-ntests 1000 --bf-values 0 0 2 --output-root 1
12:49:10    Sequencer      INFO      Added context decorator GeometryRotationDecorator
12:49:10    Sequencer      INFO      Added service RandomNumbersSvc
12:49:10    Sequencer      INFO      Appended algorithm PropagationAlgorithm
12:49:11    Sequencer      INFO      Added writer RootPropagationStepsWriter
12:49:11    Sequencer      INFO      Starting event loop for
12:49:11    Sequencer      INFO        1 services
12:49:11    Sequencer      INFO        0 readers
12:49:11    Sequencer      INFO        1 writers
12:49:11    Sequencer      INFO        1 algorithms
12:49:11    Sequencer      INFO      Run the event loop
12:49:11    Sequencer      INFO      start event 0
12:49:12    Sequencer      INFO      event 0 done
12:49:12    Sequencer      INFO      start event 1
12:49:13    Sequencer      INFO      event 1 done
12:49:13    Sequencer      INFO      start event 2
12:49:14    Sequencer      INFO      event 2 done
12:49:14    Sequencer      INFO      start event 3
12:49:15    Sequencer      INFO      event 3 done
12:49:15    Sequencer      INFO      start event 4
12:49:16    Sequencer      INFO      event 4 done
12:49:16    Sequencer      INFO      start event 5
12:49:17    Sequencer      INFO      event 5 done
12:49:17    Sequencer      INFO      start event 6
12:49:19    Sequencer      INFO      event 6 done
12:49:19    Sequencer      INFO      start event 7
12:49:19    Sequencer      INFO      event 7 done
12:49:19    Sequencer      INFO      start event 8
12:49:20    Sequencer      INFO      event 8 done
12:49:20    Sequencer      INFO      start event 9
12:49:22    Sequencer      INFO      event 9 done
12:49:22    Sequencer      INFO      Running end-of-run hooks of writers and services
```

```
salzburg$ export ACTSFW_NUM_THREADS=4
12:51:19    Sequencer      INFO      start event 0
12:51:19    Sequencer      INFO      start event 5
12:51:19    Sequencer      INFO      start event 8
12:51:19    Sequencer      INFO      start event 7
12:51:20    Sequencer      INFO      event 7 done
12:51:20    Sequencer      INFO      start event 2
12:51:21    Sequencer      INFO      event 8 done
12:51:21    Sequencer      INFO      start event 9
12:51:21    Sequencer      INFO      event 5 done
12:51:21    Sequencer      INFO      start event 6
12:51:21    Sequencer      INFO      event 0 done
12:51:21    Sequencer      INFO      start event 1
12:51:22    Sequencer      INFO      event 2 done
12:51:22    Sequencer      INFO      start event 3
12:51:23    Sequencer      INFO      event 9 done
12:51:23    Sequencer      INFO      start event 4
12:51:23    Sequencer      INFO      event 6 done
12:51:23    Sequencer      INFO      event 1 done
12:51:23    Sequencer      INFO      event 3 done
12:51:24    Sequencer      INFO      event 4 done
```

12 seconds → 5 seconds

Track fitting test with contextual **calibration**

(Different calibration every 10 events, $n_{threads}$=8)



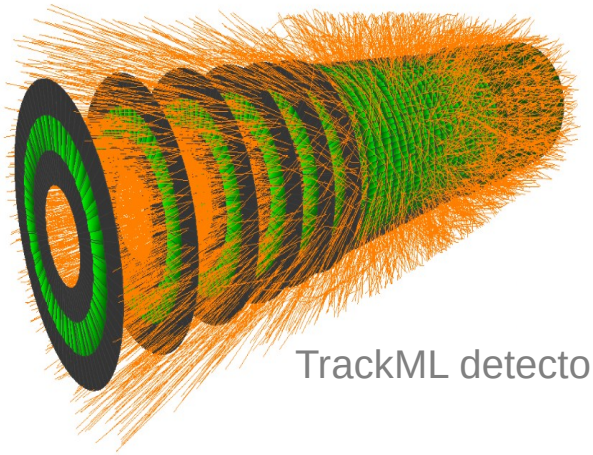
Much more progress in acts

More details in A. Salzburg's slides

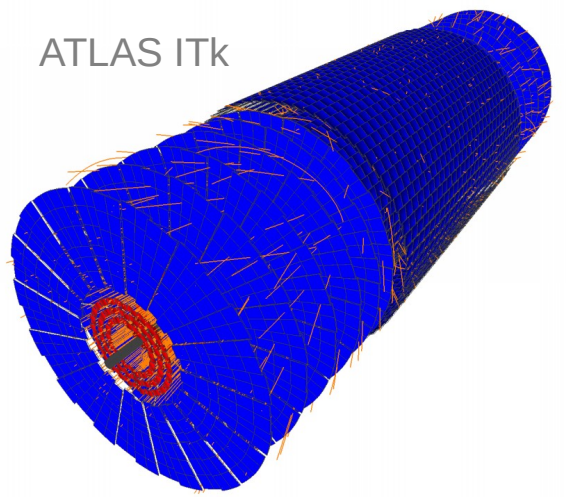
- Close-to complete Vertexing Finding&Finding (B. Schlag)
  - Currently performance validated against Athena/Tracking
- Prototype of Gaussian Sum Filter as non-gaussian extension of KF (J. Zhang)
- Improvement of Fatras simulation (M. Kiehn, F. Klimpel)
  - Energy loss & multiple scattering validated
  - Hadronic interaction currently re-parameterised
  - Foreseen use of Geant4 for particle decay
- Optimization of repository infrastructure (A. Salzburg, M. Kiehn, P. Gessinger-Befurt , F. Klimpel et al.)
  - Improved EDM, geometry, stepper etc.
- Exploration of parallelism with multi-core and accelerator
  - Intra-event parallelism (G. Mania, N. Styles)
  - Seed finder with accelerator (V. Pascuzzi, C. Legett et. al)

Detector geometry with acts

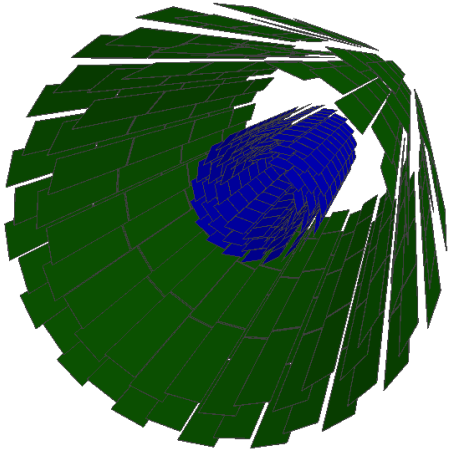
- Detector implemented:
  - ATLAS ID+Calo, ATLAS ITK, sPhenix silicon, OpenData, TrackML. FCC-hh
- On-going/planned implementation:
  - ATLAS Muon System
  - Belle-II silicon (+ Drift Chamber)
  - sPhenix TPC



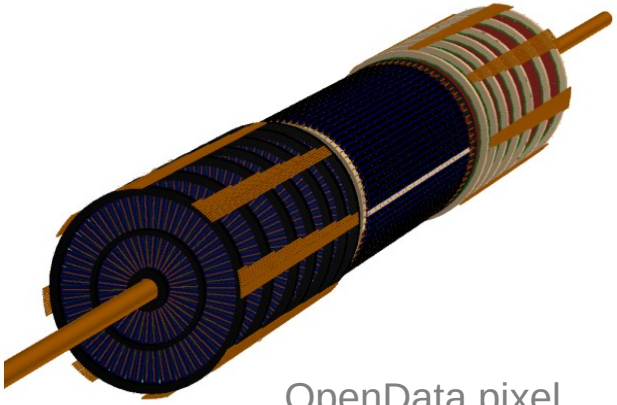
TrackML detector



ATLAS ITk



sPhenix silicon



OpenData pixel

Summary

- Large increase in track multiplicity at future colliders needs high performance tracking software
- ACTS provides a framework-independent and detector-independent tracking toolkit with inherent thread-safety
  - Well-validated tracking infrastructure (geometry, propagator, EDM)
  - Expanding tracking toolkit
    - Pattern recognition, fitting, vertexing etc.
  - New experiment detector implementation with world-wide collaboration
- Work In Progress with ACTS:
  - Extended geometry concept for various HEP detector
  - Validation of tracking tools with different detector
  - Exploration of innovative tracking techniques and speed-up with hardware architectures

backup

ACTS members/developers

Members with access to **acts** 39

Andreas Salzburger @asalzbur
Given access 3 years ago

Julia Hrdinka @jhrdinka
Given access 3 years ago

Noemi Calace @ncalace
Given access 3 years ago

Valentin Volkl @vavolkl
Given access 3 years ago

Moritz Kiehn @msmk
Given access 2 years ago

Robert Johannes Langenberg @rlangenb
Given access 2 years ago

Hadrien Benjamin Grasland @hgraslan
Given access 2 years ago

ATS Jenkins @atsjenkins
Given access 2 years ago

Stewart Martin-Haugh @smh
Given access 2 years ago

Karolos Potamianos @karolos
Given access 2 years ago

Edward Moyse @emoyse
Given access 2 years ago

Nicholas Styles @nstyles
Given access 2 years ago

Dmitry Emeliyanov @demelian
Given access 2 years ago

Paolo Calafiura @calaf
Given access 2 years ago

Benedikt Hegner @hegner
Given access 2 years ago

Sarka Todorova @nova
Given access 2 years ago

Wolfgang Liebig @liebig
Given access 2 years ago

Shaun Roe @sroe
Given access 2 years ago

Vincenzo Innocente @innocent
Given access 2 years ago

David Rousseau @droussea
Given access 2 years ago

Markus Elsing @elsing
Given access 2 years ago

Frank-Dieter Gaede @fgaede
Given access 2 years ago

David Chamont @chamont
Given access 2 years ago

Tobias Golling @golling
Given access 2 years ago

Felice Pantaleo @fpantale
Given access 2 years ago

Paul Gessinger-Befurt @pagessin
Given access 1 year ago

Marco Rovere @rovere
Given access 1 year ago

Fabian Klimpel @fklimpel
Given access 1 year ago

John Smith @jrsmith
Given access 11 months ago

Shih-Chieh Hsu @schsu
Given access 11 months ago

Heather Gray @hgray
Given access 10 months ago

Lauren Alexandra Tompkins @tompkins
Given access 10 months ago

Jin Zhang @jinz
Given access 9 months ago

Bastian Schlag @bschlag
Given access 8 months ago

Jovan Mitrevski @jmitrevs
Given access 7 months ago

Xiaocong Ai @xai It's you
Given access 7 months ago

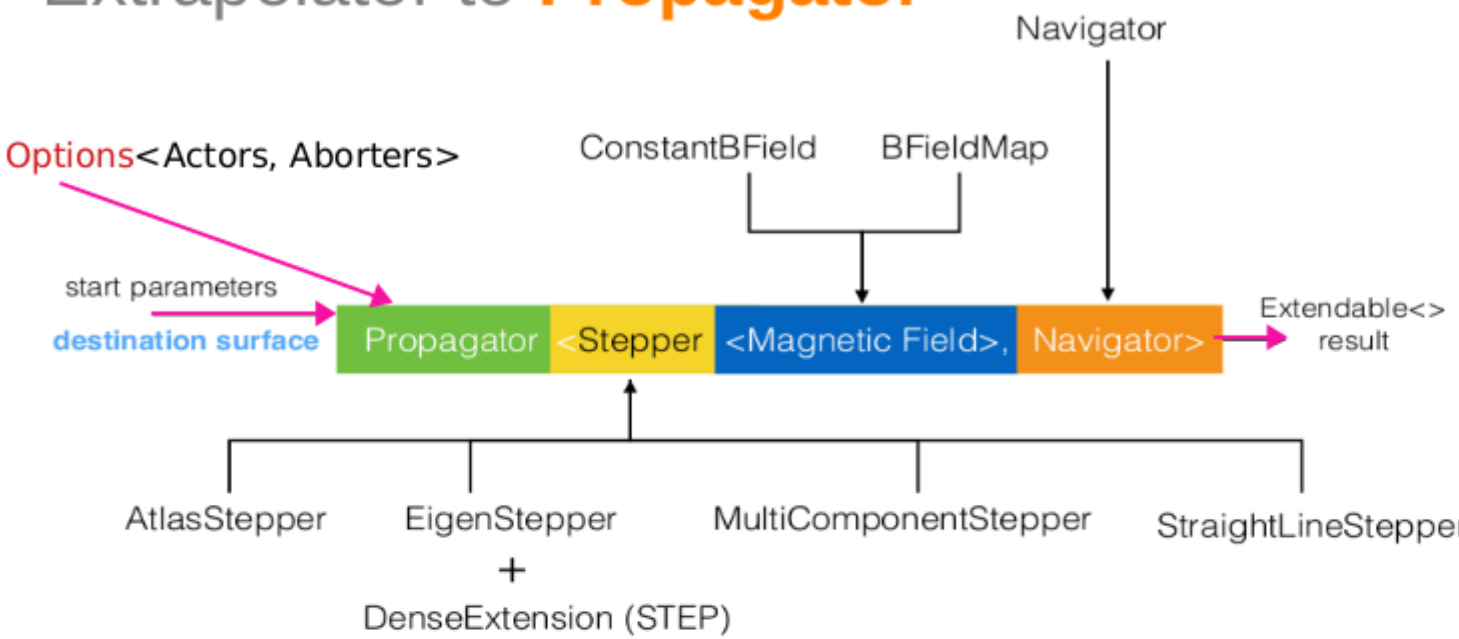
Simone Pagan Griso @spagan
Given access 5 months ago

Gang Zhang @gang
Given access 3 months ago

Tim Adye @adye
Given access 4 weeks ago

ATLAS
LHCb
FCCSW
CEPC
FASER

~~Extrapolator~~ to Propagator



- ATLAS Tracking SW:
  - Distinction between propagator (transport) and extrapolator (transport, navigation & material effects)

- Integrate transport and navigation into propagator (template design to avoid deep look-up)
  - Stepper integrates equation of motion and provides step estimation
    - **EigenStepper** (ATLAS adaptive Runge-Kutta stepper rewritten using Eigen) as the primary integrator
    - Extension for transport in dense volumes, e.g. calorimeter
  - Navigator determines navigation step size to reach detector
  - Options allows for custom implementation at integration step
    - Actors: e.g. **MaterialInteractor**, **KalmanActor**
    - Aborters: abort conditions