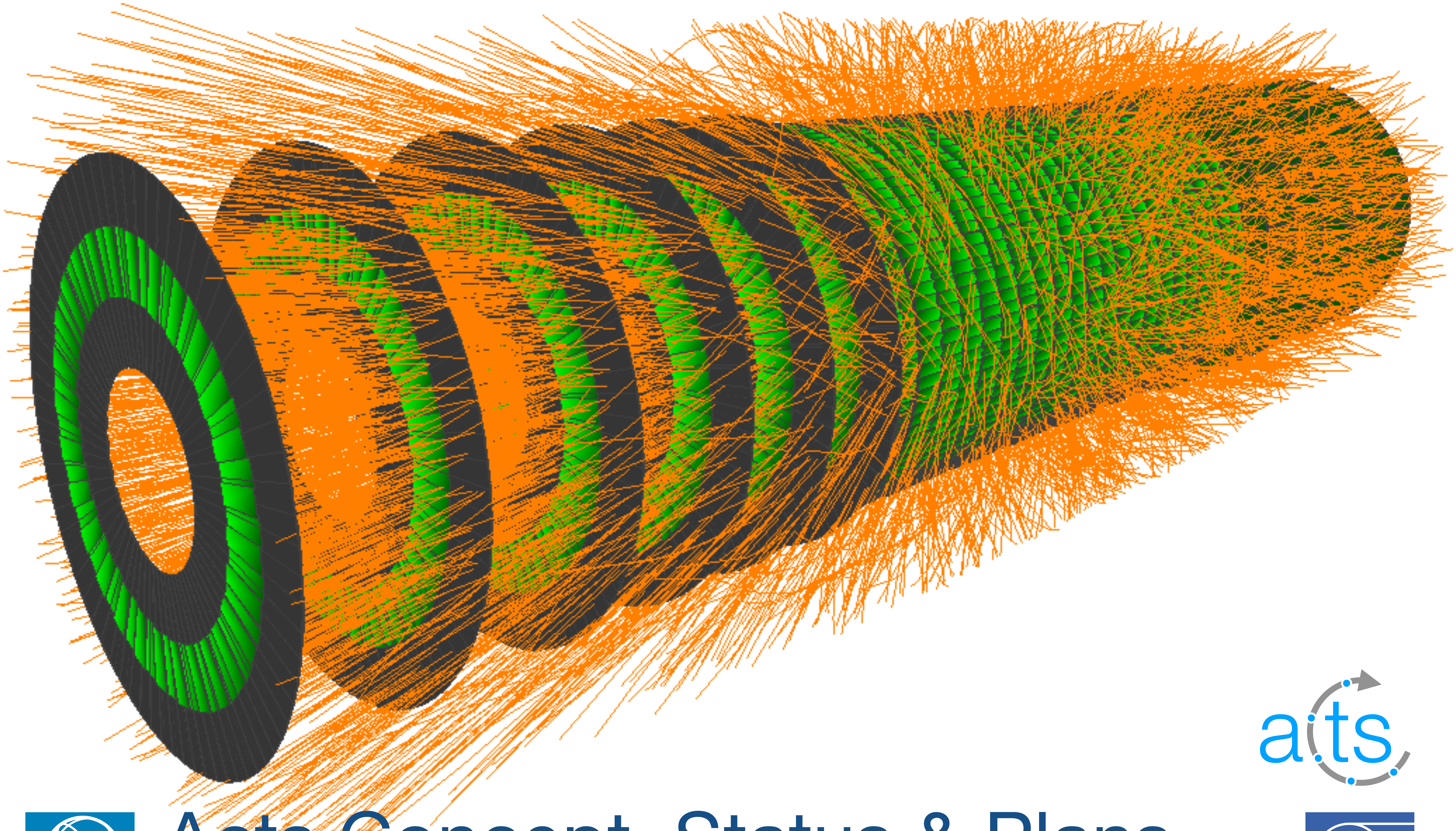


Acts Concept, Status & Plans

https://cern.ch/acts

A. Salzburger (CERN)

https://gitlab.cern.ch/acts/acts-framework/merge_requests/193

Chapter One Yesterday & Today

Motivation ATLAS Tracking SW

Common set of Tools and interfaces

Tracking					
Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	Extrapolation (TrkExtrapolation)	Fitting (TrkFitter)	Calibration, general (TrkTools)	Alignment (TrkAlignment)

Detector specific extension

InnerDetector		
EDM (InDetRecEvent)	Geometry (InDetTrackingGeometry)	Tools (InDetRecTools)

Calorimeter
Geometry (CaloTrackingGeometry)

MuonSpectrometer		
EDM (MuonRecEvent)	Geometry (MuonTrackingGeometry)	Tools (MuonRecTools)

Fast track simulation extension

ISF_Fatras		
EDM (ISF_FatrasEvent)	Geometry (ISF_FatrasDetDescrTools)	Tools (ISF_FatrasRecoTools)

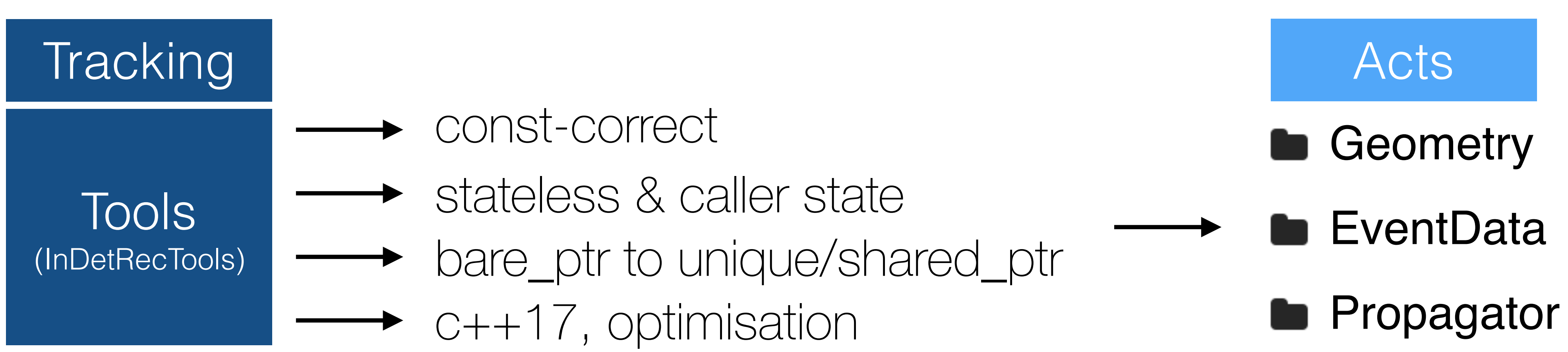
Evolution From *ATLAS* to Acts

Review

- code usage, code quality, memory usage and execution speed
- check for readiness for concurrent code execution

Update, documentation, integration & testing

- update to **`C++17`** standard,
- simplify, documentation
- Integration in Acts
- Unit tests and regression tests against ATLAS code



Review ATLAS Tracking SW

ATLAS Tracking modules

Tracking					
Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	Extrapolation (TrkExtrapolation)	Fitting (TrkFitter)	Calibration, general (TrkTools)	Alignment (TrkAlignment)

- Type safety throughout the code
- Top level agnostic code for EDM, tools and conditions
- Fast simulation capability
- Highly configurable
- Partly dynamic size EDM
- Deep level of virtual interfacing
- Lazily initialised (mostly removed in LS-1)

Review ATLAS Tracking SW

ATLAS Tracking modules

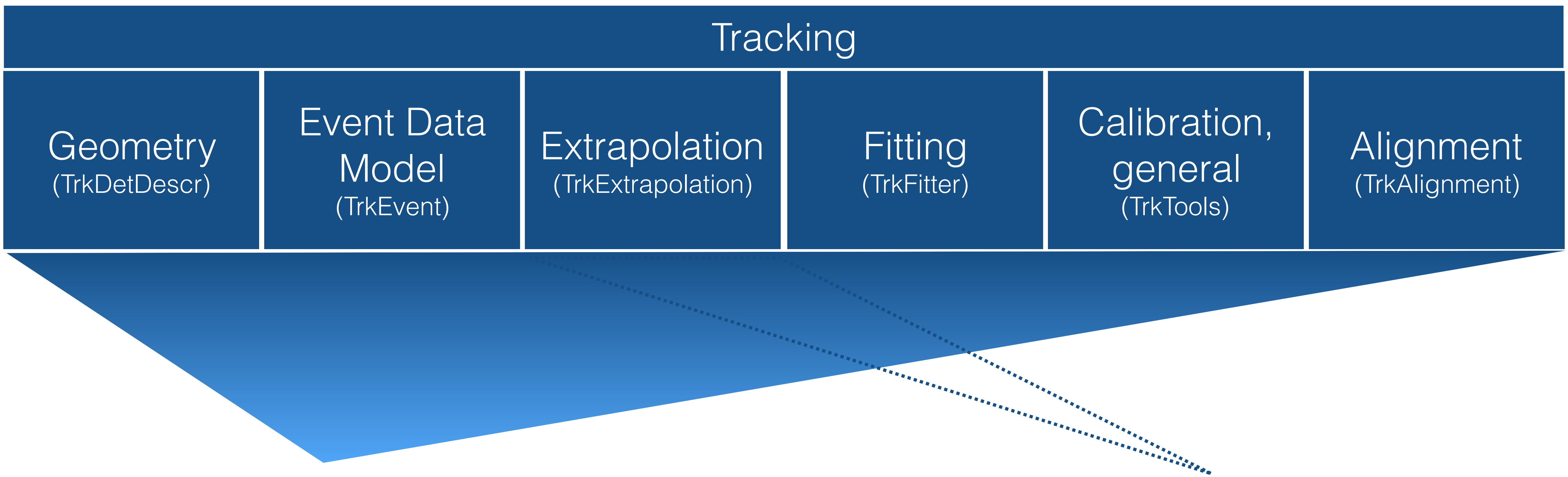
Tracking					
Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	Extrapolation (TrkExtrapolation)	Fitting (TrkFitter)	Calibration, general (TrkTools)	Alignment (TrkAlignment)

- [x] Type safety throughout the code
- [x] Top level agnostic code for EDM, tools and conditions
- [x] Fast simulation capability
- [x] Highly configurable
- [ ] ~~Partly dynamic size EDM~~
- [ ] ~~Deep level of virtual interfacing~~
- [ ] ~~Lazily initialised (mostly removed in LS-1)~~

Acts

Code from ATLAS to Acts

ATLAS Tracking modules



acts-core

acts-fatras

acts-framework

lightweight MT framework
to test integration

Renaming from ATLAS to Acts

ATLAS Tracking modules to Acts modules

Tracking

Geometry (TrkDetDescr)	Event Data Model (TrkEvent)	Extrapolation (TrkExtrapolation)	Fitting (TrkFitter)	Calibration, general (TrkTools)	Alignment (TrkAlignment)

acts

acts-core

Surfaces	Material	EventData	Propagator	Fitter	Vertexing	Utilities
Geometry	MagneticField	Seeding				

Chapter Two design choices & modules

Design choices

Compiling vs. Interfacing

- Eigen is a header-only library
- shifted some of the lifting from interfaces to compile-time checks

  Interface is checked via C++ concept classes & type trait asserts

```
template <typename T>
using step_size_t = decltype(std::declval<T>().stepSize);

// clang-format off
  template <typename S>
  constexpr bool StepperStateConcept
    = require<has_member<S, cov_transport_t, bool>,
              has_member<S, cov_t, BoundSymMatrix>,
              has_member<S, nav_dir_t, NavigationDirection>,
              has_member<S, path_accumulated_t, double>,
              has_member<S, step_size_t, detail::ConstrainedStep>
             >;
// clang-format on

// clang-format off
  template <typename S, typename state = typename S::State>
    struct StepperConcept {
      constexpr static bool state_exists = exists<state_t, S>;
      static_assert(state_exists, "State type not found");
      constexpr static bool jacobian_exists = exists<jacobian_t, S>;
      static_assert(jacobian_exists, "Jacobian type not found");
      constexpr static bool covariance_exists = exists<covariance_t, S>;
      static_assert(covariance_exists, "Covariance type not found");
      constexpr static bool bound_state_exists = exists<bound_state_t, S>;
      static_assert(bound_state_exists, "BoundState type not found");
```

- time critical components are resolved compile-time

  full stack compiles in a few minutes not moved are single-call modules, e.g. geometry building

Some level of dispatching will be necessary

- Pre-compiled modules ready for usage

Design & development choices

Heavily relying on CI checks & code review

convention

- Configuration is done by a nested **`Config`** struct
- Re-entrance is done by a nested **`State`** struct
- Runtime options are done by a **`Options`** struct

```
namespace Acts {
  /// doxygen documentation
  class WorkHorse {
   /// @struct Config of this Horse
    struct Config {
       int horseNumber = 0; ///< the passed path so far
    };
    /// @struct Cache for the WorkHorse
    struct State {
       float accumulatedPath = 0.; ///< the passed path so far
    };
    /// @struct Cache for the WorkHorse
    struct Options {
       bool runBackwards = true; ///< switch horse direction per run
    };
    /// method to make the horse run
    /// @param hState - cache tracker for this horse
    /// @param coords - place where the horse should run to
    /// @return a result, horse may drop dead if max path is reached
    const RunResult run(State& hState, const Vector3D& coords, const Options& opts) const;
 };
}
```

Geometry

Tracking

Geometry
(TrkDetDescr)

- ☑ Proxy mechanism for GeoModel sensitive elements
  (allows for alignment following)
- ☑ TrackingGeometry with intrinsic navigation
- ☑ Detector agnostic geometry description
- ☑ Surface based description for Propagation
- ☑ Surface <u>and</u> Volume based material description
- ~~☐ Distinction between free and non-free surfaces (unclear ownership)~~

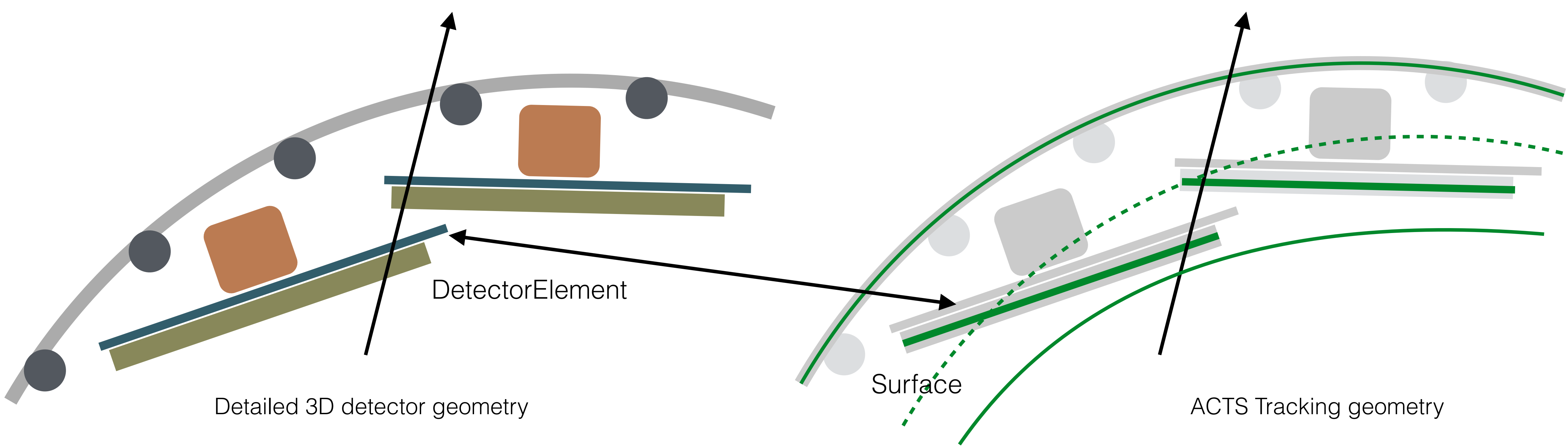
Acts

Proxy mechanism

Geometry binding via DetectorElementBase

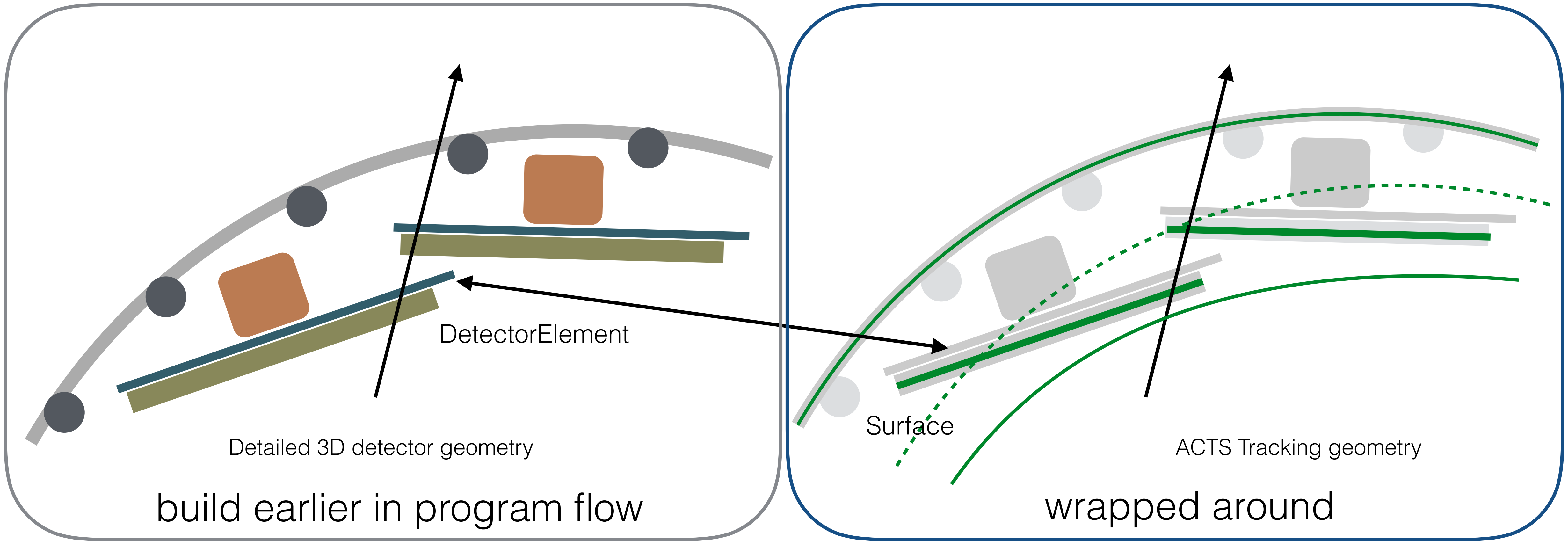
```
namespace Acts {
  /// doxygen documentation
  class DetectorElementBase {
    /// the according represented surface
    virtual const Surface& associatedSurface() const = 0;
  };
}
```

```
class MyDetectorElement {
   /// @copydoc DetectorElementBase::asscociatedSurface
   const PlaneSurface& associatedSurface() const;
 };
```



Geometry data locality

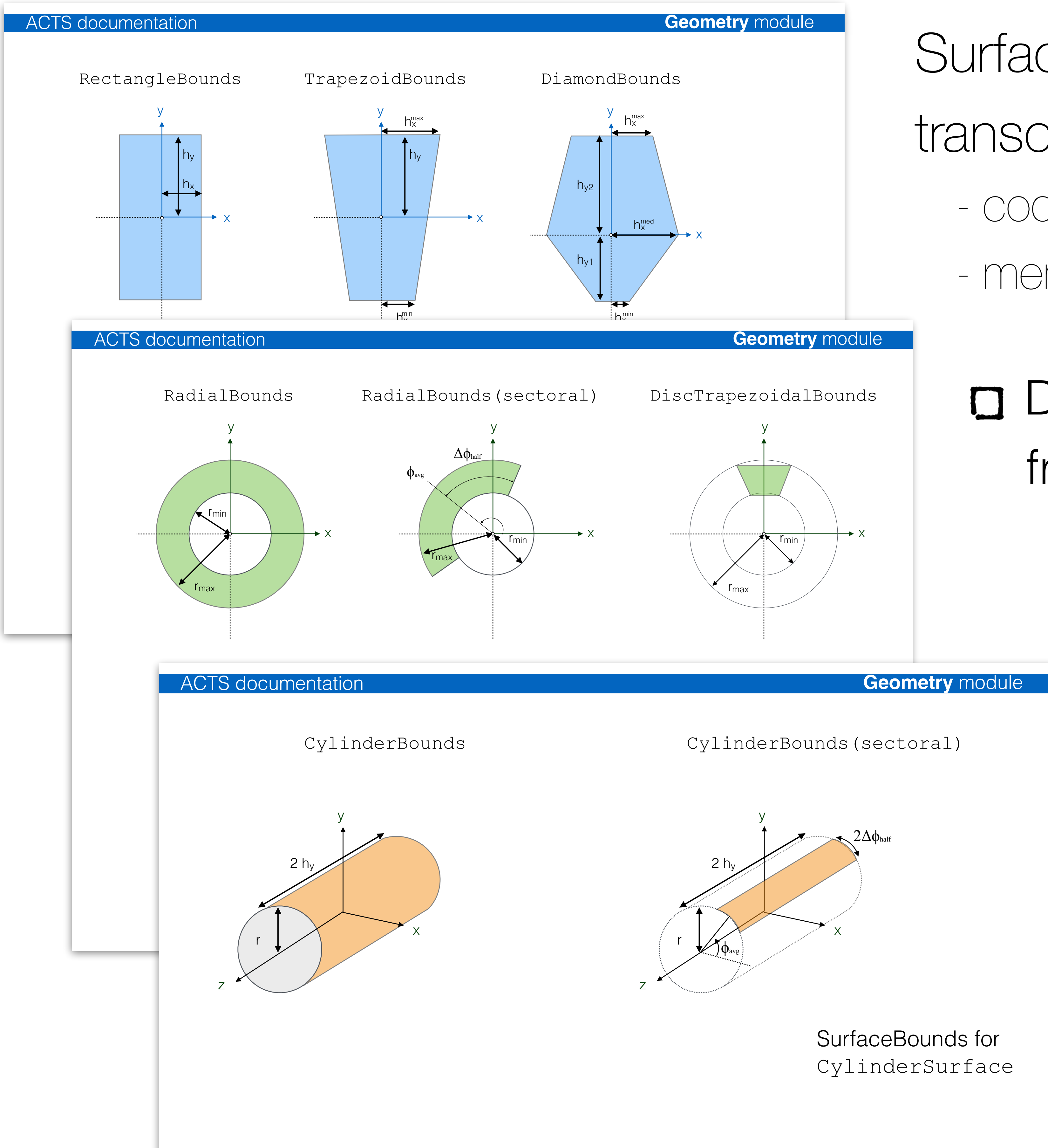
Geometry building ATLAS:



Tests showed that many cache misses in `transform()` lookup

- as surfaces are built as proxies and forward the positioning information the transform() call points to different memory
- In a multi-threaded application this is particularly tricky (see later)

Acts Surface classes



Surface classes are largely transcribed from ATLAS SW

- code simplified
- memory structure updated

- [ ] ~~Distinction between free and non-free surfaces~~

Tracking

memory management in ATLAS "by hand"

```
// Constructor with local arguments - uses global <-> local for parameters
template<int DIM,class T,class S>
ParametersT<DIM,T,S>::ParametersT(double loc1,
                                  double loc2,
                                  double phi,
                                  double theta,
                                  double qop,
                                  const S& surface,
                                  AmgSymMatrix(DIM)* cov):
  ParametersBase<DIM,T>(),
  m_parameters(),
  m_covariance(cov),
  m_position(),
  m_momentum(),
  m_surface(nullptr),
  m_chargeDef(sgn(qop))
{

  m_surface.reset((surface.isFree() ? surface.clone() : &surface));
  // check qoverp is physical
  double p = 0.;
  if(qop != 0)
    p = fabs(1./qop);
  else
  {
    // qop is unphysical. No momentum meas
    p = INVALID_P;
    qop = INVALID_QOP;
  }
```

Acts

geometry objects can only be constructed at **shared_ptr**

objects can hand back their **shared_ptr**

```
public:
 /// Destructor
 virtual ~Surface();

 /// Factory for producing memory managed instances of Surface.
 /// Will forward all parameters and will attempt to find a suitable
 /// constructor.
 template <class T, typename... Args>
 static std::shared_ptr<T> makeShared(Args&&... args) {
   return std::shared_ptr<T>(new T(std::forward<Args>(args)...));
 }

 /// Retrieve a @c std::shared_ptr for this surface (non-const version)
 ///
 /// @note Will error if this was not created through the @c makeShared factory
 ///       since it needs access to the original reference. In C++14 this is
 ///       undefined behavior (but most likely implemented as a @c bad_weak_ptr
 ///       exception), in C++17 it is defined as that exception.
 /// @note Only call this if you need shared ownership of this object.
 ///
 /// @return The shared pointer
 std::shared_ptr<Surface> getSharedPtr();
```

Geometry memory management

Acts

```
namespace Acts {

class DetectorElementBase;
class SurfaceBounds;
class ISurfaceMaterial;
class Layer;
class TrackingVolume;

/// @class Surface
///
/// @brief Abstract Base Class for tracking surfaces
///
/// The Surface class builds the core of the Acts Tracking Geom
/// All other geometrical objects are either extending the surf
/// are built from it.
///
/// Surfaces are either owned by Detector elements or the Track
/// in which case they are not copied within the data model obj
///
class Surface : public virtual GeometryObject,
                public std::enable_shared_from_this<Surface> {
 public:
  /// @enum SurfaceType
  ///
  /// This enumerator simplifies the persistency & calculations,
  /// by saving a dynamic_cast, e.g. for persistency
  enum SurfaceType {
    Cone = 0,
    Cylinder = 1,
    Disc = 2,
    Perigee = 3,
    Plane = 4,
    Straw = 5,
    Curvilinear = 6,
    Other = 7
  };
```

C++ | Utilities library | Dynamic memory management | **std::enable_shared_from_this**

std::enable_shared_from_this

Defined in header <memory>

`template< class T > class enable_shared_from_this;` (since C++11)

`std::enable_shared_from_this` allows an object `t` that is currently managed by a `std::shared_ptr` named `pt` to safely generate additional `std::shared_ptr` instances `pt1, pt2, ...` that all share ownership of `t` with `pt`.

Publicly inheriting from `std::enable_shared_from_this<T>` provides the type `T` with a member function `shared_from_this`. If an object `t` of type `T` is managed by a `std::shared_ptr<T>` named `pt`, then calling `T::shared_from_this` will return a new `std::shared_ptr<T>` that shares ownership of `t` with `pt`.

Member functions

(constructor)	constructs an `enable_shared_from_this` object (protected member function)
(destructor)	destroys an `enable_shared_from_this` object (protected member function)
operator=	returns a reference to `this` (protected member function)
shared_from_this	returns a `shared_ptr` which shares ownership of `*this` (public member function)
weak_from_this (C++17)	returns the `weak_ptr` which shares ownership of `*this` (public member function)

C++11 feature allowing **`shared_ptr`** access from plain pointers, solved geometry/event data binding

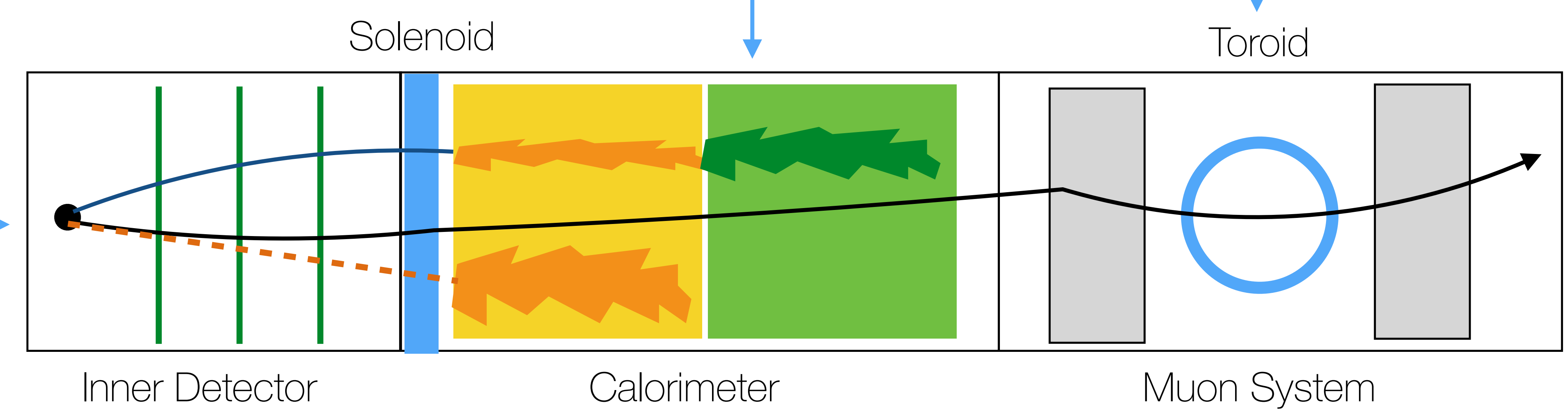
Access for calculations via **`const`** `object`**`&`**
In order to optimise access speed

TrackingGeometry

Most concepts from ATLAS TrackingGeometry adopted in ACTS

- Volumes with static layer configuration
- Volumes with dense material
- Volumes with floating objects

Not fully implemented yet



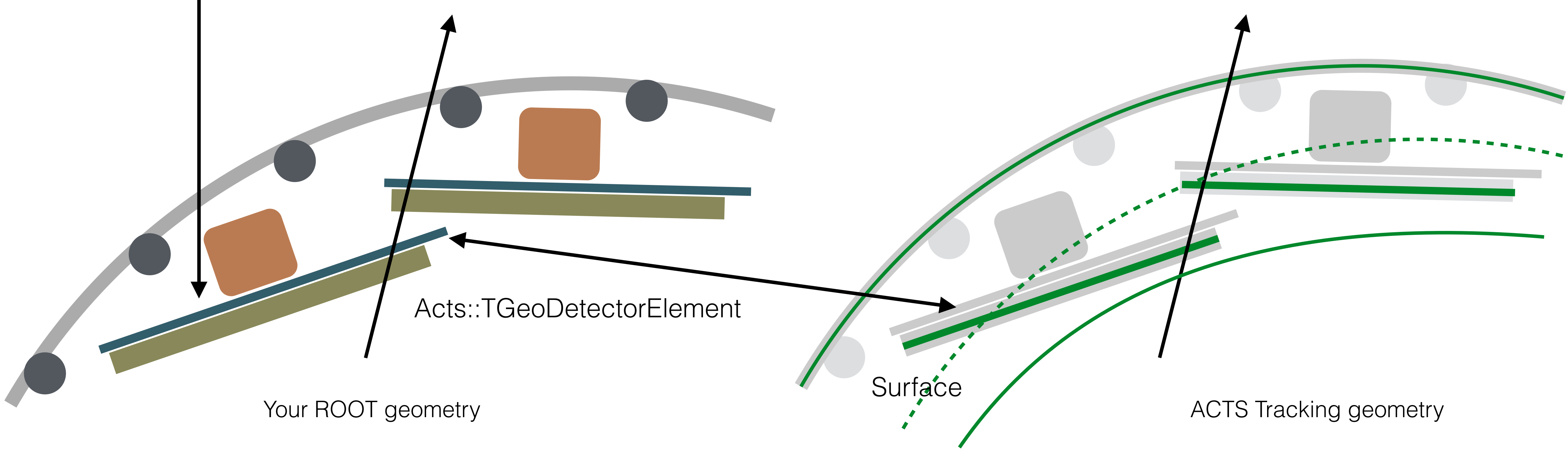
ATLAS ID setup with very similar navigation

support for a STEP extension (see later) and new cell navigation

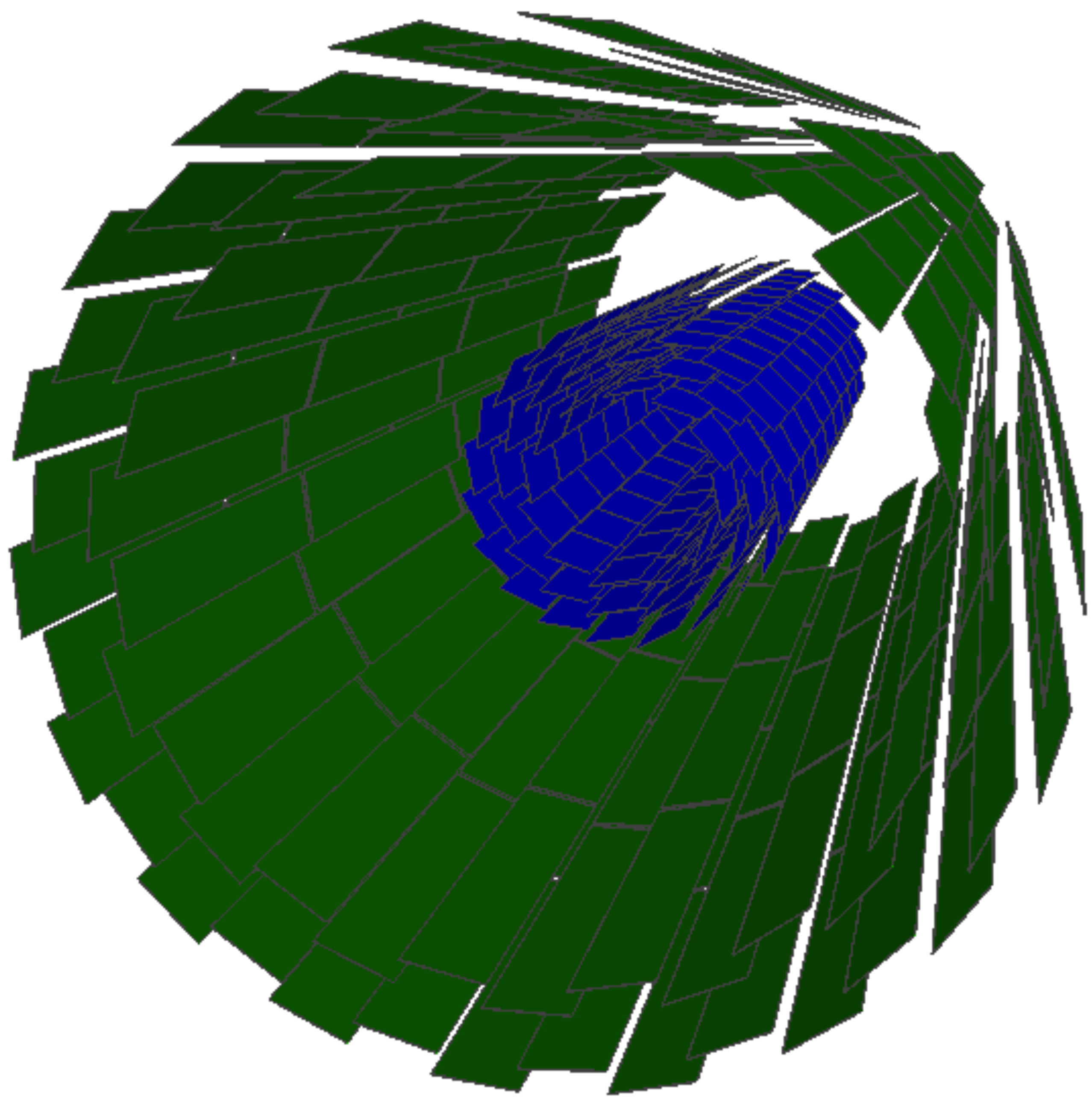
Geometry Customise

```
ACTFWTGeoGeometryExample -n1 -l0
--geo-tgeo-filename=sPhenix.root
--geo-tgeo-worldvolume="World"
--geo-subdetectors MVTX Silicon
--geo-tgeo-nlayers=0 0
--geo-tgeo-clayers=1 1
--geo-tgeo-players=0 0
--geo-tgeo-claryernames MVTX siactive
--geo-tgeo-cmodulenames MVTXSensor siactive
--geo-tgeo-cmoduleaxes xzy yzx
--geo-tgeo-clayersplits 5. 10.
--output-obj true
```

orientation wrt Acts



Geometry Customise

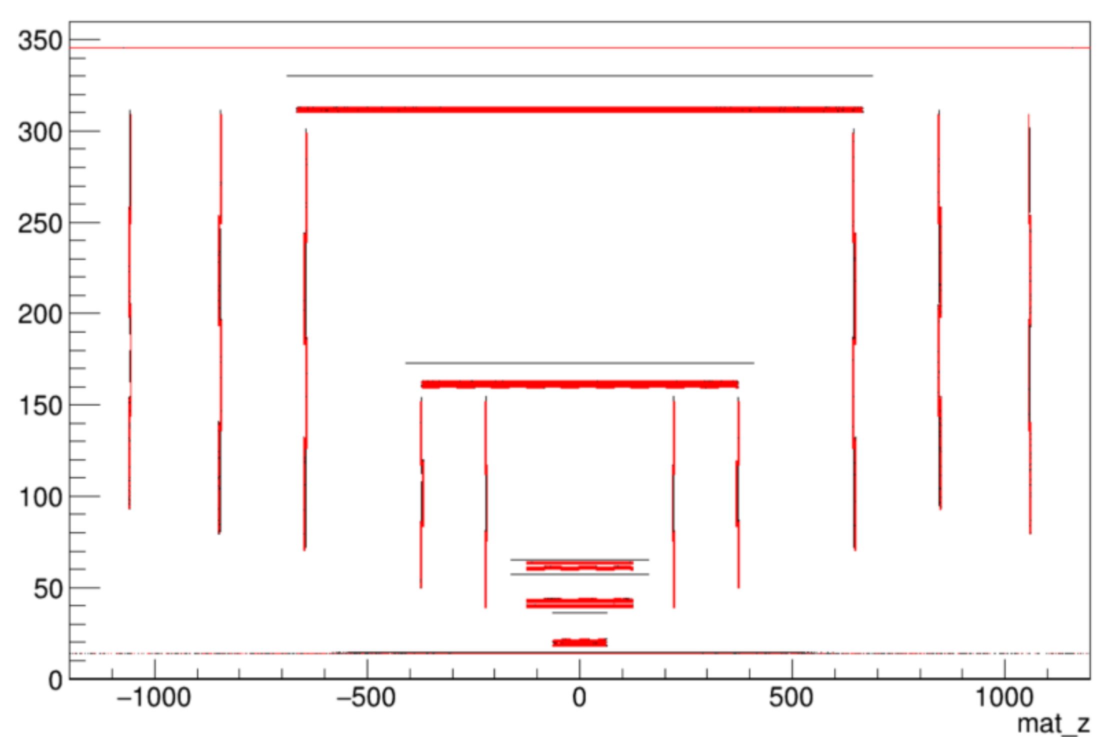


Geometry Material description

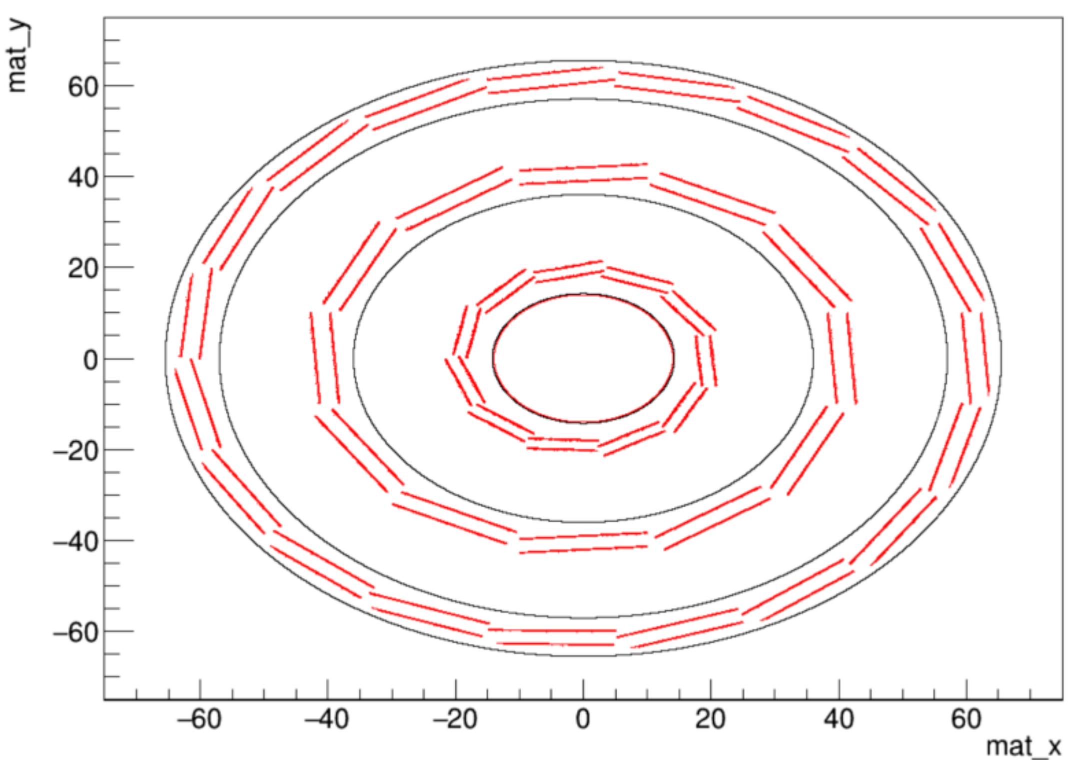
ACTS comes with a built-in material description

- material on static layer configuration
- material with dense material

Automated procedure to map material from simulation input

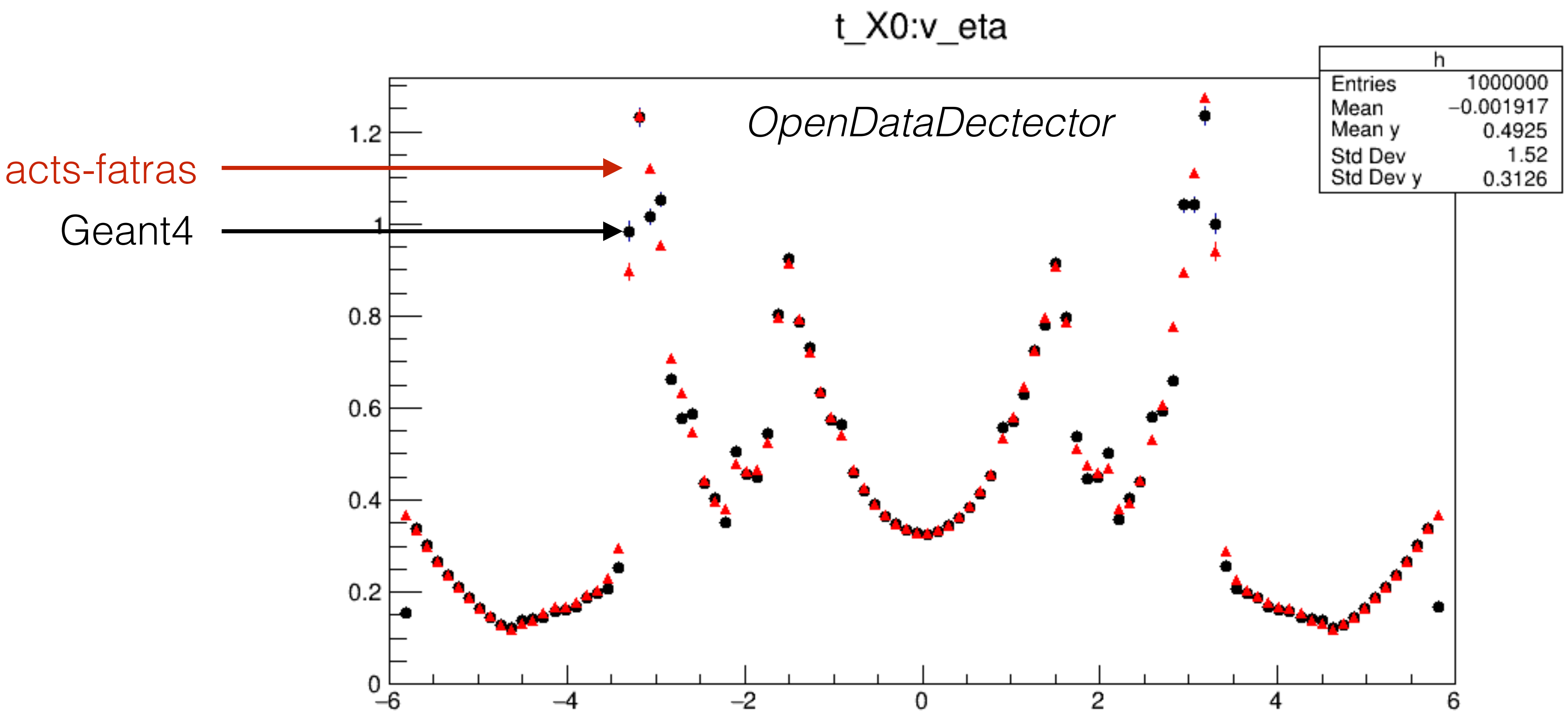


Material map to DD4HEP surfaces



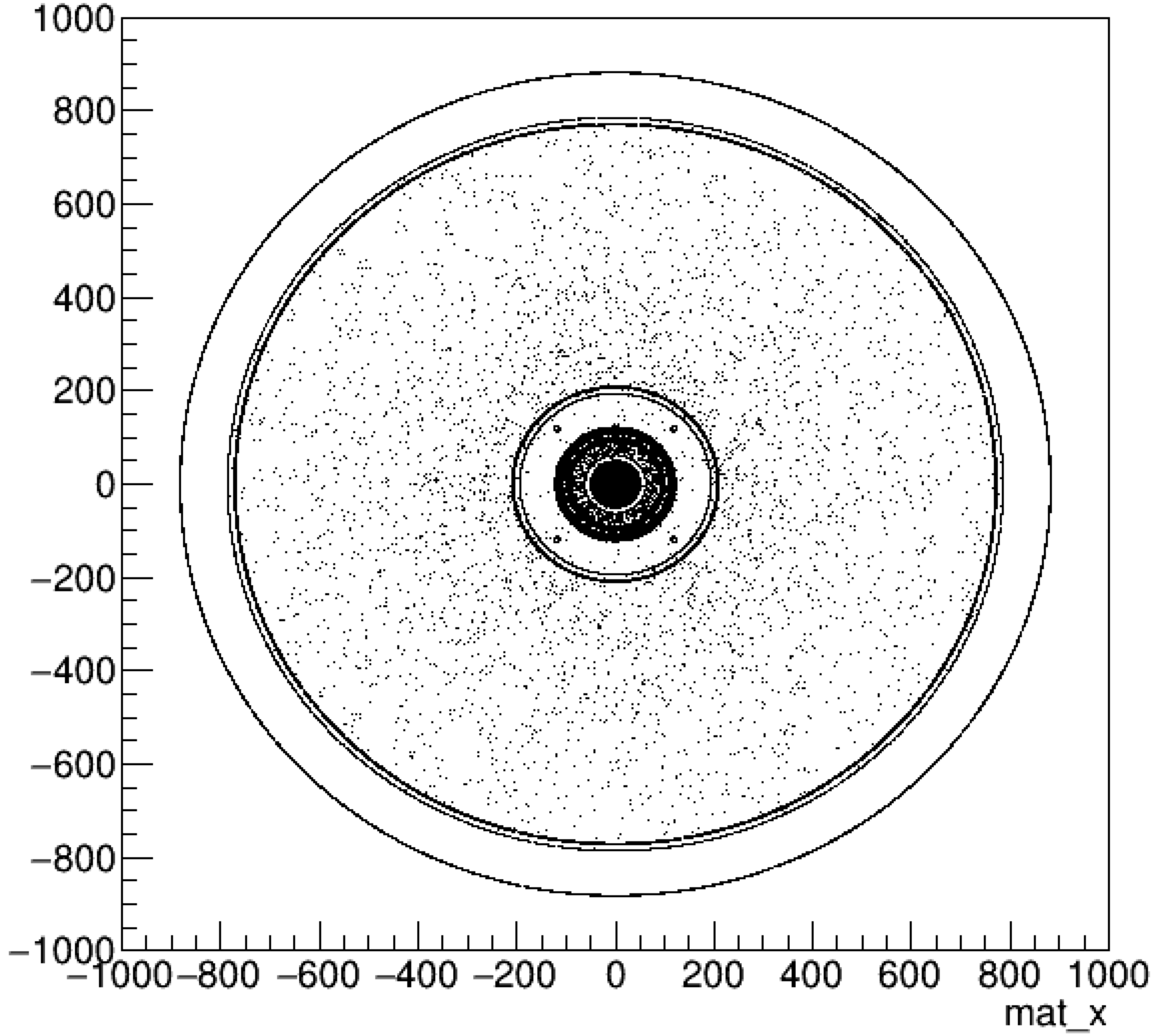
Material maps

Are stored in `root` or `json` format



Material Customize

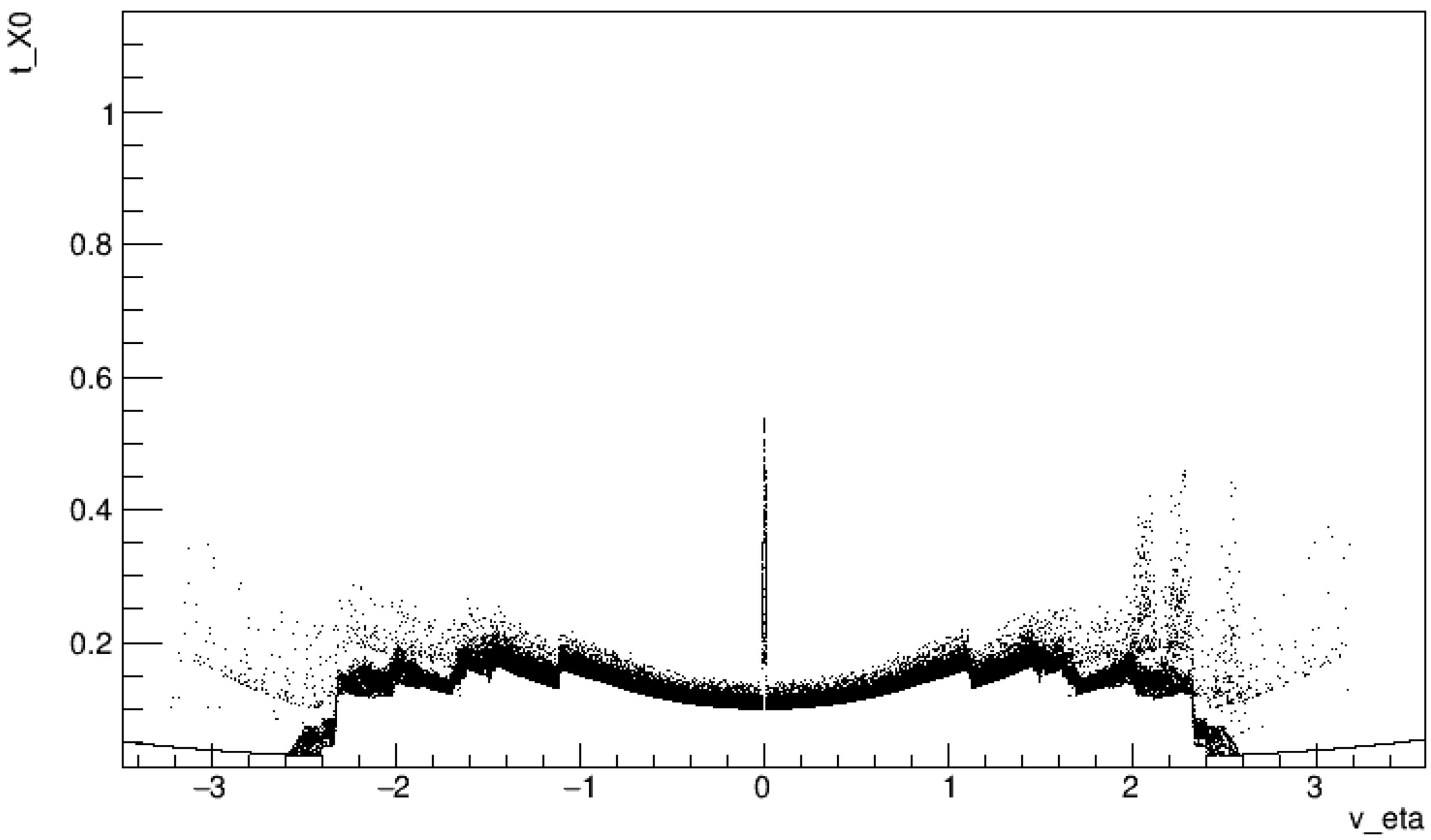
Running a dedicated Geant4 job records G4 material information



Material Customize

Running a dedicated Geant4 job records G4 material information

ROOT File (TGeo description) -> GDML



Event Data Model

Tracking

Event Data Model
(TrkEvent)

- ☑ Fixed size vector/matrix operations
- ☑ Top level detector agnostic geometry description
- ☑ Type safety
- ☑ Surface/EDM binding
- ☑ Measurement/Calibrated measurement
- ~~☐ Highly polymorphic structure with inheritance~~

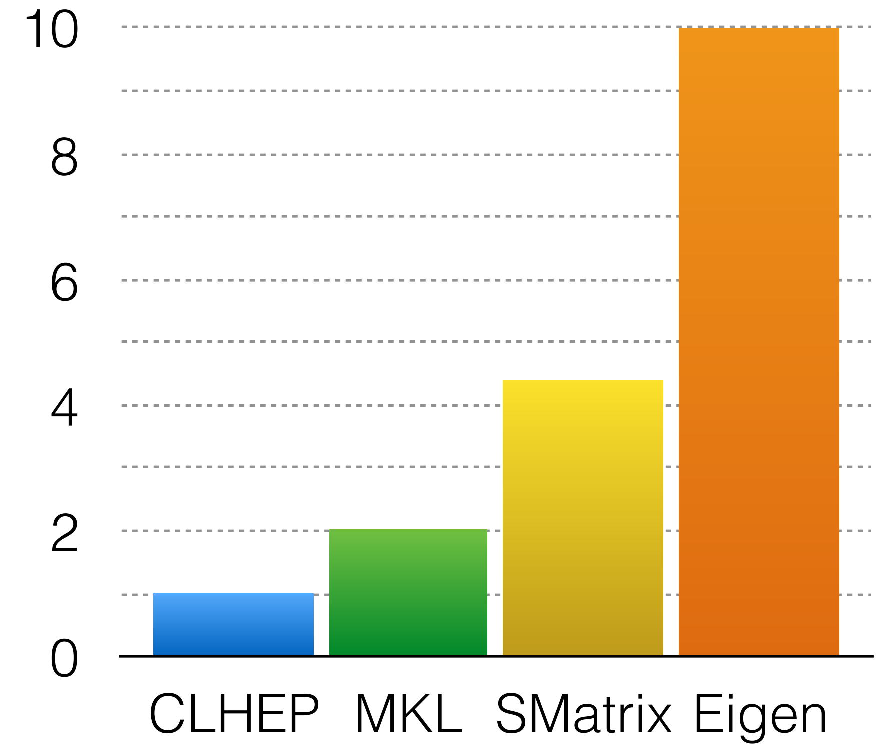
Acts

Review ATLAS Tracking Linear Algebra library

Initial ATLAS (Tracking) code was based on CLHEP

- change during LS1 to Eigen
  after extensive tests

Achieved speed-up w.r.t. CLHEP in 5x5 matrix multiplication testbed



Starting from LS1 EDM

- Eigen dependency fully extended to geometry model as well
- ATLAS GeoModel based on Eigen to be integrated easily

Projects using Eigen

Feel free to add yourself! If you don't have access to the wiki or if you are not sure about the relevance of your project, ask at the #Mailing list.

Extensions, numerical computation

- Google's TensorFlow is an Open Source Software Library for Machine Intelligence
- Google's Ceres solver is a portable C++ library that allows for modeling and solving large complicated nonlinear least squares problems.

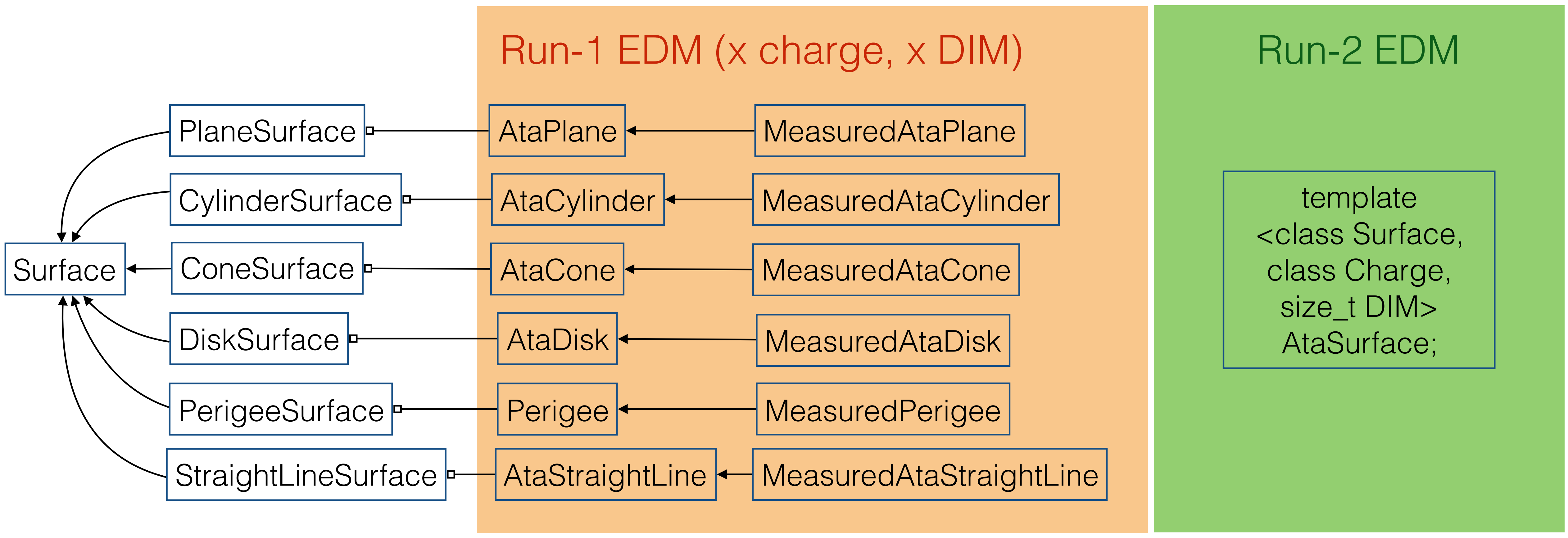
Review ATLAS Tracking Event Data Model (1)

TrackParameters

ATLAS Tracking EDM was heavily typed

- extremely complicated EDM inheritance structure
- massive code duplication (maintenance)
- problematic for persistency

starting point for Acts EDM



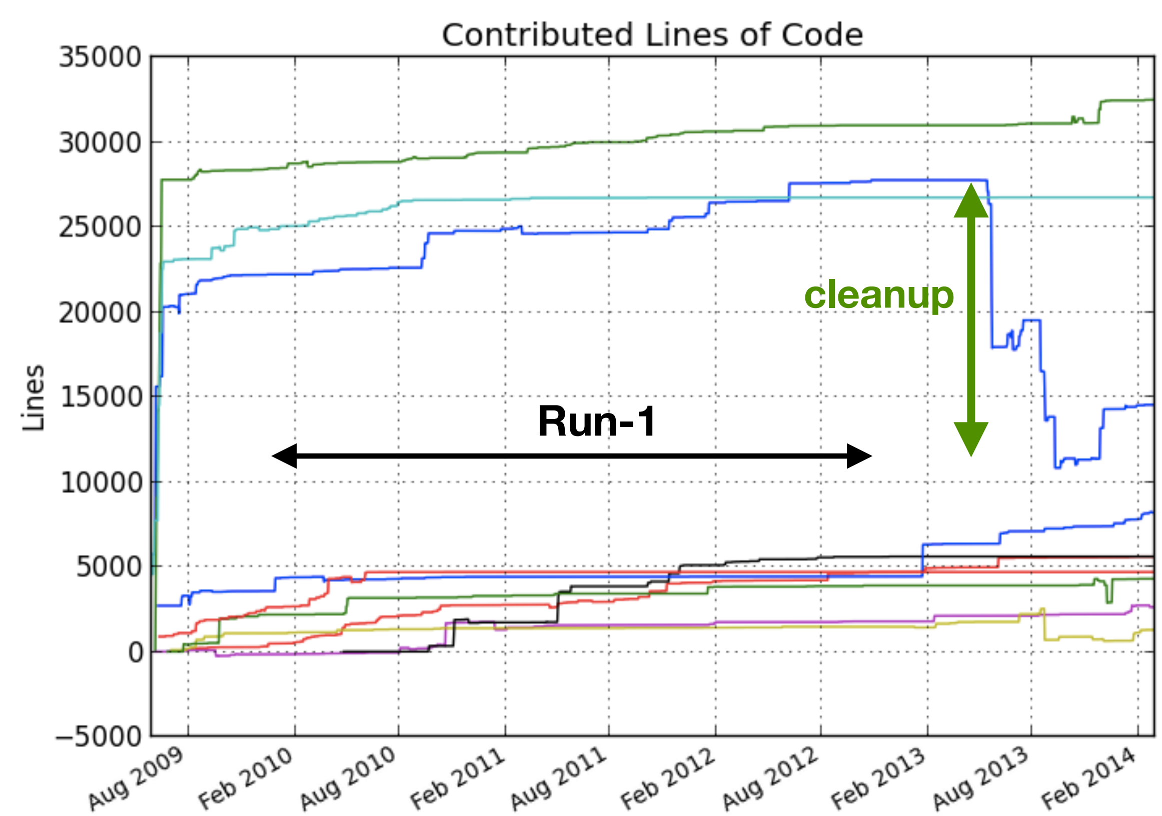
Review ATLAS Tracking Event Data Model (2)

TrackParameters

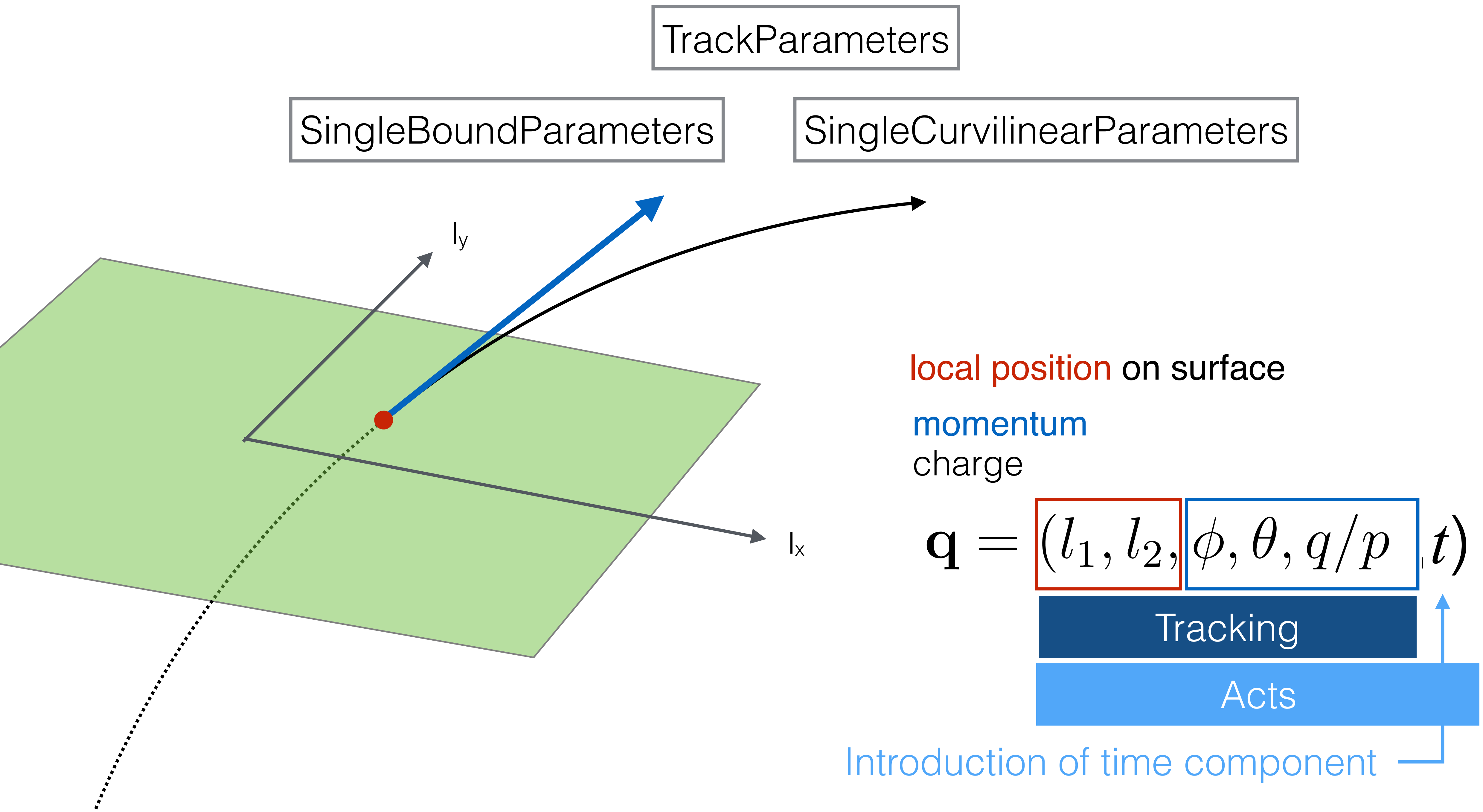
LS1 cleanup reduced number of lines massively

- keeping the same functionality & type safety

Package	C++	C/C++ Header	C++	C/C++ Header
TrkParameterBase	63	561	11	214
TrkParameters	1715	602	0	52
TrkNeutralParameters	1425	663	0	48
ExtendedTrkParameterBase	0	295	0	0
ExtendedTrkParameters	1412	514	0	0
ExtendedTrkNeutralParameters	1416	514	0	0
Total	**6031**	**3149**	**11**	**266**



Event Data Model Track Parameters



If you want the 5-parameterisation, you just write

```
auto pars = perigee.parameters().block<5,1>();
```

Event Data Model Track Parameters

TrackParameters

SingleBoundParameters

SingleCurvilinearParameters

Extension for Multi Component representation

- avoid copying of Extrapolator (as done in ATLAS) and Fitter infrastructure for multi-variant fitters (MultiTrackFitter, GSF)

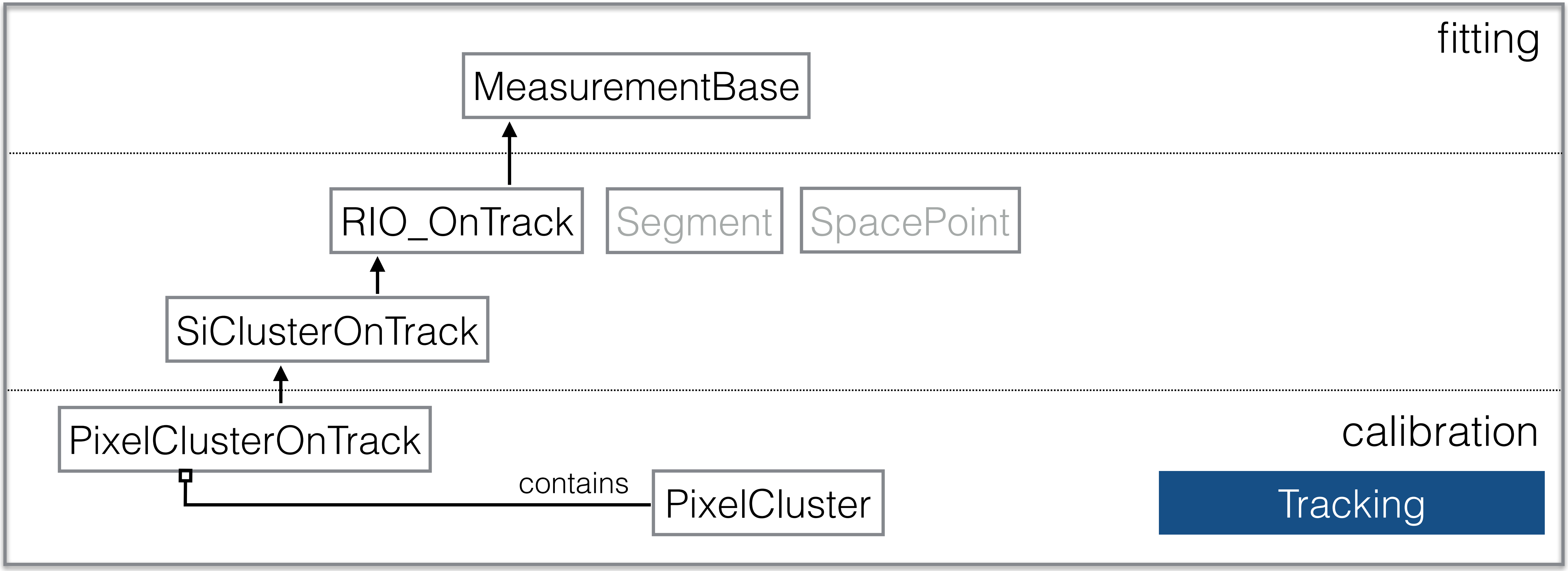
  act as single track parameters in navigation, but will be propagated as multiple components in between

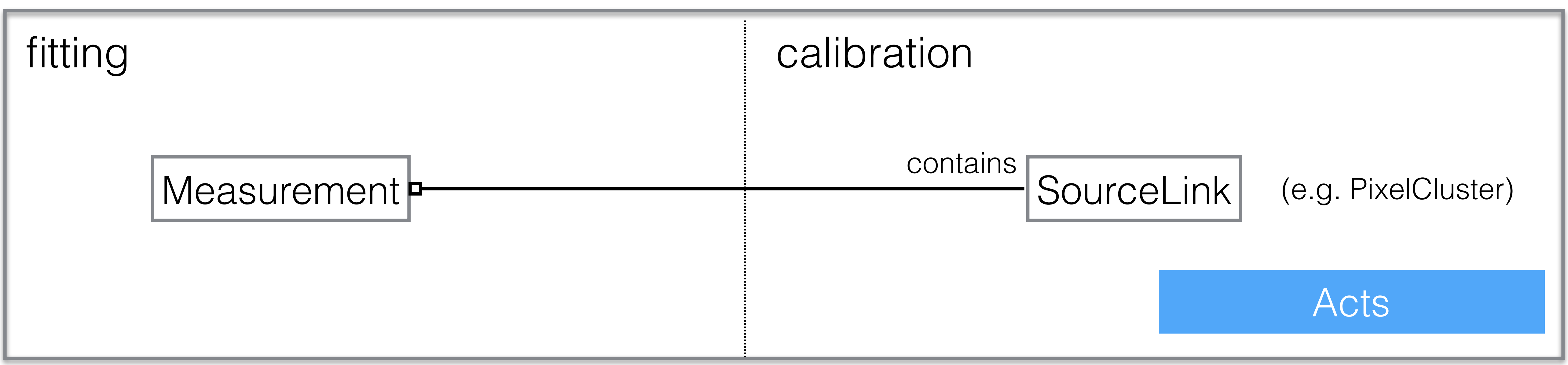
MultivariantTrackParameters

MultivariantBoundParameters

MultivariantCurvilinearParameters

Event Data Model Measurements & calibration





Event Data Model Customise

`source_link_t`

your data class representing a measurement

`params`

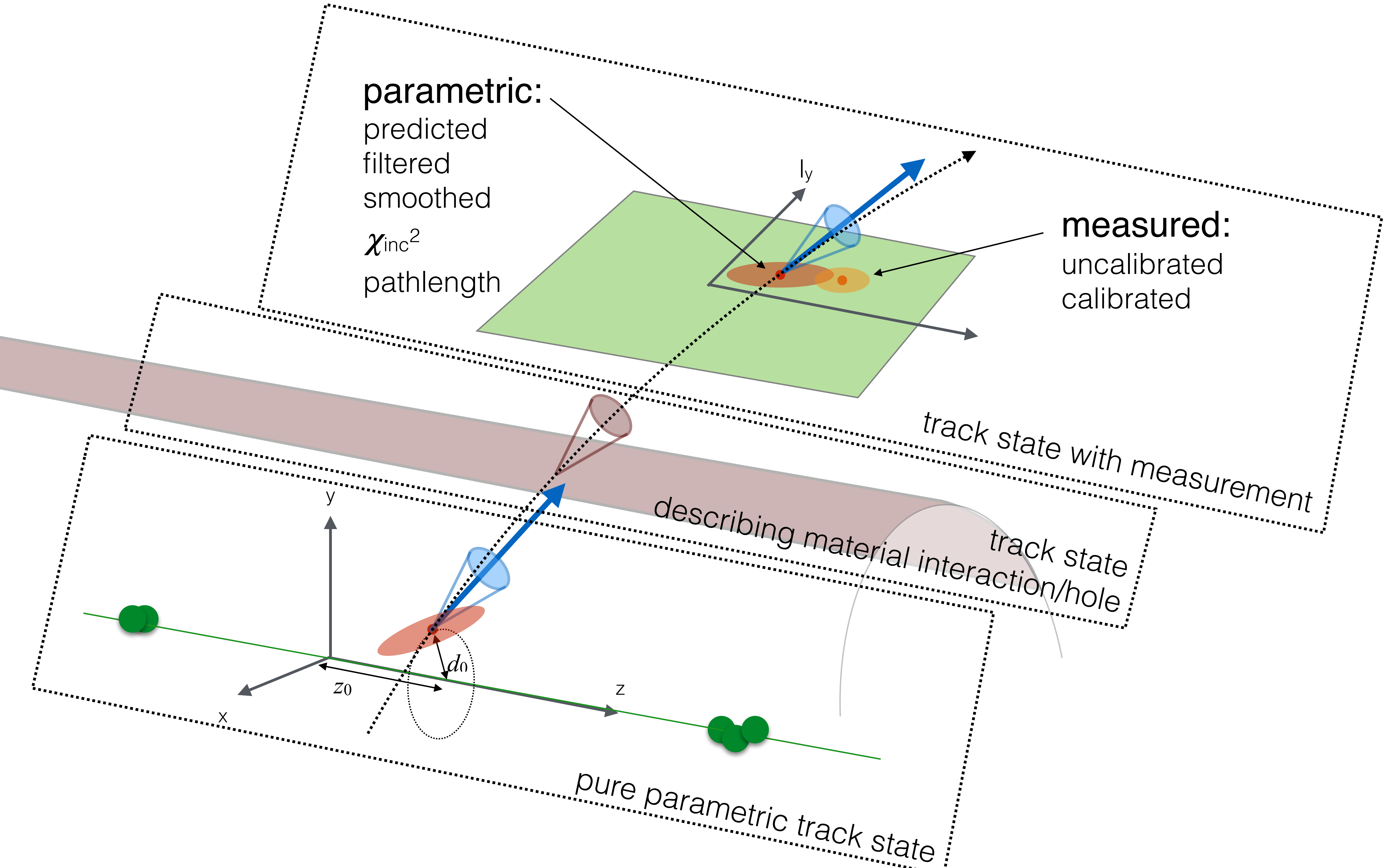
parameter pack describing
1-dim to N-dim measurement

```
///
/// @tparam Identifier identification object for this measurement
/// @tparam params     parameter pack containing the measured parameters
template <typename source_link_t, ParID_t... params>
class Measurement {
  // check type conditions
  static_assert(SourceLinkConcept<source_link_t>,
                "Source link does not fulfill SourceLinkConcept");
```

Interface needed by acts tools:

```
/// @brief access vector with measured parameter values
///
/// @return column vector whose size is equal to the dimensionality of this
/// Measurement. The values are
///         given for the measured parameters in the order defined by the
///         class
/// template argument @c params.
ParVector_t parameters() const { return m_oParameters.getParameters(); }

/// @brief access covariance matrix of the measured parameter values
///
/// @return covariance matrix of the measurement
CovMatrix_t covariance() const { return *m_oParameters.getCovariance(); }
```

Track and TrackState description



Extrapolation

Tracking

Extrapolation
(TrkExtrapolation)

- ☑ Support for different stepping methods
- ☑ Support for Layer / Dense volume / floating object navigation
- ☑ Support for Surface based & Volume based material
- ☑ STEP propagator extension of RK4
- ☐ ~~Distinction between Propagator and Extrapolator~~
- ☐ ~~(A lot of) dynamic memory allocation~~
- ☐ ~~Virtual calls to magnetic field & other tools~~
- ☐ ~~Navigation caching (not possible fro MT)~~

Acts

Magnetic field field caching

Magnetic field caching found to reduce CPU time in

- Simulation (up to 20%)
- Reconstruction (around few %)

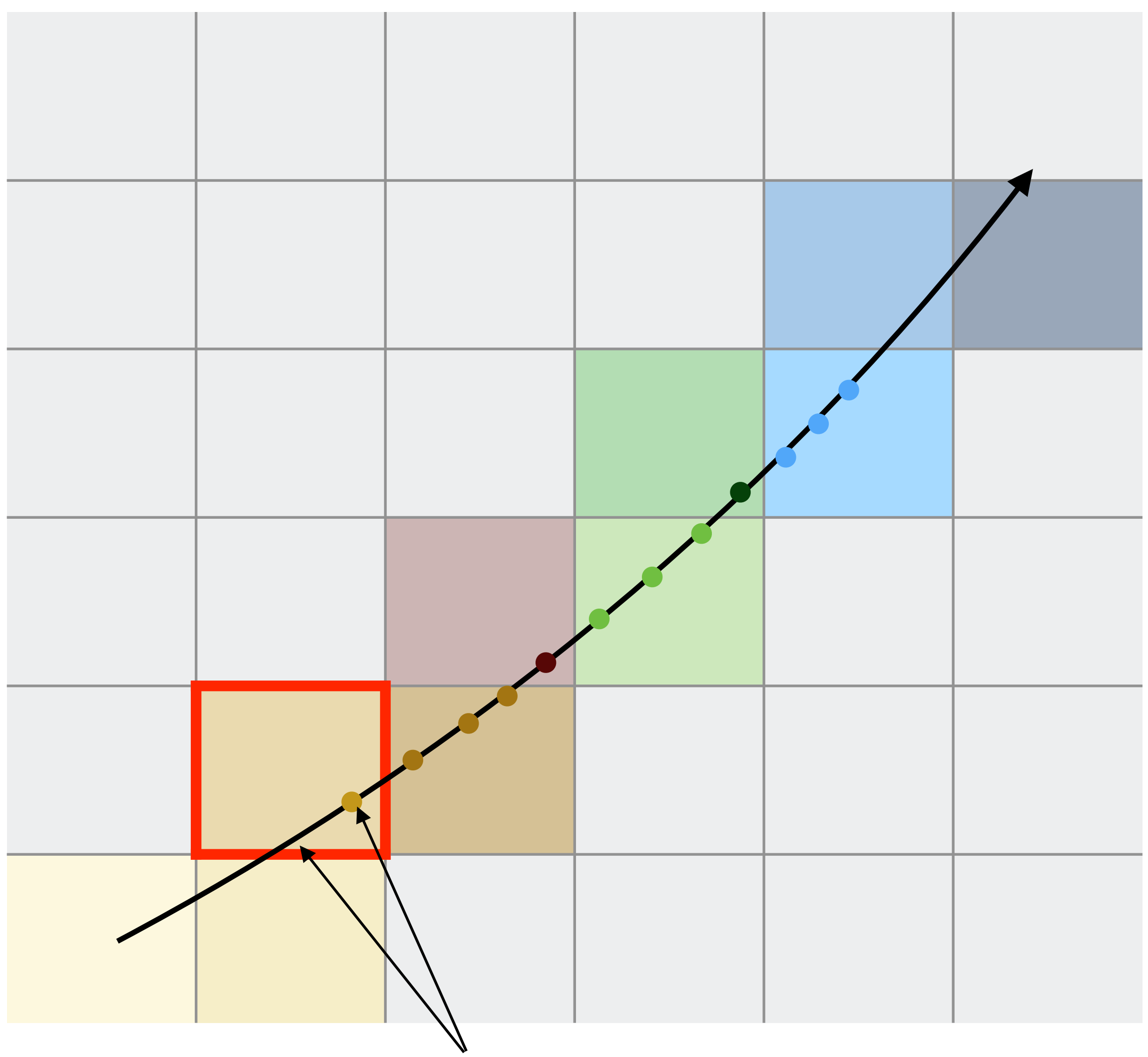
Tracking locks the field cell in the magnetic field service

- not ideal for concurrent usage

(field cell is not exposed to propagator, needs to be secured within FieldSvc)

Acts field service provides a field cell to be cached by the caller (see propagation)

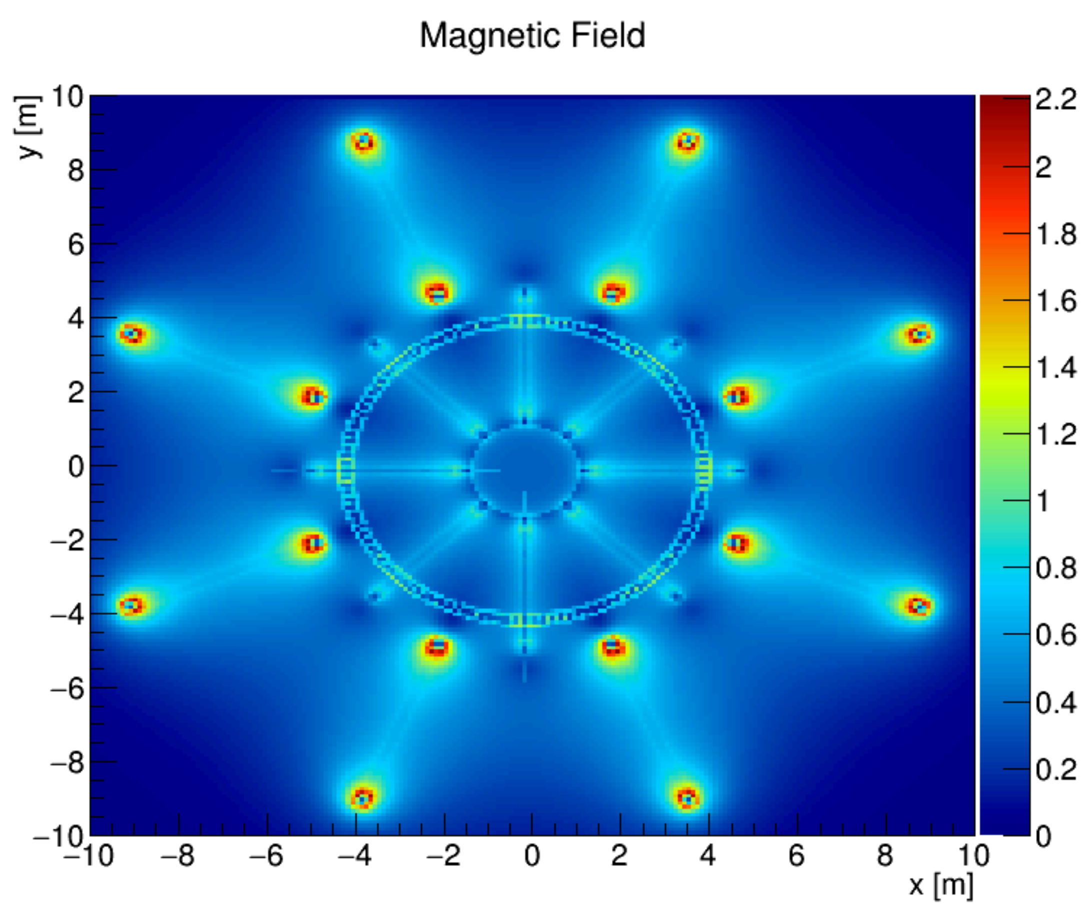
- C++ concept for field cell



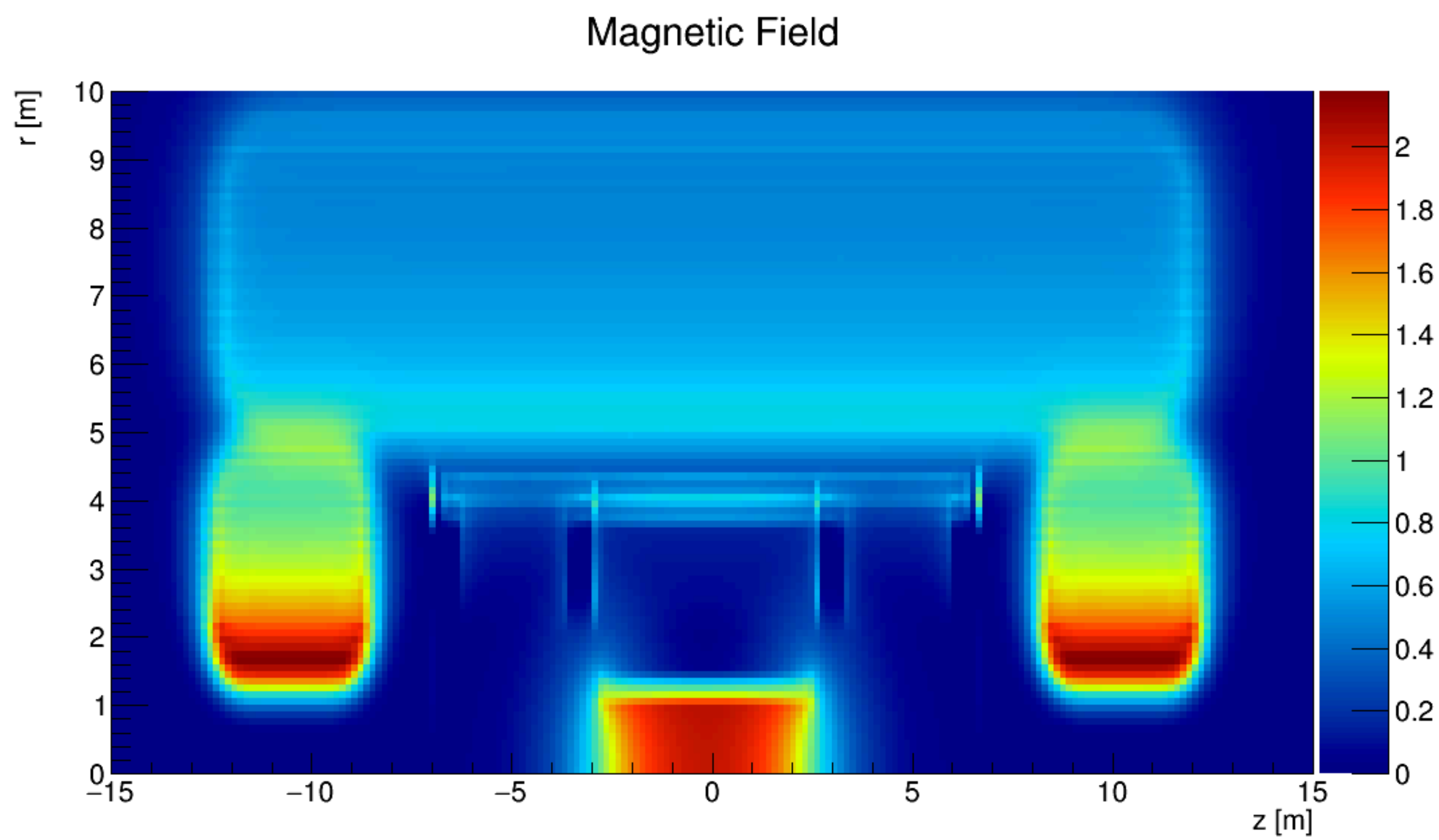
Magnetic field

Tests using different magnetic field inputs within Acts

- ATLAS map (currently converted from ATLAS root file), direct use of ATLAS MagneticFieldSvc possible (template parameter)
- FCC-hh field map



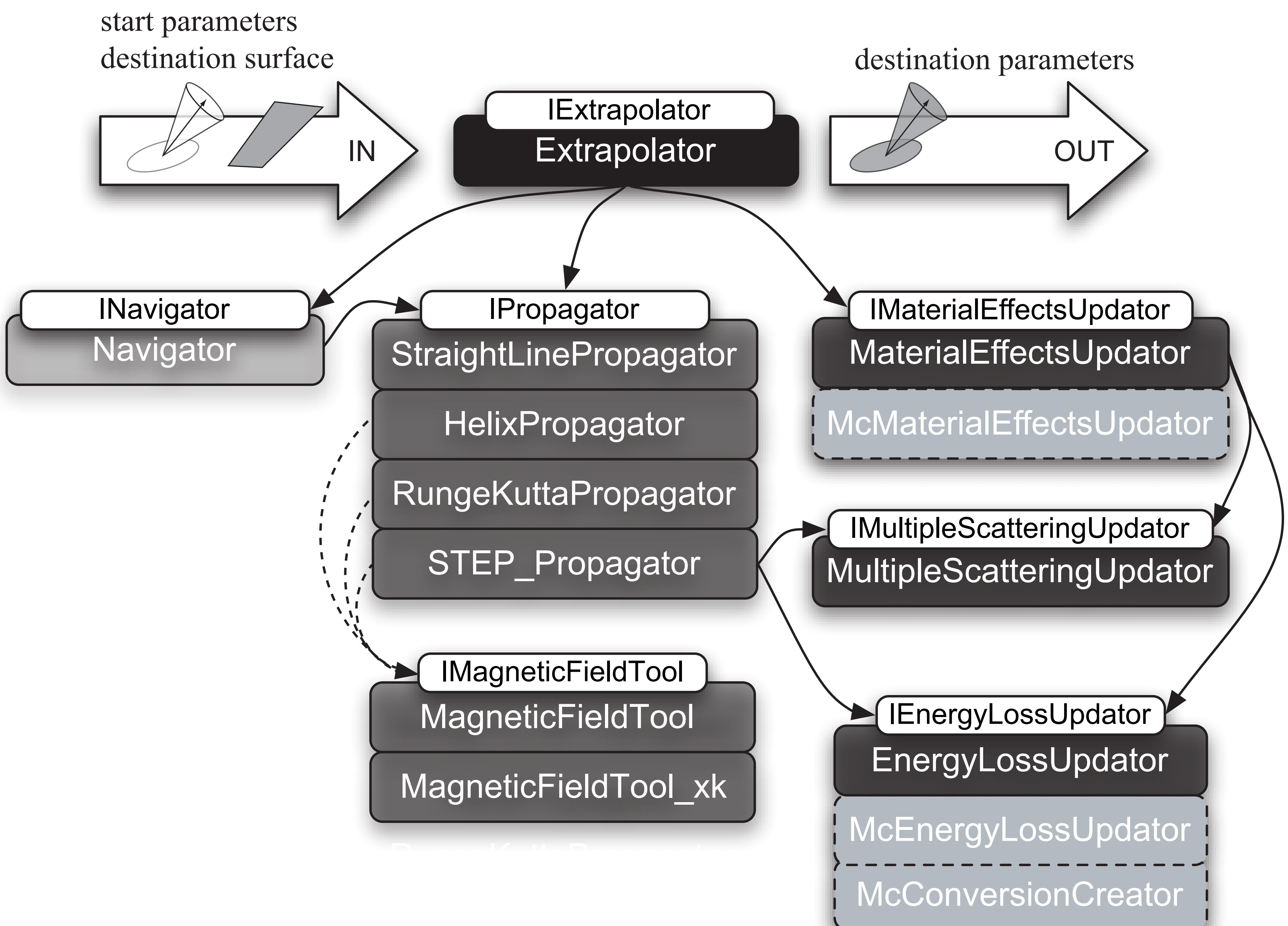
ATLAS magnetic field map in ACTS



Propagation | Extrapolation

ATLAS Tracking SW :

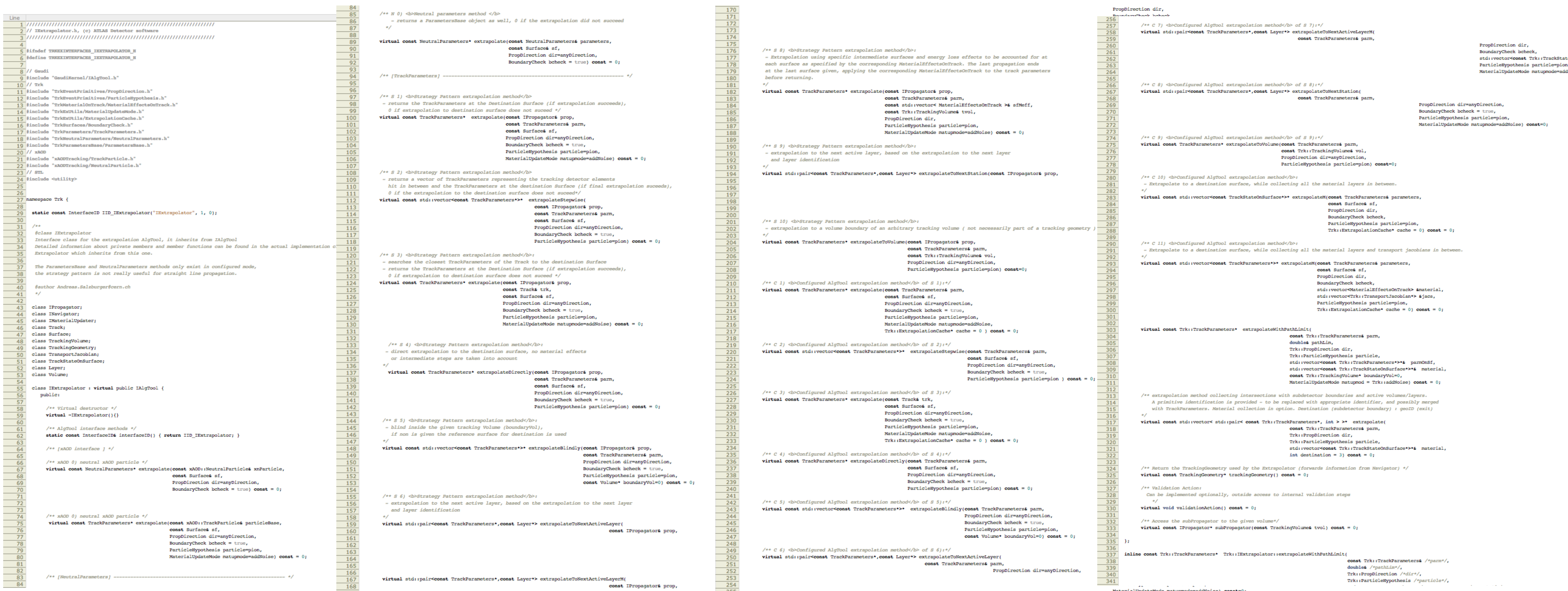
- distinction between **propagation** (transport)
  and **extrapolation** (transport, navigation & material effects integration)



Propagation | Extrapolation ATLAS

Extrapolator Tool interface was expanded to do more & more

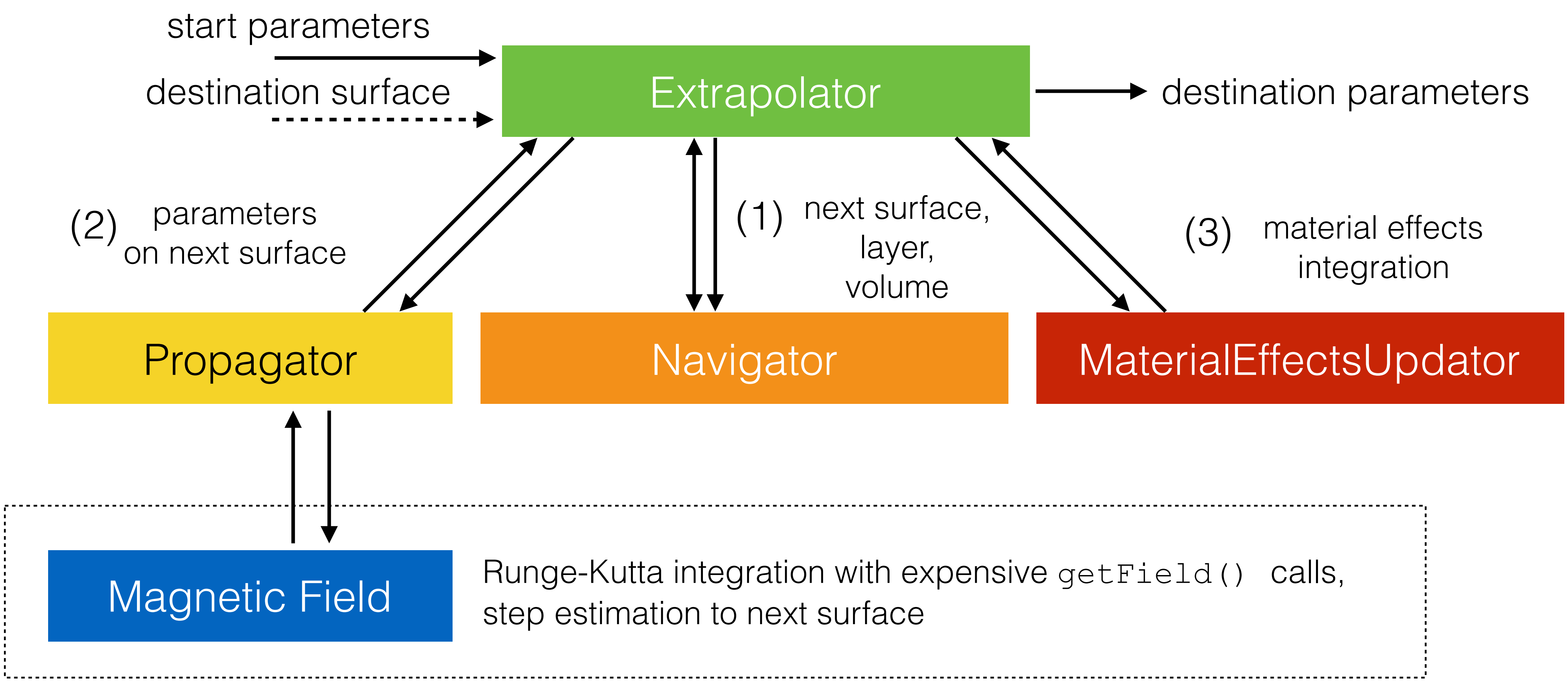
- hole search, jacobian collection, material collection
- fast simulation, track-to-cal, track-through-calo, etc. …



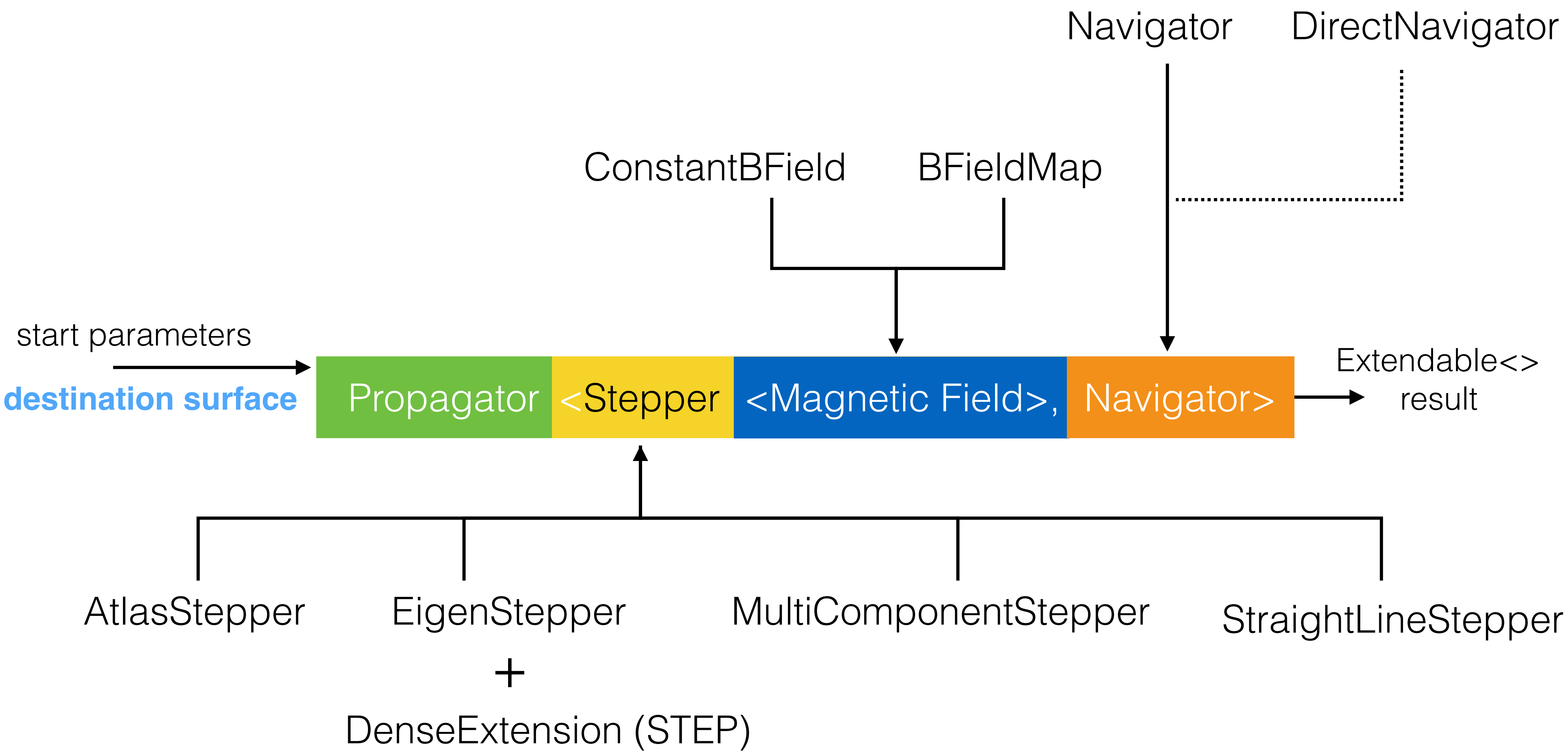
~ 360 lines interface in IExtrapolator.h

~ 4700 lines of code in Extrapolator.cxx

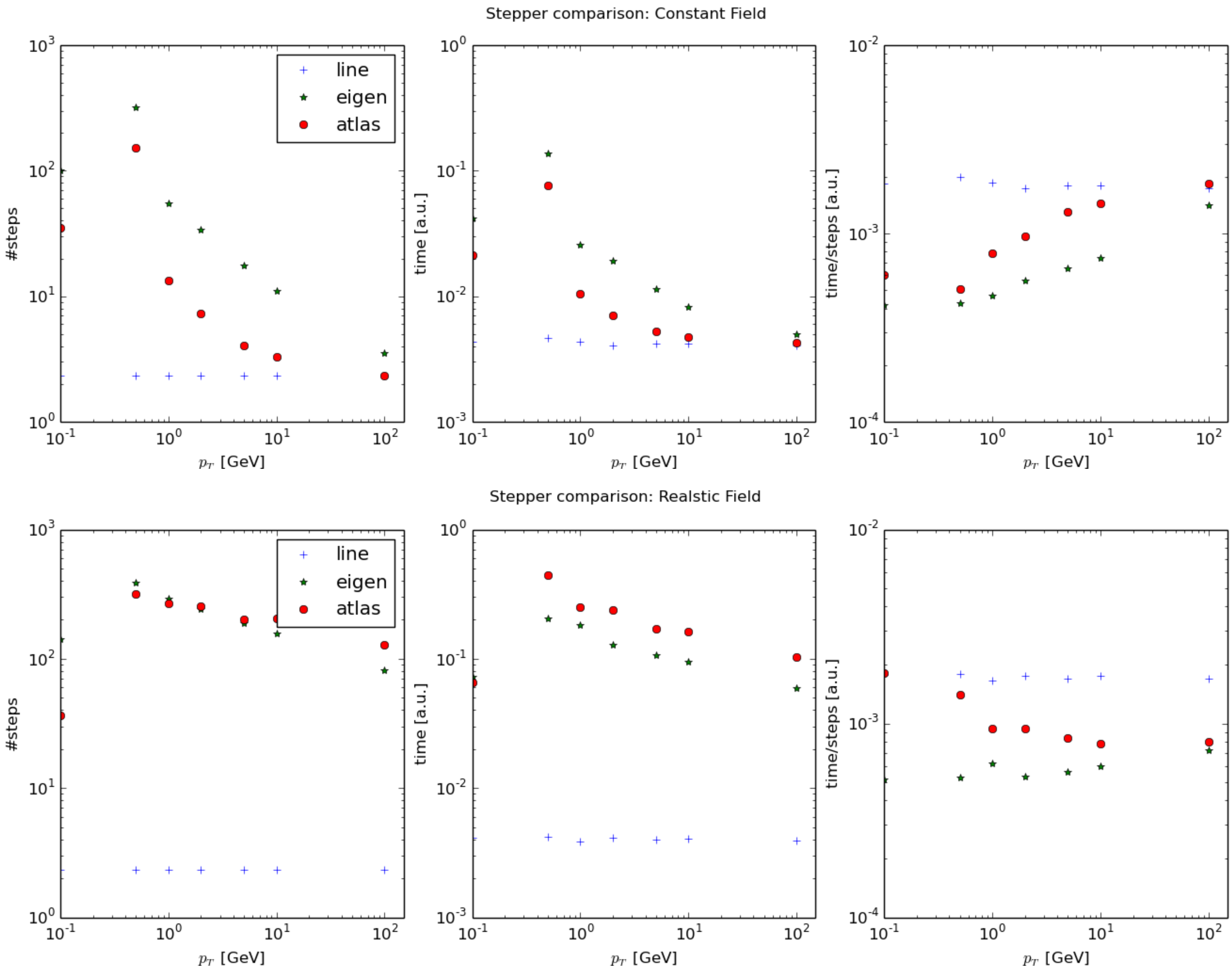
Extrapolator to Propagator



Propagator in Acts



Stepper Timing | Examples

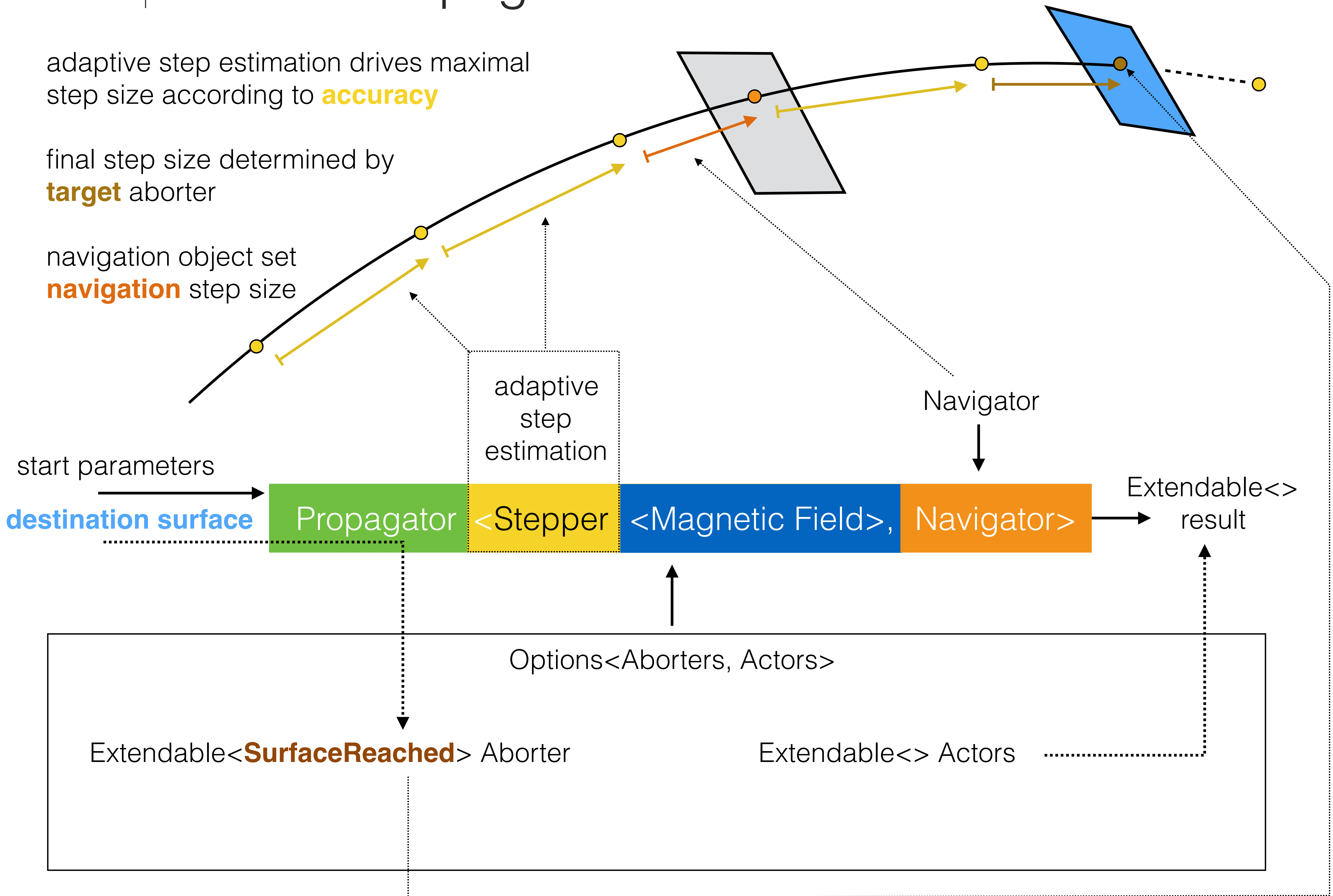


Extrapolator to Propagator

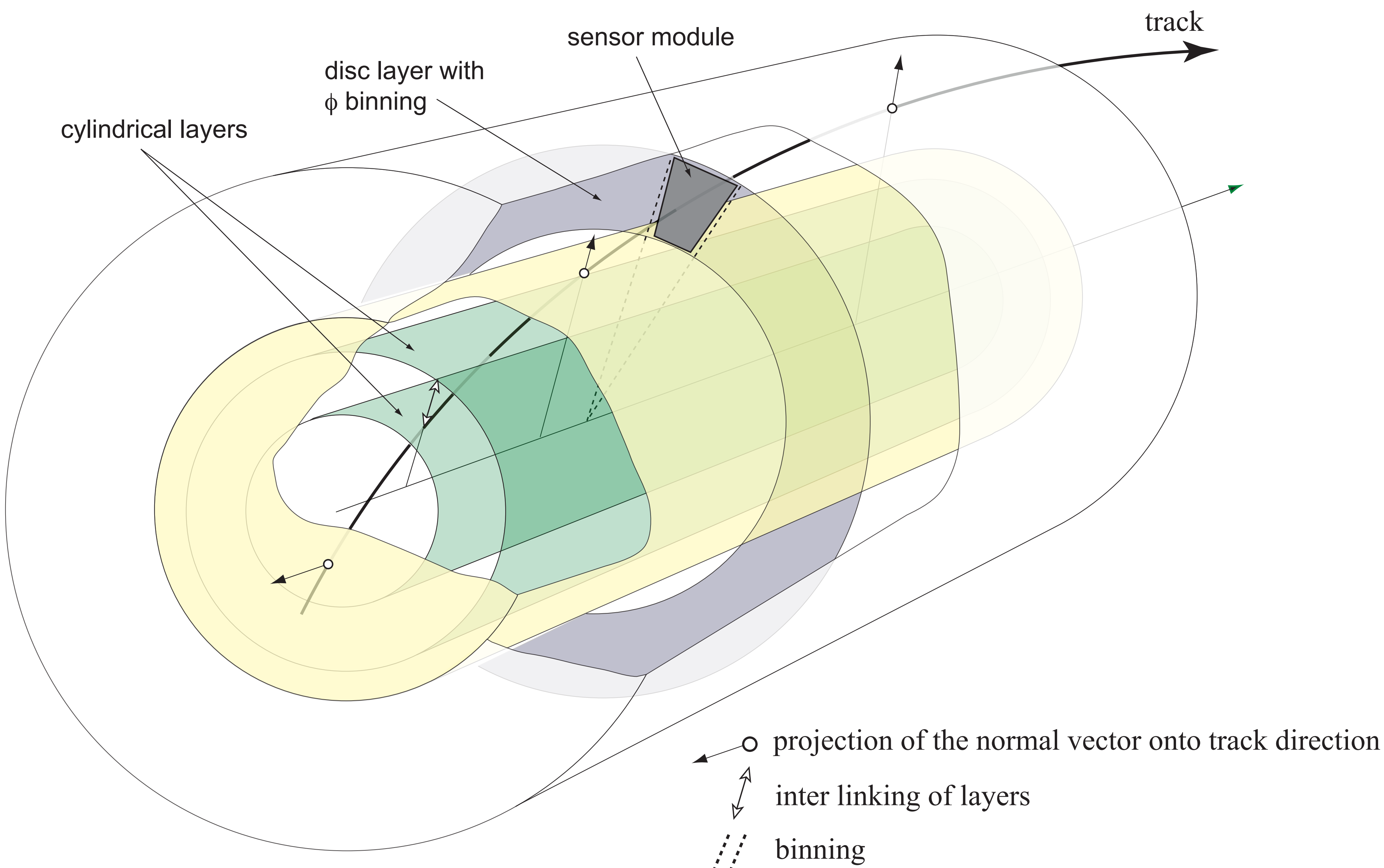
adaptive step estimation drives maximal step size according to **accuracy**

final step size determined by **target** aborter

navigation object set **navigation** step size

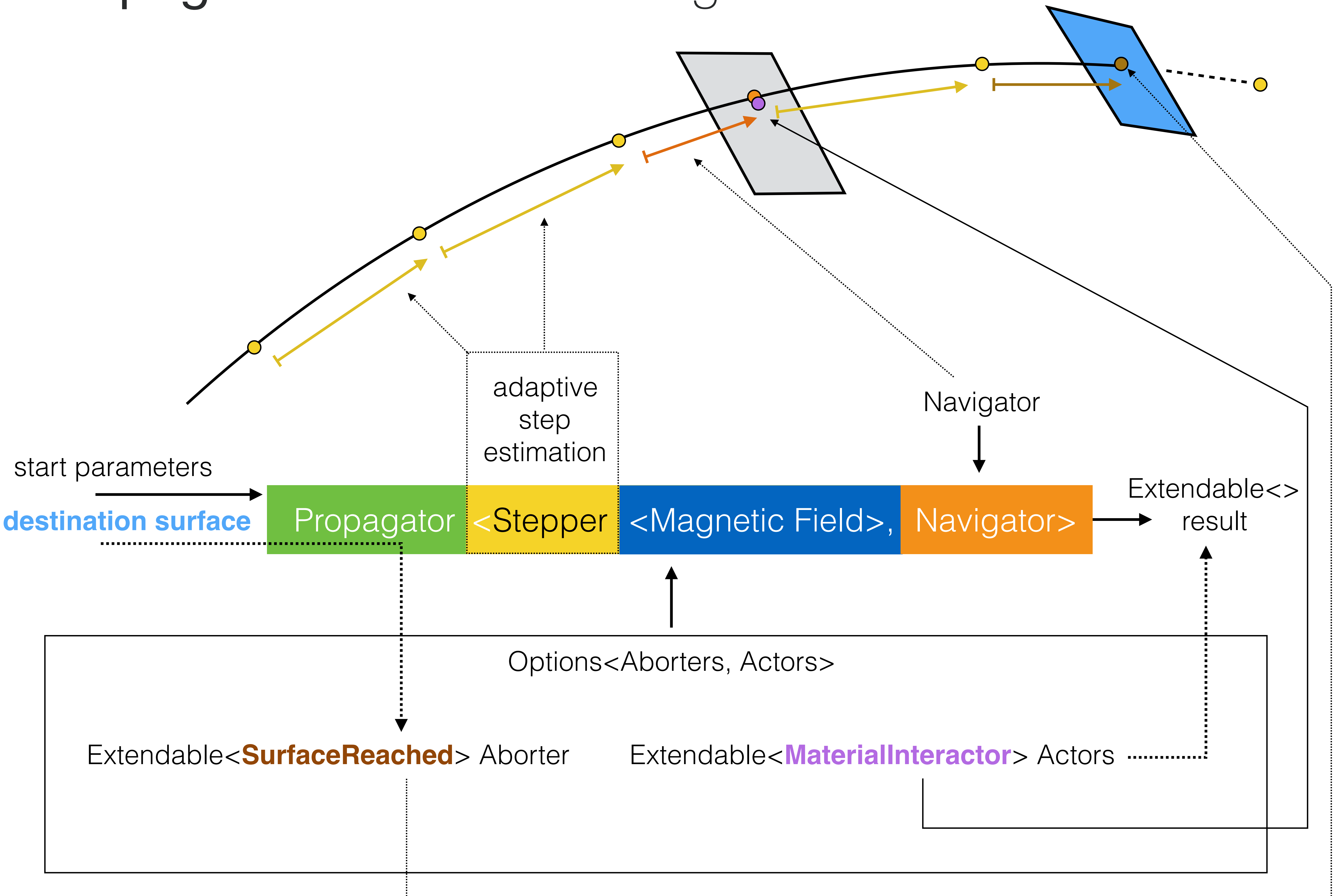


Propagation | Extrapolation



ATLAS: at every step new dynamic memory allocation (TrackParameters)

Propagator with material integration



Propagation Time component

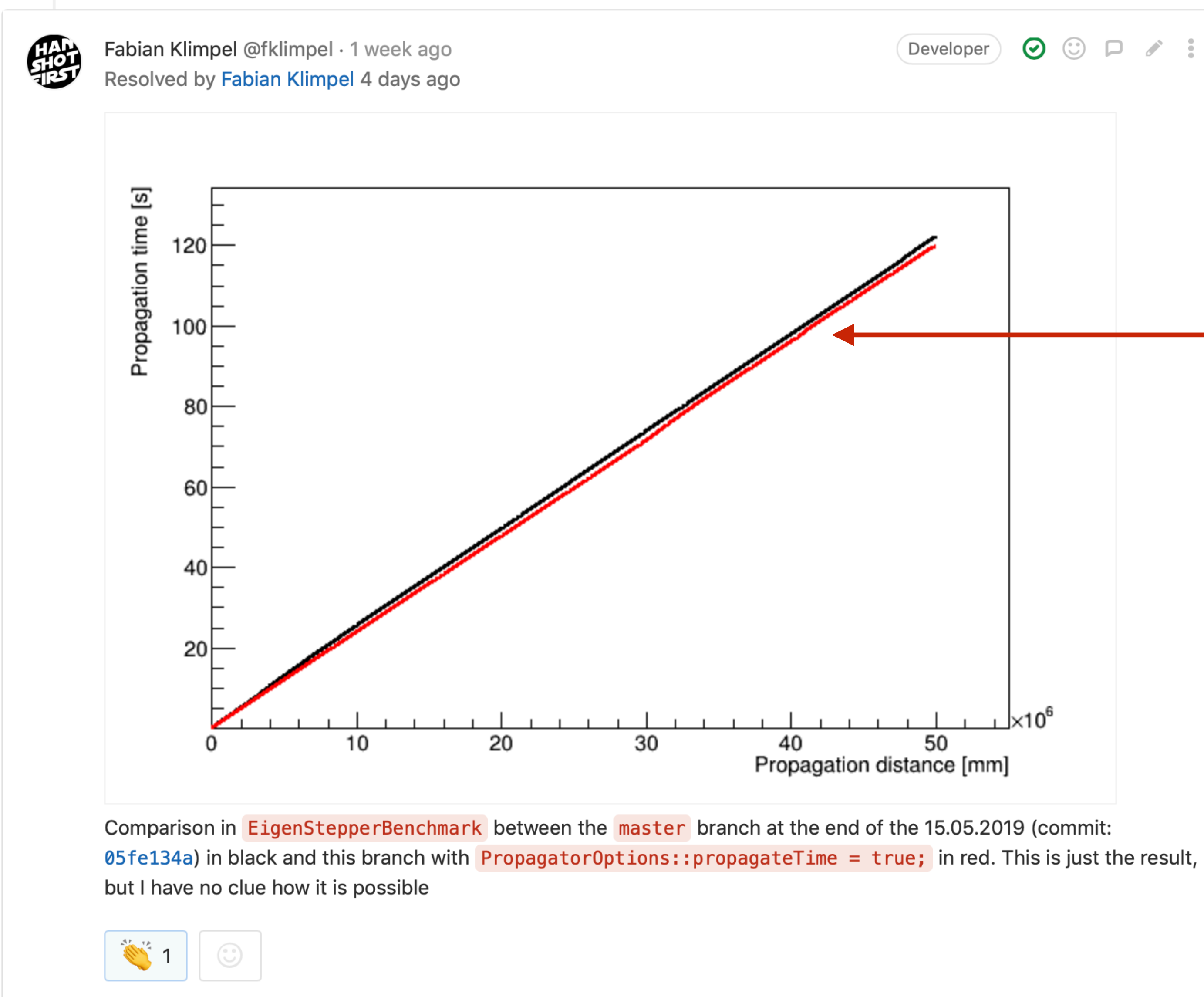
Internal representation expanded from **7x7** description to **8x8**

- full time covariance transport developed (and numerically tested)
- positive impact on execution speed

$$\mathbf{q} = (l_1, l_2, \phi, \theta, q/p, t)$$

Tracking

Acts



- could be better vectorisation, not confirmed yet

Propagation Interface

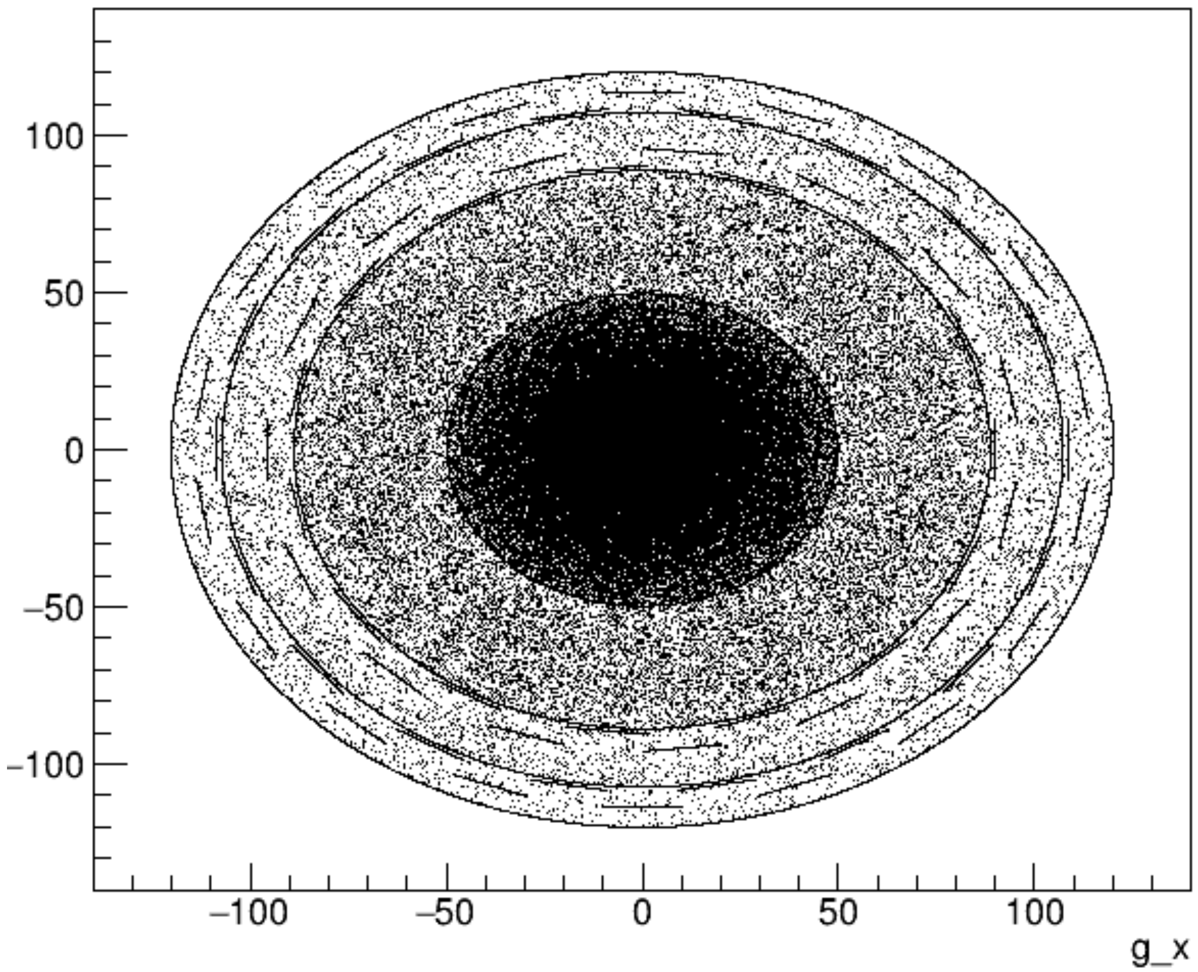
```
/// @brief Propagate track parameters
///
/// This function performs the propagation of the track parameters using the
/// internal stepper implementation, until at least one abort condition is
/// fulfilled or the maximum number of steps/path length provided in the
/// propagation options is reached.
///
/// @tparam parameters_t Type of initial track parameters to propagate
/// @tparam action_list_t Type list of actions, type ActionList<>
/// @tparam aborter_list_t Type list of abort conditions, type AbortList<>
/// @tparam propagator_options_t Type of the propagator options
///
/// @param [in] start initial track parameters to propagate
/// @param [in] options Propagation options, type Options<,>
///
/// @return Propagation result containing the propagation status, final
///         track parameters, and output of actions (if they produce any)
///
template <typename parameters_t, typename action_list_t,
          typename aborter_list_t,
          template <typename, typename> class propagator_options_t,
          typename path_aborter_t = detail::PathLimitReached>
Result<action_list_t_result_t<
    typename stepper_t::template return_parameter_type<parameters_t>,
    action_list_t>>
propagate(
    const parameters_t& start,
    const propagator_options_t<action_list_t, aborter_list_t>& options) const;
```

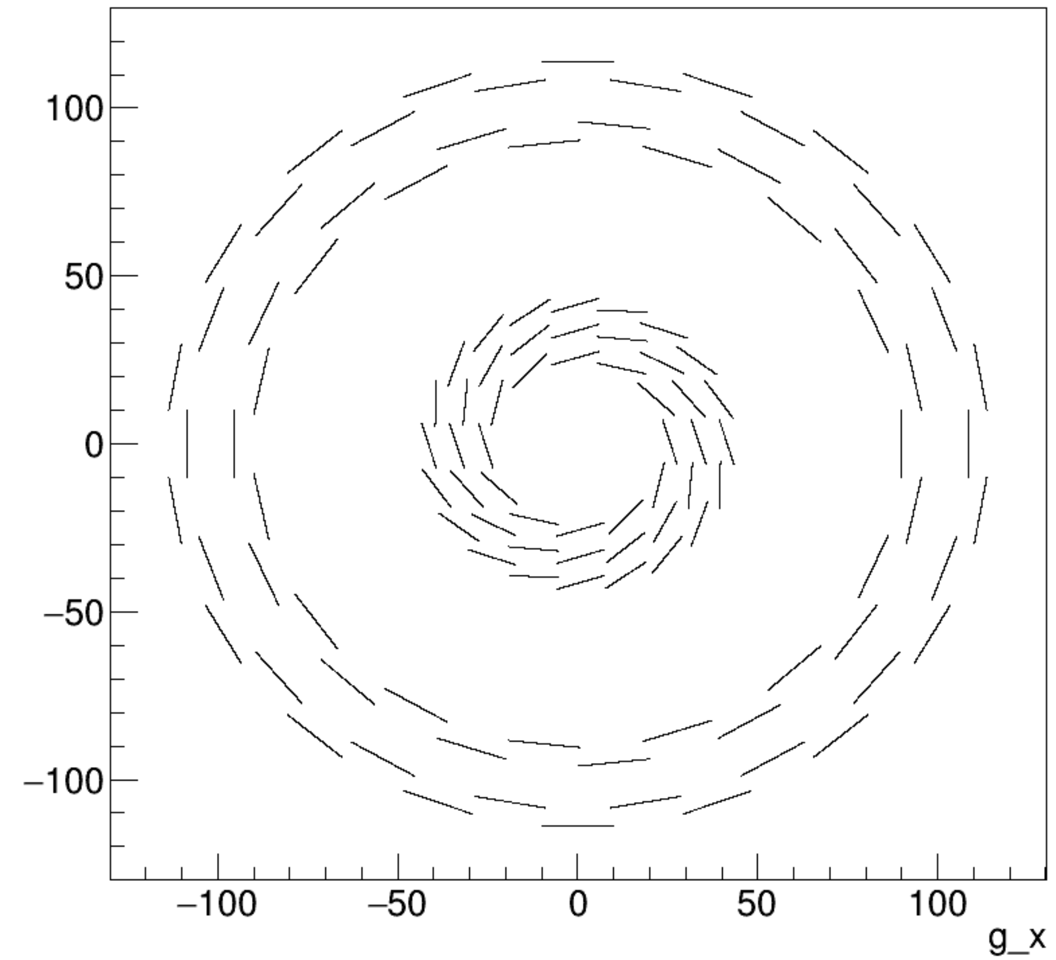
… defines result

Input

Options

Propagation Customize





Propagation Timing

```
16:36:21    Sequencer    INFO    Added algorithm 'PropagationAlgorithm'
16:36:21    Sequencer    INFO    Added writer 'RootPropagationStepsWriter'
16:36:21    Sequencer    INFO    Processing events [0, 100)
16:36:21    Sequencer    INFO    Starting event loop with 1 threads
16:36:21    Sequencer    INFO      0 services
16:36:21    Sequencer    INFO      0 context decorators
16:36:21    Sequencer    INFO      0 readers
16:36:21    Sequencer    INFO      1 algorithms
16:36:21    Sequencer    INFO      1 writers
16:36:21    Sequencer    INFO    finished event 0
[ ... ]
16:36:24    Sequencer    INFO    finished event 99
16:36:25    Sequencer    INFO    Processed 100 events in 3.953269 s (wall clock)
16:36:25    Sequencer    INFO    Average time per event: 39.055690 ms/event
```

1000 test propagations/event
full predictive navigation
(material integration, hole search)
propagation through 2 T field

Track Fitting

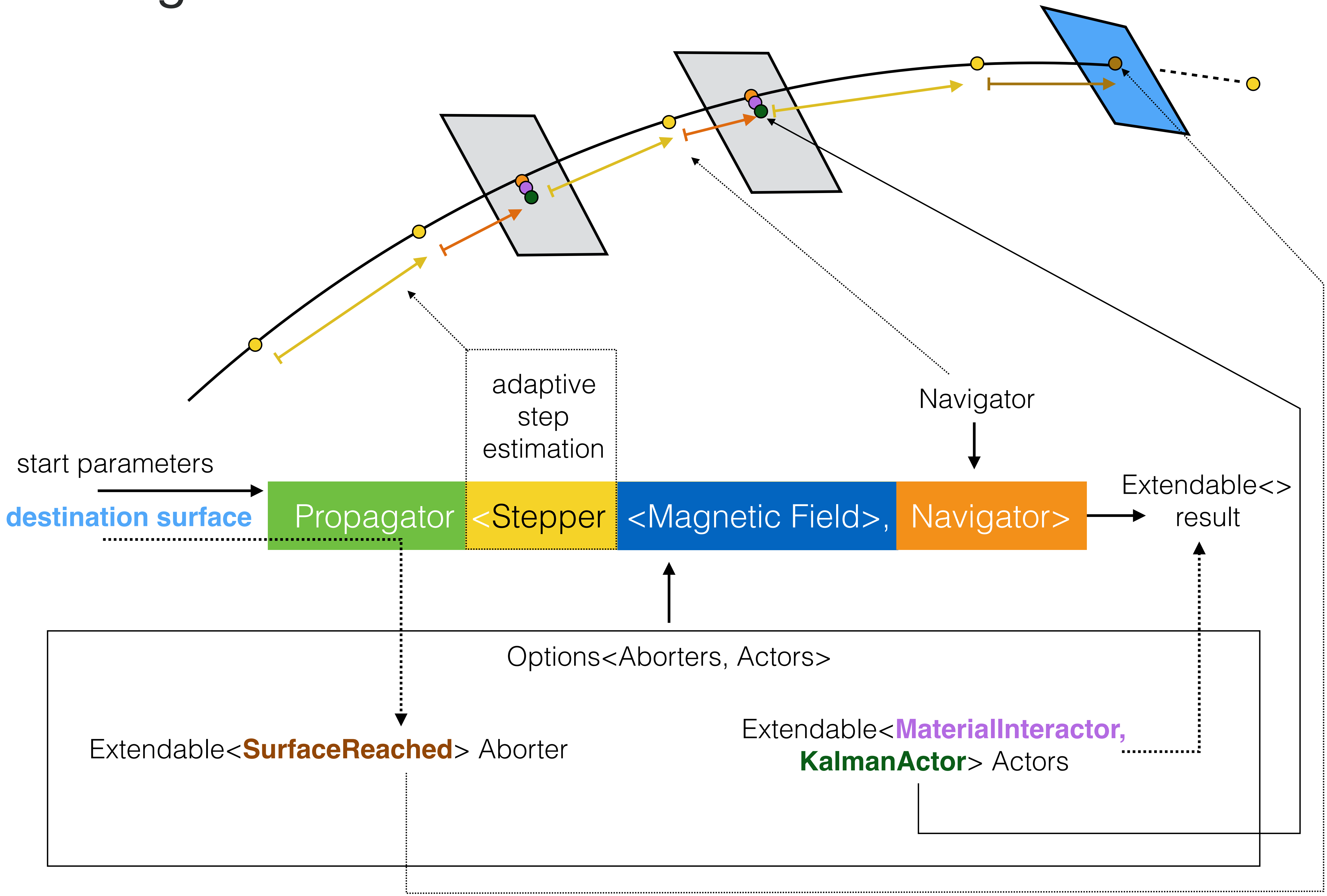
Tracking

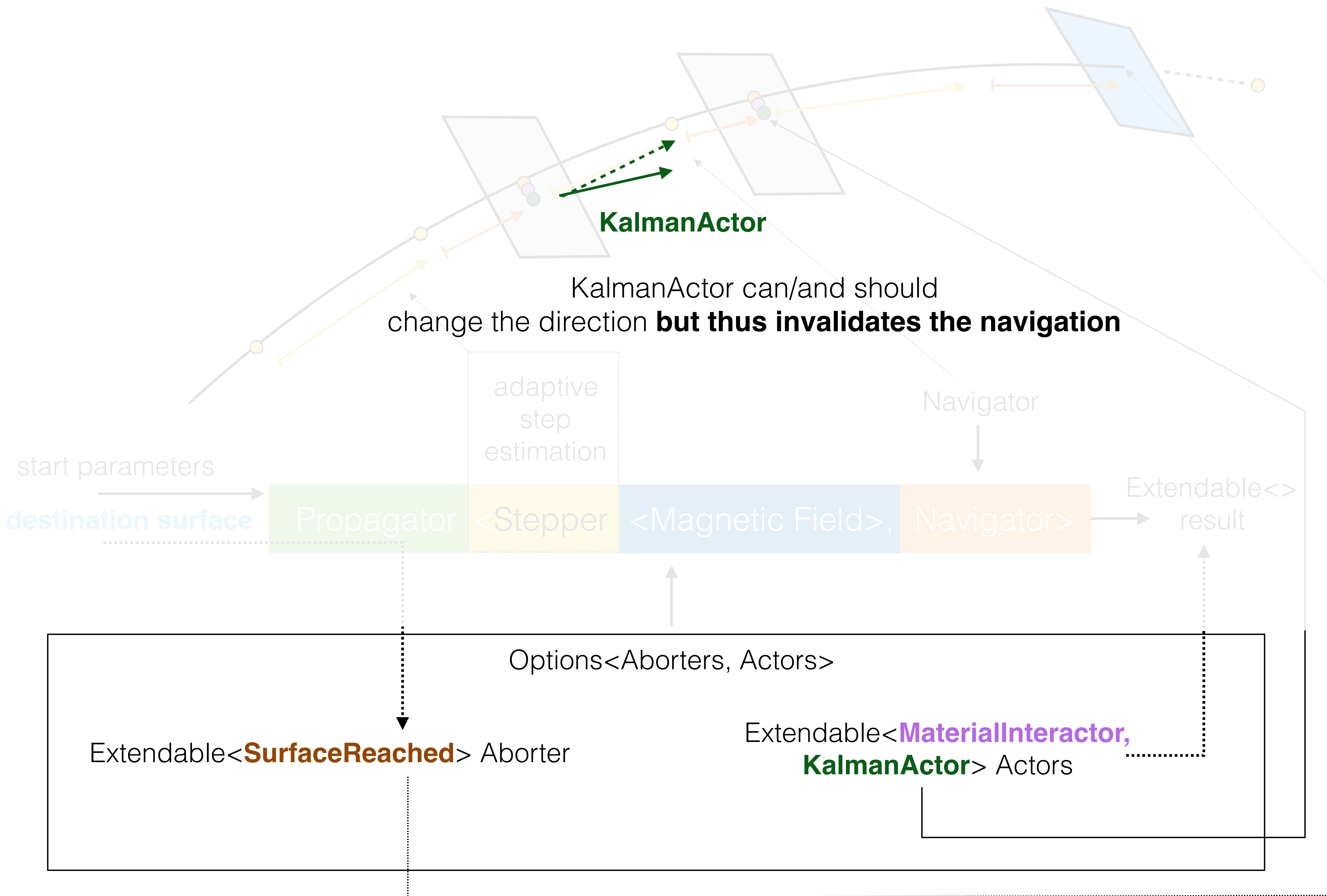
Fitting
(TrkFitter)

- ☑ Top level detector agnostic geometry description
- ☑ Calibration structure (**`PrepRawData`** -> **`RIO_OnTrack`**)
- ~~☐ Measurement sorting~~
- ~~☐ Many different interfaces~~

Acts

Fitting modules Kalman Filter

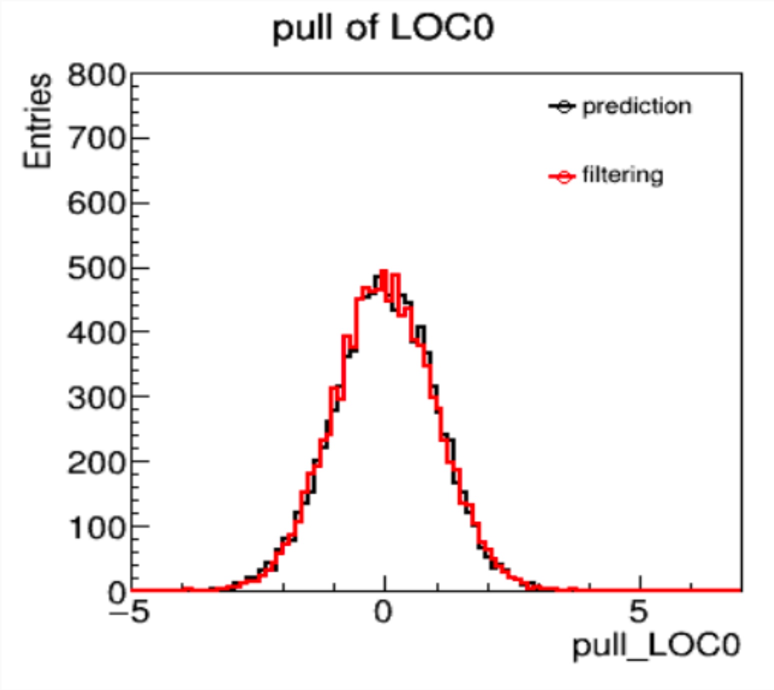


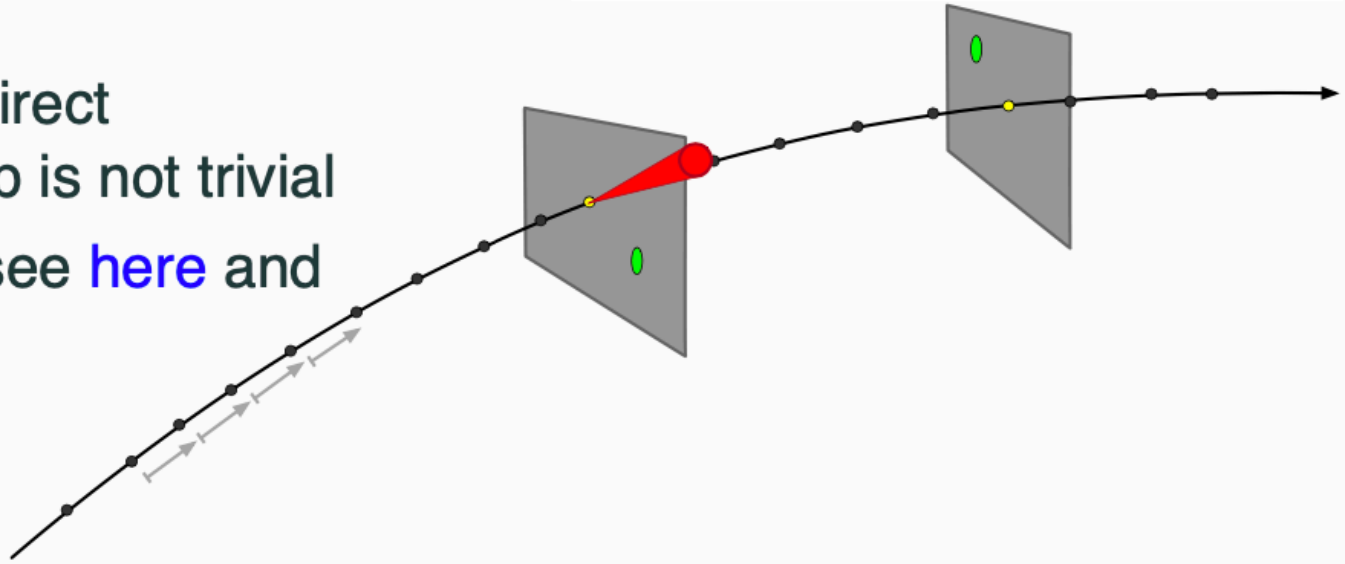

KalmanActor
KalmanActor can/and should
change the direction **but thus invalidates the navigation**
adaptive
step
estimation
Navigator
start parameters
destination surface
Propagator<Stepper, <Magnetic Field>, Navigator>
Extendable<>
result
Options<Aborters, Actors>
Extendable<**SurfaceReached**> Aborter
Extendable<**MaterialInteractor,**
KalmanActor> Actors

KalmanFitter Prototype status

Kalman Filter in Acts

- Kalman Filter is implemented as an extension to the propagator[6]
- 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)

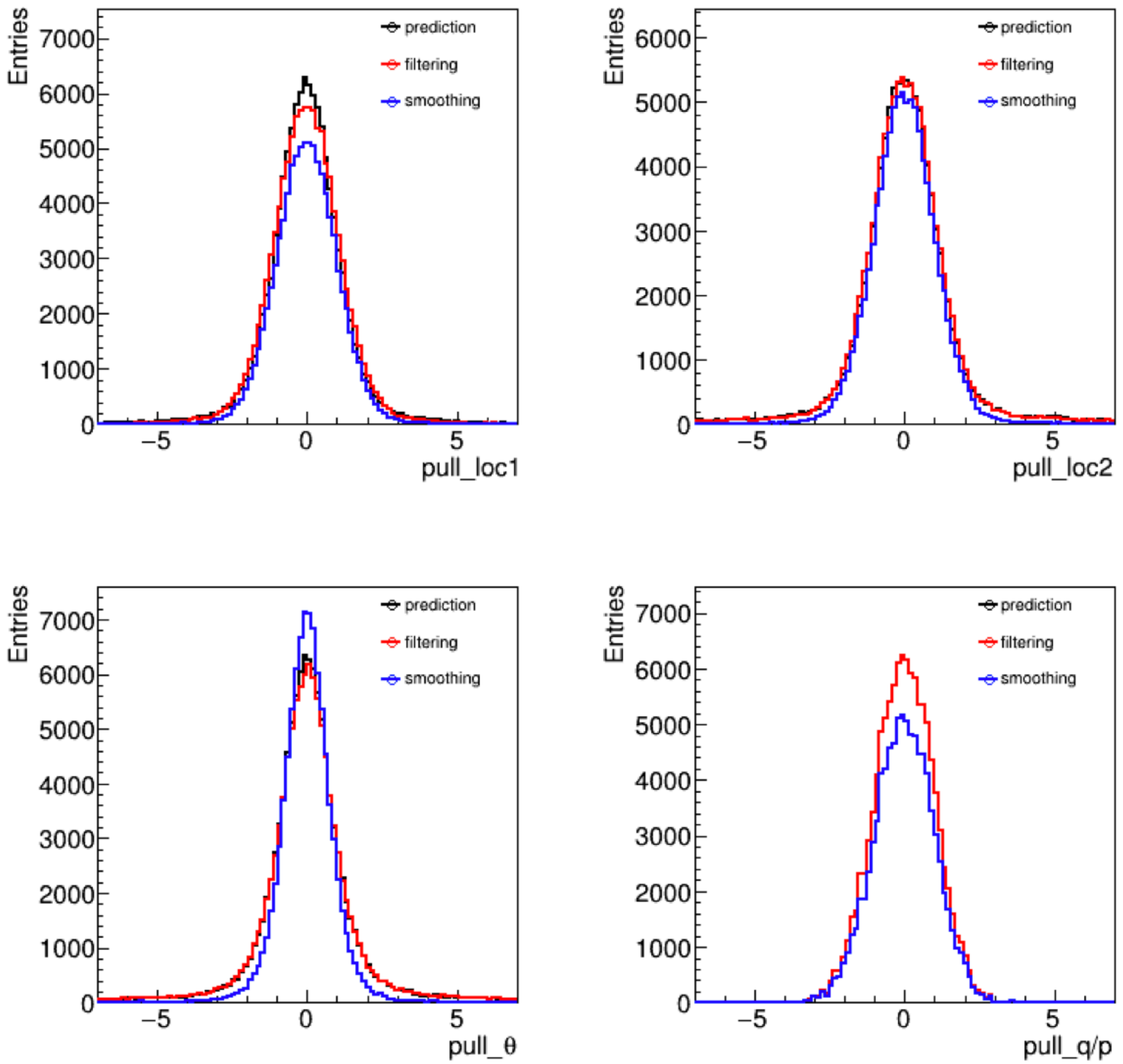




[6] *Actor*

Track parameter pull

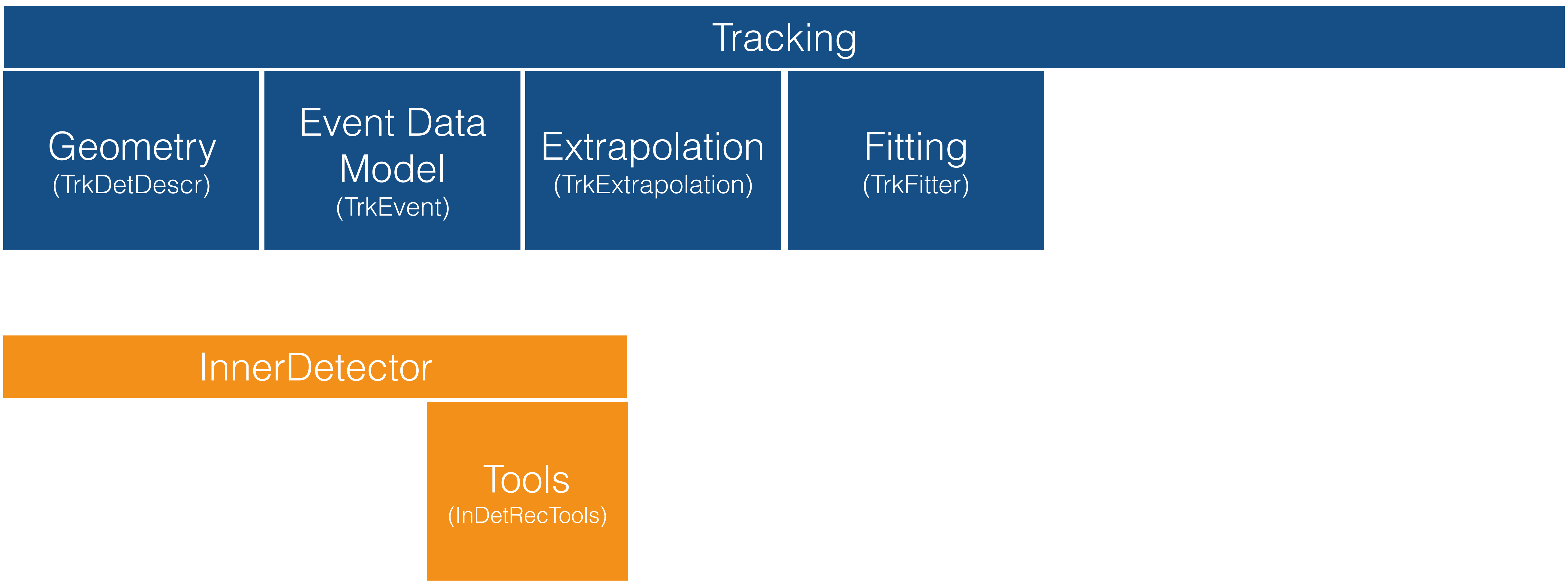
Smoothing with backward filtering (one propagation)



Generic Detector

- PT = 10 Gev
- Eta = [-4, 4]
- Include material effects

Pattern recognition

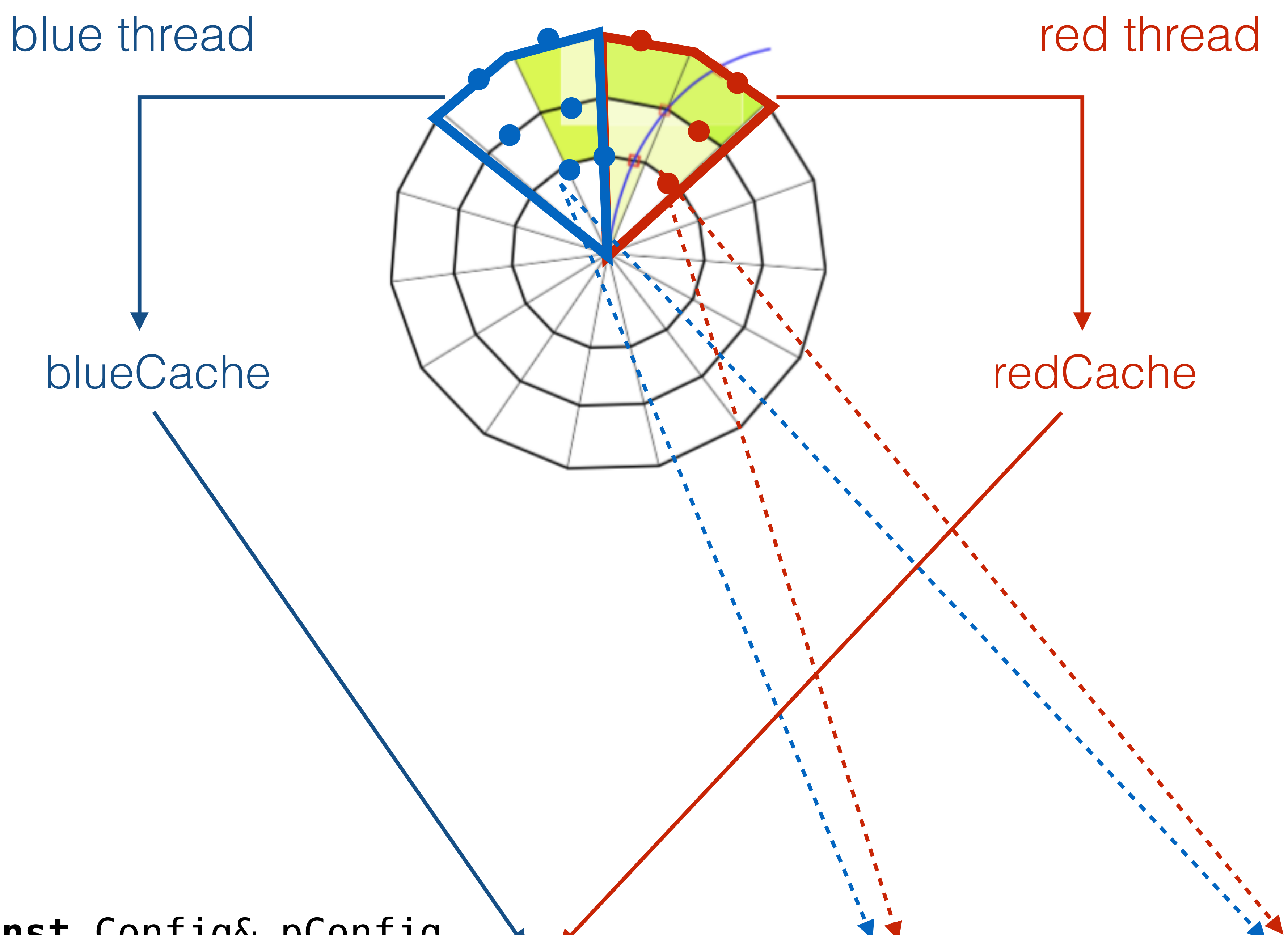


Pattern recognition Strategy

Transcribe ATLAS pattern recognition code into ACTS code pattern

```
namespace Acts {
  /// doxygen documentation
  template <typename iterator_type>
  class PatternReco {
    /// @struct Config for To
    struct Cache {
       Store someCacheStore; ///< necessary cache for pattern reco
    };
    /// method to make the horse run
    /// @param pCache - cache for this pattern
    /// @param pConfig - configuration for this pattern
    /// @param coords - place where the horse should run to
    /// @return a result, horse may drop dead if max path is reached
      const Result seedFinder(
         const Config& pConfig,
         Cache& pCache,
         iterator_type start
         iterator_type end) const;

 };
}
```

Pattern recognition Strategy (1)

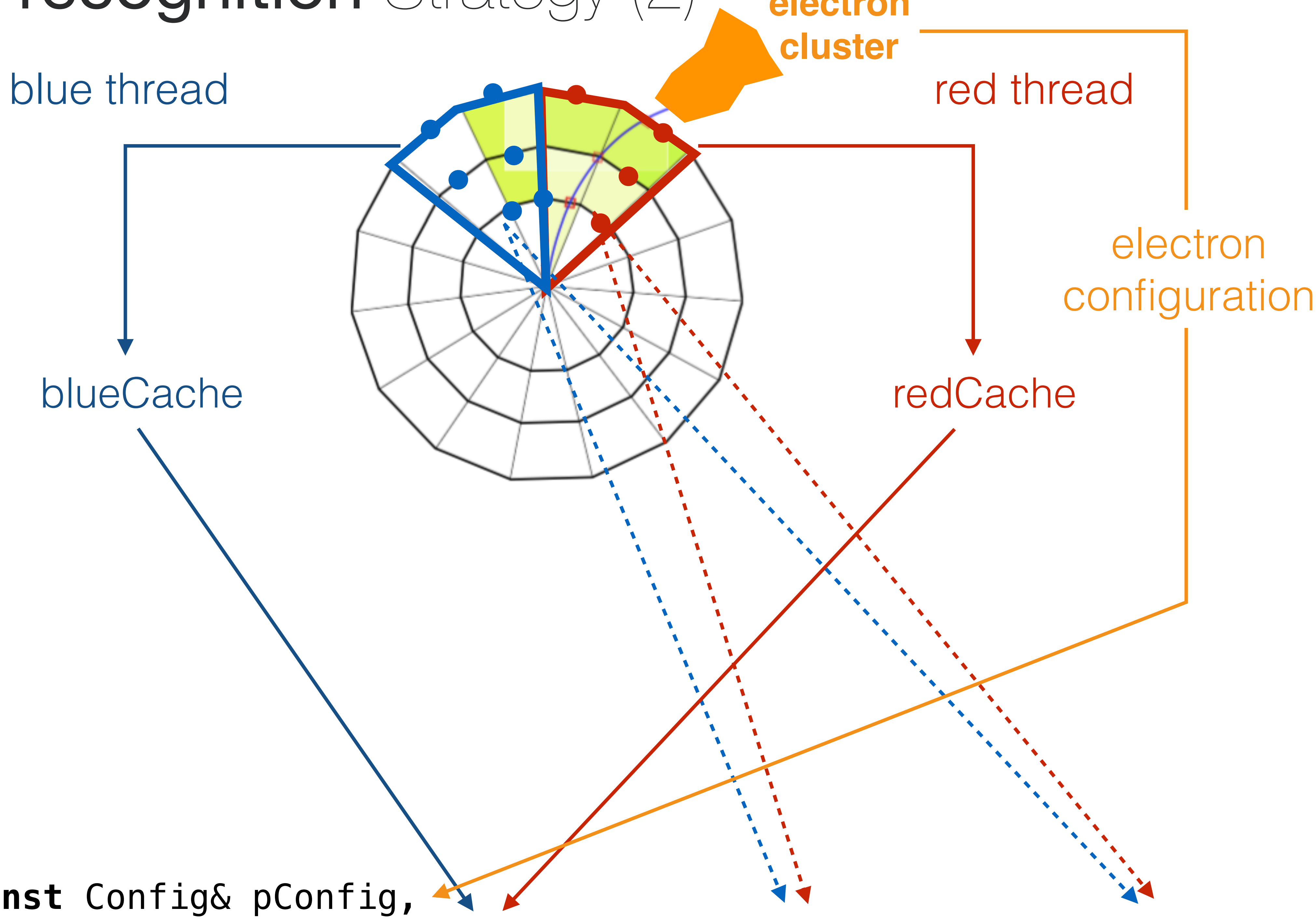



```
seedFinder(const Config& pConfig,
                    Cache& hCache, iterator_type start, iterator_type end)
```

```
SeedContainer
```

Overlapping regions ? Result merging ? Data pre-dividing ? Thread pools ?

Pattern recognition Strategy (2)




```
seedFinder(const Config& pConfig,
                Cache& hCache, iterator_type start, iterator_type end)
```

`SeedContainer`

Overlapping regions ? Result merging ? Data pre-dividing ? Thread pools ?

Seeding Status

Seeding was first module integrated from ATLAS pattern recognition

- all 'magic numbers' documented
- ATLAS specifics have been encapsulated

Tested and runs in AthenaMT

- gives comparable results to ATLAS seeding
- drop-in replacement not straight forward, as seeding in ATLAS is part of a higher level algoritm

```
template <typename SpacePoint>
float ATLASCuts<SpacePoint>::seedWeight(
    const InternalSpacePoint<SpacePoint>& bottom,
    const InternalSpacePoint<SpacePoint>&,
    const InternalSpacePoint<SpacePoint>& top) const {
  float weight = 0;
  if (bottom.radius() > 150) {
    weight = 400;
  }
  if (top.radius() < 150) {
    weight = 200;
  }
  return weight;
}

template <typename SpacePoint>
bool ATLASCuts<SpacePoint>::singleSeedCut(
    float weight, const InternalSpacePoint<SpacePoint>& b,
    const InternalSpacePoint<SpacePoint>&,
    const InternalSpacePoint<SpacePoint>&) const {
  return !(b.radius() > 150. && weight < 380.);
}
```

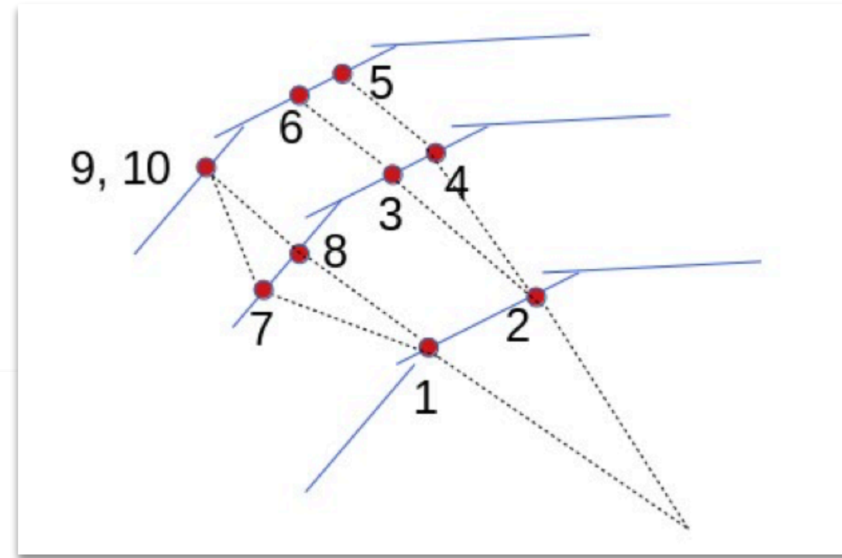
Define a seeding module in ATLAS

- Would also allow ML seeders to be deployed

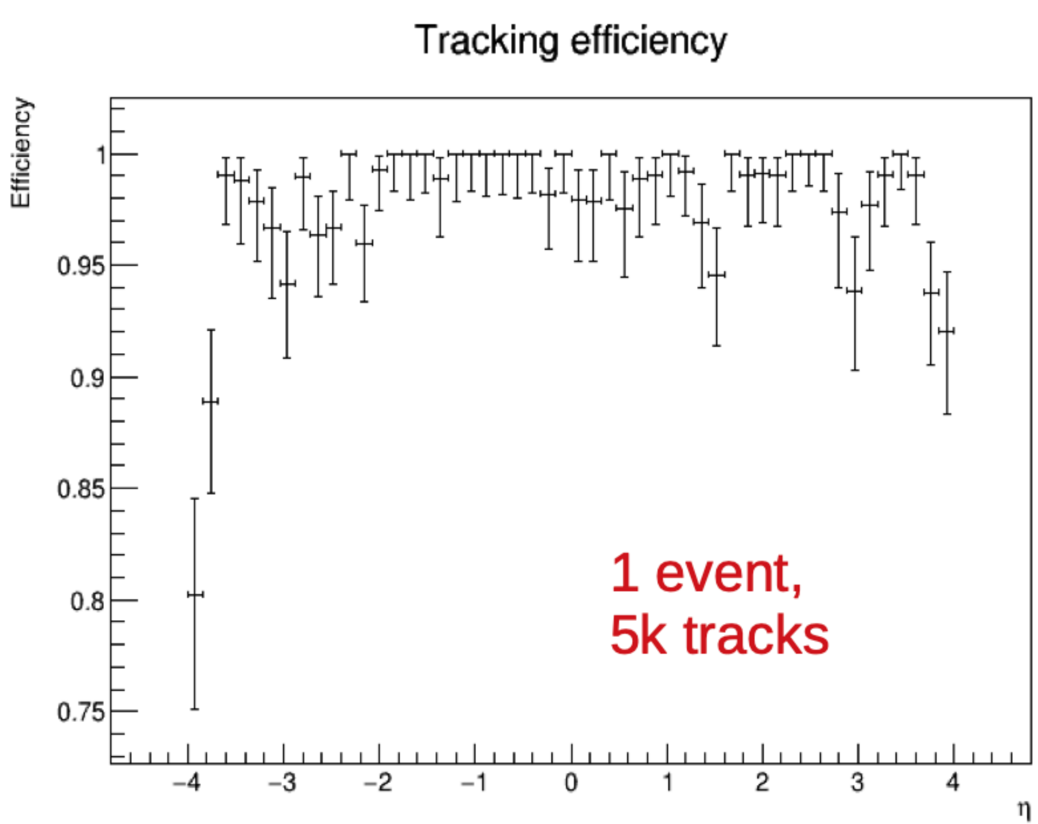
CKF Prototype

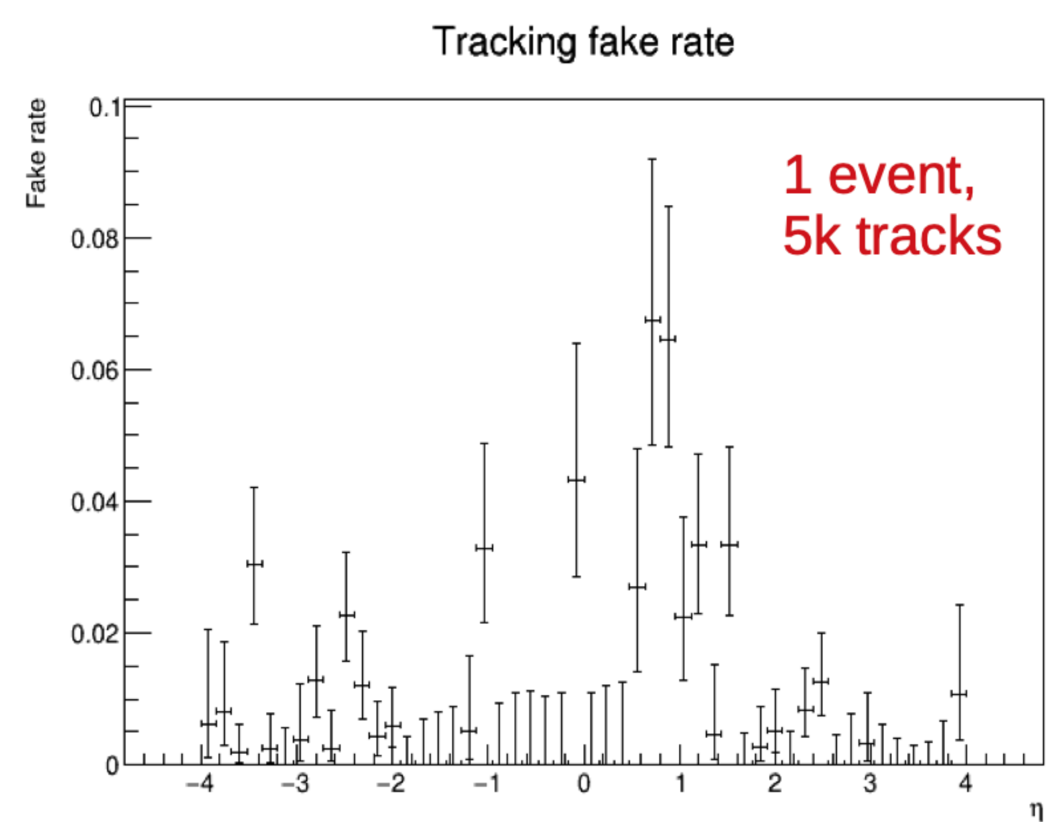
Dedicated development meetings:

https://indico.cern.ch/event/877617/



Efficiency/fake rate





Vertexing New to the family

Tracking

Vertexing
(TrkFitter)

- ☑ Support for various finder and fitters
- ~~☐ Strict division between finder and fitter interface~~
- ☑ Tracking / Analysis data model support

Acts

Vertexing Status

acts › acts-core › Merge Requests › **!535**

Merged Opened 2 months ago by **Bastian Schlag** Edit Report abuse

Iterative vertex finder

Implements the Iterative Vertex Finder together with the ZScanVertexFinder used as the vertex seeding algorithm. Already adapted to 4D vertexing, ready to be merged after !576 (merged).

Edited 1 week ago by Bastian Schlag

Request to merge `iterative_vertex_finder` **into** `master`

acts › acts-core › Merge Requests › **!566**

Open Opened 1 month ago by **Bastian Schlag** Edit Close merge request

WIP: Multi adaptive vertex fitter

Implements the multi adaptive vertex fitter, depends on some features from !535 (merged)

Edited 1 month ago by Bastian Schlag

Request to merge `MultiAdaptiveVertexF…` **into** `master`
The source branch is 62 commits behind the target branch

Open in Web IDE Check out branch

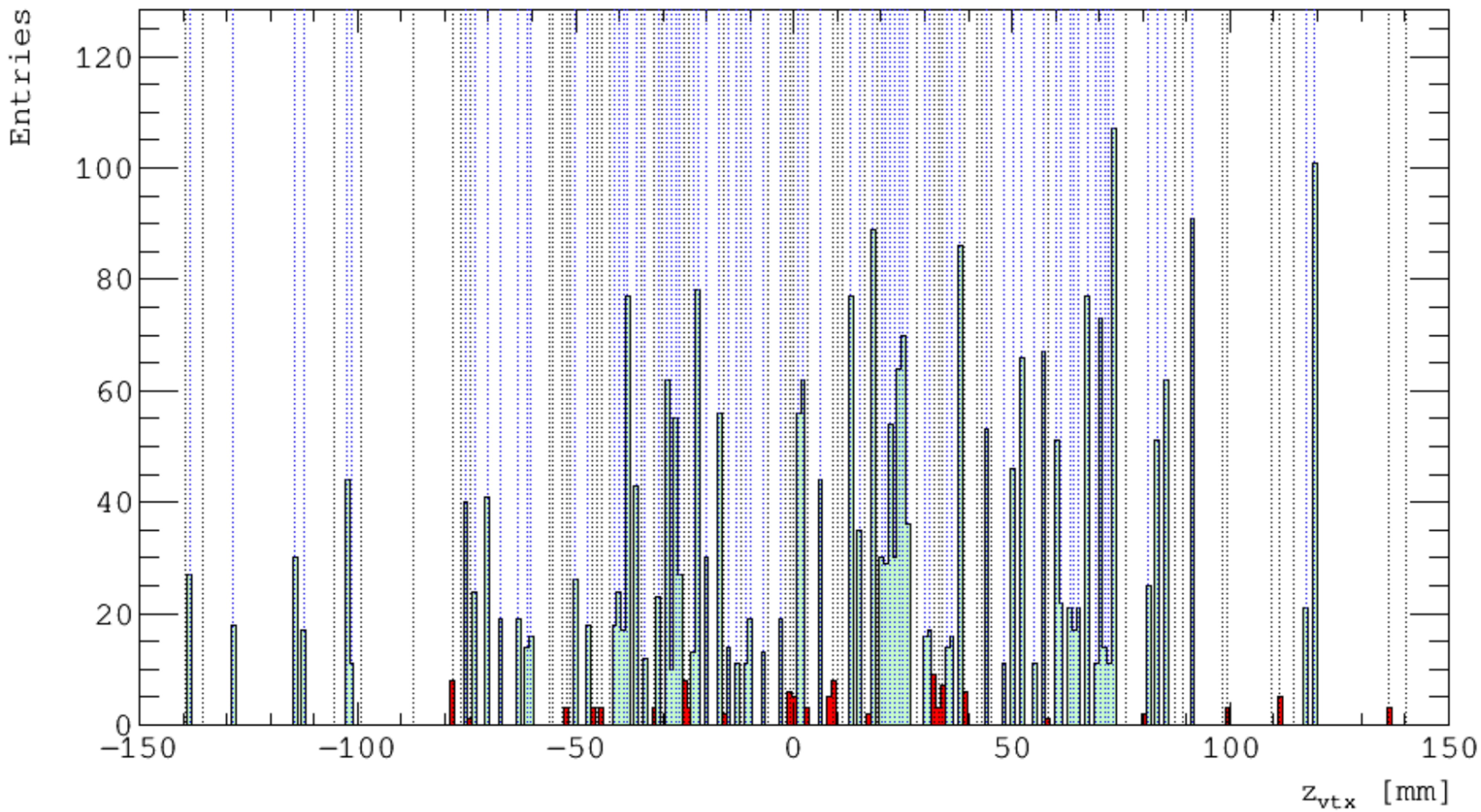
Now iterating on the design and optimising

Full vertexing suite available now

z-Finder

Original module from ATLAS Atlantis

- fast (ms) z-finder on space points
- Code is fully implemented but not yet imported into acts



Chapter Three Configuration & integration

Status Binding to detector software & framework

Acts designed to have minimal overhead when being integrated in detector software

Algebra library is Eigen but dependencies are minimal

- may change to a template implementation (if beneficial)

No dependency on Identifier

- Detector calibration is resolved in detector geometry

Screen logging can be replaced by ld pre-loading

- needs a simple struct on the detector framework side that provides a logger() method.
- tested with different loggers:
  - Acts logger in acts-framework
  - Gaudi logger within FCCSW

Status Binding to framework configuration

ACTS tools have a nested configuration struct:

```
namespace Acts {
  /// doxygen documentation
  class WorkHorse {
    /// @struct Config for To
    struct Config {
       float coatColor; ///< configure the coat color
       float maxPath;    ///< set the max path this horse can run
    };
  };
}
```

These structs are then configured by the detector framework, e.g. through Gaudi/Athena

```
 /// feed from Framework into ACTS configuration
 declareProperty(“CoatColor”, m_cfg.coatColor);
 declareProperty(“MaxPath”,   m_cfg.maxPath);
```

tested with Gaudi for FCCSW & AthenaMT

Configuration Strategy

Nested configuration struct by convention

```
namespace Acts {
  /// doxygen documentation
  class SomeComponent {
    /// @struct Config for this Component
    struct Config {
       bool run_faster = false; ///< configuration flag
    };
  /// Constructor with config object
   SomeComponent(Config& cfg);
 };
}
```

Inside the framework Wrapper

```
#include "ACTS/Package/SomeComponent.hpp"

…
  /// create the config sruct
  Acts::SomeComponent::Config scConfig;

  /// bind to your framework configuration
  declareProperty("RunFastVersion", scConfig.run_faster);
   Acts::SomeComponent sc(scConfig);
```

Chapter four multi-threading

Concurrency Strategy

const-correctness

- [ ] **Remove every use of "mutable" in ACTS**
!265 · opened 3 days ago by Hadrien Grasland
1 1 9
updated 3 days ago

statelessness engines

- cache visitor pattern for calls that need to run concurrently

```
namespace Acts {
  /// doxygen documentation
  class WorkHorse {
    /// @struct Cache for the WorkHorse
    struct State {
       float accumulatedPath = 0.; ///< the passed path so far
    };
    /// method to make the horse run
    /// @param hState - cache tracker for this horse
    /// @param coords - place where the horse should run to
    /// @return a result, horse may drop dead if max path is reached
    const RunResult run(State& hState, const Vector3D& coords) const;
 };
}
```

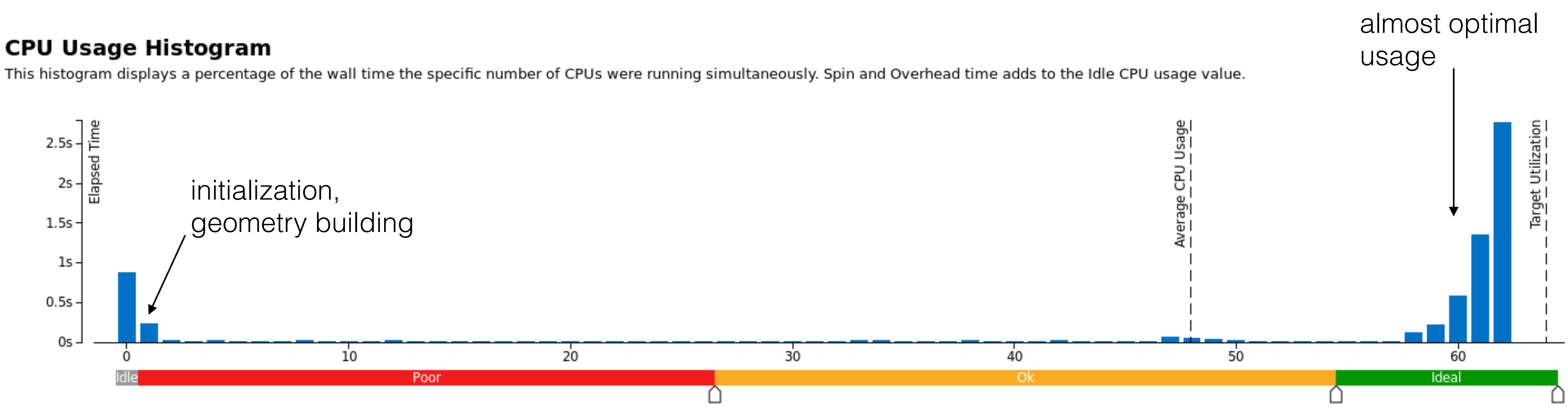
Concurrency Tests

Acts test framework runs with TBB multithreaded mode

- running extrapolations through a test detector
- test programs are run using a single threaded setup vs. multi-threaded event processing
- this consistency check is part of the acts-framework CI

CERNopenlab

Intel Xeon e5-2698 v3, 2 sockets
32 Cores, 2 threads per core
64 Processors(cpu's)



ACTS with Context

Introduced context objects in `acts-core` & testes in `acts-framework`

- nomen est omen

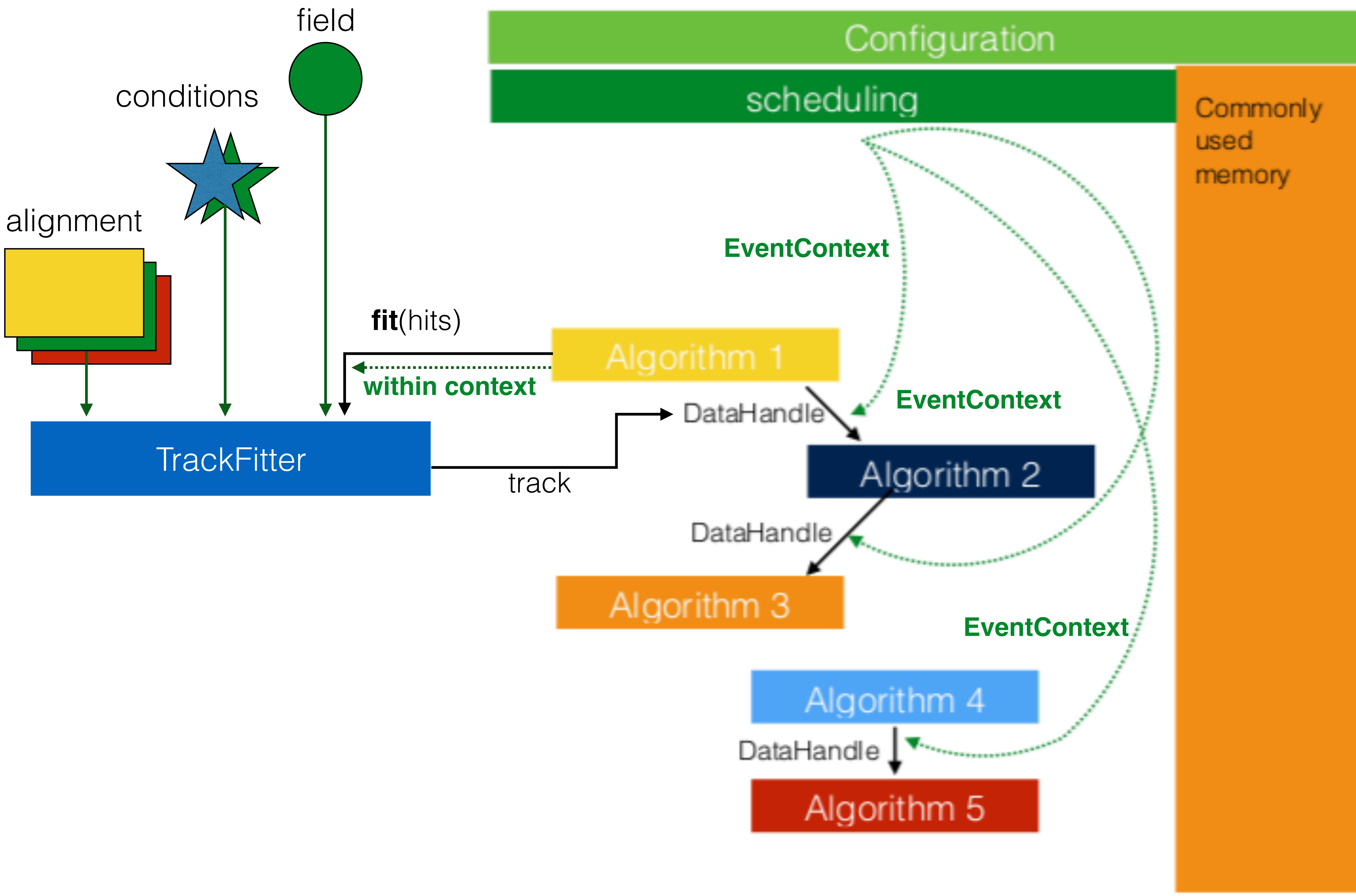
```
/// Aggregated information to run one algorithm over one event.
struct AlgorithmContext
{
  size_t                       algorithmNumber;    ///< Unique algorithm identifier
  size_t                       eventNumber;       ///< Unique event identifier
  WhiteBoard&                  eventStore;        ///< Per-event data store
  Acts::GeometryContext        geoContext;        ///< Per-event geometry context
  Acts::MagneticFieldContext   magFieldContext;   ///< Per-event magnetic Field context
  Acts::CalibrationContext     calibContext;      ///< Per-event calibration context
};
```

While they are untouched in `acts-core` and simply defined as

```
#pragma once

/// Set the identifier PLUGIN
#ifdef ACTS_CORE_GEOMETRYCONTEXT_PLUGIN
#include ACTS_CORE_GEOMETRYCONTEXT_PLUGIN
#else
#include <any>

namespace Acts {

using GeometryContext        = std::any;
using DefaultGeometryContext = GeometryContext;

}  // namespace Acts

#endif
```

← can even be overloaded (pointing to `#ifdef ACTS_CORE_GEOMETRYCONTEXT_PLUGIN`)

Contextual Detector The “clean” solution



Parallelism testbed

Test with different alignment every single event

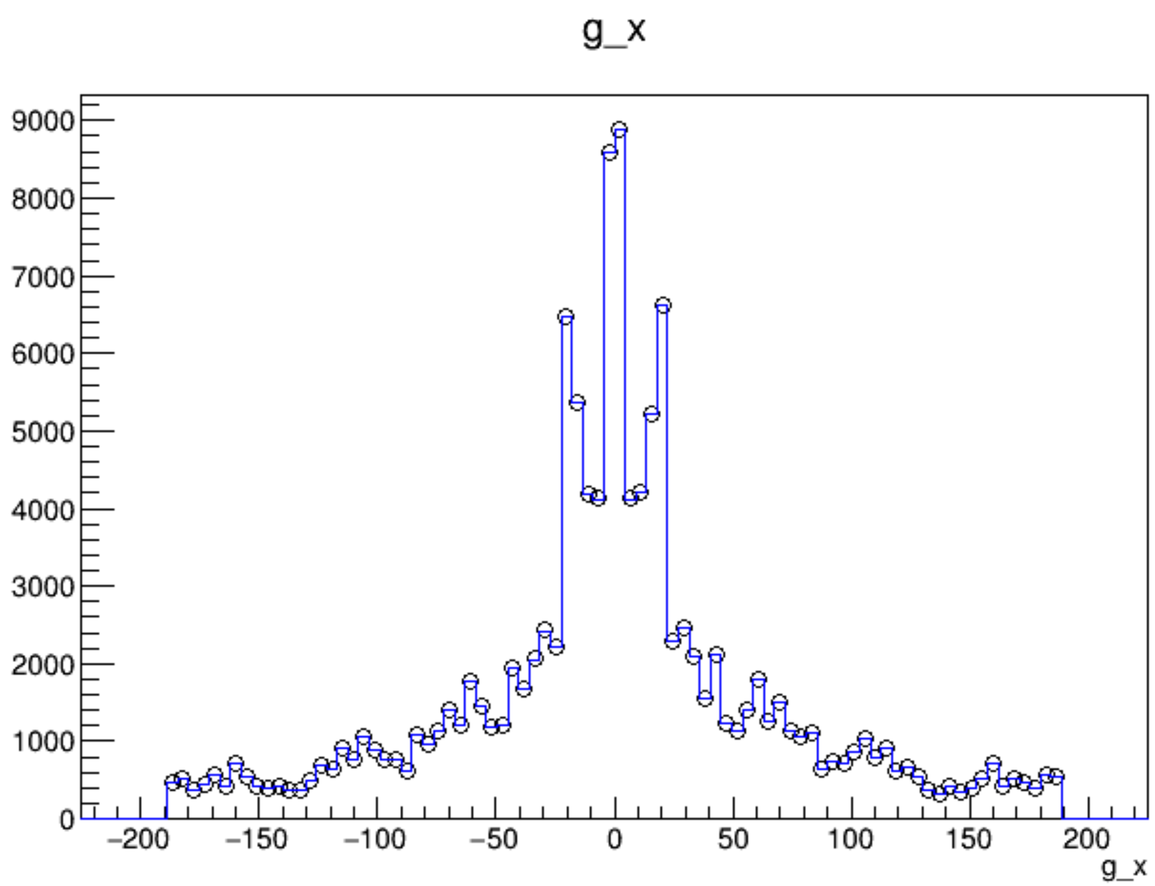
```
salzburg$ export ACTSFW_NUM_THREADS=1
salzburg$ ./ACTFWAlignablePropagationExample -n10 --prop-ntests 1000 --bf-values 0 0 2 --output-root 1
12:49:10    Sequencer      INFO      Added context decorator GeometryRotationDecorator
12:49:10    Sequencer      INFO      Added service RandomNumbersSvc
12:49:10    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
```

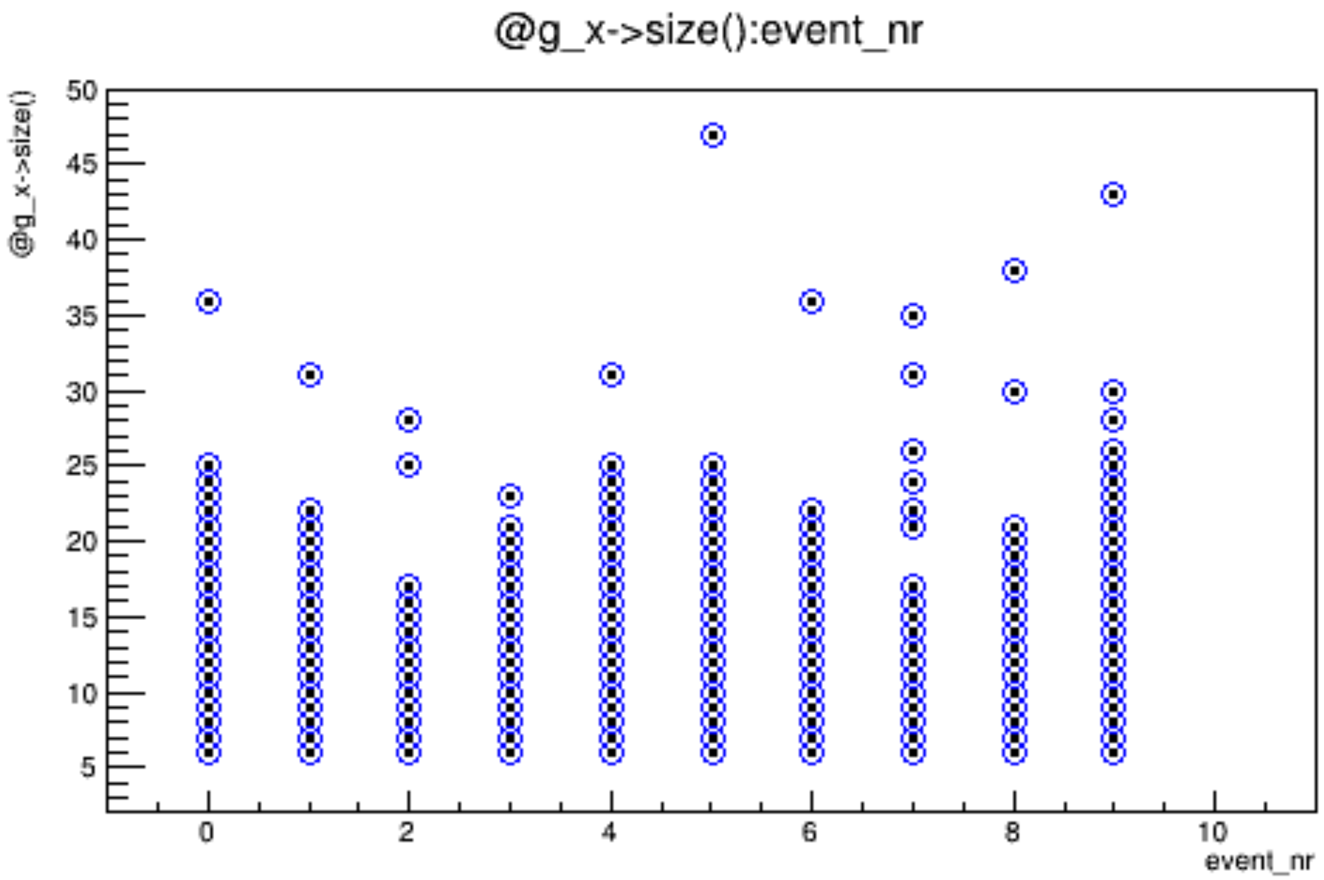
12 seconds → 5 seconds

GeometryContext Comparing the two

Total comparison:

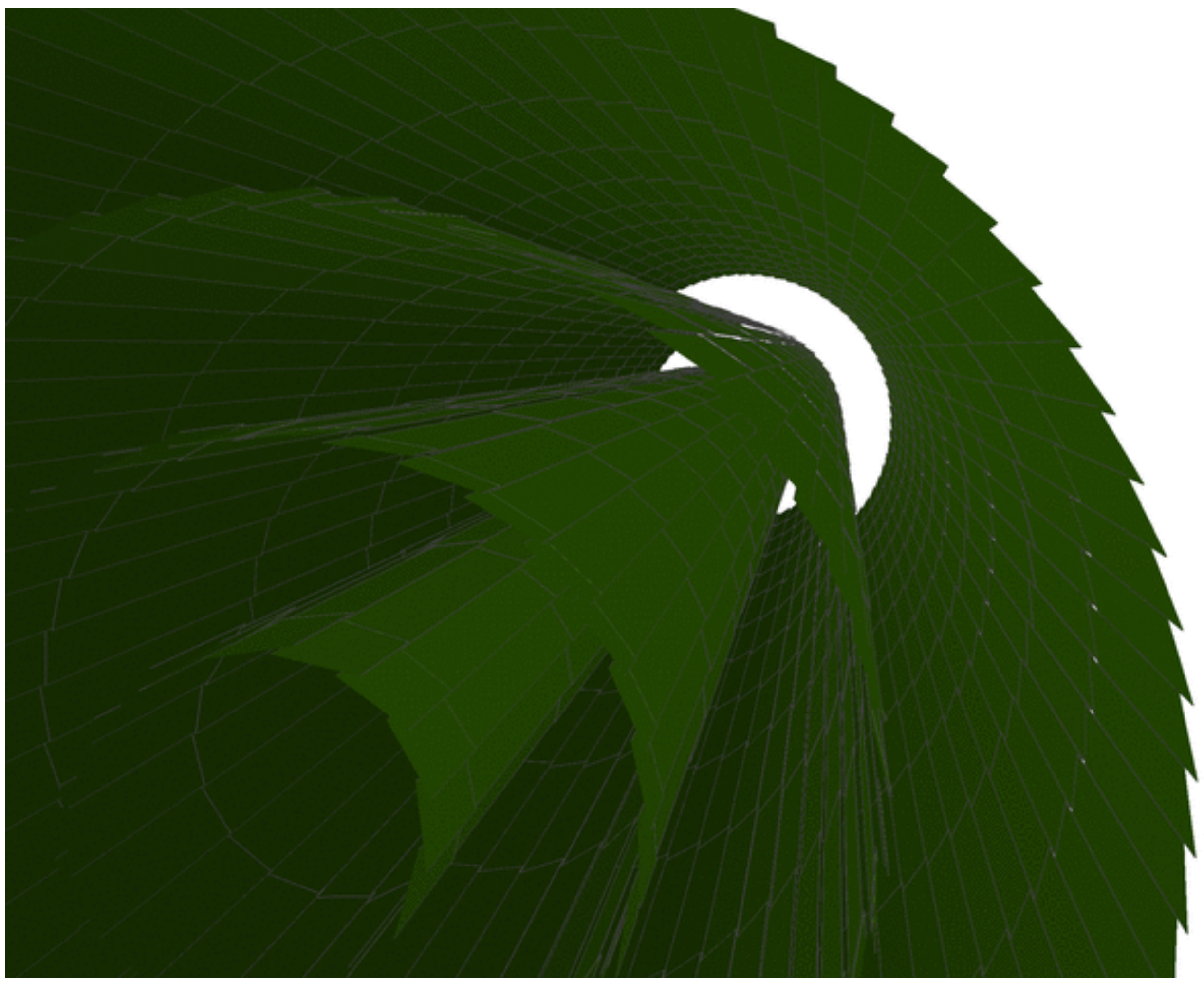


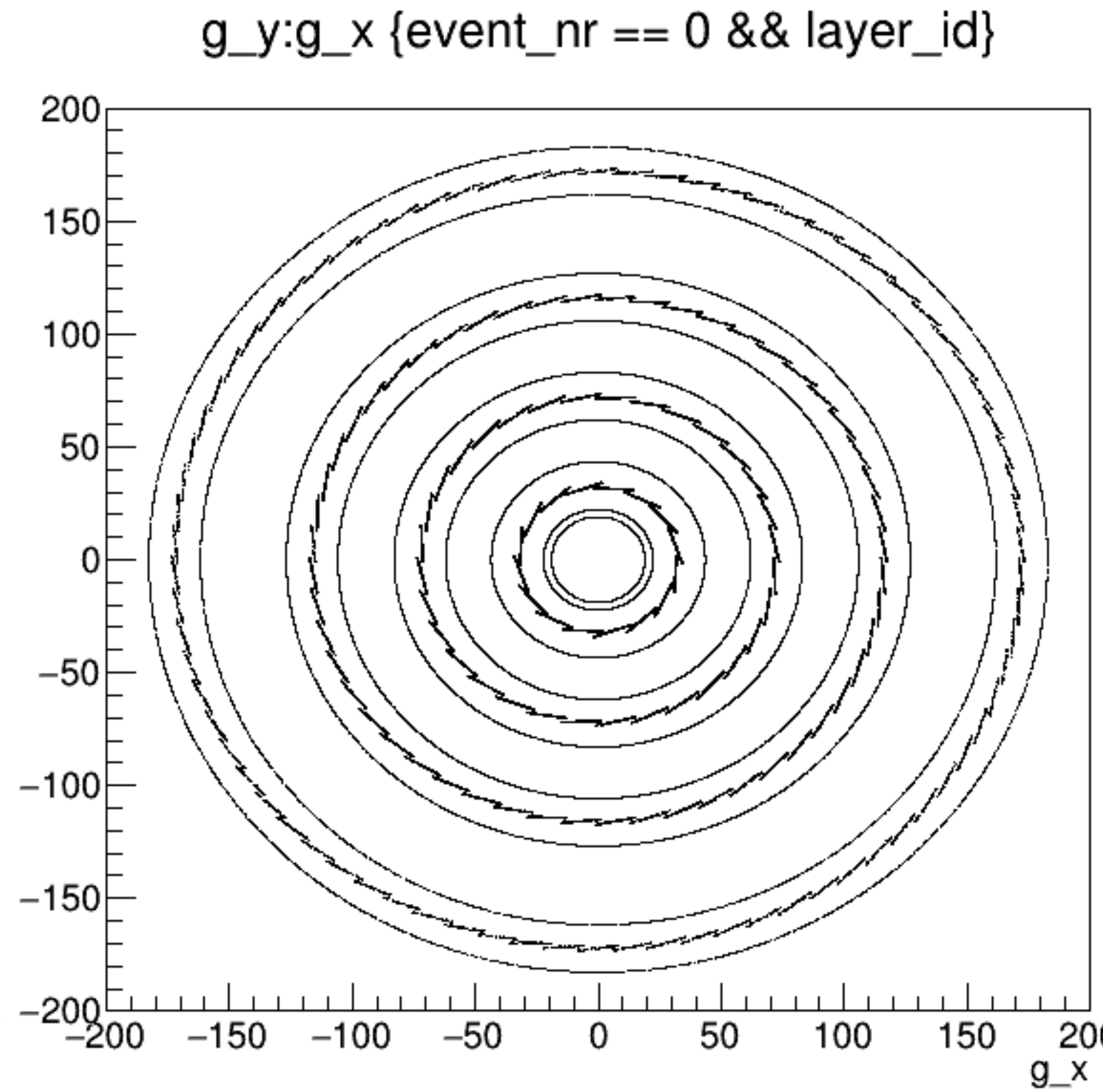
Per event comparison:



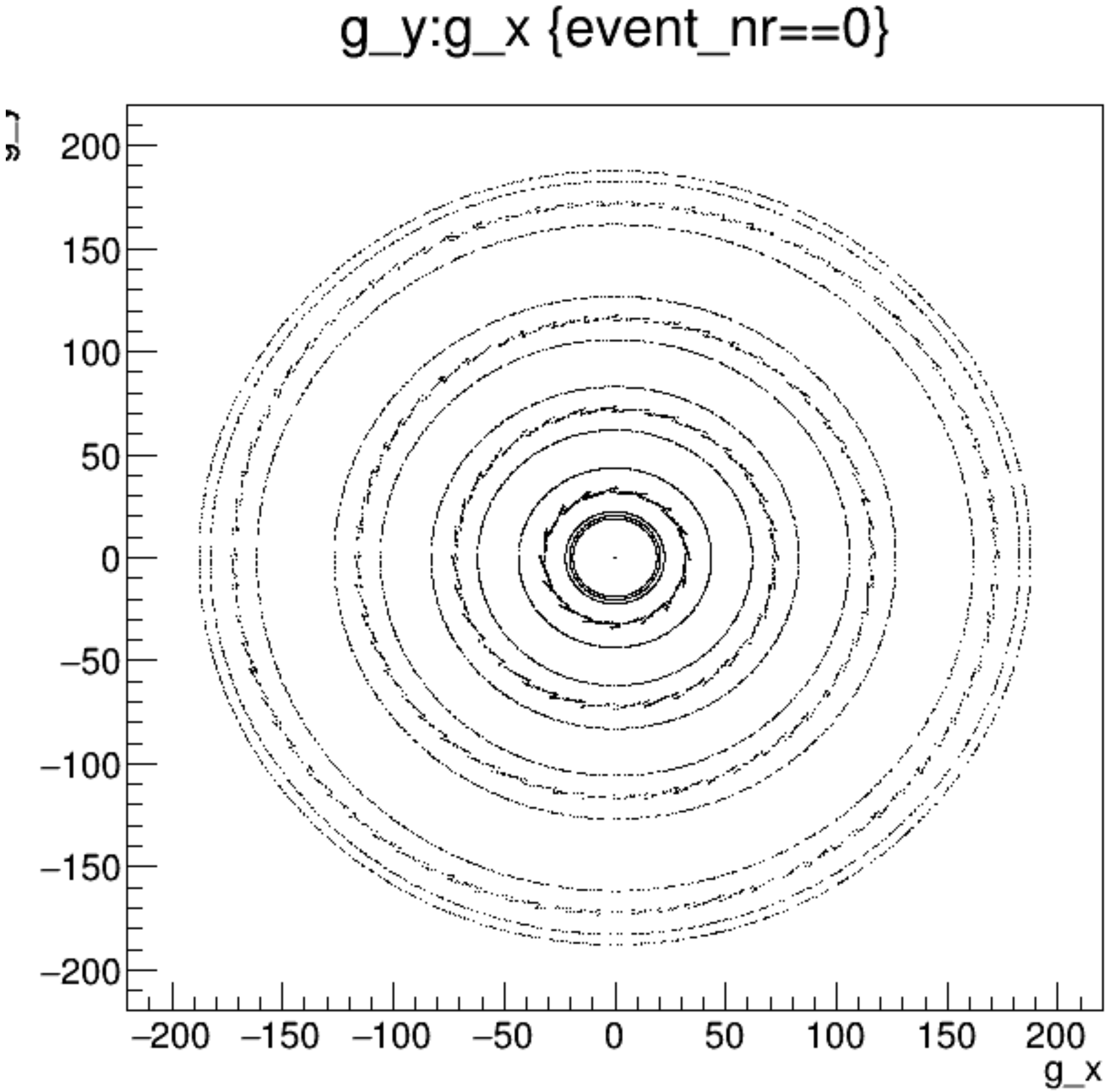

```
salzburg$ export ACTSFW_NUM_THREADS=1
salzburg$ export ACTSFW_NUM_THREADS=4
```

GeometryContext In Action





MagneticFieldContext In Action



Detectors & Framework Current support

FCC-hh detector

- via DD4Hep geometry description
- in Gaudi event processing framework

ACTS generic detector (TML)

- via python input
- in acts-framework

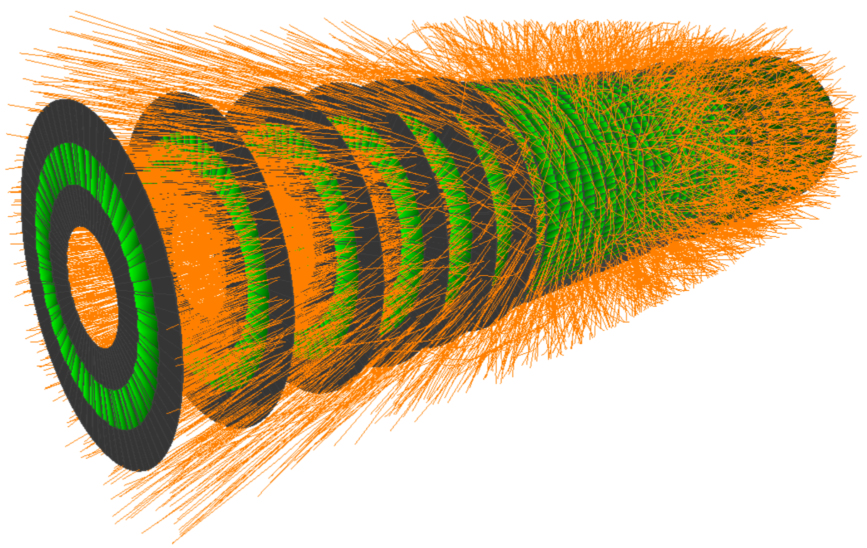
ATLAS Pixels and ITK detector

- currently via GDML input
- in acts-framework

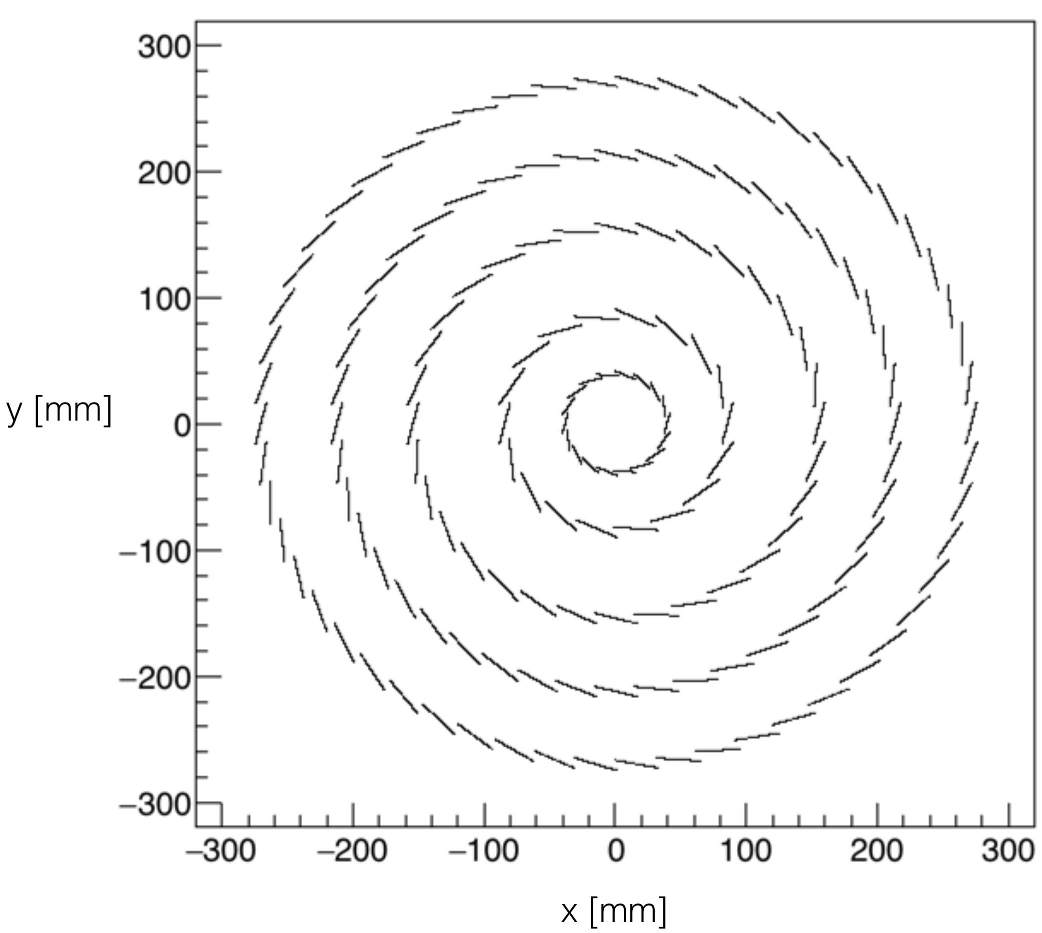
Geometry Plugins available

- DD4Hep plugin
- TGeo plugin

Tracking ML detector with 1000 events
ACTS fast simulation



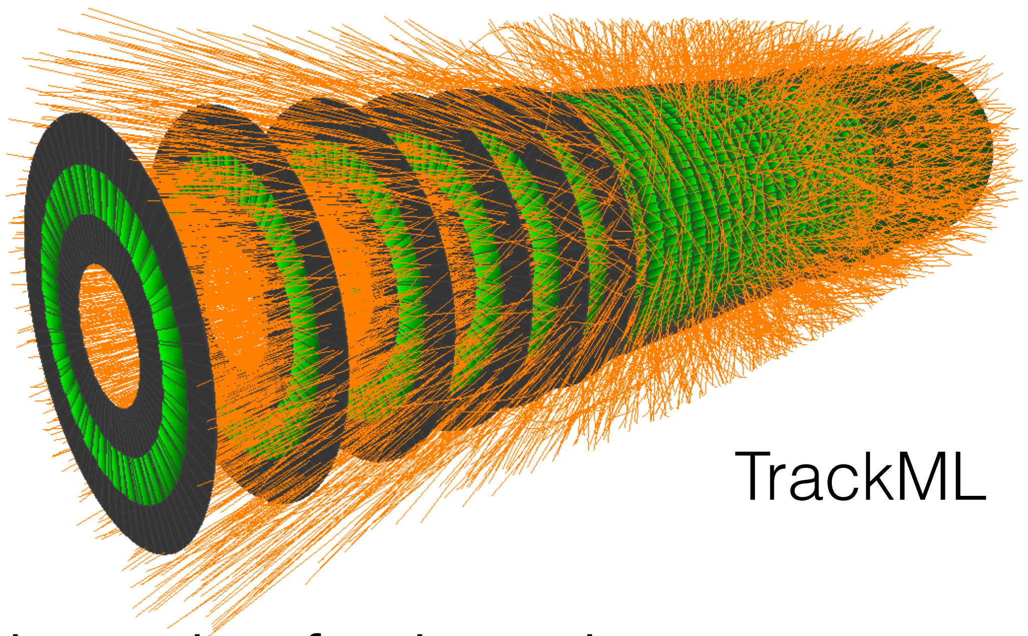
ATLAS ITk Pixel Barrel, sensitive hits



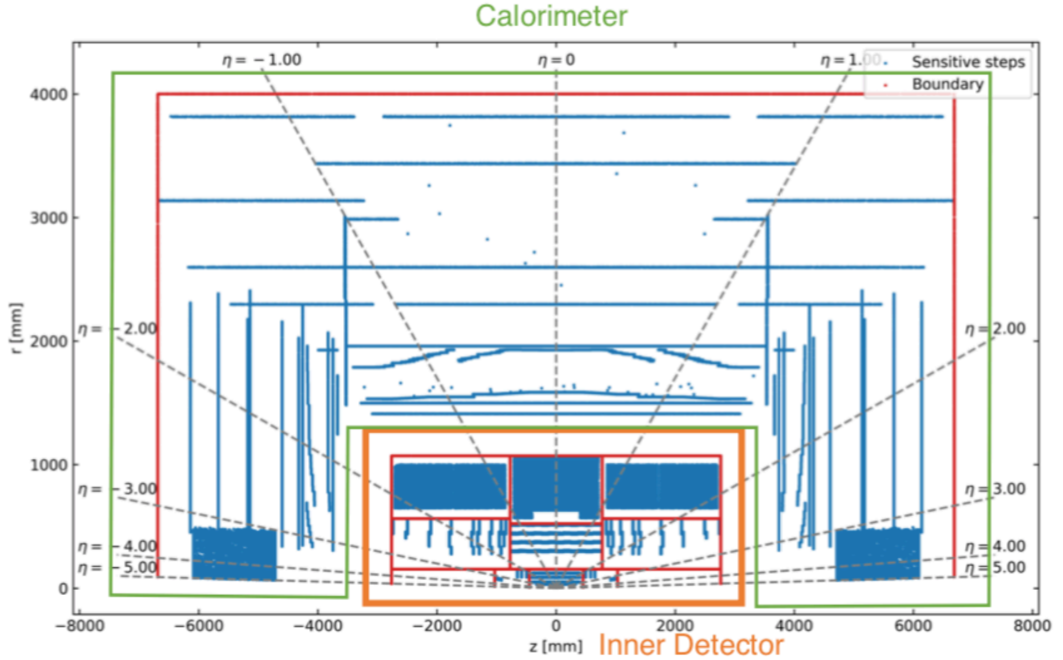
TrackML Aftermath

Started to port first TrackML algorithms into `acts-framework`

- Idea is to create a testbed for algorithm development and templating
- provide several detectors to test on

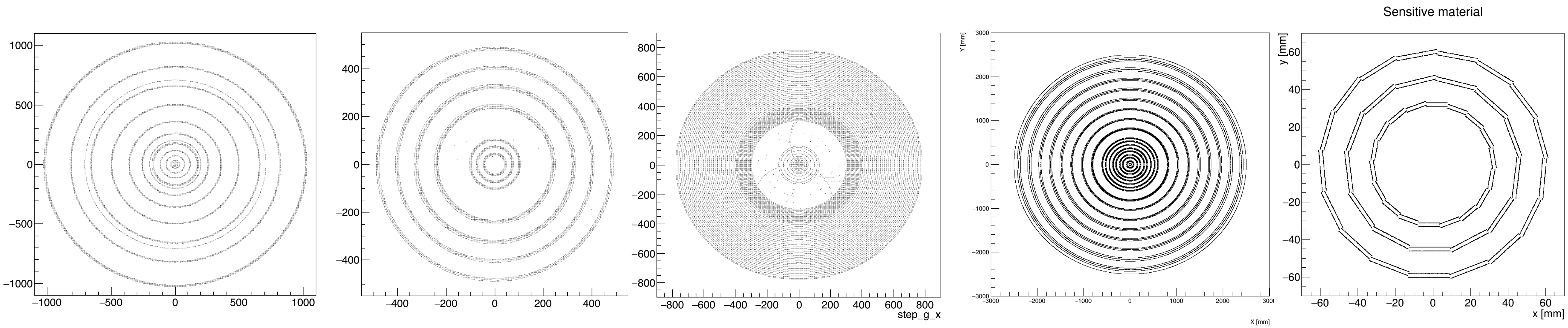


TrackML



ATLAS

A bunch of other detectors:

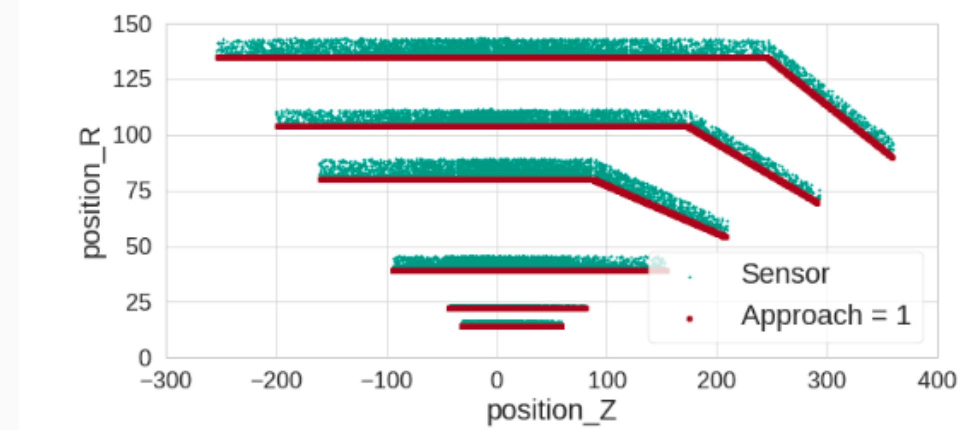


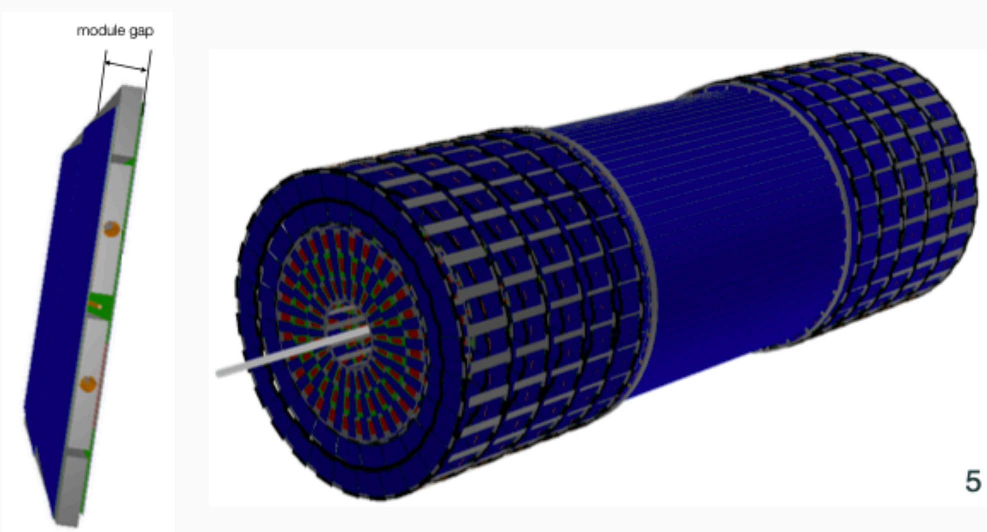
Examples Detectors in Acts

Contributions and developments

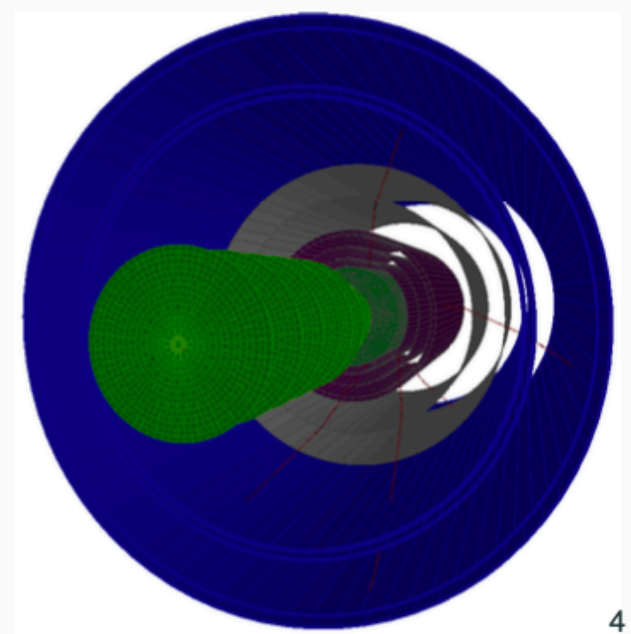
- Contributions mostly from ATLAS so far, but from outside too: **Belle-2**, **FCC-hh**
- **TrackML challenge** phase 2 completed[2]
- **OpenDataDetector** for open dataset under development

Belle-2 VXD



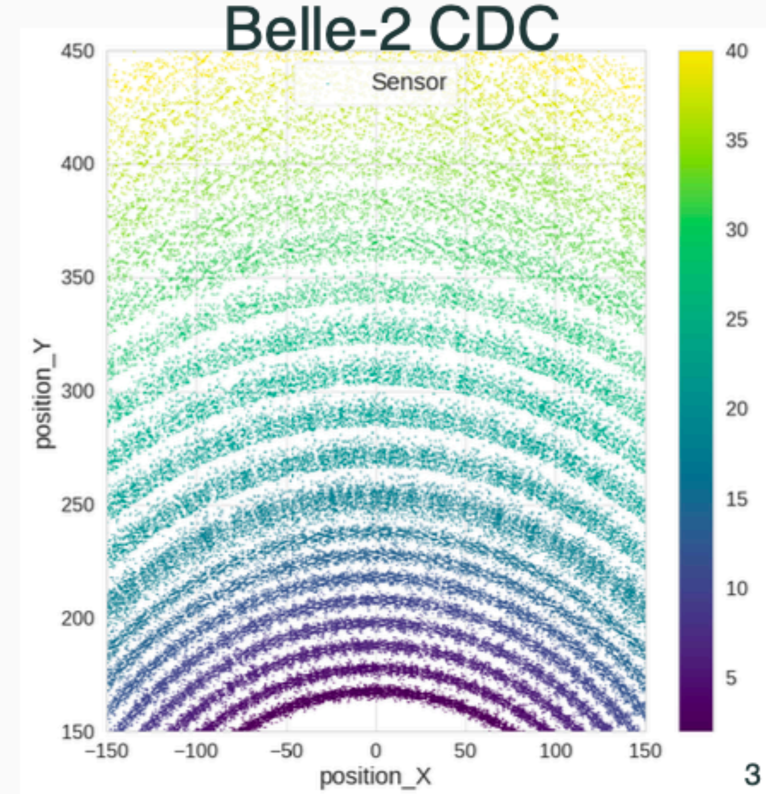


5


4

FCC-hh tracker

Belle-2 CDC

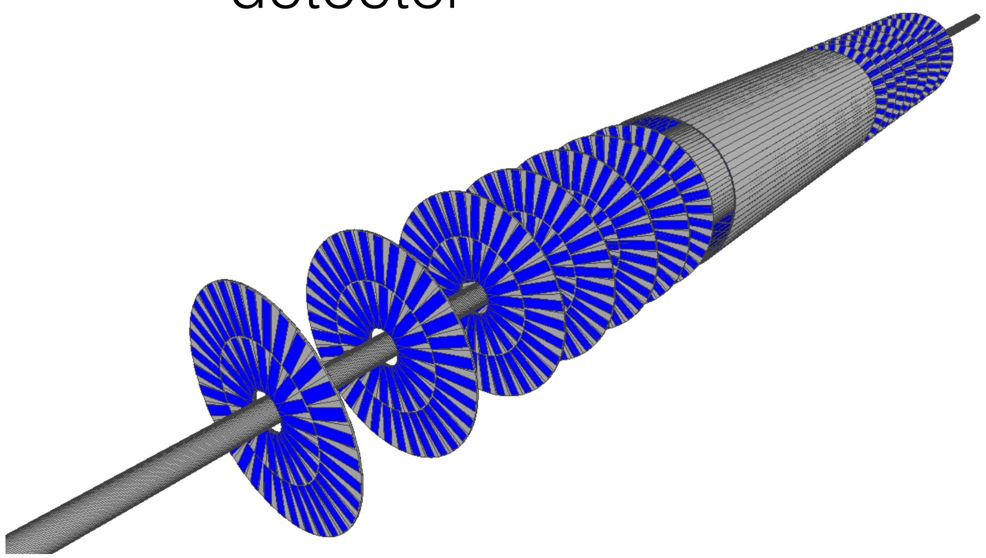


3

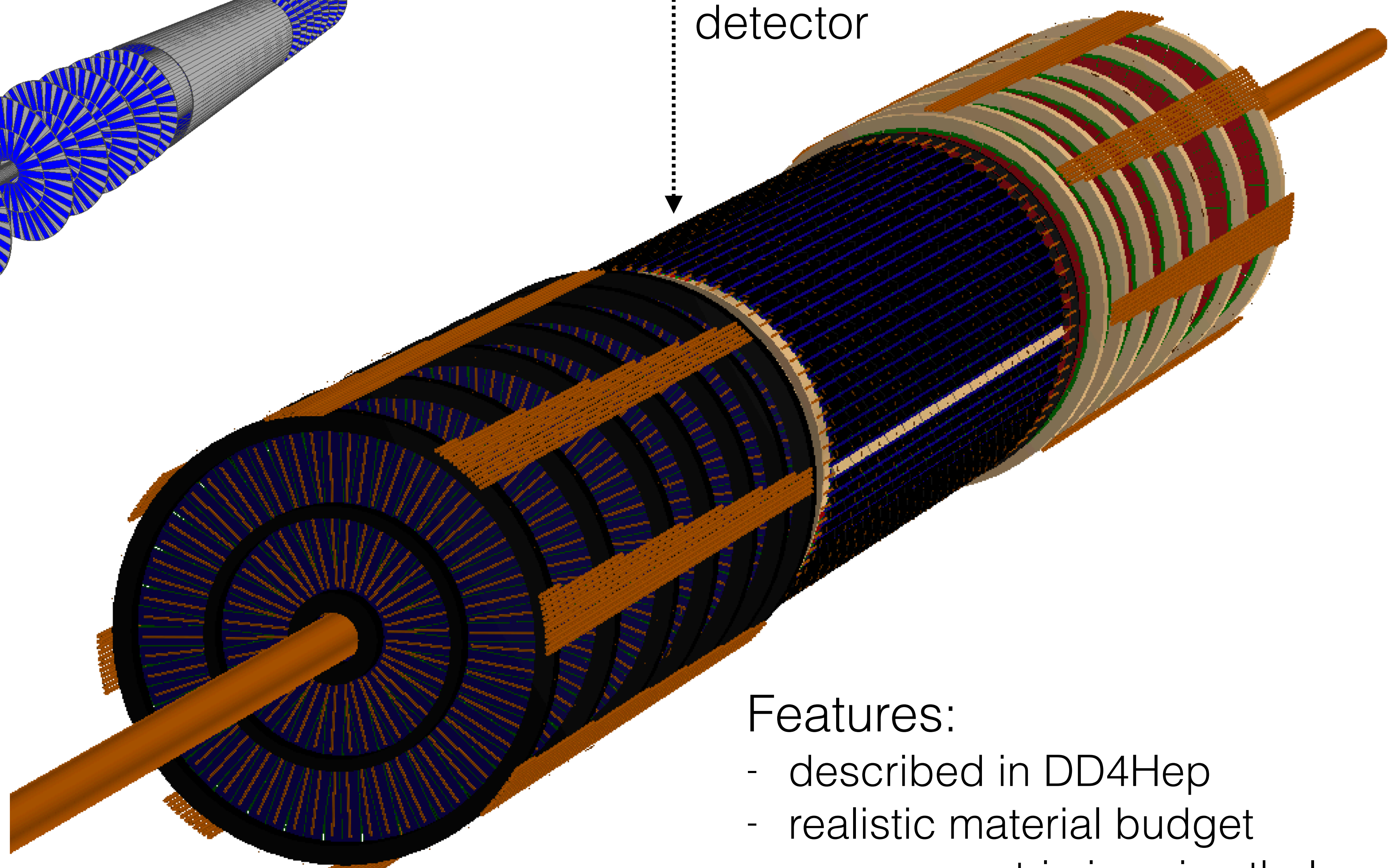
[2] Grand Finale in July 2019
[3] Talk by N. Braun
[4] CERN-THESIS-2019-136, J. Hrdinka
[5] A. Salzburger

OpenData Detector

TrackML Pixel detector



OpenData Pixel detector



Features:

- described in DD4Hep
- realistic material budget
- non-symmetric in azimuthal angle
- full (G4) and fast (ACTS) simulation
- misalignment possibility

A brain storming list for sPHENIX

Geometry & material description

- finish material maps (first fully layer based),
  will try to implement/demonstrate that on Wednesday
- Volume based material for TPC (faster)

Import z-finder into framework

- adapt to sPhenix parameters

Create truth tracking example

- by-passing of pattern recognition

Running vertex reconstruction example