Overview of the ACTS project

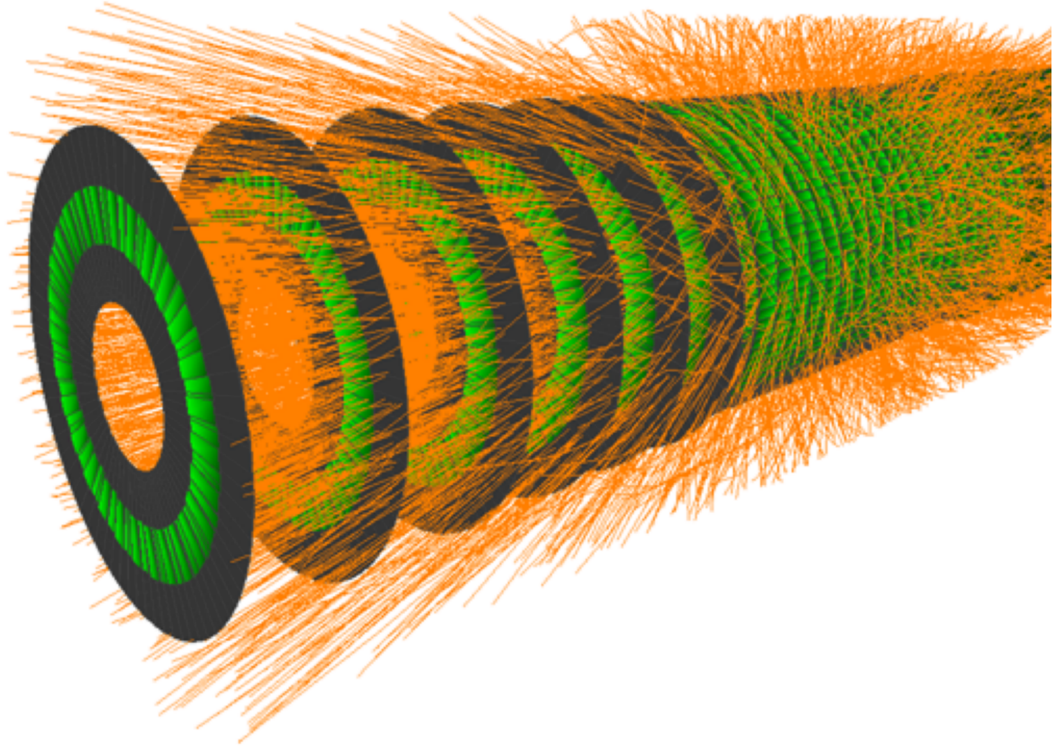
Moritz Kiehn
Université de Genève

for the ACTS Developers

HEP Tracking Workshop, Berkeley, 14.01.2019

http://cern.ch/acts

A high level overview



A Common Tracking Software

You call us,
we don’t call you

A standalone C++ software library for tracking

Minimal dependencies

Sharable between experiments

Not bound to experiment schedule

Geometry & Navigation

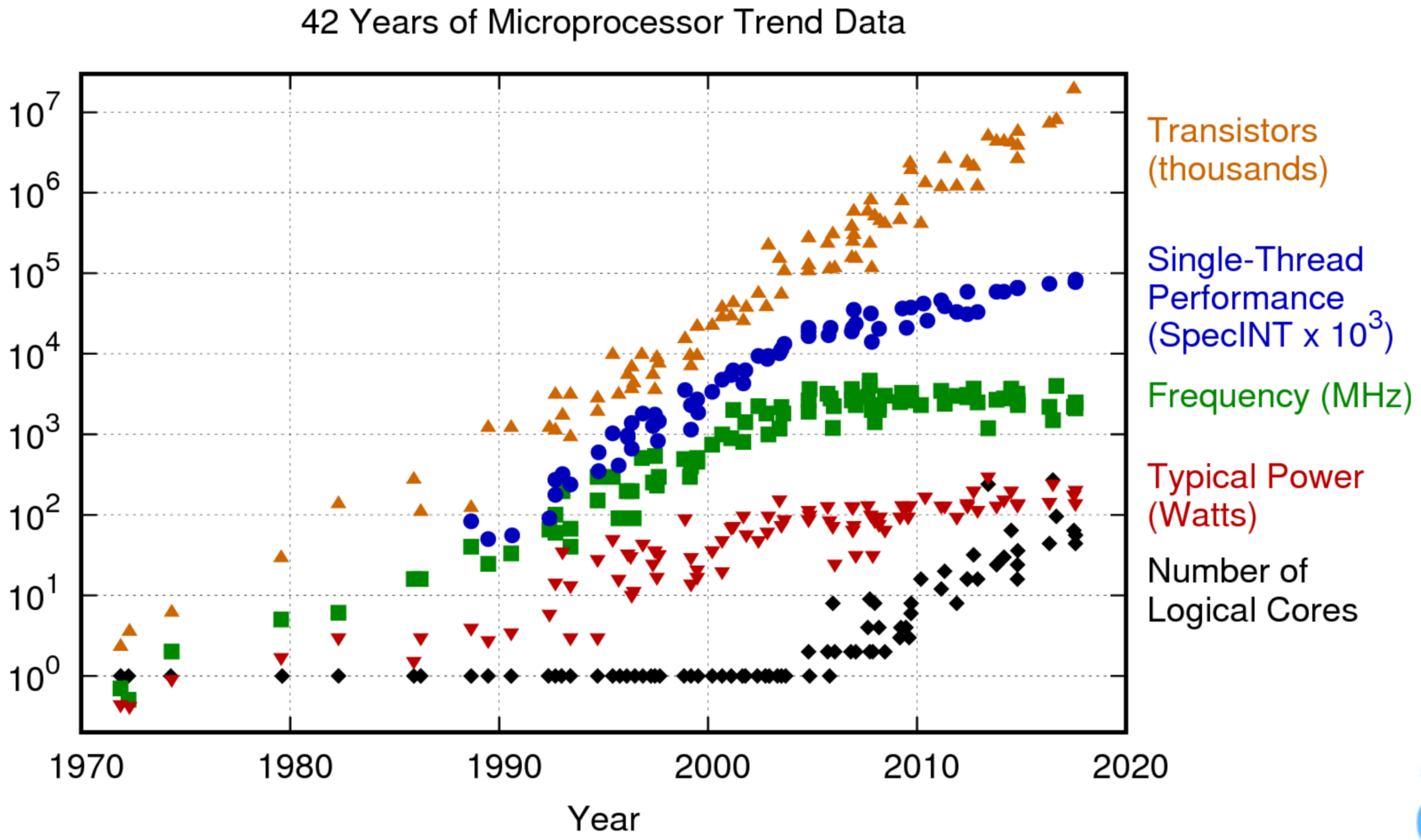
Propagation

Track finding & fitting

Vertexing

…

Parallelization is necessary



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

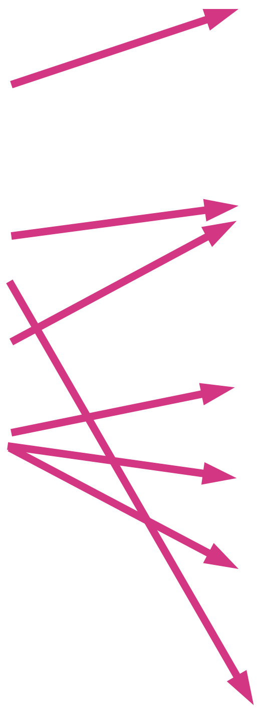
Goals and guidelines

Retain/improve physics performance

Increase maintainability

Increase usability

Allow parallelization/ vectorization



Base on established code (ATLAS), but with simplified design

Use modern C++ (14/17), e.g.

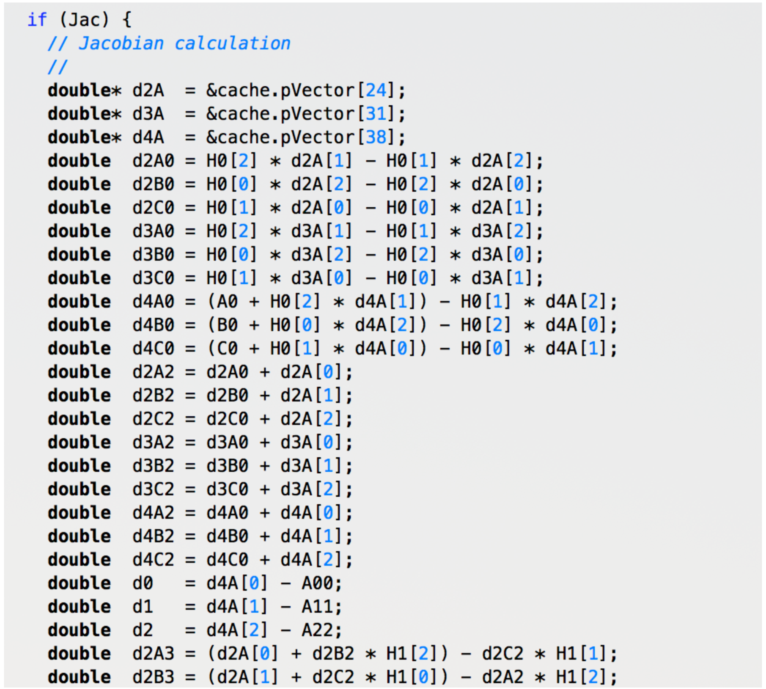
- std::shared_ptr et. al.
- strict const-correctness
- data is shared^mutable

Use Eigen linear algebra library

Unit and regression tests

Maintainability

ATLAS Stepper implementation

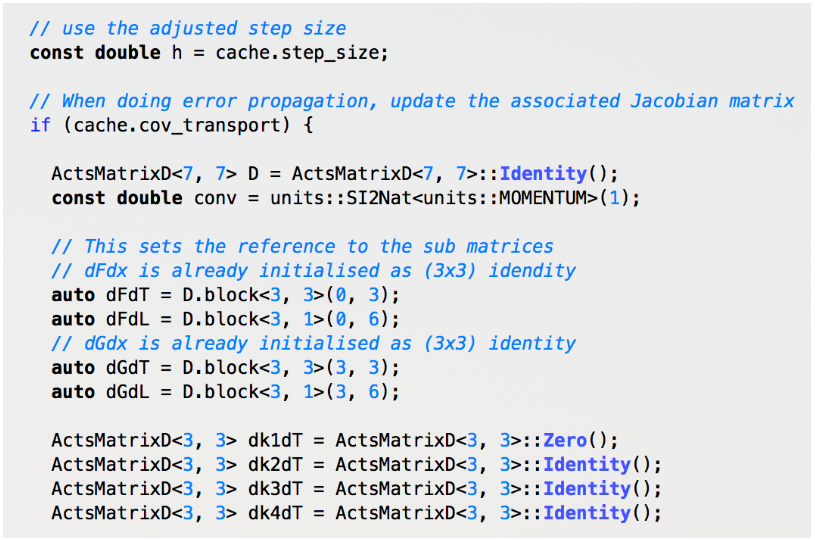


```
if (Jac) {
  // Jacobian calculation
  //
  double* d2A  = &cache.pVector[24];
  double* d3A  = &cache.pVector[31];
  double* d4A  = &cache.pVector[38];
  double  d2A0 = H0[2] * d2A[1] - H0[1] * d2A[2];
  double  d2B0 = H0[0] * d2A[2] - H0[2] * d2A[0];
  double  d2C0 = H0[1] * d2A[0] - H0[0] * d2A[1];
  double  d3A0 = H0[2] * d3A[1] - H0[1] * d3A[2];
  double  d3B0 = H0[0] * d3A[2] - H0[2] * d3A[0];
  double  d3C0 = H0[1] * d3A[0] - H0[0] * d3A[1];
  double  d4A0 = (A0 + H0[2] * d4A[1]) - H0[1] * d4A[2];
  double  d4B0 = (B0 + H0[0] * d4A[2]) - H0[2] * d4A[0];
  double  d4C0 = (C0 + H0[1] * d4A[0]) - H0[0] * d4A[1];
  double  d2A2 = d2A0 + d2A[0];
  double  d2B2 = d2B0 + d2A[1];
  double  d2C2 = d2C0 + d2A[2];
  double  d3A2 = d3A0 + d3A[0];
  double  d3B2 = d3B0 + d3A[1];
  double  d3C2 = d3C0 + d3A[2];
  double  d4A2 = d4A0 + d4A[0];
  double  d4B2 = d4B0 + d4A[1];
  double  d4C2 = d4C0 + d4A[2];
  double  d0   = d4A[0] - A00;
  double  d1   = d4A[1] - A11;
  double  d2   = d4A[2] - A22;
  double  d2A3 = (d2A[0] + d2B2 * H1[2]) - d2C2 * H1[1];
  double  d2B3 = (d2A[1] + d2C2 * H1[0]) - d2A2 * H1[2];
```

+ >1k more lines

VS

ACTS Eigen Stepper implementation



```
// use the adjusted step size
const double h = cache.step_size;

// When doing error propagation, update the associated Jacobian matrix
if (cache.cov_transport) {

  ActsMatrixD<7, 7> D = ActsMatrixD<7, 7>::Identity();
  const double conv = units::SI2Nat<units::MOMENTUM>(1);

  // This sets the reference to the sub matrices
  // dFdx is already initialised as (3x3) idendity
  auto dFdT = D.block<3, 3>(0, 3);
  auto dFdL = D.block<3, 1>(0, 6);
  // dGdx is already initialised as (3x3) identity
  auto dGdT = D.block<3, 3>(3, 3);
  auto dGdL = D.block<3, 1>(3, 6);

  ActsMatrixD<3, 3> dk1dT = ActsMatrixD<3, 3>::Zero();
  ActsMatrixD<3, 3> dk2dT = ActsMatrixD<3, 3>::Identity();
  ActsMatrixD<3, 3> dk3dT = ActsMatrixD<3, 3>::Identity();
  ActsMatrixD<3, 3> dk4dT = ActsMatrixD<3, 3>::Identity();
```

Testing strategy

Unit tests for Core tools

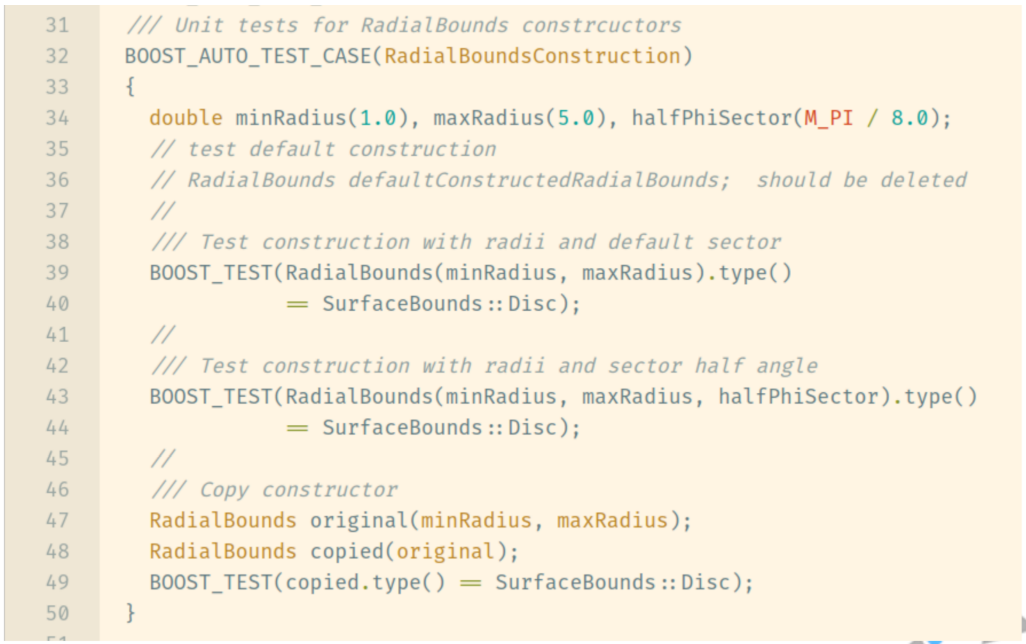
- Interfaces & invariants

Larger core integration tests

- Tool combinations

Full examples

Example Unit Test



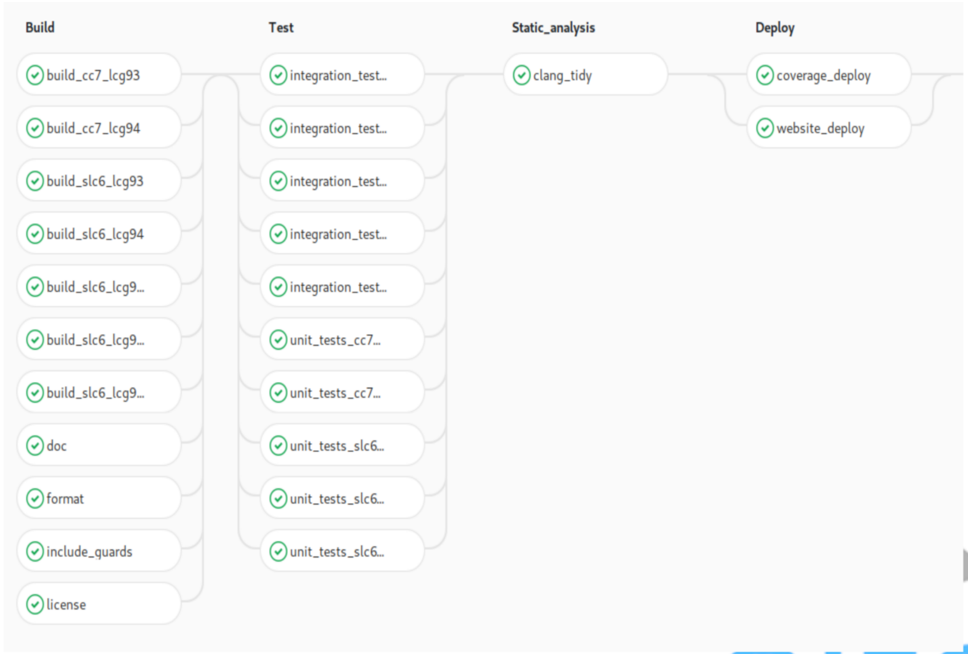
```cpp
/// Unit tests for RadialBounds constrcuctors
BOOST_AUTO_TEST_CASE(RadialBoundsConstruction)
{
  double minRadius(1.0), maxRadius(5.0), halfPhiSector(M_PI / 8.0);
  // test default construction
  // RadialBounds defaultConstructedRadialBounds;  should be deleted
  //
  /// Test construction with radii and default sector
  BOOST_TEST(RadialBounds(minRadius, maxRadius).type()
             == SurfaceBounds::Disc);
  //
  /// Test construction with radii and sector half angle
  BOOST_TEST(RadialBounds(minRadius, maxRadius, halfPhiSector).type()
             == SurfaceBounds::Disc);
  //
  /// Copy constructor
  RadialBounds original(minRadius, maxRadius);
  RadialBounds copied(original);
  BOOST_TEST(copied.type() == SurfaceBounds::Disc);
}
```

Continuous integration

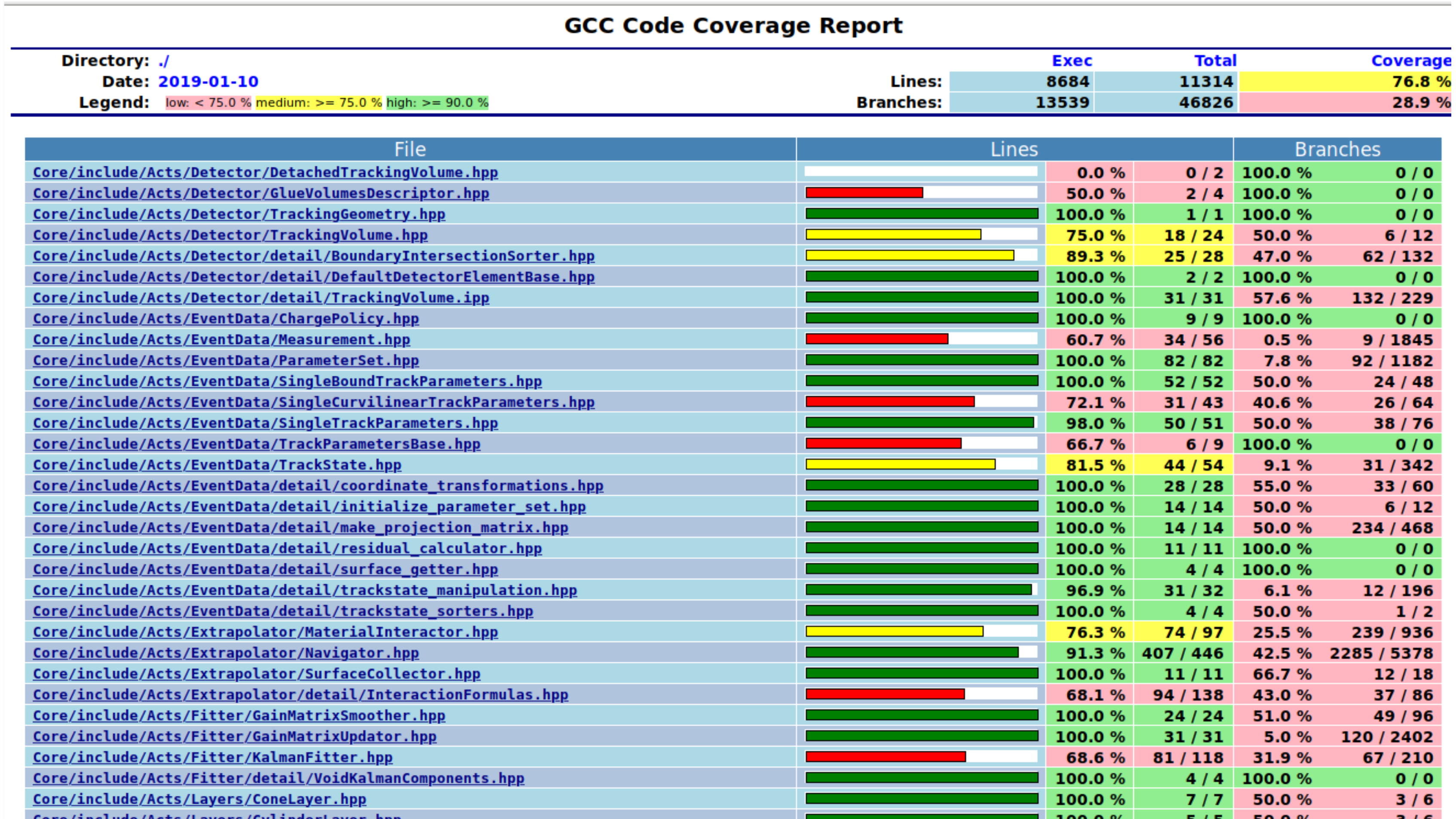
Run tests for every change (pull request)

- LCG93, LCG94
- SLC6, CentOS7
- GCC, clang

Additional (static) analyses



Continous integration / coverage



GCC Code Coverage Report

			Exec	Total	Coverage
Directory:	./				
Date:	2019-01-10	**Lines:**	8684	11314	76.8 %
Legend:	low: < 75.0 % medium: >= 75.0 % high: >= 90.0 %	**Branches:**	13539	46826	28.9 %

File	Lines			Branches	
Core/include/Acts/Detector/DetachedTrackingVolume.hpp	⬜	0.0 %	0 / 2	100.0 %	0 / 0
Core/include/Acts/Detector/GlueVolumesDescriptor.hpp	🟥	50.0 %	2 / 4	100.0 %	0 / 0
Core/include/Acts/Detector/TrackingGeometry.hpp	🟩	100.0 %	1 / 1	100.0 %	0 / 0
Core/include/Acts/Detector/TrackingVolume.hpp	🟨	75.0 %	18 / 24	50.0 %	6 / 12
Core/include/Acts/Detector/detail/BoundaryIntersectionSorter.hpp	🟨	89.3 %	25 / 28	47.0 %	62 / 132
Core/include/Acts/Detector/detail/DefaultDetectorElementBase.hpp	🟩	100.0 %	2 / 2	100.0 %	0 / 0
Core/include/Acts/Detector/detail/TrackingVolume.ipp	🟩	100.0 %	31 / 31	57.6 %	132 / 229
Core/include/Acts/EventData/ChargePolicy.hpp	🟩	100.0 %	9 / 9	100.0 %	0 / 0
Core/include/Acts/EventData/Measurement.hpp	🟥	60.7 %	34 / 56	0.5 %	9 / 1845
Core/include/Acts/EventData/ParameterSet.hpp	🟩	100.0 %	82 / 82	7.8 %	92 / 1182
Core/include/Acts/EventData/SingleBoundTrackParameters.hpp	🟩	100.0 %	52 / 52	50.0 %	24 / 48
Core/include/Acts/EventData/SingleCurvilinearTrackParameters.hpp	🟥	72.1 %	31 / 43	40.6 %	26 / 64
Core/include/Acts/EventData/SingleTrackParameters.hpp	🟩	98.0 %	50 / 51	50.0 %	38 / 76
Core/include/Acts/EventData/TrackParametersBase.hpp	🟥	66.7 %	6 / 9	100.0 %	0 / 0
Core/include/Acts/EventData/TrackState.hpp	🟨	81.5 %	44 / 54	9.1 %	31 / 342
Core/include/Acts/EventData/detail/coordinate_transformations.hpp	🟩	100.0 %	28 / 28	55.0 %	33 / 60
Core/include/Acts/EventData/detail/initialize_parameter_set.hpp	🟩	100.0 %	14 / 14	50.0 %	6 / 12
Core/include/Acts/EventData/detail/make_projection_matrix.hpp	🟩	100.0 %	14 / 14	50.0 %	234 / 468
Core/include/Acts/EventData/detail/residual_calculator.hpp	🟩	100.0 %	11 / 11	100.0 %	0 / 0
Core/include/Acts/EventData/detail/surface_getter.hpp	🟩	100.0 %	4 / 4	100.0 %	0 / 0
Core/include/Acts/EventData/detail/trackstate_manipulation.hpp	🟩	96.9 %	31 / 32	6.1 %	12 / 196
Core/include/Acts/EventData/detail/trackstate_sorters.hpp	🟩	100.0 %	4 / 4	50.0 %	1 / 2
Core/include/Acts/Extrapolator/MaterialInteractor.hpp	🟨	76.3 %	74 / 97	25.5 %	239 / 936
Core/include/Acts/Extrapolator/Navigator.hpp	🟩	91.3 %	407 / 446	42.5 %	2285 / 5378
Core/include/Acts/Extrapolator/SurfaceCollector.hpp	🟩	100.0 %	11 / 11	66.7 %	12 / 18
Core/include/Acts/Extrapolator/detail/InteractionFormulas.hpp	🟥	68.1 %	94 / 138	43.0 %	37 / 86
Core/include/Acts/Fitter/GainMatrixSmoother.hpp	🟩	100.0 %	24 / 24	51.0 %	49 / 96
Core/include/Acts/Fitter/GainMatrixUpdator.hpp	🟩	100.0 %	31 / 31	5.0 %	120 / 2402
Core/include/Acts/Fitter/KalmanFitter.hpp	🟥	68.6 %	81 / 118	31.9 %	67 / 210
Core/include/Acts/Fitter/detail/VoidKalmanComponents.hpp	🟩	100.0 %	4 / 4	100.0 %	0 / 0
Core/include/Acts/Layers/ConeLayer.hpp	🟩	100.0 %	7 / 7	50.0 %	3 / 6
Core/include/Acts/Layers/CylinderLayer.hpp	🟩	100.0 %	5 / 5	50.0 %	3 / 6

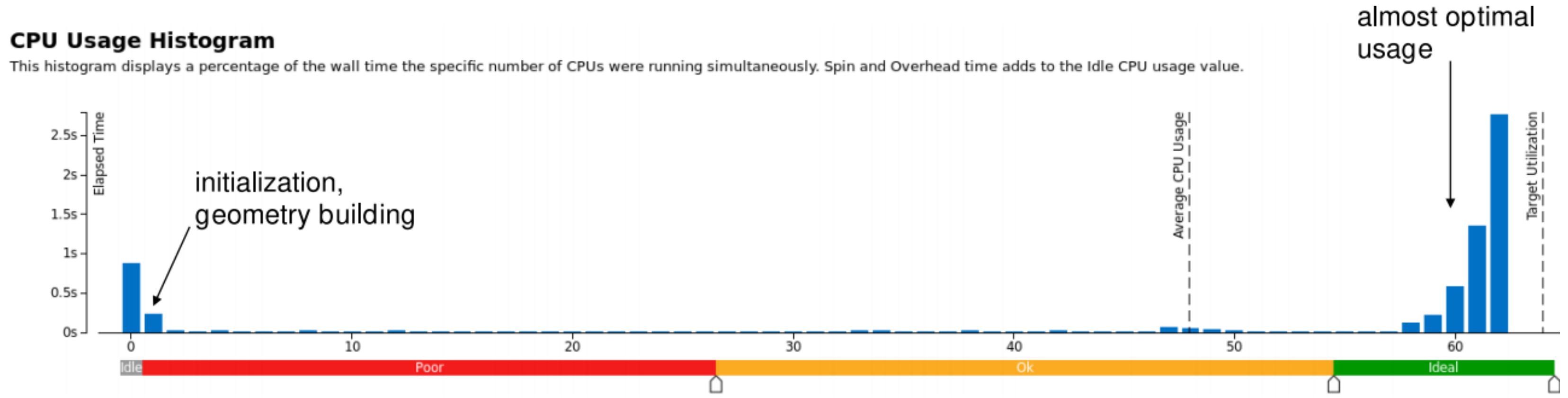
Concurrency tests

H. Grasland

Run full example chain parallelized

- Check possible bottlenecks
- Check access violations bottlenecks
- Check serial/parallized consistency

Intel Xeon e5-2698
32 cores
64 threads



Project organization

Hosted on CERN Gitlab

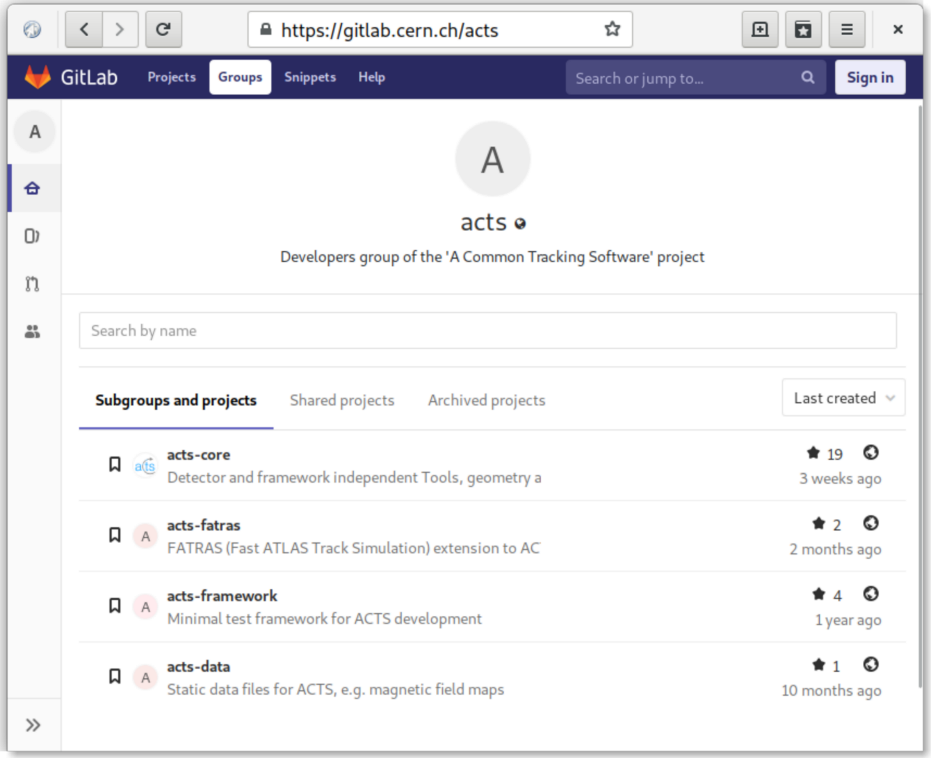
Core library

- This is what should be used by an experiment

Fast simulation tools

Minimal event framework

Data files for tests



acts

Developers group of the 'A Common Tracking Software' project

acts-core	Detector and framework independent Tools, geometry a	19 · 3 weeks ago
acts-fatras	FATRAS (Fast ATLAS Track Simulation) extension to AC	2 · 2 months ago
acts-framework	Minimal test framework for ACTS development	4 · 1 year ago
acts-data	Static data files for ACTS, e.g. magnetic field maps	1 · 10 months ago

https://gitlab.cern.ch/acts

Timeline

Q1 Q2 Q3 Q4

2016

Jun 2016 - 0.01.00

prototype version,
f rst geometry functionality

Sep 2016 - 0.02.00

prototype Kalman fl ter,
ROOT geometry plugin

Nov 2016 - 0.03.00

message service,
magnetic f eld, CI,
Gaudi wrapping

2017

Apr 2017 - 0.04.00

const-correct release
multithreaded tests

Aug 2017 - 0.05.00

work horse release for
f tter development
new propagation

Q3/4 2017 - 0.05.0X

bugf x releases
- **FCC testbeds**
- **Tracking ML**

2018

Apr 2018 - 0.06.00

ATLAS InnerDetector
- **geometry building**
- **extrapolation ID**

Sep 2018 - 0.07.00

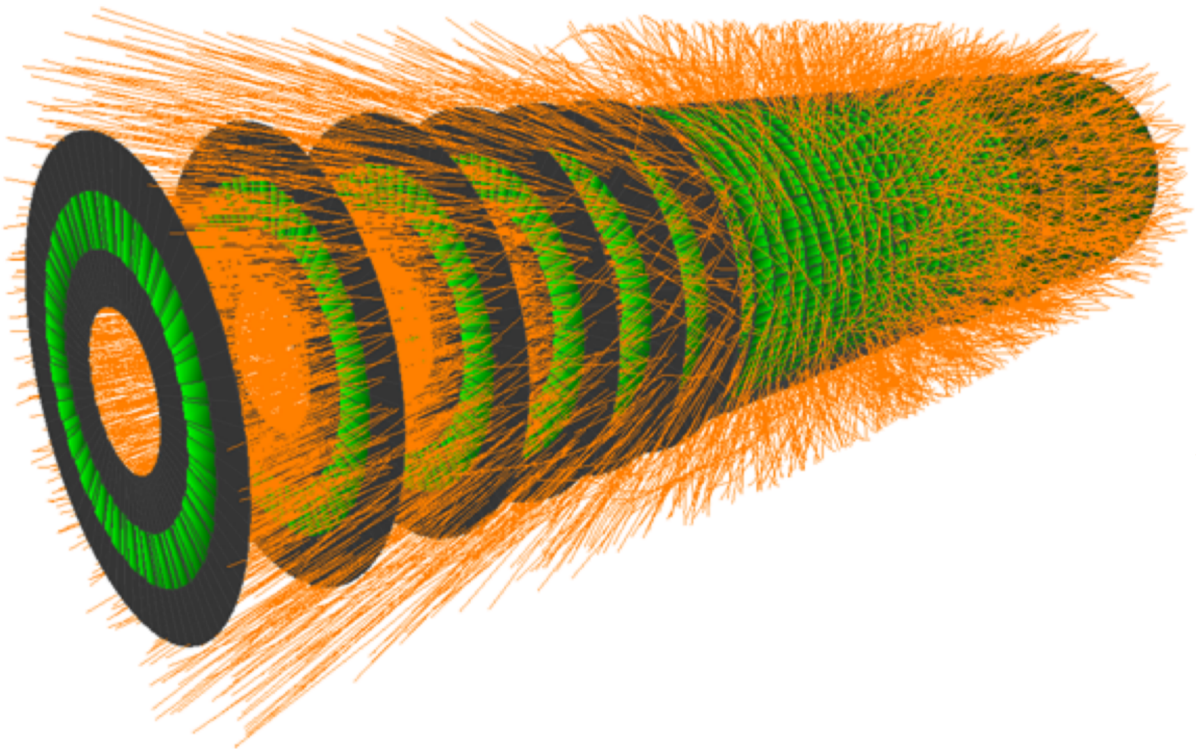
New Propagation
- **KF track f tting**
- **ATLAS seeding**

Oct 2018 - 0.08.00
- **track seeding**
- **track f nding**
- **calorimeter extrapolation support**

RC 01.00.00

Tracking feature (in-)complete
track seeding
track f nding
track f tting

Application: TrackML Challenged



Native ACTS Geometry

Pixels & strips in barrels & disks

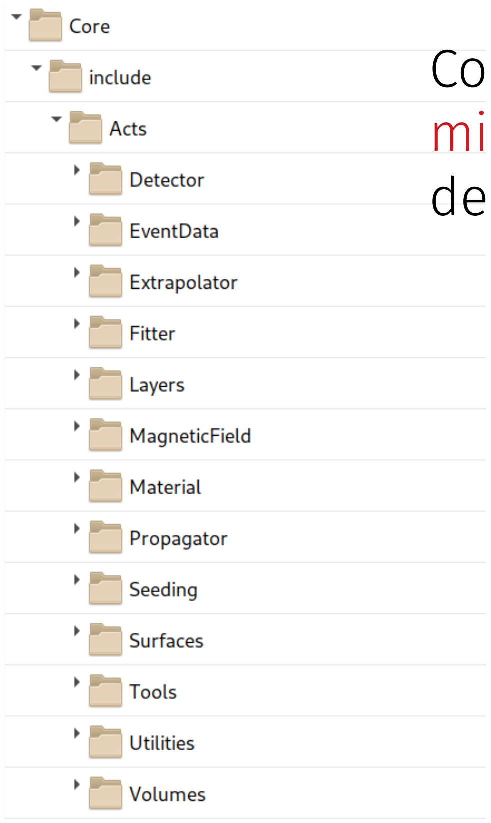
ATLAS magnetic field

Fast simulation w/ multiple scattering, nuclear interactions

https://sites.google.com/site/trackmlparticle/

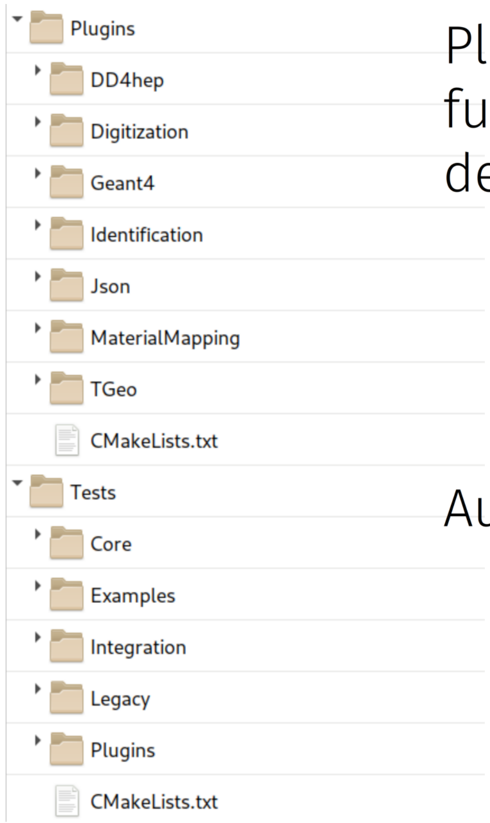
ACTS components

Structure on disk



- Core
  - include
    - Acts
      - Detector
      - EventData
      - Extrapolator
      - Fitter
      - Layers
      - MagneticField
      - Material
      - Propagator
      - Seeding
      - Surfaces
      - Tools
      - Utilities
      - Volumes

Core w/ minimal dependencies

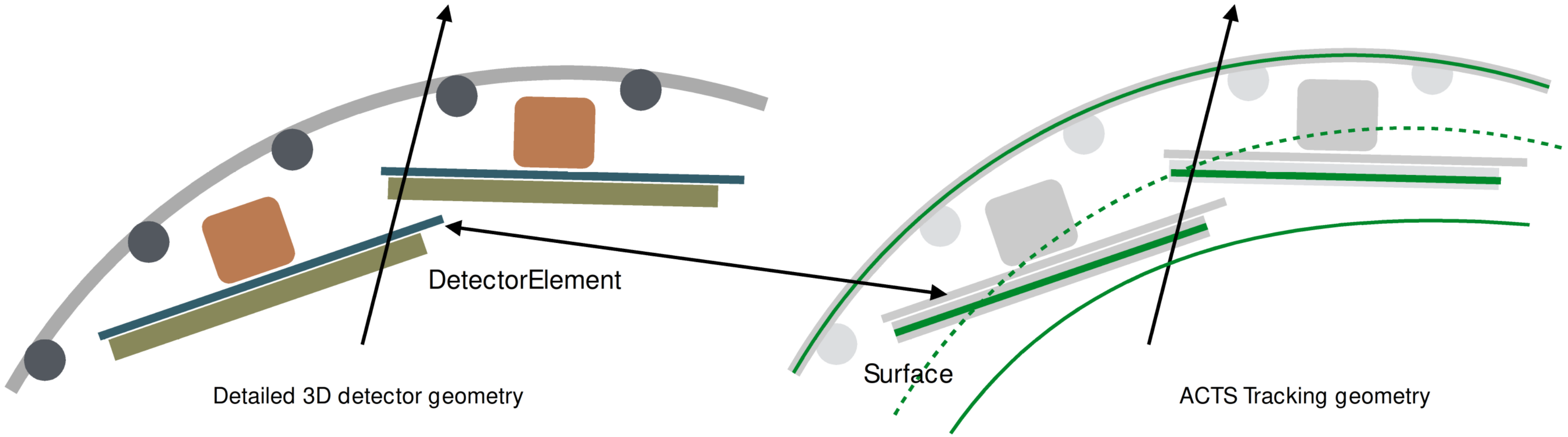


- Plugins
  - DD4hep
  - Digitization
  - Geant4
  - Identification
  - Json
  - MaterialMapping
  - TGeo
  - CMakeLists.txt
- Tests
  - Core
  - Examples
  - Integration
  - Legacy
  - Plugins
  - CMakeLists.txt

Plugins for optional functional w/ extra dependencies

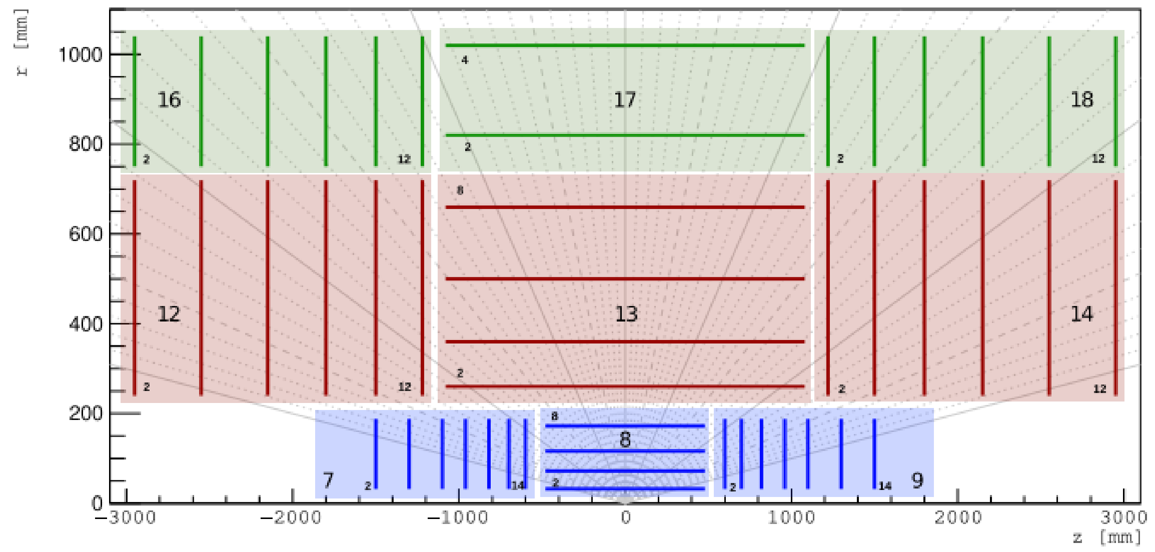
Automated tests

(Tracking) Geometry



Simplified geometry based on tracking surfaces

Geometry Layout



Generic example detector (TrackML)

Organized by

- Volume
- Layer
- Module

for navigation

(Minimal) event data model

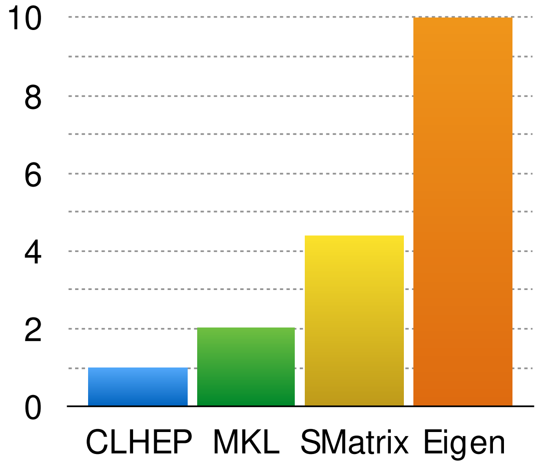
Tracks

- Acts::TrackParameter
- Acts::SingleBoundParameter
- Acts::SingleCurvilinearParameter

Measurements

- Acts::Measurement
- Acts::CalibratedMeasurement

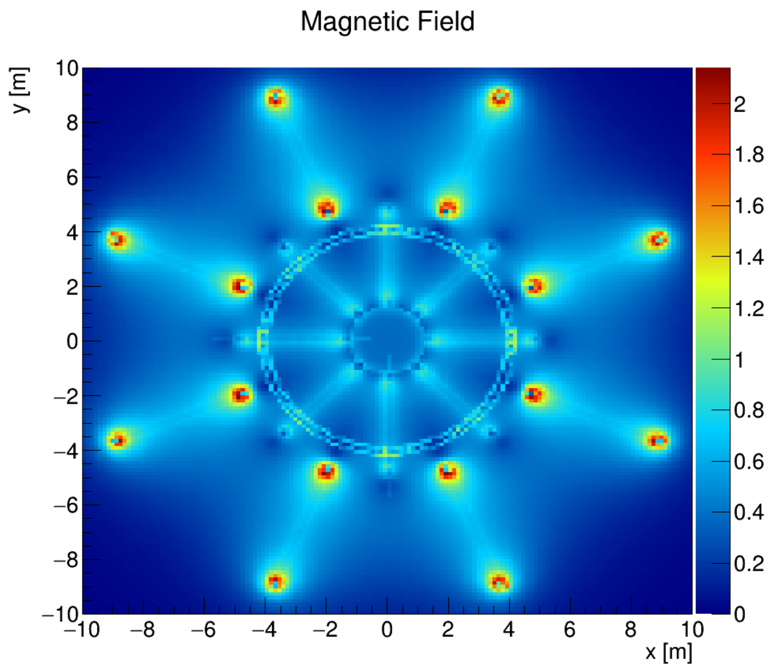
Based on Eigen library

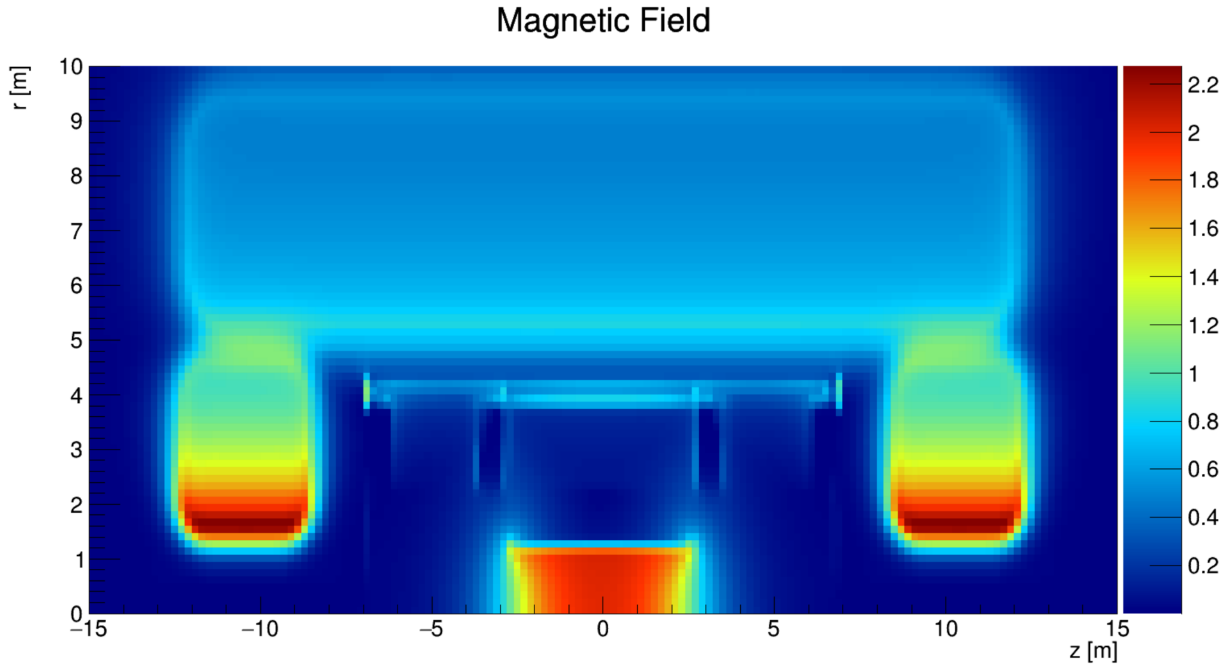


ATLAS LS1 performance comparison
Speed 5×5 matrix multiplication

Magnetic fields

Example: ATLAS field





Interpolated field maps
Different examples available

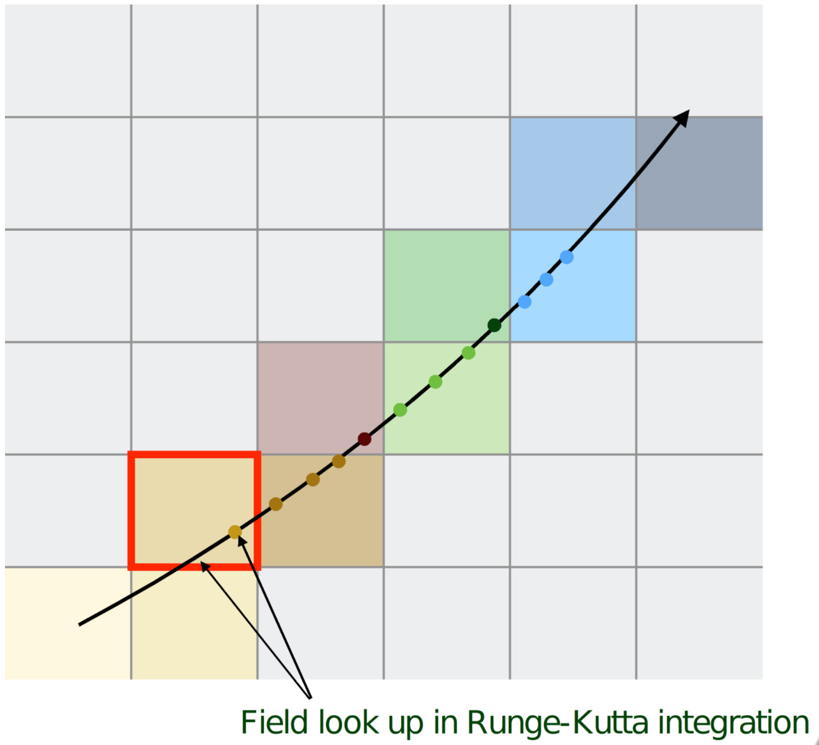
Magnetic field cell caching

Cache local extrapolation cell

Performance improvements

- 20% in simulation
- Few % in reconstruction

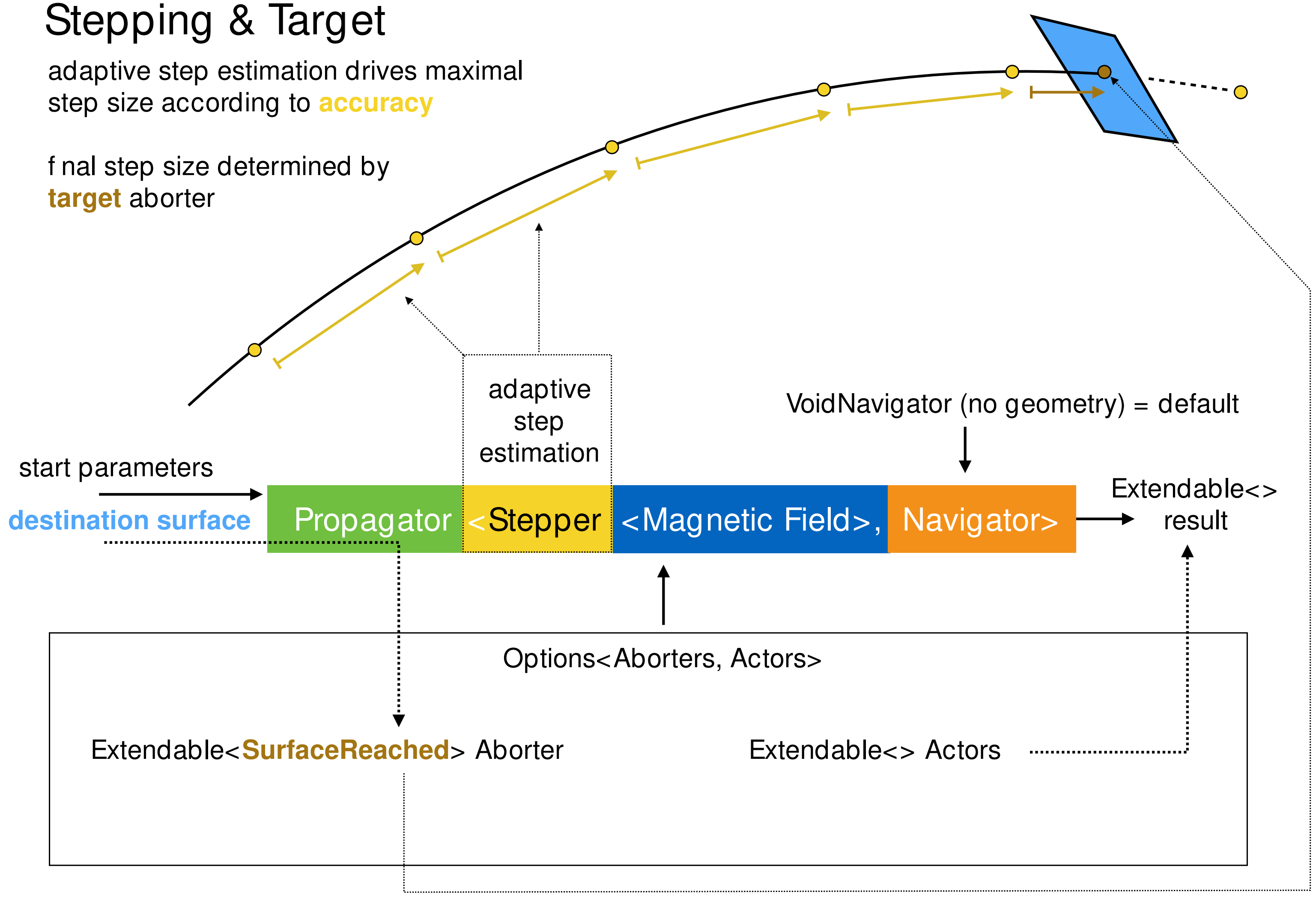
Localizes field access



Stepping & Target

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

f nal step size determined by **target** aborter

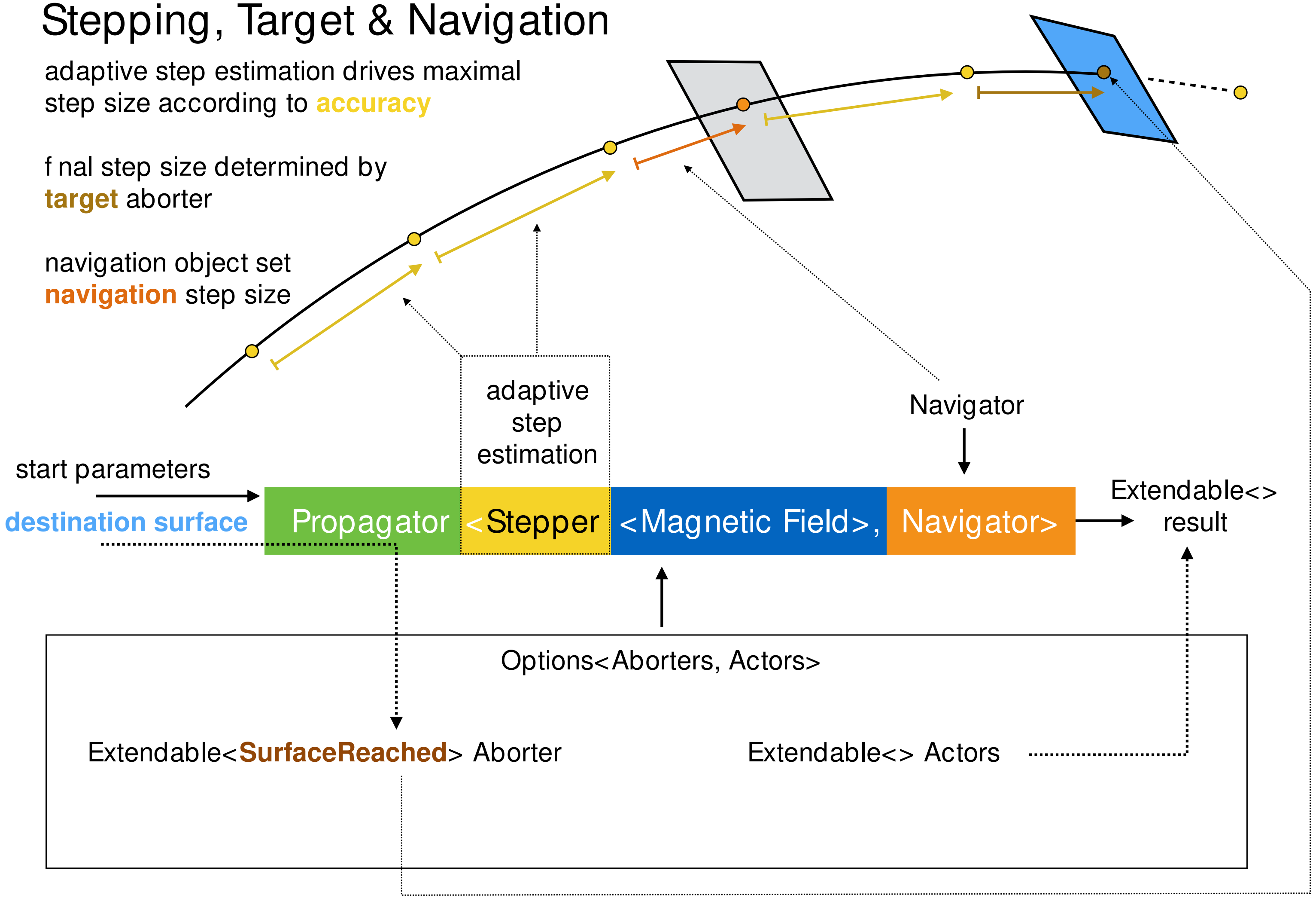

adaptive step estimation
VoidNavigator (no geometry) = default
start parameters
destination surface
Propagator <Stepper <Magnetic Field>, Navigator>
Extendable<> result
Options<Aborters, Actors>
Extendable<SurfaceReached> Aborter
Extendable<> Actors

Stepping, Target & Navigation

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

f nal step size determined by **target** aborter

navigation object set **navigation** step size



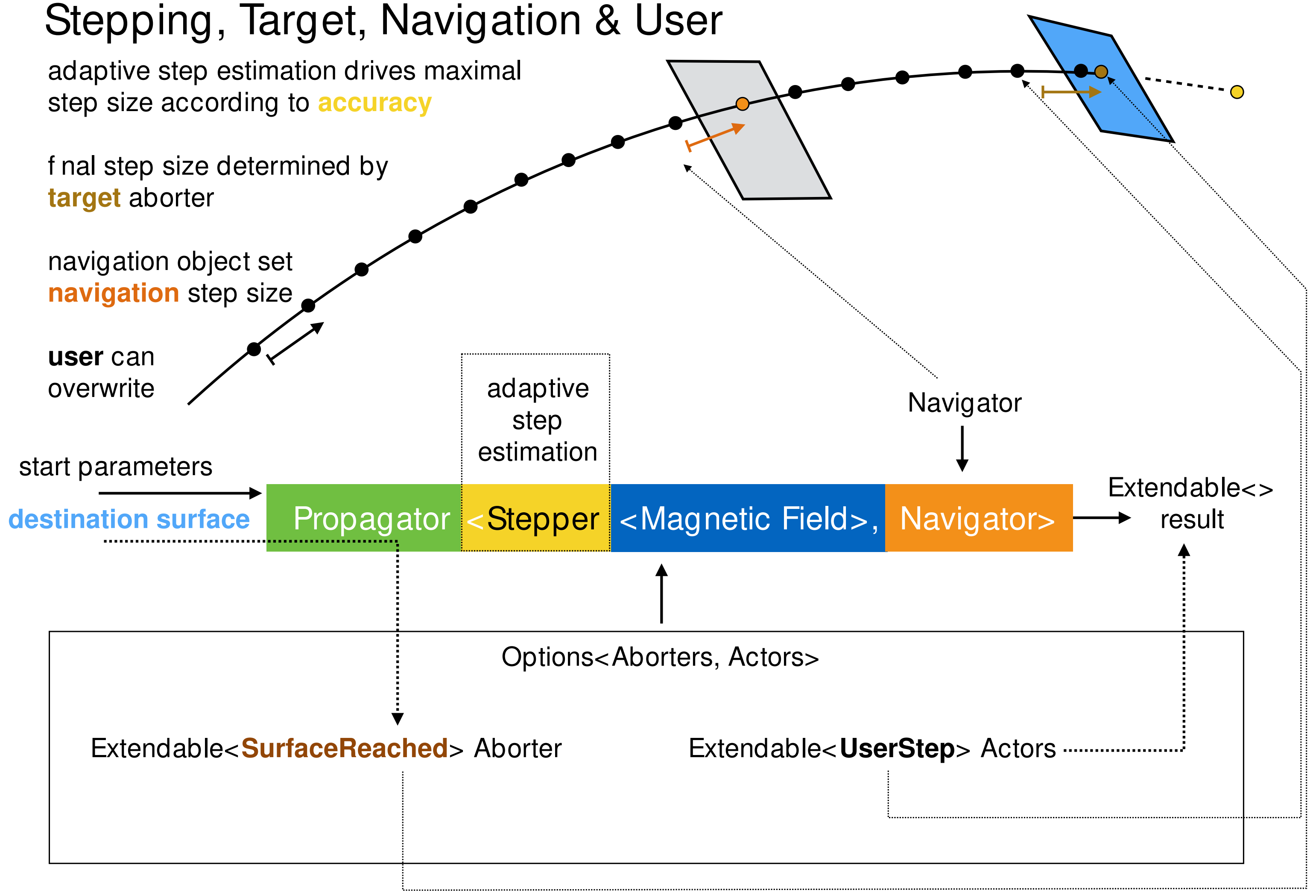
Stepping, Target, Navigation & User

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

f nal step size determined by **target** aborter

navigation object set **navigation** step size

user can overwrite



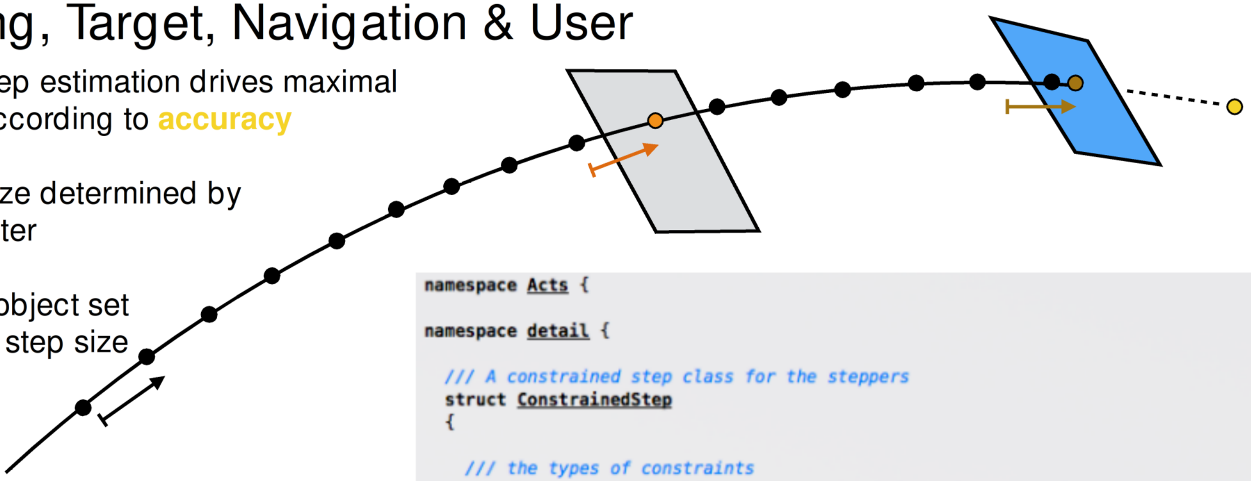
Stepping, Target, Navigation & User

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

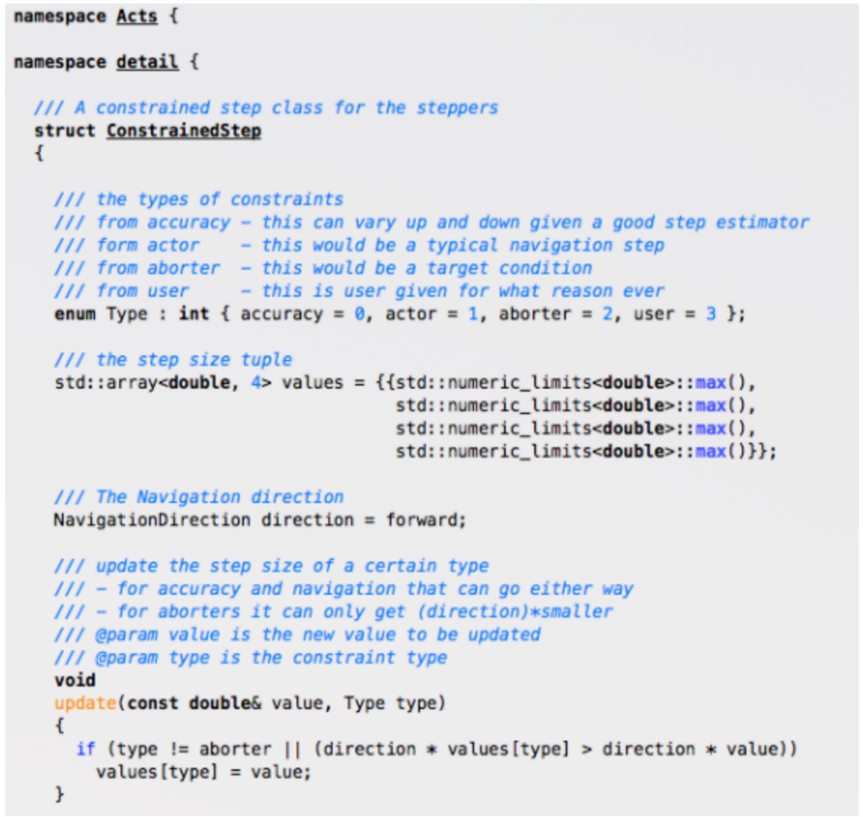
f nal step size determined by **target** aborter

navigation object set **navigation** step size

user can overwrite

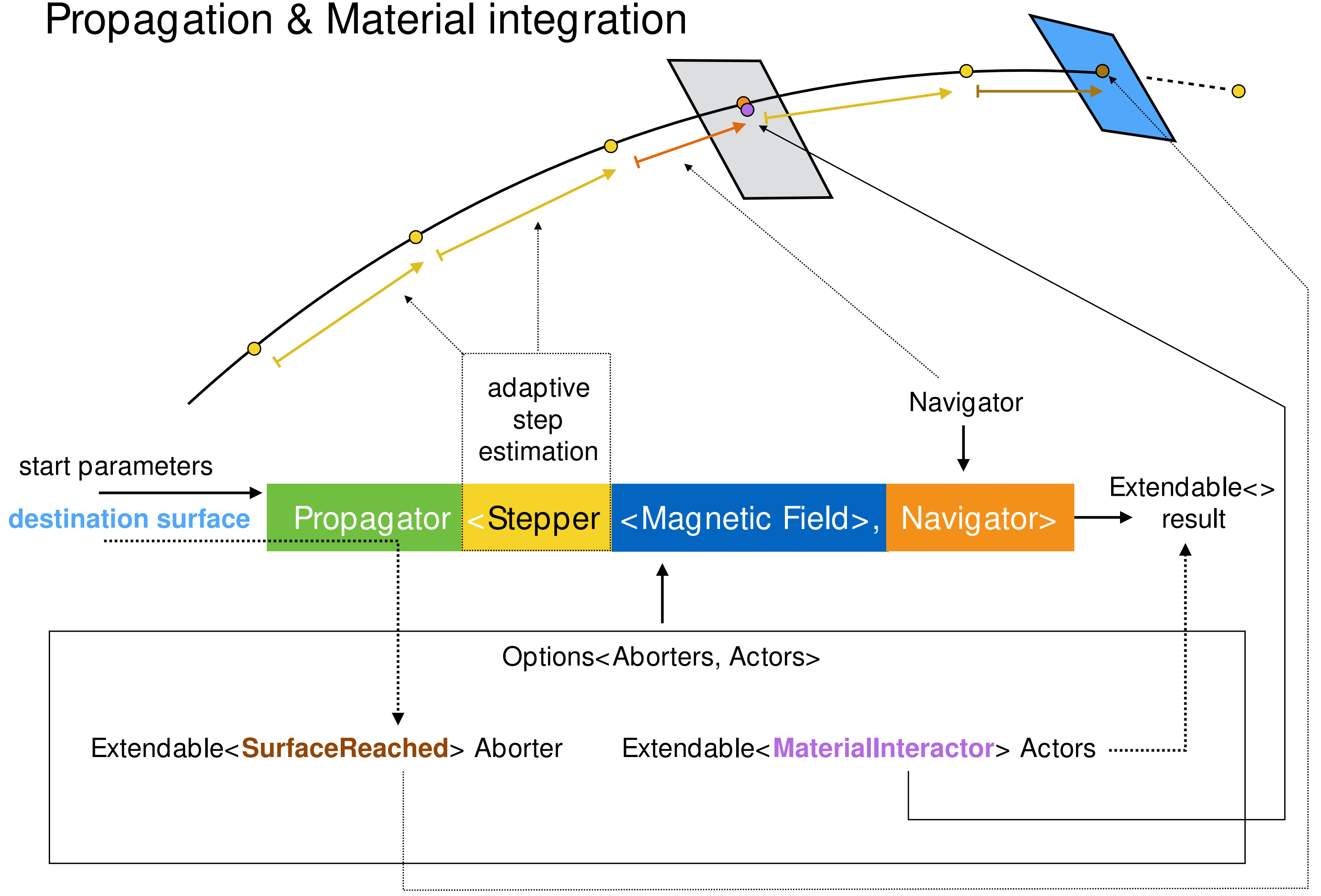


ConstrainedStep

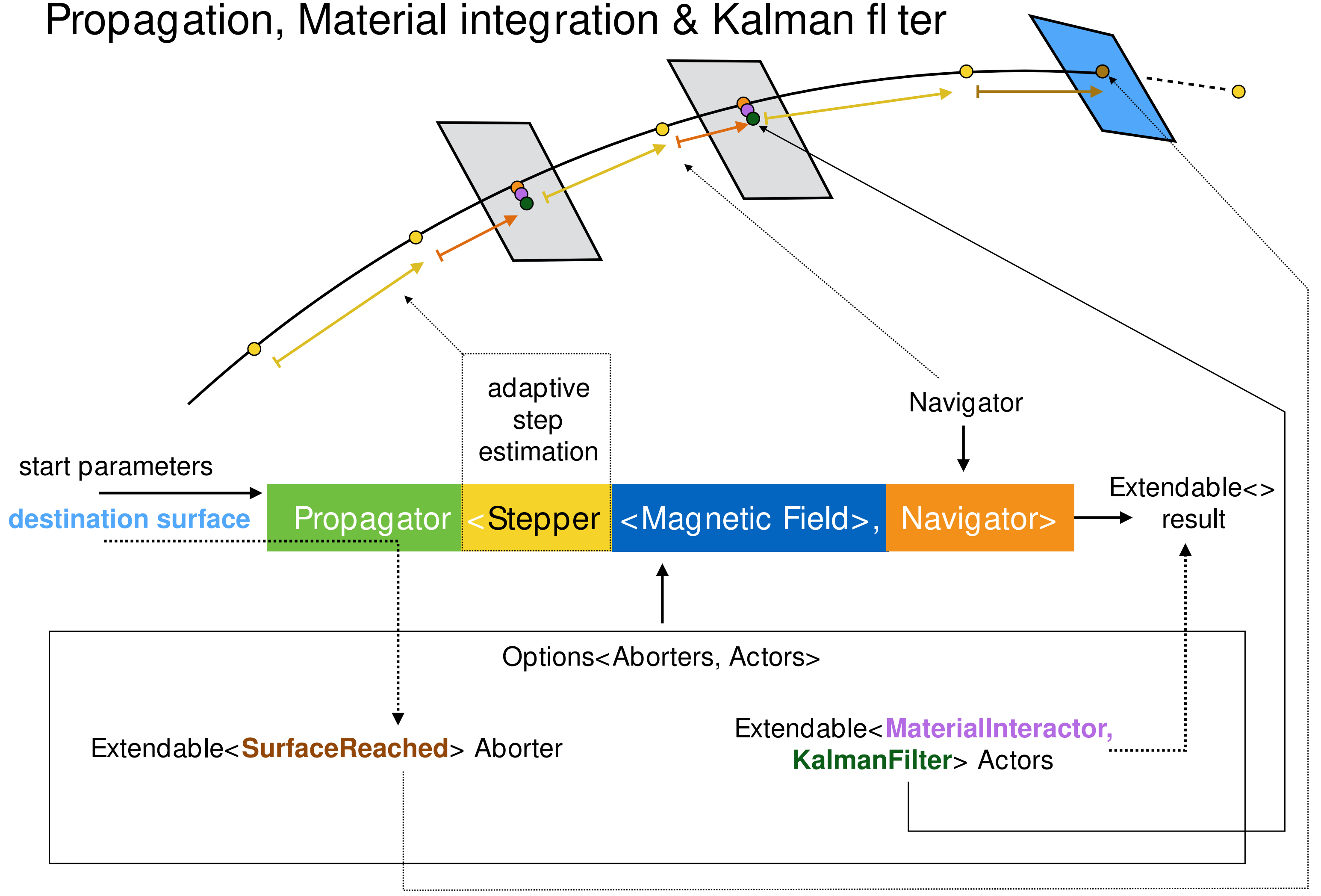


```
namespace Acts {

namespace detail {

  /// A constrained step class for the steppers
  struct ConstrainedStep
  {

    /// the types of constraints
    /// from accuracy - this can vary up and down given a good step estimator
    /// form actor    - this would be a typical navigation step
    /// from aborter  - this would be a target condition
    /// from user     - this is user given for what reason ever
    enum Type : int { accuracy = 0, actor = 1, aborter = 2, user = 3 };

    /// the step size tuple
    std::array<double, 4> values = {{std::numeric_limits<double>::max(),
                                     std::numeric_limits<double>::max(),
                                     std::numeric_limits<double>::max(),
                                     std::numeric_limits<double>::max()}};

    /// The Navigation direction
    NavigationDirection direction = forward;

    /// update the step size of a certain type
    /// - for accuracy and navigation that can go either way
    /// - for aborters it can only get (direction)*smaller
    /// @param value is the new value to be updated
    /// @param type is the constraint type
    void
    update(const double& value, Type type)
    {
      if (type != aborter || (direction * values[type] > direction * value))
        values[type] = value;
    }
```

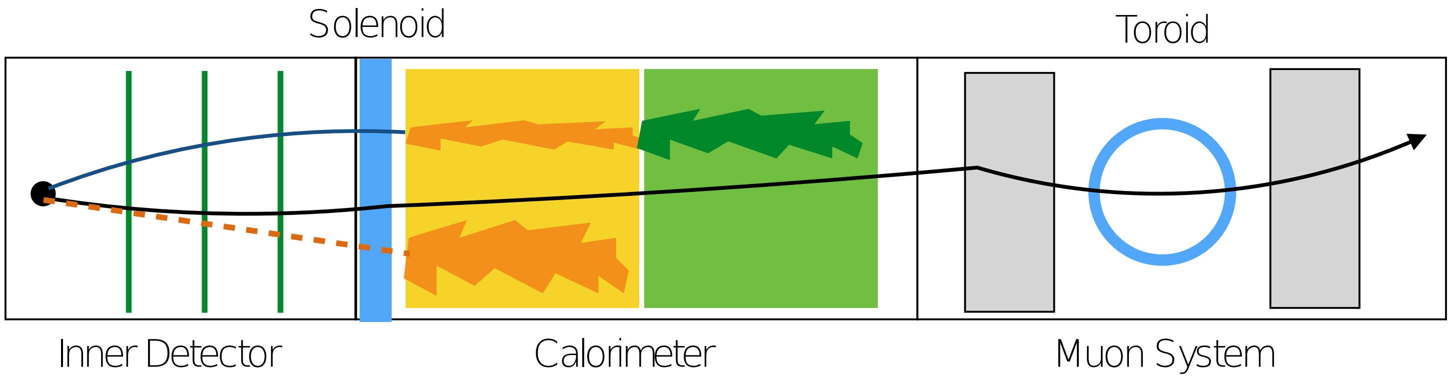
Propagation & Material integration



Propagation, Material integration & Kalman fl ter


adaptive
step
estimation
Navigator
start parameters
destination surface
Propagator <Stepper <Magnetic Field>, Navigator>
Extendable<>
result
Options<Aborters, Actors>
Extendable<SurfaceReached> Aborter
Extendable<MaterialInteractor,
KalmanFilter> Actors

Beyond the (ATLAS) inner tracking detector



Mostly covered

Requires STEP propagator (prototype in 0.8.0)

Requires geometry support

Not yet covered

Open questions

Tools

- Non-silicon detectors
- Track follower
- Global χ^2 track fitter (ATLAS)
- Vertex finder
- Vertex fitter (in-progress)
- …

Design

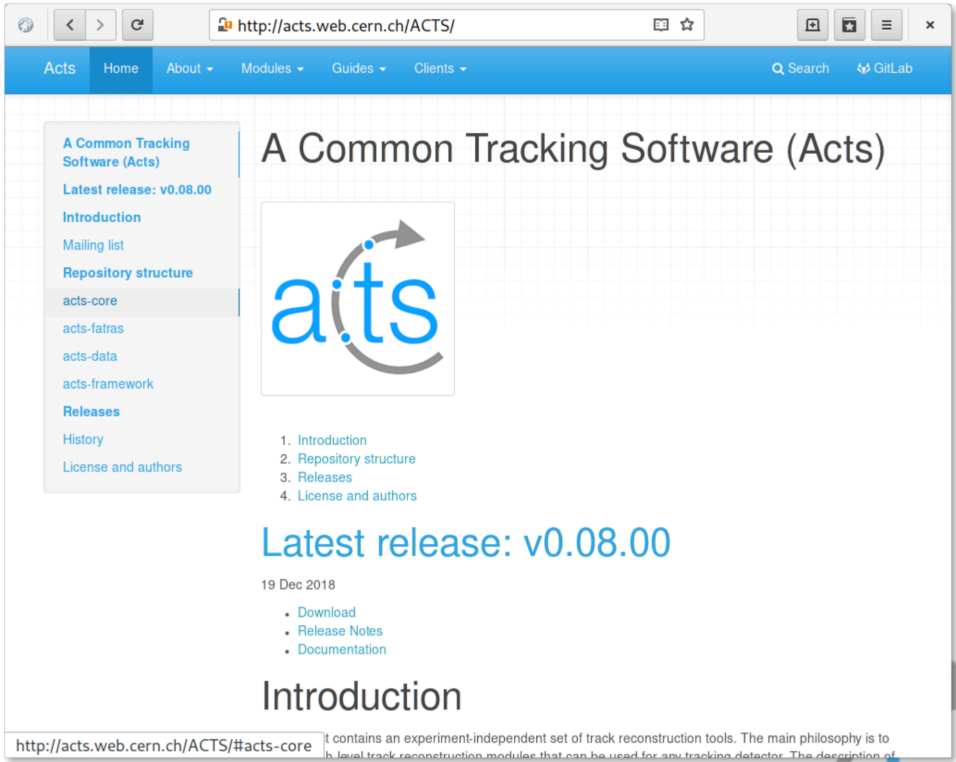
- Vectorizable DEM, AoS vs. SoA
- Parallelizable conditions handling (Paul)
- Detector alignment
- Parallelization within ACTS
- …

Summary

A standalone C++ software library for tracking

- Derived from ATLAS code
- Basic functionality available

Progressing towards full tracking solution


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

A Common Tracking Software (Acts)
Latest release: v0.08.00
Introduction
Mailing list
Repository structure
acts-core
acts-fatras
acts-data
acts-framework
Releases
History
License and authors

A Common Tracking Software (Acts)

1. Introduction
2. Repository structure
3. Releases
4. License and authors

Latest release: v0.08.00

19 Dec 2018

- Download
- Release Notes
- Documentation

Introduction

t contains an experiment-independent set of track reconstruction tools. The main philosophy is to

http://acts.web.cern.ch/ACTS/#acts-core

http://cern.ch/acts