

CHEP 2024 KRAKOW, POLAND



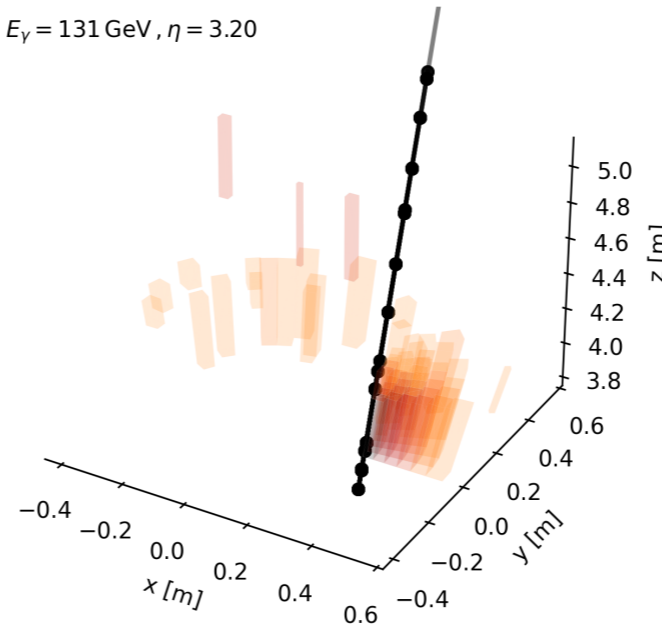
Towards an experiment-independent toolkit for fast calorimeter simulation

JOSHUA F. BEIRER (CERN)

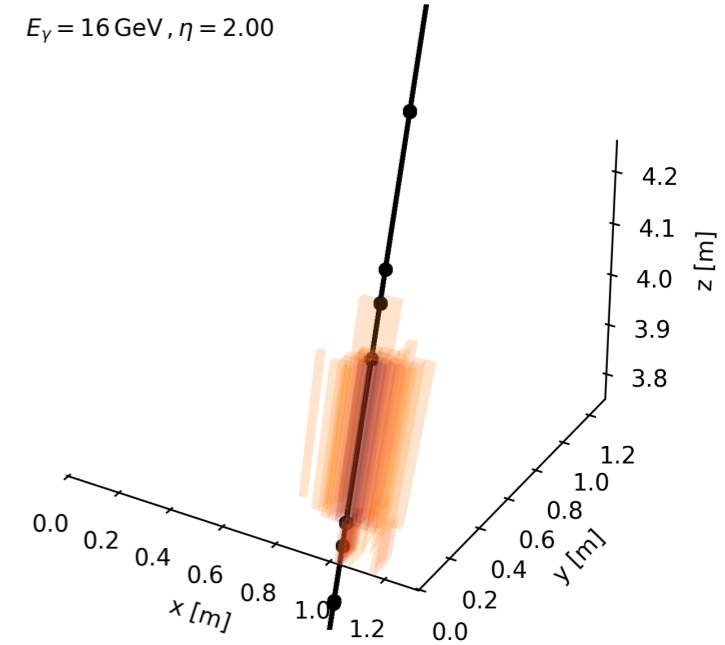
ON BEHALF OF THE ATLAS COLLABORATION



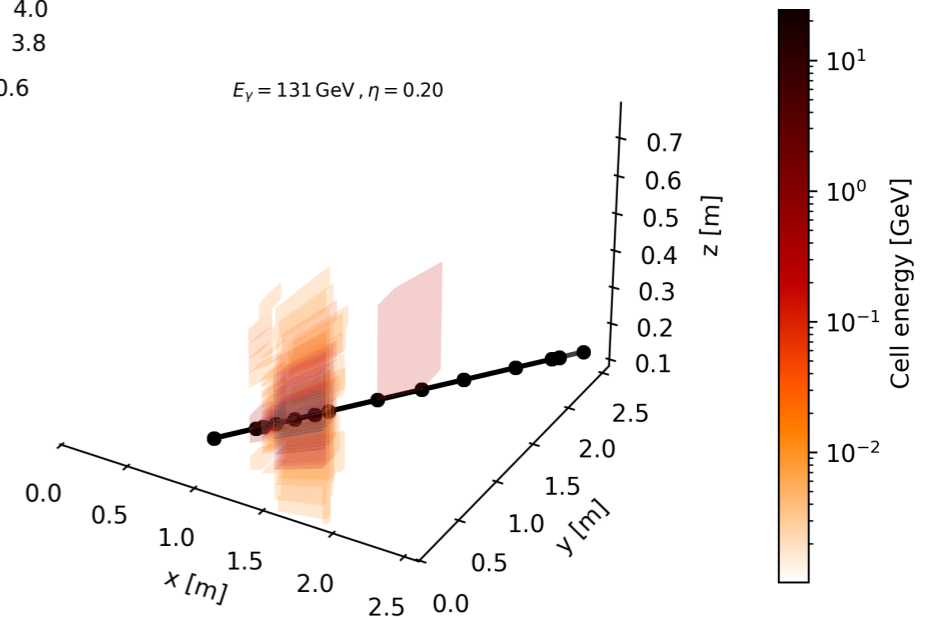
$E_\gamma = 131 \text{ GeV}, \eta = 3.20$

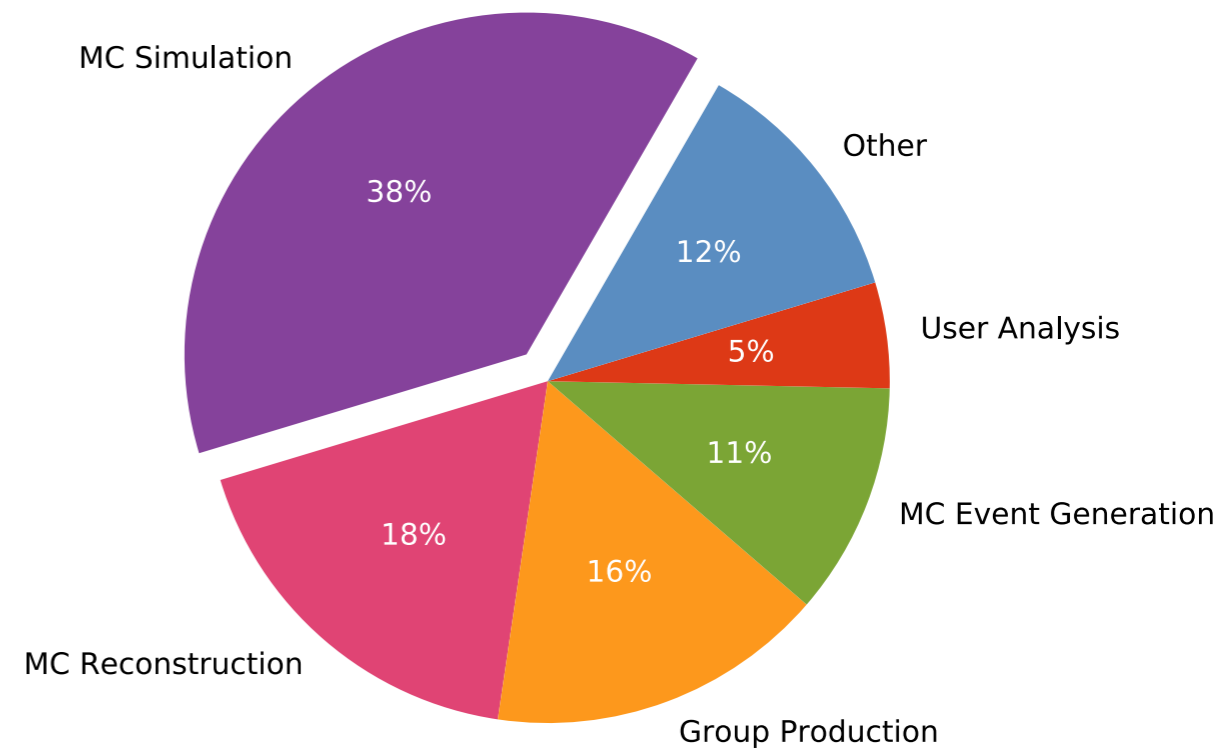
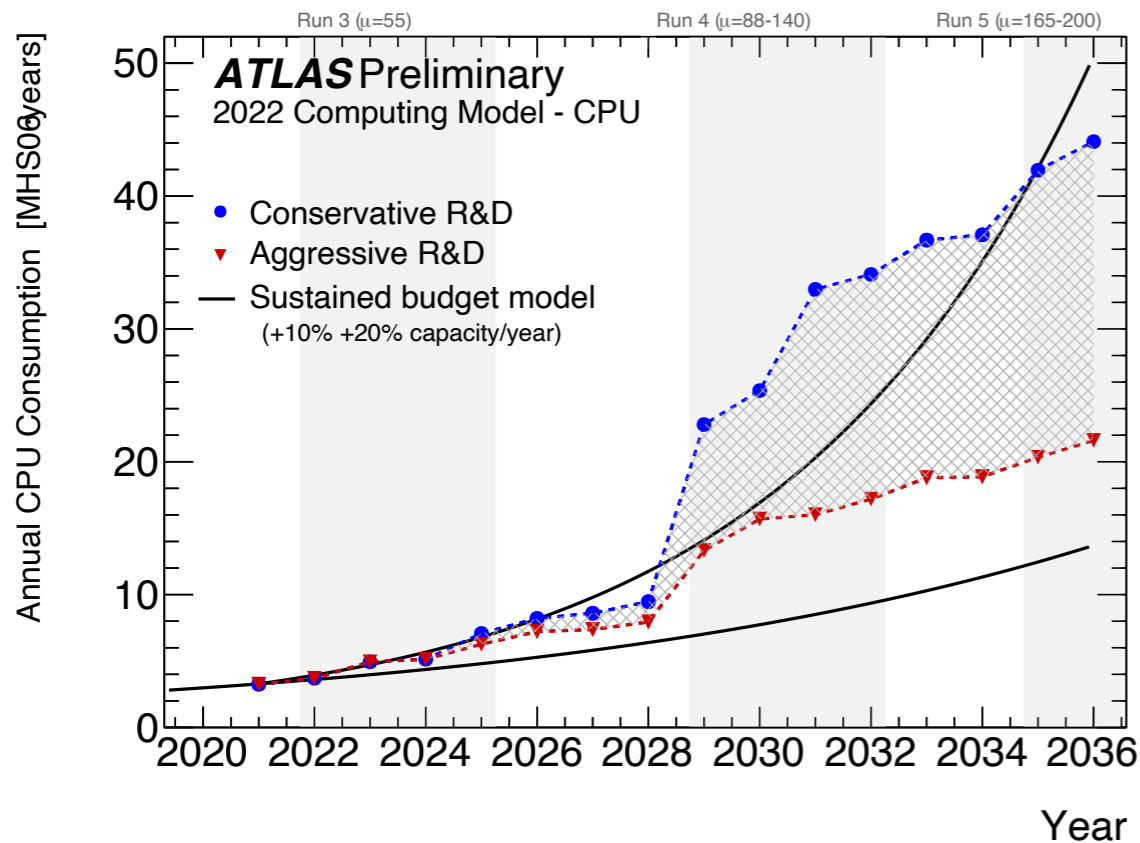


$E_\gamma = 16 \text{ GeV}, \eta = 2.00$



$E_\gamma = 131 \text{ GeV}, \eta = 0.20$





- ATLAS needs to produce billions of MC events, but is limited by CPU constraints
- MC simulation has largest single share of total CPU usage
- About 80 - 90 % of CPU time spent on simulation of showers in calorimeter system

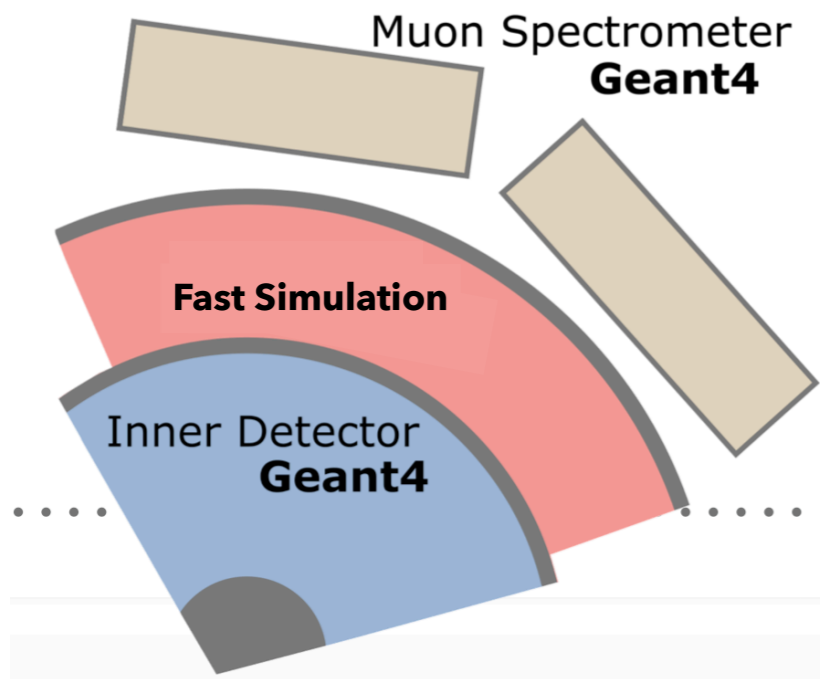
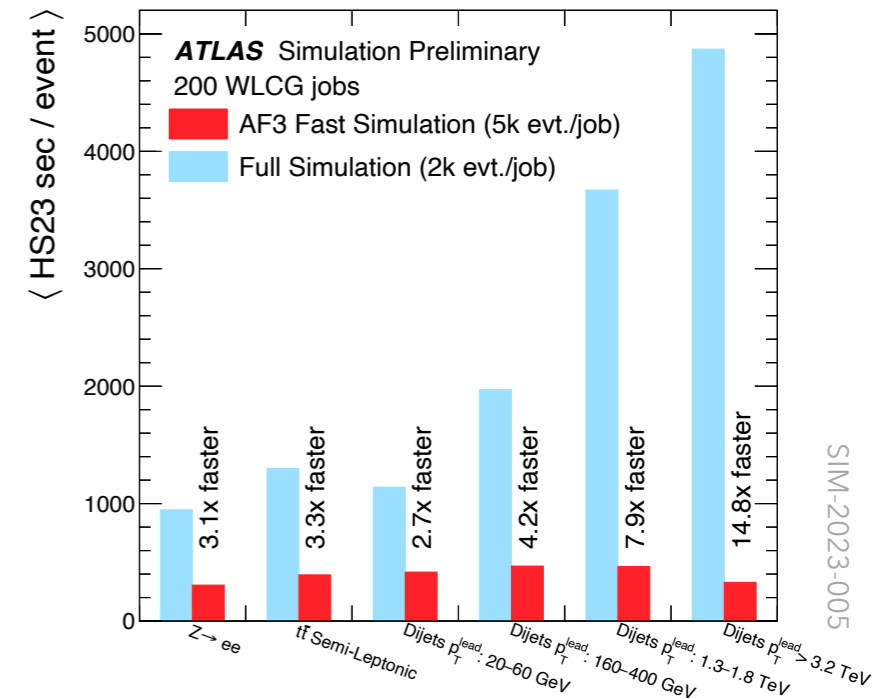


use (fast) parametric models to reproduce Geant4 simulation as accurately as possible

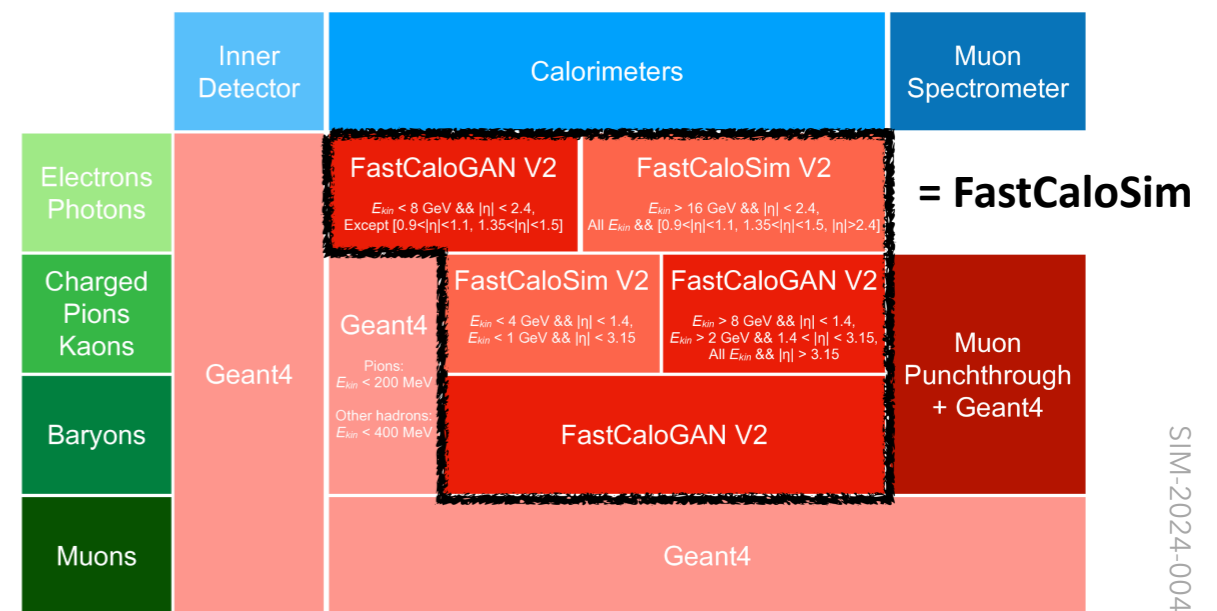
Current State-of-the-Art fast simulation tool in ATLAS · **AtlFast3!**

- **Basic Principle:** instead of tracking each particle in calorimeter showers, parametrise energy response with single particles
- Two distinct approaches of shower generation:
 - **FastCaloSimV2:** histogram-based parametrised modelling
 - **FastCaloGAN:** Generative Adversarial Network
- **3-15x** increase in simulation speed with respect to Geant4
- AF3 offers drastically improved physics performance with respect to Run 2 predecessor

3-15 x increase in simulation speed

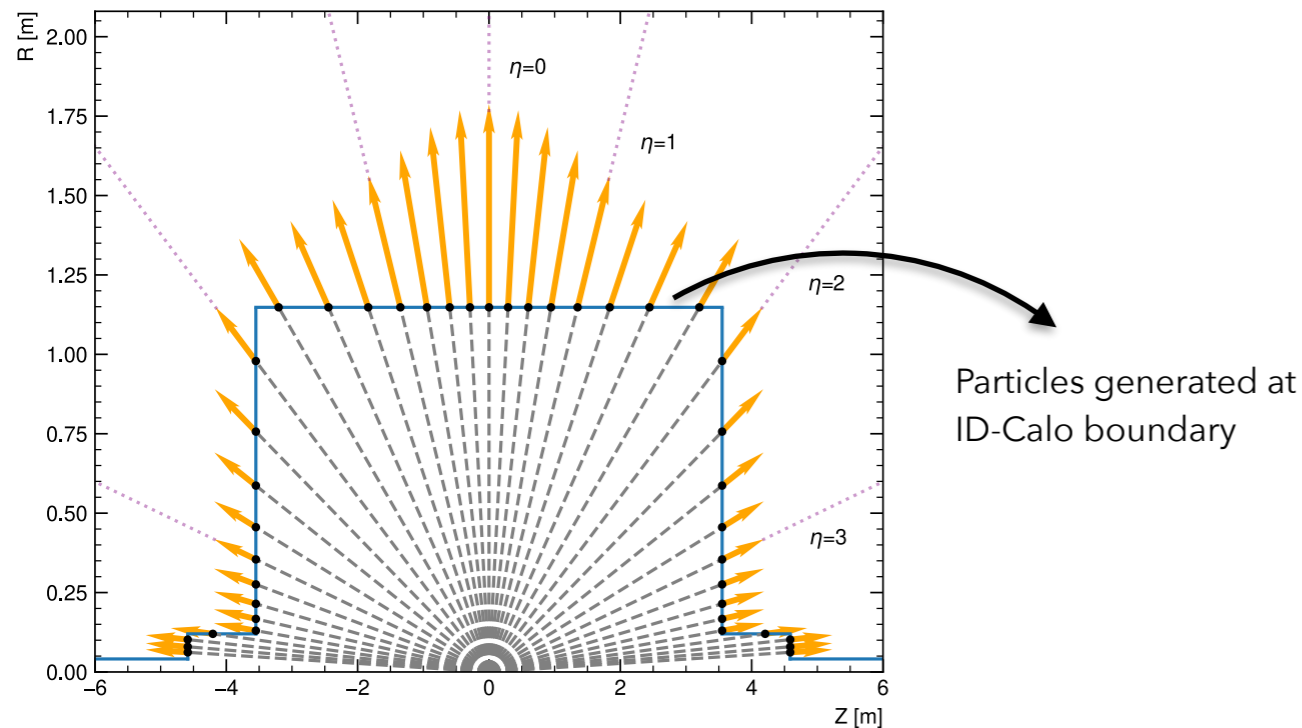
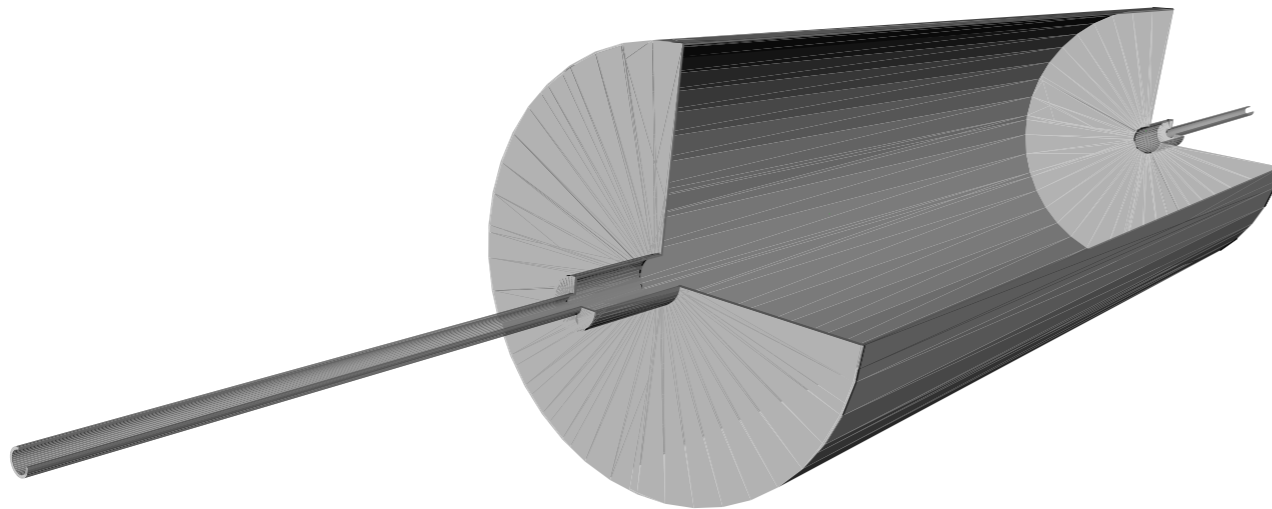


Run 3 Configuration of AtlFast3



Simulation time fully dominated by Geant4 simulation in ID!

Boundary between Inner Detector and Calorimeter System



- Three separate parametrisations
 - **EM showers:**
 1. Photons γ
 2. Electrons / Positrons e^\pm
 - **Hadronic Showers:**
 3. Charged Pions π^\pm (+ dedicated p GANs)
- Particles for parametrisation generated at the boundary between Inner Detector and Calorimeter System
- Parametrisation depending on incident particle energy and direction:
 - 17 log-bins of **truth momentum** from 64MeV to 4TeV
 - 100 bins of $|\eta|$ from 0 to 5.0

More information on AtlFast3 in [Federico's talk](#)

- **FastCaloSim** currently implemented in the Integrated Simulation Framework (ISF) in **athena** to combine with multiple simulators
- ISF is flexible, but very complex
 - originally designed for complex use-cases that are not needed in ATLAS
 - disproportionate growth in complexity over the years
 - increasingly hard to maintain for the collaboration
- Our goal is to severely simplify our simulation infrastructure and at the same time make FastCaloSim available to future experiments:

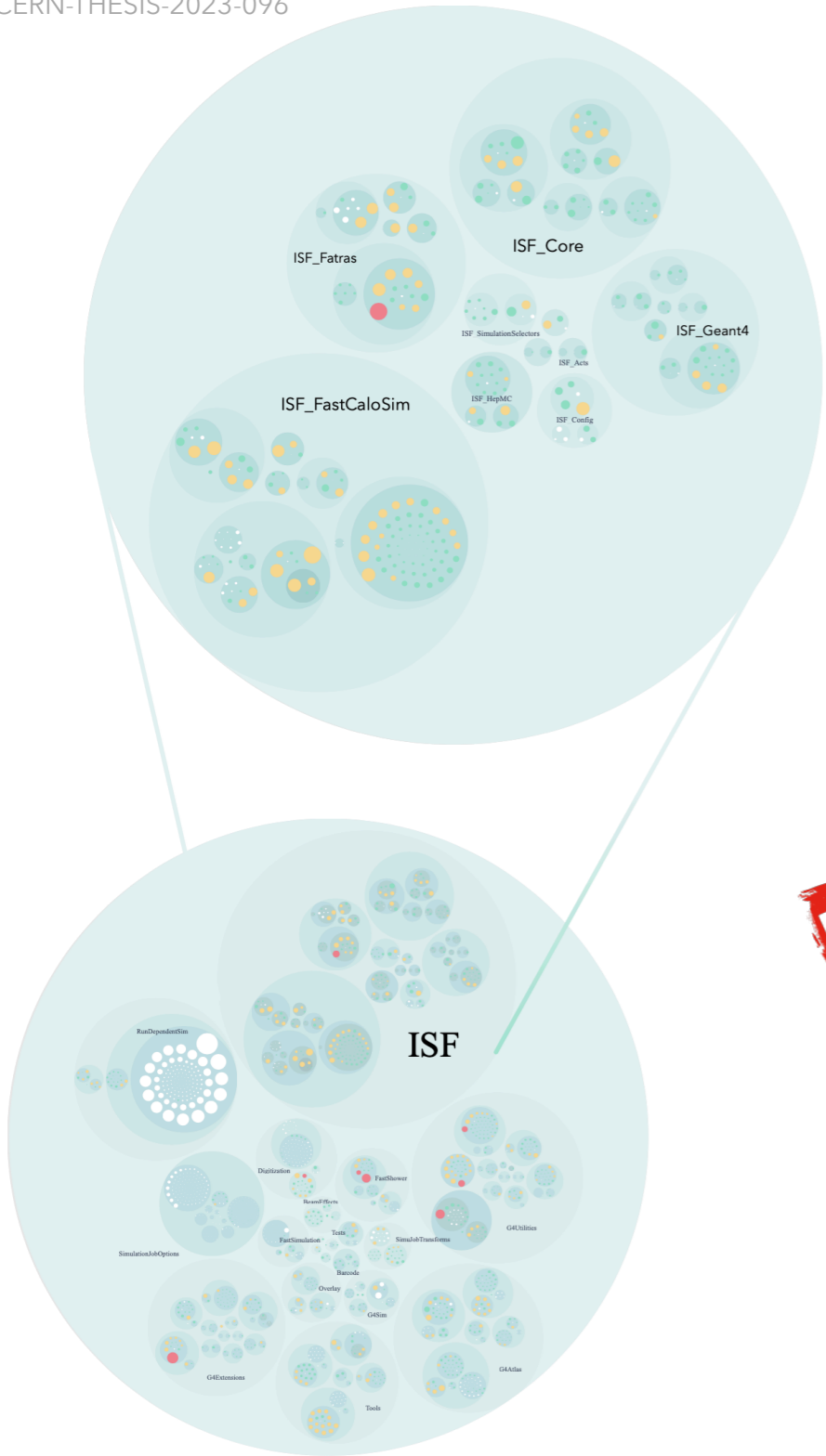
DONE

Implement FastCaloSim as Geant4 fast simulation model

Promote FastCaloSim to an experiment-independent library

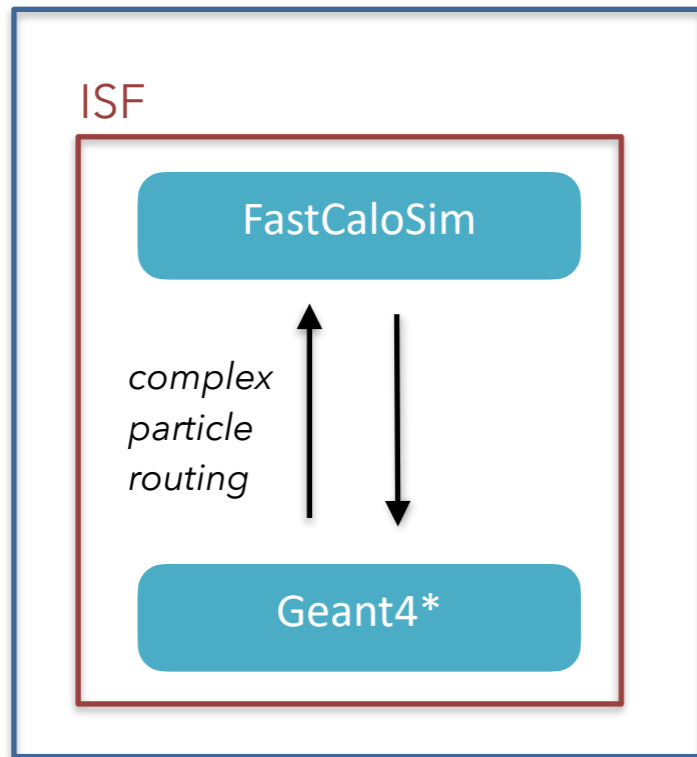
More on this in my [ACAT 2024 talk](#)

Goal: **simple** and **streamlined** ISF-independent ATLAS simulation!

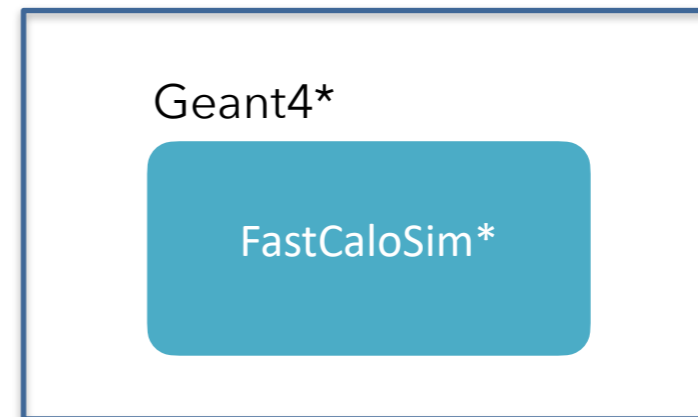




* external library



Experiments software framework



tight integration

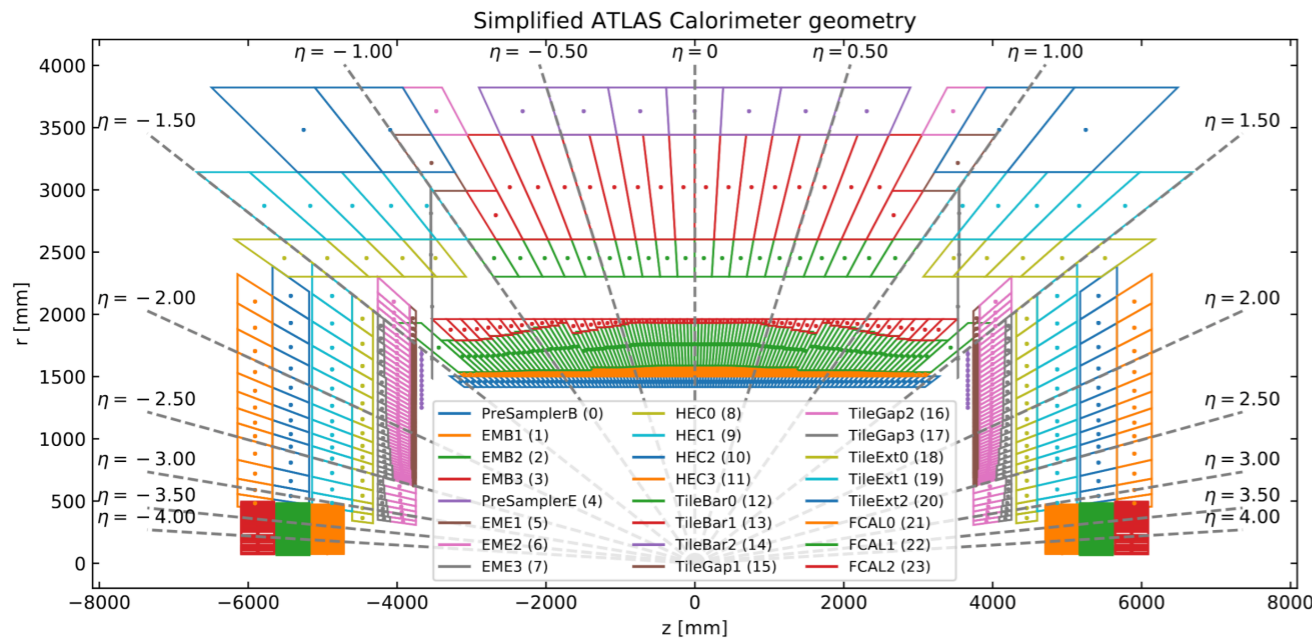
experiment-dependent ISF implementation

experiment-independent Geant4 implementation

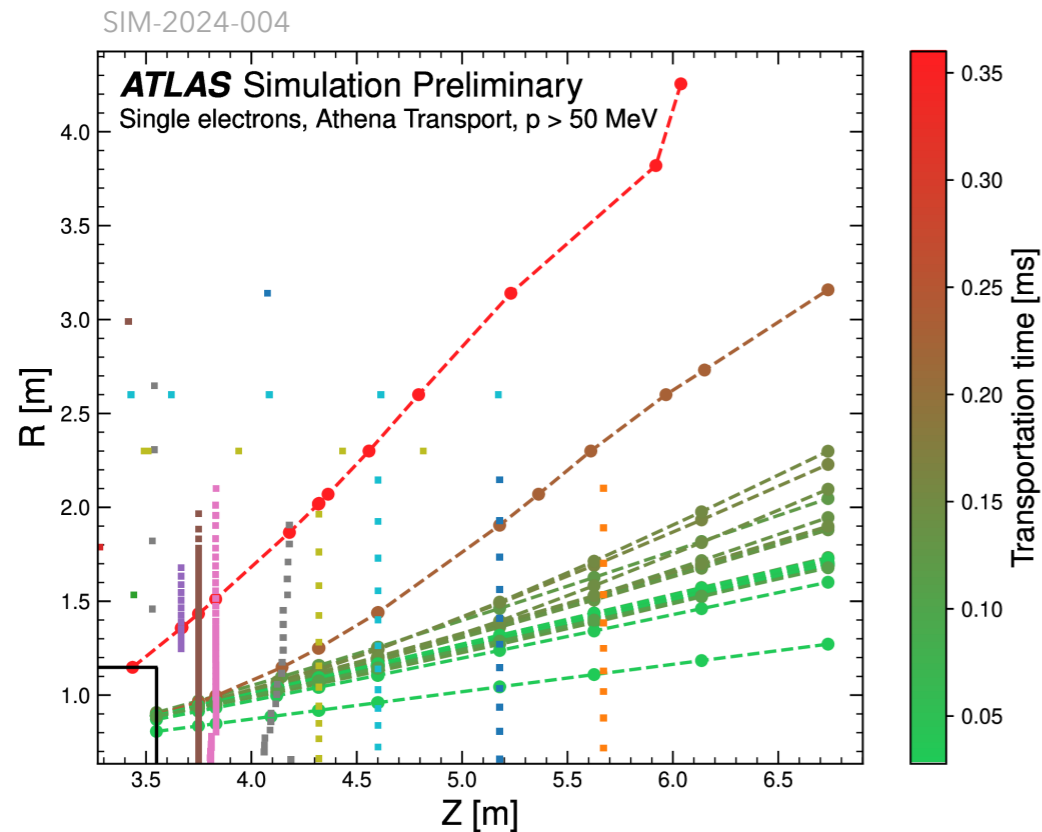
Advantages:

1. Fully eliminates the need for ISF, severely reducing codebase to maintain
2. Fully experiment-independent without athena or ISF dependencies
3. Allows fast local code development in containers
4. Allows to enforce modern best practices in controlled environment
5. Allows for contributions of other HEP experiments

What was required for the restructuring?



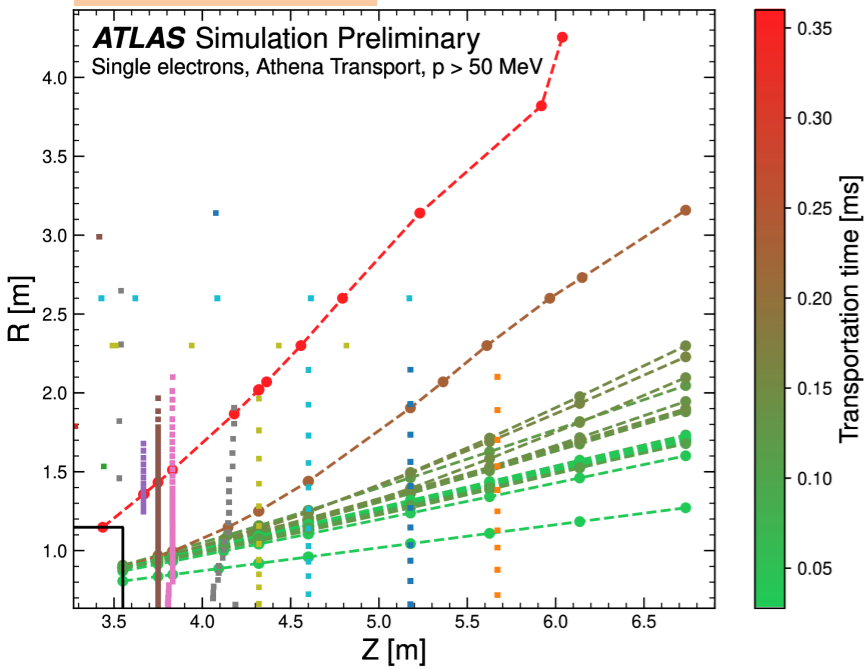
- FastCaloSim relies heavily on an accurate determination of the shower centre position in each calorimeter layer
- To determine the positions, tracks need to be transported through the ATLAS calorimeter system, taking into account magnetic field



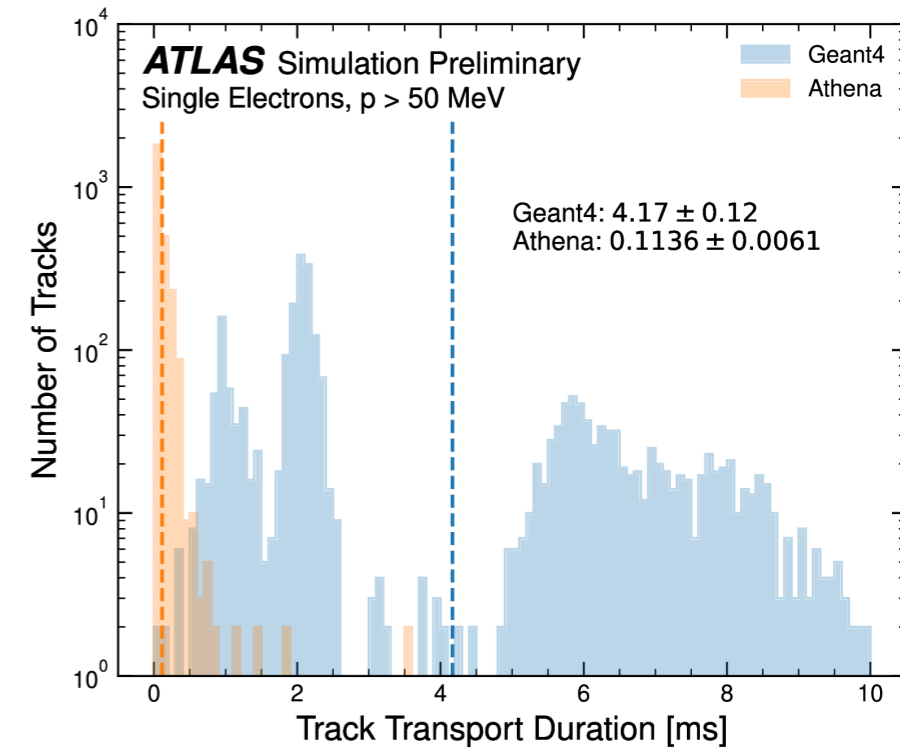
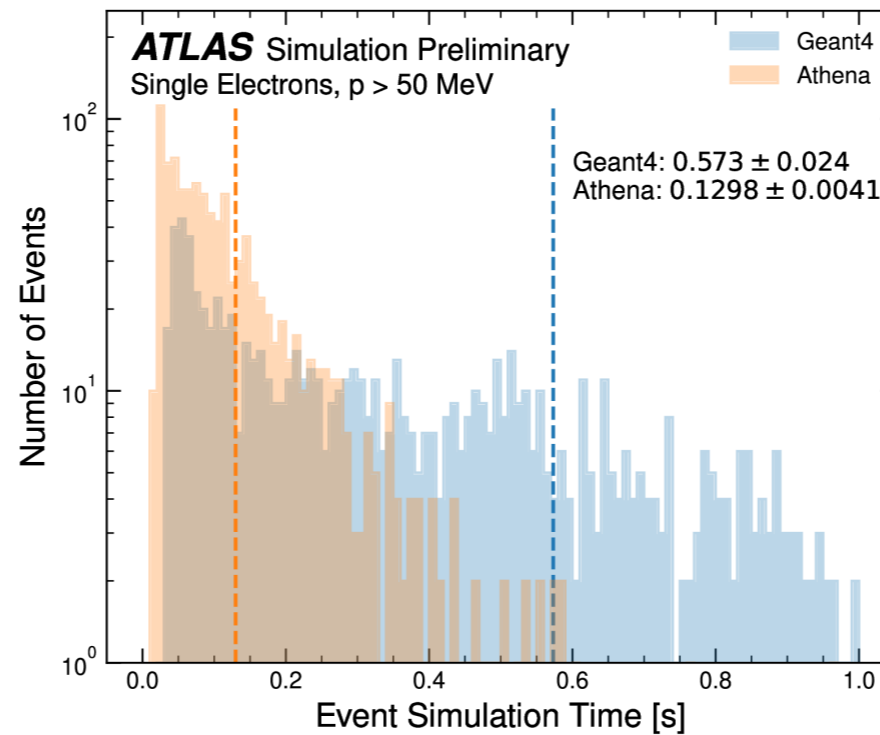
- FastCaloSim currently uses proprietary **Athena tracking tools** to transport particles
- Intersections with active calorimeter layers given as input to FastCaloSim

Athena transport needs to be replaced with experiment-independent Geant4 solutions

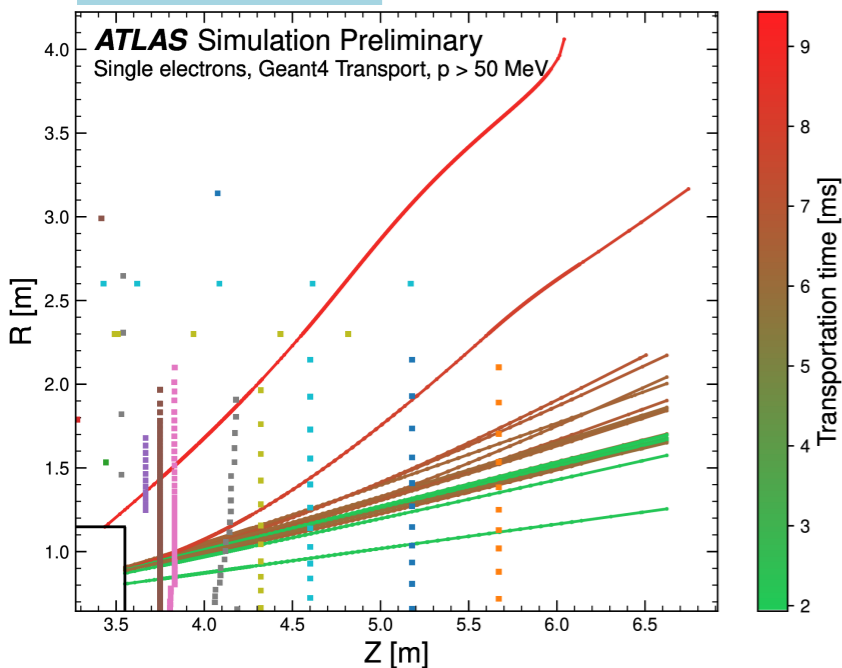
Athena Transport



First attempt: Exploit default Geant4 engine for track transportation



Geant4 Transport



Single electron event simulation time:
x 3 slower with Geant4 transport

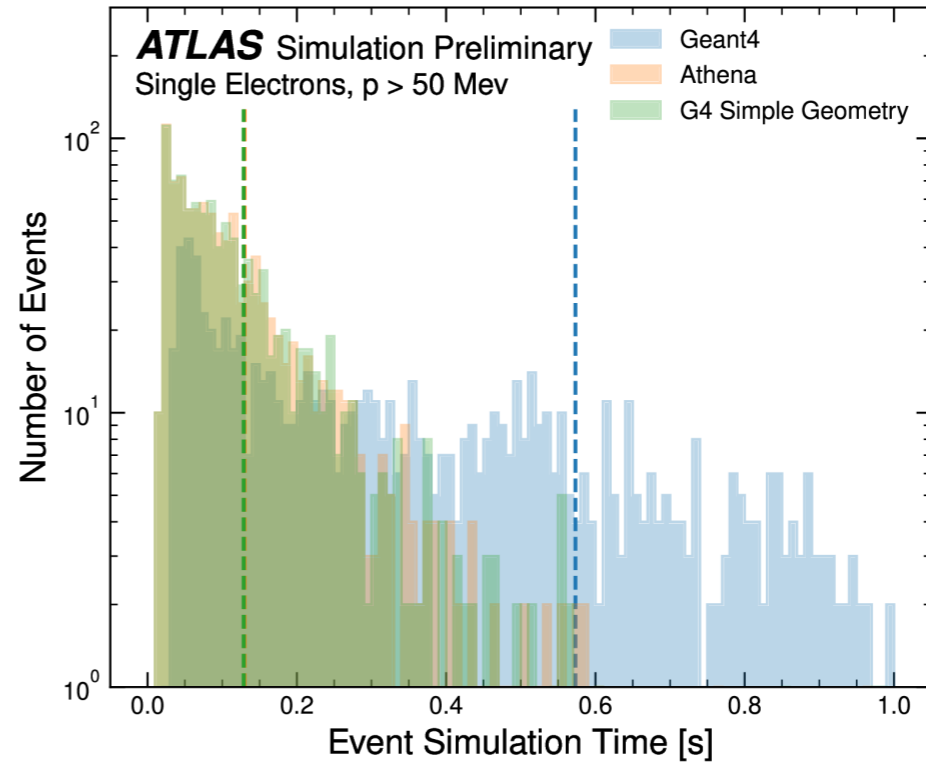
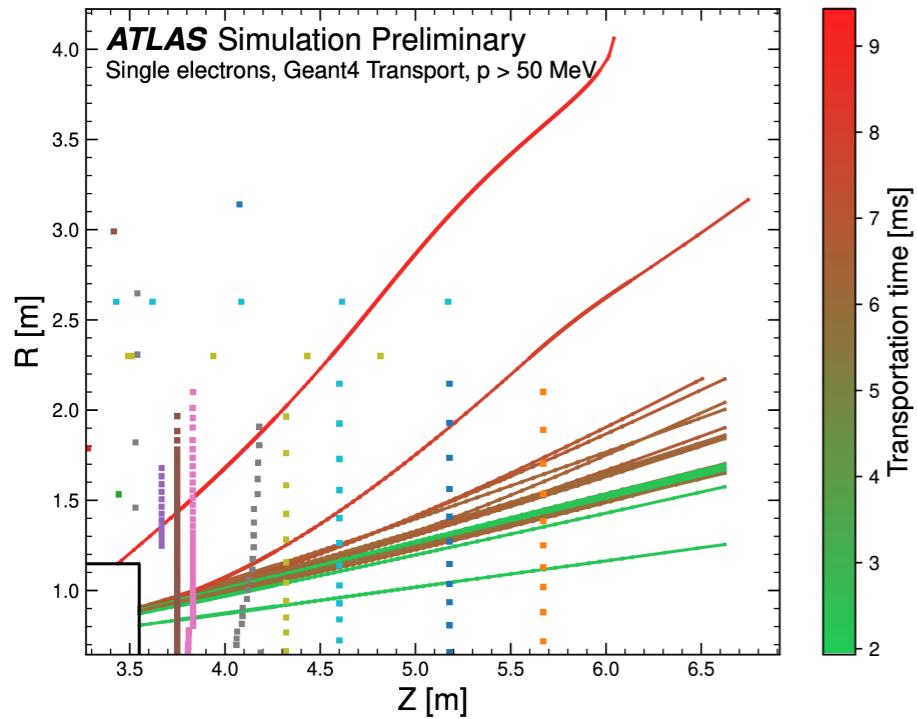
Transport duration per track:
x 34 slower with Geant4 transport



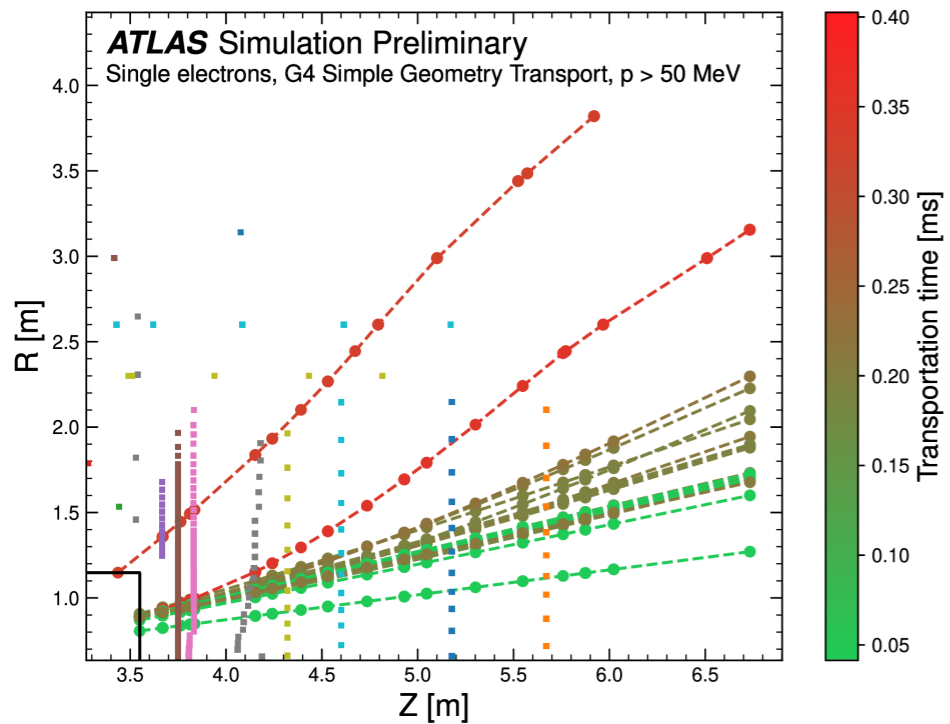
First very naive approach of Geant4 transport **prohibitively slow**

How can we do better?

Geant4 Transport



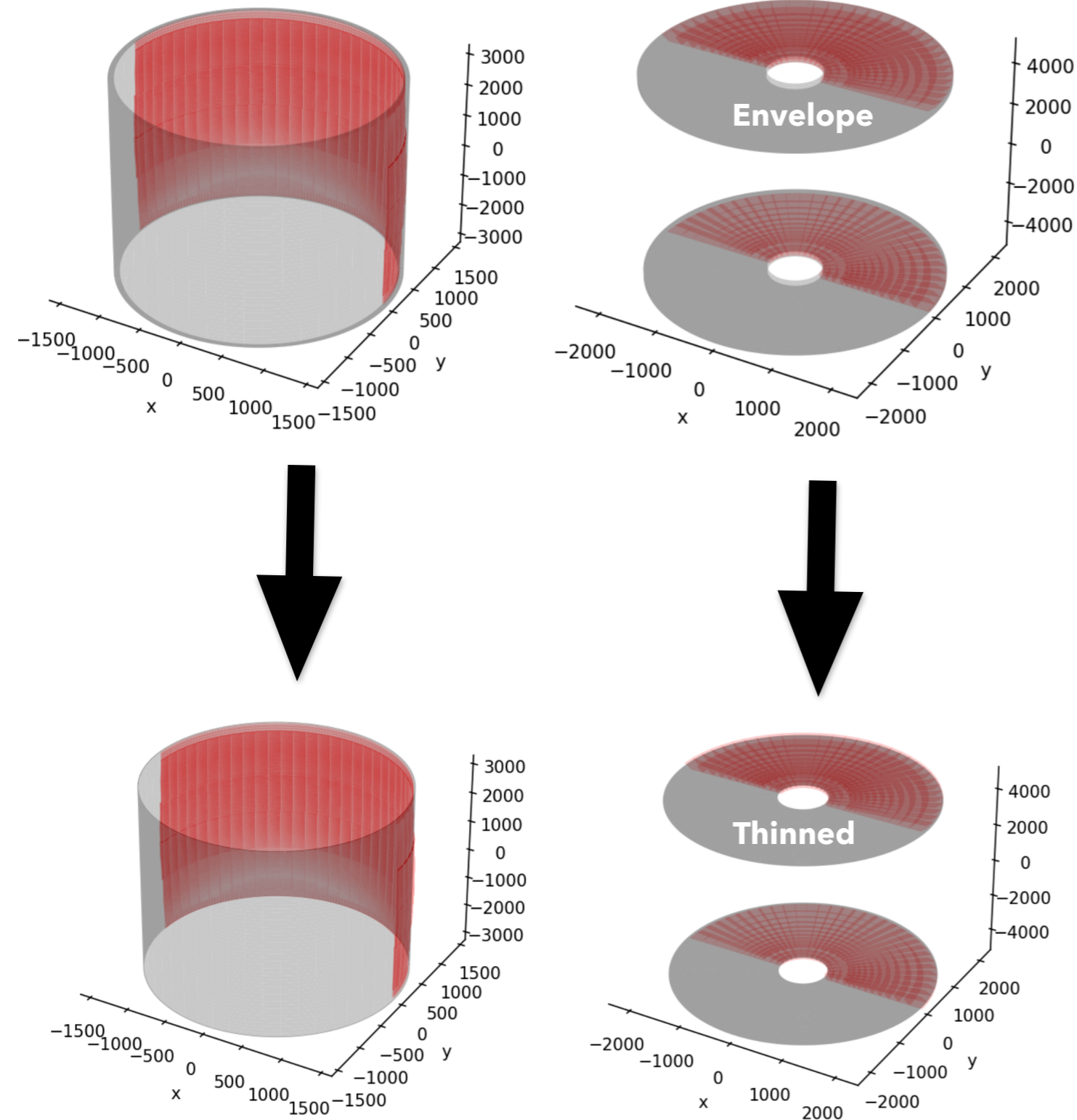
Geant4 Transport in simplified geometry



- Interested only in track position at entry and exit of each calorimeter layer
- Instead of transporting through full calorimeter geometry, do navigation in simplified (layer-based) geometry

Event simulation time of Athena tracking tools recovered with Geant4 navigation in simplified geometry

How to construct simplified geometry?



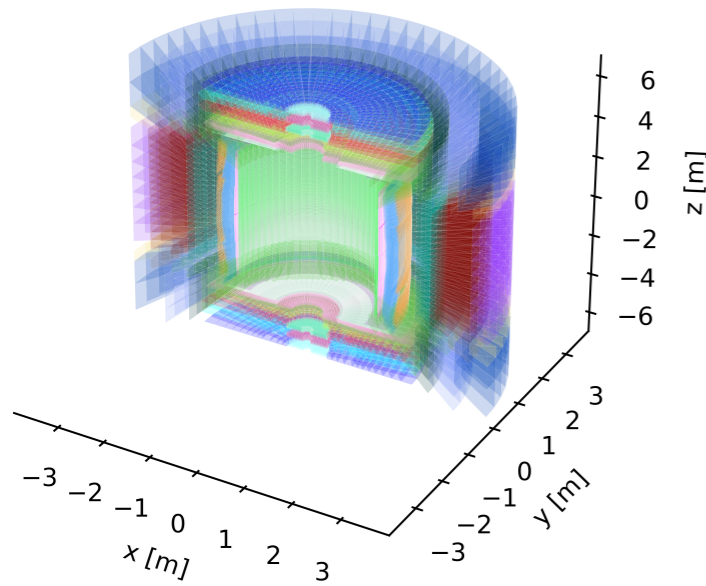
Goal: *Experiment-independent* package to automatically build simplified geometry based on detector cells where:

- Layers are modelled as cylinders
- Cylinder surfaces approximately correspond to real entry and exit of detector layers
- Clash-free

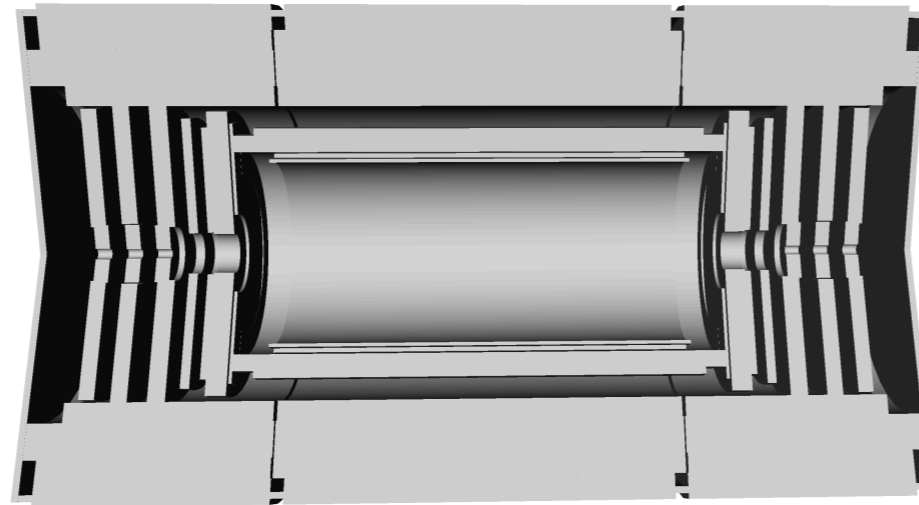
Approach (simplified):

1. Model all layers as cylindrical hulls (=‘envelopes’) from the maximum geometric extension of the cells
2. *Thin down* hulls to generate clash-free geometry
 → for barrel layers thinned $r = r_{mid}^{hull}$
 → for endcap layers thinned $z = z_{mid}^{hull}$
3. Attempt to grow back thinned down layers to original size of envelopes (constraint: only grow up to the limiting layer, i.e. w/o creating overlaps)

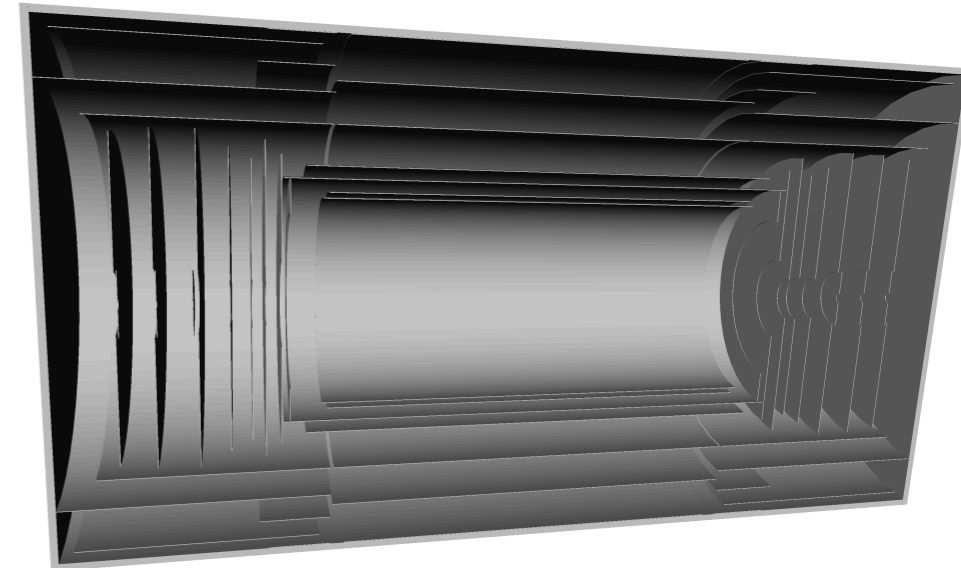
1) Calorimeter Cells



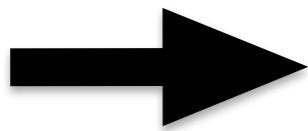
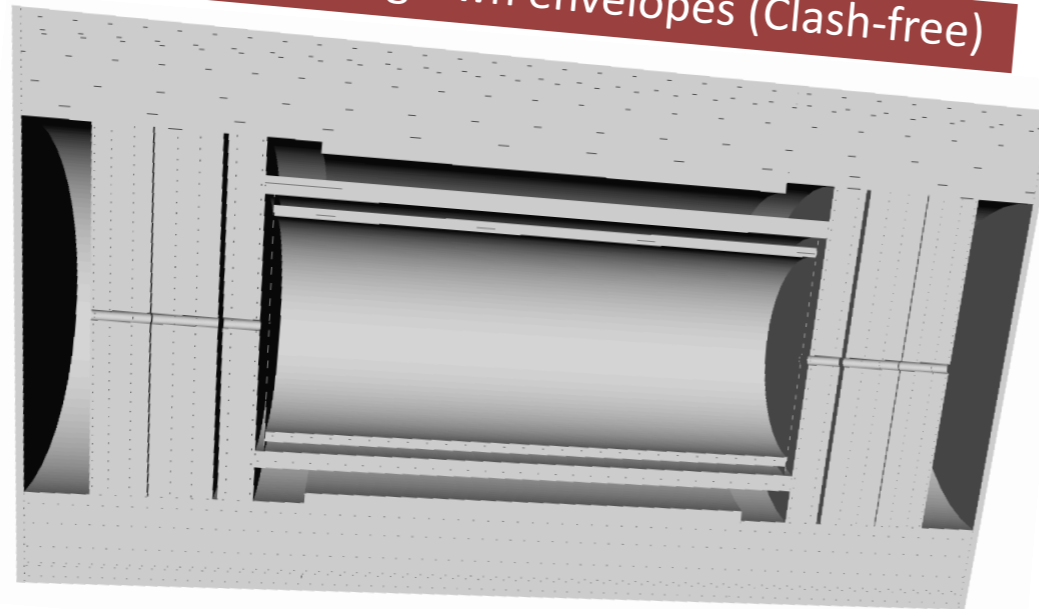
2) Cell Envelopes (NOT clash-free)



3) Thinned envelopes (Clash-free)



4) Processed / grown envelopes (Clash-free)



```

**** Real time elapsed   : 0.00852796
**** User time elapsed  : 0.01
**** System time elapsed : 0

Size of G4SolidStore      : 50
Size of G4LogicalVolumeStore : 50
Size of G4PhysicalVolumeStore : 50

Number of volumes : 50 (2 levels)
Number of volumes checked : 50
Number of clashes detected : 0
    
```

pygeosimplify Public

Pin Unwatch 1 Fork 0 Starred 1

main 7 Branches 10 Tags

Go to file Add file Code

About

Python package for automatic, cell-based inference of clash-free simplified detector geometry

Readme MIT license Activity 1 star 1 watching 0 forks

jbeirer Merge pull request #142 from jbeirer/dependabot/pip/exceptiongro... 3e65b5e · 3 weeks ago 169 Commits

Folder	Check Status	Time	Age
.devcontainer	All checks have passed	9 successful checks	3 months ago
.github			4 months ago
docs	Main / quality (push) Successful in 1m	Details	10 months ago
pygeosimplify	validate-codecov-config / validate-codecov-config (push) Successful in 8s	Details	3 weeks ago
tests	Main / tox (3.9) (push) Successful in 2m	Details	3 weeks ago
	Main / tox (3.10) (push) Successful in 2m	Details	
	Main / tox (3.11) (push) Successful in 2m	Details	
	Main / tox (3.12) (push) Successful in 2m	Details	
	Main / check-docs (push) Successful in 2m	Details	
	codecov/patch - Coverage not affected when comparing 3f36eb1...3e65b5e	Details	
	codecov/project - 90.7% (target 90.0%)	Details	

Project description

pyGeoSimplify

release v0.0.4 build passing codecov 93% commit activity 78/month license MIT

Welcome to pyGeoSimplify!

Download pyGeoSimplify

```
pip install pygeosimplify
```

Quick Start

```
import pygeosimplify as pgs
from pygeosimplify.simplify.layer import GeoLayer
from pygeosimplify.simplify.detector import SimplifiedDetector

# Set names of branches that specify coordinate system of cells
pgs.set_coordinate_branch("XYZ", "isCartesian")

# Load geometry
geo = pgs.load_geometry("DetectorCells.root", tree_name='treeName')

# Create simplified detector
detector = SimplifiedDetector()

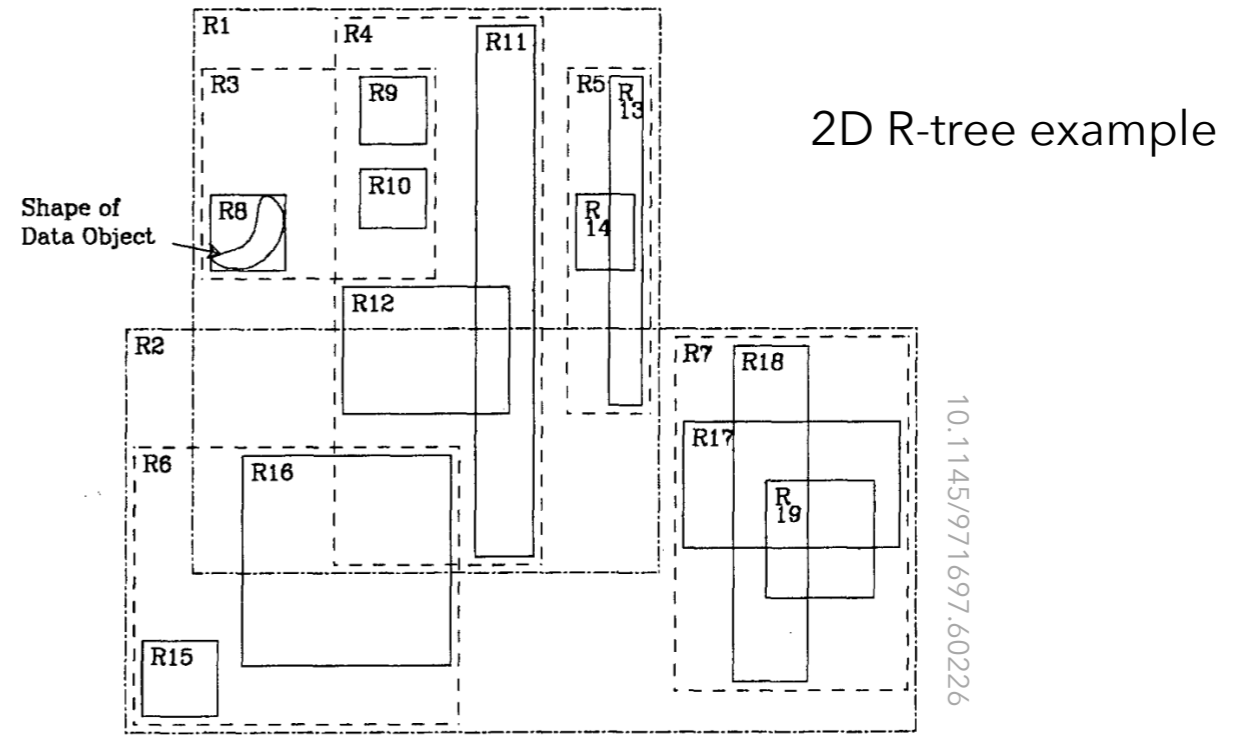
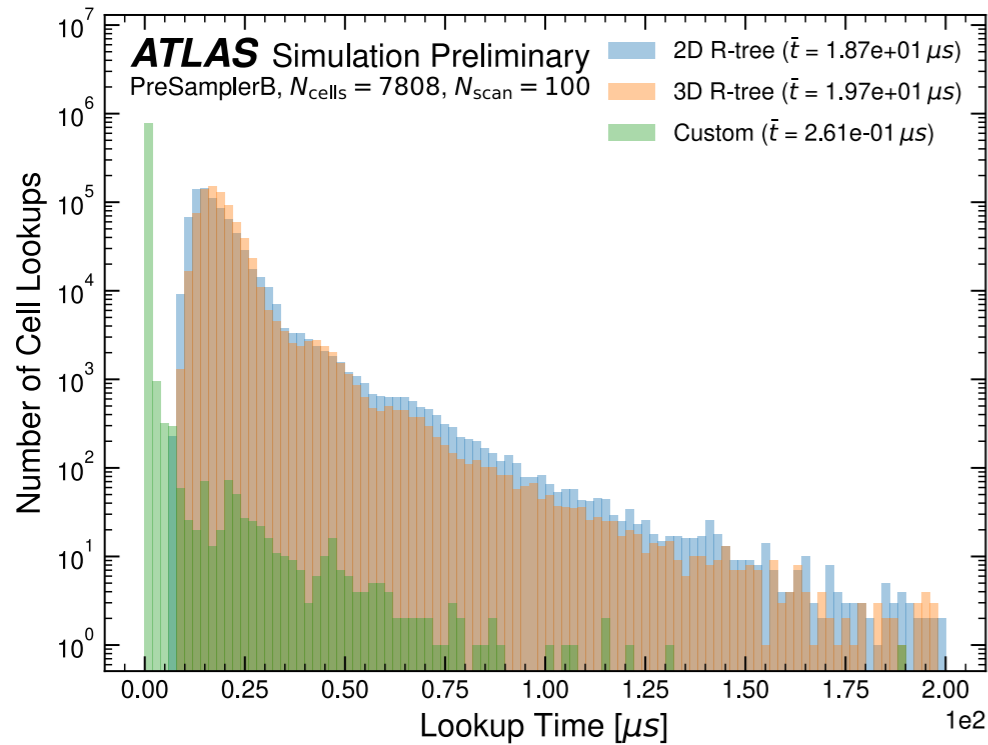
# Add detector layers to detector
layer = GeoLayer(geo, layer_idx)
detector.add_layer(layer)

# Process detector
detector.process()

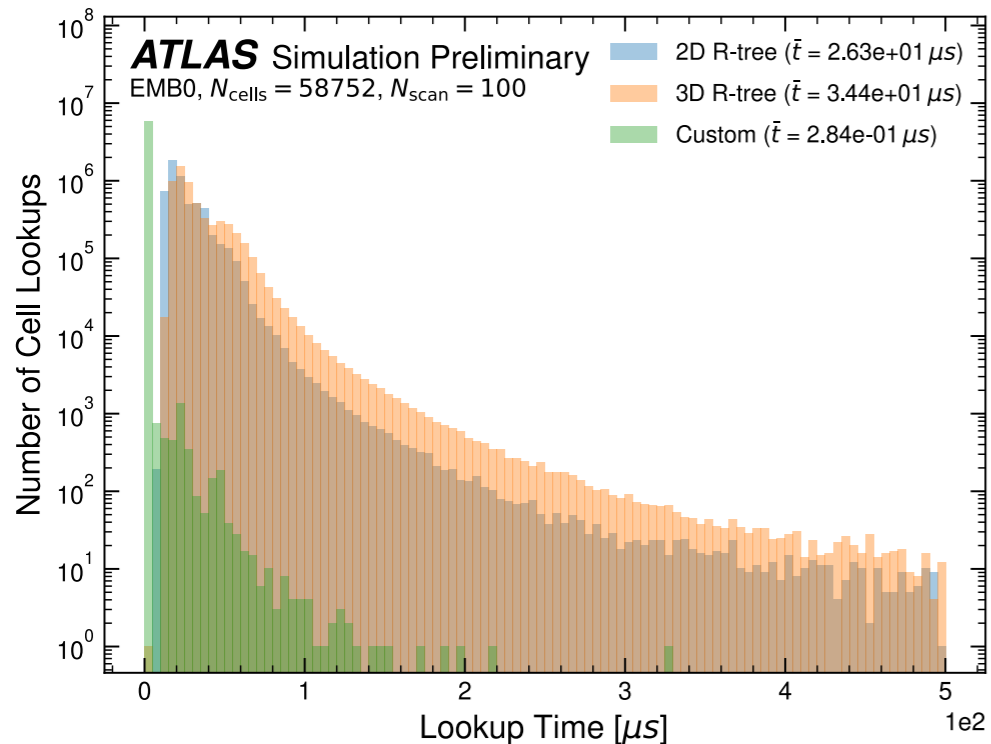
# Save simplified detector to gdml file
detector.save_to_gdml(cyl_type='processed', output_path='processed.gdml')
```

- All functionality integrated in python package names pyGeoSimplify
- **Input:** ROOT file with cell positions and dimensions
- **Output:** GDML file of clash-free simplified detector
- Extensive testing with over 90% test coverage
- [pyGeoSimplify](#) available on PyPi: `pip install pygeosimplify`

SIM-2024-009

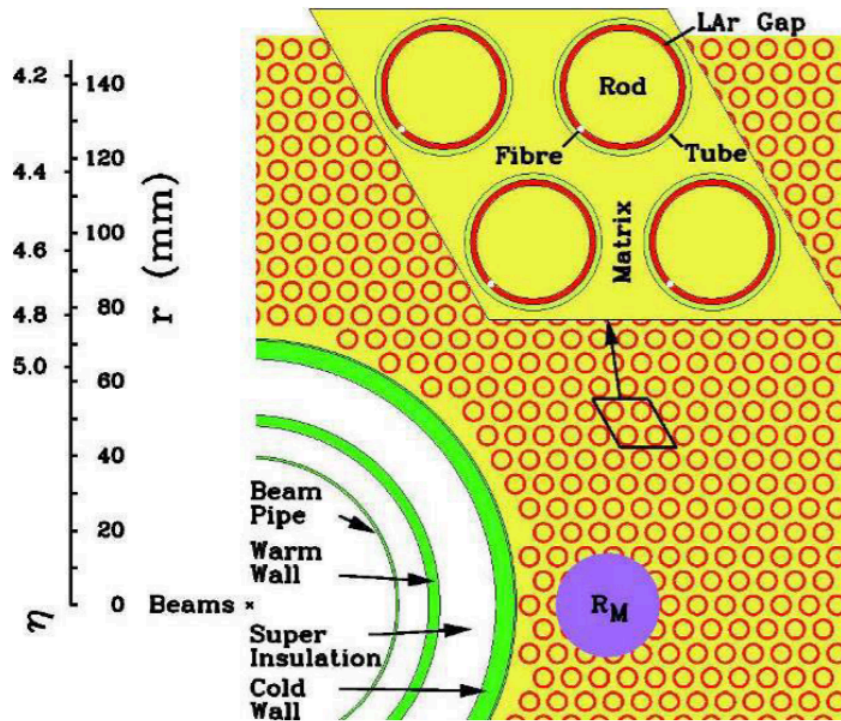


SIM-2024-009



- During simulation, need to match sampled hits to best-matching cells
- Cells used to be found by storing complex, experiment-dependent geometry maps
- Re-implemented hit-to-cell matching by organizing cell information into [R-trees](#) for **fast experiment-independent lookup**, allowing to remove thousands of lines of code
- Experiment-independent lookup up to 100x slower compared to maps, but not expected to become a bottleneck

ATLAS (irregular) FCAL geometry



- The hit-to-cell matching assumes detector cells can approximately be described as cuboids in some coordinate system
- If this approximation is not good enough (or experiments need a faster cell lookup), custom geometry handlers can be easily implemented
- For ATLAS, the cells of the forward calorimeter are highly irregular and cuboid description not sufficient:

```
// Create alternative geometry handler for FCal
std::shared_ptr<FCal> fcal_geo = std::make_shared<FCal>();
// Load the FCal geometry from the files
fcal_geo->load(AtlasGeoTestsConfig::FCAL_ELECTRODE_FILES);
// Cast the FCal geometry handler to the geo interface
std::shared_ptr<CaloGeo> fcal_geo_handler =
    std::static_pointer_cast<CaloGeo>(fcal_geo);
// Set the alternative geometry handler for the FCal layers (21 - 23)
geo->set_alt_geo_handler(21, 23, fcal_geo_handler);
```

Class that implements method to return cell id, given a position in detector

Alternative geo handler can be set for arbitrary detector layers

main 3 Branches 1 Tag

Go to file Code

jbeirer Update CMakeLists.txt ✓ 3a84e79 · 18 minutes ago 143 Commits

.devcontainer	Add gitlens to devcontainer.json	2 months ago
.github/workflows	Update ci.yml	29 minutes ago
.hooks	Add pre-commit hooks (#24)	4 months ago
cmake	Update CMakeLists for Athena (#43)	last month
docker	Add licensing (#46)	yesterday
docs	Initial commit	7 months ago
example	Add licensing (#46)	yesterday
include/FastCaloSim	Update ParticleData.h	1 hour ago
source	Add licensing (#46)	yesterday
test	Add licensing (#46)	yesterday
.clang-format	Initial commit	7 months ago
.clang-tidy	Initial commit	7 months ago
.clangd	Fix CMake preset config (#38)	last month
.codespellrc	Remove HepPDT dep (#45)	last month
.gitignore	Fix CMake preset config (#38)	last month

About
An experiment-independent library for fast calorimeter simulation

- Readme
- Apache-2.0 license
- Code of conduct
- Activity
- Custom properties
- 2 stars
- 1 watching
- 0 forks

Report repository

Releases
1 tags

Packages
No packages published

Languages

C++	93.7%
C	2.5%
CMake	2.1%
Python	1.1%
Other	0.6%

- The new experiment-independent FastCaloSim was open-sourced TODAY!

fcs-proj / FastCaloSim Public



- Built with modern development practices in mind:

- Modern CMake configuration with [CMake presets](#)
- Unit testing with [googletest](#)
- Static code analysis with [clang-tidy](#) and [cppcheck](#)
- Code linting and formatting with [clang-format](#)
- Spell check with [codespell](#)
- [LCov](#) to provide test coverage information
- Docker images to create reproducible test and development environment

README Code of conduct Apache-2.0 license

FastCaloSim

Pipeline passing release v0.1.0-alpha codecov 37% docs Doxygen License Apache 2.0 Contributor Covenant 2.1

FastCaloSim is an experiment-independent toolkit for the fast parametrised and ML-based simulation of electromagnetic and hadronic showers in (high energy) physics experiments implemented in C++.

Quick Start

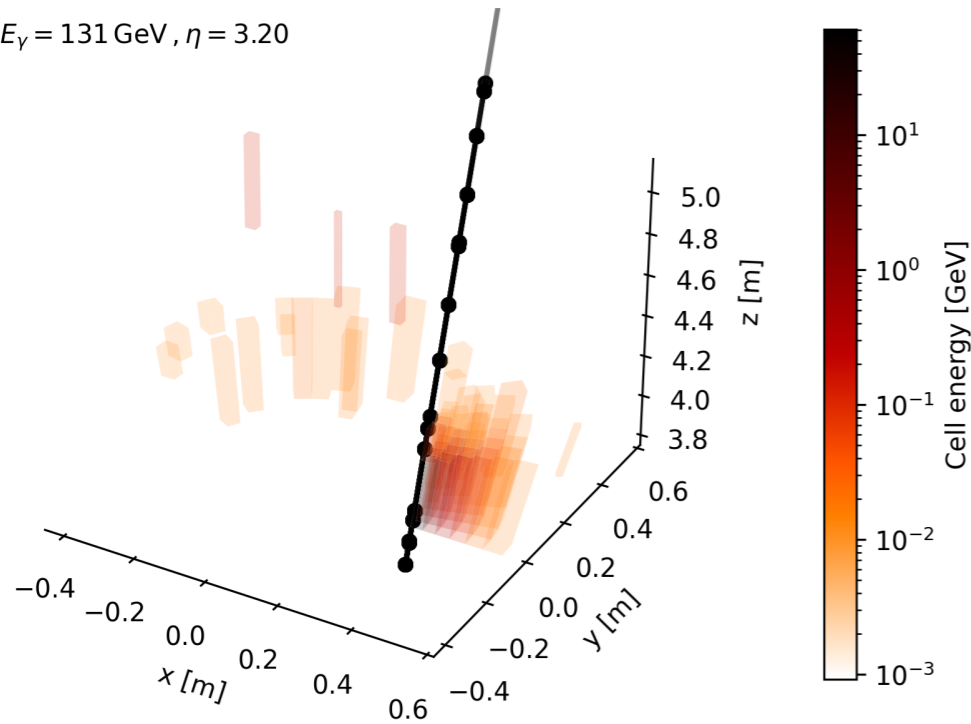
FastCaloSim is developed in C++ and is build using [CMake](#). The following commands will clone the repository, configure, and build the library

```
git clone https://github.com/fcs-proj/FastCaloSim <source>
cmake -B <build> -S <source> -D CMAKE_BUILD_TYPE=Release
cmake --build <build>
```

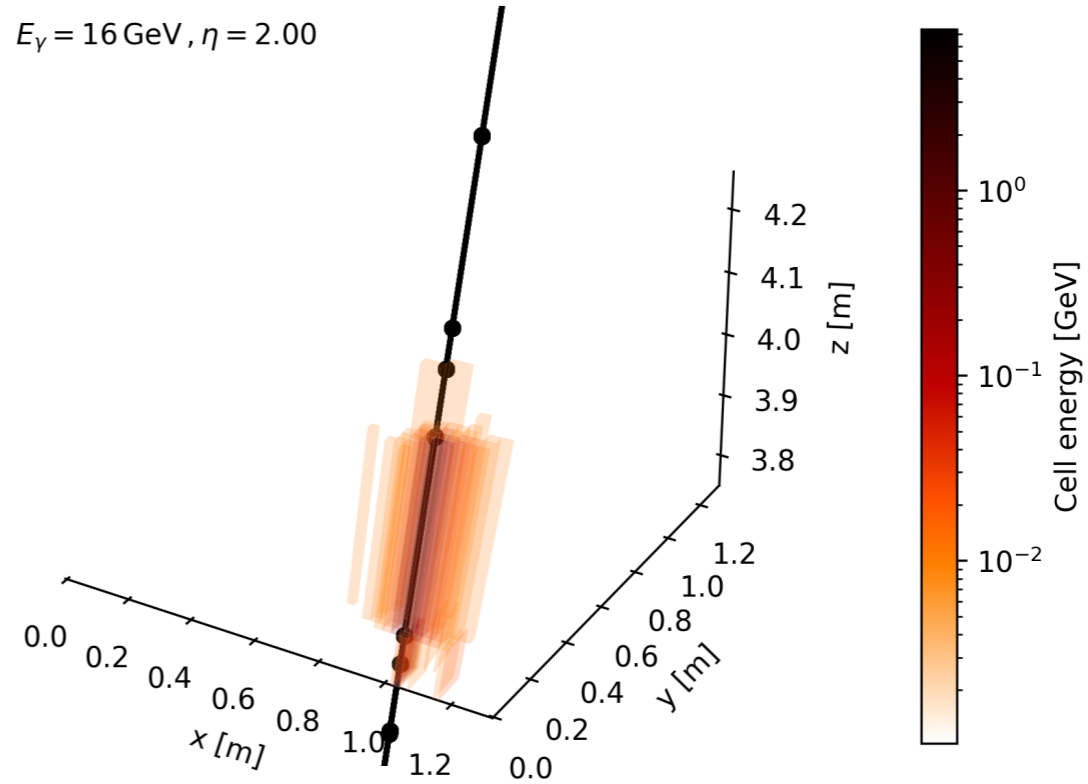
For install options and instruction on how to include FastCaloSim in your experiment see the [BUILDING](#) document. For advanced developer configuration with [presets](#) and other useful information see the [HACKING](#) document.

Continuously tested on alma9, ubuntu24 and LCG releases

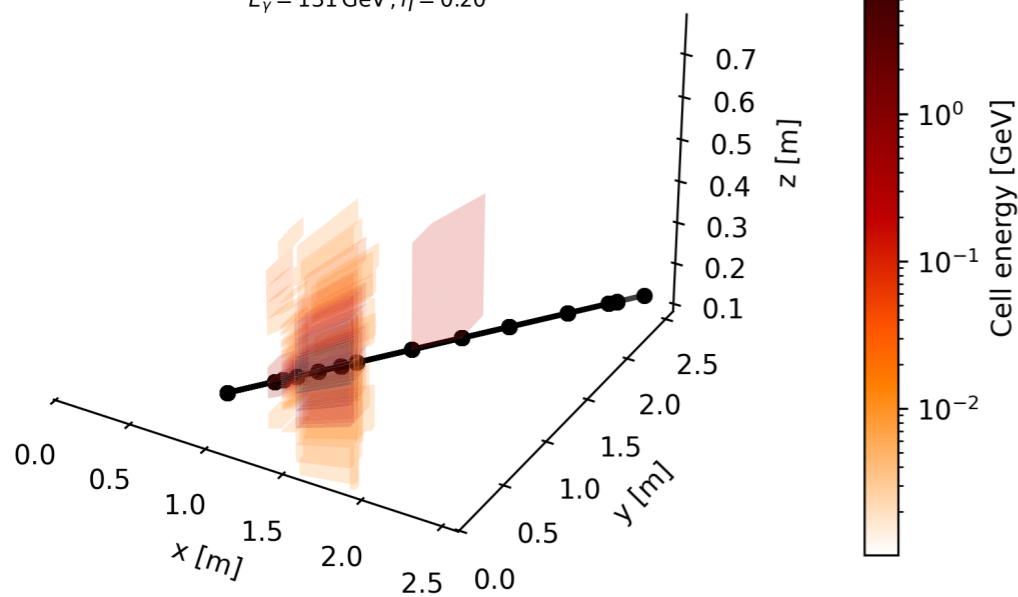
$E_\gamma = 131 \text{ GeV}, \eta = 3.20$



$E_\gamma = 16 \text{ GeV}, \eta = 2.00$



$E_\gamma = 131 \text{ GeV}, \eta = 0.20$



- A full example implementation for FastCaloSim for ATLAS as Geant4 fast simulation model is provided
- The full simulation chain is unit-tested (transport, extrapolation, simulation)
- Visualisations of the shower simulation are created on-the-fly in the CI

Summary

- For the first time, FastCaloSim is implemented as single external library that can be used outside Athena without any ATLAS geometry dependencies
- [pyGeoSimplify](#) allows to automatically generate clash-free simplified geometry needed by FastCaloSim
- New FastCaloSim repository follows modern practices and allows for far more efficient development
- Very much on track to fully phasing out ISF, creating a simple and streamlined (fast) simulation in ATLAS

Outlook

- Core library experiment-independent, but tools to allow easy creation of parametrization still in development
- Plan is to use Open Data Detector (ODD) as proof-of-concept implementation starting from generation of Geant4 input samples, creation of parametrization and simulation

Thank You

BACKUP