ACAT 2024

STONY BROOK, NEW YORK

ATLAS EXPERIMENT

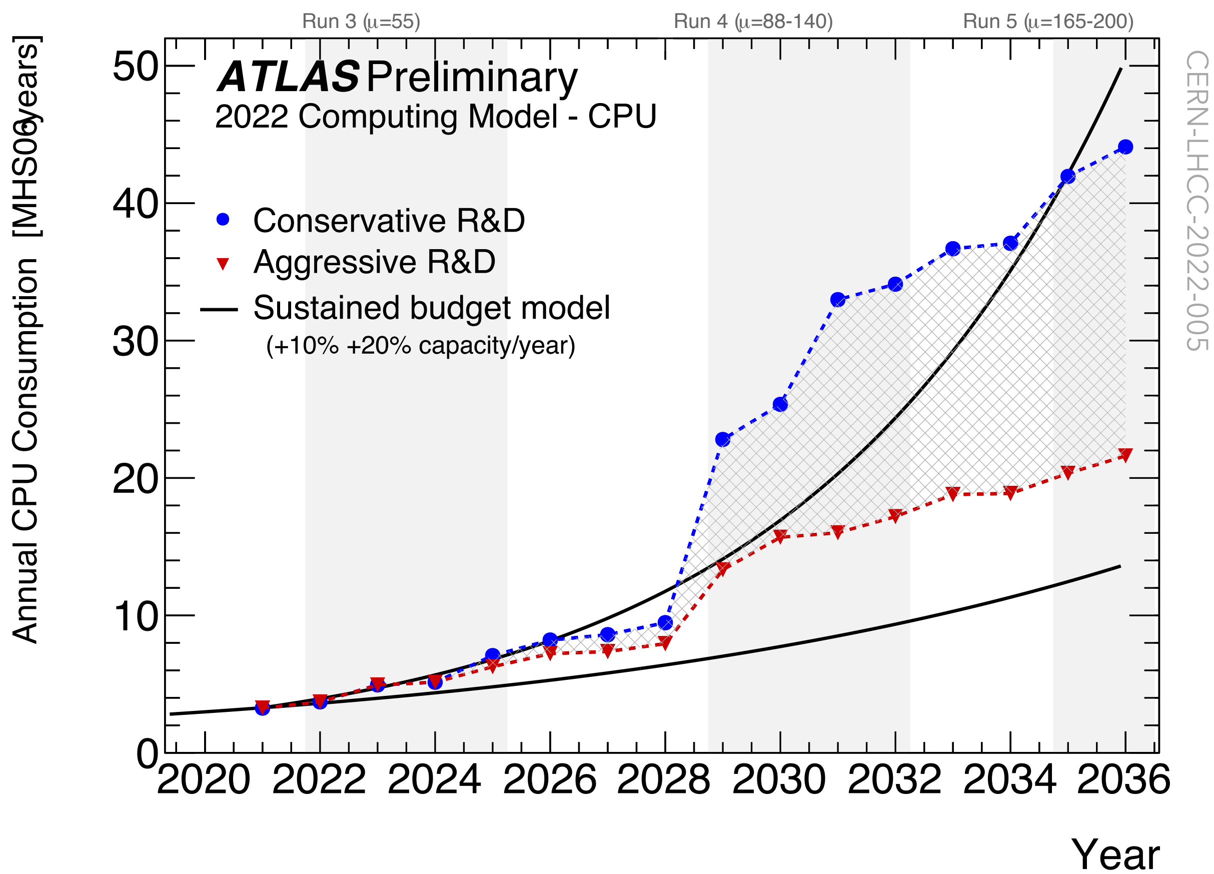
Towards a simplified (Fast) Simulation Infrastructure in ATLAS

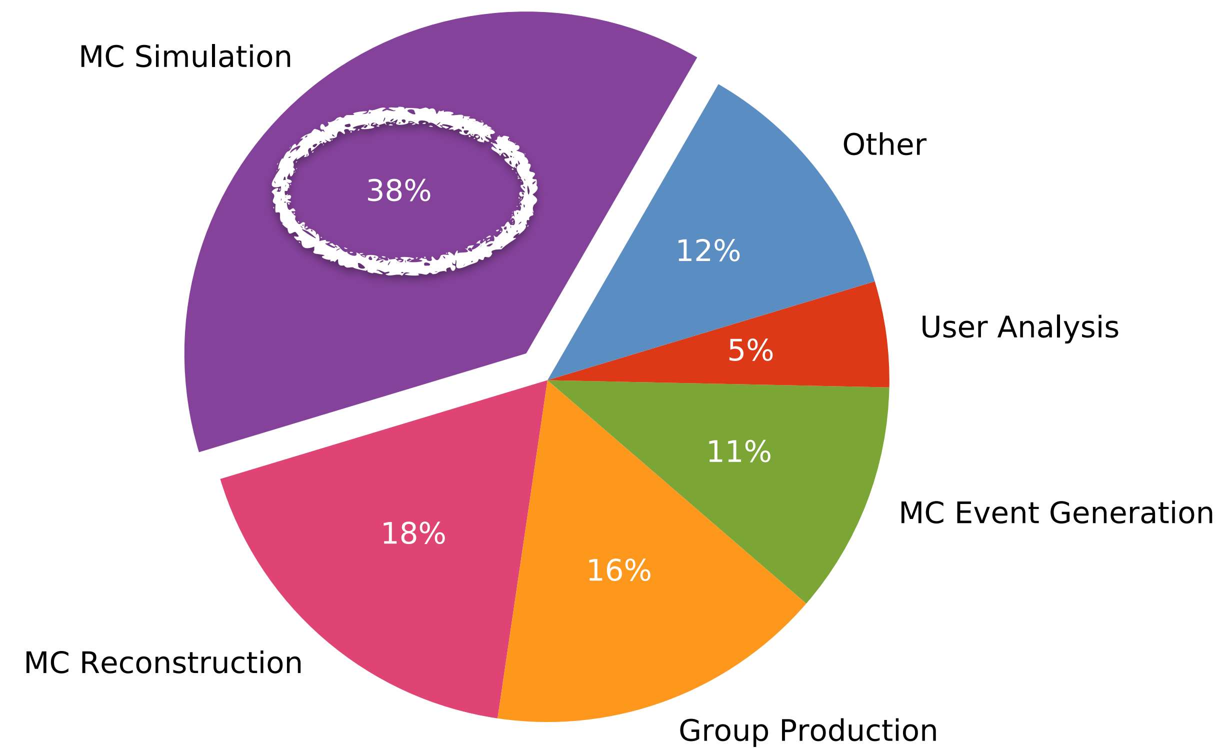
JOSHUA F. BEIRER (CERN)

ON BEHALF OF THE ATLAS COLLABORATION

EP R&D

Motivation: Fast Simulation





- 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

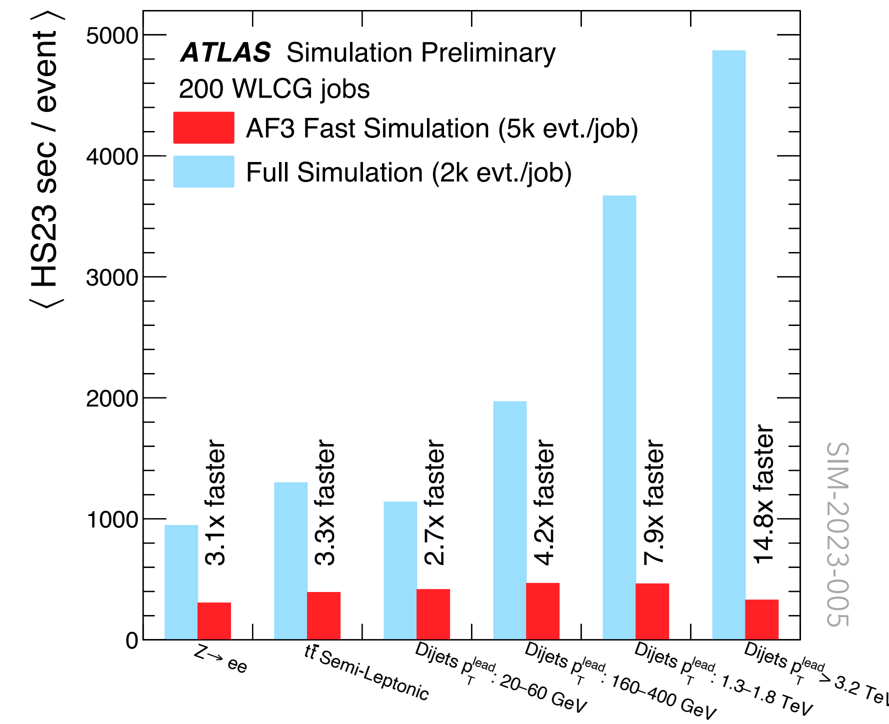
→ **use (fast) parametric models to reproduce Geant4 simulation as accurately as possible**

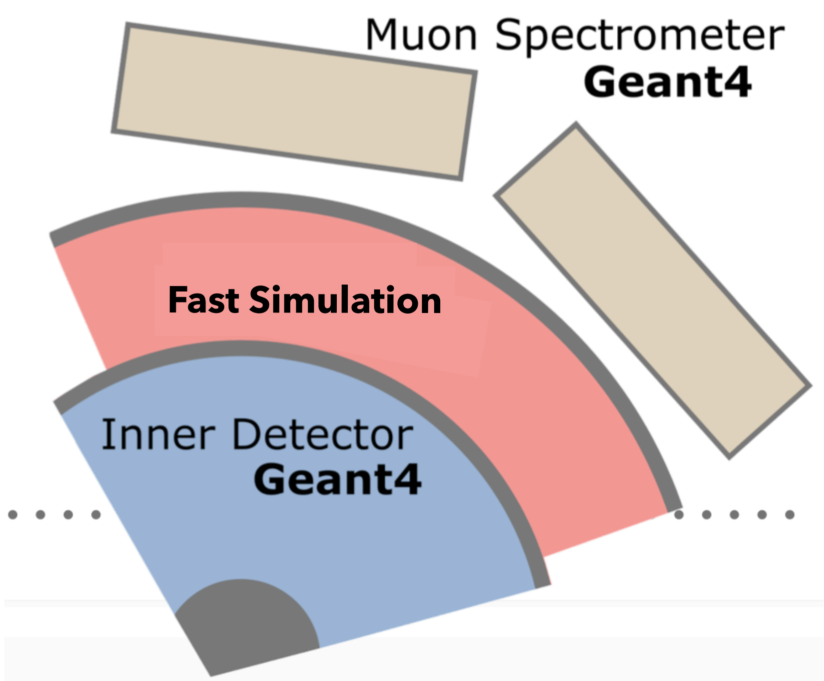
AtlFast3: State-of-the-Art Fast Simulation

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

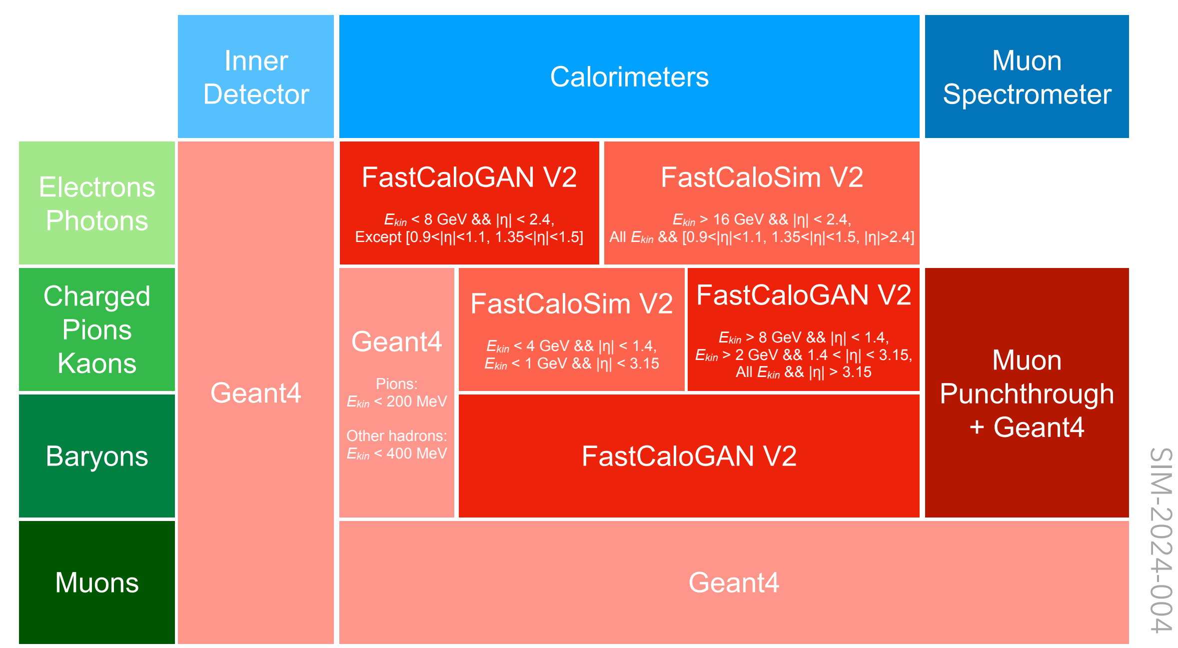
- **Basic Principle:** instead of tracking each particle in calorimeter showers, parametrise energy response with single particles
- Two distinct approaches of shower generation:
  - ***FastCaloSimV2:*** classical parametrised modelling
  - ***FastCaloGAN:*** Generative Adversarial Network
- **3-15x**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

3-15 x increase in simulation speed



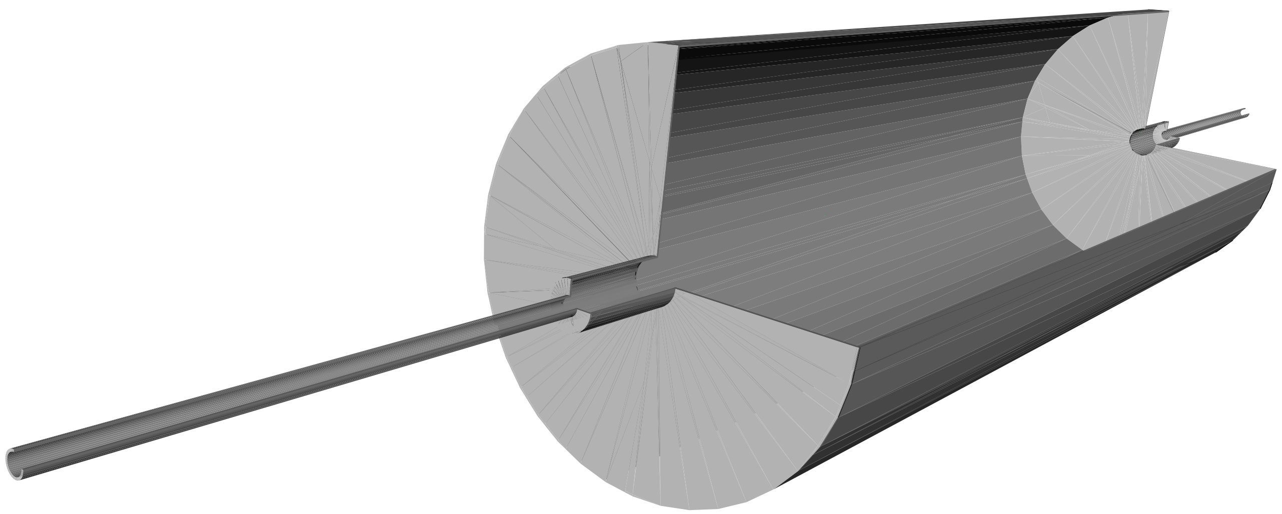


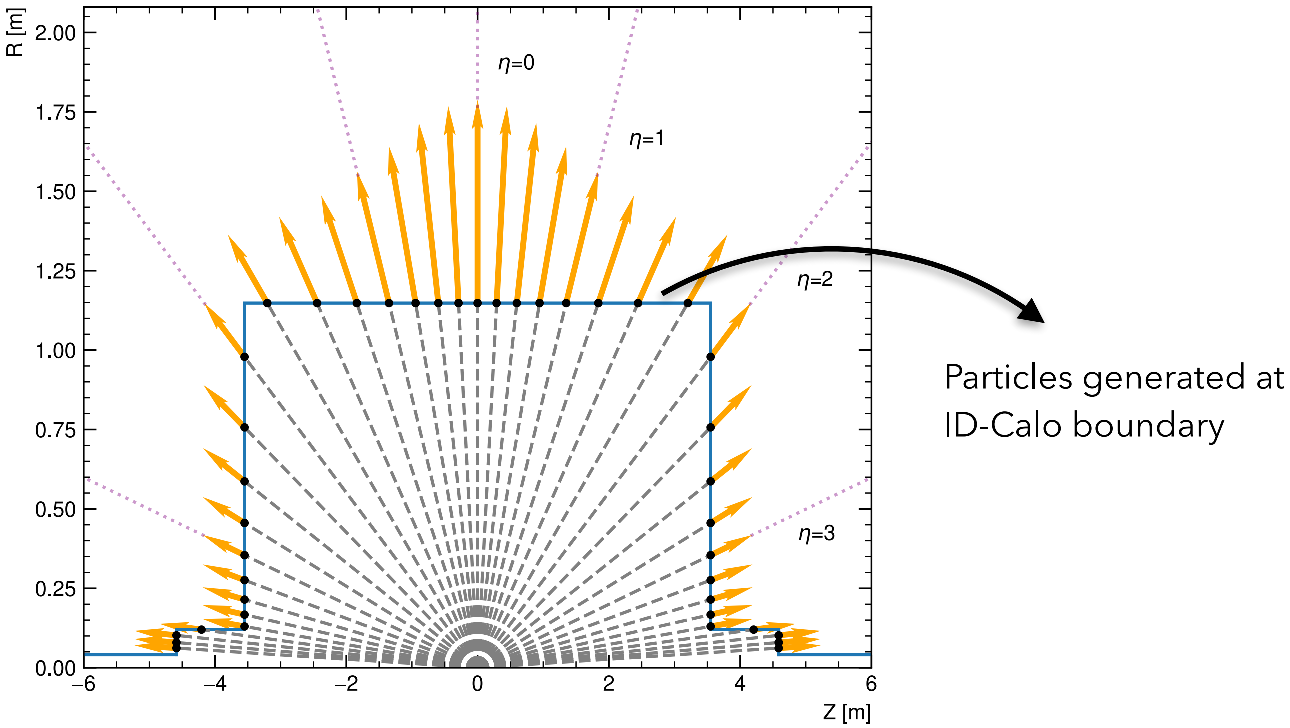
Run 3 Contiguration



Simulation time fully dominated by Geant4 simulation in ID!

Boundary between Inner Detector and Calorimeter System





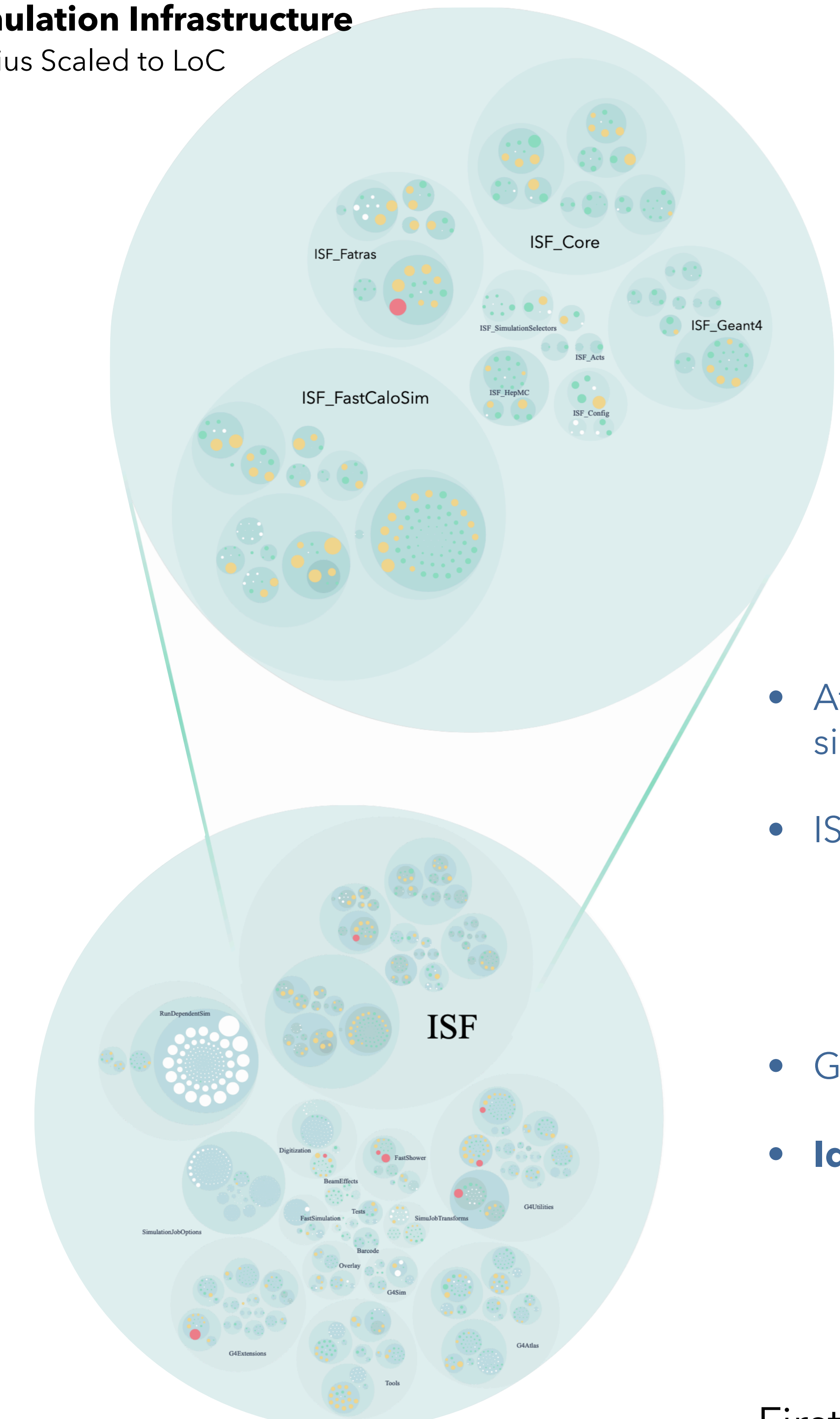
CERN-THESIS-2023-096

- Three separate parametrisations
  - **EM showers:**
    1. Photons γ
    2. Electrons / Positrons $e^{\pm}$
  - **Hadronic Showers:**
    3. Charged Pions $\pi^{\pm}$
- 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 |η| from 0 to 5.0

Talk to me for more on AtlFast3 for Run 3!

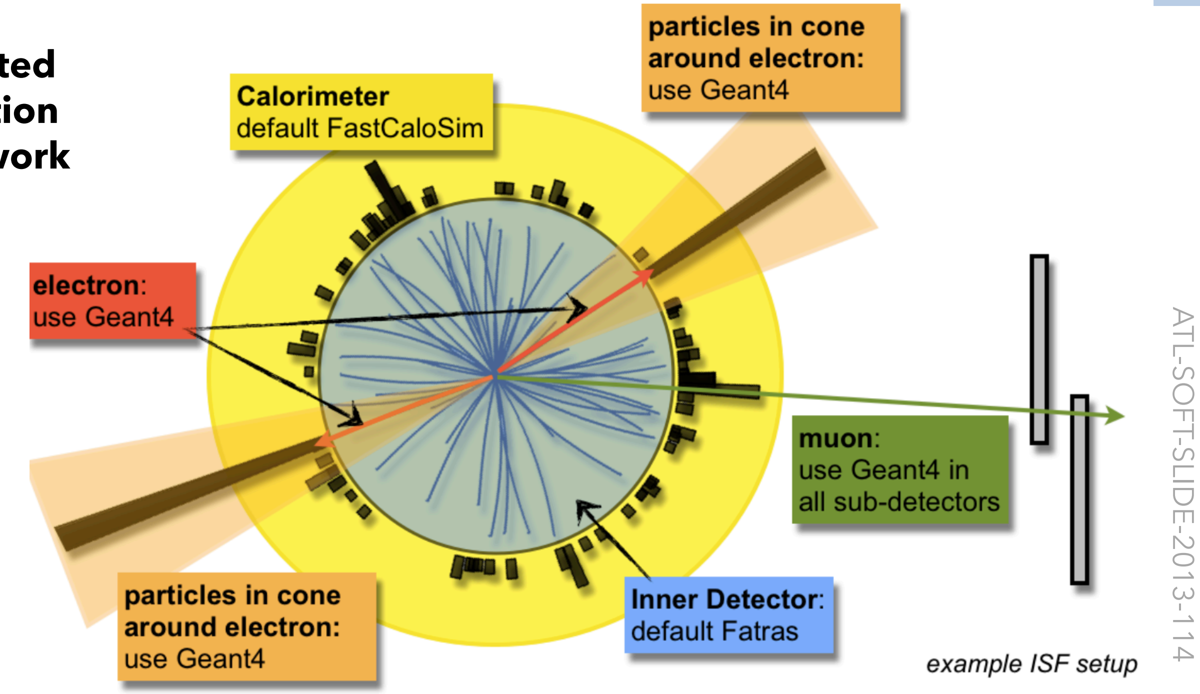
FastCaloSim in Geant4: Motivation

Simulation Infrastructure
Radius Scaled to LoC



CERN-THESIS-2023-096

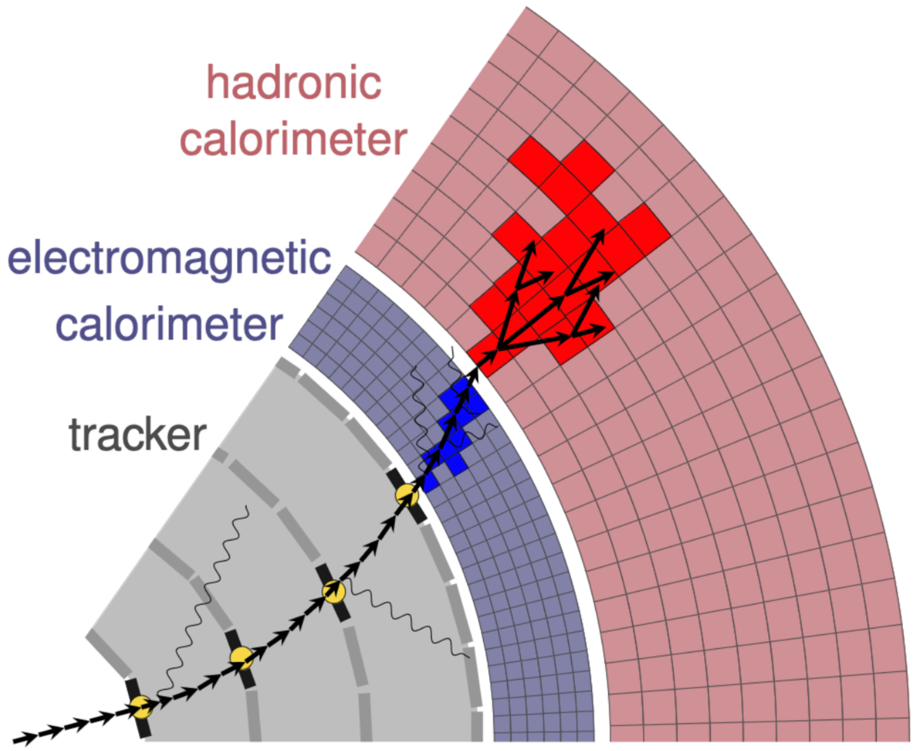
Integrated Simulation Framework



- AtlFast3 embedded in the Integrated Simulation Framework (ISF) to combine multiple simulators within Athena
- ISF is flexible, but 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
- Geant4 allows to directly integrate fast simulation tools
- **Idea:**
  1. Implement FastCaloSimV2 as a Geant4 fast simulation model **(VALIDATED!)**
  2. Evolve FastCaloSimV2 into a fully experiment-independent external library **(TO-DO!)**
  3. Fully deprecate ISF as a particle-stack dispatcher in ATLAS **(TO-DO!)**

First step towards a **simple** and **streamlined** ISF-independent ATLAS simulation!

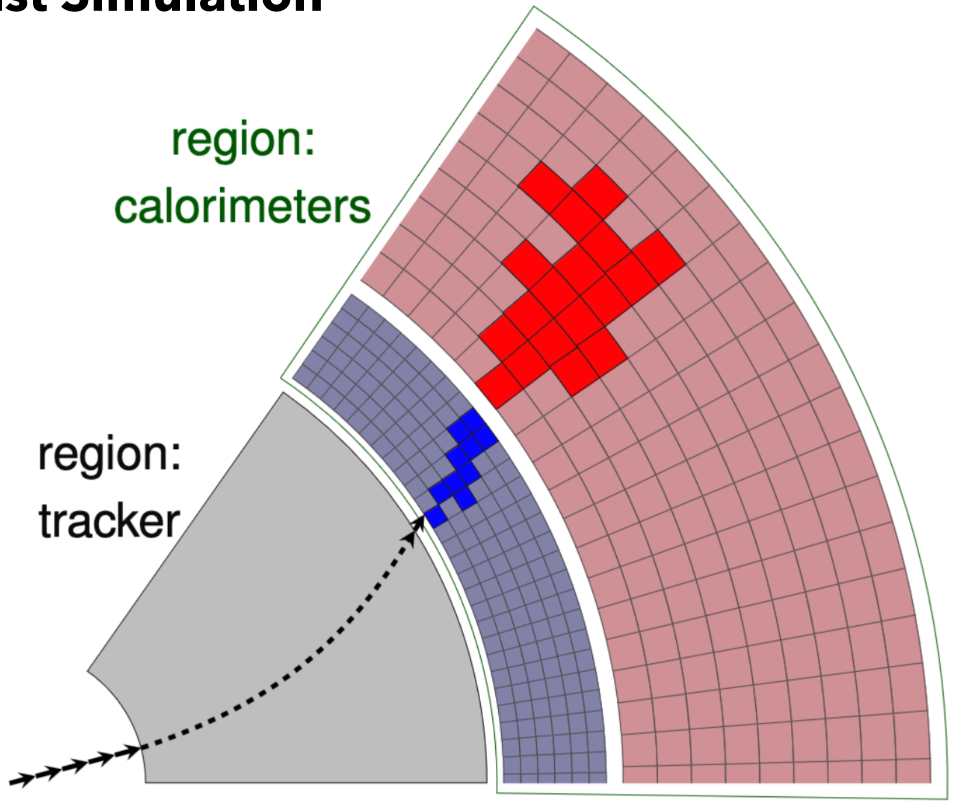
Full Simulation



- detailed detector description
- definitions of particles and processes
- transport in EM field

Fast Simulation

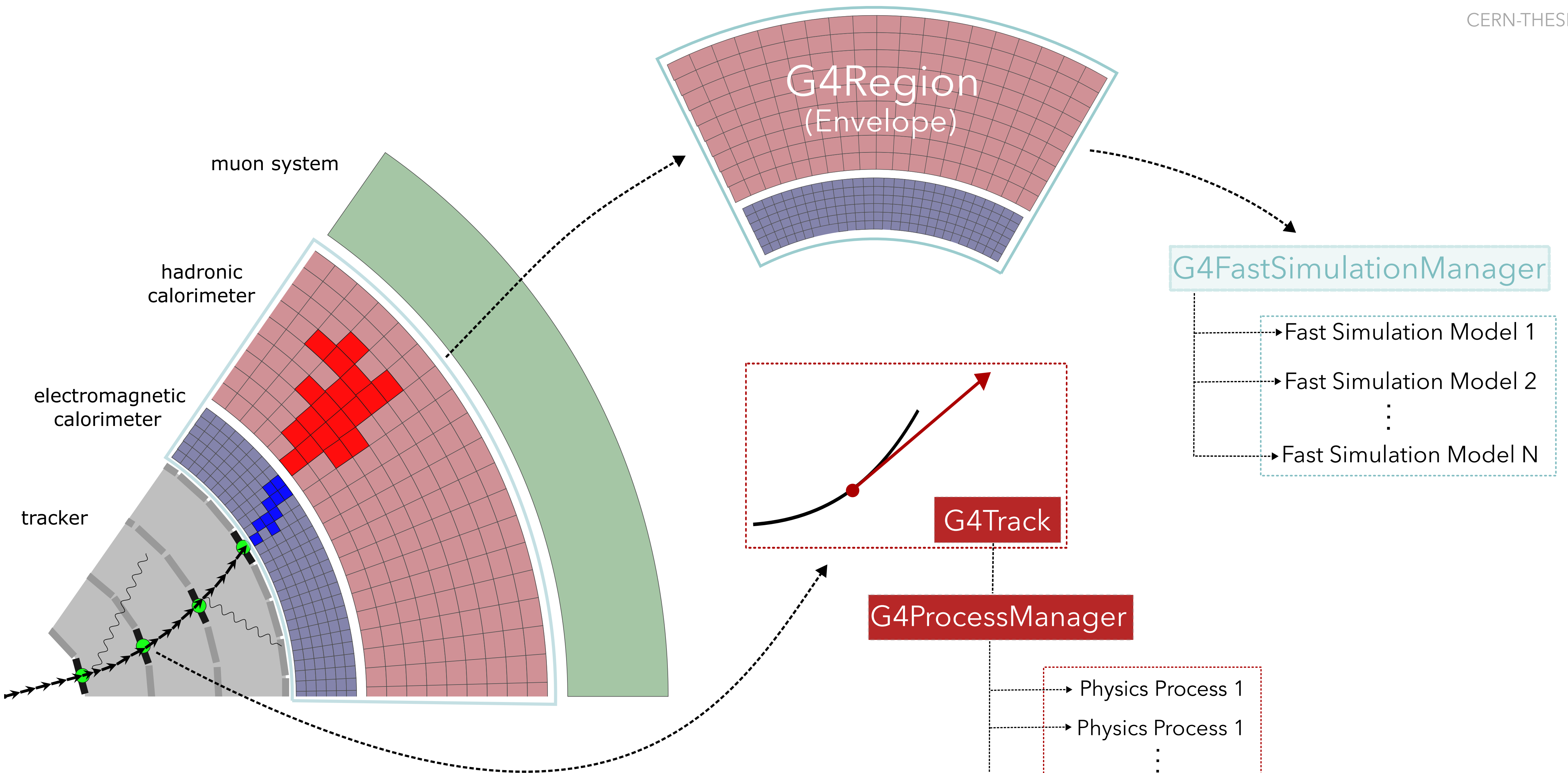
J. Phys.: Conf. Ser. 898 042053



- **where** particles are parametrised
- **which** particles are parametrised
- **what** happens instead of full simulation?

FastCaloSim in Geant4: Fast Simulation Models

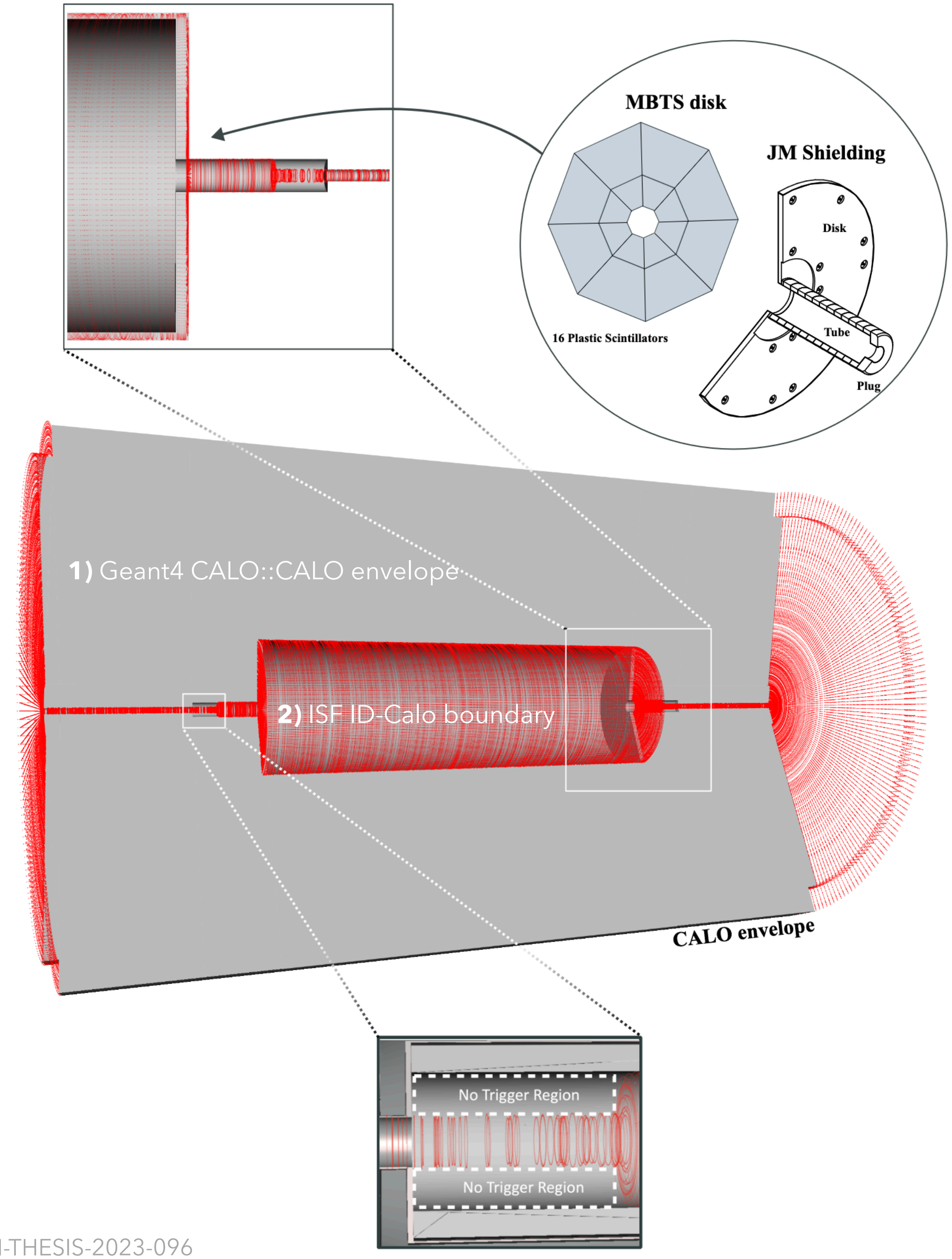
CERN-THESIS-2023-096



1. Attach fast simulation model to specific detector envelopes

2. If particle reaches envelope, invoke fast simulation and kill particle

Step I: Choice of Trigger Volumes



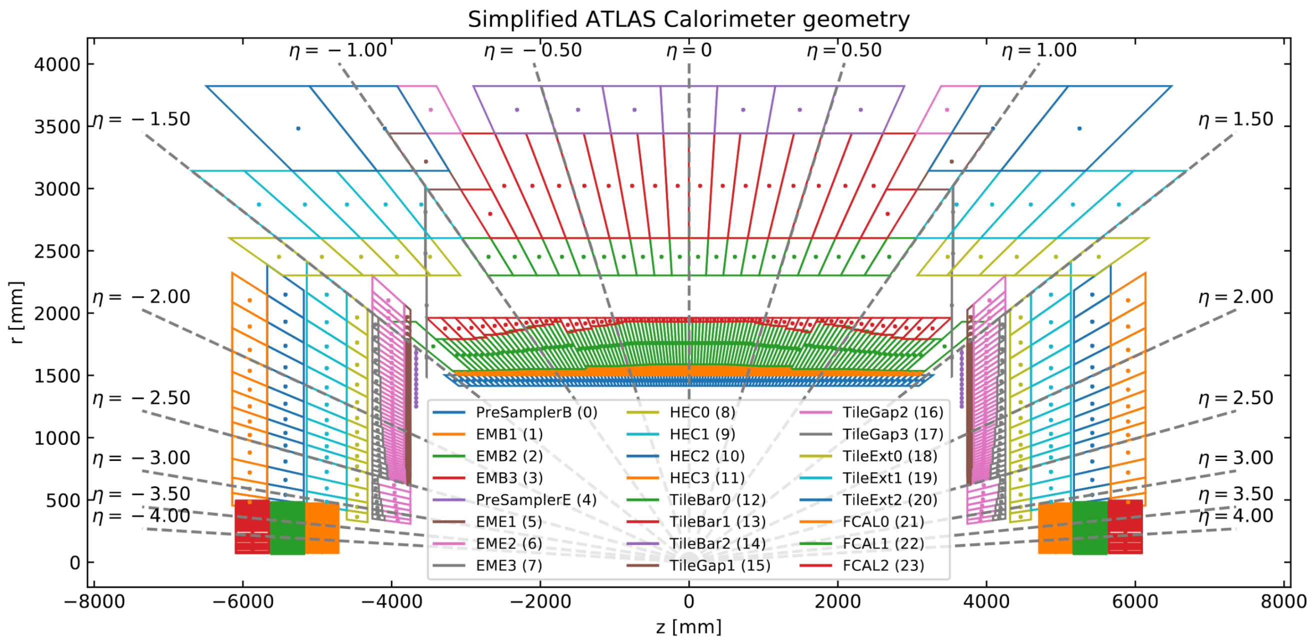
- **Initial implementation goal:** unaltered physics with respect to ISF implementation
- **Challenge**: Geant4 CALO envelope ≠ ISF ID-Calo boundary
  → choice of trigger volumes not trivial
- **Temporary solution:** minimal set of Geant4 volumes covering ISF boundary + conditional checks to ensure that fast simulation is ONLY invoked at ISF boundary

Easily avoidable for Run 4 and beyond by ensuring **trigger boundary = parametrisation boundary**

CERN-THESIS-2023-096

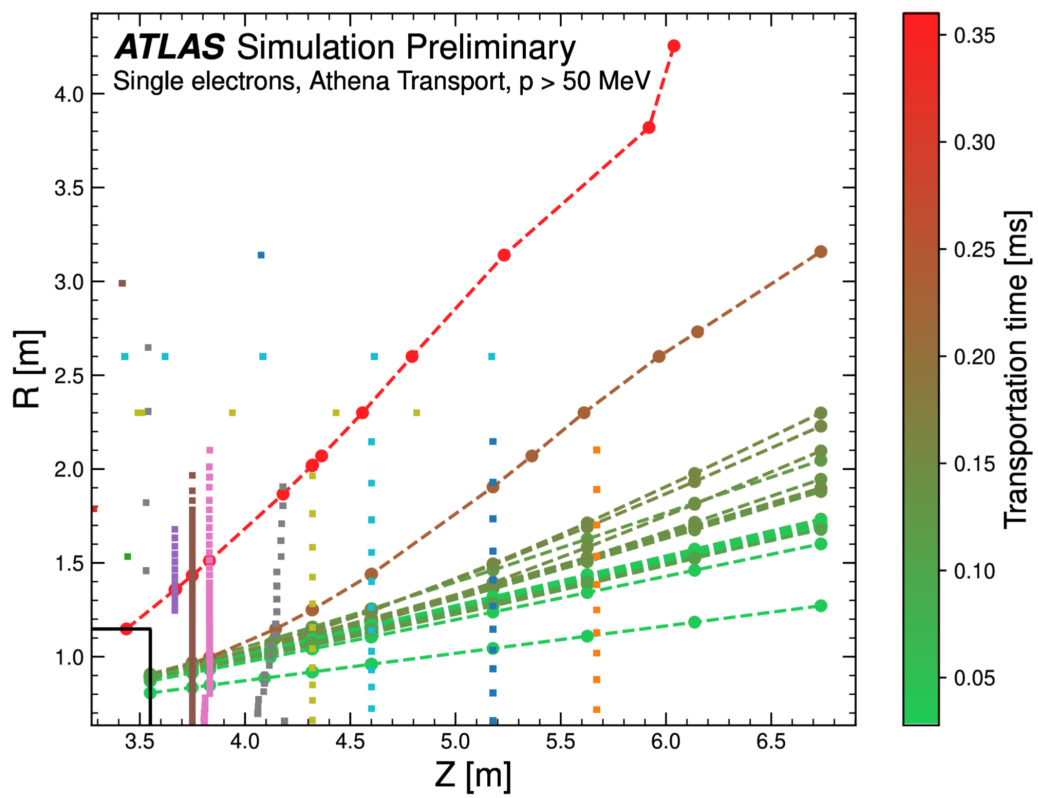
Step II: Replacement of ATLAS Track Transport

EPJ Web Conf., 245 (2020) 10003



- AtlFast3 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

SIM-2024-004



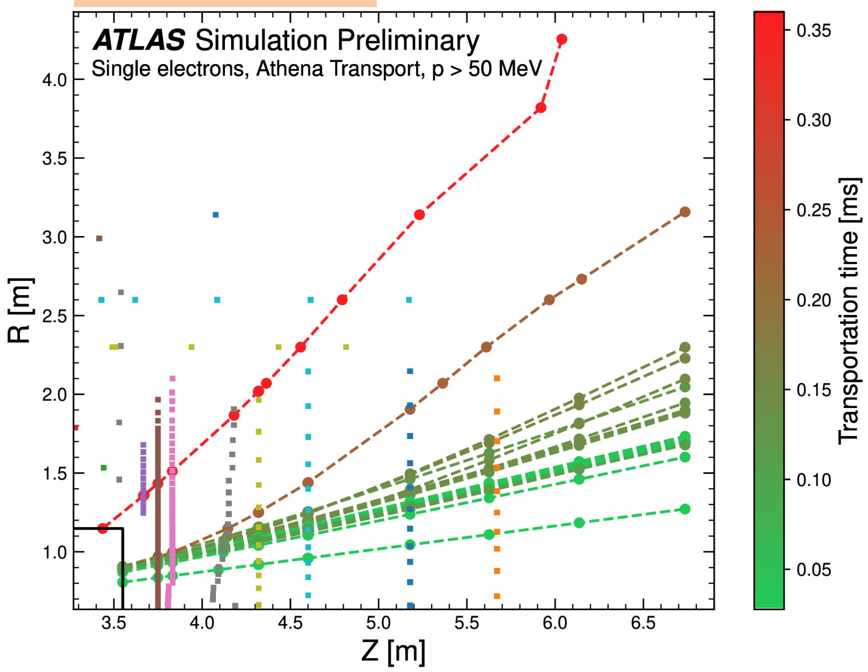
- AtlFast3 uses proprietary **Athena tracking tools** to transport particles
- Intersections with active calorimeter layers given as input to experiment-independent extrapolation algorithm

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

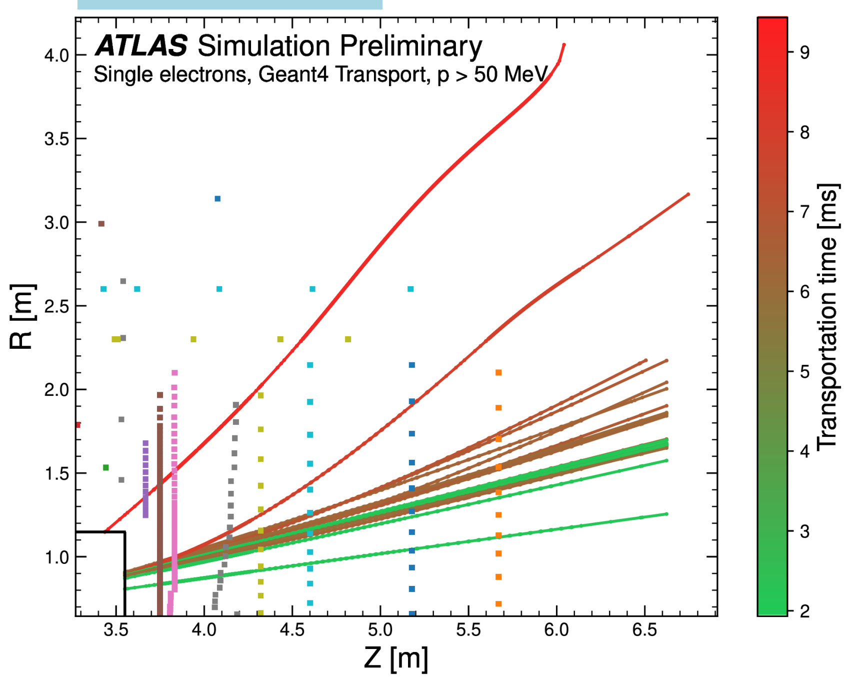
Step II: Replacement of ATLAS Track Transport

SIM-2024-004

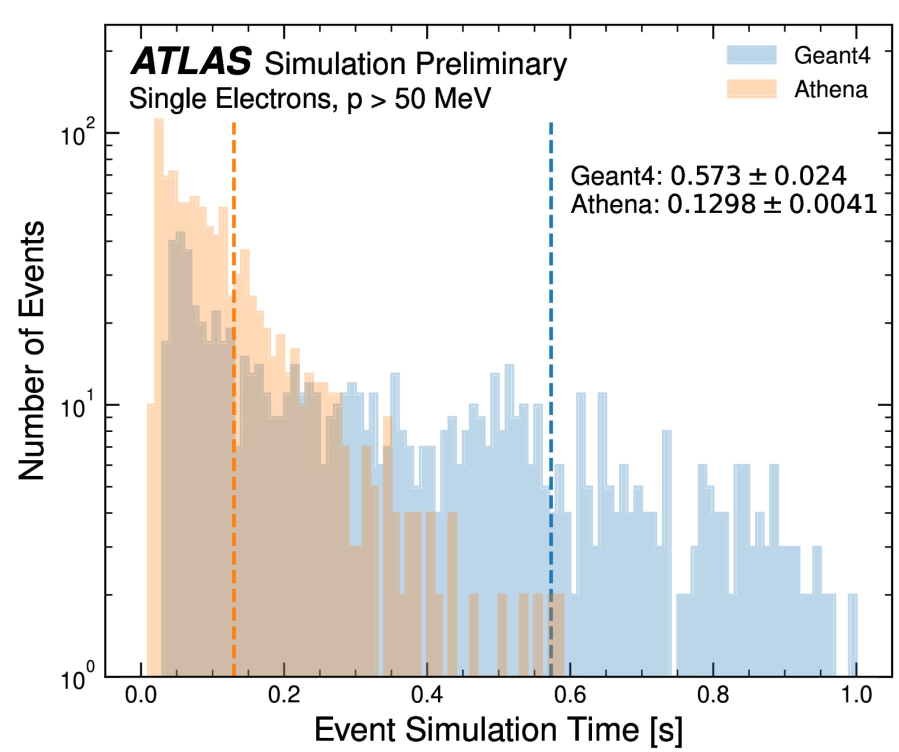
Athena Transport

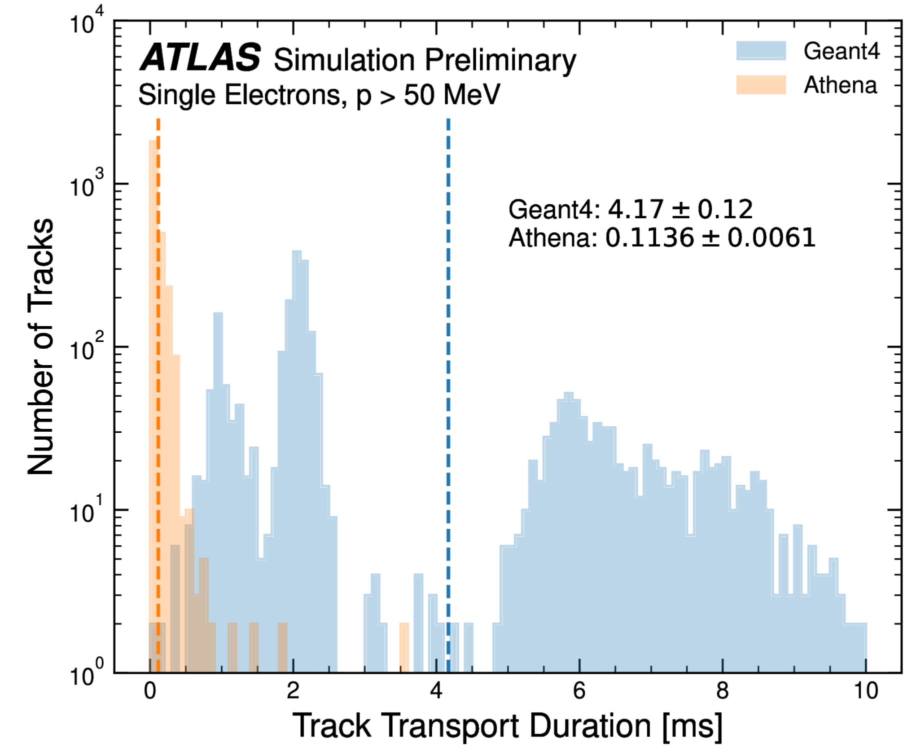


Geant4 Transport



First attempt: Exploit default Geant4 engine for track transportation





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

Transport duration per track:
x 34 slower with Geant4 transport

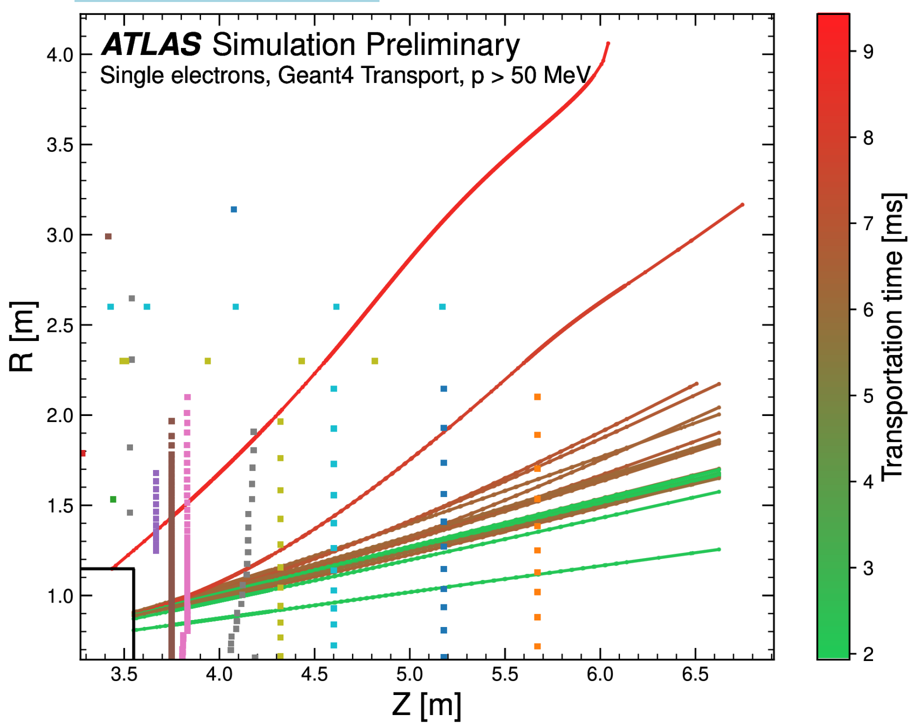
First very naive approach of Geant4 transport **prohibitively slow**

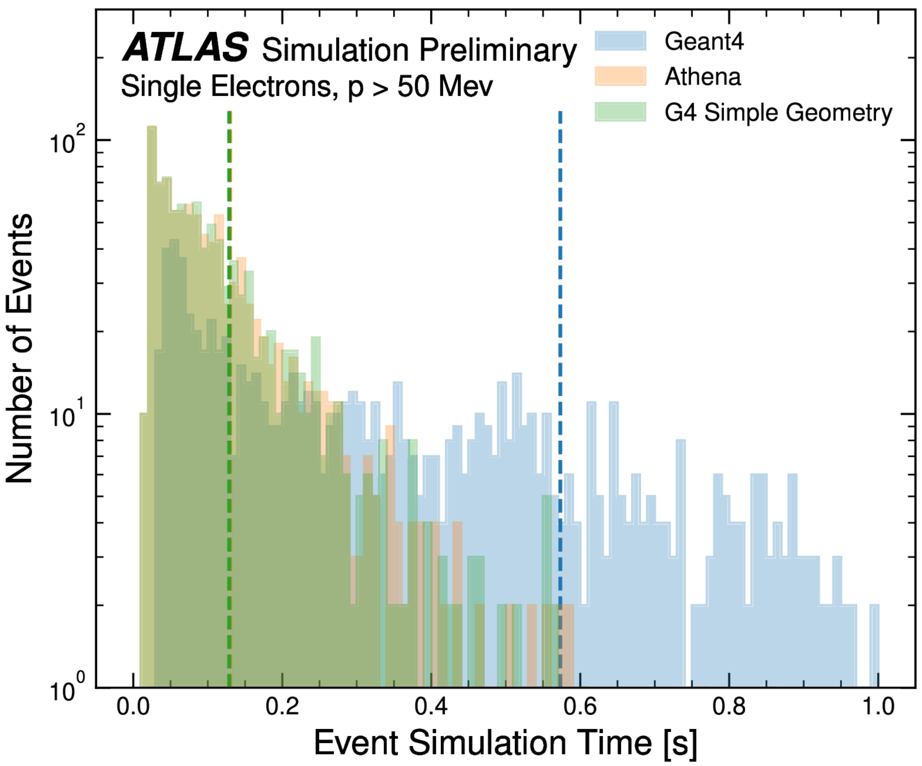
How can we do better?

Step II: Replacement of ATLAS Track Transport

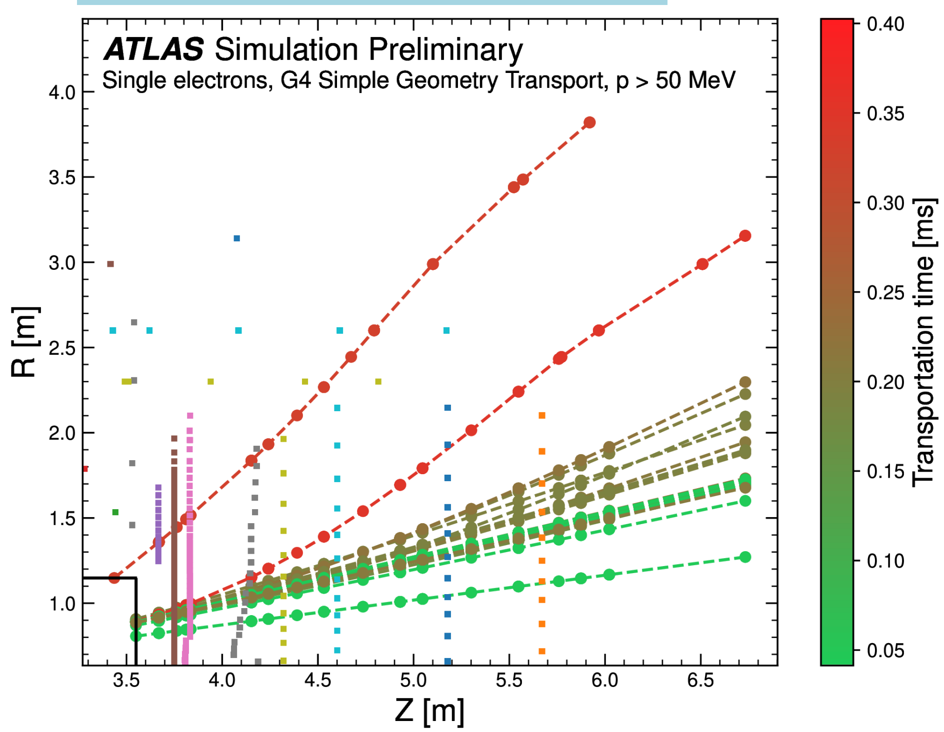
SIM-2024-004

Geant4 Transport





Geant4 Transport in simplified geometry

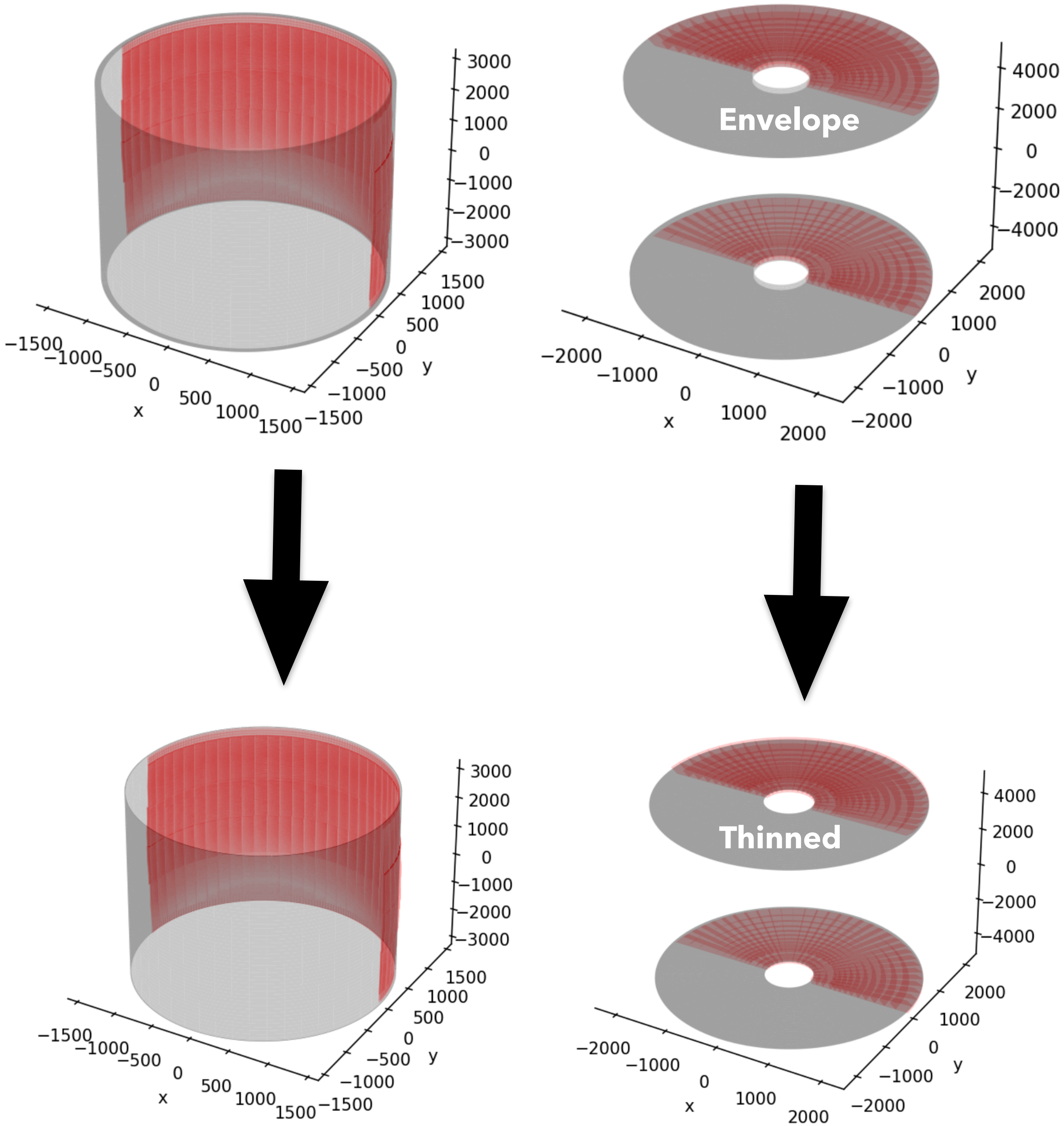


- Interested only in track position at entry and exit of each calorimeter layer
- Instead of transporting through full calorimeter geometry, do navigation in simplified (layer-based) geometry

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

How to construct simplified geometry?

Simplified Geometry: Construction



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):

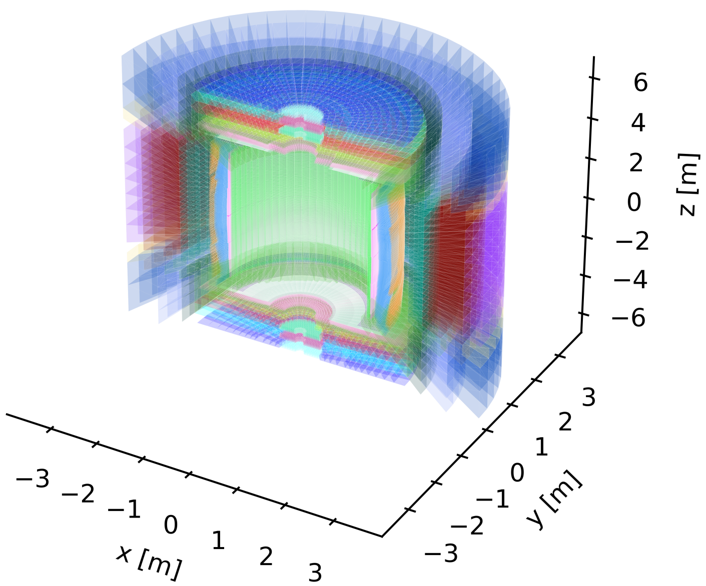
1. Model all layers as cylindrical hulls (='envelopes') from the maximum geometric extension of the cells

2. *Thin down* hulls to generate clash-free geometry
→ for barrel layers small $r = r^{\text{hull}}_{\text{mid}}$
→ for endcap layers small $z = z^{\text{hull}}_{\text{mid}}$

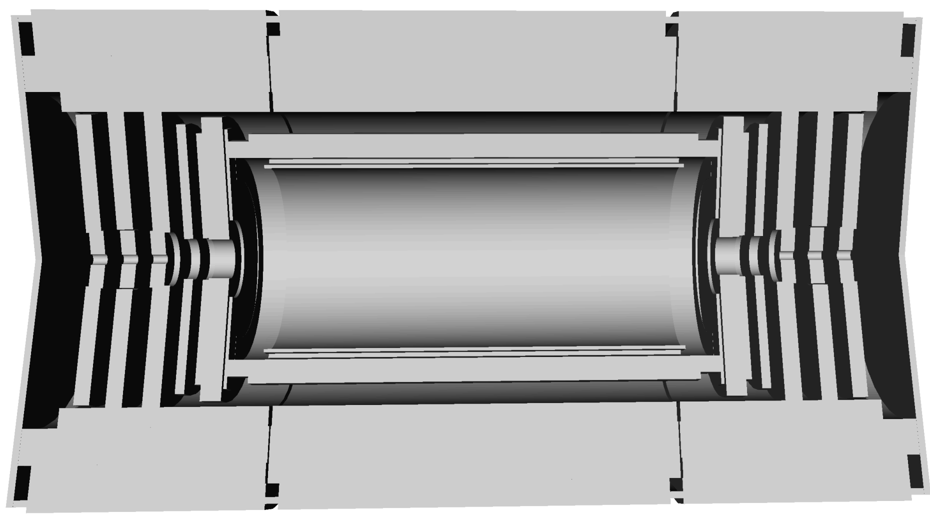
3. Attempt to grow back thinned down layers to original size of envelopes (constraint: only grow up to the limiting layer, i.e. w/o creating overlaps)

Simplified Geometry: Construction

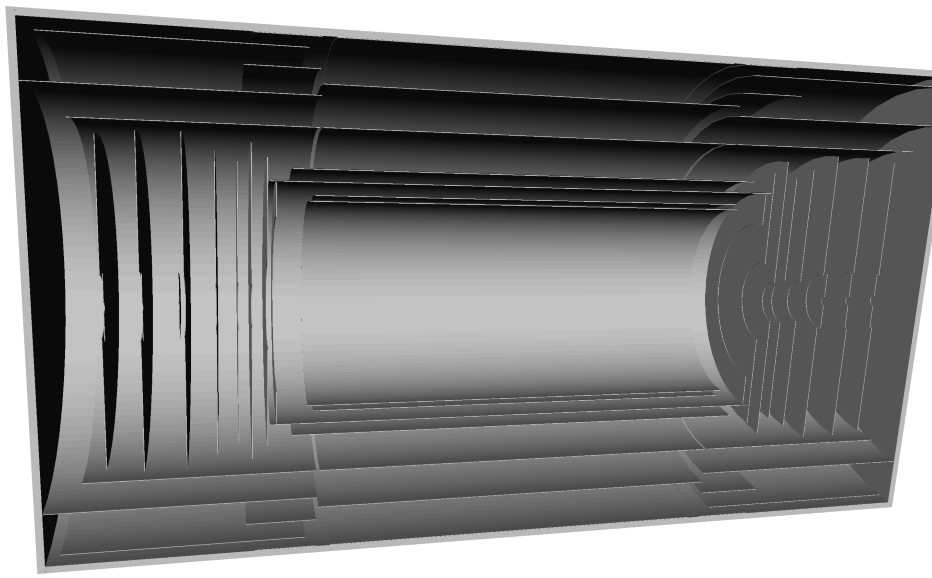
1) Calorimeter Cells



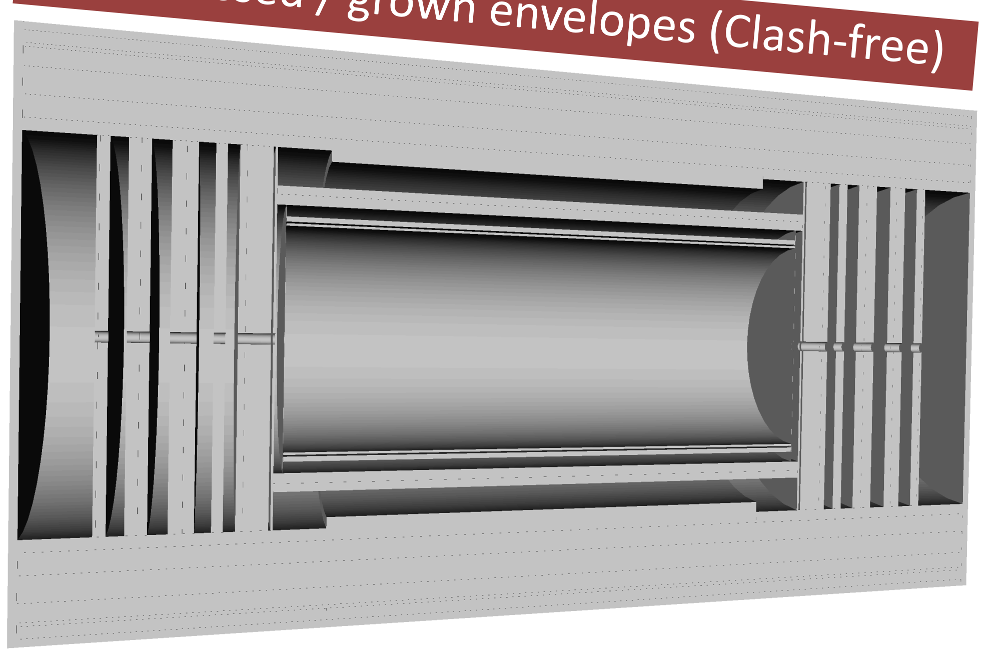
2) Cell Envelopes (**NOT** clash-free)



3) Thinned envelopes (Clash-free)

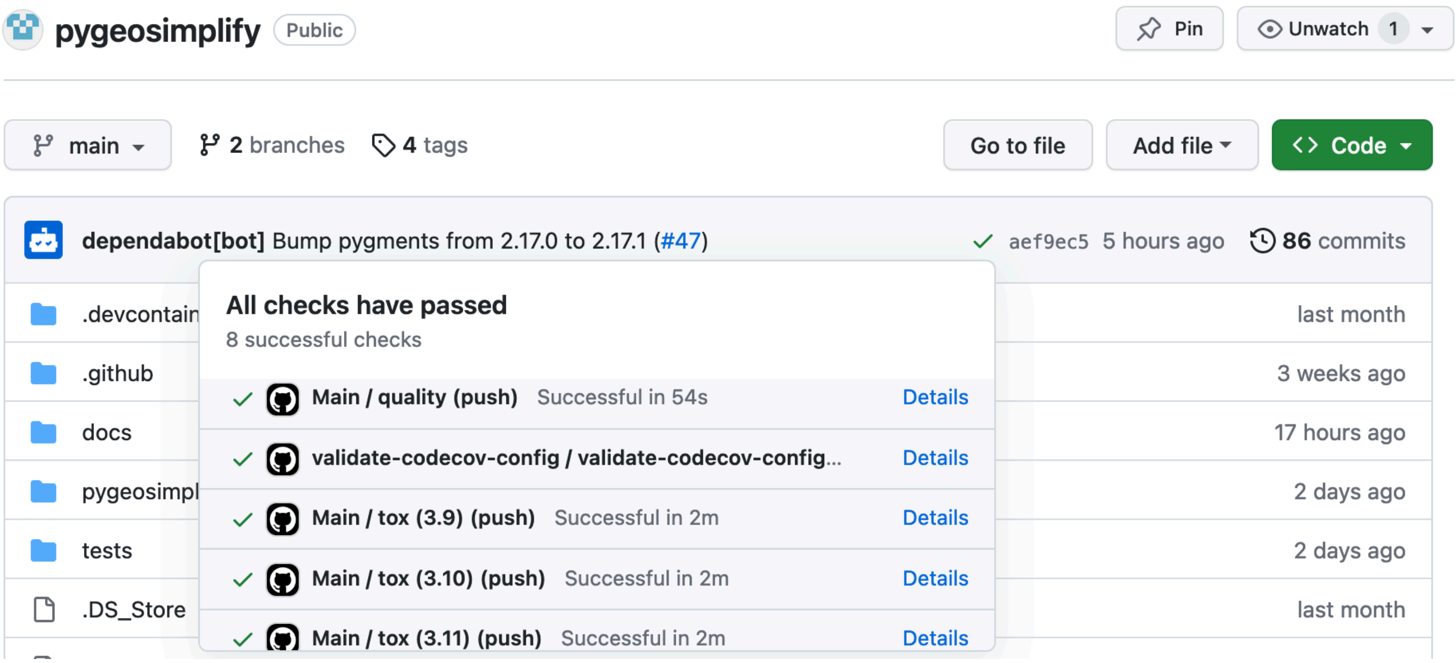


4) Processed / grown envelopes (Clash-free)



```
**** 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
```

Python package: pyGeoSimplify



- 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`

Project description

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

Welcome to pyGeoSimplify!

Download pyGeoSimplify

```
pip install pygeosimplify
```

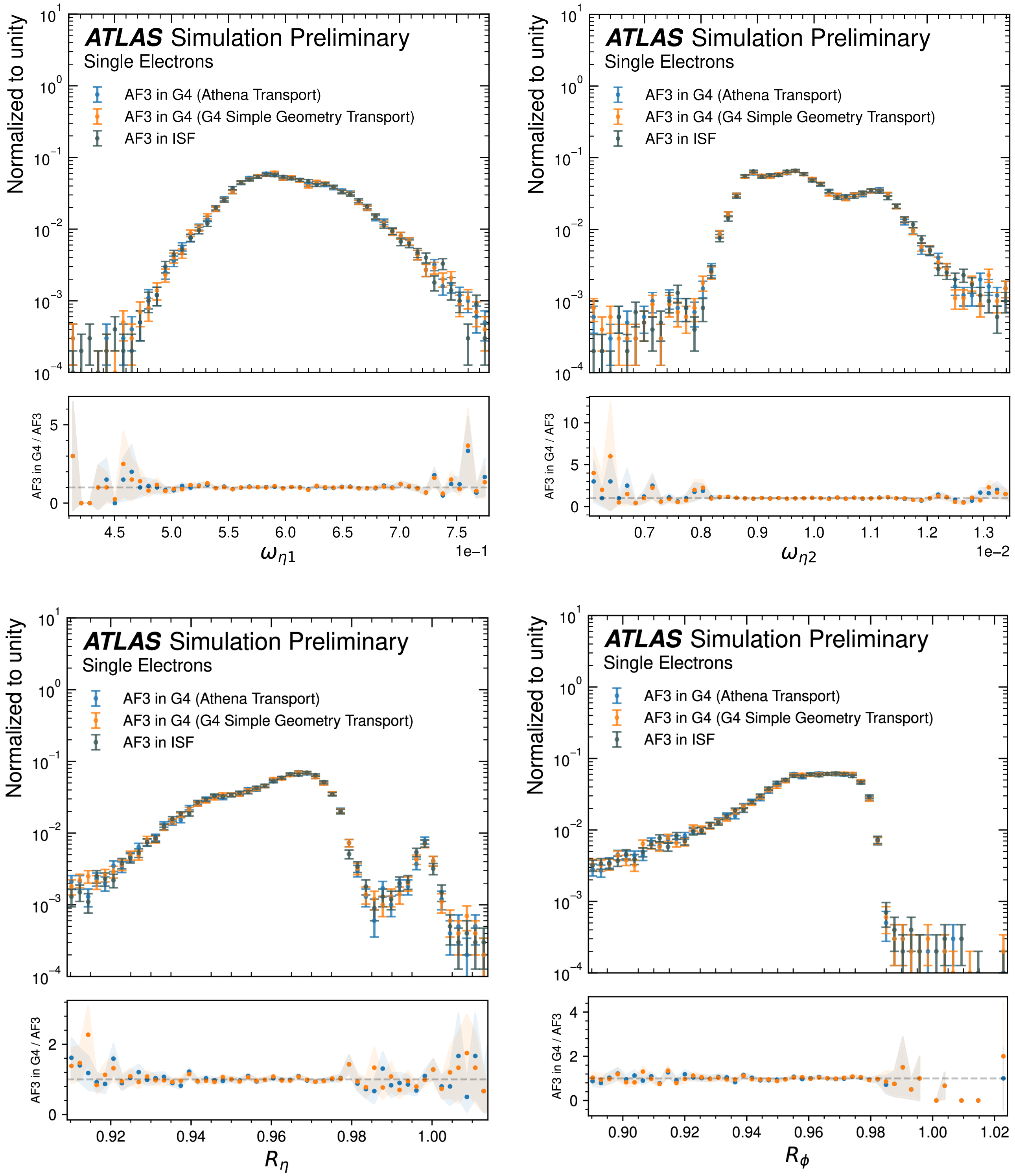
Quick Start

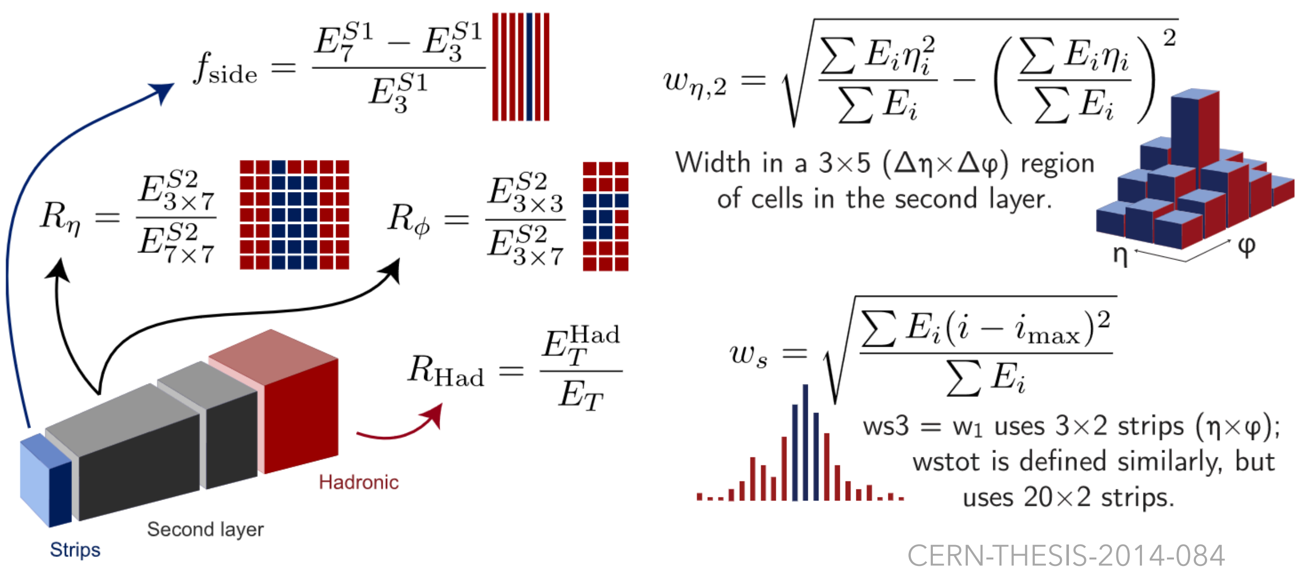
```
import pygeosimplify as pgs
from pygeosimplify.simplify.layer import GeoLayer
from pygeosimplify.simplify.detector import SimplifiedDetector

# 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 dector 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')
```

Validation: e/γ

Single Electron Shower Shapes

SIM-2024-004





CERN-THESIS-2014-084

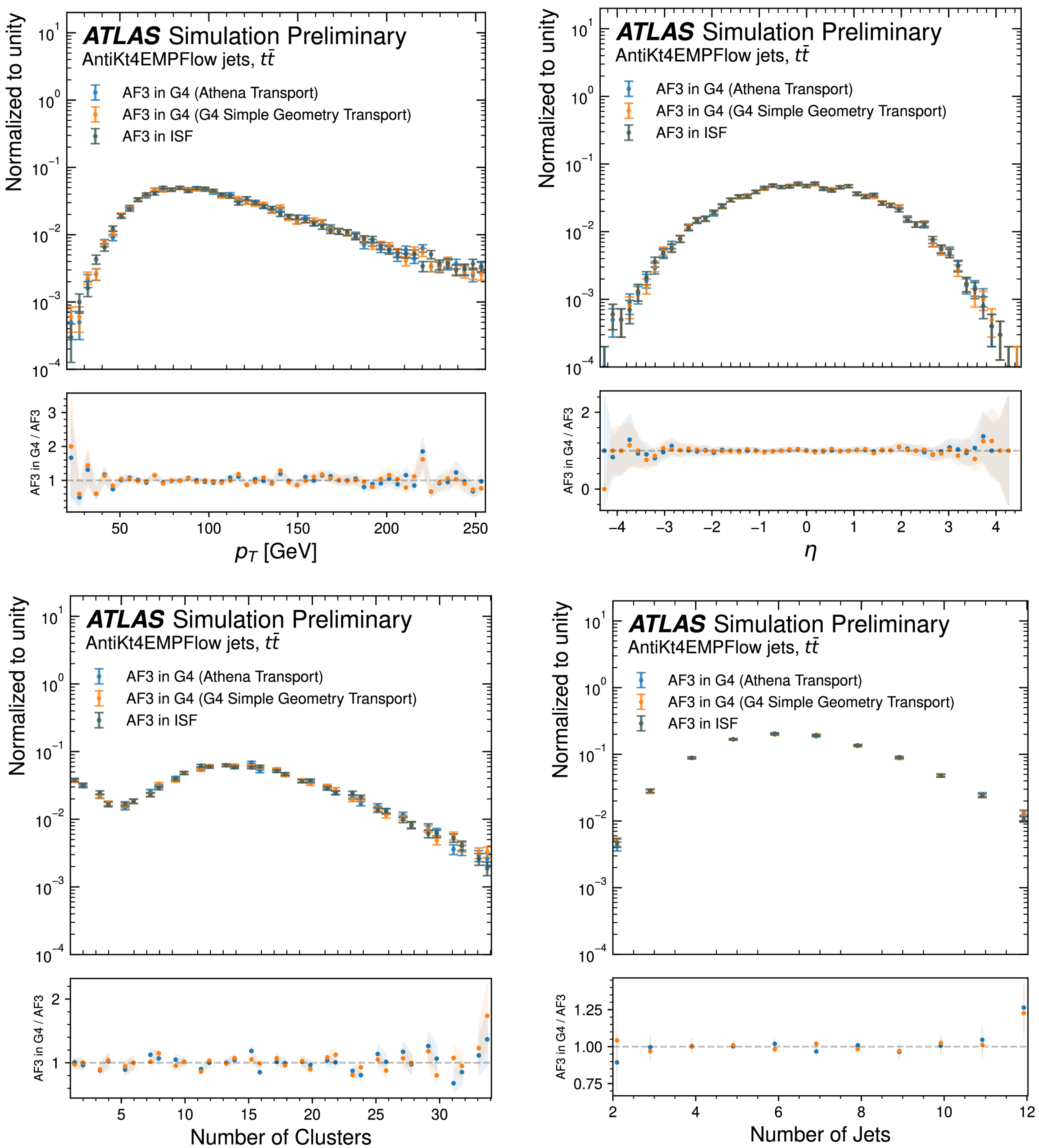
- Electrons and photons in ATLAS reconstructed based on topological calorimeter clusters
- Prompt e/γ identification heavily relies on calorimetric shower shape variables
- Shown here: reconstructed single electrons

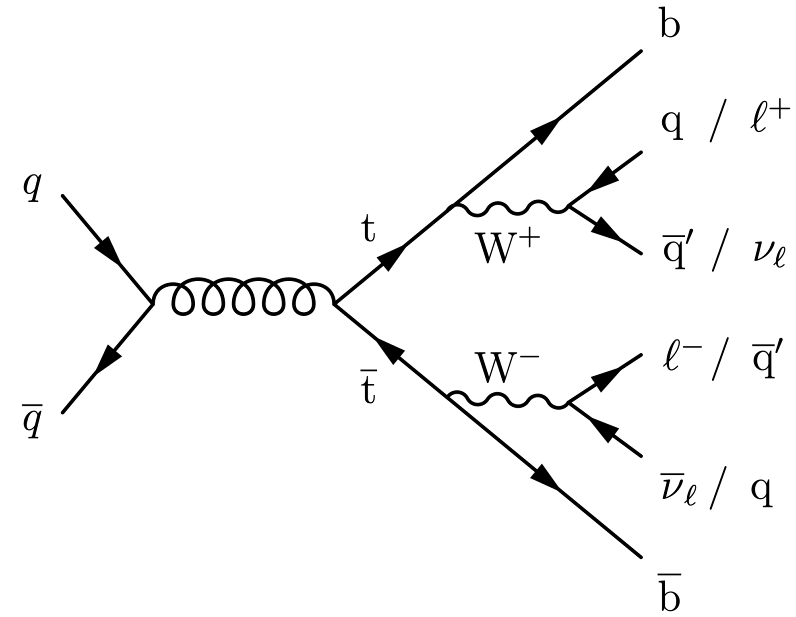
No significant differences observed in shower shapes between ISF and Geant4 implementation

Validation: Jets

AntiKT4EMPFlow Jets from $t\bar{t}$ sample

SIM-2024-004





- Jets reconstructed using topological clusters and tracks from the Inner Detector
- Shown here: Reconstructed top jets from $t\bar{t}$ events

No significant differences observed in jet observables between ISF and Geant4 implementation

Summary & Outlook

Summary

- ATLAS employs AtlFast3 for the simulation of billions of MC events to leverage CPU resources
- First implementation of the AtlFast3 service as Geant4 fast simulation model in ATLAS
- Replaced Athena tracking tools (heaviest ISF coupling) with Geant4 track propagation in simplified geometry
- Python-package for **experiment-independent automatic cell-based inference of simplified detector geometry** · might also be interesting for other use cases, e.g. Fast ATLAS Track Simulation (FATRAS)
- Validation of Geant4 fast simulation model shows **no significant differences relative to usage of default ISF implementation** for reconstructed objects

Outlook

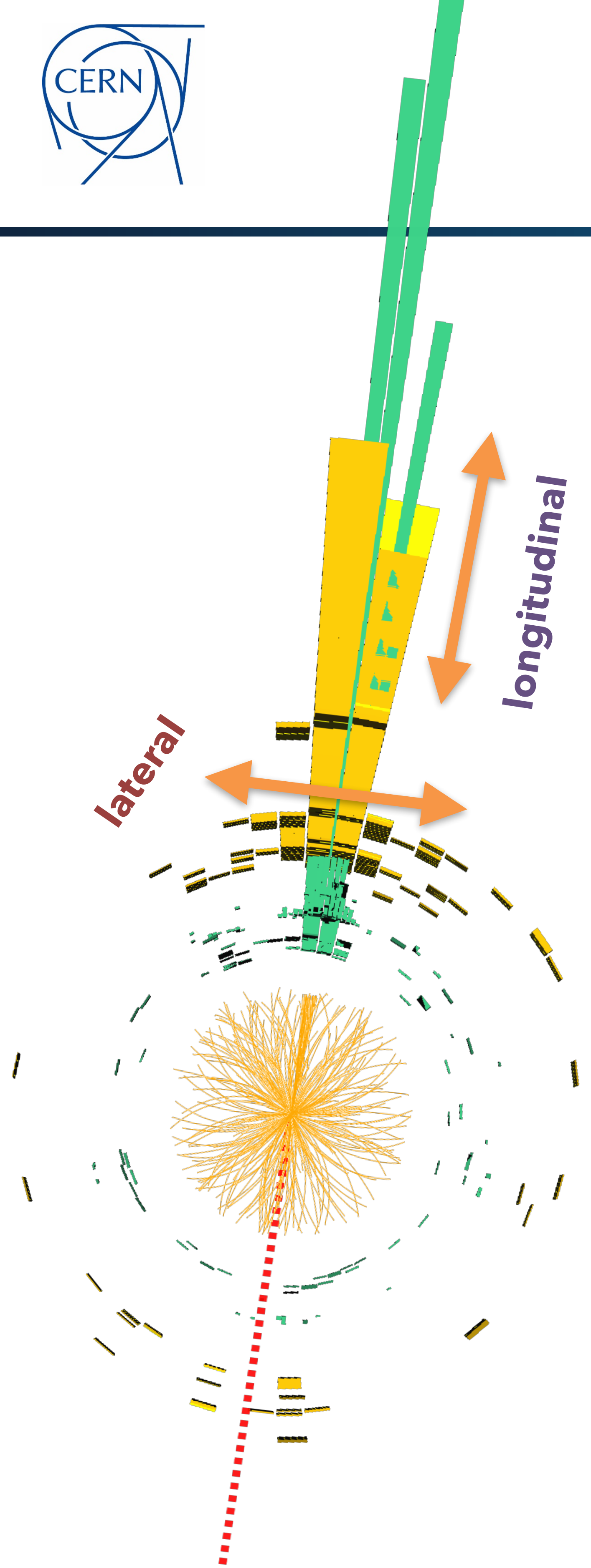
- Most of the FastCaloSim code already compiles standalone, but still has various ATLAS dependencies that need to be replaced with experiment-independent implementations
- Goal is to have single external library that can be hooked to Geant4 fast simulation model
- In addition to ATLAS, use Open Data Detector (ODD) as proof-of-concept for experiment-independent implementation

Thank You

BACKUP

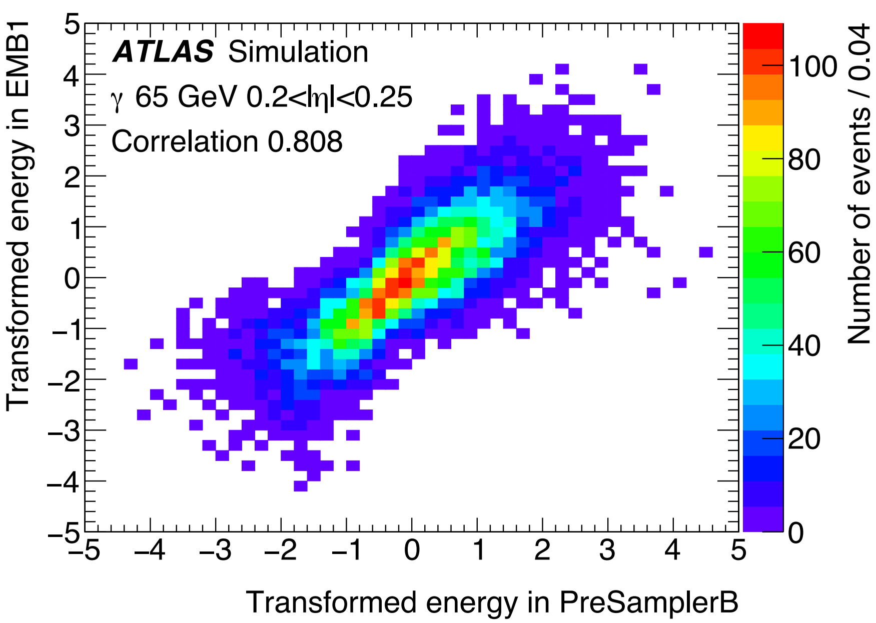
FastCaloSimV2: Energy Profile

Separate parametrisation in longitudinal and lateral component



- **What is total energy and how is energy shared between layers?**
  - → **longitudinal** energy profile
  - → crucial for energy measurements

- **How is energy distributed within layers?**
  - → **lateral** energy profile
  - → crucial for particle identification



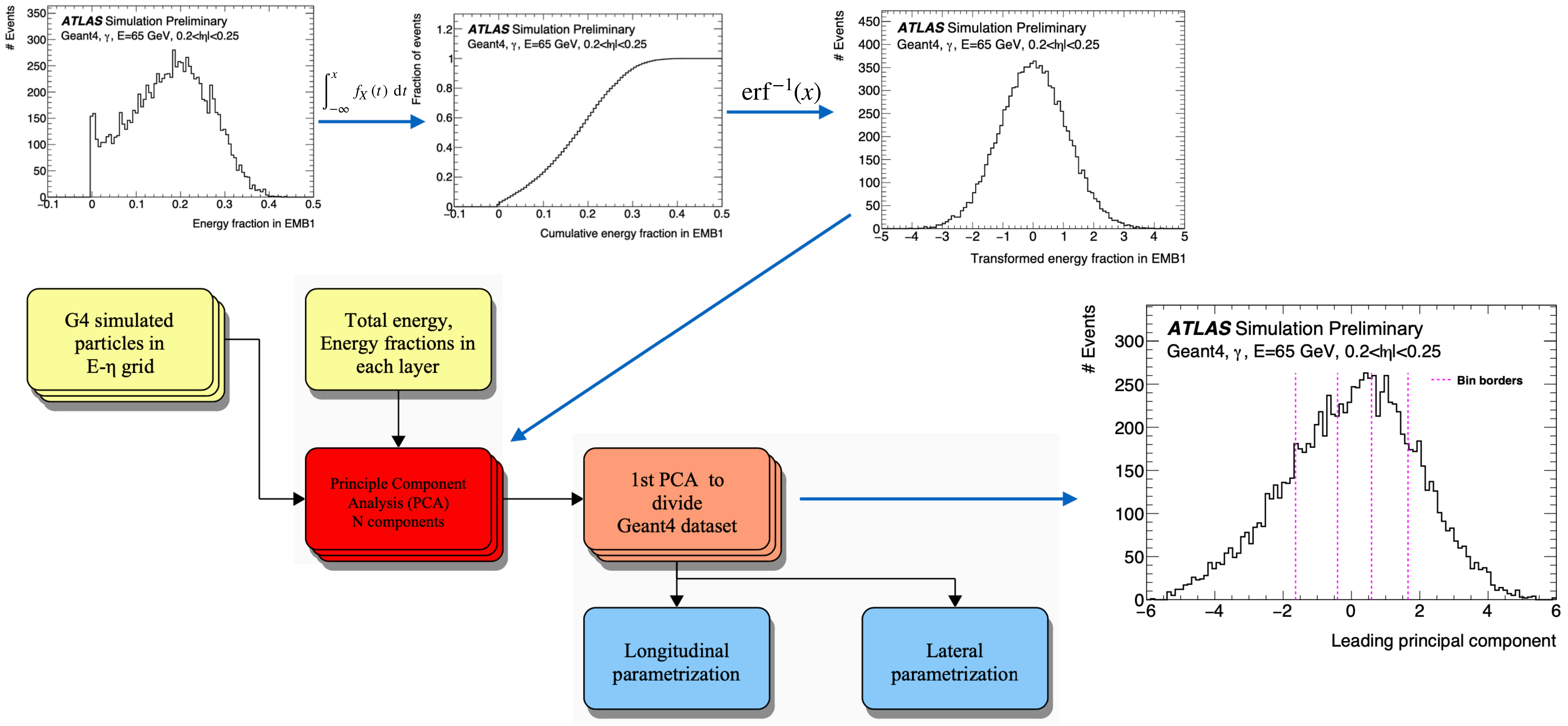
Problem:

- Energy depositions across layers highly correlated
  - → difficult to model energy response in each layer independently

Strategy:

- Decorrelate energy depositions in layers using Principal Component Analysis (PCA)

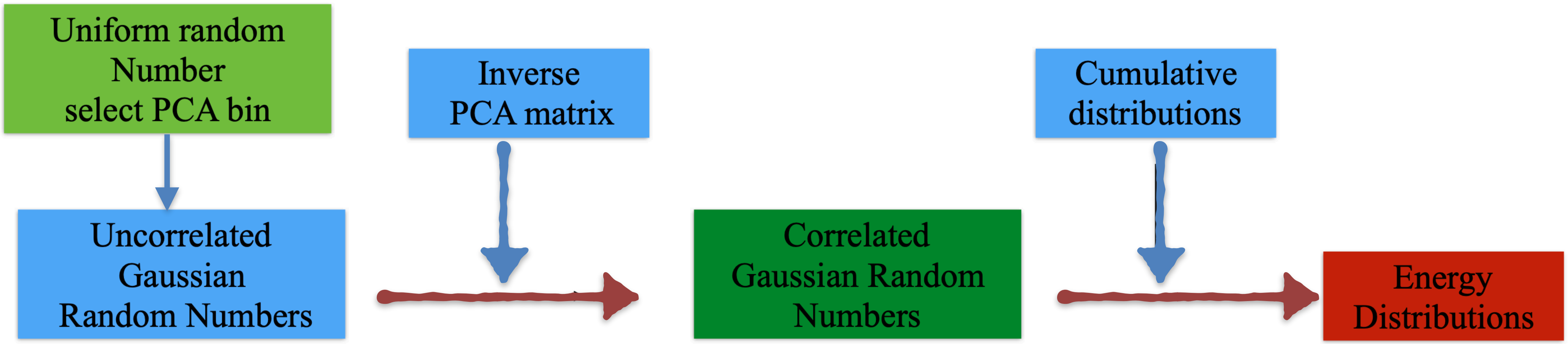
FastCaloSimV2: Longitudinal Profile



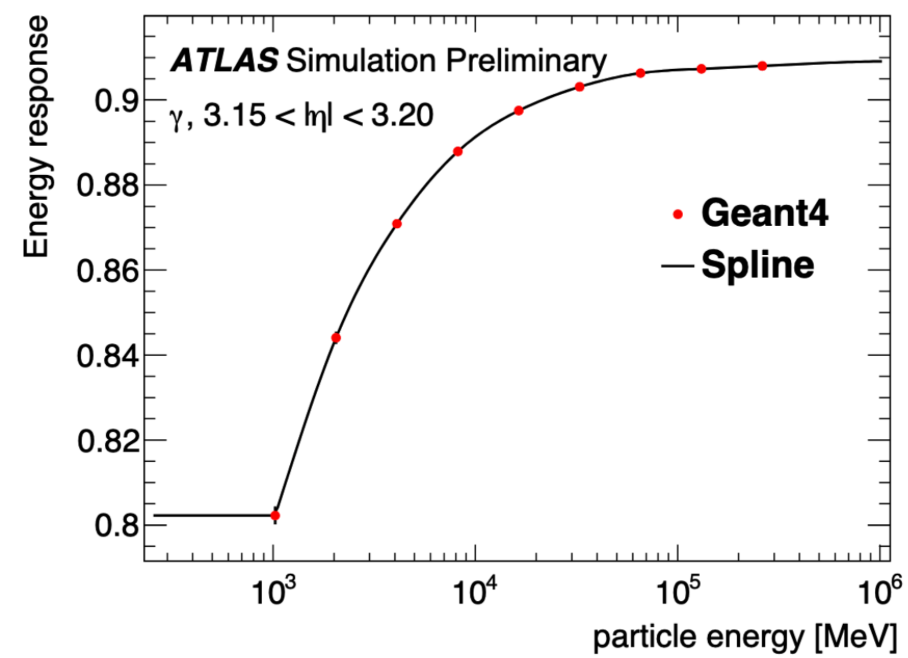
Store cumulative energy fractions, mean and RMS of gaussians and covariance matrix

FastCaloSimV2: Longitudinal Profile

During simulation, chain is performed backwards:

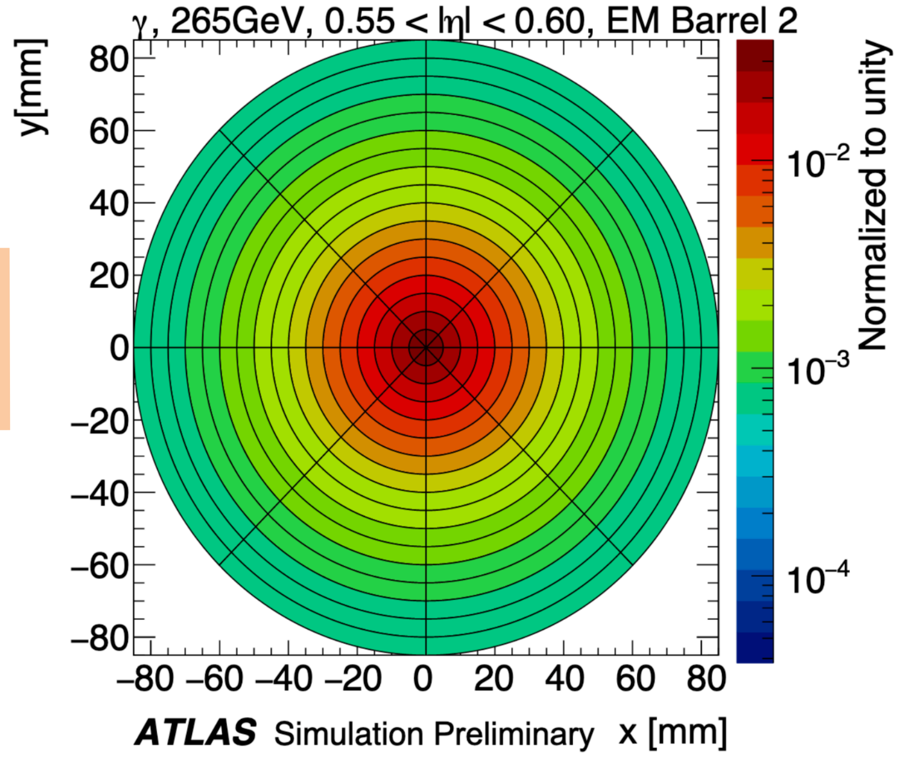


Interpolate energy response using splines:

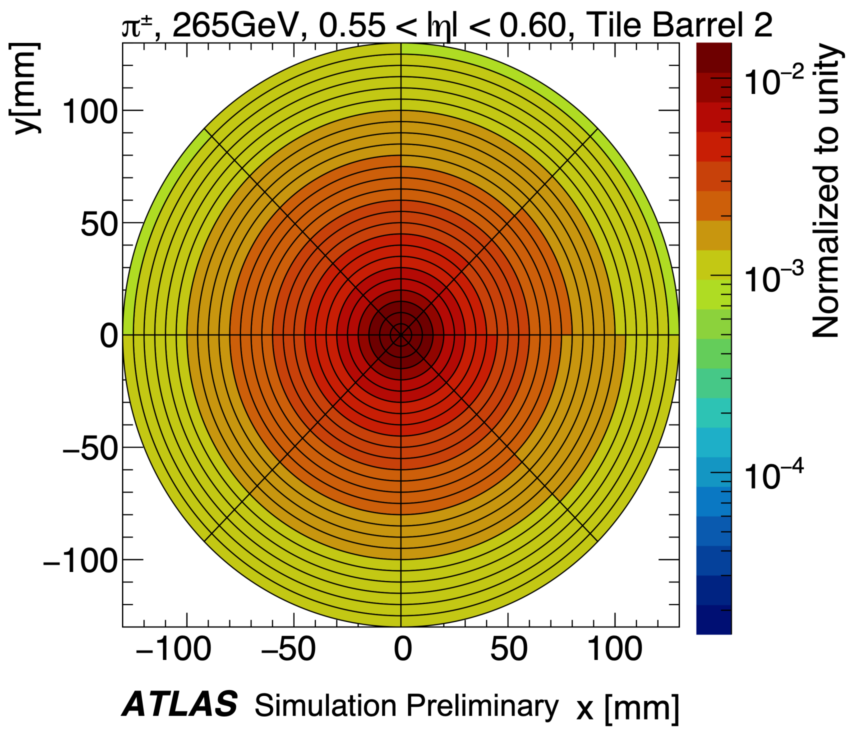


FastCaloSimV2: Lateral Profile

Photons



Pions



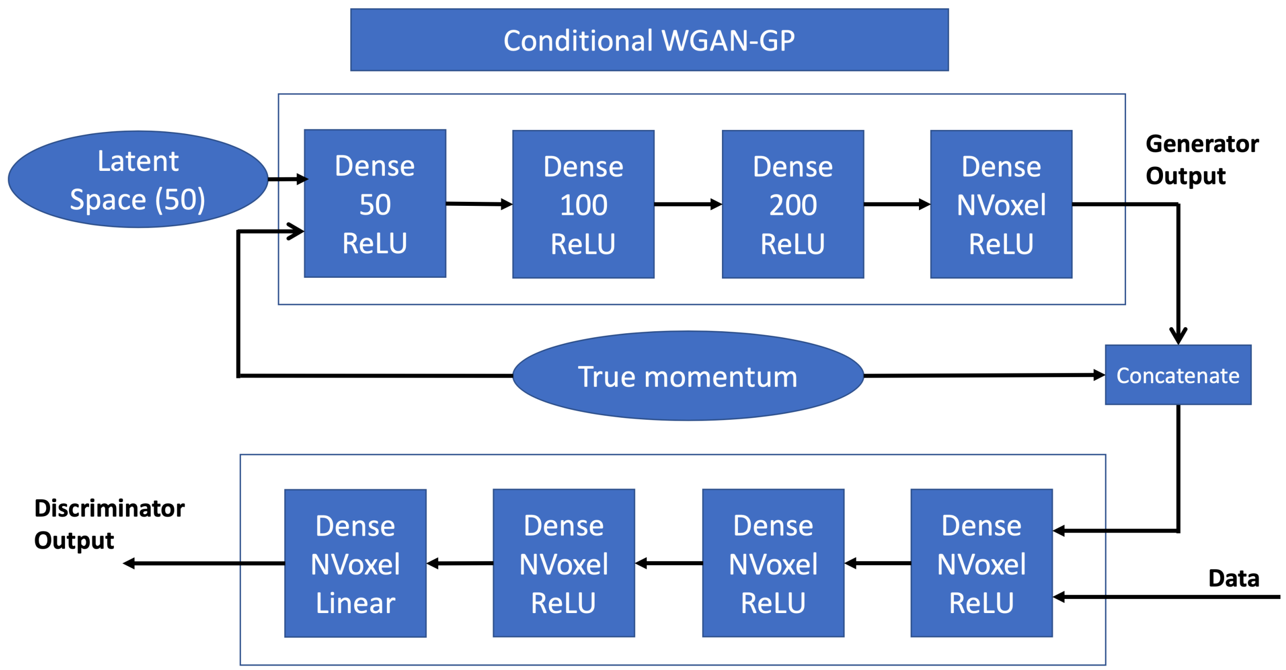
- Parametrise average energy distribution in lateral direction over radial distance containing 99.5% of total energy and 8-bins in angular direction
- Parametrisation for each particle, energy, eta, calorimeter layers and bins of 1st PCA
- During simulation, randomly sample quantised energy deposits from 2D shape histograms (PDFs)
- To each hit, assign hit energy

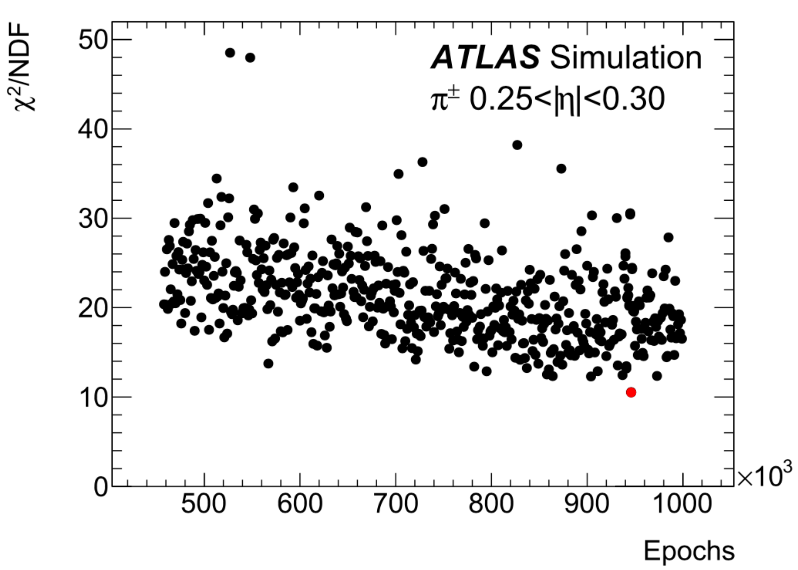
$$E_{\text{hit}} = \frac{E_{\text{layer}}}{N_{\text{hits}}^{\text{layer}}} \times w$$

Weight dependent on radial position of hit

FastCaloGAN: Overview

- FastCaloGAN based on **WGAN-GP** algorithm which offers more stable training compared to conventional GANs
- Electrons, photons and pions used to train the network
- **One GAN is trained for each of the 100 bins** in |η| (0 - 5.0)
- Total of 300 GANs to cover full detector region
- GAN **trained to reproduce voxels and energies in the layer as well as total energy** in one single step
- Each GAN trained for 1M epochs with a checkpoint saved every 1K epochs





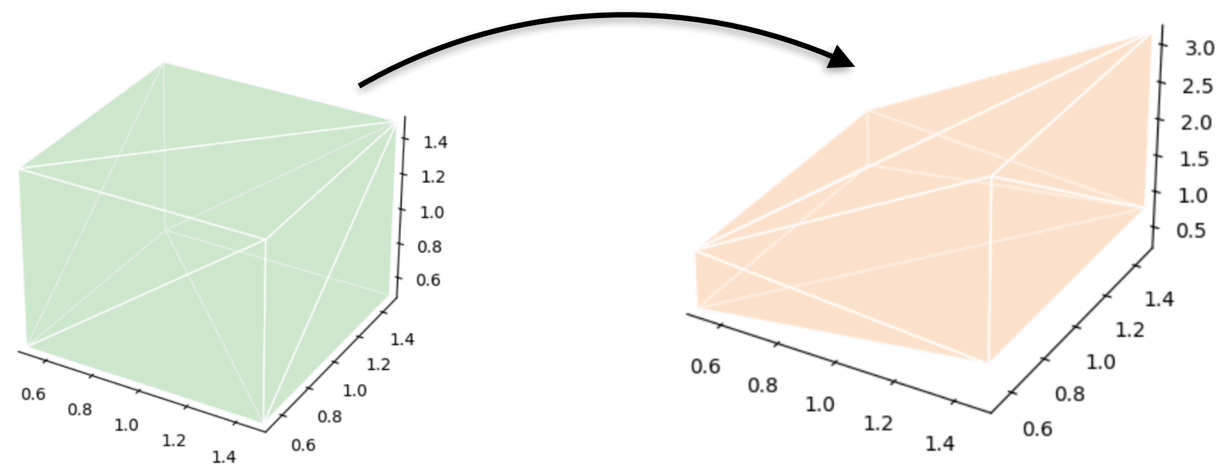
NVoxel	Number of voxels
Generator nodes	50, 50, 100, 200, NVoxel
Discriminator nodes	NVoxel, NVoxel, NVoxel, NVoxel, 1
Activation function	ReLU
Optimizer	Adam [50]
Learning rate	10^{-4}
$\beta 1$	0.5
$\beta 2$	0.999
Batch size	128
Training ratio (D/G)	5
Gradient penalty (λ)	10

WGAN-GP parameters

Simplified Geometry: Modelling Layer Hulls

How can we (in general) model cells to compute the layer hulls?

Coordinate transform

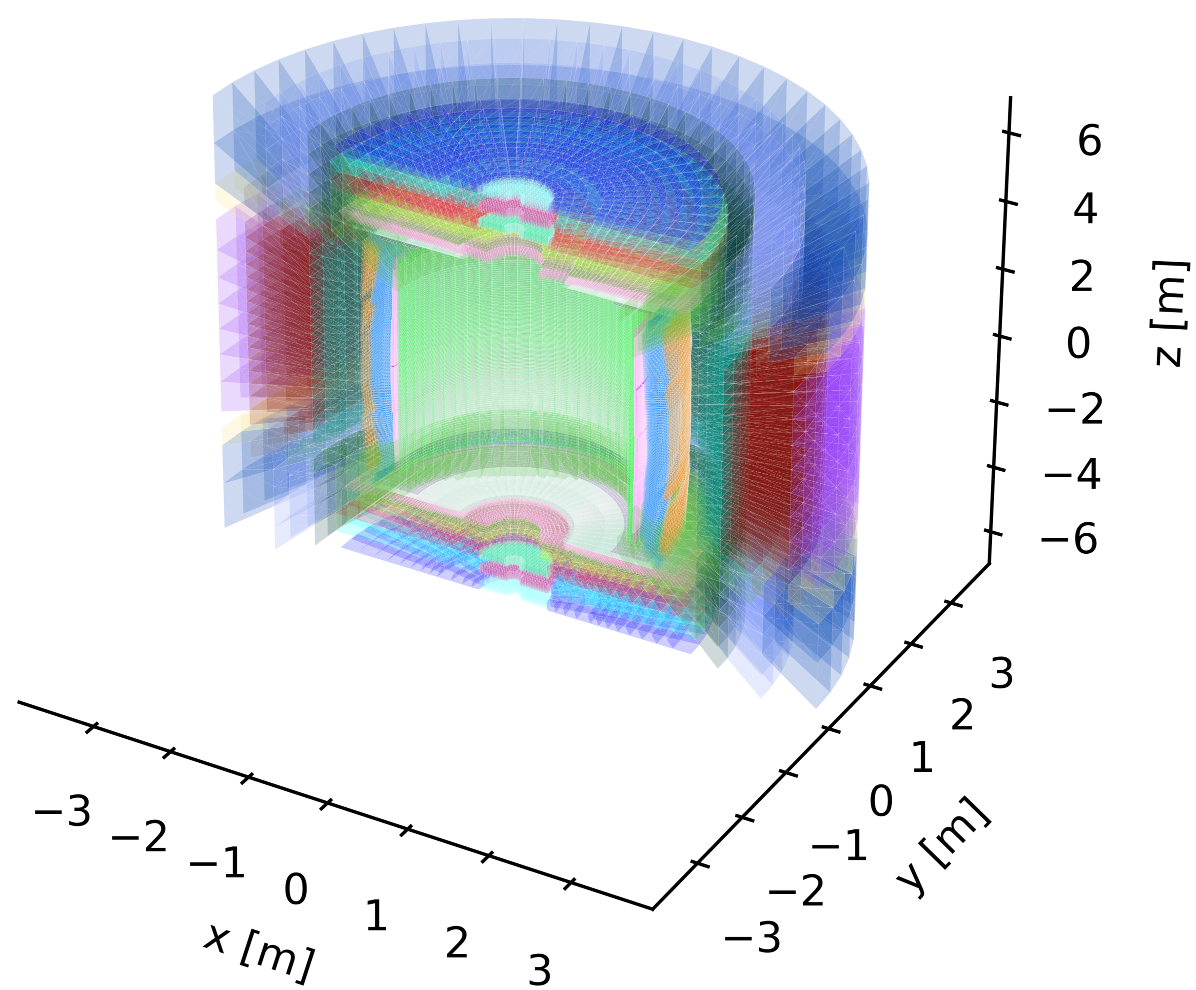


Cell defined by $\Delta r, \Delta\phi, \Delta z$

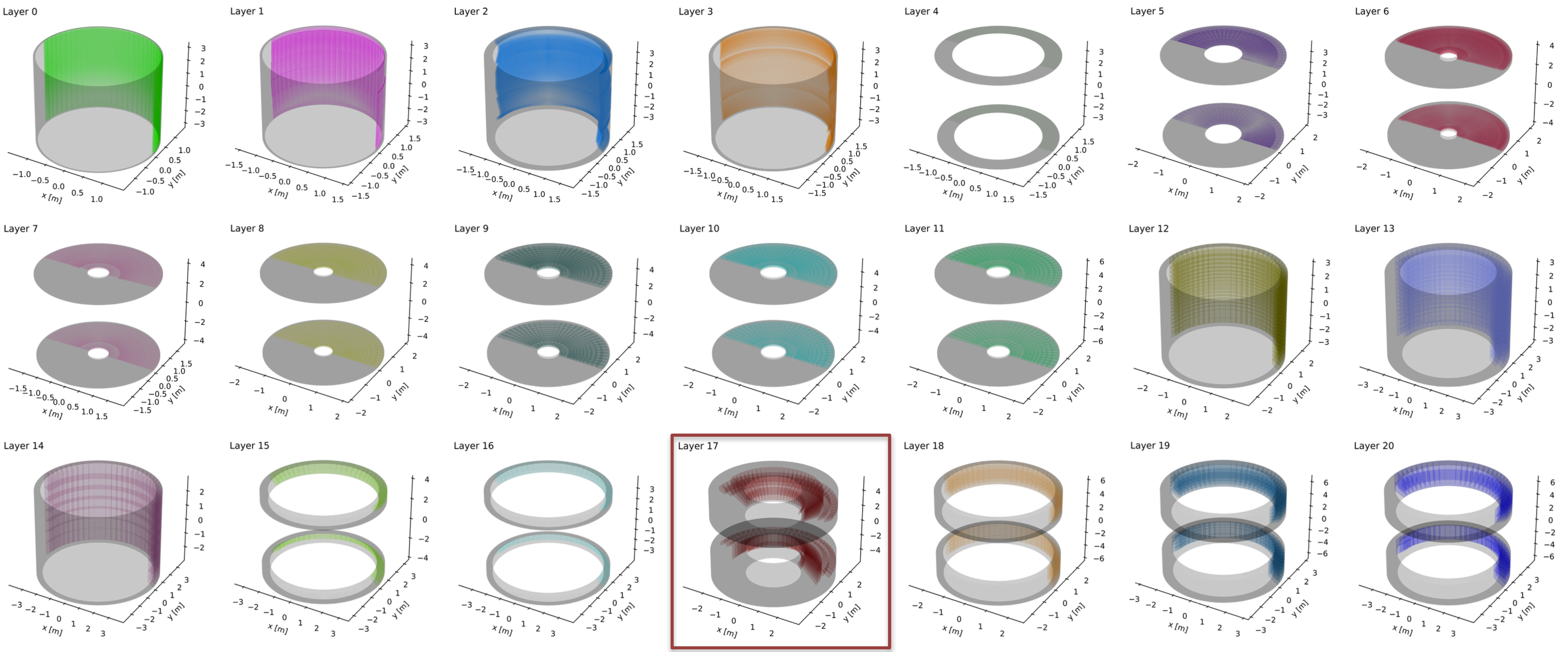
Cell in x, y, z

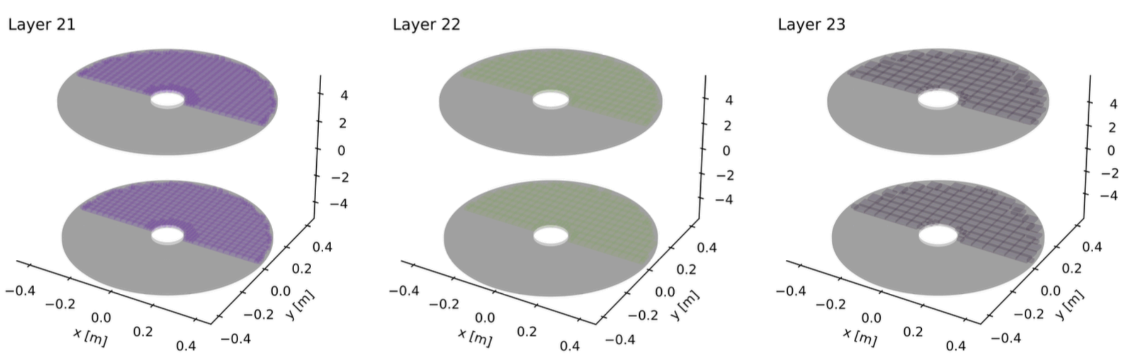
1. Model calo cells as cube in one of three coordinate systems
   - Barrel / Tile cells: cubes in $\Delta\eta, \Delta\phi, \Delta r$
   - Endcap cells: cubes in $\Delta\eta, \Delta\phi, \Delta z$
   - FCAL cells cubes in $\Delta x, \Delta y, \Delta z$
2. Transform vertices of cube in desired coordinate system
3. Transformed cell: convex hull of transformed vertices

Visualisation of full calorimeter system from all individual cells



Simplified Geometry: Modelling Layer Hulls





- Model layers as cylinders with R / Z computed from maximum extent of individual cells
- Automated modelling works very nicely in general
- Only exception is TileGap3 (layer 17), which is very far from being a cylinder and for now excluded

pyGeoSimplify: Geometry Loading

```python
import pygeosimplify as pgs
from pygeosimplify.simplify.layer import GeoLayer
from pygeosimplify.simplify.detector import SimplifiedDetector
```
[1] ✓ 3.0s

Import modules

```python
pgs.set_coordinate_branch("XYZ", "isCartesian")
pgs.set_coordinate_branch("EtaPhiR", "isCylindrical")
pgs.set_coordinate_branch("EtaPhiZ", "isECCylindrical")
```
[2] ✓ 0.0s

Set the name of the branches indicating the coordinate system of the cells. Supported: (XYZ, EtaPhiR, EtaPhiZ)

```python
geo = pgs.load_geometry("../tests/data/ATLASCaloCells.root", tree_name='caloDetCells')
geo.head(5)
```
[3] ✓ 0.4s

Load the geometry from ROOT file containing cell information

	id	layer	isBarrel	isCylindrical	isECCylindrical	isCartesian	eta	phi	r	x	y	z	deta	dphi	dr	dx	dy	dz
0	3179541336923570176	6	0	0	1	0	-2.559710	0.053900	617.735962	616.838867	33.279976	-3970.418457	0.1	0.098175	0.0	0.0	0.0	219.418503
1	3179541345513504768	6	0	0	1	0	-2.559648	0.151909	617.774719	610.660461	93.484917	-3970.418457	0.1	0.098175	0.0	0.0	0.0	219.418503
2	3179541354103439360	6	0	0	1	0	-2.559603	0.249912	617.803223	598.610657	152.794373	-3970.418457	0.1	0.098175	0.0	0.0	0.0	219.418503
3	3179541362693373952	6	0	0	1	0	-2.559574	0.347912	617.821228	580.805481	210.637146	-3970.418457	0.1	0.098175	0.0	0.0	0.0	219.418503
4	3179541371283308544	6	0	0	1	0	-2.559562	0.445909	617.828552	557.416626	266.456024	-3970.418457	0.1	0.098175	0.0	0.0	0.0	219.418503

- For all cells: need to provide r, x, y, z, η, ϕ of center position
- Additionally need to provide:
  - Cell widths $\Delta X, \Delta Y, \Delta Z$ for cells with XYZ coordinate system
  - Cell widths $\Delta\eta, \Delta\phi, \Delta r$ for cells with EtaPhiR coordinate system
  - Cell widths $\Delta\eta, \Delta\phi, \Delta z$ for cells with EtaPhiZ coordinate system
- All cells must be assigned to a layer (number) and corresponding layer type (barrel or endcap layer) must be indicated