ATLAS Detector Simulation in the Integrated Simulation Framework applied to the W Boson Mass Measurement

Dissertation
zur Erlangung des akademischen Grades
Doctor of Philosophy

eingereicht an der
Fakultät für Mathematik, Informatik und Physik
der
Universität Innsbruck

von
Mag. rer. nat. Elmar Ritsch
`elmar.ritsch@uibk.ac.at`

Betreuer der Dissertation:
Ao. Univ.-Prof. Dr. Emmerich Kneringer (Institut für Astro- und Teilchenphysik)
Prof. Dr. Daniel Froidevaux (CERN & Radboud University Nijmegen)
Dr. Andreas Salzburger (CERN)

Innsbruck, Dezember 2014

Abstract

One of the cornerstones for the success of the ATLAS experiment at the Large Hadron Collider (LHC) is a very accurate Monte Carlo detector simulation. However, a limit is being reached regarding the amount of simulated data which can be produced and stored with the computing resources available through the worldwide LHC computing grid (WLCG).

The Integrated Simulation Framework (ISF) is a novel approach to detector simulation which enables a more efficient use of these computing resources and thus allows for the generation of more simulated data. Various simulation technologies are combined to allow for faster simulation approaches which are targeted at the specific needs of individual physics studies. Costly full simulation technologies are only used where high accuracy is required by physics analyses and fast simulation technologies are applied everywhere else.

As one of the first applications of the ISF, a new combined simulation approach is developed for the generation of detector calibration samples used in the W boson mass measurement. The precise measurement of the W boson mass is crucial for testing the validity of the Standard Model (SM) of particle physics. With a targeted relative error of less than 10^{-4} for the measurement with the ATLAS detector, billions of simulated collision events are required. Using the flexibility of the ISF, an outstanding agreement with full Geant4 simulation is achieved for the simulation of the crucial lepton observables in $Z \to ee$ events. File sizes which are 2 times smaller and execution speeds which are about 5 times faster than traditional Geant4 simulation approaches illustrate that an important step towards the large-scale Monte Carlo production for the W boson mass measurement is achieved through using the ISF.

Acknowledgements

The writing of this thesis was not possible without the help and support of so many people.

I want to begin by thanking Dr. Andreas Salzburger for being a great teacher since my very first days in the field of high energy physics when I was summer student at CERN. Your support and guidance all along the way allowed me to be involved in projects which are essential for the operation of physics experiments in a large collaboration. I benefit from this valuable knowledge and experience for the rest of my personal and professional life.

Thank you Prof. Dr. Emmerich Kneringer, for your excellent support from the final years of my diploma studies onwards and throughout my entire time as a doctoral student. Your support enabled me to carry out the doctoral studies at the University in Innsbruck, while being based in Geneva for most of the time.

Prof. Dr. Dietmar Kuhn I want to thank you for giving me the great opportunity to discover the field of high energy physics. Without your support and the support of the entire group, I would not be writing this PhD thesis nor would I experience the unique work environment at CERN.

This thesis would not have been possible without the help and support of many colleagues and friends in the ATLAS collaboration. I thank particularly Dr. John Chapman for your support of my PhD project from the very first day on, and also for being very patient when mistakes were made (usually by me). Thank you Dr. Andreas Schälicke and Dr. Peter Sherwood for your valuable inspiration and guidance in the early stage of my PhD project. I thank Dr. Michael Dührssen and Dr. Zach Marshall for taking your time in answering numerous questions that I had.

A big thanks goes to my (partly former) office mates Dr. Anthony Morley and Dr. James Catmore for patiently helping me out so many times and finding answers to problems which I was not able to solve alone. You really showed me what the collaborative spirit is all about and I can not thank you guys enough for it.

Thank you Cécile and Bouhlala for your patience in times when work demanded a great deal of my time. It is your support which gives me the power to write this thesis and carry out my PhD studies.

Contents

Part I

Introduction and Overview

Chapter 1

Introduction and Motivation

Particle physics is among the most fundamental approaches of natural science aimed at understanding the laws of nature. It concerns the study of the fundamental forces and processes acting upon the elementary parts of matter. These elementary parts of matter, and composite objects formed by small numbers of elementary parts of matter, are generally referred to as *particles*. A number of sciences benefit from the understanding gathered in particle physics and its experiments. Its application ranges from studying the (early) development of the universe to radiation therapy in medicine.

Numerous particle physics experiments are dedicated to measuring particle properties in order to study the interactions between them. In doing so, new (unstable) particles are discovered and theoretical models are being tested by experiments. This thesis concerns the ATLAS experiment [1] at the Large Hadron Collider (LHC) [2] at the CERN laboratory near Geneva, Switzerland. In the LHC, collisions of protons (or lead ions) at high energies[1] are performed to probe the structure of matter. The particles emerging from these collisions are measured as they traverse particle detectors like the ATLAS detector. The measurements of these particles are needed to understand the underlying processes involved in the creation of the particles and to compare this to different theoretical predictions.

Modern particle physics relies strongly on computer simulation in order to facilitate accurate physics studies. The physics processes taking place at the initial particle collision are simulated with so-called event generators. Detector simulation, on the other hand, aims to describe the effect of the detector to traversing particles emerging from the collision. Detector digitization emulates the response of the detector to the particles which are simulated by the detector simulation. The particles traversing the detector interact with its material or decay into other particles. Due to the highly complex design of modern high energy physics detectors and the stochastical behaviour of the numerous particle-matter interaction processes, it is nearly impossible to predict the detector measurements for a given input analytically – even if the underlying processes were entirely understood. Therefore, software tools are used to simulate the most relevant processes and their impact on the measurements of the particles traversing such a particle detec-

[1]The LHC is designed for collision energies up to 14 TeV centre-of-mass energy.

tor. These detector simulations together with detector digitizations enable a detailed modelling of the detector response to incident particles, which is a necessity for many physics studies. Monte Carlo (MC) methods [3, 4] are used to simulate the stochastical behaviour of the particle decay and interaction processes. Reconstruction software is used to identify and measure particles in both, simulated and recorded data.

Many particle physics studies require simulated collision events in order to estimate uncertainties in the experimental setup. Figure 1.1 illustrates the importance of detector simulation in the search for the Higgs boson and its discovery [5, 6]. In this study, detector simulation is carried out to compute the uncertainty of the predicted measurement outcome of the null hypothesis, in which the Higgs boson would not exist. The significance of the null hypothesis is tested by comparing the predicted measurement outcome and its uncertainty with the data recorded by the detector. This directly leads to the discovery of the Higgs particle, due to a very low significance of the computed null hypothesis with the measured data.

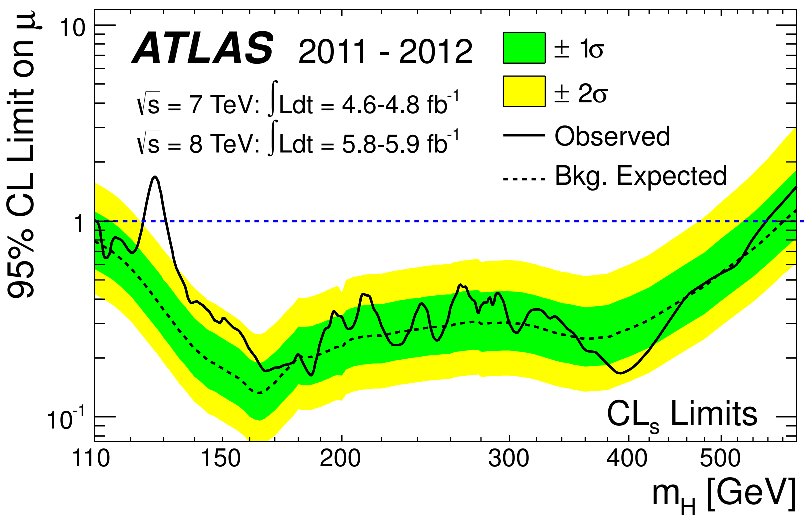


Figure 1.1: In the search for the Higgs particle, Monte Carlo detector simulation is crucial for predicting the uncertainty of the expected measurement of the null hypothesis (background-only hypothesis). The 95% confidence level (CL) upper limit on the signal strength μ is shown as a function of the Higgs mass m_H. A significant disagreement between the predicted null hypothesis and the measured data is visible at $m_H \approx 126.5$ GeV (image: [5]).

Besides the crucial role of simulation in the discovery of the Higgs particle, it is an important tool for data analysis in almost any high energy physics study. Some studies required high-statistics simulation samples in order to achieve significant results. About one billion simulated collision events were required for the measurement of the W boson with the D0 experiment [7]. Due to this high demand for simulation samples, a dedicated fast simulation approach was developed for this particular analysis.

In general, the most accurate simulation methods are often very time consuming. The resulting lack of high statistics simulation samples can become a limiting factor for the accuracy of physics analyses. Therefore, different fast simulation approaches are developed by the high energy physics community in order to speed up the generation of simulated data. Such methods are of crucial importance for physics analyses that aim to test a variety of models by *scanning* a parameter space (that is inherent to the respective theoretical model) and comparing the compatibility between the resulting simulated data and the measured data.

In this thesis a flexible simulation framework for the ATLAS experiment is presented, named the Integrated Simulation Framework (ISF). The framework enables fast detector simulation approaches by balancing high simulation accuracy with high simulation speed. It allows for dynamically defined regions of interest (ROIs) within individual events due to its innovative particle routing algorithms. Different simulation techniques may be used to simulate particles within the different ROIs. Thus, allowing to use highly accurate particle simulation where required by physics studies and fast simulation techniques where less critical for subsequent analyses.

Moreover, the Integrated Simulation Framework is developed to unify all ATLAS detector simulation approaches into one common program flow. This enables the use of the ISF as the default simulation framework for official ATLAS Monte Carlo production. The ISF is responsible for carrying out tasks which are common to all ATLAS detector simulation setups. Various full and fast simulation approaches were developed over time, however, compatibility between the individual approaches was generally not supported before the implementation of the ISF. Simulation techniques which are integrated into the ISF are compatible with each other and can be used in combination within the framework. This enables entirely new ATLAS detector simulation approaches, which were not possible in the past.

1.1 MC Production Computing Resource Usage

Given the importance of MC detector simulation to physics analyses, several billions of collision events are typically simulated in various ATLAS Monte Carlo production campaigns. Four ATLAS MC production campaigns were carried out during the first period of LHC operation (Run 1): mc09, mc10, mc11 and mc12. Table 1.1 summarizes the approximate number of events simulated in full and fast ATLAS detector simulation in the last two campaigns during Run 1 (mc11 and mc12) and in the ongoing mc14 MC production campaign.

The computing grid, a worldwide network of computing centres, is used by the ATLAS collaboration for simulation production, data processing and for the storage of datasets. Figure 1.2 illustrates that the ATLAS computing grid usage is dominated by Monte Carlo simulation.

[2]mc14 is the currently ongoing ATLAS Monte Carlo production campaign. Therefore, the provided numbers of simulated events are not final.

MC Campaign	Full Simulation (10^9 events)	Fast Simulation (10^9 events)
mc11	3.64	3.27
mc12	6.37	6.43
mc14[2]	0.85	–

Table 1.1: Billions of events simulated with full and fast ATLAS detector simulation in the mc11, mc12 and mc14 MC production campaigns, respectively (source: [8, 9]).

The demand for computationally intensive MC production will further increase with the start of LHC Run 2 in spring 2015. Collision data with an integrated luminosity of 26.4 fb^{-1} was recorded by the ATLAS detector throughout LHC Run 1 from 2010 to 2012 [10]. An integrated luminosity of about 100 fb^{-1} is expected to be recorded by the ATLAS detector during LHC Run 2 from 2015 to 2017 [11]. This significant increase of recorded luminosity is enabled through a higher instantaneous luminosity provided by the LHC due to more challenging pile-up[3] conditions. With the given "flat computing budget" throughout the next years [12], the available computing resources must be utilized more efficiently in order to cope with the increased demand.

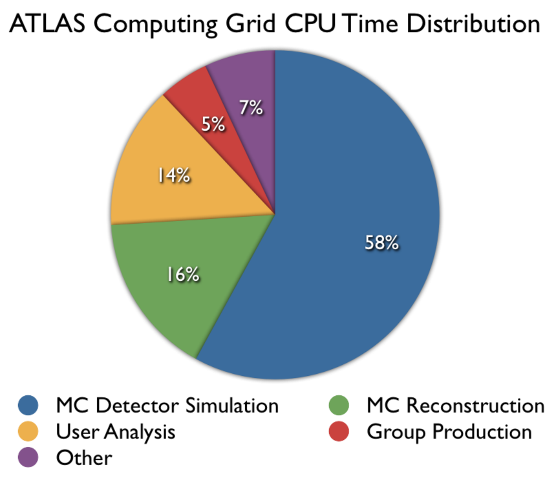
The costs of generating the entire mc12 simulation sample can be estimated to be about 16 million CHF[4] if it were generated on the Amazon Elastic Compute Cloud (EC2) [15]. This estimate likely provides an upper bound of the real cost as the computing resources used for ATLAS MC production are generally operated by non-commercial institutions. With this estimate, a 1% speed improvement in the mc12 production algorithms results in the generation of additional MC samples worth approximately 160 000 CHF in computing costs.

Fast MC production algorithms significantly increase the MC production output rate with respect to (more accurate) full simulation methods. In other words, fast algorithms allow for the production of more simulated events per calendar year with a given amount of computing resources (or budget) than full simulation does. Higher execution speed is generally achieved with less accurate algorithms. A number of fast algorithms have been developed by the ATLAS collaboration in order to speed up the individual steps in the MC production chain and to reduce the corresponding output file sizes:

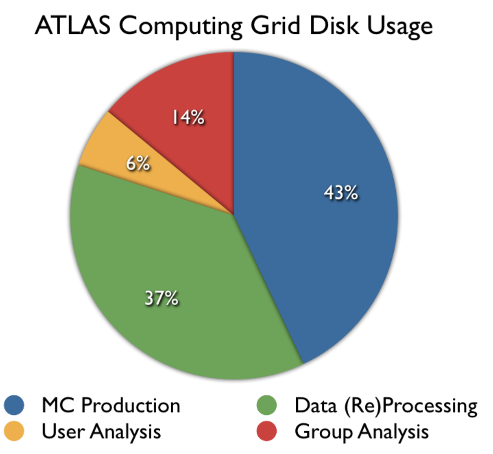
Fast detector simulation methods aim to increase the speed of MC detector simulation. As detector simulation accounts for the greatest share of ATLAS computing

[3] *Pile-up*: in-time and out-of-time proton-proton collisions from the same, previous or next bunch-crossings.

[4] This is based on a total number of $\sim$ 12.7 billion simulated events with an average CPU time requirement of $\sim$ 125 seconds per event for detector simulation and $\sim$ 37 seconds per event for digitization and reconstruction (source: ATLAS dashboard). Public figures for ATLAS computing costs are not available. The computing costs (not including storage) are therefore estimated by the cost of equivalent computing time on the Amazon Elastic Compute Cloud (EC2) with 2 GiB memory per CPU core in Europe, which equates to about 0.028 CHF/hour.



(a)



(b)

Figure 1.2: The ATLAS computing grid utilization between January 2010 and November 2014, averaged over all grid centres. The (a) CPU time consumption and (b) physical disk sizes of individual ATLAS projects are illustrated relative to the total grid usage. Monte Carlo production, which consists of MC detector simulation and MC reconstruction, takes the biggest share of the overall grid resource usage. Within MC production, detector simulation is by far the most time consuming step (source: ATLAS dashboard [13, 14]).

resource usage (Figure 1.2a), fast detector simulation can significantly increase the overall MC production output rate. The Integrated Simulation Framework enables significant speed improvements in this production step, with a minimal loss of accuracy. In the ISF, full simulation algorithms may only be applied in regions where high accuracy is necessary for subsequent analyses and fast simulation algorithms are applied everywhere else.

Fast digitization methods approximate the detector response to the detector simulation output.

Fast reconstruction methods use the MC truth information generated by the detector simulation in order to lower the algorithmic complexity in the track reconstruction step.

Approximately half of all simulated events in official ATLAS MC production are generated using fast simulation methods (Table 1.1). The percentage of fast simulation is expected to increase in order to cope with the stringent demand during LHC Run 2 operation.

A fast ATLAS Monte Carlo chain is currently being developed. The aim is to combine the fast algorithms of the three steps described above into one step, potentially achieving a Hertz-level event simulation output rate. Intermediate output formats in the MC production chain will not be recorded to disk, thus saving disk space and reducing the computational costs for input-output processing. The ISF plays a crucial role in enabling very fast detector simulation approaches. Detector simulation with execution speeds up to 3000 times faster than full detector simulation has been demonstrated within the ISF.

Chapter 2

The Standard Model of Particle Physics

The Standard Model (SM) of particle physics [16] is a theory which forms the basis of modern high energy physics. It describes three of the four fundamental forces of nature: the electromagnetic force, the weak nuclear force and the strong nuclear force. The Standard Model does not describe gravity[1].

This chapter provides a basic description of the most fundamental terms defined by the Standard Model. The underlying motivation and derivation in mathematical terms can be found in the references provided throughout this chapter.

Section 2.1 describes the building blocks of matter, their properties and how they interact with each other. The following Section 2.2 covers the importance of the W boson mass measurement in order to test the consistency of the Standard Model through experiments. The simulation framework described in this thesis is applied to the measurement of the W boson mass with the ATLAS experiment (covered in Chapter 12).

2.1 Particles in the Standard Model

The fundamental building blocks of matter and the interactions between them are described by particles. The theory characterizes four fundamental types of particles (Figure 2.1):

Leptons are subject to the weak force and electrically charged leptons are also subject to the electromagnetic force. Leptons exist in three generations. Each generation is formed by an electrically charged particle (with $Q = -1e$) and a corresponding (electrically neutral) neutrino particle. Antiparticles with inverted charges exist for each particle in each generation. The first generation consists of the electron (e^-) and the electron neutrino (ν_e). The second generation is formed by the muon particle (μ^-) and the muon neutrino (ν_μ). Finally, the third lepton generation consists of the tau particle (τ^-) and the tau neutrino (ν_τ). The respective masses

[1]Though, extensions to the Standard Model are being studied, which try to incorporate gravity.

of the charged leptons increase from lower to higher generations. Thus, the electron forms the lightest charged lepton, the muon particle the second lightest/heaviest and the tau particle the heaviest. Neutrinos are particularly difficult to measure directly, as they only interact through the weak interaction (and gravity) with other particles. Only upper bounds for their respective masses have been determined so far [17].

Quarks are subject to the electromagnetic force, the strong nuclear force and the weak nuclear force. Quarks have an electric charge of either $Q_+ = +\frac{2}{3}e$ or $Q_- = -\frac{1}{3}e$. Like leptons, quarks exist in three generations. Each generation consists of a doublet of quarks with charges Q_+ and Q_- respectively. In addition to the electric charge, quarks carry one of three colour charges. Antiparticles with inverted charges exist in each respective generation. The first generation is formed by the up quark (u) and the down quark (d). The second generation consists of the charm quark (c) and the strange quark (s). The third generation is formed by the top quark (t) and the bottom quark (b). The top quark is the heaviest of all fundamental particles in the SM. With the exception of the strange quark, the quarks in each generation are heavier than the leptons of the respective generation. The force acting between quarks increases with distance, preventing quarks from appearing as free particles. Free particles composed of multiple quarks exist only with a net colour charge of zero (they are "colourless").

Gauge bosons are the carriers of the fundamental forces of nature which are described by the Standard Model. Gauge bosons are spin-1 particles with an electric charge of either $Q = \pm e$ or $Q = 0$. Four different gauge bosons exist. The photon (γ) is the carrier particle of the electromagnetic force. The electrically charged $W^\pm$ and neutral Z particles are the carrier particles of the weak nuclear force. The gluon particles (g) are the carrier particles of the strong nuclear force. At higher energy scales the weak interaction is comparable in strength to the electromagnetic interaction, hence, both are described in the common electroweak theory (which is part of the SM).

The Higgs boson is a key particle predicted by the Standard Model and recently discovered by the ATLAS and CMS collaborations [5, 6] with the Large Hadron Collider at the CERN laboratory. It is a particle with spin zero and it corresponds to an excitation of the Higgs field. The non-zero vacuum expectation value of the Higgs field gives rise to masses of leptons, quarks and gauge bosons. The Higgs mechanism[2] generates the mass of the $W^\pm$ and Z gauge bosons through electroweak symmetry breaking [18–23]. Lepton an quark masses are generated through the so-called Yukawa interaction [16].

The fundamental particles described above can form both stable and unstable composite objects. Bound states of quarks (through the strong nuclear force) are classified

[2]The correct full name is Englert-Brout-Higgs-Guralnik-Hagen-Kibble mechanism.

as hadrons. Hadron particles are further categorized into mesons and baryons. Meson particles are quasi-stable bound states of a quark and an antiquark particle. Baryons are bound states of three quarks. Whereas all meson particles are unstable, one stable baryon does exist: the proton (uud). Together with the neutron[3] (ddu) it forms all stable nuclei of the visible mass in the universe.

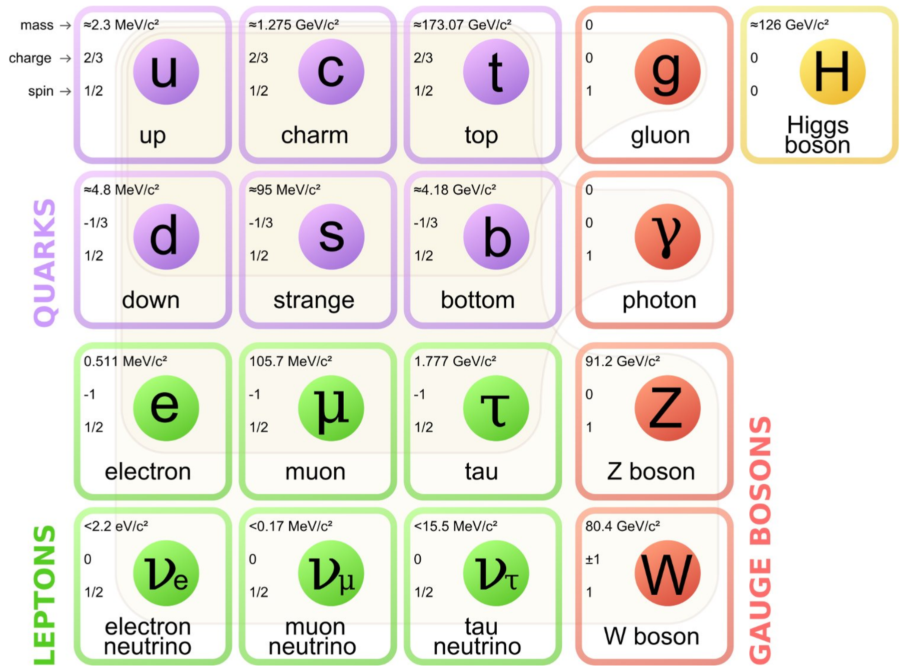


Figure 2.1: The particles of the Standard Model of particle physics. Quark and lepton particles are classified in three generations, which are arranged by particle mass. Gauge bosons mediate the fundamental forces described by the Standard Model: the electromagnetic force (photon), the weak nuclear force ($W^{\pm}$ and Z bosons) and the strong nuclear force (gluon). The recently discovered Higgs boson is an excitation of the Higgs field which gives rise to mass of the leptons, quarks and gauge bosons (image: [26]).

The validity and accuracy of the Standard Model has been tested by numerous experiments throughout the last decades. So far, the Standard Model has not yet been disproven by any experimental result. The Higgs boson is among a number of particles which have been experimentally discovered after their existence has been predicted by the SM. Nevertheless, the SM contains 19 free parameters[4] which can be determined only through experimental measurements.

[3]Free neutrons have a half life of $\tau_n = 878.5 \pm 0.8$ s [17, 24, 25]. However, neutrons do form stable nuclei together with protons.

[4]In fact 7 additional parameters are required to describe non-zero neutrino masses in the model. Three parameters for the respective neutrino masses and seven parameters to describe the mixing between the neutrino types.

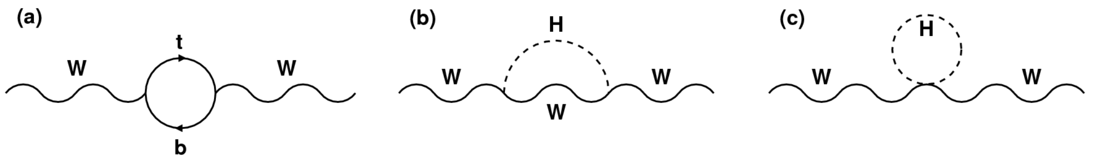


Figure 2.2: *"Feynman diagrams of loop processes that lead to a dependence of the W boson propagator on (a) the top quark mass and (b, c) the Higgs boson mass"* (images and caption from [27]).

2.2 The W Boson Mass

The $W^{\pm}$ boson is a fundamental particle of the Standard Model. Together with the Z boson, it is the force carrier of the weak nuclear force. Both particles were discovered in the year 1983 by the UA1 and UA2 experiments at the Super Proton Synchrotron accelerator-collider [28]. Their masses are generated through electroweak symmetry breaking which is described by the Higgs mechanism in the SM.

Loop corrections to the W boson propagator (Figure 2.2) introduce a direct relationship between the $W^{\pm}$ boson mass (M_W), the top quark mass (m_t) and the Higgs boson mass (M_H) [17, 29–33]. The loop corrections (to all orders) are denoted as Δr in the following relationship [33]

$$M_W^2 = M_Z^2 \left\{ \frac{1}{2} \sqrt{\frac{1}{4} - \frac{\pi\alpha}{\sqrt{2} G_\mu M_Z^2} \left[1 + \Delta r \left(M_W, M_Z, M_H, m_t, ...\right)\right]} \right\} , \qquad (2.1)$$

where α is the fine structure constant, G_μ is the Fermi constant and M_Z is the Z boson mass. Due to the M_W-dependence of Δr, an iterative procedure is applied to compute M_W in this equation. Taking into account only one-loop corrections, the dominating terms in Δr are proportional to m_t^2 and $\log(M_H)$ [17].

Given the precise measurements of α[5], G_μ[6], M_Z[7] and the recently measured M_H, the dominating experimental uncertainties for testing the validity of Equation 2.1 are M_W (relative error of measurement is 10^{-4}) and m_t (10^{-3}) [17]. It is possible to overconstrain the parameters of the SM in the electroweak sector such that an "allowed" parameter space for M_W and m_t can be computed [36]. In this computation, M_W and m_t are free fit parameters and any experimental measurement of them is not taken into account. Comparing the measured values of M_W and m_t with this parameter space allows to stringently test the validity of the SM (see Figure 2.3).

Furthermore, the parameters in the electroweak sector can be overconstrained to allow for a direct computation of M_W. In [38] the W boson mass is computed through

[5]Recent measurements have determined α with an impressive relative uncertainty of 6.6×10^{-10} [34].

[6]G_F usually is determined through a precise measurement of the muon particle lifetime τ_μ. Recent experiments have determined G_F with a relative uncertainty of 5×10^{-6} [35].

[7]A combined measurement of five collaborations (ALEPH, DELPHI, L3, OPAL, SLD) has determined M_Z with a relative uncertainty in the order of 10^{-5} [30].

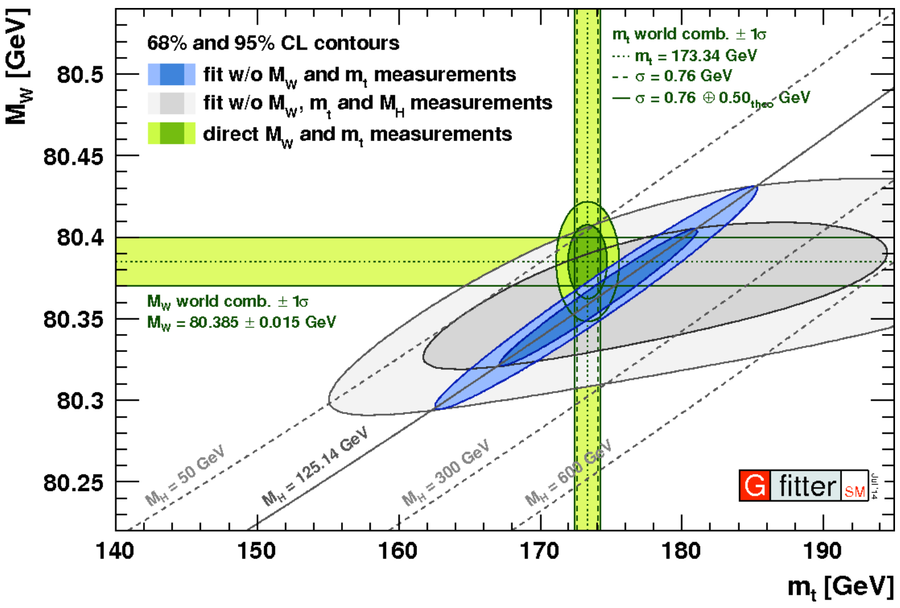


Figure 2.3: A comparison of the directly measured values for the W boson mass (M_W) and the top quark mass (m_t) with the parameter space consistent with the Standard Model of particle physics. The green area illustrates the combination of the direct experimental measurements of M_W and m_t, respectively. In the blue and grey areas, the parameters M_W and m_t are free fit parameters. The blue area illustrates the parameter space consistent with the other parameters in the SM, taking into account all experimental and theoretical uncertainties and the recent measurement of the Higgs boson mass (M_H). The grey area does not take into account the M_H measurement (image: [37]).

combining NNLO theoretical predictions with the measurements (world average) of all other parameters in the electroweak sector. This leads to

$$\begin{aligned} M_W^{\text{fit}} &= 80.3584 \pm 0.0046_{m_t} \pm 0.0030_{\delta_{\text{theo}} m_t} \pm 0.0026_{M_Z} \pm 0.0018_{\Delta\alpha_{had}} \\ &\qquad \pm 0.0020_{\alpha_S} \pm 0.0001_{M_H} \pm 0.0040_{\delta_{\text{theo}} M_W} \text{ GeV} , \\ &= 80.358 \pm 0.008_{\text{tot}} \text{ GeV} , \end{aligned} \tag{2.2}$$

with the individual measurement and theoretical uncertainties described in detail in Reference [38]. The current measured value (world average) is [17]

$$M_W = 80.385 \pm 0.015 \text{ GeV} . \tag{2.3}$$

The computed W boson mass in Equation 2.2 and the measured value in Equation 2.3 differ by more than one standard deviation. Moreover, the uncertainty of the experimentally determined value (world average) is more than twice the size of the error in the theoretical computation. Hence, it is of particular interest to measure M_W more accurately in order to determine whether the difference between the computed and the measured W boson mass is caused solely by measurement uncertainties, or whether this indeed points to an inconsistency in the electroweak sector of the SM. A permanent discrepancy could point to loop corrections due to other (yet unknown) particles that are not considered in the SM fit. Hence, a measurement of M_W is an indirect probe for physics beyond the Standard Model.

Extensive Monte Carlo simulation is fundamental to the method of measuring the W boson mass with the ATLAS detector. It is used as a tool to the study and minimize the relevant experimental uncertainties, as well as to conduct the final measurement of M_W. The simulation framework presented in this thesis is being validated with the aim of generating high-statistics simulation samples for the W boson mass measurement in due time (Chapter 12). It will ultimately enable the accurate measurement of M_W with the ATLAS experiment, for which a systematic uncertainty of about 7 MeV and a statistical uncertainty of about 2 MeV is expected with 10 fb^{-1} of integrated luminosity collected by the experiment [39]. This measurement will significantly contribute to the world average value of all M_W measurements and hence allow for a more stringent test of the SM.

Chapter 3

The ATLAS Experiment

The ATLAS (A Toroidal LHC ApparatuS) experiment is a general purpose particle detector at the LHC (Large Hadron Collider) at CERN laboratory near Geneva, Switzerland. The experiment is situated in a cavern approximately 100 meters underground at point 1 on the LHC ring. It is one of four main detectors which measures particles emerging from collisions of high energy protons or lead ions in the LHC. The other three detectors are ALICE [40], CMS [41] and LHCb [42] (see Figure 3.1).

The LHC accelerator is designed to produce proton-proton collisions with centre-of-mass energies of up to 14 TeV. To date, the highest generated centre-of-mass energy was 8 TeV for proton-proton collisions. There are four collision points along the LHC, each one situated at one of the four main experiments mentioned above.

The shape and size of the ATLAS detector can be approximated by a cylinder with length 44 meters and diameter 25 meters (see Figure 3.2). The LHC particle beam is directed along the cylinder axis and the nominal particle collision point is in the centre of the detector. Thus, collision products will traverse the detector from the inside towards the outside and in doing so, they will interact with the surrounding detector components. This leads to the layer-based design of the detector sub-systems (or sub-detectors) which guarantees efficient particle identification capabilities and accurate particle measurements. The innermost system is the ATLAS inner detector which is used for particle tracking (described in Section 3.2). Subsequently, particles will enter the calorimeter system where the particle energies are measured (Section 3.3). The muon spectrometer, surrounding the calorimeter system, is a second tracking detector system optimized for muon particle measurements (Section 3.4).

The components of each individual sub-detector can be further divided into active and passive detector parts. Active detector parts are involved in the measurement of collision products (e.g. silicon pixels measuring particle position and deposited energy). Active components are embedded in and surrounded by passive detector parts. As such, these are not directly involved in particle measurements but they are required by the active components to function (e.g. support structures, cooling systems or electronics). Particles will interact with both, active and passive detector parts. The signals created by active detector parts are further referred to as *readout signals* or *energy measure-*

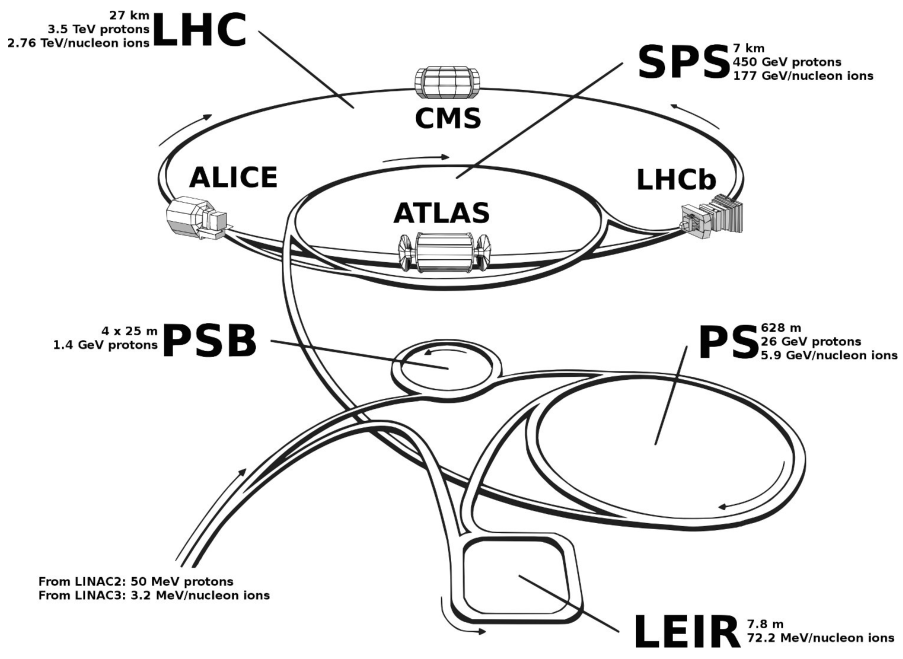


Figure 3.1: Overview of the LHC accelerator complex at CERN. Protons and lead ions run through a chain of accelerators before they are filled into the final LHC accelerator. Each individual accelerator along the chain adds more energy to the particles. The image also shows the position of the four main LHC experiments ATLAS, ALICE, CMS and LHCb (image: [43]).

ments. In the ATLAS inner detector and muon spectrometer, these signals are used to reconstruct particle trajectories (further called *tracks*) through the detector. The *energy measurements* in the calorimeter are used to reconstruct the energy deposited by the particles traversing the ATLAS calorimeter. From this, particle properties such as total energy, momentum and charge can be determined, which allow to study the preceding interaction process.

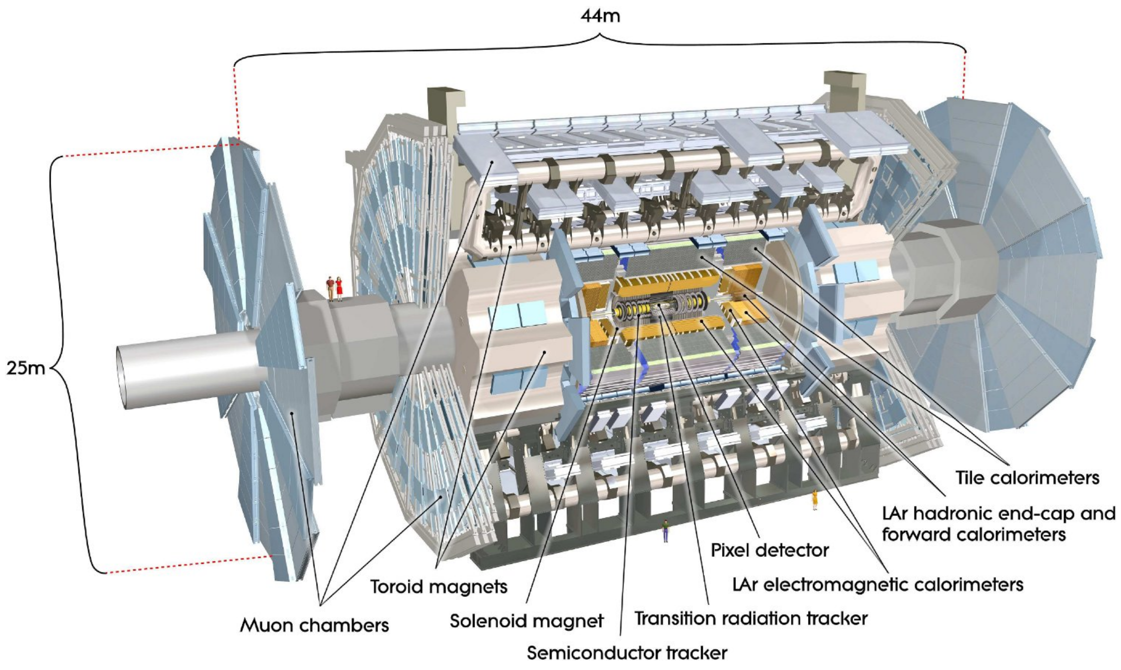

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Transition radiation tracker
Muon chambers
Solenoid magnet
Semiconductor tracker

Figure 3.2: Cut-away view of the ATLAS detector showing the individual detector technologies in use. The inner detector consists of the pixel detector, the semiconductor tracker and the transition radiation tracker. A solenoid magnet generates an almost uniform magnetic field with a strength of two tesla throughout all inner detector components. The calorimeter system consists of the liquid argon (LAr) and the scintillator-tile calorimeter systems. The muon system together with the superconducting toroid magnets forms the outermost part of the ATLAS detector (image: [44]).

3.1 Coordinate System

The global coordinate system [1, 45] used by the ATLAS collaboration has its origin at the nominal interaction (collision) point in the centre of the detector. The positive x-axis is defined to point from the interaction point to the centre of the LHC ring, the positive y-axis points towards the surface and the z-axis points along the beam direction forming a right-handed coordinate system. The azimuthal angle $\phi \in [-\pi, \pi)$ is measured in the x-y plane around the beam axis, with the positive x-axis at $\phi = 0$ and the positive y-axis at $\phi = \pi/2$. The polar angle $\theta \in [0, \pi)$ is measured from the positive z-axis at $\theta = 0$ to the negative z-axis at $\theta = \pi$.

The pseudorapidity η of a particle is defined as the rapidity y [46] of an equivalent, but massless particle:

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{3.1}$$

3.2 The Inner Detector Subsystem

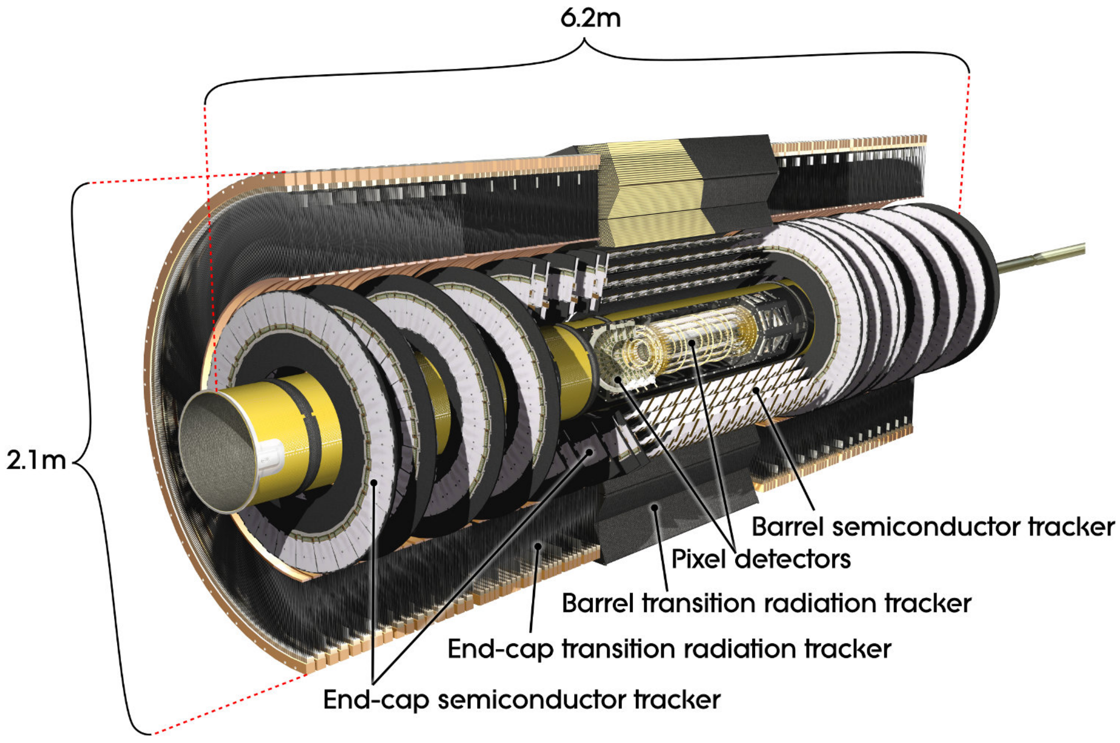

6.2m
2.1m
Barrel semiconductor tracker
Pixel detectors
Barrel transition radiation tracker
End-cap transition radiation tracker
End-cap semiconductor tracker

Figure 3.3: The ATLAS inner detector sub-system consists of the pixel detectors, the semiconductor tracker (SCT) and the transition radiation tracker (TRT). All three detector technologies are have a central barrel region and a two forward end-cap regions, respectively (image: [44]).

The inner detector (ID) is the innermost layer of the ATLAS experiment and therefore closest to the collision point. Particles generated in LHC particle collisions first pass through the ID before they traverse any other part of the ATLAS detector.

The main purpose of the ID is to track charged particles in order to measure their charge and momentum, and to locate their production vertices. The ID is embedded in a solenoidal magnetic field with a central strength of approximately 2 T, causing charged particles to have a curved trajectory when passing through it. From this, the curvature can be measured which in turn allows to compute the momentum.

The ATLAS inner detector consists of three sub-detector components (see Figure 3.3) which cover a pseudorapidity range of $|\eta| < 2.5$:

The Silicon Pixel Detector is the innermost detector system in ATLAS. It consists of three cylindrical layers formed concentrically around the beam pipe in centre of the detector. Each pixel detector layer consists of a silicon module with a thickness of 250 μm. Charged particles traversing the pixel detector will induce free charges in the silicon. The readout electronics attached to the silicon layers divides the layers into individual pixels of size 50×400 mm^2. The whole ATLAS pixel detector provides approximately 80.4 million readout channels (individual pixels) where each channel provides information about the free charge in the corresponding pixel through a time over threshold (ToT) signal. With the pixel detector being closest to the initial interaction point, its particle track measurements are essential for reconstruction of production vertices.

The Semiconductor Tracker (SCT) uses the same silicon-based detector principle as the pixel detector. The main difference to the pixel detector is that the SCT is formed with strips of sensitive silicon material and that it has a binary readout. The strips have a width (pitch) of 80 μm and a length of 6.4 cm. Each SCT layer is formed by a combination of two silicon strip layers which are twisted by a few mrad with respect to each other. A charged particle traversing one SCT layer will create readout signals in two separate silicon strips, which in turn allows to accurately measure the particle track position (space-point) at the crossing point of both strips. There are a total of four cylindrically-shaped SCT layers surrounding the pixel detector. Contrary to the pixel detector, the SCT readout does only provide information about whether or not a fixed free charge threshold was reached by each individual silicon strip.

The Transition Radiation Tracker (TRT) consists of approximately 351 000 straw drift tubes with a diameter of 4 mm each. Each tube is filled with a gas mixture (Xe-based in LHC Run 1) and has an anode wire running along its centre. Charged particles traversing the TRT will ionize the gas mixture inside the straws. The free charge will drift to the central anode wire, where the signal is measured and subsequently used to compute the closes approach radius of the charged particle. The TRT covers the detector region with pseudorapidity $|\eta| < 2.0$. Particles traversing the TRT create on average 36 measurements in the TRT straws. These measurements contribute to the inner detector track reconstruction and in particular to the measurement of particle momenta. In addition to the charged particle track measurements, the TRT also provides particle identification capabilities. The TRT straws are interleaved with fibers and foils which generate significant transition radiation if crossed by particles with a high Lorentz factor γ [47, 48]. The transition radiation photons will subsequently contribute to the ionization of the gas in the in the TRT straws along the initial particle's track. The TRT signals enhanced by transition radiation photons are classified as *high threshold* signals. The number of high threshold signals along a particle track allows to discriminate electrons from pions in particular.

3.3 The Calorimeter Subsystem

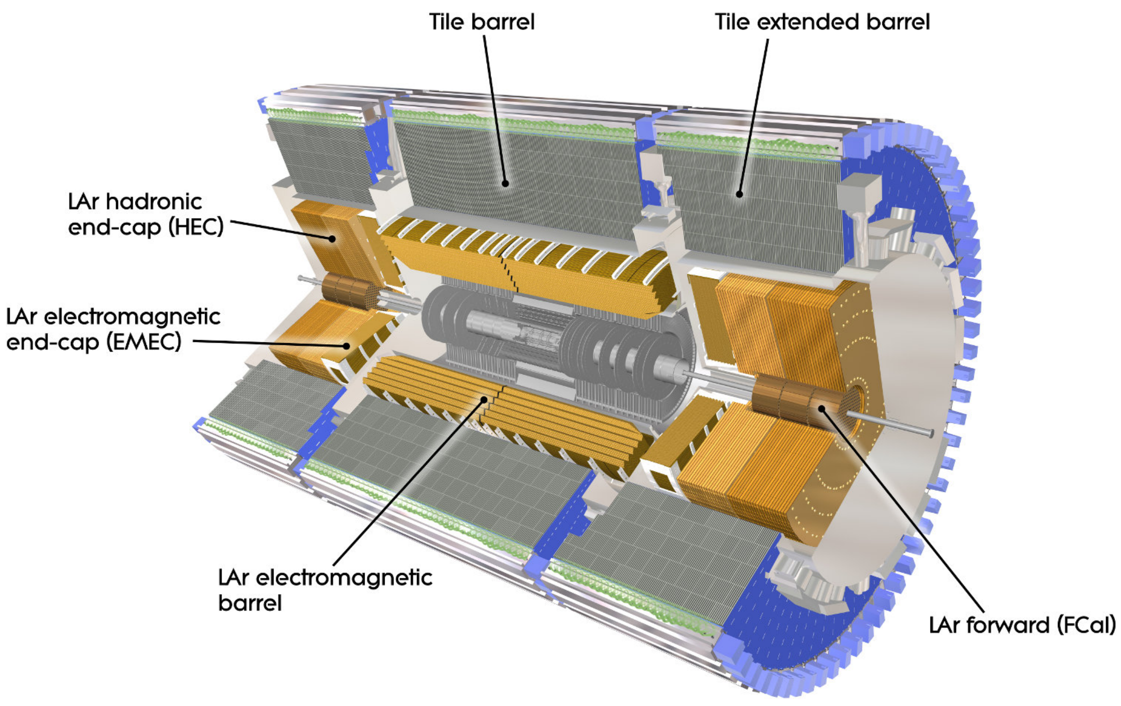

Tile barrel
Tile extended barrel
LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)

Figure 3.4: The ATLAS calorimeter sub-system with the electromagnetic calorimeter (EM) on the inside, surrounded by the hadronic calorimeter. The EM calorimeter consists of liquid argon-based (LAr) calorimeter systems in the barrel and end-cap regions. The hadronic calorimeter is composed of scintillator tiles in the barrel and extended barrel region, whereas the end-cap and forward hadronic calorimeter is formed by LAr-based components (image: [44]).

Particles originating from the interaction point enter the ATLAS calorimeter system after they have passed through the ID. Its main purpose is to measure the energy of incident particles. The ATLAS calorimeter is a sampling calorimeter and it consists of two main parts, the inner *electromagnetic (EM) calorimeter* and the outer *hadronic calorimeter* (see Figure 3.4).

Due to the fine granularity of the electromagnetic calorimeter, it provides precise energy measurements of particles that interact through electromagnetic processes with the detector material, in particular photons and electrons. The hadronic calorimeter provides energy measurements which are suited for jet reconstruction and missing transverse energy ($\not{E}_{\mathrm{T}}$) measurements.

Two different detector technologies are applied in the ATLAS calorimeter system. The first technology uses liquid argon (LAr) as the active detector medium and lead as the absorber medium. The second technology uses scintillator tiles as the active medium and steel as the absorber medium. The dense absorber media cause incident particles to interact with the detector and to create a showers of particles inside the calorimeter. Shower particles traversing the LAr calorimeter will ionize the liquid argon and thus create signals on the electrodes surrounding the LAr. The scintillator tiles create light signals corresponding to the energy of the shower particles it is being traversed by. Both calorimeter types only measure a fraction of the total shower energy in their active media (sampling calorimeter), the absorber media have no means of providing a readout signal.

The electromagnetic calorimeter is formed by LAr calorimeters in the barrel and end-cap region, covering a pseudorapidity range of $|\eta| < 3.2$. The hadronic calorimeter consists of a tile calorimeter in the barrel and extended barrel region ($|\eta| < 1.7$), a LAr calorimeter in the hadronic end-cap region ($1.5 < |\eta| < 3.2$) and LAr calorimeter in the forward region ($3.1 < |\eta| < 3.9$). The size of the individual calorimeter cells (granularity) is dependent on the region of the detector. The EM calorimeter has a finer granularity in the central η-region which is also covered by the inner detector, thus providing precise measurements of electrons on photons.

In order to maximize the chance of fully containing incident particles, the total thickness of the ATLAS calorimeter is approximately 10 interaction lengths (λ). The thickness of the EM calorimeter is more than 22 radiation lengths (X_0). Despite the size of the calorimeters, is possible for some particles in a particle shower to leak out of the calorimeter and enter the outermost sub-detector, the muon spectrometer (Section 3.4).

Muons do not create significant particle showers in the calorimeter's absorber media. Their interaction is mainly limited to multiple Coulomb scattering and ionization energy loss processes. Thus, muons (with energies above approximately 4 GeV) will likely pass the calorimeter and reach the next ATLAS sub-detector, the muon spectrometer.

3.4 The Muon Spectrometer Subsystem

The muon system or muon spectrometer (MS) is the outermost part of the ATLAS experiment. It is a particle tracking device with precise track reconstruction capabilities. The name stems from the fact that mostly muons will reach the MS to cause signals on sensitive detector parts. Ideally, other particle types should either loose all their energy in the calorimeter (hadrons, electrons, photons) or do not create any signal in the MS (neutrinos). However, "hadronic leakage" and decay in flight effects in the calorimeter do impact measurements in the MS.

Eight superconducting coils generate a (strongly non-uniform) toroid-shaped magnetic field throughout the muon system. This allows to measure each particle's charge over momentum ratio q/p while traversing the magnetic field. Combining this measurement with the corresponding inner detector q/p measurement, a higher accuracy on the overall q/p measurement is obtained for these particles.

Four different detector technologies are forming the sensitive detectors in the ATLAS muon spectrometer:

Monitored Drift Tubes (MDT) are gas filled tubes with a diameter of 29.97 mm and a length between 1 and 6 meters each. An anode wire runs along the centre of each tube and the tube surface acts as a cathode. Charged particles passing through a MDT ionize the Ar/CO_2 gas inside the tube and the freed electrons travel towards the central anode wire. The readout electronics converts the electron signal at the anode wire into processable data. The monitored drift tubes provide precise tracking measurements of charged particles (muons) in the MS throughout a region of $|\eta| < 2.0$.

Cathode Strip Chambers (CSC) are multi-wire proportional chambers covering a region of $2.0 < |\eta| < 2.7$ in the ATLAS muon system. The increased flux of particles in regions of higher pseudorapidity renders the use of MDTs ineffective for particle tracking. Thus, multi-wire proportional chambers are installed, which are capable of providing particle track measurements with counting rates up to 1000 Hz/cm^2.

Resistive Plate Chambers (RPC) are part of the ATLAS trigger system (see Section 3.6) in the muon system. RPCs are capable of generating fast response signals about traversing particles. RPCs consist of two resistive plates (phenolic-melaminic plastic laminate) aligned parallel to each other and separated by a distance of 2 mm. The gap between the plates is filled with a mixture of gases and an electric field is applied perpendicular to the plates. Charged particles traversing the gas will cause electron avalanches which are read out via metallic strips fixed on the outer surface of the resistive plates. The RPCs cover a region of $|\eta| < 1.05$ in the ATLAS muon system. In addition to their use in the trigger system, the RPC is also used in the muon track reconstruction, where they provide measurements arranged orthogonally to the precise MDT chambers.

Thin Gap Chambers (TGC) are part of the ATLAS trigger system covering the forward muon system regions of $1.05 < |\eta| < 2.7$. Like any detector technology used in the trigger system, also TGCs generate fast responses to traversing particles. TGCs are multi-wire proportional chambers. The name thin gap chamber was given due to the fact that the wire-to-cathode distance (1.4 mm) is smaller than the wire-to-wire distance (1.8 mm). Common with the RPC measurements, TGC measurements also provide input to the muon track reconstruction.

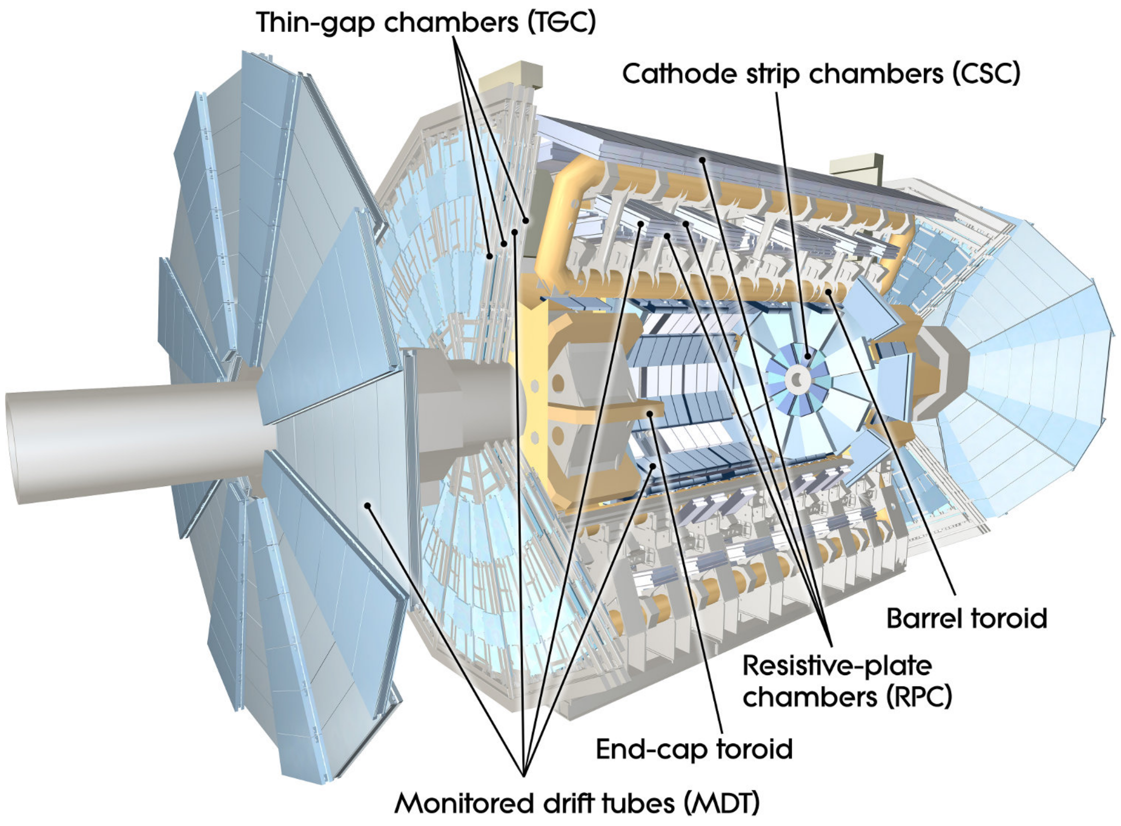

Thin-gap chambers (TGC)
Cathode strip chambers (CSC)
Barrel toroid
Resistive-plate chambers (RPC)
End-cap toroid
Monitored drift tubes (MDT)

Figure 3.5: The ATLAS muon system with the eight superconducting toroid magnets and its various sub-systems. Monitored Drift Tubes (MDT) provide precise measurements of muon particle tracks. Cathode Strip Chambers (CSC) are capable of generating particle measurements in the high particle flux environment of the forward detector region. Resistive Plate Chambers (RPC) and Thing Gap Chambers (TGC) generate very fast measurements which are used by the ATLAS trigger system (image: [44]).

3.5 Particle Signatures

The ATLAS detector is designed such that different types of particles will leave behind distinct signatures inside the detector. The combined information from the individual detector technologies and the sub-detectors allows to extract a great amount of information from each measured particle. Whereas individual sub-detectors may provide precise measurements of particular quantities of a particle (e.g. momentum or energy measurements), the combined information enables efficient particle type identification. Figure 3.6 shows typical signatures for common types of particles.

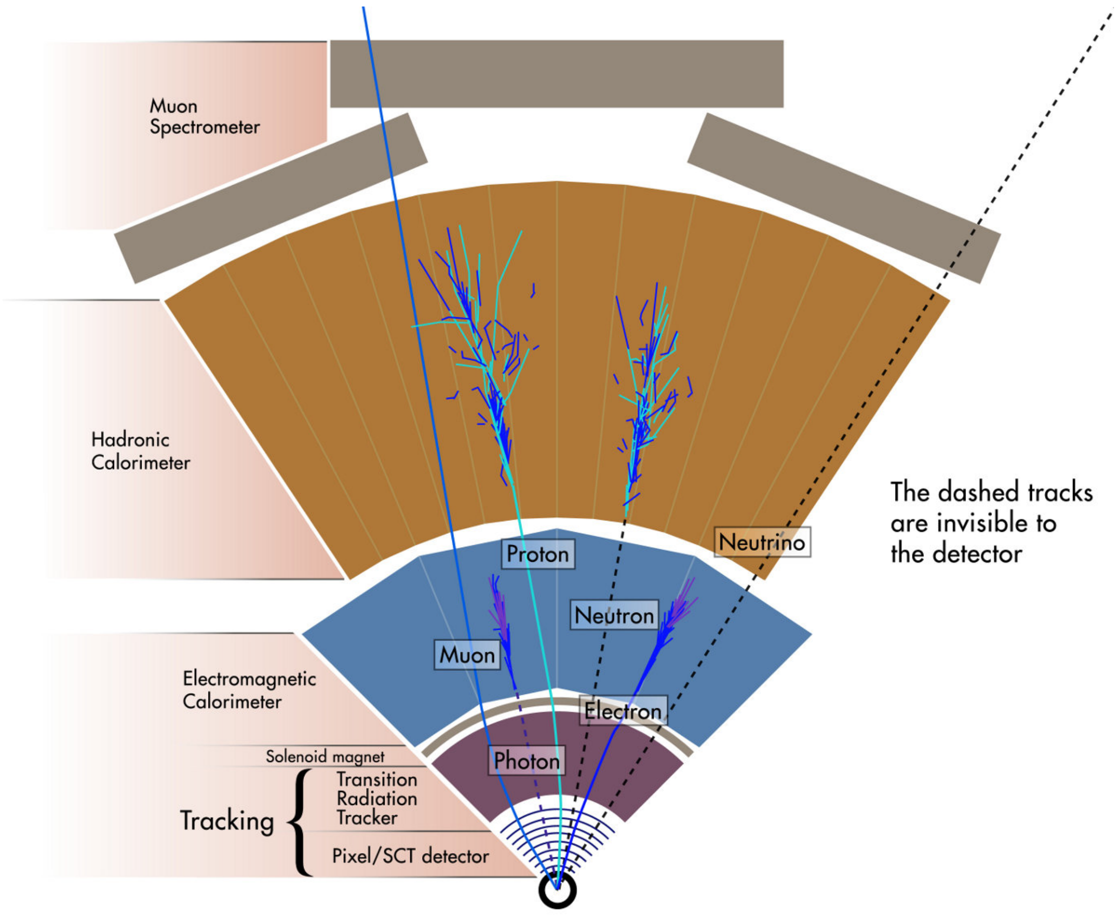


Figure 3.6: Particle signatures for different particle types when traversing the ATLAS detector. The particles originate at the interaction point and traverse the ATLAS detector in a radial direction. Charged particle trajectories are bent in the inner detector and the muon system due to the present magnetic fields (image: [44]).

3.6 Trigger System

The LHC is designed to generate one proton-proton collision inside the ATLAS detector every 25 ns. This corresponds to a collision rate of 40 MHz. With an average event size in the order of a few megabytes, it is not possible with the technology available today, to record and fully process every single collision event. The processes taking place in most of the collisions are of little or no relevance for subsequent physics analysis (Figure 3.7). Thus, a trigger system is introduced to only record events that pass well defined criteria at a defined rate (400 Hz during Run 1 and 1 kHz in Run 2). These criteria are set to maximize the chance of recording *interesting* events, i.e. events which contain objects which are to be studied in later physics analysis. In general, events containing leptons or particle jets with high transverse momenta, or events having a large missing transverse energy in the calorimeter, will be of interest and thus be recorded. The specifics about the trigger criteria are beyond this work and will thus not be discussed here.

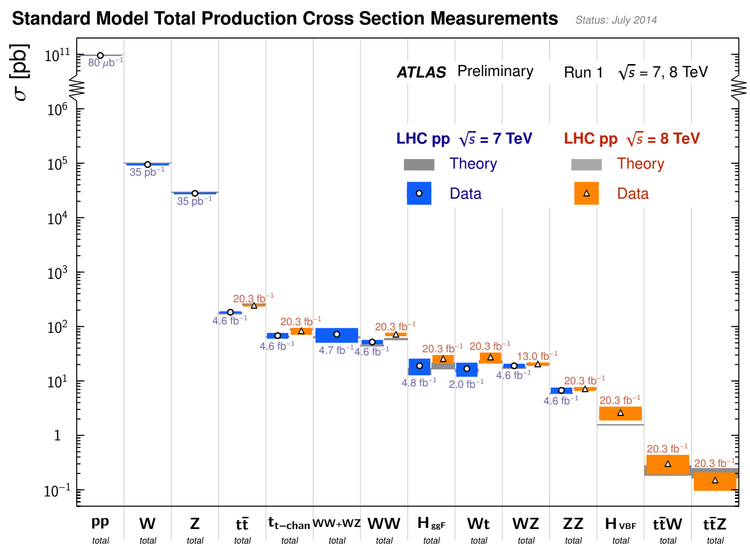


Figure 3.7: Standard Model cross sections measured with the ATLAS detector compared to theoretical predictions (at NLO or higher). The figure illustrates the many orders of magnitude separating the total cross section of inelastic proton-proton (pp) interactions and various cross sections for the generation of particles which are studied in Standard Model physics analyses. A trigger system is required to select and store only those collision events which are potentially relevant for subsequent physics studies (image: [49]).

The ATLAS trigger system is divided into three trigger stages (levels):

The Level-1 (L1) trigger lowers the detector output rate to about 75 kHz. It uses a limited amount of the total detector information to define regions of interest

(ROIs) in each collision event. Due to the detector specific requirements and the high event rates in the first trigger stage, the L1 is implemented with custom-built electronics.

The Level-2 (L2) trigger lowers the detector output rate to about 3.5 kHz. For this it uses the full detector information inside the previously defined ROIs. The information inside the ROIs contains only about 2% of the total event data measured in the ATLAS detector.

The Event Filter (EF) uses offline computing tools to determine whether the information present in the ROIs renders the event worth for permanent storage. This final stage in the ATLAS trigger chain reduces the detector event rate (trigger rate) to about 400 Hz during Run 1 and 1 kHz in Run 2.

The stages L2 and EF together form the High-Level-Trigger (HLT) which consists mainly of commercially available computers and networking hardware.

Specific triggers selecting electrons and muons are used for the detector calibration and the W boson mass measurement which is covered in Chapter 12.

3.7 ATLAS Computing Grid

The Worldwide LHC Computing Grid (WLCG) [50, 51] is used to store and process the vast amounts of simulated and recorded data which are produced by the ATLAS collaboration. The WLCG is divided into three different types of computing sites (see Figure 3.8):

The Tier 0 is the centre of the WLCG. Any collision event passing the trigger(s) of an LHC experiment will be sent to the Tier 0 facility. The Tier 0 will execute a first-pass processing of the detector data (RAW data) using experiment-specific reconstruction algorithms (Section 4.2.4). The reconstructed datasets (and subsets of the RAW data) are subsequently distributed among the Tier 1 sites to make them available for user analysis (and reprocessing). A copy of the RAW data is archived at the Tier 0 site. Tier 0 responsibilities are currently shared between the CERN Computing Centre and the Wigner Research Centre for Physics in Budapest, Hungary.

Tier 1 Facilities are mainly responsible for providing datasets which are of interest for the whole collaboration. Tier 1 facilities provide access to reconstructed detector data as well as simulated event data. Each Tier 1 site stores some fraction of the Tier 0's RAW detector data and it will be responsible for reprocessing this data in a reprocessing campaign.

Tier 2 Facilities are mainly used to process physics analysis and simulation jobs. About 140 Tier 2 sites are currently being hosted by various institutes around the world. Tier 2 sites differ in size an capacity and thus each site has a specific role which will differ from the roles of other Tier 2 sites. An institute hosting a Tier 2 site that is also involved in a particular physics analysis, will usually use their local Tier 2 to process and store the corresponding analysis jobs and datasets.

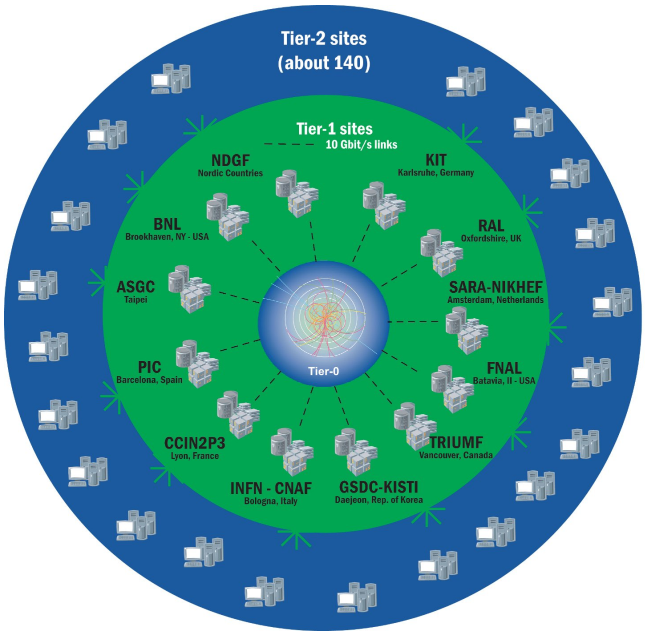


Figure 3.8: The structure of the Worldwide LHC Computing Grid (WLCG). Data from the LHC detectors is directly sent to the Tier 0 sites in the centre. The Tier 0 facility processes and distributes this data among the Tier 1 sites which are responsible for providing access for further analysis. About 140 Tier 2 sites are used for processing and storing datasets for physics analyses (image: [51]).

Chapter 4

ATLAS Monte Carlo Simulation

Monte Carlo detector simulation [52] is a tool used in high energy physics to emulate the response of a particle detector to individual particles emerging from collision events. It is used to predict the outcome of various types of measurements – from detector performance studies to physics analyses. Due to this, Monte Carlo simulation is a central tool not only for physics studies, but also for detector design and development.

This chapter discusses the individual steps involved in the creation of Monte Carlo simulation samples for the ATLAS detector. The Athena software framework, which serves as the underlying software infrastructure for detector simulation and the ATLAS offline software, is discussed in Section 4.1. Analysis algorithms may also use the Athena framework, but this is not a necessity. Section 4.2 discusses the typical Monte Carlo simulation scheme in ATLAS. In Section 4.3 two different simulation concepts are introduced: full and fast detector simulation. Both concepts find their use in the ATLAS collaboration and the individual detector simulation engines are described in this section.

4.1 The ATLAS Offline Software Framework (Athena)

The Athena framework [53, 54] is the main software infrastructure used by the ATLAS collaboration. This includes all steps involved in Monte Carlo simulation, data reconstruction and to some extend physics analysis. Athena is based on the Gaudi [55] framework which was initially developed by the LHCb collaboration.

User implemented C++ classes can be integrated into the Athena framework, through which access to the event input and event output is provided via so called *StoreGate containers.* Athena manages the event loop and the sequence in which different algorithms are executed inside the event loop (the Athena `AlgSequence`). Thus, Athena-based programs need to be configured according to the required data flow of the algorithms involved.

The setup of Athena-based programs is done via Python script files, so called `jobOptions`. These scripts are either provided as an argument to an `athena` shell command line or, alternatively, generated by *job transform* wrapper scripts [56]. Job transform scripts are used extensively in ATLAS Monte Carlo production campaigns.

The modular design of the framework together with the use of job transform or `jobOptions` Python scripts offers great flexibility to configure various types of Athena programs. Many core functionalities can be adjusted and changed without the need to re-compile the underlying implementation, written in the C++ programming language. For the Athena programs discussed in this theses the following properties will generally be adjusted individually for each program execution:

- defining the input dataset to read and the output dataset to write
- defining which ATLAS offline C++ (or Python) software algorithms will be executed
- defining and changing (optional) parameters for these algorithms and the tools they use

4.1.1 Athena Application Flow

The Athena framework supports custom algorithms and components which are implemented in C++ and Python programming languages. Some older components of the ATLAS offline software are implemented in Fortran 90, for which C++ wrappers are implemented to allow execution of this code within the Athena framework[1]. User defined algorithms are implemented into one or many of the different components provided by the Athena framework. A complete list of all Athena components is given in Reference [53]. The most relevant components in the context of this work are: *Athena Algorithms*, *Athena Tools* and *Athena Services*. A component can be implemented in either of these categories, however, most commonly a combination of different components is used to form a complete user algorithm. Figure 4.1 gives an overview of how a combination of Athena algorithms and Athena tools are used and how they are integrated into an Athena based program.

Athena algorithms, tools and services are generally implemented by the ATLAS software developers as a part of the official ATLAS offline software release. Alternatively, individual components can be implemented on top of the official ATLAS offline software release in order to carry out specific studies and measurements of (simulated or recorded) datasets.

Athena Algorithms – `AthAlgorithm`: The idea of processing individual (particle collision) events – each of them with its own input and output data – is one of the design principles of Athena and Gaudi. In order to do so, the framework runs in sequence through the events in an input dataset and processes them one by one. This loop over each individual event is know as the *event loop*. Individual C++ (or Python) based Athena algorithms are executed on each input event. An Athena algorithm is a C++

[1]Most of the Fortran-based source code has been replaced over the last years by equivalent algorithms implemented in C++.

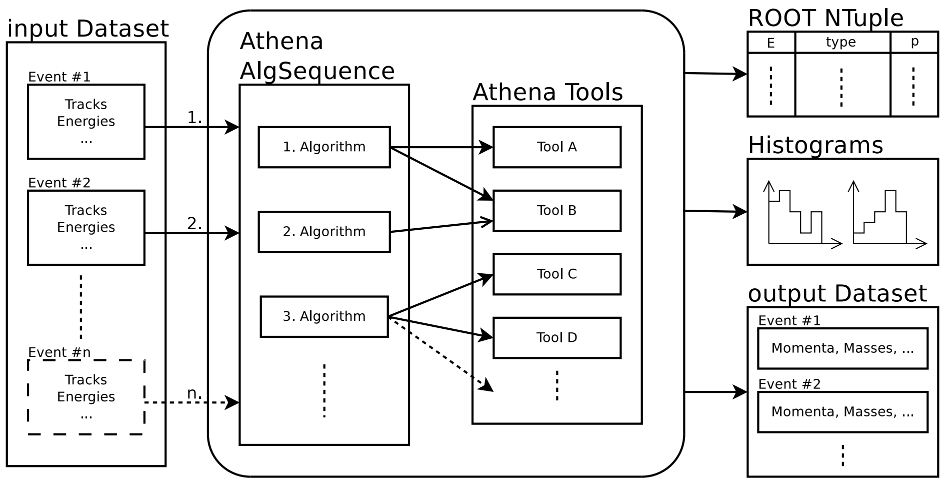


Figure 4.1: Athena application flow when running multiple Athena algorithms which use different Athena tools. The data containers for each event are handed to the Athena algorithms one at a time. The input and output types shown are examples for typical types used in combination with Athena algorithms and Athena tools. Within one Athena program execution, the Athena algorithms are called in the same sequence for each event. This sequence is given by the `AlgSequence` object, which is configured during the initialization phase through the Python `jobOptions` or job transform scripts. In most cases an input dataset is provided. However, there are certain cases where no input dataset is needed (for example physics event generators or single particle detector simulation).

(or Python) based class which inherits from the Athena `AthAlgorithm` class. While processing any event, an algorithm has access only to the data relevant to the currently processed event. This follows along with the underlying design principle of processing data which is generated by independent collision events. Thus such data needs to be processed individually. After reading the input data for the current event, the Athena algorithms are run in a specified order. This order is defined in the `AlgSequence` Python object, which is configured for each Athena program execution individually. Each of the Athena algorithms specified in this sequence is executed once per event. More specifically, the `execute()` method of each `AthAlgorithm` instance is called exactly once per event through the Athena framework. In general, the list of executed Athena algorithms depends on the kind of Athena program that is being executed. For ATLAS Monte Carlo simulation the `AlgSequence` gets defined via the job transform scripts. For example, during ATLAS detector simulation usually only the SimulationKernel algorithm (Sections 8.2 and A.2) is being executed, whereas in event reconstruction various different algorithms are being executed.

Athena Tools – `AthAlgTool`: The previously described Athena algorithms may execute common computations or apply common tasks, which will not be implemented in each of the algorithms separately. Thus such tasks are implemented only once in corresponding *Athena tools*. Athena tools are C++ classes which inherit from the Athena `AthAlgTool` class. These tools can be used by different Athena algorithms with different in- and outputs. Contrary to the Athena algorithms, Athena tools can be called multiple times per event. Athena tools are not executed in a predefined sequence, nor are they called by the Athena framework directly.

Athena Services – `AthService`: Athena services are similar to Athena tools, in the sense that they can be run multiple times within a single event. Unlike Athena tools, services usually provide more general tasks. Therefore, the same service might be used by many – or even all – Athena algorithms and Athena tools. Examples for Athena services are the message reporting service, random number generator services or the StoreGate service. Athena services can be implemented by inheriting from the `AthService` C++ class which is provided by the Athena framework.

4.1.2 StoreGate

StoreGate (SG) is a central Athena service which manages transient (T) and persistent (P) data objects needed by one or many Athena algorithms, tools or services. A detailed description of StoreGate can be found in [57]. Here, only the most important features of StoreGate are discussed:

- SG does the memory management for most of the registered data objects. This includes, for example, memory deallocation in case a data object is not needed any more in its transient form.
- SG manages the conversion from transient data to persistent data formats and vice versa. For this, StoreGate utilizes so-called TP-converter classes.
- SG data objects can be accessed by users (i.e. Athena algorithms, tools and services).
- New data types can be implemented and subsequently handled by SG.

The data objects managed by StoreGate are called collections. Examples of typical StoreGate collections are TrackCollections which contain all reconstructed particle tracks and their properties.

4.2 Monte Carlo Simulation Chain and Persistent Data Formats

This section describes the steps required to generate simulated data for the ATLAS detector. In order to allow a straightforward comparison of simulated data with recorded data, common data formats are generated from the event reconstruction output onwards (ESD onwards). A brief introduction into the ATLAS full simulation chain and how to set it up is given in Reference [58]. Further details can be found in Reference [59] and Reference [53]. Though, partly due to this work, some particular details in these references may be outdated but the overall data flow has not changed.

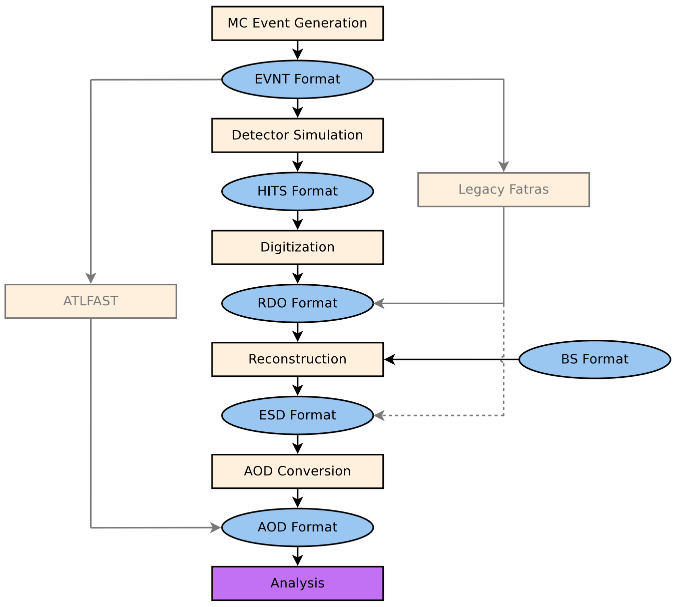


Figure 4.2: ATLAS MC data flow from event generation to AOD files. Elliptically-shaped boxes represent persistent data objects, rectangular boxes are Athena-based algorithms, which are processing the corresponding data objects. The focus of this work (the Integrated Simulation Framework) concerns the detector simulation part of the chain. The image also shows the data flow of the ATLFAST and legacy Fatras fast simulation setups[3]. Though, neither of these two fast simulation setups is currently in use by the ATLAS collaboration (image: [58, 60]).

Figure 4.2 illustrates the data flow in the generation of simulated ATLAS event data. Athena-based algorithms are used to process the data in each individual step. The first step in the chain is the event generation which produces the EVNT output format. The EVNT files are subsequently read in by the detector simulation which generates the HITS output format. The detector simulation output is further processed through the digitization step, generating the RDO format. All subsequent steps are the same for simulated data and recorded detector data. Thus, the event reconstruction step (generating

[3]ATLFAST was used in an early stage of the ATLAS experiment, before particle collisions were recorded by the detector. It is not supported in current ATLAS offline software releases any more. ATLFAST provided very fast (but less accurate) simulation of detector effects for individual physics objects. The legacy Fatras fast simulation existed until ATLAS offline software release 16 and is not supported any longer. ATLFAST and legacy Fatras obtained most of their speedup (compared to the full simulation chain) by combining a number of different steps into one simulation step.

the ESD format) and the AOD conversion step are in general not specific to simulation. Though, additional information may be present in simulation-based ESD/AOD files compared to files generated from detector measurements. MC truth information or links between reconstructed particles and the initial generator particles are two examples of additional information which is present in simulated data only. The following subsections cover each of the steps in the MC chain and the corresponding persistent data formats in detail.

The persistent data formats described in this section are based on a common *POOL ROOT* format which enables the persistent storage of C++ objects. Further information about POOL ROOT can be found in References [53, 61, 62]. This section covers only the most relevant persistent data formats for the context of this thesis.

Individual job transform scripts exists to execute each individual step in the ATLAS Monte Carlo simulation chain:

`Sim_tf.py`
: ATLAS detector simulation using the ISF framework (Section 4.2.2).

`Digi_tf.py`
: Digitization of the detector simulation output (Section 4.2.3).

`Reco_tf.py`
: Reconstruction of the digitization output or the detector readout (Section 4.2.4). This script can also be used to run digitization and reconstruction in one execution step.

`EVNT/HITS/RDO/ESD/AOD/NTUPMerge_tf.py`
: Various job transform scripts used for file merging (the individual data formats are described below).

4.2.1 Event Generation (EVNT Format)

The basis of each event simulation is a physics event generator. Event generators create data objects representing final state particles, much like the LHC is creating final state particles through high-energy particle collisions. These particles are stored in the EVNT data format in the form of HepMC [63, 64] event records. Some event generators also store a tree of intermittent particles into the EVNT file, which connect the initial collision particles with the final state particles. Nevertheless, the EVNT file sizes are in the order of a few ten kilobytes per event. This is much less than other persistent data formats. This format is purely simulation-based, thus no counterpart in the data stream from the real detector exists.

The final state particles are input to the detector simulation (the subsequent step). The particle properties are based on theoretical (and computational) models – such as the Standard Model of particle physics, supersymmetric models, etc. – which are to be tested against measured data. Yet, due to the complexity of the physics processes involved in LHC particle collision, some event generators apply certain simplifications and approximations and must thus be tuned against measured data. Therefore, different tunes may exist which will generally evolve over time. Changing the settings of an event generator usually leads to a re-simulation of the detector response and hence leads to a re-run of *all* steps in the Monte Carlo production chain.

Event generators usually are provided with a set of input parameters, such as the centre-of-mass energy of the initial colliding protons or other physical properties.

Event generators typically in use by the ATLAS collaboration are Pythia 6 [65], Pythia 8 [66] and Herwig/Jimmy [67, 68].

4.2.2 Detector Simulation (HITS Format)

The next step in the Monte Carlo chain is the simulation of interactions between the final state particles created by the event generators and the ATLAS detector. In general, the detector simulation step computes particle trajectories, particle-matter interactions and particle decays within the ATLAS detector volume.

The output format of the most common ATLAS detector simulations is the HITS file format. It contains simulated energy deposits caused by particles traversing the ATLAS sensitive detector elements. HITS file sizes are usually in the order of hundreds of kilobytes per event. As with the EVNT data format, HITS files are purely simulation based, thus, no corresponding file format exists in the ATLAS detector data stream.

Over time, a number of different detector simulation techniques were developer by the ATLAS collaboration. The most detailed and most commonly used method uses the Geant4 [69] toolkit (Section 4.3.1). Other faster methods, such as Fatras (Section 4.3.2) and FastCaloSim (Section 4.3.3) are also being developed, applied and maintained. In order to form simulation setups which are capable of simulating the entire ATLAS detector, various combinations of the different simulation methods exist: ATLFASTII combines FastCaloSim with Geant4 (Section 4.4.1) and ATLFASTIIF combines FastCaloSim with Fatras (Section 4.4.2). ATLFASTII and Fatras are not to be confused with ATLFAST and legacy Fatras, which both are not in use any more. The level of accuracy with which particle interactions with the detector material are simulated, may vary between the different approaches.

The detector simulation usually ends when either all particles have energies below defined thresholds or all particles have left the detector volume.

The work discussed in this thesis, is a framework (the Integrated Simulation Framework) which allows to combine different detector simulation techniques withing one simulated collision event. Thus, enabling a region of interest simulation, in which an accurate simulation technique can be used for particles of interest (e.g. signal particles of a physics analysis) and a fast method can be used for the remaining particles in the event. This approach is demonstrated in a case study for the measurement of the W boson mass with the ATLAS detector in Chapter 12.

4.2.3 Digitization (RDO Format)

After the detector simulation step, the simulated detector hits need to be converted into a data format which corresponds to the format that is retrieved from the detector. This conversion is carried out by the detector digitization step which creates the RDO format (RAW Data Objects).

The digitization aims to transform the previously simulated interactions of particles with sensitive detector material into measurable quantities, such as the free charge drifted to the readout electronics in tracking detectors, or the energy measured in photomultipliers in the tile calorimeter. The RDO format does not store any particle information nor physical interpretation of the simulated detector measurements. The output of the detector itself is in the ByteStream (BS) format, which contains the same level of information as the RDO format. Due to this similarity, RDO data files can be converted into BS files. RDO file sizes are usually in the order of a few megabytes per event. All subsequent steps after the detector digitization are not specific to simulation and they are carried out for recorded data and simulated data in the same manner.

Besides creating simulated detector output signals, event pile-up can be added to the simulated signal event during the digitization step. For this, RAW data objects of minimum bias events are merged with the RAW data objects of the simulated signal event. The minimum bias samples in this case are either pre-simulated (and digitized) events or recorded minimum bias collisions from the detector.

4.2.4 Reconstruction (ESD and AOD Formats)

The RDO data obtained from the previous digitization step or from detector measurements, need to be interpreted in order to identify and measure particle properties. Numerous reconstruction algorithms are executed in order to find particle tracks (connecting detector hits with realistic particle trajectories), measure particle momenta and energies (track curvature and calorimeter measurements) and do particle identification (various methods combining the preceding reconstruction results).

The results obtained by the event reconstruction algorithms contain the fundamental quantities needed for subsequent physics analyses. Reconstructed data are stored in the Event Summary Data (ESD) format. Both, measured data and simulated data, share the same common ESD data format. ESD files contain physics objects which are derived from the information present in simulated (RDO) or recorded (BS) sensitive detector hits. ESD files from simulated data additionally contain Monte Carlo truth information about the simulated event. This allows subsequent physics analyses to associate reconstructed objects with the underlying simulated particles. Apart from the Monte Carlo truth information, ESD files from recorded data and simulated data are indistinguishable. A few common examples for ESD-level physics objects are particle track collections or jet collections. In addition to physics objects, ESD files do contain many sub-detector specific data collections as well. Due to this high amount of information present in the ESD format, file sizes of a few megabytes per event are not uncommon.

The Analysis Object Data (AOD) format is derived from the ESD format and it is the basis for any ATLAS physics analysis. AOD files contain mostly physics objects such as individual collections for various identified particle types. In contrast to the ESD data format, AOD data usually does not contain detailed information about the individual sub-detectors, and thus has a significantly smaller file size of typically a few hundred kilobytes per event.

The ESD data format is currently not intended for long term storage on the ATLAS computing grid (Section 3.7) since the AOD format contains all the objects which are required by subsequent physics analyses.

The xAOD format replaces the AOD format from Run 2 onwards [70]. It is an improved version of the AOD format with is compatible with the Athena framework and with the ROOT data analysis framework [71].

4.3 Full and Fast Detector Simulation

The motivation to develop and use fast detector simulation becomes evident when measuring the simulation time for *single* particle events[4] using Geant4, the most accurate ATLAS detector simulation. Figure 4.3 illustrates that the detector simulation with one initial low-energy electron takes more than one second on average per event. The simulation time increases with higher initial particle energies (due to the increase in the number of generated secondary particles, Figure 4.4) or if the simulated event contains more than one initial particle.

It has been shown in a study of the charged-particle multiplicity for pp interactions at a centre-of-mass energy of $\sqrt{s} = 900$ GeV, that up to $n_{ch} \approx 60$ charged particles (with $p_{\mathrm{T}} > 500$ MeV and $|\eta| < 2.5$) are measured per event in the ATLAS inner detector [72]. For higher centre-of-mass energies, even more particles traverse the ATLAS inner detector: up to $n_{ch} \approx 90$ are measured in charged-particle multiplicity studies with an initial centre-of-mass energy of $\sqrt{s} = 7$ TeV [73]. Both studies account only for particles with $p_{\mathrm{T}} > 500$ MeV, though, the majority of particles are actually generated with energies below this threshold. It is therefore not surprising to measure detector simulation times in the order of a few minutes per event for these types of events using the most accurate ATLAS detector simulation.

After particles have been simulated through the ATLAS inner detector, they will enter the calorimeter sub-system. It has been shown, that more than 90% of the full (Geant4) detector simulation time is generally spent inside the calorimeter [74]. This is due to the high number of secondary particles that are generated when particles traverse the dense ATLAS calorimeter material. Therefore, the full detector simulation will spend the majority of its runtime with the simulation of particle showers inside the calorimeter, which are triggered by the particles leaving the inner detector. As size of the particle showers in the calorimeter increases with higher particle energies (Figure 4.4), consequently the simulation time increases accordingly. Thus, simulation time

[4] *Single particle event:* Events with one initial particle. Detector simulation for such events will usually process more than one particle in the end. This is due to numerous particles created in interaction processes of the initial particle with detector material (e.g. particle showers inside the calorimeter).

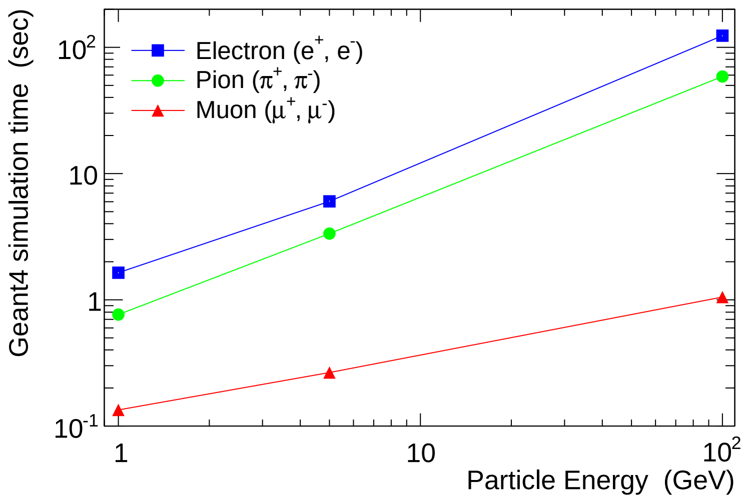


Figure 4.3: Average simulation time in seconds of single particle events using a Geant4-based full ATLAS detector simulation (50k events per data point, ATLAS offline software release 16.0.3.2 with detector description `ATLAS-GEO-16-00-00` on a Intel Xeon X5570 @ 2.93 GHz CPU)

becomes a concern for physics studies which require a high number of simulated events (e.g. supersymmetry studies which need to scan a parameter space in their physical model or W boson mass measurement which requires a very detailed understanding of the detector response to signal particles). A detailed comparison of simulation time for the different ATLAS detector simulations is given in Reference [59].

As described at the beginning of this chapter, detector simulation usually ends when either all particles have energies below defined thresholds or all particles have left the detector volume. The energy thresholds are chosen to balance simulation time with accuracy. Fast simulation energy thresholds are usually much higher than full (Geant4) simulation energy thresholds. This is partly due to the fact that the physics models used in fast simulation are less capable of describing effects on particles with low energies. In many cases, low energy particles will not have a significant impact on a subsequent physics analysis, and therefore it is acceptable for the simulation to not treat these particles any further.

The following subsections give an overview of different detector simulation engines that are used by the ATLAS collaboration. The discussion starts with the most detailed and most accurate simulation: Geant4 in Section 4.3.1. The fast tracker simulation Fatras is discussed in Section 4.3.2, followed by the fast calorimeter simulation FastCaloSim in Section 4.3.3. Section 4.4.1 discusses the ATLFASTII simulation setup, which uses a combination of Geant4 and FastCaloSim to simulate particles traversing the ATLAS detector. Combining different detector simulation engines to form one common ATLAS

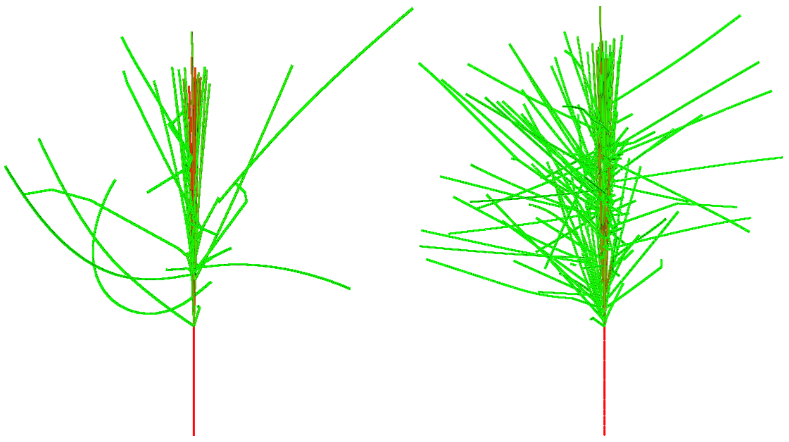


Figure 4.4: Qualitative comparison of particle shower sizes for a pion particle with low energy (left) and high energy (right) when entering the ATLAS calorimeter in a full (Geant4) simulation. The red line illustrates the initial pion particle. The green lines show secondary particles created by the detector simulation due to interaction of the pion with the ATLAS calorimeter detector material.

detector simulation is the basis for this work, the Integrated Simulation Framework (ISF). The details of the ISF design and implementation are covered in part II of this thesis.

4.3.1 Geant4

Geant4 (G4) is a widely used toolkit for the simulation of particles traversing matter. It is developed by the Geant4 collaboration [75]. Its main applications are found in detector simulation for high energy physics experiments, radiation simulation in medical sciences and a number of space physics projects.

Geant4 is able to simulate interactions for a variety of particle types with many different materials over a wide range of particle energies. The software is based on different physics models describing many different kinds of particle-matter interactions. Furthermore, Geant4 also simulates decays of unstable particles. Geant4 allows for application-specific geometrical descriptions of the material (or detector) that is to be simulated.

Geant4 provides a number of parameters which offer the ability to fundamentally change the simulation's behaviour and output. Parameters, such as the physics models to be used (physics list), the simulated physics effects, stepping sizes etc. may affect the computing performance and accuracy of the Geant4 simulation.

Due to its long-time operating experience, Geant4 has become a highly validated and sophisticated simulation for many particle physics experiments. Therefore Geant4 is the main (and most accurate) detector simulation in use by the ATLAS collaboration. This comes, however, with the price of an immense demand for computing resources (Section 1.1).

The Geant4 toolkit is commonly used by the ATLAS collaboration and often referred to *full* detector simulation. A Python and C++ based Geant4 simulation framework for the ATLAS detector (named FADS) [59] has been implemented even before the use of the Athena software framework was adopted by the collaboration. The FADS framework is used since then and is still configuring the Geant4 toolkit to the specific needs of the ATLAS detector simulation. The integration of the Geant4 toolkit into the ISF framework is discussed later in this document in Section 9.1.

4.3.2 Fast ATLAS Track Simulation (Fatras)

Fatras [76–78] (Fast ATLAS Track Simulation) is a fast ATLAS detector simulation for the inner detector, the muon spectrometer and partially for the calorimeter. Fatras was re-implemented in the context of this work, however, the underlying fast simulation principles did not change. Thus, most items covered in this section apply to the legacy Fatras and the current Fatras implementation.

In Fatras, particle tracks are simulated with the standard ATLAS reconstruction tools. The detector geometry is based on the *reconstruction geometry* [79], which is a much more simplified detector description compared to the detector geometry used for full Geant4 simulation. The reconstruction geometry describes the ATLAS detector material almost entirely by a few thin and discrete layers of different materials (Figure 4.5). The layers are arranged such that the overall material effects of the detector on the traversing particles is reproduced accurately. A dense volume description of the material inside the calorimeter was recently introduced, to allow the simulation of muon particles through the calorimeter towards the muon spectrometer more accurately.

In addition to the simplifications in the material distribution, Fatras uses *averaged* materials which do not necessarily correspond to any real physical material which is built into the detector. Particle interactions with these averaged materials are computed either by parameterized algorithms (which are Fatras-specific) or by accessing routines from the Geant4 library. For the latter case, the average material parameters are converted into the closest physically existing material and subsequently, Geant4 routines[5] are used to compute the products of inelastic particle-material interactions.

Due to the techniques described above, Fatras detector simulation executes about a factor 100 faster in the ATLAS inner detector compared to Geant4-based full detector simulation.

Compared to Geant4, Fatras gains additional computing time by spending less time on the simulation (or creation) of secondary particles which are produced in particle-material interactions. Clearly, this comes at the price of a less accurate description of the processes occurring in the physical ATLAS detector. However, Fatras still shows good overall agreement with the full Geant4 detector simulation and with recorded data

[5]The Geant4 routines to compute hadronic interactions have proven to have sufficiently high execution speeds for their use in fast simulators.

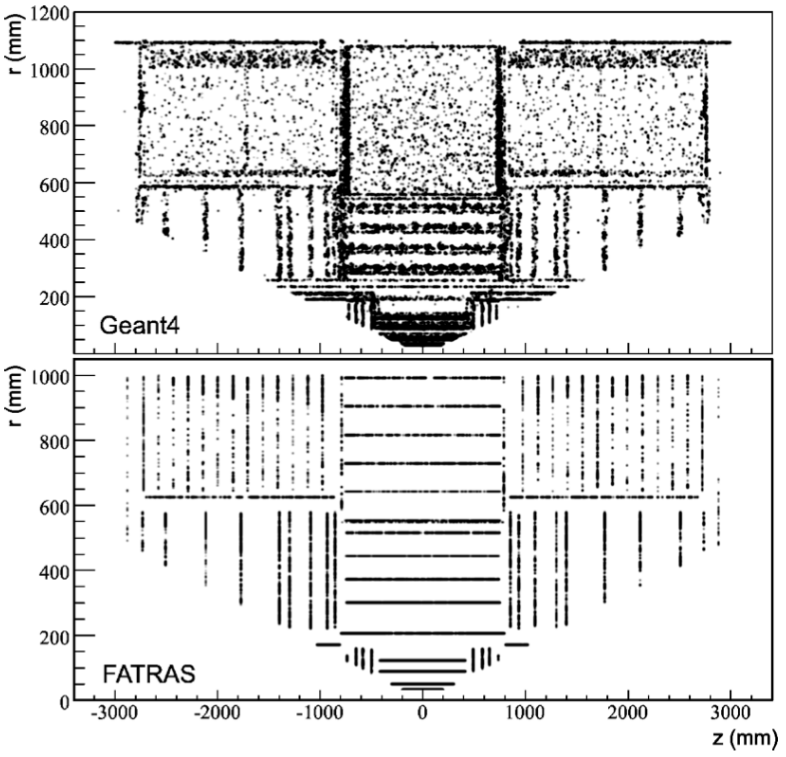


Figure 4.5: Photon conversion points shown in Geant4 and Fatras. Since photon conversion processes are enabled by dense material both depictions show the distribution of material inside the detector simulation. The discrete and thin layers of material used in the Fatras simulation are clearly visible (image: [45]).

from the ATLAS detector (Figure 4.6).

The legacy Fatras implementation did replace parts of the standard detector digitization algorithms and computed an approximated detector response in the fast simulation step. In addition, it was capable of creating reconstruction-level output if needed (Figure 4.2) . In order to fulfill the requirements of the new simulation framework (Section 7.1.1), the output of the current Fatras implementation is the same format as the Geant4 simulation output and thus feeds into the same digitization chain. The fast digitization modules of the legacy Fatras implementation have been encapsulated and are being prepared to run in a dedicated fast-digitization chain, where the simulated hits from any simulator may be processed.

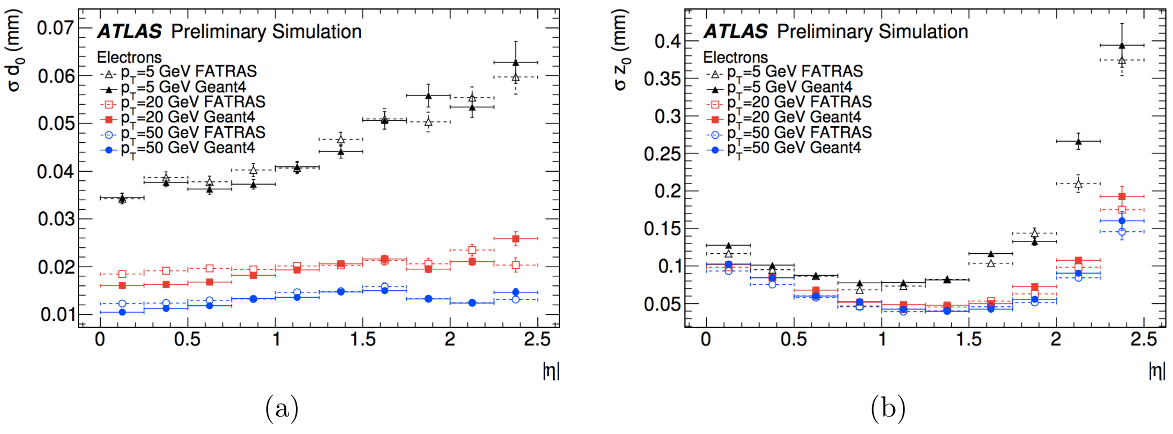


Figure 4.6: Reconstructed track resolution for electrons simulated with Fatras at different particle momenta ($p_{\mathrm{T}} = 5, 20, 50$ GeV): (a) d_0 track parameter resolution (b) z_0 track parameter resolution (images: [78]).

4.3.3 FastCaloSim

FastCaloSim is a parameterized ATLAS calorimeter simulation [74, 78, 80, 81]. Its parametrization is based on Geant4 ATLAS detector simulation results. The parameterized simulation approach offers various parameters that allow to tune the simulation output against other references. In current ATLAS Monte Carlo production campaigns, a FastCaloSim tuning against data is used, as it allows to describe certain aspects of the calorimeter response even more accurately than full Geant4 detector simulation (Section 4.4.1 and Figure 4.7).

As FastCaloSim is restricted to ATLAS calorimeter simulation only, it is usually combined with other simulators (such as Geant4 or Fatras) to form a complete ATLAS detector simulation. The combination of Geant4 and FastCaloSim is called ATLFASTII and is discussed in the following Section 4.4.1. The combination of Fatras and FastCaloSim is called ATLFASTIIF and is discussed in Section 4.4.2.

4.4 Combined ATLAS Detector Simulations

In contrast to the full Geant4 detector simulation, which is capable of simulating tracking detectors and calorimeters, fast ATLAS detector simulators are specific to certain ATLAS sub-detectors only. Thus, in order to simulate the entire ATLAS detector using fast simulation, a combination of different simulators is required. The following subsections discuss different combinations of full and fast simulations. Geant4 combined with FastCaloSim is discussed in Section 4.4.1, Fatras combined with FastCaloSim in Section 4.4.2.

4.4.1 ATLFASTII

ATLFASTII is a fast ATLAS detector simulation utilizing the FastCaloSim ATLAS calorimeter simulation. The Geant4 toolkit is used for ATLAS inner detector and muon spectrometer simulation. Any muon particle traversing the ATLAS calorimeter is simulated through Geant4 as well. This setup results in an ATLAS detector simulation which is about ten times faster than the full Geant4 ATLAS detector simulation [59].

The ATLFASTII simulation begins with the simulation of the final state particles from the event generator EVNT file through the ATLAS inner detector volume using Geant4. Each particle leaving the ID volume is checked for its particle type and only muons are further processed by the Geant4 simulation in the calorimeter. All particles above an energy threshold will cause energy deposits in the calorimeter, which are computed by the FastCaloSim module. Since FastCaloSim does not perform a particle transport, muon particles are the only particle type able to enter the ATLAS muon system in the ATLFASTII simulation setup. Consequently, internal processes in the calorimeter, such as decays and punch-through are not simulated in ATLFASTII.

Due to the use of Geant4 to simulate the ID and MS in ATLFASTII, the standard ATLAS reconstruction algorithms can be used to reconstruct particle information for these sub-detectors. Moreover, the FastCaloSim output format can be converted into the Geant4 calorimeter simulation output format. Therefore, the calorimeter reconstruction can be carried out by the standard ATLAS algorithms as well.

Tuning parameters inside FastCaloSim allow to tune the entire ATLFASTII simulation output against any reference. One commonly used ATLFASTII-tune uses recorded detector data as the reference. With this tune, in particular calorimeter-specific quantities are in better agreement between ATLFASTII and recorded data as it is the case for the Geant4 detector simulation output (Figure 4.7).

The ATLFASTII implementation used by the ATLAS collaboration prior to the implementation of the ISF, required two separate detector simulation steps to be executed within the ATLAS Monte Carlo chain. The first step processed the Geant4 detector simulation and the second step processed the FastCaloSim calorimeter response. Within the ISF, both simulators are executed within one step, thus lowering complexity of the Monte Carlo production chain and improving computing performance at the same time.

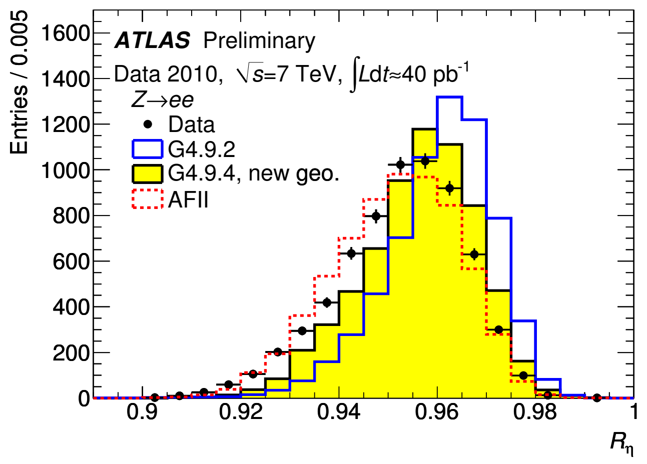


Figure 4.7: Energy ratio R_η of a cluster of $\Delta\eta\Delta\phi = 3 \times 7$ cells with respect to a cluster with 7×7 cells in the bulk EM calorimeter layer 2. The image shows the improved agreement between recorded detector data and Geant4 ATLAS detector simulation due to a newer Geant4 software version and an updated detector geometry description. Clearly visible is the effect of tuning the FastCaloSim simulation inside ATLFASTII against data, which provides an even greater accuracy of the simulation output than full Geant4 detector simulation (image: [82]).

4.4.2 ATLFASTIIF

ATLFASTIIF is a fast ATLAS detector simulation, which combines the fast simulators FastCaloSim and Fatras to form a complete ATLAS detector simulation. It is in many ways similar to ATLFASTII, however, instead of Geant4 it uses Fatras to simulate the tracking detectors and muon particles. Thus, Fatras is used for the inner detector and the muon spectrometer simulation, as well as muon particles inside the calorimeter. All other particles entering the calorimeter are simulated with FastCaloSim. This combination results in a complete ATLAS detector simulation which is about one hundred times faster than the full Geant4 ATLAS detector simulation [59].

ATLFASTIIF is under validation and it was therefore never used in ATLAS Monte Carlo production campaigns so far.

Part II

The Integrated Simulation Framework

Chapter 5

The Vision

A number of full and fast detector simulation approaches are used in high energy physics experiments in order to cope with the high demand for Monte Carlo simulation samples. Each type of detector simulation balances accuracy and execution speed in a distinctive manner (Figure 5.1). A general, rather trivial relation is that more accurate simulations require more computing resources and are thus slower. A common ATLAS simulation framework is developed which allows to combine the different simulations and choose an appropriate simulation technique for each individual particle in the detector simulation. This chapter covers the specific requirements and design choices for the implementation of such a framework for the ATLAS experiment. The full list of requirements are realized in the ATLAS Integrated Simulation Framework (ISF).

The simulation techniques developed by the ATLAS collaboration (also referred to as simulators) were previously introduced in Section 4.3. Before the implementation of the ISF, many of these simulation techniques were mutually exclusive and could not be applied in combination within one simulated event. Mixing different full and fast simulation techniques for different particles within one event allows to achieve a balanced combination of high simulation accuracy and high execution speed for the entire event. Highly accurate simulation (with low simulation speed) may be required only for those particles which play a significant role in subsequent physics studies. Lower simulation accuracy (high simulation speed) may be acceptable for the remaining particles in the event.

One type of combined detector simulation approach has proven to be successful in a number of ATLAS Monte Carlo production campaigns throughout the previous years: ATLFASTII. ATLFASTII combines the most accurate simulator Geant4 in the inner detector with the fast calorimeter simulation FastCaloSim. As the individual simulation techniques in ATLFASTII are only applied globally within the respective sub-detectors, particles can not be selected individually for either accurate or fast detector simulation. However, this might be a requirement in the detector simulation of signal events for physics studies. A highly accurate simulation of the signal decay products may be needed, in order to study the initial signal particle with the highest accuracy available. The remaining particles in an event may be simulated with a less accurate,

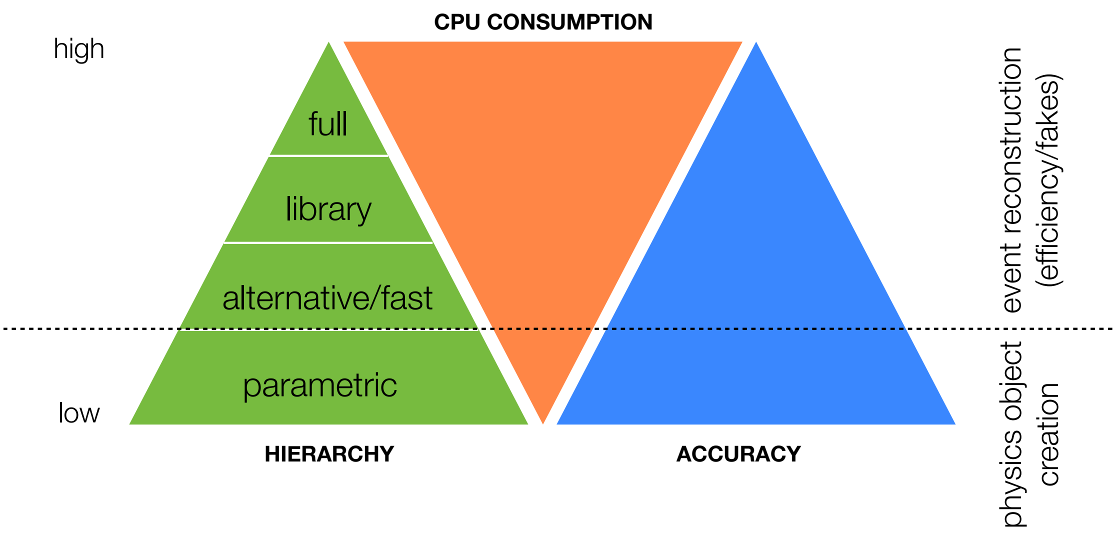


Figure 5.1: The detector simulation hierarchy. Each simulation method used by the ATLAS collaboration (green) distinctively balances simulation accuracy (blue) and execution speed (orange) (image: A. Salzburger).

fast simulation. While maintaining the highest possible accuracy where it is needed, this approach increases the execution speed of the overall detector simulation due to using fast simulators where less accuracy is required. Neither the ATLFASTII nor the ATLFASTIIF simulation approach provide the flexibility to configure the individual simulators as required by the example given above.

The first set of requirements for an Integrated Simulation Framework emerge from this limitation present in ATLFASTII. The ISF must allow for various combinations of full and fast simulation techniques to be used for individual particles within one event. The framework must also allow to define a number of rules that determine which simulator is to be used for each particle in the event. Figure 5.2 shows a sketch for an example ISF setup, where electrons are the particles associated with the physics signature that is studied with this simulation. Therefore they are simulated in the most accurate simulator, Geant4. The rest of the event (with the exception of muon particles) is simulated in the Fatras or FastCaloSim fast simulator.

In addition to allowing for a flexible simulator configuration, the ISF must serve as the common ATLAS detector simulation framework which is to be used for any type of ATLAS detector simulation. As such, it must implement core ATLAS detector simulation functionalities, which are to be used by all simulators, independent of the specific detector simulation setup. Among the core functionalities of the simulation framework are the processing of the simulation input and output and the Monte Carlo truth recording. Figure 5.3 illustrates the components of the ISF and its role as an ATLAS detector simulation framework. The first step within the ISF is the input processing, during which beam conditions and particle filters are applied to the particle collection created by the event generator. The ISF routing algorithm identifies the best suited simulator for each of the particles that passes the filter criteria. These particles are subsequently sent to the respective simulator for detector simulation. Simulators may return child particles to the particle router, which will again identify the best suited simulator for each of the particles. The sensitive detector (SD) hits computed by simulators are recorded directly into the respective StoreGate collection. Simulators register particle interactions and decays to a central Monte Carlo truth recorder. The MC truth recorder generates a consistent MC truth representation of each simulated event and stores it in the corresponding StoreGate collection. Particle barcodes (and interaction vertex barcodes) are generated by a barcode generator which is independent from the ISF implementation. The barcodes are provided to the simulator and the MC truth recorder, where they are stored in SD hits and the MC truth representation, respectively. Standardized ISF interfaces are used to exchange information between the simulators and the framework. This enables the integration of various types of simulators into one common framework. Even future simulators (e.g. fully parametric simulators) can be integrated through the use of these interfaces. Changes to the ISF code base are therefore not required when the functionality of individual simulators change or new simulators are integrated into the framework.

As a result of introducing the ISF as a common simulation framework, the two separate detector simulation steps which were previously required for the ATLFASTII setup are now combined in one simulation step. This is possible as Geant4 and FastCaloSim are registered as two independent simulators within the ISF, with each processing different particles of the event.

The following sections cover the specific requirements to the ISF in more detail.

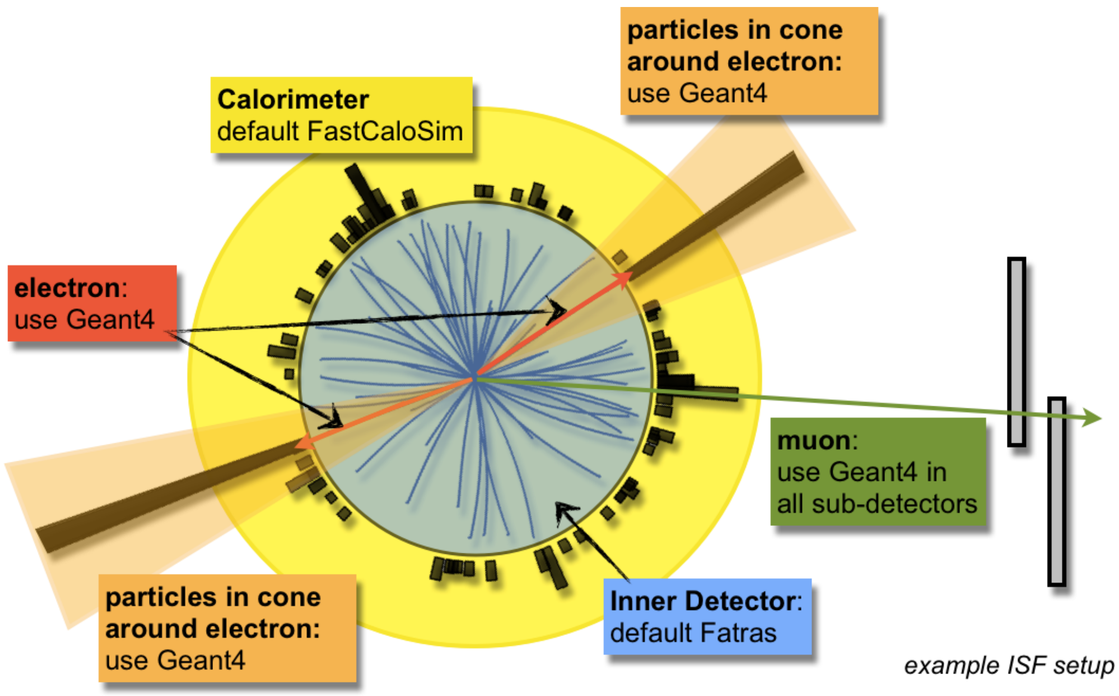


Figure 5.2: An example ISF simulation setup. The electrons in this example are the particles associated with the physics signature that is studied with this simulation. Hence, they require a highly accurate detector simulation, which is chosen to be Geant4. Since particles close to the electrons may impact their measurement or may be relevant for subsequent physics analyses, a region of interest is defined around the electrons. All particles in cones around the electrons are therefore also simulated in Geant4. As muon particle simulation in Geant4 is much faster than the simulation of other particle types, muon particles in the event are also processed by Geant4. The rest of the event is processed by fast simulations: Fatras in the inner detector and FastCaloSim in the calorimeter.

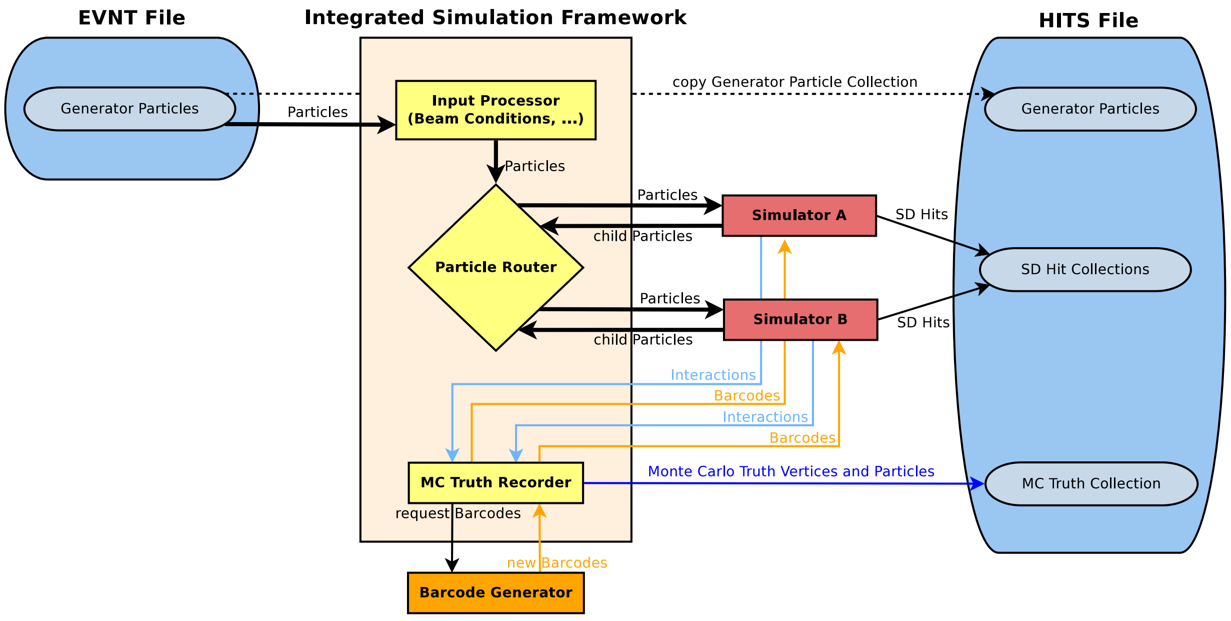


Figure 5.3: The data flow in ATLAS detector simulation with the Integrated Simulation Framework. At the start of each event, the ISF prepares and filters the event generator particle collection for detector simulation. The particle router identifies the appropriate simulator for each particle in the detector simulation. Particles which are generated inside the simulator by interaction or decay processes may be returned to the ISF for re-routing. Simulators record the computed sensitive detector hits (or higher-level simulation output) directly to the corresponding StoreGate collection. All simulators register particle decays and interactions to a common Monte Carlo truth recorder, which generates a MC truth representation of the simulated particles. An external barcode generator is used to compute particle and vertex barcodes, which are stored in SD hits and in the MC truth representation, respectively.

5.1 Common Simulation Framework

The ISF is designed to serve as the detector simulation framework which is to be commonly used for any ATLAS detector simulation – full and fast. As such, the framework has core responsibilities, which are required by all simulators in any simulation setup:

Common Simulation Tasks : The framework implements central services for input processing (considering beam conditions), Monte Carlo truth recording and particle and vertex barcode handling. Interfaces and fundamental data types (such as the ISF particle representation) are implemented inside the framework, in order to allow a simulator-independent data flow between components of the ISF and the individual simulators.

Simulator integration : The ISF supports the integration of all currently existing ATLAS detector simulators: Geant4, Fatras and FastCaloSim. As the individual simulator implementations are (to a large extent)[1] independent from one another, they are realized in separate libraries.

Simulation setup and configuration : Any ATLAS detector simulation is initiated and configured through the ISF. The ISF allows for selecting a predefined detector simulation configuration or setup through a string identifier (i.e. the simulation setup name). Athena ConfigurableFactory methods (also called ConfiguredFactory or ConfGetter) [83] are used to instantiate and configure the Athena algorithms, tools and services which involved in the detector simulation process in accordance with the chosen simulation setup. A modular implementation of the ISF configuration is possible due to using `ConfiguredFactory` methods.

5.2 Simulation Flavours and Simulator Mixing

A key innovation of the ISF is its flexibility to combine various simulators into one ATLAS detector simulation setup. This is possible as the framework can process particles through one or many different simulators within one event. This flexibility is enabled due to the ISF particle routing algorithm which fulfills the following requirements:

- **Support existing simulation flavours** : The ISF supports the traditional full Geant4 and ATLFASTII fast simulation flavours. This allows a transparent adaptation of the framework for official ATLAS MC production, replacing the two previously independent simulation configurations.

- **Region of Interest (ROI)** : The simulation framework supports user-defined regions of interest within the simulated events. Different simulators may be used

[1]Cross-dependencies, such as e.g. the Fatras simulator using decay and hadronic interaction processes from Geant4, are separated out into dedicated packages in order to isolate such dependencies during build-time.

within different regions of interest. Separate ROIs may be defined per ATLAS sub-detector. Thus, allowing to choose an appropriate simulator for detector simulation within each respective sub-detector.

- **Dynamic ROI** : Regions of interest may be dynamically defined, based on the topology of the respective simulation input event (i.e. the generator output).

5.3 Integration into ATLAS Monte Carlo Production

To guarantee a swift integration of the ISF into the ATLAS offline software framework and the existing Monte Carlo production chain, the following criteria are considered in the ISF design:

Athena integration : The ISF is implemented within the ATLAS offline software framework Athena. This implies the use of `AthAlgorithms`, `AthServices` and `AthAlgTools` for the implementation and configuration of all ISF tasks and algorithms.

Simulation output format : The detector simulation output format is compatible with the standard ATLAS digitization and reconstruction steps. If the standard digitization and reconstruction algorithms are to be used for the output of any particular detector simulator, this simulator is required to generate the corresponding sensitive detector hits output format. Multiple simulators within the ISF may be creating SD hits within one event, thus the corresponding SD hit collection will contain a combination of hits from each of the simulators.

Job transform script : Each individual step in the ATLAS Monte Carlo production chain is steered through a respective job transform script. The ISF is therefore configured through a dedicated job transform script named `Sim_tf.py`. Any form of ATLAS detector simulation can be configured and executed through this script.

5.4 Extensibility

With the central role of the ISF in ATLAS Monte Carlo production, it is essential that the framework is designed for compatibility towards future developments and extensions to its functionality. Therefore the following criteria are implemented and considered in the ISF design:

Extensibility of simulators : The framework allows for the integration of new simulation techniques or simulators alongside the currently existing simulators. As the individual simulators in the ISF are integrated through ISF-specific interfaces, new simulators can be added by implementing these interfaces into the respective simulator.

Concurrency readiness : Concurrent processing is currently not exploited in ATLAS detector simulation. However, due to current trends in the processor market, concurrency may become an essential requirement in the near future. The modular design of the framework allows for individual components to be replaced by components which support concurrent processing techniques.

5.5 ISF within a Fast ATLAS Monte Carlo Chain

The ISF is one of many steps towards improving the computing performance of the entire ATLAS Monte Carlo production chain. Fast digitization and fast reconstruction techniques are currently being developed and validated for use in official ATLAS Monte Carlo production campaigns [84, 85].

In a first phase, the ISF is to be integrated into the existing MC production chain. The ISF uses full or fast simulators to process the event generator output through the detector and, from this, generates detector simulation output which is subsequent processed by the standard ATLAS detector digitization. In the second phase, the ISF may feed its output into fast detector digitization and reconstruction algorithms. This development, however, is independent of the detector simulation framework. In the third phase, very fast (parametric) detector simulators will be integrated into the ISF. These simulators generate reconstruction-level output for the particles they simulate, thus subsequent digitization and reconstruction steps are not required.

In either of the phases described above, the ISF serves as a common framework for any of the detector simulators (full, fast, parametric).

Chapter 6

ISF Particle Routing

Based on the requirements stated in the previous chapter, a completely new algorithmic approach is needed to solve the problem of identifying the appropriate simulator for each particle in a simulated event. The algorithm must consider user-defined simulation selection rules which may even change from one simulated event to another. The problem becomes increasingly complex with more simulators or regions of interest that are defined in a detector simulation setup. In particular with some simulators (such as Geant4) offering full and fast simulation approaches within the same implementation, which may be seen as two individual simulators in the end. Since this problem concerns the routing of particles through the simulation framework into the simulators, it is therefore also called the *particle routing problem*.

Figure 6.1 illustrates the particle routing problem in simplified terms. Initially, a set of particles (ISF particle collection) is available that needs to be simulated through the ATLAS detector. A particle routing algorithm is required to find the appropriate simulator (among a number of different simulators) for each of the particles. Evidently, a unique solution is required for each particle in order to guarantee a reproducible behaviour of the simulation framework. While the initial particles are being simulated, new secondary particles may be returned by the simulators to the ISF particle collection. Same as for the initial particles, the routing algorithm must find an appropriate simulator for each of the secondary particles in the ISF particle collection.

In the following sections, various different routing classifications (Section 6.1) and their algorithmic realizations (Section 6.2) are discussed .

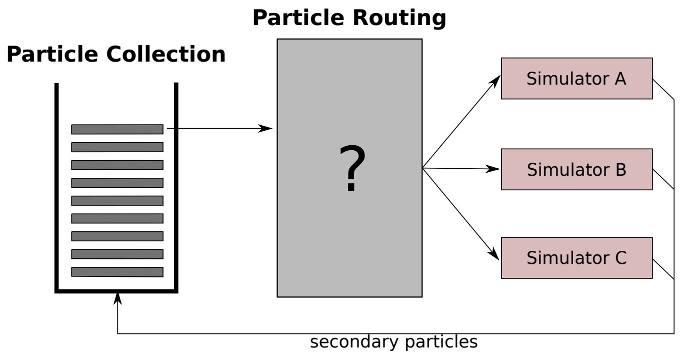


Figure 6.1: The routing problem which needs to be solved in the ISF. Starting with an initial list of particles, a particle routing algorithm must decide on the simulator to use for each particle. The simulators may generate secondary particles which will be added to the particle collection. For these, the routing algorithm must also determine the appropriate simulators. The process is re-iterated until no more particles are returned by the simulators and thus, all particles have been simulated through the ATLAS detector.

6.1 Routing Rules

Routing rules are user defined rules which are utilized to determine a detector simulator for each particle within the ISF. For a given particle P, a routing rule i may or may not suggest one simulator S_i which is to be used for the simulation of this particle. For any condition C_i and a defined simulator S_i, any routing rule i will follow this structure:

- If the given particle P fulfills condition C_i then use simulator S_i.

For a given routing rule, the simulator S_i is a constant throughout the execution of a detector simulation job.

The following list shows a few illustrative examples of routing rules (in fact these rules are *static* routing rules, which are discussed in detail in the following subsection):

1. Simulate all muon particles with Geant4.
2. Discard all particles with a momentum vector orientation of $|\eta| > 5$ from detector simulation.
3. Simulate all particles inside the ATLAS inner detector with Fatras.
4. Simulate all particles inside the ATLAS calorimeter with FastCaloSim.

Typically, a combination of various different routing rules are used to form a realistic detector simulation setup. However, different individual rules may suggest different simulators and thus may contradict each other. For example, if one were to combine rule 1 and 2 from the examples above, it becomes evident that the rules give contradicting results for a muon particle travelling towards the forward ATLAS detector region with

$\eta > 5$. To solve such conflicts two solutions are considered relevant for the purpose of the ISF:

Prioritization of routing rules : The individual routing rules are given in a prioritized order. In the case of conflicts between routing rules, the decision of the rule with a higher priority is favored. For the example above, this would mean if rule 1 has a higher priority than rule 2, then a muon in the forward region would be sent to Geant4 rather than being dropped from detector simulation.

Prioritization of simulators : Alternatively, one may prioritize the individual simulators rather than the routing rules. For this, all routing rules are evaluated for a given particle and one simulator is chosen among all the suggested simulators. If there are no conflicting rules for a given particle, a maximum of one simulator is suggested by any set of routing rules. In this case, the one simulator is chosen for the simulation of the given particle. If there are conflicting rules for a given particle, more than one simulator will be suggested. The chosen simulator will be the one simulator with the highest priority among the suggested simulators. In the example above, if Geant4 were given a higher priority than not simulating a particle at all, a combination of rule 1 and 2 would send a muon in the forward detector region to Geant4.

The prioritization of routing rules approach was favoured for the purpose of the ISF, due to its more intuitive behaviour.

Whether a routing rule i suggests a simulator S_i or not for any one physical particle, may change during the detector simulation. The outcome of a routing rule can change if either the condition C_i changes (discussed in the following subsections) or the particle itself changes. Apart from fully parametric detector simulations, simulators will continuously update the particles involved in the simulation until a set of final criteria are met and the simulation is terminated. Consequently, these particles will change and if a given routing rule is consulted for the very same physical particle at different times, the outcome may differ. Two simple examples are: a particle which has been propagated until a geometrical condition in C_i is not fulfilled any longer, or a particle has lost energy due to ionization such that an energy threshold in condition C_i is not fulfilled any longer. For the implementation of any routing algorithm, it must be decided, whether continuously changing simulator decisions for the same physical particle are acceptable or whether this is to be avoided. In the context of the ISF routing implementation, it was decided that for each particle one simulator decision per ATLAS sub-detector is sufficient (see Section 6.2 for details).

The amount of information that is accessed to test condition C_i, varies between the different classes of routing rules. The various classes are described in detail in the following subsections: Purely static rules, which do not change from one simulated event to another, are discussed in Subsection 6.1.1. The routing rules discussed in Section 6.1.2 will change depending on the event topology of the simulation input. Eventually, Section 6.1.3 discusses fully dynamic routing rules, which update their conditions during the detector simulation of one event.

6.1.1 Static Routing Rules

Static routing rules are a class of routing rules for which the according conditions C_i do not change throughout the execution of a detector simulation job. In addition, such rules must only take into account the information associated with the particle P for which a routing decision has to be made. If information about other particles in the event is used to make a routing decision for any one particle, the rule would not be static and thus the implications discussed in this section would not apply.

Static routing rules can be used to define regions of interest within the ATLAS detector or with respect to particle kinematics and properties. For instance in the list of examples provided above, rule number 2 defines a region of interest within the ATLAS detector for the pseudorapidity range of $|\eta| > 5$. It is possible for a particle to start within the ROI and, during the process of the detector simulation, to leave this ROI. How this effect is taken into account, depends on the specific implementation of the routing algorithm. As discussed previously, the ISF evaluates routing rules for each particle once per ATLAS sub-detector. Thus, if a particle were to leave an ROI that is defined by a static routing rule within one sub-detector, it would remain within the same simulator that processed it to this point.

In order to avoid losing particles in the detector simulation, rules selecting a default simulator may be defined for each detector region. Such rules do not put any restrictions on a particle and will always suggest a simulator for a given detector region. In the routing examples provided above, rules 3 and 4 are default simulator rules. These rules guarantee that each particle within the ATLAS inner detector and calorimeter will indeed be simulated, even if no other rule applies to a given particle.

Prioritized static routing rules can be used to form the traditional ATLAS detector simulation setups (low number means high priority):

Full Geant4 detector simulation:

1. Simulate all particles with Geant4.

ATLFASTII fast detector simulation:

1. Simulate all particles in the ATLAS inner detector with Geant4.
2. Simulate muon particles in the ATLAS calorimeter with Geant4.
3. Discard all particles with a momentum vector orientation of $\eta > 5$ in the ATLAS calorimeter from any detector simulation.
4. Simulate all particles in the ATLAS calorimeter with FastCaloSim.
5. Simulate all particles in the ATLAS muon spectrometer with Geant4.

ATLFASTIIF very fast detector simulation:

1. Simulate all particles in the ATLAS inner detector with Fatras.
2. Simulate muon particles in the ATLAS calorimeter with Fatras.

3. Discard all particles with a momentum vector orientation of $\eta > 5$ in the ATLAS calorimeter from any detector simulation.
4. Simulate all particles in the ATLAS calorimeter with FastCaloSim.
5. Simulate all particles in the ATLAS muon spectrometer with Fatras.

Due to the flexibility and simplicity of static routing rules, the ISF routing algorithm does support static routing rules (Section 6.2.1).

6.1.2 Semi-Dynamic Routing Rules

Semi-dynamic routing rules are a class of routing rules, for which the condition C is (at least partly) based on information from the initial particles in the simulation input event (the EVNT format, Section 4.2). A semi-dynamic routing rule must not take into account particles outside the simulation input of the current event.

In consequence, condition C must be set at the start of the detector simulation for each event respectively. It will stay the same throughout the simulation of this event. Before the start of the simulation of the next event, condition C must be reset and updated according to the new input event. At this point, condition C must discard all information that is held about the previous event. If this information is not discarded, the detector simulation framework that is using such a routing rule, may introduce correlations between independent events.

Semi-dynamic routing rules can be used to cover a wide range of possible applications. Most commonly, they are used to define regions of interest within an event, based on the simulation input. For example, such ROIs may be defined in cones around signal particles in the EVNT file. One application of semi-dynamic routing rules is studied in detail later in this thesis in Chapter 12.

Figure 6.2 illustrates that ROIs which are defined by the initial particles can lead to changing simulator decisions during the course of the detector simulation. Thus, if the semi-dynamic routing rules were strictly enforced, the routing rules would need to be re-evaluated for each single simulation step of each particle. As mentioned previously, the approach taken in the ISF routing design evaluates the routing rules for each particle only once per ATLAS sub-detector.

A region of interest, which is defined solely by the initial conditions of a detector simulation event (i.e. the generated event), can not adjust to changing conditions which may occur during the detector simulation. Figure 6.3 illustrates a case where semi-dynamic routing rules are defined by the initial particle states. During the course of the detector simulation, the particle defining the ROI leaves this region. According to the definition of semi-dynamic routing rules, the ROI can not be updated during the simulation of one event (this is possible with dynamic routing rules, see Section 6.1.3). This example shows that care must be taken when defining the ROIs using initial particle states. In this case, either the cone size could be increased, or alternatively, cone-shaped ROIs could only be defined around particles with a high enough momentum, such that they are unlikely to bend out of their own ROI.

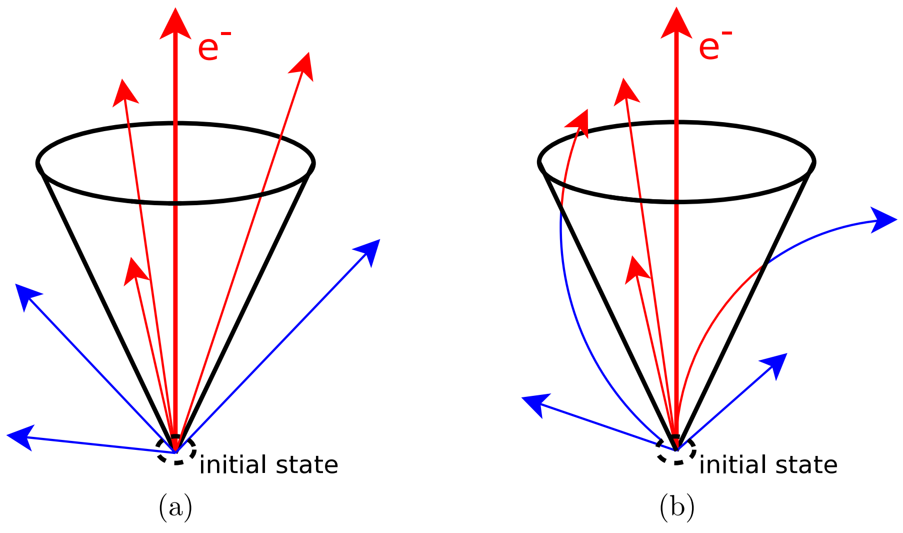


Figure 6.2: A region of interest which is defined solely by the initial conditions of a detector simulation event, can lead to changing simulator decisions during the evolution of the detector simulation. Figure (a) and (b) show a cone-shaped ROI which is defined by the initial condition of the electron in the event, respectively. Semi-dynamic routing rules are used to determine the simulator S for particles inside this ROI (red) and other routing rules are used outside (blue). Figure (a): The particles involved continue to stay within the ROI throughout the detector simulation. Thus, the simulator decisions will not change. Figure (b): The particles bend out of and into the cone ROI. Thus, the simulator decision for these particles may change, if the semi-dynamic routing rules are re-evaluated for these particles.

The fact that particles do not change their simulator when bending out of cone-shaped ROIs is desirable for subsequent processing steps and physics analyses. In the ATLAS inner detector, the particle measurement in the perigee representation (i.e. point of closes approach to a particular decay vertex or beam-interaction region) is most relevant for physics analyses. If the perigee position of a particle is close to a vertex of interest, the particle may be relevant for the analyses. In such a case, the particle will initially be located within the cone-shaped ROI. The perigee measurement, however, is derived from *all* sensitive detector signals of this particle throughout the entire ATLAS inner detector. In order to achieve the desired accuracy in this particle's perigee measurement, it is important that *all* inner detector hits are created by the simulator which is to be used within the ROI – even if some hits may be located outside the ROI. Thus, if a particle starts within a cone-shaped ROI and bends out during the detector simulation, it is simulated with the desired simulator close to the perigee position and throughout the rest of the inner detector simulation.

Due to the flexibility of semi-dynamic routing rules, the ISF routing algorithm does support semi-dynamic routing rules (Section 6.2.2).

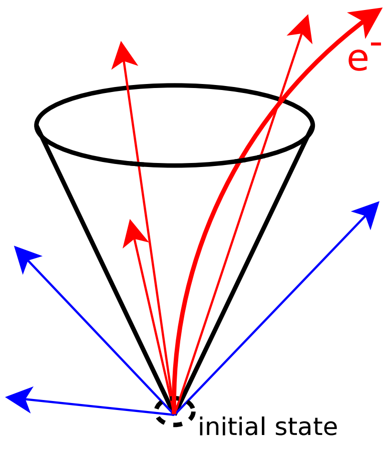


Figure 6.3: In this example, a cone-shaped region of interest is defined around the initial state of an electron in the detector simulation input event. Due to the magnetic field present in the ATLAS detector, the simulated electron trajectory leaves the ROI that was defined around its initial state. An increased cone size, or alternatively, a minimum momentum requirement for particles creating cone-shaped ROIs, may prevent particles from bending out of their own ROIs.

6.1.3 Dynamic Routing Rules

Dynamic routing rules are a class of routing rules for which the according conditions C_i may change during the detector simulation of a given event. Dynamic routing rules can be used to define and update regions of interest in accordance with changing conditions within a detector simulation event. Figure 6.3 shows an example where the particle that initially defines the ROI in a semi-dynamic routing rule, is leaving this ROI during the detector simulation. With dynamic routing rules, the ROI can be adjusted to the actual trajectory of the initial particle as it is simulated through the ATLAS detector.

Another example for the use of dynamic routing rules is shown in Figure 6.4. In this case, a dynamic routing rule creates cone-shaped regions of interest around each electron and positron in the event. The one electron which is among the initial particles defines the first ROI (Figure 6.4a). The detector simulation for the photon particle, computes a conversion into an electron-positron pair [86] (Figure 6.4b). Thus, the dynamic routing rule defines new ROIs from the resulting electron-positron pair (Figure 6.4c). As a consequence, the pion particle will now fulfill condition C of the dynamic routing rule and thus be sent to simulator S of this rule.

The example above shows, that the order in which particles are simulated has a direct impact on which particles are selected by dynamic routing rules. If the pion particle in Figure 6.4 were to be simulated before the photon particle, then it would no fulfill the condition C of the dynamic routing rule. Thus, the pion will probably be simulated

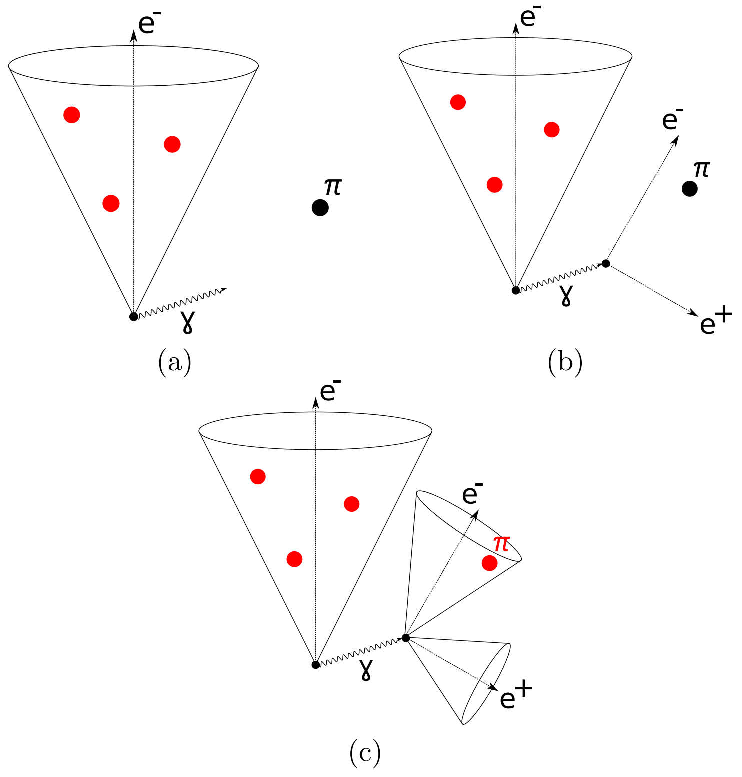


Figure 6.4: A dynamic routing rule that defines cone-shaped regions of interest around all electrons and positrons in the event. The particles that are selected by this rule are marked red, the others black. Figure (a) shows the initial state of the event simulation with an electron, a photon, a pion and a few other particles that are inside the cone ROI. Figure (b) shows the state after the photon has been simulated and, in this case, it undergoes a conversion into an electron-positron pair. Subsequently, in Figure (c), the resulting cones around the electron and positron are shown, which now also contain the pion particle.

by a different simulator compared to the case where the photon (which undergoes a conversion) were simulated first.

This may lead to, what is called, *inconsistent* simulator decisions for the particles in the event: Looking at the event after it has been simulated, one may find particles that should have been selected by a given dynamic routing rule, where in fact they were simulated by a different simulator. This inconsistency arises if the ROI that includes the particle in question, is created after this particle has already been simulated. Such inconsistent simulator decisions contradict the users' intention when defining dynamic routing rules. Thus, this effect renders the simulation behaviour somewhat non-intuitive and, therefore, must be avoided.

There are two solutions to solve or minimize this inconsistency:

Prioritization of routing rules : It is possible to minimize this inconsistency by applying a prioritization of the routing rules (Section 6.1) and trigger the simulation of a particle as soon as its simulator is determined. To do so, *all* particles that can *potentially* cause a change to the ROIs of the dynamic routing rules, must be simulated early on. More specifically, these particles must be simulated before any dynamic routing rule is evaluated for any one particle in the event. In the example above (Figure 6.4), all photons would need to be selected for simulation by a dedicated photon routing rule, that has a higher priority than the dynamic routing rule. After this selection is made, each photon would need to be simulated right away. Thus, when the dynamic routing rule is evaluated for the first time, the ROIs will be based on the initial event input and the photon simulation results (potential electron-positron pair conversions). However, the electron or positron constantly emits bremsstrahlung [87] during its detector simulation due to the presence of a magnetic field. Some of these bremsstrahlung photons, again, might undergo conversions into electron-positron pairs. Thus, new ROIs around these electrons and positrons will be registered by the dynamic routing rule. To avoid simulating particles that end up inside these ROIs before they are even created, also electrons and positrons would need to be selected and simulated prior to the evaluation of the dynamic routing rule.

"Un-doing" simulated particles : It is possible to get fully consistent simulator decisions if the simulation framework is capable of discarding all simulation results of individual, already simulated, particles. This would need to be applied to all particles for which the simulator decision changes, after they had already been processed through a detector simulation. In the context of dynamic routing rules, this will happen if an ROI is created that includes a particle which had already been simulated at the time of the creation of the ROI. All simulation results of this particle's detector simulation must be purged: among other objects, this applies to all sensitive detector hits, secondary particles and even dynamic routing rules which were updated due to the secondary particles. Once these simulation results haven been purged, the particle can be re-simulated within a different simulator. The purging of simulation results of a particle with many daughter particles, illustrates that this method may introduce a significant computational overhead. This overhead may outweigh the benefits of having fully consistent routing decisions. In addition, a very detailed bookkeeping would be necessary to associate all simulation results that are produced by any one particle. Thus, this method is considered as being inefficient in the context of the ISF.

Figure 6.5 illustrates the difference between two methods for dynamic routing rules to test whether a particle is positioned inside a cone-shaped ROI (with cone size ΔR) or not. The first method uses the particle position to determine whether it is located within the ROI. In the figure, the creation vertex of positron e_2^+ is positioned within the ROI (as $|\eta_{pos} - \eta_0| \leq \Delta R$ is fulfilled) thus, the particle is selected by the dynamic routing rule (red). The second method uses the particle momentum to test the condition of the cone-shaped ROI. The momentum direction η_2 of the positron e_2^+ in the figure does not fulfill the ROI condition (as $|\eta_2 - \eta_0| \nleq \Delta R$) thus, the particle will not be selected by the dynamic routing rule (blue). Either method may be implemented in dynamic routing rules.

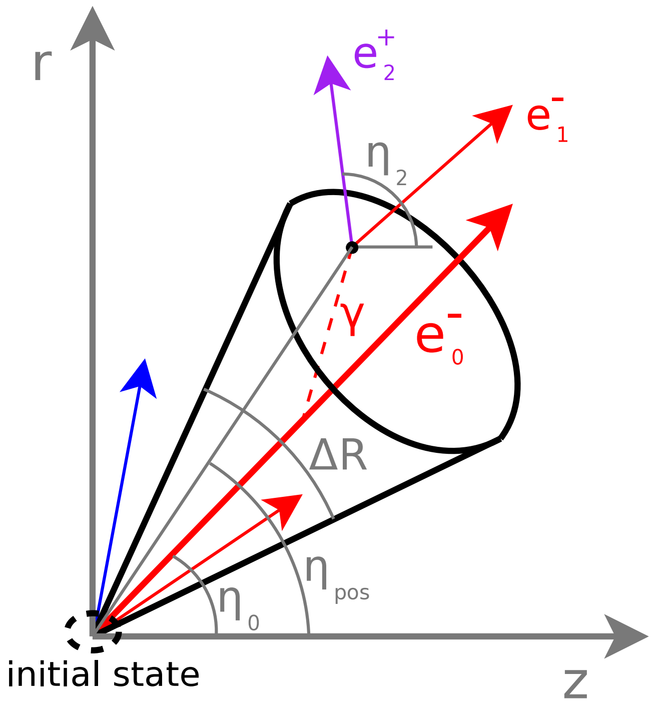


Figure 6.5: An illustration about the difference between position-based and momentum-based testing whether a particle is within a cone-shaped ROI or not. In the former method, the position e_2^+ is considered inside the cone, as the condition $|\eta_{pos} - \eta_0| \leq \Delta R$ is fulfilled. In the latter case, the same position is outside the cone-shaped ROI, as the condition is not fulfilled with $|\eta_2 - \eta_0| \nleq \Delta R$.

Due to the complex implications of dynamic routing rules, the ISF routing algorithm does not support dynamic routing rules. However, a possible extension to the ISF routing algorithm, that will also support dynamic routing rules, is discussed in Section 6.2.3.

6.2 The ISF Routing Chain

This section discusses the details of the ISF routing algorithm, the *ISF routing chain*, and its implications. Subsections 6.2.1 and 6.2.2 cover the specifics of static and semi-dynamic routing rules within the ISF routing chain, respectively. A possible extension to the current routing algorithm, in order to support dynamic routing rules, is discussed in Subsection 6.2.3. A detailed documentation about the routing chain's implementation can be found in a later section, Section A.3.

The ISF routing algorithm was built, based upon these fundamental design decisions:

Order independent : The routing decisions taken by the ISF routing algorithm must be independent of the order in which the particles are read-in and processed by the framework. This requirement becomes particularly critical in the light of concurrent processing of particles within the simulators and within the simulation framework in the near future. In addition, the simulator decisions must not depend on the order in which any secondary particles may be returned from the simulators.

Supported classes of routing rules : Due to the complex implications of dynamic routing rules, the ISF routing algorithm was designed to support only two classes of routing rules: static and semi-dynamic routing rules.

Routing rule prioritization : The ISF routing algorithm requires that routing rules are prioritized with respect to each other, such that two routing rules can not have the same priority within one detector simulation job.

Separate set of routing rules per sub-detector : Due to the completely different simulation approaches for the individual ATLAS sub-detectors, one set of routing rules is to be defined per ATLAS sub-detector. The routing rules though, can be identical for different sub-detectors.

One evaluation per sub-detector : The simulator for any given particle will be determined only once per ATLAS sub-detector. This is to avoid the significant computational overhead of re-evaluating each particle's routing decision in each individual simulation step. Consequently, the routing rules are only evaluated once per sub-detector for a given particle. This comes at the price, that a particle might leave an ROI of a static or semi-dynamic routing rule and its routing decision will not be updated.

The implementation resulting from the list of requirements above, is designed as a chain of routing rules which are evaluated in order. Each routing rule i in the chain has a condition C_i and a simulator S_i (see Figure 6.6). In the context of the ISF, a routing rule is also called a `SimulationSelector`, as this is the class name of the C++ interface and implementation (Section A.3.1). A chain of routing rules has to be defined for each ATLAS sub-detector individually. Though, in some cases, the individual chains might be equivalent or contain the same routing rules.

The process of identifying a simulator S for a given particle P is the following:

1. The ISF determines the applicable routing chain, based on the position of the particle. Each ATLAS region or sub-detector has a separate routing chain.

2. The ISF evaluates the routing rules according to their priority. It starts with the routing rule that has the highest priority and moves towards rules with lower priority. In Figure 6.6 the rules are evaluated from the left to the right.

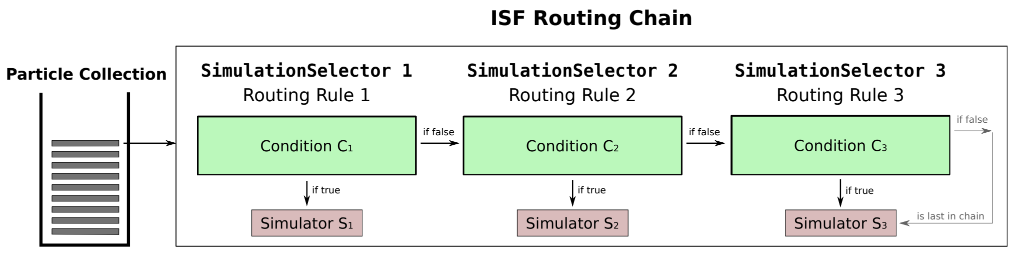


Figure 6.6: The ISF routing chain. Each routing rule i is associated with a condition C_i and simulator S_i. In the context of the ISF, a routing rule is also called a `SimulationSelector`. The individual routing rules are arranged in a prioritized order. To determine the simulator for a given particle, the conditions C_i are evaluated in the order of the prioritization. The simulator for a given particle is set to S_i, where i is such that C_i is the first condition in the chain that is fulfilled by the given particle.

3. In this process of stepping through the routing rules, the first condition C_i that is fulfilled by the particle P determines that the simulator for this particle will be set to S_i. After this, no further routing rule evaluations will be carried out for the given particle.

4. The routing rule with the lowest priority in the chain will accept any particle. If a particle was not selected by any prior routing rule, it will be assigned to the simulator of the last routing rule. This behaviour guarantees that every particle will be assigned to a well defined simulator.

6.2.1 Static Routing Rules in the Routing Chain

Static routing rules (Section 6.1.1) are the least complex and most basic class of routing rules that are supported by the ISF routing algorithm. By definition, the conditions C_i of static routing rules will not change throughout the execution of the ISF detector simulation. However, static routing rules are still flexible enough such that all traditional ATLAS simulation setups (full Geant4, ATLFASTII and ATLFASTIIF) can be formed by a set of static routing rules.

6.2.2 Semi-Dynamic Routing Rules in the Routing Chain

Semi-dynamic routing rules (Section 6.1.2 are a class of routing rules which can incorporate properties of the initial event (i.e. the event generator output) into the conditions C_i.

At the same time as the ISF is preparing the simulation input for a given event, each semi-dynamic routing rule is updated with the initial list of particles in this event (Figure 6.7a). The ISF does not request a routing decision for these particles at this point, but the individual semi-dynamic routing rules will adjust their conditions C_i according to the properties of the initial list of particles. After the semi-dynamic routing rules were updated in this way, the conditions C_i remain unchanged (they are "locked") throughout the simulation of the current event. The actual particle routing algorithm, as discussed in Section 6.2, will use these updated routing rules to determine a simulator for each particle (Figure 6.7b). Already the routing of the initial list of particles will be done using the updated routing rules.

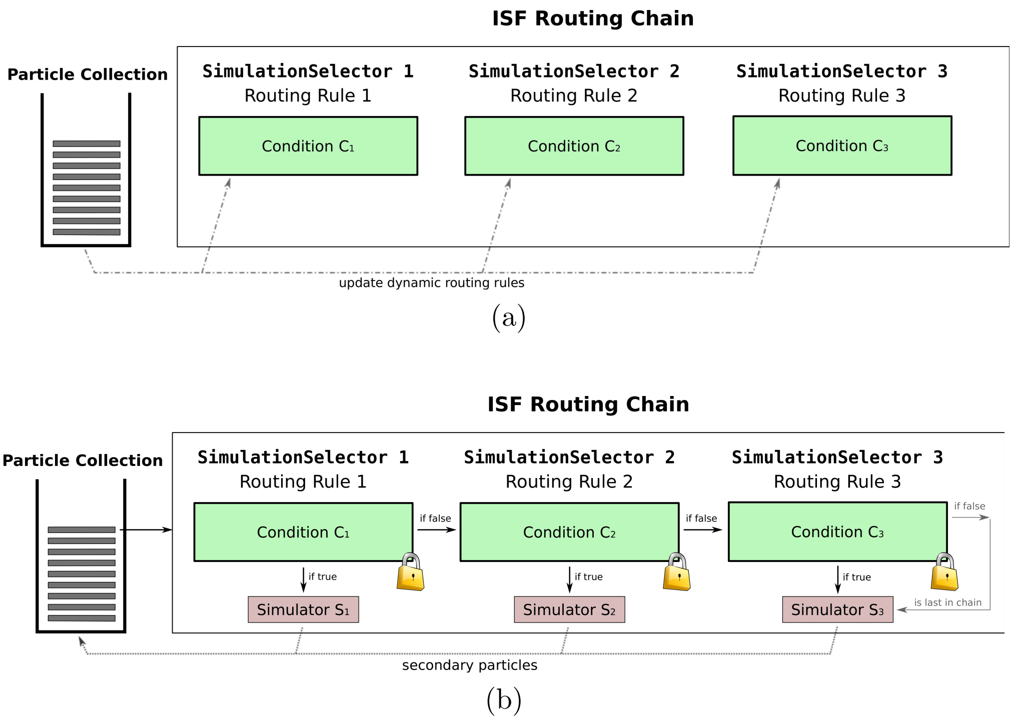


Figure 6.7: The ISF routing chain supports semi-dynamic routing rules: (a) At the beginning of each event, the semi-dynamic routing rules are updated according to the particles in the initial particle list. (b) After this, the conditions C_i are be locked and will not change throughout the rest of the current event simulation. These updated conditions will be used to find particle routing decisions for all particles in the event.

6.2.3 Dynamic Routing Rules in the Routing Chain

Due to the complex implications, dynamic routing rules (Section 6.1.3) are not supported by the current implementation of the ISF routing algorithm. This section discusses a possible extension of the ISF routing chain, which emerged in the design phase of the ISF.

The main challenge to solve, is the possible inconsistency between previously made simulator decisions and continuously updated dynamic routing rules. This inconsistency can arise if the condition C_i of a dynamic routing rule is updated, such that a simulator decision might change for a particle that has already been simulated. As discussed previously, this inconsistency can be minimized, if dynamic routing rules are provided as part of a set of prioritized routing rules. The aim of this set of routing rules is, to simulate as many of the particles that could potentially change the conditions C_i of the dynamic routing rules, as early as possible in the event. Based on the principles of this method, a routing chain design with an *incremental* locking of the routing rules emerged (Figure 6.8). The incremental locking is currently not implemented into the ISF routing chain. Therefore, this section only discusses the theoretical concept and implications of such an algorithm.

The routing chain with incremental locking is an extension of the algorithm described in the previous Section 6.2:

1. The conditions C_i of all dynamic routing rules are updated according to the initial list of particles in the event (Figure 6.8a).

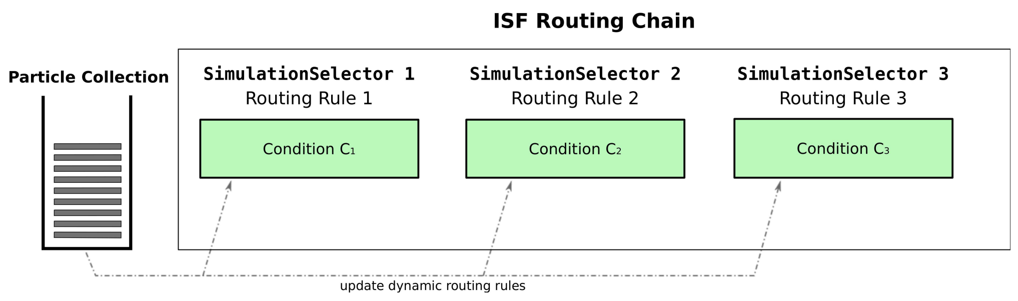
2. After all routing rules were updated in this way, the condition C_1 of the first routing rule is locked (Figure 6.8b). This means that the condition of this routing rule will stay constant until the end of the current event simulation. Subsequently, all particles that are selected by the first routing rule are simulated through S_1. The first routing rule is also evaluated for all secondary particles, regardless of their generation number. Particles that are selected by the first routing rule will be sent to the assigned simulator right away. At the same time, the conditions C_i of all unlocked routing rules are updated with the secondary particles produced by S_1. The particles which are not selected by the first routing rule are put aside ("put on hold") for later processing.

3. Once the first routing rule does not select any more particles, i.e. there are no more particles to be simulate by S_1, the second routing rule is locked (Figure 6.8c). At this point, all particles are in the "on hold" collection after the first routing rule. The second routing rule is now evaluated for these particles that were put aside in the previous step. As before, all locked routing rules in the chain will be evaluated (in order) for the secondary particles, regardless of their generation number. All particles that are selected by either the first or the second routing rule are simulated right away in simulator S_1 or S_2, respectively. Particles that are selected by neither of the two, are put aside for later processing.

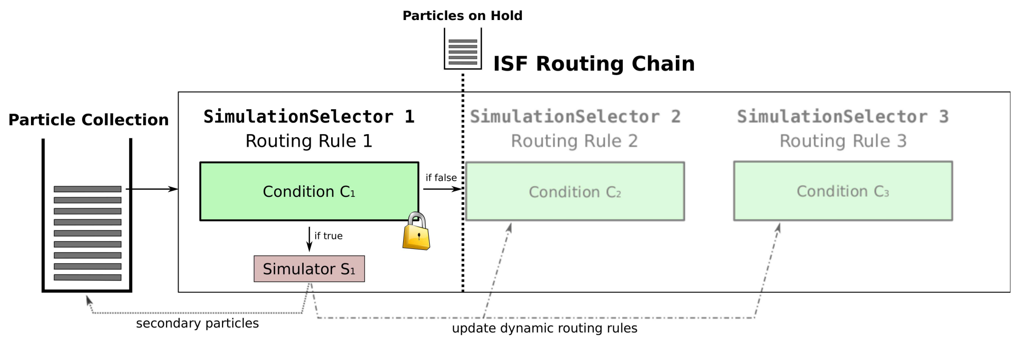
4. This scheme continues until the last routing rule in the chain is locked. At this point, all conditions C_i will be locked and all routing rules are evaluated (in order) for the particles in the simulation framework.

5. The detector simulation ends, if all particles (including all secondaries) have been completely processed by the simulators S_i and no more particles need to be sent to any simulator.

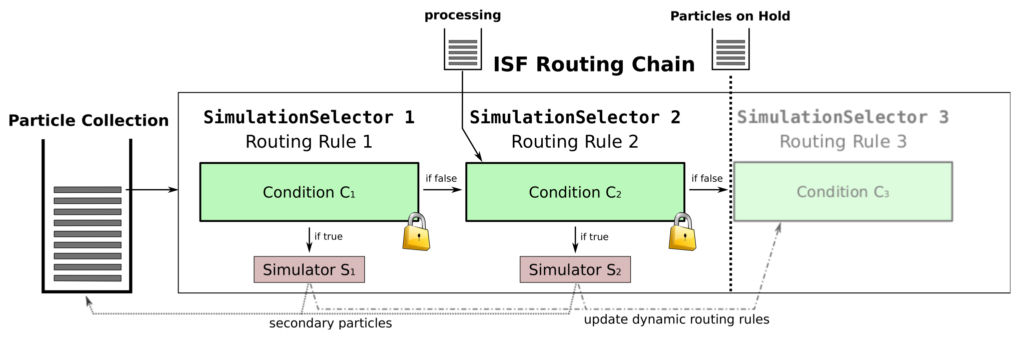
The algorithm discussed above, guarantees that the dynamic routing rules are updated (i.e. stay unlocked) as long as possible throughout the simulation of an event. The conditions C_i of the dynamic routing rules will take into account all particles (including secondaries) that were present until the routing rule is evaluated for the first time. The



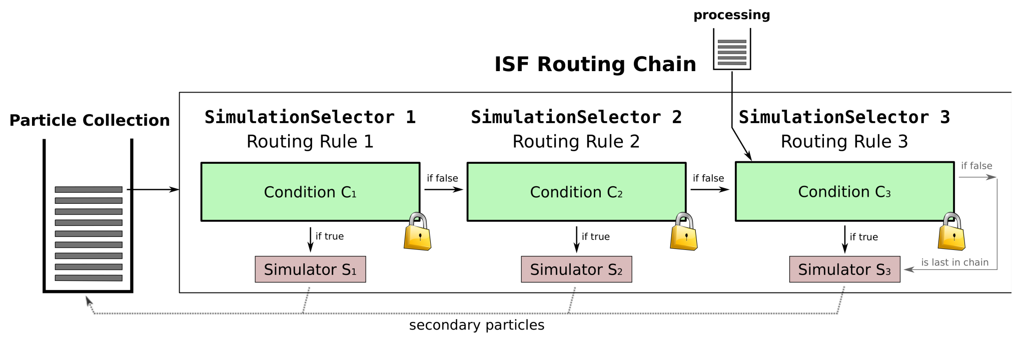
(a)



(b)



(c)



(d)

Figure 6.8: A concept for an ISF routing chain that supports dynamic routing rules. (a) All dynamic routing rules are updated with the particles in the initial particle list. (b) Subsequently, the condition C_1 of the first routing rule is locked. All particles (including all secondaries) that are selected by the first routing rule are simulated through simulator S_1. All unlocked dynamic routing rules are updated with the secondary particles. (c) The second routing rule is locked and all particles (including all secondaries), that are selected by either the first or the second routing rule, are simulated through S_1 or S_2, respectively. (d) This scheme continues until the last routing role is locked. At this point, all remaining particles (and secondaries) will be simulated and all of the conditions C_i are locked.

rules need to be locked, however, as soon as they are used to assign simulators to particles. This locking guarantees that the decisions taken by a dynamic routing rule will not change until the end of the current event simulation. If a dynamic routing rule were used to assign a simulator to a particle, and this routing rule can change later on as a result of the ongoing event simulation, the decisions that have already been taken by the routing rule might become inconsistent. That is to say, the updated condition C_i of such a routing rule, may give different results for the particles that were already selected by this rule prior to the update. If one were to look at the condition C_i of such a routing rule at the end of the event simulation, it would not be possible to determine which particles were selected by the routing rule during the simulation of the event. Even more importantly, the locking of dynamic routing rules guarantees that the routing algorithm as a whole is order independent. Assuming a dynamic routing rule with a condition C_i that varies throughout the simulation of an event, is used to determine a simulator for a particle. The decisions taken by this dynamic routing rule would be different, if the particles which update the routing rule were provided in a different order. Thus the evaluation of unlocked dynamic routing rules in the routing algorithm, causes the simulator decisions to be dependent on the order in which the particles are processed. This would violate one of the fundamental requirements to any ISF routing algorithm (Section 6.2).

Chapter 7

ISF Components

The requirements of the Integrated Simulation Framework (as covered in Chapter 5) arise from years of experience using various ATLAS detector simulation approaches. To guarantee a swift integration of the ISF into the existing Monte Carlo simulation infrastructure, the framework is designed as a drop-in replacement for existing ATLAS detector simulation setups. New features, such as advanced particle routing algorithms (Chapter 6), are transparent to subsequent steps in the ATLAS Monte Carlo production chain (Section 4.2). The input and output data formats are compatible with the ATLAS detector simulation technologies preceding the ISF. Some components within the ISF are therefore adaptations of already existing algorithms (examples are Monte Carlo truth recording and particle barcode generation), which guarantees a high level of compatibility with the existing MC production chain. In many cases, these algorithms were previously simulator-specific implementations, whereas they are now centrally handled by the ISF and used for any ATLAS detector simulation setup.

The main components of the ISF are the individual detector simulators (Section 7.1), which process the particles through the ATLAS detector volume and ultimately generate the corresponding particle energy depositions, detector signals or measurements. As each simulator implements a different algorithm to do so, the various detector simulators are implemented in independent components. The simulators retrieve the particles for simulation from the framework, which uses the newly developed particle routing algorithm to identify an appropriate simulator for each particle in the event (Section 7.2). At the beginning of each simulated event, the framework uses the routing algorithm to determine the according simulator for each particle in the input. Depending on the specific configuration, the routing algorithm may be used to determine the appropriate simulators for secondary particles which are created during the detector simulation as well. As the particle routing algorithm uses different sets of routing rules for different regions in the detector, a precise definition of ATLAS regions and sub-detectors inside the detector simulation is a fundamental part of the ISF (Section 7.3). A Monte Carlo truth representation of the particles involved and the interactions occurring during the detector simulation is essential for subsequent performance studies or physics analyses (Section 7.4). Hence, the ISF implements a central MC truth service which records a representation of the particles simulated through the detector in the HepMC data format. Contrary to previous implementations, the algorithms and rules concerning the recording of Monte Carlo truth inside the ISF are independent from the particular simulators involved in the detector simulation. The ISF MC truth components are centrally configured and the same level of MC truth information is recorded for all simulators, provided the respective simulators are able to generate the required particle-level truth information in the first place. The ISF generates and assigns barcodes to the individual particles involved in the detector simulation (Section 7.5), as this allows to associate objects which will later be reconstructed with the underlying Monte Carlo truth object.

The modular design of the ISF allows for multiple implementations of various core ISF components. The individual components of the core framework can be replaced by alternative implementations, which must provide the necessary functionality and adhere to the specific implementation requirements (covered in the implementation appendix A). The ISF components described in detail in this chapter are the default components, but alternative implementations may exist and may be used for particular ISF setups or studies.

7.1 Simulators and SimulationServices

The core components of any ATLAS detector simulation are the algorithms which compute particle energy depositions, detector responses or measurements of particles traversing the ATLAS detector. In order to achieve this, different types of simulation algorithms exist. Some algorithms compute particle interactions, decays and energy loss of particles traversing the ATLAS volume. Other algorithms compute the detector response or particle measurements corresponding to the simulation input. In the context of the ISF, these algorithms are called simulators, sometimes also referred to as simulation engines or simulation services. The various full and fast simulation approaches available to the ATLAS collaboration were described in Section 4.3. This section covers the implications of using these in the context of the ISF.

Simulators are exchangeable modules in the simulation framework and the ISF does not distinguish between different types of simulators[1]. The ISF uses particle routing algorithms with routing rules to determine the appropriate simulator for each particle in the event (Section 7.2). Once the simulator is determined, the particle is sent to it for the generation of the simulation result in the corresponding output data format. Depending on the simulator type, it is within the simulators responsibility to compute particle trajectories, particle decays, particle-material interactions, energy deposits in sensitive detector (SD) elements, detector responses or particle measurements.

[1]For technical reasons, the ISF can address different simulators through a unique identifier, which is assigned to each simulator at the beginning of the Athena job.

7.1.1 Requirements for Simulators

This section provides a brief summary of the most important requirements a simulator must fulfill for successful integration into the ISF. A more detailed list, focussing on the specific requirements of the implementation, is available in the appendix Section A.4.1.

There are no restrictions on the type of simulator to be integrated into the ISF. Full detector simulation, pre-simulated shower libraries and parameterized simulation approaches are already successfully integrated into the ISF.

Algorithmic Properties

A simulation service used by the ISF must be a deterministic algorithm and therefore create reproducible output. Consequently, if the ISF detector simulation is run multiple times with the exact same input and the same configuration, the output must be bitwise identical. This is less a requirement by the simulation framework implementation, but a generic requirement to guarantee efficient development and validation of ATLAS detector simulations.

In addition, a simulator has to finish processing of particles in finite time. If any one simulator in the ISF enters an endless loop, the entire ISF simulation process will loop endlessly. The ISF does not have measures in place, to detect such misbehaviour of a simulator. Eventually, the Athena framework will trigger a timeout condition and the process will be terminated. Hence, this requirement must be validated during the design and development phase of each respective simulator.

Particle Decays or Interactions with Detector Material

Particle interactions and decays computed inside a simulator are converted into a simulator-independent data format and registered with the ISF TruthService (Section 7.4). From this, the TruthService creates a consistent Monte Carlo truth representation of the simulated event.

However, simulators in the ISF are not generally required to compute particle decays or interactions with detector material. For example parametric simulators (such as FastCaloSim), compute a detector response directly from the simulator input, without simulating each particle's interactions with the detector material. Thus, this requirement only applies to simulators which do compute particle interactions and decays.

Secondary Particles

Secondary particles created by simulators may be returned to the ISF by sending them to the ISF ParticleBroker (Section 7.2). A common data format (ISFParticle) is used by all simulators for returning particles to the framework. Any particle (primary or secondary) crossing a boundary between ATLAS regions or sub-detectors (Section 7.3) within a simulator must be returned to the ISF ParticleBroker. This allows the framework to re-evaluate the routing decision for this particle. One exception to this is the full simulation

ISF setup, for which Geant4 is the only simulator present in ISF and therefore all particles remain inside Geant4 throughout the entire ATLAS detector volume (Section 9.1).

Similar to the previous requirement, simulators in the ISF are not generally required to generate secondary particles. Therefore this requirement only applies to all simulators which do generate secondary particles.

Sensitive Detector Hits

With the ISF serving as a common framework for different simulator implementations, a common data format for sensitive detector hits is used by all simulators which generate SD hits. Simulators are required to add simulated SD hits directly to the various StoreGate collections, each representing a different type of sensitive detector. Simulators are not generally required to generate SD hits, however, if they do they must adhere to the common SD hit format.

Random Number Streams

Each Monte Carlo simulator must utilize a separate stream of random numbers. This is to allow testing of simulators independently, in particular ISF setups where a mix of different simulators is used.

7.2 Particle Routing and the ParticleBroker

The ISF particle routing is the most innovative and essential part of the simulation framework. A ParticleBroker identifies an appropriate simulator for each particle in the event. The ParticleBroker implements the ISF routing chain algorithm (Section 6.2) and allows for static and semi-dynamic routing rules. The ISF particle routing identifies the simulator for primary particles present in the EVNT input, as well as for secondary particles which are returned by the simulators to the ISF at any point during the event simulation.

As different simulations are capable of simulating different parts of the ATLAS detector, the ParticleBroker holds one routing chain per ATLAS region or sub-detector (Section 7.3). This enables the user to configure the particle routing in accordance with the simulators that are available for the individual regions. Due to this, cross-checks to determine whether a particle is within a valid region for a given simulator are not necessary. For example, FastCaloSim is only capable of simulating particles in the ATLAS calorimeter, thus this simulator may only be configured in the routing chain for the calorimeter region.

Routing Rules and SimulationSelectors

The routing rules in the context of the ISF ParticleBroker are called SimulationSelectors. Each SimulationSelector i is associated with a simulator S_i and a condition C_i. Static SimulationSelectors base their decision on the information present about a given particle.

Semi-dynamic SimulationSelectors also take into account the particles present in the EVNT simulation input, in order to decide whether or not they select a given particle. Thus, the condition C_i of semi-dynamic SimulationSelectors is updated for each event.

The routing chain for each ATLAS detector region is formed by a list of Simulation-Selectors which are provided in a prioritized order. To determine the simulator for a given ISFParticle, the ParticleBroker iterates through the list of SimulationSelectors of the ATLAS detector region the particle is in. The simulator for the particle is defined as the simulator S_i which is attached to the first SimulationSelector in the chain that selects the particle. Thus, not all routing rules may be evaluated for a given particle in order to determine its simulator.

Re-Evaluating Routing Decisions

Once the simulator for a given particle is determined, the particle is temporarily stored in the ParticleBroker until the ISF program flow determines that the particle will be sent to the assigned simulator for detector simulation (Chapter 8). A re-evaluation of the simulator decision is not required before reaching the boundary of the current ATLAS region or sub-detector. Thus, the simulator may process the particle until this boundary, but must return it at the boundary. Since the ParticleBroker stores separate routing chains for each ATLAS region, it will consult a different routing chain for this particle when it crosses a boundary and thus the simulator decision is re-evaluated. At any time, simulators may return secondary particles that are created during the event simulation to the ParticleBroker. Upon receiving a particle from a simulator, the ParticleBroker informs the EntryLayerTool (Section 7.4.1) about the particle to ensure that the corresponding MC truth records in the EntryLayer collections are created, if applicable.

7.3 ATLAS Detector Regions

Essential for ISF particle routing is a precise definition of the individual ATLAS detector regions or sub-detectors. Based on the layout of the physical ATLAS sub-detectors, the ATLAS detector volume in ISF is divided into the following regions:

- ATLAS inner detector
- ATLAS forward or beam pipe
- ATLAS calorimeter
- ATLAS muon spectrometer
- ATLAS cavern

Figure 7.1 illustrates the size of the individual regions as defined in the ISF. There are no gaps between the regions, as the particle routing would not be able to determine a unique routing solution inside such gaps. The ATLAS cavern region extents from the muon spectrometer and forward region onwards and differs in size for the simulation of collision events and cosmic ray events, respectively.

The ATLAS regions are mirror symmetric in the $x-y$ plane and rotational symmetric around the z axis. Thus, each region is fully defined by a list of points in $(r, |z|)$ space which describe its boundary surface.

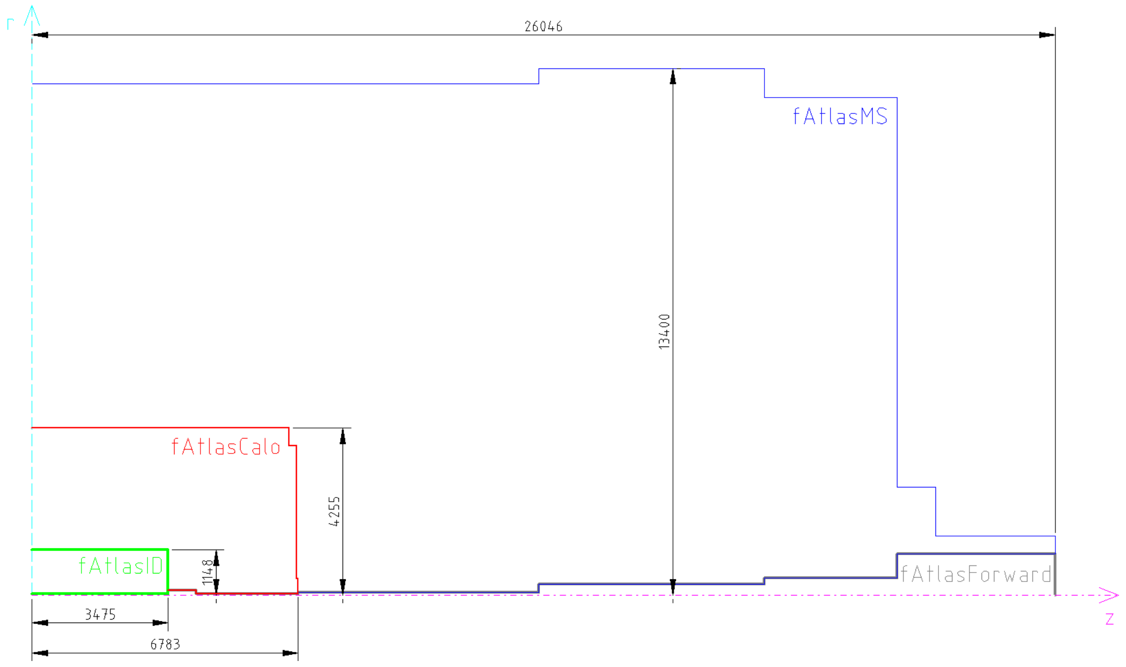


Figure 7.1: The ATLAS detector regions in ISF as defined in release `17.7.5.4`. The naming convention corresponds to the respective entries in the `AtlasDetDescr::AtlasRegion`. The drawing shows the maximum extension (in mm) for the individual regions. Not shown is the ATLAS cavern region.

Due to particle routing considerations, the ATLAS region definitions in the ISF differ from the ATLAS region definitions used inside ATLAS Geant4 simulation. In the ATLAS Geant4 definition, gaps are required between many of the envelope boundaries. This is implemented for technical reasons, resulting in a higher ATLAS Geant4 simulation speed. In addition, in the Geant4 definition the inner detector region surrounds the central part of the beam pipe, which is part of the beam pipe region. In the ISF definition, this central part of the beam pipe is contained in the inner detector region and the beam pipe region extends outwards (to higher $|z|$) from the most forward boundary of the inner detector region. This modification is to minimize the computational costs and configuration complexity of the particle routing algorithms implemented in the ISF. As mentioned before, the ISF routing algorithm requires a routing chain definition for each ATLAS detector region, respectively. For the simulation of collision events, the ATLAS inner detector volume is the first crucial volume that particles traverse. Though, in the ATLAS detector, particles first traverse the central beam pipe volume, a separate routing configuration in this part of the beam pipe is not required. With the central beam pipe volume being part of the inner detector region, it is guaranteed that the same routing rules will be applied throughout the region, including this central part of the beam pipe volume. This minimizes the chance of misconfiguration. In addition it increases the computing performance of the simulation framework, as simulators are required to hand back all particles at region boundaries. With the modified inner detector region, the collision point lies within the inner detector region. Thus, particles emerging from the collision point will traverse the central part of the beam pipe and the rest of the inner detector region before reaching the first region boundary. At this point the particles are returned to the ISF ParticleBroker by their simulator for the first time.

7.3.1 Identifying ATLAS Detector Regions with the GeoIDService

The knowledge of the ATLAS region that simulated particles are located in or are going to enter next in the process of the detector simulation is vital to ISF detector simulation. The GeoIDService is capable of solving the following three tasks based on position and momentum information, all of which are covered in detail in this section:

1. Identifying the ATLAS region within which the given position or particle is located in.
2. Predicting the ATLAS region a particle will be simulated in for the next simulation step.
3. Determining whether the given position or particle is located on a boundary surface between two ATLAS regions.

Identifying ATLAS region of a given position

The GeoIDService resolves the ATLAS detector region or sub-detector that a particle resides in based on its position. Due to its critical role and frequent use in the ISF ParticleBroker and in the Geant4 simulation, the GeoIDService is optimized for high execution speed.

The GeoIDService maps the entire ATLAS region definition (Figure 7.1) onto a map of rectangular-shaped areas (or bins) in $(r, |z|)$ space. Each bin is associated to one ATLAS region and represents a (hollow) cylinder volume in the ATLAS detector. Figure 7.2 illustrates the $(r, |z|)$ map constructed by the GeoIDService.

The global coordinate system used in ATLAS detector simulation is a Cartesian coordinate system centred at the nominal collision point. The GeoIDService converts this into the corresponding $(r, |z|)$ coordinate before resolving the ATLAS region for a given position (x, y, z). Subsequently, the GeoIDService identifies the bin which contains the coordinate and returns an identifier (`AtlasDetDescr::AtlasRegion` enum) of the ATLAS region stored in this bin.

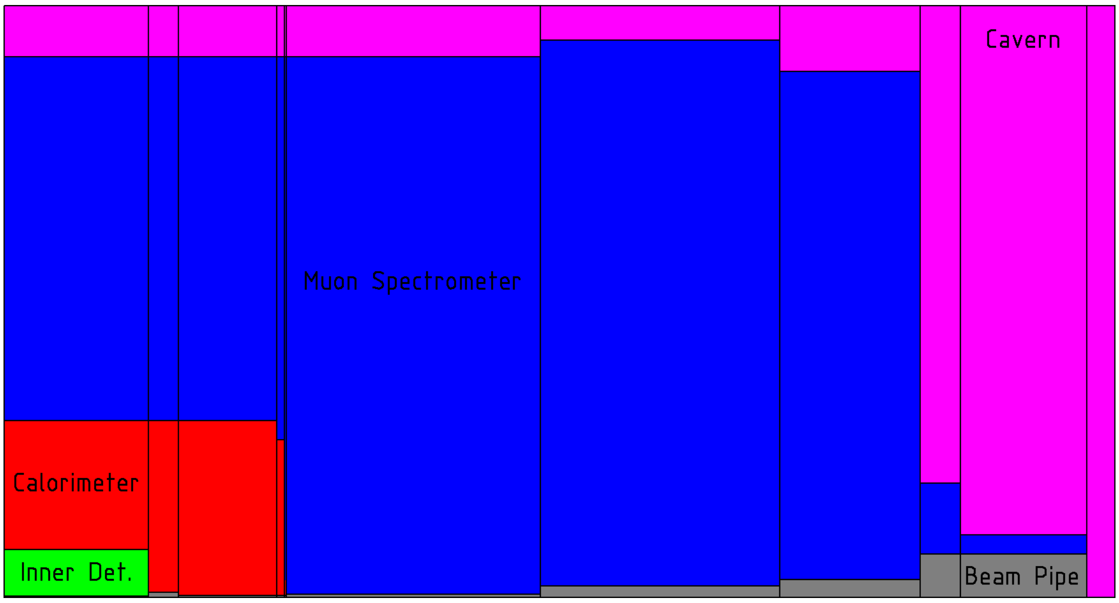


Figure 7.2: The ATLAS region map in $(r, |z|)$ space constructed by the GeoIDService ($|z|$ horizontal, r vertical). Each rectangular-shaped area (bin) is associated to one ATLAS region. To resolve the ATLAS region of a given position, the GeoIDService identifies the bin this position lies within and returns the ATLAS region identifier of this bin.

Geant4 uses the GeoIDService to determine, whether the region a particle is in has changed with the previous simulation step of this particle (Section 9.1). If this is the case, Geant4 returns the particle to the ISF ParticleBroker. This is crucial, as simulators in ISF are required to return particles to the ParticleBroker when crossing a region boundary.

Predicting next ATLAS region for a particle

In addition to resolving the ATLAS region corresponding to a (particle) position, the GeoIDService can predict the region within a particle will be simulated in the next simulation step. This information is crucial for the ISF particle routing, as simulators return particles on region boundaries and the ParticleBroker needs to determine which region the particle will enter after leaving the boundary surface. The ParticleBroker uses this information to consult the appropriate routing chain to find a routing decision.

The GeoIDService applies a straight-line extrapolation to determine the next ATLAS region for a given particle. For a particle with position $X = (x, y, z)$, the GeoIDService resolves the ATLAS region of an extrapolated particle position X', which is shifted away from X along the particle's momentum p by a distance of μ (default 10^{-5} mm):

$$X' = X + \mu \frac{p}{\|p\|} \tag{7.1}$$

This straight-line extrapolation is fast, but approximative. The accuracy can be adjusted with the parameter μ, which is therefore also called tolerance. Even with a magnetic field present, the straight line extrapolation over short distances provides sufficient accuracy for the purpose of the particle routing. Errors in predicting the next ATLAS region may occur for charged particles with very low momenta, as their real trajectory in the magnetic field might deviate from the estimated straight line extrapolation. However, the occurrence of these particles at a boundary between two regions is rare and particles with very low momenta are often close to minimum energy thresholds of simulators. Thus this effect can be neglected.

Determining whether position is on boundary between two regions

In addition to the functionalities described above, the GeoIDService is capable of determining whether or not a given position is on a boundary surface between two ATLAS regions. This functionality is essential for creating the EntryLayer track record collections, which store MC truth information of simulated particles traversing ATLAS region or sub-detector boundaries (Section 7.4.1).

Each particle returned from a simulator to the simulation framework is tested through the GeoIDService. If it resides on one of the relevant boundary surfaces, the particle is added to the corresponding EntryLayer track record collection.

The GeoIDService applies a fast and approximative method to test if given a position is between to ATLAS regions: First, two positions X_{fwd} and X_{aft} near the original position $X = (x, y, z)$ are computed. Next, the GeoIDService uses the method described above, to identify the ATLAS region within which each of the positions is located in:

$$d = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{7.2}$$

$$X_{\text{fwd}} = X + \mu \frac{d}{\|d\|} \tag{7.3}$$

$$X_{\text{aft}} = X - \mu \frac{d}{\|d\|} \tag{7.4}$$

X_{fwd} and X_{aft} are each within a distance of μ (tolerance) of the original position X. The new positions are linear extrapolations of the original position X in the direction d and $-d$. The direction d must fulfill the condition that a position X which is close to a boundary will cross this boundary if linearly extrapolated along d or $-d$. Thus, d must not be parallel to any ATLAS region boundary. The numerical values chosen for d are arbitrary, but fulfill this condition. The GeoIDService uses the method described above to identify the ATLAS region corresponding to each of the two new positions. Two different outcomes are possible:

X_{fwd} and X_{aft} are in the same ATLAS region : In this case the position X is not located on the boundary of an ATLAS region. Instead the position X is well within the region which is determined for both, X_{fwd} and X_{aft}.

X_{fwd} and X_{aft} are in different ATLAS regions : In this case the position X is located on the boundary between the two different regions.

7.4 Monte Carlo Truth Recording with the TruthService

The Monte Carlo truth representation is an essential part of the simulation output for any ATLAS detector simulation. The MC truth representation allows performance studies or physics analyses to associate reconstructed objects with the underlying simulated object(s).

The ISF TruthService's main responsibility is to generate a consistent representation of the most important particle interactions and decays that occur within different simulators inside ISF. Due to the great number of interactions and decays that are computed by the simulators, only a subset can be recorded persistently in order to not exceed disk space requirements. In general, interactions are only recorded if the particles involved are above predefined transverse-momentum or energy thresholds, as low energy interactions are not relevant for subsequent analyses. The MC truth representation generated by the ISF TruthService is in the HepMC format [63, 64] and is stored in the `TruthEvent` StoreGate collection.

Simulators in the ISF update the central ISF TruthService with so-called TruthIncidents about all particle interactions and decays that occur within them. TruthIncidents are a simulator-independent data format, which allows the TruthService to access details about the interaction or decay, independent of the data format used in the underlying simulator. TruthIncidents can be seen as wrappers for the ISF TruthService to access information which is otherwise simulator-specific.

TruthStrategies are an essential part of the ISF TruthService for determining the subset of all TruthIncidents which is to be recorded persistently. Due to different levels of accuracy required in the MC truth representation for different parts of the ATLAS detector, separate sets of TruthStrategies are defined per ATLAS detector region. For example, due to the high rate of particle interactions in the ATLAS calorimeter, a lower level of detail in the MC truth representation is recorded in this part of the detector. If this were not implemented, the recorded MC truth representation would exceed disk space requirements.

Upon receiving a TruthIncident from a simulator, the TruthService calls all TruthStrategies which are registered in the region the TruthIncident is located in. If any one TruthStrategy in this region tags the given TruthIncident for persistent storage, the TruthService will convert the information in the TruthIncident to the HepMC format and attach it to the MC truth tree structure stored in the `TruthEvent` StoreGate collection. It follows that, adding a new TruthIncident to any set of TruthIncidents may only add entries to the persistent MC truth record for a given simulated event. Thus, if a particular interaction process is of interest for subsequent analyses, a new dedicated TruthStrategy can be implemented and added to the existing set of TruthStrategies in the respective detector region of the respective ISF setup.

7.4.1 Track Records at ATLAS Region Boundaries trough the Entry-LayerTool

An important part of the ISF MC truth output are the EntryLayer track record collections. EntryLayer collections contain detailed information about simulated particles at the crossing point between certain ATLAS regions (Section 7.3).

The EntryLayers enable subsequent analyses to identify the particles that entered or left certain parts of the ATLAS detector during the detector simulation. This is used to study the detector responses in these regions independently from the other regions a particle might have traversed before or after.

Due to disk space requirements of the detector simulation output, only a subset of all particles traversing ATLAS region boundaries are recorded to respective EntryLayer collections. Particles must be represented in the `TruthEvent` MC truth collection in order to be added to any EntryLayer collection. Thus, changes in the TruthStrategies can indirectly affect the contents of the EntryLayer collections.

The following StoreGate collections contain the EntryLayer records of the respective ATLAS region boundaries:

The `CaloEntryLayer` contains records of particles which are crossing the boundary surface shared between the ATLAS inner detector region and the calorimeter region.

The `MuonEntryLayer` contains records of particles which are crossing the boundary surface shared between the ATLAS calorimeter region and the muon spectrometer region.

The `MuonExitLayer` contains records of particles which are crossing the boundary surface of the ATLAS muon spectrometer region that is not shared with the calorimeter region boundary.

The same criteria apply for the simulation of cosmic ray events and for collision events. The direction in which a particle traverses any of the surfaces mentioned above, is not taken into account when recording the respective EntryLayer collections.

In ISF, the EntryLayerTool fills the various EntryLayer collections according to the criteria mentioned above. It uses the GeoIDService (Section 7.3.1) to determine whether a given particle is located on a boundary surface, and if so, which boundary surface this is.

7.5 Particle Barcodes with the BarcodeService

Particle barcodes are used to associate each simulated sensitive detector hit (or simulated detector response or simulated measurement) with the underlying simulated particle.

The simulated sensitive detector hits created by ATLAS detector simulators contain the barcode of the particle that is causing the hit. Particle barcodes are represented by integer numbers. For a given particle, identical integer values are used to identify this particle in the MC truth representation of the simulated event and in the simulated SD hit. The HepMC format of the MC truth representation requires all particle barcodes to be unique within a given event. Since both, the barcode information in the SD hits and the MC truth representation are propagated through all subsequent Monte Carlo production steps, the barcodes can be used in performance or physics studies to uniquely associate reconstructed objects with MC truth objects.

Particle barcodes in the ISF are computed by a central BarcodeService, which is independent of the ATLAS detector simulation codebase.

The ISF uses the BarcodeService also to generate vertex barcodes, which are unique within a given event. These barcodes are assigned to the vertices representing the particle interactions or decays in the MC truth representation. The vertices connect incoming and outgoing particles for each recorded interaction.

Prior to the ISF, particle barcodes were generated only for particles which are represented in the MC truth. Though, this allows to associate individual detector hits with the particles in the MC truth representation, not all particles in the event may be represented in the MC truth event in the first place. For example, secondary particles with low energies may not be recorded in the MC truth event, but still create SD detector hits. Thus, not all SD hits in the simulation output may be associated with the simulated particle.

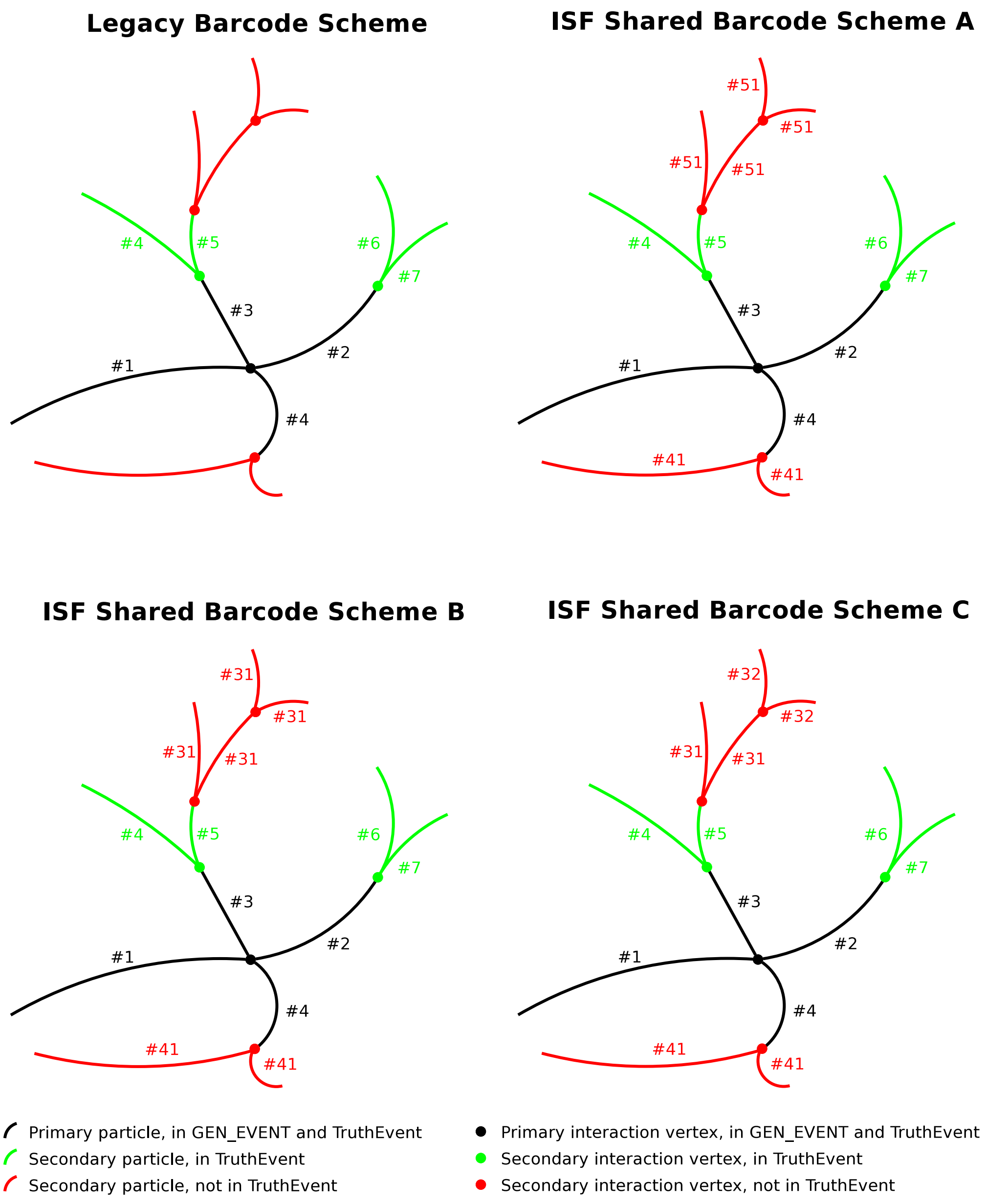
For backwards compatibility, the ISF is capable of reproducing the same barcode scheme as generated by non-ISF full Geant4 and ATLFASTII detector simulation, which is described in the previous paragraph. This is referred to as the *legacy* barcode scheme in this work. However, the ISF also supports a new barcode scheme. In this new scheme, the ISF uses the BarcodeService to compute particle barcodes for all simulated particles, independent of whether or not they are represented in the MC truth. This removes the requirement that all particle barcodes must be unique within the current event. In this new scheme, the ISF creates unique barcodes for all particles which are represented in the MC truth and *shared particle barcodes* for particles which are created in an interaction that is not recorded in the MC truth. The shared barcodes contain encoded information about the primary particle (from the EVNT file) or the last particle present in the MC truth leading up to the creation of the shared barcode particle. Same as for unique particle barcodes, the ISF uses the BarcodeService also to generate the shared particle barcodes. Thus, the amount of information encoded in the shared particle barcodes depends on the specific implementation of the BarcodeService. Figure 7.3 illustrates the difference between the legacy and various shared particle barcode schemes.

Shared particle barcodes do not allow for unique identification of the simulated particle which *directly* created a given SD hit from the simulation output. However, they do contain information which allows to uniquely identify the particle in the MC truth representation which *indirectly* causes the SD hit. A particle indirectly causes a SD hit if either one of its daughter particles or a higher order daughter particle (daughter of a daughter ...) creates the SD hit. In many cases, the association between the primary particles in the EVNT file and the respective SD hits is sufficient – independent of whether the hit was created directly or indirectly. Figure 7.4 illustrates the difference between the legacy and the ISF shared particle barcode scheme, with respect to the sensitive detector hits.

The stand-alone design of the BarcodeService allows for a dual use of one service implementation:

- The BarcodeService **encodes** information into particle and vertex barcodes during the ATLAS detector simulation step.
- The BarcodeService **decodes** information from a given particle or vertex barcode at any stage. This allows analyses to use the BarcodeService to read and interpret barcodes which were generated in a preceding ATLAS detector simulation step.

Different barcode schemes are applied by using a dedicated BarcodeService implementation for each scheme. To guarantee consistency between barcode encoding and decoding, the same BarcodeService implementation must be used in the detector simulation step and in any subsequent analysis.



Legacy Barcode Scheme
#1
#2
#3
#4
#4
#5
#6
#7
ISF Shared Barcode Scheme A
#51
#51
#51
#51
#41
#41
ISF Shared Barcode Scheme B
#31
#31
#31
#31
#41
#41
ISF Shared Barcode Scheme C
#32
#32
#31
#31
#41
#41
Primary particle, in GEN_EVENT and TruthEvent
Secondary particle, in TruthEvent
Secondary particle, not in TruthEvent
Primary interaction vertex, in GEN_EVENT and TruthEvent
Secondary interaction vertex, in TruthEvent
Secondary interaction vertex, not in TruthEvent

Figure 7.3: A comparison of the legacy particle barcode scheme and different ISF shared particle barcode schemes. The ISF schemes are examples to illustrate the capability of the framework, thus the list is not exhaustive. The legacy barcode scheme (upper left) generates barcodes only for particles which are represented in the MC truth (black and green lines). The ISF shared particle barcode scheme also encodes information in secondary particles which are not represented in the MC truth (red lines). Scheme A (upper right) links secondary particles to the last particle present in the MC truth. Scheme B (lower left) links secondary particles to the primary particle. Scheme C (lower right) same as scheme B but it additionally encodes the generation number of the secondary particle.

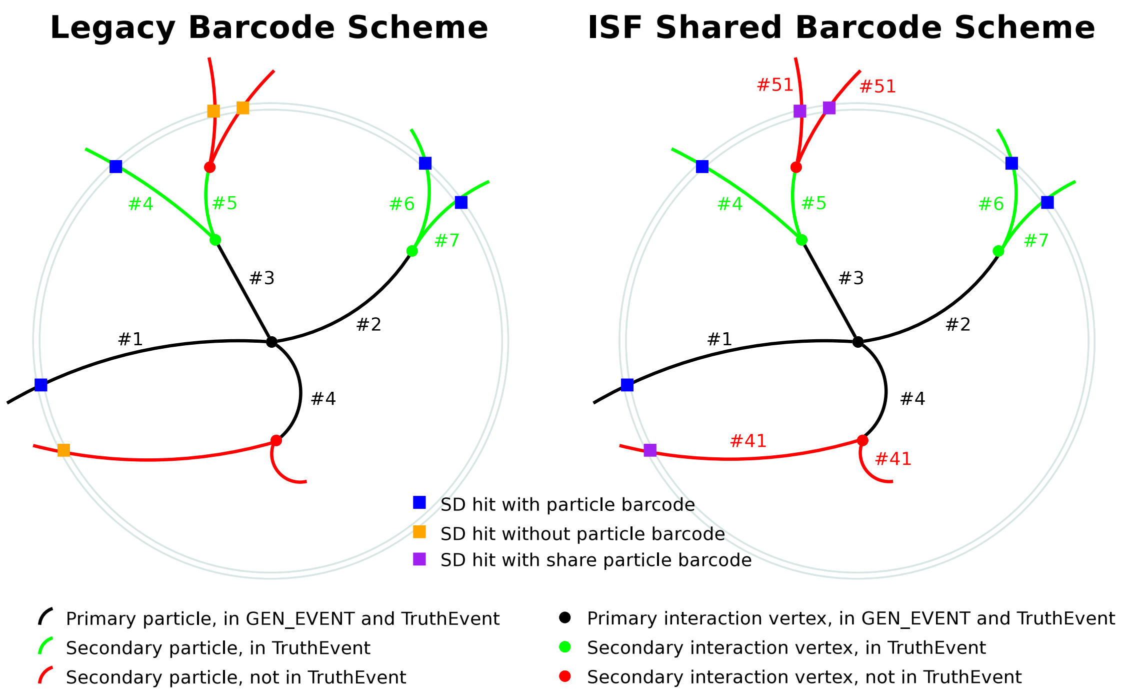

Legacy Barcode Scheme
ISF Shared Barcode Scheme
#1
#2
#3
#4
#5
#6
#7
#4
#1
#2
#3
#4
#5
#6
#7
#4
#41
#41
#51
#51
SD hit with particle barcode
SD hit without particle barcode
SD hit with share particle barcode
Primary particle, in GEN_EVENT and TruthEvent
Secondary particle, in TruthEvent
Secondary particle, not in TruthEvent
Primary interaction vertex, in GEN_EVENT and TruthEvent
Secondary interaction vertex, in TruthEvent
Secondary interaction vertex, not in TruthEvent

Figure 7.4: The legacy particle barcode scheme (left) compared to the ISF shared particle barcode scheme (right). With the same amount of MC truth information (black and green lines), the ISF uses shared particle barcodes (numbers #41 and #51) to store additional information in sensitive detector hits. The shared barcodes associate sensitive detector hits with the root particle that causing the hit (directly or indirectly).

Chapter 8

ISF Program Flow

The individual components of the Integrated Simulation Framework form independent modules with clearly defined functionalities and requirements. Each component in itself was described in the previous chapter. This chapter, on the other hand, describes the interaction between the components and how this forms the entire ATLAS Integrated Simulation Framework.

Prior to processing particles through the ATLAS detector volume, the simulation input is read in and prepared for ISF detector simulation (Section 8.1). Once the simulation input is prepared, the particle loop manages the particle flow until all particles in the event are simulated (Section 8.2). The creation of a consistent Monte Carlo truth representation is an essential component of the ISF detector simulation output (Section 8.3).

8.1 Simulation Input Processing

The simulation input is provided in the EVNT file format (Section 4.2) which contains a tree structure of the particles created by the event generator in the HepMC data format. In many cases, the EVNT file contains a mix of intermediate and final state particles. In such cases, only a subset of the particles in the EVNT file will be simulated through the ATLAS detector. The ISF input processing algorithms identify this subset of particles and convert the data type into an ISF-internal data format, the ISFParticle. In addition to filtering the simulation input, the ISF input processing algorithms set the primary vertex position in accordance with the beam conditions as defined in the detector simulation configuration.

In the default ISF setup, the StackFiller tool manages the tasks described above. The StackFiller tool is called at the beginning of each Athena event. It executes the following tasks in the same order for each Athena event:

1. The StackFiller tool copies the `GEN_EVENT` StoreGate collection from the EVNT input file into the `TruthEvent` StoreGate collection in the HITS output file. Both collections have the exact same contents at this point.

2. The StackFiller tool calls a set of Athena tools (GenEventManipulators), which position the primary vertex in the `TruthEvent` collection in accordance with the beam conditions defined in the simulation configuration. In the same step, a simple sanity check is performed to verify the consistency and validity of the `TruthEvent` at this stage, i.e. vertex coordinates must be finite numbers.

3. For each particle in the `TruthEvent`, a GenParticleFilter tool determines whether or not the particle will be simulated by the ISF detector simulation. Details about the conditions and requirements of the input particle filtering are covered in the following Section 8.1.1. If a particle passes the filter criteria, it is converted into the ISFParticle data format.

The particles which passed the filter criteria in the last step are present in the ISF-Particle data format. The ISFParticles are sent to the ParticleBroker, which determines the corresponding simulator for each particle, by using the routing chains that are defined in the simulation configuration. Once the simulator is determined for each input particle, the particle loop starts (following Section 8.2).

8.1.1 Input Particle Filtering with GenParticleFilters

Three individual filter steps are applied to all input particles, before they are simulated by any of the ATLAS detector simulators in ISF. A particle must pass all filters, otherwise it is discarded from the ISF detector simulation. The filter decisions do not affect the contents or structure of the `TruthEvent` StoreGate collection. The filters decide which subset of the particles that are present in the `TruthEvent` at this stage, will subsequently be converted into ISFParticles and simulated by the ISF.

A final state filter discards all particles which are connected to an end vertex (e.g. intermediate generator particles) and which are of a type that will not interact with the detector material (e.g. neutrions).

A particle position filter only lets particles pass if they are within the ATLAS inner detector region. This filter is disabled for the detector simulation of cosmic ray events.

A pseudorapidity filter requires particles to have a momentum vector with a pseudorapidity of $\eta < 6.0$ in order to be simulated by the ISF. This filter is disabled for the detector simulation of cosmic ray events and for certain upgrade studies.

8.2 The Particle Loop

Inside the particle loop, individual particles are sent to the corresponding simulators for detector simulation. The particle loop ends when neither of the simulators is processing any particles and there are no more particles to be simulated in the current Athena event.

In the ISF, the SimulationKernel (or SimKernel) implements the particle loop. The SimulationKernel is the only Athena algorithm (`AthAlgorithm`) implemented in the ISF. In the particle loop, the SimulationKernel requests particles (ISFParticle data type) from the ParticleBroker and sends them to the corresponding simulator. The ISFParticles that the SimulationKernel receives from the ParticleBroker, are particles for which the ParticleBroker has previously determined the simulator, in accordance with the routing chain configuration. Figure 8.1 illustrates the central role of the SimulationKernel and the information flow between the central ISF components.

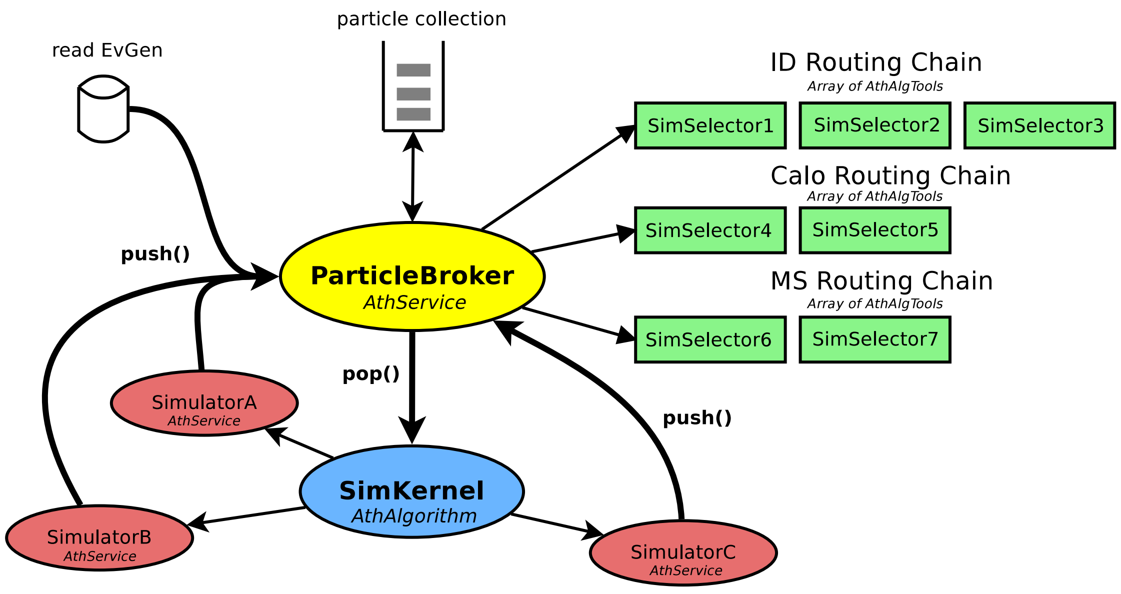


Figure 8.1: The ISF program flow. The initial list of particles is read in from the EVNT file and provided to the ParticleBroker. The ParticleBroker uses routing chains with SimulationSelectors to determine the simulator for each particle in the event. The SimulationKernel sends the particles to the corresponding, previously determined, simulator. The simulators may return secondary particles to the ParticleBroker, for which the simulator is determined in the same manner as for the initial particles. The detector simulation ends when all particles are completely processed through the simulators and no more particles are to be simulated in the current event.

Secondary particles created by a simulator are returned to the ISF ParticleBroker. The latter will determine the appropriate simulator for each particle by applying the routing chain algorithm, based on the SimulationSelectors configured in the simulation setup. Once the simulator is identified, the ParticleBroker temporarily stores the particle in an internal particle collection. Particles are removed from this particle collection and returned to the SimulationKernel upon requests of the latter. The SimulationKernel requests particles when the simulation of the previous set of particles has finished within the respective simulator. All particles which are to be sent to the same simulator, and that are in the same detector region, are returned within one set of ISFParticles from the ParticleBroker to the SimulationKernel. The SimulationKernel will send the entire set of particles to the corresponding simulator at one time. This allows for a more efficient data flow into the simulators which support processing of sets of particles, rather than single particles individually. For instance, this method reduces the computational overhead in the (time critical) Geant4 full detector simulation setup.

If the ParticleBroker does not store any more particles which are to be simulated through the detector, the SimulationKernel receives an empty set of particles. At this point, the detector simulation of all particles in the current event is completed and thus the particle loop ends. The SimulationKernel informs the simulators and the Particle-Broker about the completion of the currently simulated event and the SimulationKernel `AthAlgorithm` execution ends for the current Athena event.

Athena will subsequently finalize the processing of the current event and prepare the input of the next event. Following that, the ISF begins with processing and preparing the simulation input (Section 8.1).

The current ISF SimulationKernel implementation relies on a serial data flow between the SimulationKernel, the simulators and the ParticleBroker. As such, the SimulationKernel does not support concurrent processing and requires the simulators to be executed in the same thread as the SimulationKernel and the ParticleBroker. Due to the modular design of the ISF, a separate SimulationKernel implementation for concurrent processing may be implemented and configured when required, instead of the default SimulationKernel implementation.

8.3 Monte Carlo Truth Recording

The Monte Carlo truth record created by the ISF contains the simulation input as well as the most significant secondary particles that were generated by the detector simulation. The ISF MC truth representation is recorded into the `TruthEvent` StoreGate collection using the HepMC data format.

Figure 8.2 illustrates the flow of information regarding the generation of the MC truth representation by the ISF. In the beginning of each Athena event, the ISF input is read from the `GEN_EVENT` StoreGate collection, which is generated by Monte Carlo event generators, in a step prior to detector simulation. The StackFiller tool (Section 8.1) prepares the simulation input from the contents of the `GEN_EVENT` collection. With the primary vertex position in the `GEN_EVENT` collection located at the origin, the ISF needs to reposition the contents of the `GEN_EVENT` collection to the actual beam position as defined in the simulation configuration. A modified copy of the `GEN_EVENT` with the primary vertex position set in accordance with the beam conditions, is stored into the `TruthEvent` collection at the beginning of each event simulation. The `TruthEvent` contains the actual simulation input particles, which form the first important part of the MC truth representation.

Once the input is converted into the ISF-internal ISFParticle data format, this format is used to exchange particle information between the individual ISF components (StackFiller, ParticleBroker, SimulationKernel and the simulators). The individual simulators may compute particle decays or interactions which potentially need to be recorded to the MC truth representation in the `TruthEvent` collection. A simulator-independent data format, the TruthIncident (or ITruthIncident interface), is used to allow a central TruthService to access details about the interactions occurring inside simulators. The TruthService receives TruthIncidents for all interaction processes occurring in all simulators within the ISF (Figure 8.3). The TruthService uses TruthStrategies to determine which of the TruthIncidents will be recorded into the MC truth representation. If the TruthStrategies determine that a TruthIncident will be recorded persistently, the TruthService converts the TruthIncident information into the HepMC data format. Subsequently, this HepMC-formatted TruthIncident is added to the `TruthEvent` collection.

The TruthService uses a BarcodeService to compute vertex and particle barcodes for particles that are generated in the interaction or decay process (Section 7.5). Since particle and vertex barcodes are intrinsically supported (in fact required) by the HepMC format, the barcodes computed by the ISF BarcodeService are stored in the MC truth representation.

In addition to writing the MC truth representation, the TruthService updates the TruthIncidents with the respective newly computed particle barcodes. The TruthIncident implementation then forwards this information to the simulator. In the simulator, the particle barcodes are recorded with each sensitive detector hit a particle causes. The particle barcode information in SD hits allows for subsequent association of reconstructed and simulated objects. Thus, it is crucial that new particle barcodes generated by the BarcodeService are updated in the simulator for the corresponding particle.

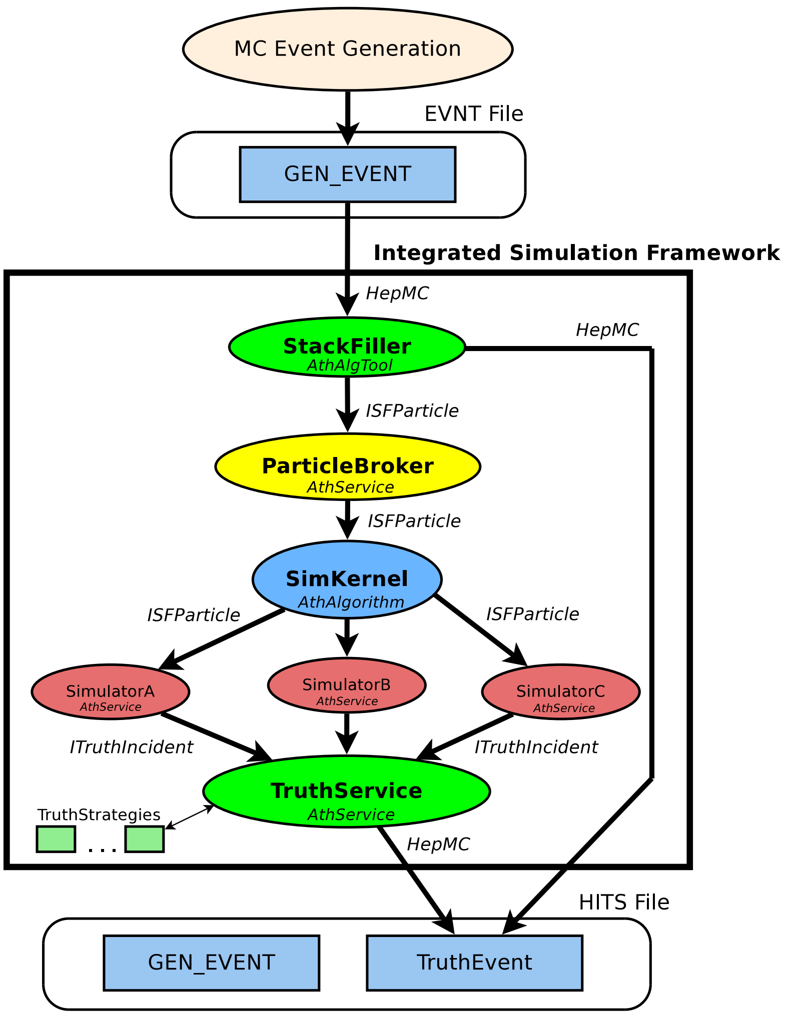


Figure 8.2: The Monte Carlo truth information in the Integrated Simulation Framework has its origin in the `GEN_EVENT` collection created by the MC event generators. The StackFiller copies the `GEN_EVENT` collection into the `TruthEvent` collection and modifies the primary vertex position in the latter in accordance with the beam conditions of the detector simulation setup. The final state particles are converted into ISFParticles and will further be routed through the ParticleBroker and the SimulationKernel, where they get sent to the corresponding simulator. The simulators will compute secondary particles and particle-material interactions. This information is provided to the TruthService in the form of TruthIncidents. The TruthService uses TruthStrategies in order to filter out a subset of the truth information that is recorded into the `TruthEvent` collection. The input `GEN_EVENT` collection and the output `TruthEvent` collection are both in the HepMC format.

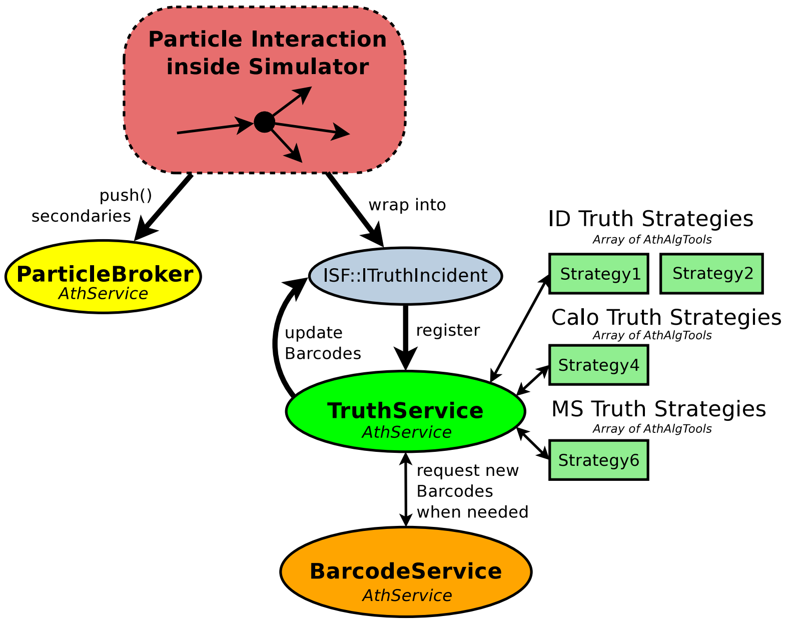


Figure 8.3: The flow of information if a particle-material interaction (or decay) occurs inside a simulator. The primary and secondary particles involved in the interaction are wrapped into a TruthIncident by the simulator. This truth incident is subsequently registered with the central ISF TruthService. The TruthService uses TruthStrategies to determine whether the interaction will be permanently stored into the `TruthEvent` StoreGate collection. A barcode service is used to compute barcodes for the new secondary particles and the interaction vertex. The TruthService updates the particle barcode information in the TruthIncident, which in turn updates the particle barcode used inside the simulator. Some simulators may return the secondary particles to the ISF ParticleBroker for subsequent processing.

Chapter 9

Simulator Integration into the Integrated Simulation Framework

With the core components and new functionalities of the ISF covered in the previous chapters, this chapter describes how the individual simulators are integrated into the ISF. As described in Section 4.3, different full and fast detector simulators are in use by the ATLAS collaboration. Most of the simulators covered in this thesis were developed prior to the ISF and needed to be adopted to function within the framework.

The integration of the Geant4 simulator into the ISF is covered in Section 9.1. The Fatras simulator was re-implemented for the ISF and is discussed in Section 9.2. Section 9.3 covers the integration of FastCaloSim into the ISF, which also enables a new way of combining FastCaloSim with inner detector simulation. The integration of the new ”particle killer” simulator is covered in Section 9.4.

The combined detector simulation setups ATLFASTII and ATLFASTIIF make use of the simulators described above. They are covered in Section 9.5.

9.1 Geant4

Due to Geant4 being the most accurate simulator for the ATLAS detector, it is one of the most critical simulators within the ISF. In particular so, as it may be used for signal particle simulation in advanced particle routing concepts (for example in Chapter 12).

Geant4 also offers the capability to execute fast simulation methods in dedicated regions of the detector. Since the ISF serves as a central framework which steers different types of ATLAS detector simulation, it was decided to integrate Geant4 as a simulator among others in the framework. Geant4 fast simulation methods have not been used in official ATLAS MC production so far, thus the following approach is to be tested in the future. If Geant4 fast simulation techniques were used within the ISF, the decision for a particle to be simulated by this fast method will be taken by the central ISF routing algorithm. In this case, one Geant4 instance may be registered as two separate Geant4 simulators to the ISF: one for full simulation and one for fast simulation. The difference between both simulators being the conversion of the ISF particle information into the respective Geant4 types. The particles will be processed by the Geant4 instance common to both.

The Geant4 simulation toolkit is configured through the FADS (Framework for ATLAS Detector Simulation) for ATLAS detector simulation. The use of the FADS by the ATLAS collaboration predates the Athena framework and consequently also the ISF. For ISF detector simulation, where Geant4 is one of potentially many simulators processing particles within an event, the FADS is used to configure solely the Geant4 simulator. Very minor modification to the FADS were required to integrate the Geant4 configuration into the ISF.

New routines are implemented to enable the use of Geant4 within ISF. The Geant4 TransportTool and the Geant4 SimService implementation are essential Athena tools and services for passing information between the ISF core framework and the Geant4 simulation toolkit. During the initialization phase of the detector simulation job, the Geant4 TransportTool triggers the initialization of the following Geant4 items:

Geant4 RunManager : The ATLAS-specific implementation of Geant4 RunManager (G4AtlasRunManager) is the first component of Geant4 which needs to be instantiated and initialized to integrate the Geant4 simulation toolkit into ISF. No modifications to the G4AtlasRunManager from stand-alone Geant4 ATLAS detector simulation setup were required for its integration into ISF. The Geant4 TransportTool uses a G4RunManagerHelper to retrieve and initialize the G4AtlasRunManager. The helper avoids double-initialization of the ATLAS Geant4 RunManager. This is required as other components in the ISF (such as the Fatras hadronic interaction processor) can trigger the instantiation and initialization of the Geant4 RunManager as well within the same ISF simulation process.

ATLAS detector geometry (GeoModel) : The Geant4 SimService coordinates the initialization of the ATLAS detector geometry with the GeoModelService Athena service.

Magnetic field : The Geant4 TransportTool controls the magnetic field initialization of the Geant4 simulation toolkit.

Geant4 random number stream : In the ISF, each simulator must utilize a separate random number stream. Thus, a dedicated Geant4 random number stream is configured in the initialization phase of the TransportTool.

FADS user actions : The FADS implements all Geant4 user action types, that is: EventAction, StackingAction, SteppingAction and TrackingAction. The FADS user actions are registered with Geant4 at the beginning of the Athena job and they are configured specifically for ISF Geant4 detector simulation. To integrate Geant4 into the ISF, two new FADS TrackingAction are implemented and used by Geant4 (see Section 9.1.1): the MCTruthUserAction and the TrackProcessorUserAction.

Particles in the ISF which are to be simulated by Geant4, are sent to the Geant4 SimService by the ISF SimulationKernel from inside the particle loop. Upon receiving a set of ISFParticles, the Geant4 SimService calls the Geant4 TransportTool, which instantiates a new G4Event and converts each respective ISFParticle into a G4PrimaryParticle representation. The G4PrimaryParticles are added to the G4Event and the Geant4 RunManager is instructed to process the G4Event.

Once the event processing in the Geant4 RunManager terminates, the Geant4 TransportTool and the Geant4 SimService return control to the particle loop inside the ISF SimulationKernel.

9.1.1 FADS User Actions for ISF Integration

Due to the specific requirements of the ISF to the simulators (Section 7.1.1), two new FADS user actions are implemented and used by Geant4 inside the ISF.

MCTruthUserAction

The MCTruthUserAction monitors in each Geant4 simulation step, whether secondary particles were generated by Geant4. If so, the MCTruthUserAction wraps the information regarding this interaction into the TruthIncident data format and registers this TruthIncident with the central ISF TruthService. The ISF TruthService will subsequently determine whether the interaction is recorded persistently or not (Section 8.3).

TrackProcessorUserAction

The TrackProcessorUserAction determines in each Geant4 simulation step, whether the currently simulated particle has left the current ATLAS region. A main requirement by the ISF particle routing algorithm is, that simulators return all particles (primary and secondary) on boundaries between ATLAS detector regions. This is necessary as the particle routing decision must be re-evaluated for each particle in each ATLAS detector region, since separate routing chains exist for each respective ATLAS region.

The TrackProcessorUserAction uses the fast GeoIDService implementation (Section 7.3.1) to identify the ATLAS region of each particle in every Geant4 simulation step. If the ATLAS region of a particle changes compared to its previous step, the simulation of the particle inside Geant4 ends, the particle is converted into the ISFParticle data format and returned to the ISF ParticleBroker.

There are two cases, where particles traversing an ATLAS region boundary are not returned to the ISF ParticleBroker:

1. To lower the computational costs of sending high numbers of (less significant) **particles with low energies** from Geant4 to ISF, particles must have at least 50 keV of kinetic energy to be sent to the ISF ParticleBroker when traversing an ATLAS region boundary in Geant4. If this energy threshold is not met, the particle will remain inside Geant4 until its end.

2. To lower the computational costs of the **full Geant4 simulation setup** in ISF, particles crossing an ATLAS region boundary in this simulation setup are not

returned to the ISF routing algorithms. Their simulation continues within Geant4 until the end of each individual particle. For the creation of a consistent MC truth representation, it is required that the ISF ParticleBroker is informed about particles crossing an ATLAS region boundary. However, the particle is not removed from Geant4 simulation at that point.

9.2 Fatras

Fatras was re-implemented for the ISF and thus its implementation is driven by the requirements of the simulation framework. Fatras uses the track reconstruction tools to simulate particles through the ATLAS detector volume. The reconstruction geometry serves as the ATLAS detector description.

The ISF SimulationKernel sends particles to the FatrasSimService (also FatrasSimSvc) for Fatras detector simulation. The FatrasSimService calls the Fatras TransportTool, which converts the ISFParticle format into the corresponding data type of the track reconstruction project (CurvilinearParameters). The TransportTool uses the TimedExtrapolator implementation to simulate the particle through the ATLAS detector material.

The TimedExtrapolator is configured to call Fatras-specific Athena tools in order to compute particle decays and interactions within the detector material. Parameterized models are used to accurately simulate electromagnetic particle-matter interactions in Fatras. Hadronic interactions are simulated either through a parameterized model or through Geant4 hadronic interaction modules. In the latter case, Fatras uses a G4RunManagerHelper to coordinate the initialization of the Geant4 RunManager with the ISF Geant4 implementation. If the Geant4 hadronic interaction module is used, Fatras converts the material description of the detector layer (from the reconstruction geometry) that is currently traversed by a particle, into a Geant4 material description and executes the corresponding Geant4 interaction routines. To simulate particle decays, Fatras accesses the Geant4 particle decay table and determines the path lengths of unstable particles in the detector simulation.

Due to the lack of a Fatras-internal particle loop, any secondary particle that is created during the Fatras detector simulation is sent to the ISF ParticleBroker. The ParticleBroker applies the routing chain algorithm to determine the simulator for these particles and caches them until the SimulationKernel requests them for detector simulation in the respective simulator. If Fatras computes and interaction process where secondary particles are created but the primary particle is not consumed (e.g. bremsstrahlung), the updated primary particle will remain within Fatras and only the secondary particles are returned to the ISF ParticleBroker.

To allow for the creation of a consistent MC truth representation, any particle interaction computed by Fatras which creates secondary particles is registered to the central ISF TruthService.

The sizes of the ATLAS regions in the TrackingGeometry that is used by Fatras, coincide with the ISF definition of the ATLAS regions (Section 7.3). This enables the

TimedExtrapolator to terminate particle extrapolation at region boundaries and subsequently allows Fatras to return all particles to the ISF ParticleBroker when they leave an ATLAS region.

9.3 FastCaloSim

To integrate FastCaloSim into the ISF, a wrapper Athena service is implemented which calls the same core FastCaloSim implementation which is used for ATLFASTII detector simulation outside the ISF. This Athena service is called FastCaloSimSvc and it receives particles from the ISF SimulationKernel if they are to be simulated by the FastCaloSim.

The FastCaloSimSvc implementation allows for two different modes of operation. In the *legacy mode*, FastCaloSim uses the MC truth representation at the end of each simulated event to compute the calorimeter response. This mode is the standard mode in the traditional ATLFASTII setup and is covered in more detail in Section 9.3.1. In the *ISF mode* (or *non-legacy mode*), FastCaloSim computes the calorimeter response individually for each particle that it receives from the ISF SimulationKernel. This mode is covered in more detail in Section 9.3.2.

In either mode, FastCaloSim generates a calorimeter response in a reconstruction-level data format, the CaloCellContainer. This CaloCellContainer is not recorded persistently and remains in transient storage within StoreGate until the end of each respective Athena event. For compatibility reasons, the CaloCellContainer is converted into the a hits-level data format, which is the same format that Geant4 detector simulation generates. This conversion is triggered by the FastCaloSimSvc at the end of each Athena event. This renders the specific ISF simulation setup transparent to subsequent digitization and reconstruction steps. That is to say, the same digitization and reconstruction configuration may be applied on the ISF simulation output, independent of whether or not FastCaloSim was used.

If enabled in the ISF simulation setup, the FastCaloSimSvc calls the fast calorimeter punch-through simulation 9.3.3 for each individual particle it receives from the ISF SimulationKernel.

9.3.1 Legacy Mode

The legacy FastCaloSimSvc mode is the default mode of operation for ATLFASTII detector simulation within the ISF and outside of the ISF. In this mode, FastCaloSim computes the full calorimeter response at the end of each Athena event, based on the preceding ATLAS inner detector simulation output. FastCaloSim must be the only simulator in this detector setup that computes sensitive detector hits in the calorimeter – with the exception of muon particles that are passing through the ATLAS calorimeter.

In the legacy mode, FastCaloSim parses the MC truth representation (`TruthEvent` StoreGate collection) for input particles in order to compute the calorimeter response. Using the HepMC tree structure present in the MC truth representation, FastCaloSim traces secondary particles with lower energies in the inner detector region back to common parent particles with higher energies. The calorimeter response is computed from the higher energy particles identified in the ATLAS inner detector region. The FastCaloSim parametrization takes into account the possible loss of energy due to interaction processes which occur to these particles when traversing the ATLAS inner detector volume.

This approach requires that the MC truth representation in the inner detector region is complete at the time the calorimeter response is computed by FastCaloSim. This is requirement is fulfilled, if the simulation of all particles in all ATLAS regions (except for the calorimeter energy deposits) is complete at the time FastCaloSim is called. Thus, in the legacy mode the FastCaloSim energy deposits are computed after the particle loop in the ISF SimulationKernel has ended. At this point, the simulation of all particles in the event is completed (except for FastCaloSim) and the contents of MC truth representation is unchanged until the end of the Athena event.

The simulation output of FastCaloSim in the legacy mode is most accurate with the MC truth representation generated by the Geant4 inner detector simulation (Section 11.1).

9.3.2 ISF Mode (Non-Legacy Mode)

The ISF mode (or non-legacy mode) the FastCaloSimSvc computes the fast calorimeter response for each particle individually when it is received by the FastCaloSimSvc from the ISF SimulationKernel. This allows for other calorimeter simulators to be present in the same detector simulation setup besides FastCaloSim. Thus, ISF simulation setups with advanced routing rules (e.g. regions of interests) with different types of simulators for the calorimeter region are required to use the ISF mode of the FastCaloSimSvc.

The particles received by the FastCaloSimSvc are typically positioned on the boundary between the ATLAS inner detector and the calorimeter region. This is due to the ISF requirement that all particles are to be returned to the ISF ParticleBroker at ATLAS region boundaries. At this point the particle routing algorithm determines the simulator of the particle in the next region.

In ISF mode, the FastCaloSimSvc converts each incoming ISFParticle data type into the corresponding HepMC type before computing the calorimeter response. This is required as the same core FastCaloSim implementation is used to compute the calorimeter response in the ISF mode and the legacy mode. In the legacy mode, the core FastCaloSim implementation would read the input from the HepMC-formatted `TruthEvent` StoreGate collection. Thus, the core FastCaloSim implementation requires the input to be in the HepMC data format.

If the ISF mode were used in the ATLFASTII detector setup and compared to the legacy mode, the overall FastCaloSim calorimeter response would consist of a higher number of individual calorimeter responses for particles with lower energies on average.

9.3.3 Fast Calorimeter Punch-Through Simulation

A parameterized calorimeter punch-through simulator [88] is integrated into the FastCaloSimSvc. If enabled, the punch-through simulator evaluates punch-through probabilities and computes calorimeter punch-through particles for each individual particle that is received by the FastCaloSimSvc. The ISFParticle data format serves directly as input to the punch-through simulator, thus a type conversion is not required.

In the current implementation, the punch-through simulation output is not correlated with the calorimeter response that is computed by FastCaloSim. However, current developments regarding a re-parametrization of FastCaloSim will lead to a newly parameterized punch-through simulation which will be correlated with FastCaloSim.

9.4 Particle Killer Simulator

The "particle killer" simulator enhances the capabilities of the ISF and allows for the use advanced routing configurations within the framework. Upon receiving a particle from the ISF SimulationKernel, this simulator removes the particle from the ISF detector simulation at this point. Thus, the "particle killer" simulator does not create sensitive detector hits nor does it generate secondary particles.

This simulator is implemented in the ParticleKillerSimSvc Athena service.

The ParticleKillerSimSvc is used in the ATLFASTII and ATLFASTIIF ISF setups to remove particles which are outside the parameter space relevant for fast detector simulation (i.e. particles with a high pseudorapidity).

9.5 Combined ATLAS Detector Simulation Setups

A main requirement for the design of the ISF the implementation of all simulation setups that are in use by the ATLAS collaboration. As such, the ISF is required to support combined ATLAS detector simulation setups, such as ATLFASTII (Section 9.5.1) and ATLFASTIIF (Section 9.5.2).

The combined simulation setups make use of the integration of the individual simulators which are covered in the previous sections.

9.5.1 ATLFASTII

The ATLFASTII simulation setup (Section 4.4.1) is an essential component of fast Monte Carlo production for the ATLAS collaboration. ATLFASTII consists of Geant4 inner detector simulation, Geant4 simulation for muon particles throughout the entire detector and FastCaloSim to compute the calorimeter response.

The ATLFASTII configuration in ISF consists of the following routing rules in each detector region:

- **Forward region**

1. Simulate all particles with Geant4.

- **Inner detector region**

  1. Simulate all particles with Geant4.

- **Calorimeter region**

  1. Simulate all muon particles with Geant4.
  2. Destroy particles if they have a momentum vector of $|\eta| > 5$, i.e. simulate these particles with the ParticleKillerSimSvc.
  3. Simulate all particles with FastCaloSim (legacy mode).

- **Muon spectrometer region**

  1. Simulate all particles with Geant4.

- **Cavern region**

  1. Destroy all particles, i.e. simulate all particles with the ParticleKillerSimSvc.

Due to FastCaloSim being optimized for the legacy mode in the ATLFASTII setup outside of ISF, this mode also provides the most accurate simulation results inside ISF (Section 11.1). Thus, the FastCaloSim legacy mode was chosen for ATLFASTII detector simulation within the ISF.

9.5.2 ATLFASTIIF

The ATLFASTIIF simulation setup (Section 4.4.2) forms a fast ATLAS detector simulation by combining the Fatras and FastCaloSim simulators. The ATLFASTIIF setup is almost identical to the ATLFASTII setup if the Geant4 simulator is replaced by the Fatras simulator:

- **Forward region**

  1. Destroy all particles, i.e. simulate all particles with the ParticleKillerSimSvc.

- **Inner detector region**

  1. Simulate all particles with Fatras.

- **Calorimeter region**

  1. Simulate all muon particles with Fatras.
  2. Destroy particles if they have a momentum vector of $|\eta| > 5$, i.e. simulate these particles with the ParticleKillerSimSvc.
  3. Simulate all particles with FastCaloSim (non-legacy mode).

- **Muon spectrometer region**

  1. Simulate all particles with Fatras.

- **Cavern region**

  1. Destroy all particles, i.e. simulate all particles with the ParticleKillerSimSvc.

Due to the increased systematic error of combining the FastCaloSim legacy mode with Fatras inner detector simulation (Section 11.2), the ISF mode of FastCaloSim is applied in the ATLFASTIIF detector simulation setup.

Part III

ISF Fast Simulation Results and Performance Measurements

Chapter 10

ISF Computing Performance

The main aim of the Integrated Simulation Framework is to achieve a significant reduction of the computing time required for the production of ATLAS detector simulation samples. This is achieved as the ISF allows to combine full and fast detector simulation technologies for the simulation of individual events. With the ISF serving as the common detector simulation framework for any ATLAS detector simulation, it is essential that the computational cost of the components of the framework are kept to a minimum.

This chapter covers the methods used to quantify the computing performance of the ISF as well as the results obtained for different ISF simulation setups (also called simulation flavours). The tools that are used to measured the computing performance of ATLAS offline software algorithms are discussed in Section 10.1. A study comparing the average simulation time per event for different ISF simulation flavours is presented in Section 10.2. Detailed CPU profile measurements of different ISF simulation setups are presented in Section 10.3.

10.1 Performance Analysis Tools

Fundamental to any performance analysis are the tools used to quantify the computing performance of the tested algorithm(s). The two tools presented in this section are integrated into the ATLAS Athena software framework and are used to measure the performance of the ISF: PerfMon [89, 90] and GPerftools [90, 91]

10.1.1 PerfMon

PerfMon is a lightweight performance monitoring tool, developed by the ATLAS collaboration specifically for the Athena software framework. It measures CPU and memory consumption of Athena processes at the level of individual Athena algorithms, tools and services. The computational overhead introduced by PerfMon is less than 0.1% additional CPU time for production-type jobs (simulation, digitization, reconstruction).

The PerfMonSD (semi-detailed) mode is enabled for all ATLAS production jobs in order to provide data for performance monitoring and performance analysis. In the finalization phase of an Athena process, PerfMonSD provides the measured average CPU time required to process one event and the memory allocation of the current job. The performance analyses presented in sections 10.2 and 12.5.2 are based on these measurements.

10.1.2 GPerftools

GPerftools (or gperftools, formerly Google Performance Tools) is a lightweight performance analysis tool. It provides a C++ API to enable CPU and heap profiling in user-implemented applications. GPerftools records a program's stack trace at regular time intervals (default: 100 stack traces recorded per second) to provide CPU profiling information. After the program has terminated, GPerftools performs an analysis of the recorded stack traces and provides the results in various data formats (callgrind, plain text, PDF, etc.). Computationally demanding functions will appear more often in the stack traces compared to functions which need only little time to complete execution. Thus, the number of appearances of a function in the stack traces indicates the computational costs of this function as well as the time spent inside this function during program execution.

GPerftools is integrated into the ATLAS software framework to allow profiling of Athena-based programs. The `GPT::ProfilerService` (with the `GPT::IProfilerSvc` interface) is an `AthService` implementation which accesses the GPerftools API to steer CPU profiling of Athena-based programs. Figure 10.1 illustrates the UML class diagram of the `GPT::ProfilerService` implementation. To avoid profiling the initialization and finalization stage of Athena-based programs, the `GPT::ProfilerService` is usually configured to activate CPU profiling only during the execution stage of Athena programs.

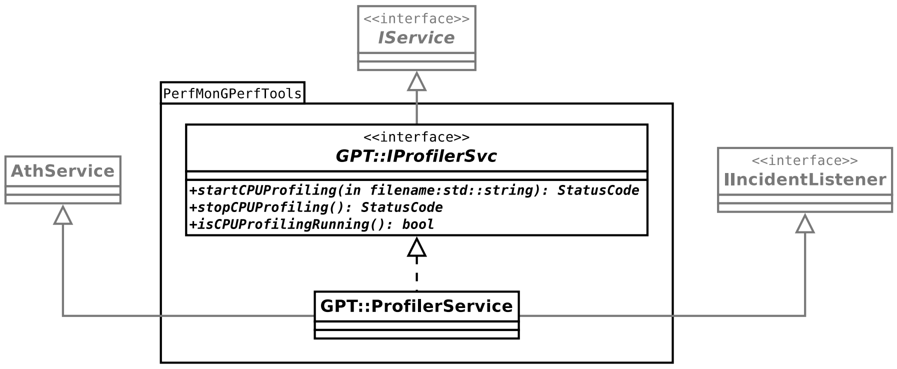


Figure 10.1: The UML class diagram of the `GPT::IProfilerSvc` interface and `GPT::ProfilerService` implementation. The latter accesses the GPerftools API to steer CPU profiling of Athena-based programs.

The `gathena` command line tool is implemented as a wrapper to the `athena` command. It enables CPU profiling for the provided Athena job.

The current `GPT::ProfilerService` implementation does not support heap profiling but this functionality may be added to the Athena service and its interface in the future.

10.2 CPU Time per Simulated Event

A quick and powerful indicator to evaluate simulation performance is the average CPU time required to simulate one event. This measurement is carried out by the PerfMonSD tool (Section 10.1.1) for all production-type jobs, and the results are provided at the end of the respective Athena job. Table 10.1 shows the required CPU time per event to simulate $t\bar{t}$ events with differently configured ISF simulation setups. A significant increase in simulation speed is observed when replacing the Geant4 calorimeter simulation with FastCaloSim in ATLFASTII. This is consistent with previous measurements, which determined that the ATLAS Geant4 detector simulation time is dominated by the calorimeter sub-system [80]. Additionally replacing Geant4 by Fatras simulation in all sub-detectors achieves even higher simulation speeds with the so-called ATLFASTIIF setup. The accuracy of Fatras is studied in [78].

Detector Simulation Setup	CPU time (s/event)	Speedup w.r.t. full G4
Full Geant4	195 ± 9	1
ATLFASTII	17.1 ± 0.3	11
ATLFASTIIF	0.919 ± 0.003	212

Table 10.1: The average CPU time required to simulate one $t\bar{t}$ event with differently configured ISF setups in the ATLAS offline software release `17.7.5.4`. The CPU time is averaged over 50 events simulated in Geant4, 500 events simulated in ATLFASTII and 8000 events simulated in ATLFASTIIF. The CPU time required to process the first event is not included in each respective measurement. The individual simulation processes were executed one at a time and they were the only high-workload processes on the computer[2]at the time of the measurement.

10.2.1 Partial Event Simulation

With the implementation of the ISF, new types of very fast ATLAS detector simulation approaches are now possible. Table 10.2 shows the average CPU time required to simulate one $ggH \to \gamma\gamma$ event through the ATLAS detector with differently configured ISF setups. The fastest setup uses a new technique called *partial event simulation.* For partial event simulation, only a subset of the particles of the detector simulation input are processed through the detector. This is enabled due to the implementation of the ISF

[2]The specifications of the computer on which the measurements are performed: Scientific Linux CERN 6, 64 GiB memory and 4 Intel® Xeon® E5-2650 v2 @ 2.60 GHz processors.

and has not be possible with previously existing ATLAS detector simulation frameworks. The ATLFASTIIF partial event simulation presented in this table only simulates particles in cones around signal photons. Due to this, the majority of particles created by the event generator are skipped in the detector simulation step, which leads to a significantly increased execution speed compared to plain ATLFASTIIF. Figure 10.2 illustrates the cone-shaped region of interest around the signal photons. Fatras and FastCaloSim create sensitive detector hits only for the particles inside these cones.

Detector Simulation Setup	CPU time (s/event)	Speedup w.r.t. full G4
Full Geant4	560	1
ATLFASTII	25	~ 25
ATLFASTIIF	0.75	~ 750
ATLFASTIIF partial event simulation	0.18	~ 3000

Table 10.2: The average CPU time required to simulate one $ggH \to \gamma\gamma$ event through the ATLAS detector with differently configured ISF setups in the ATLAS offline software release `17.6.0.7`. In the ATLFASTIIF partial event simulation setup only the particles in cones (size $\Delta R = 0.4$) around the signal photons are simulated, the rest of the particles in the event are discarded from detector simulation. This leads to a significantly faster detector simulation compared to the plain ATLFASTIIF setup.

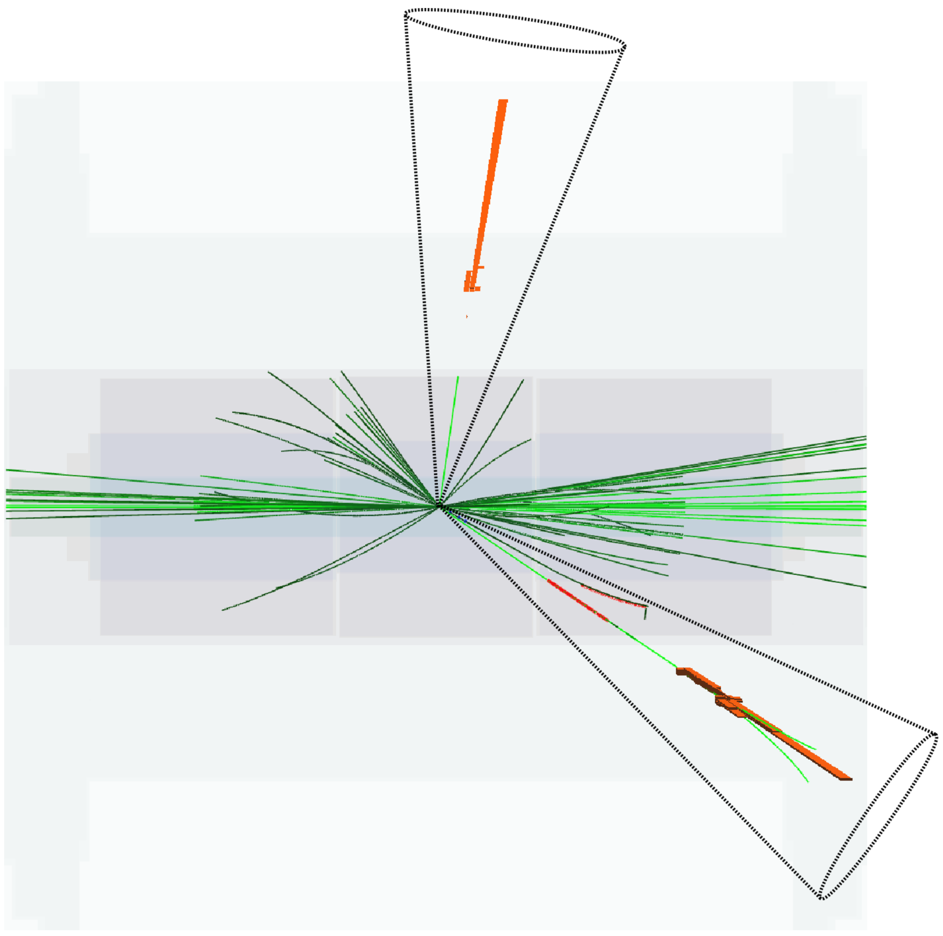


Figure 10.2: Event display of ATLFASTIIF partial event simulation for $ggH \rightarrow \gamma\gamma$ events in ATLAS offline software release `17.6.0.7`. Cone-shaped regions of interest (size $\Delta R = 0.4$) are defined around the signal photons. Only the particles inside these ROIs are simulated by Fatras and FastCaloSim. Particles outside the ROIs are not simulated through the detector. The green lines are particle trajectories estimated by the event display, based on the contents of the Monte Carlo truth representation (including event generator particles). Fatras SD hits are illustrated as dots along charged particle trajectories. FastCaloSim energy deposits are illustrated as three-dimensional bins representing the amount of energy deposited in different calorimeter cells.

10.3 ISF CPU Profiling

This section covers the use of CPU profilers to determine the computational cost of different components within the ISF detector simulation. The GPerftools CPU profiler (Section 10.1.2) is used to generate the individual measurements. The CPU profiles were generated one at a time on the same machine[3]. The profiled processes were the only high-workload processes scheduled on this computer at the time of the measurement.

10.3.1 ISF Geant4 Detector Simulation

With the Geant4 detector simulation setup being the most accurate detector simulation available to the ATLAS collaboration, it is of particular importance for many physics analyses and performance studies. However, the high accuracy comes at the price of high computational costs (see Section 10.2). Thus, the implementation of the Geant4 simulation setup is highly optimized within the ISF in order to minimize the computational costs of the individual components and of the entire setup.

Figure 10.3 shows the CPU profile measured for full Geant4 simulation within the ISF, visualized with KCachegrind [92]. In this image, single computationally demanding functions appear as large areas filled with a solid colour. Such functions are often the first candidates for further optimization as they may form a performance bottleneck in the profiled program. A performance improvement of these functions may result in an improvement of the overall program performance. The figure reveals that there are no obvious performance bottlenecks in the full Geant4 setup within the ISF.

The following functions, however, are pronounced in the profile and may become subject to further performance optimizations:

- `master.0.gbmagzsb_` and `bsolinterp_` are Fortran 90 implementations of the magnetic field computation inside the ATLAS detector volume. In newer ATLAS offline software releases (version `19.0.0` and higher) this implementation is replaced by a more efficient algorithm written in C++.

- `iGeant4::TrackProcessorUserAction::SteppingAction` is a Geant4 user action implementation specifically for ATLAS detector simulation with the ISF. This function is called for each simulation step that any particle makes within Geant4 in ISF. Even though this function is highly optimized, the fact that it is typically called millions of times per simulated event, adds up the costs of each individual function call to a measurable fraction of the overall simulation time.

- `LArWheelSolid::search_for_nearest_point` and `LArWheelCalculator` are used to implement a custom geometrical shape (custom Geant4 solid) that describes the ATLAS LAr EMEC (liquid argon electromagnetic endcap calorimeter) volume. Due to the complex accordion-like internal structure of the EMEC, costly

[3]The specifications of the computer are: Scientific Linux CERN 6, 64 GiB memory and 4 Intel® Xeon® E5-2650 v2 @ 2.60 GHz processors.

computation is required for the navigation of Geant4 particles through this volume. An ongoing project is aiming at speeding up the implementation used in the CPU profile. Significant improvements have been achieved by code refactoring and more efficient use of costly mathematical functions and branches. The results are currently being validated for future production use.

- `G4VProcess::SubtractNumberOfInteractionLengthLeft`, `G4CrossSectionDataStore::GetCrossSection`, `G4UniversalFluctuation::SampleFluctuations` and other functions of the Geant4 simulation toolkit (all class names starting with `G4`) appear as potential candidates for optimization in the CPU profile. These functions are not specific to the ATLAS detector simulation and are developed, maintained and optimized by the Geant4 collaboration.

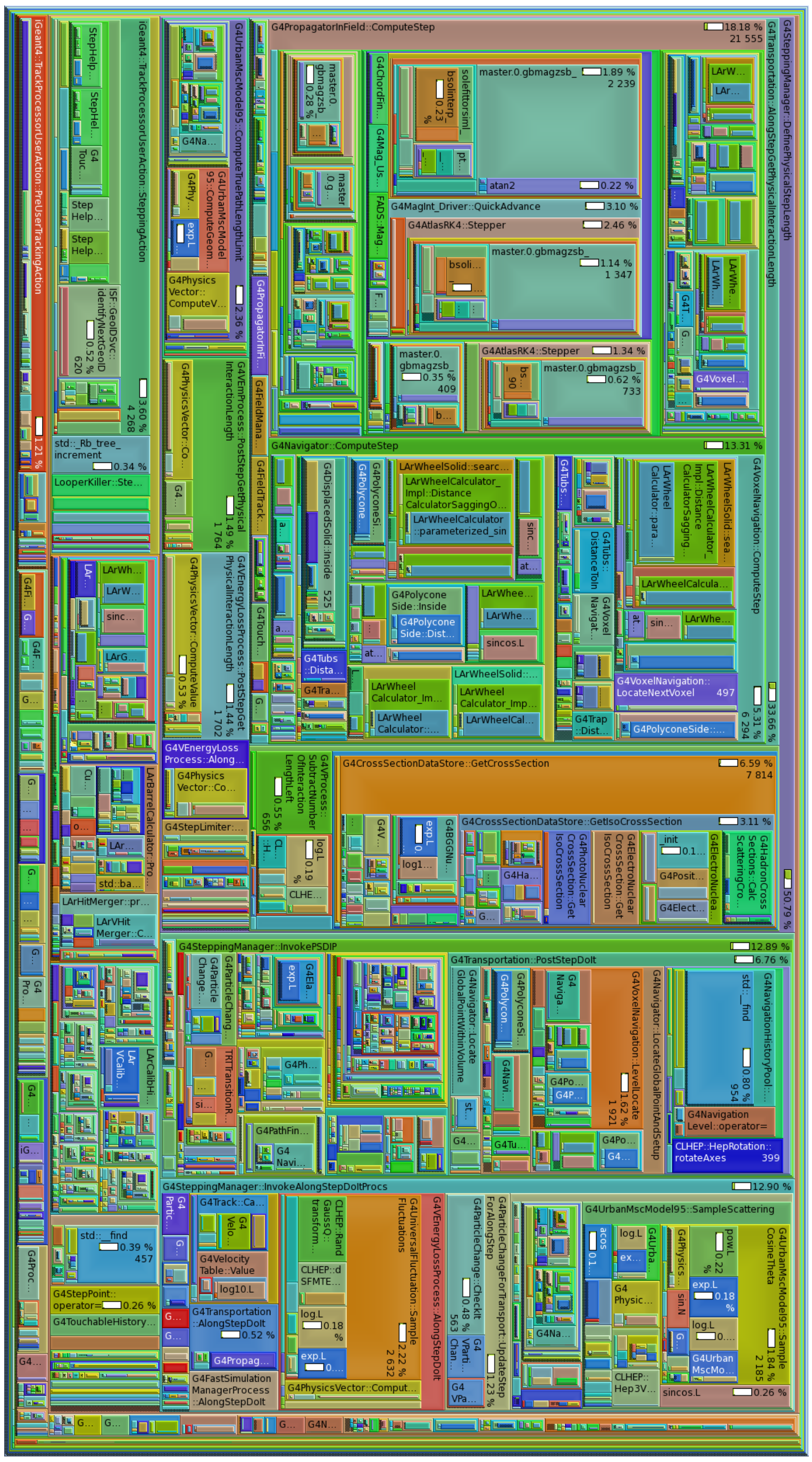


Figure 10.3: CPU profile for Geant4 detector simulation of 7 $t\bar{t}$ events with the ISF in the ATLAS offline software release 17.7.7.5.4. The profile is recorded with the GPerftools CPU profiler and visualized with KCachegrind.

10.3.2 ISF ATLFASTII Detector Simulation

The ATLFASTII simulation setup has been used extensively by physics analyses and performance studies throughout the previous years. Due to the use of fast simulators for the calorimeter, the ATLFASTII setup is significantly faster than the full Geant4 detector simulation (Section 10.2). With the high simulation speed achieved due to this algorithmic changed with respect to full Geant4, less optimization effort was spent to further improve the computing efficiency of the ATLFASTII implementation. Thus, a few hot spots are noticeable in the respective CPU profiles.

Figure 10.4 shows the CPU profile obtained with GPerftools for ATLFASTII detector simulation in the ISF. The most noticeable hot spots in the profile are:

- Similar to the full Geant4 detector simulation, the functions `master.0.gbmagzsb_` and `bsolinterp_` for the computation of the magnetic field inside the ATLAS volume, appear as significant hot spots in the CPU profile. Due to the increased simulation speed of ATLFASTII with respect to full Geant4, the magnetic field computation accounts for a two to three times higher fraction of the total CPU costs in the ATLFASTII setup.

- Again, similar to the full Geant4 simulation setup, various functions of the Geant4 simulation toolkit are noticeable as potential hot spots in the CPU profile, namely `G4UnionSolid::Inside`, `G4DisplacedSolid::Inside`, `G4CrossSectionDataStore::GetCrossSection`, etc. However, since Geant4 simulates particles mainly through the ATLAS inner detector in the ATLFASTII setup, different functions of the Geant4 simulation toolkit appear as hot spots compared to the full Geant4 detector simulation setup.

- The `iGeant4::TrackProcessorUserAction::SteppingAction` function appears in the profile, with about half of the relative cost compared to the full Geant4 detector simulation setup. This is likely due to Geant4 mainly simulating particles inside the ATLAS inner detector volume in the ATLFASTII setup. Lower particle multiplicities and higher average particle energies can be expected for Geant4 inner detector simulation compared to Geant4 calorimeter simulation. Hence, fewer simulation steps are being executed by Geant4 in ATLFASTII, which results in fewer calls to the Geant4 user action.

The CPU profile and call tree in Figure 10.5 reveals that about 94% of the CPU time in ISF ATLFASTII simulation is spent inside Geant4 and about 4% is spent inside FastCaloSim routines. The remaining 2% CPU time is spent inside core Athena components (e.g. input and output processing, etc.). The ATLFASTII simulation speed is therefore dominated by the speed of Geant4 detector simulation. Any improvement in speed or optimization of Geant4 will result in faster execution of both setups, full Geant4 detector simulation and ATLFASTII detector simulation.

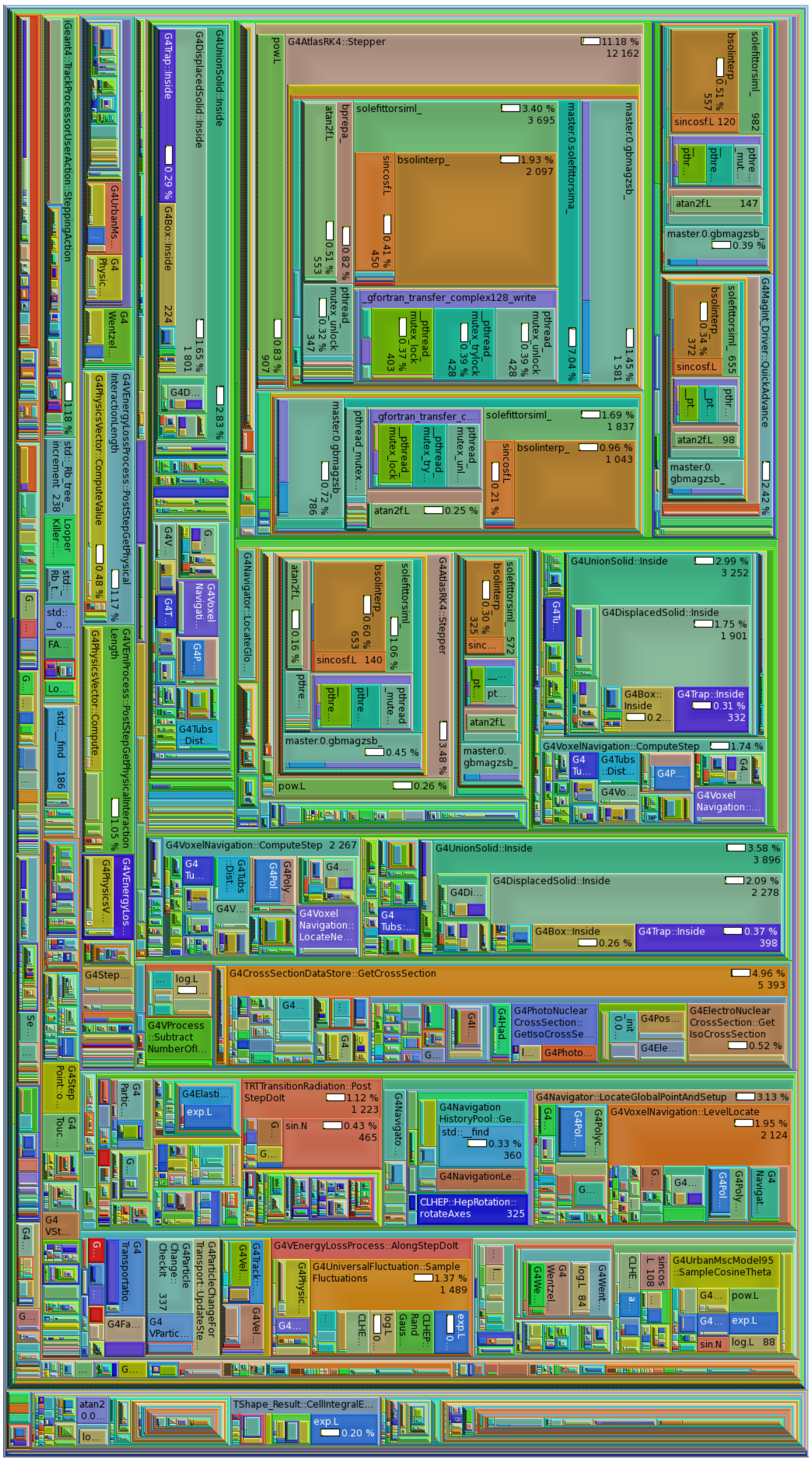


Figure 10.4: CPU profile for ATLFASTII detector simulation of 60 $t\bar{t}$ events with the ISF in the ATLAS offline software release 17.7.5.4 The profile is recorded with the GPerftools CPU profiler and visualized with KCachegrind.

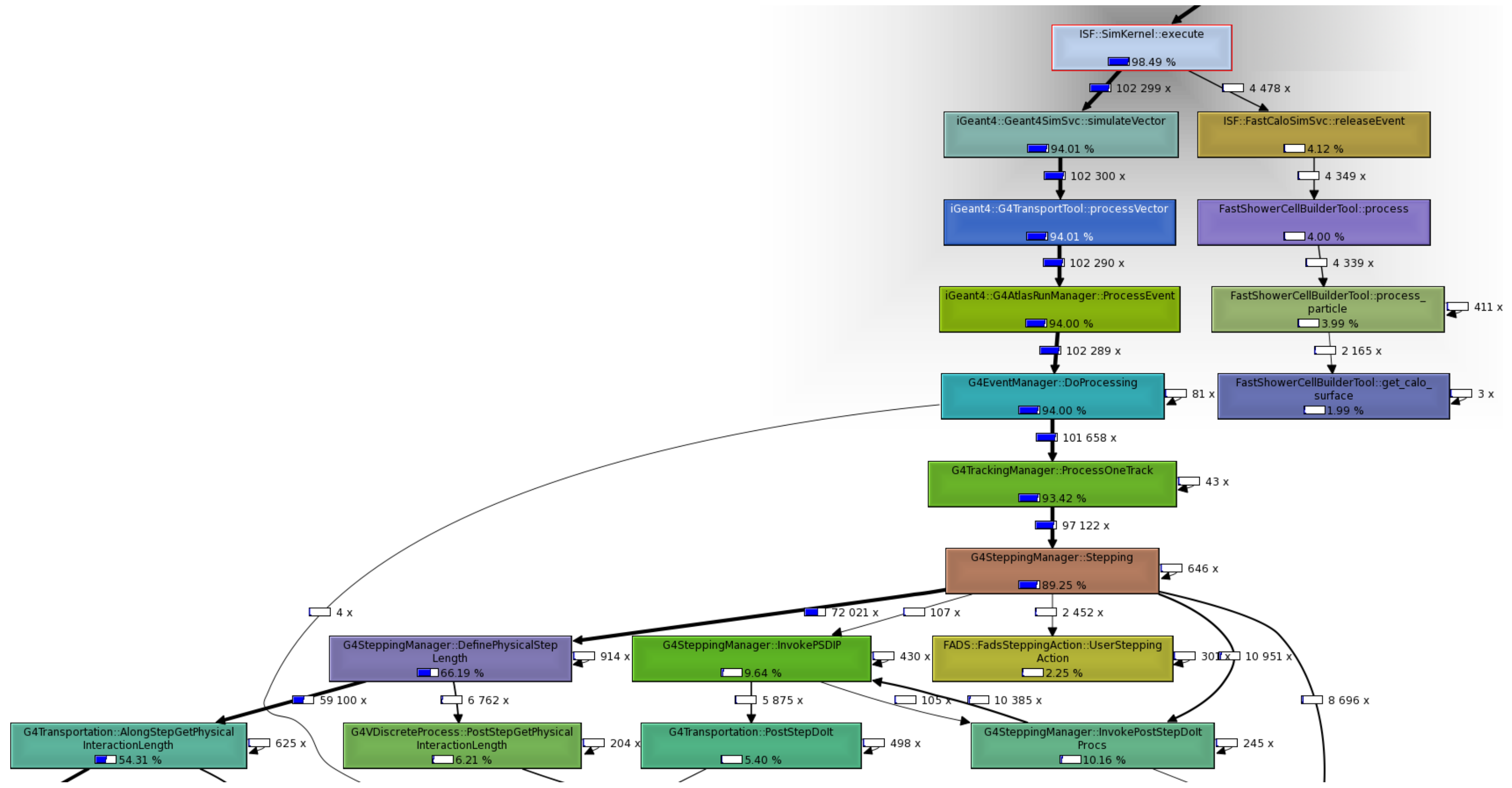

ISF::SimKernel::execute
98.49 %
102 299 x
4 478 x
iGeant4::Geant4SimSvc::simulateVector
94.01 %
ISF::FastCaloSimSvc::releaseEvent
4.12 %
102 300 x
4 349 x
iGeant4::G4TransportTool::processVector
94.01 %
FastShowerCellBuilderTool::process
4.00 %
102 290 x
4 339 x
iGeant4::G4AtlasRunManager::ProcessEvent
94.00 %
FastShowerCellBuilderTool::process_ particle
3.99 %
411 x
102 289 x
2 165 x
G4EventManager::DoProcessing
94.00 %
81 x
FastShowerCellBuilderTool::get_calo_ surface
1.99 %
3 x
101 658 x
G4TrackingManager::ProcessOneTrack
93.42 %
43 x
97 122 x
G4SteppingManager::Stepping
89.25 %
646 x
4 x
72 021 x
107 x
2 452 x
G4SteppingManager::DefinePhysicalStep Length
66.19 %
914 x
G4SteppingManager::InvokePSDIP
9.64 %
430 x
FADS::FadsSteppingAction::UserStepping Action
2.25 %
301 x
10 951 x
59 100 x
6 762 x
5 875 x
105 x
10 385 x
8 696 x
G4Transportation::AlongStepGetPhysical InteractionLength
54.31 %
625 x
G4VDiscreteProcess::PostStepGetPhysical InteractionLength
6.21 %
204 x
G4Transportation::PostStepDoIt
5.40 %
498 x
G4SteppingManager::InvokePostStepDoIt Procs
10.16 %
245 x

Figure 10.5: CPU profile and call tree for ATLFASTII detector simulation of 60 $t\bar{t}$ events with the ISF in the ATLAS offline software release `17.7.5.4`. The profile is recorded with the GPerftools CPU profiler and visualized with KCachegrind. The percentages shown are the CPU spent inside each individual function with respect to the CPU time spent in the entire Athena algorithm execute step. Functions with less than two percent relative CPU cost are not visualized. The profile reveals that about 94% of the CPU time in the ATLFASTII ISF setup is spent inside Geant4 and about 4% is spent inside FastCaloSim. About 98.5% of the CPU time in the Athena algorithm execute step are spent inside the ISF SimulationKernel, the remaining 1.5% are spent inside the Athena framework, e.g. input/output processing.

10.3.3 ISF ATLFASTIIF Detector Simulation

With the implementation of the ISF, the ATLFASTIIF simulation setup can be configured through the same framework as the legacy ATLFASTII and full Geant4 simulation flavours. As the simulation output of ATLFASTIIF detector simulation is compatible with the existing Monte Carlo production chain a straight-forward integration of ATLFASTIIF into the chain is guaranteed. Hence, the production of the ATLFASTIIF simulation samples is expected to grow throughout the following years. Given this situation, a study of ATLFASTIIF computing performance and computing efficiency is of great relevance.

With the fast algorithmic approaches implemented in the ATLFASTIIF setup (Fatras and FastCaloSim), this detector simulation is significantly faster than full Geant4 detector simulation and even ATLFASTII (Section 10.2). Thus, further optimization of the computing efficiency of ATLFASTIIF is less critical than for the other two setups.

Figure 10.6 illustrates the CPU profile of ATLFASTIIF simulation within the ISF, visualized with KCachegrind. The most noticeable hot spots in the profile are:

- Similar to the two previously discussed simulation setups, the functions `master.0.gbmagzsb_` and `bsolinterp_` for the computation of the magnetic field inside the ATLAS volume, appear as significant hot spots in the CPU profile. Lower particle multiplicities are expected for Fatras inner detector simulation compared to Geant4 inner detector simulation. This results in a lower number of calls to the magnetic field routing in Fatras simulation. However, due to the much faster simulation speed of ATLFASTIIF, the relative costs of the magnetic field routines is about twice that in the full Geant4 setup.

- The `exp.A`, `log.A`, `atan2`, `atan2f.A` and other functions from the mathematics library account for a significant fraction of the CPU time costs in the ATLFASTIIF simulation setup. The Intel® Math Library `libimf` [90, 93] is used for the computation of mathematics functions in the ATLAS detector simulation as it provides highly optimized function implementations. The library is dynamically loaded at the beginning of the Athena process. Its implementation is maintained by Intel®, therefore it is not accessible for optimization by the ATLAS collaboration. However, the relative cost of these functions may be lowered, by decreasing the number of calls to them in the ATLAS codebase.

- The `Trk::RungeKuttaPropagator::rungeKuttaStep` and `Trk::STEP_Propagator::propagateWithJacobian` are implementations of the Runge-Kutta [94–96] method to compute particle trajectories through the ATLAS detector volume.

- The `Trk::CylinderSurface::straightLineDistanceEstimate` and `Trk::DiscSurface::straightLineDistanceEstimate` functions are used for navigation through the ATLAS detector description in the Fatras simulation (called TrackingGeometry [79]).

- `TShape_Result::CellIntegralEtaPhi`, `TShape_Result::f_2DSpline` and others (class names starting with `T`) are functions implemented by the ROOT data analysis framework [71]. In the ATLFASTIIF detector simulation setup, they are called by the FastCaloSim simulator. The function implementation is maintained by the ROOT collaboration. The relative cost of these functions within the ATLFASTIIF setup may be lowered by decreasing the number of calls from within FastCaloSim.

The CPU profile in Figure 10.7 reveals that about 56% of the CPU time is spent inside Fatras and about 37% is spent inside FastCaloSim routines. The CPU time measurement is relative to the entire CPU time spent inside the Athena algorithm execute step. Due to the high execution speed of ATLFASTIIF, the relative costs of the Athena framework (i.e. input and output processing) are higher in this simulation setup than in other setups. This is noticeable in the CPU profile as the relative CPU time spent inside the ISF SimulationKernel is about 94%, which is less than for the other ISF flavours presented in the previous sections. Consequently, about 6% of the CPU time is spent inside core components of the Athena framework.

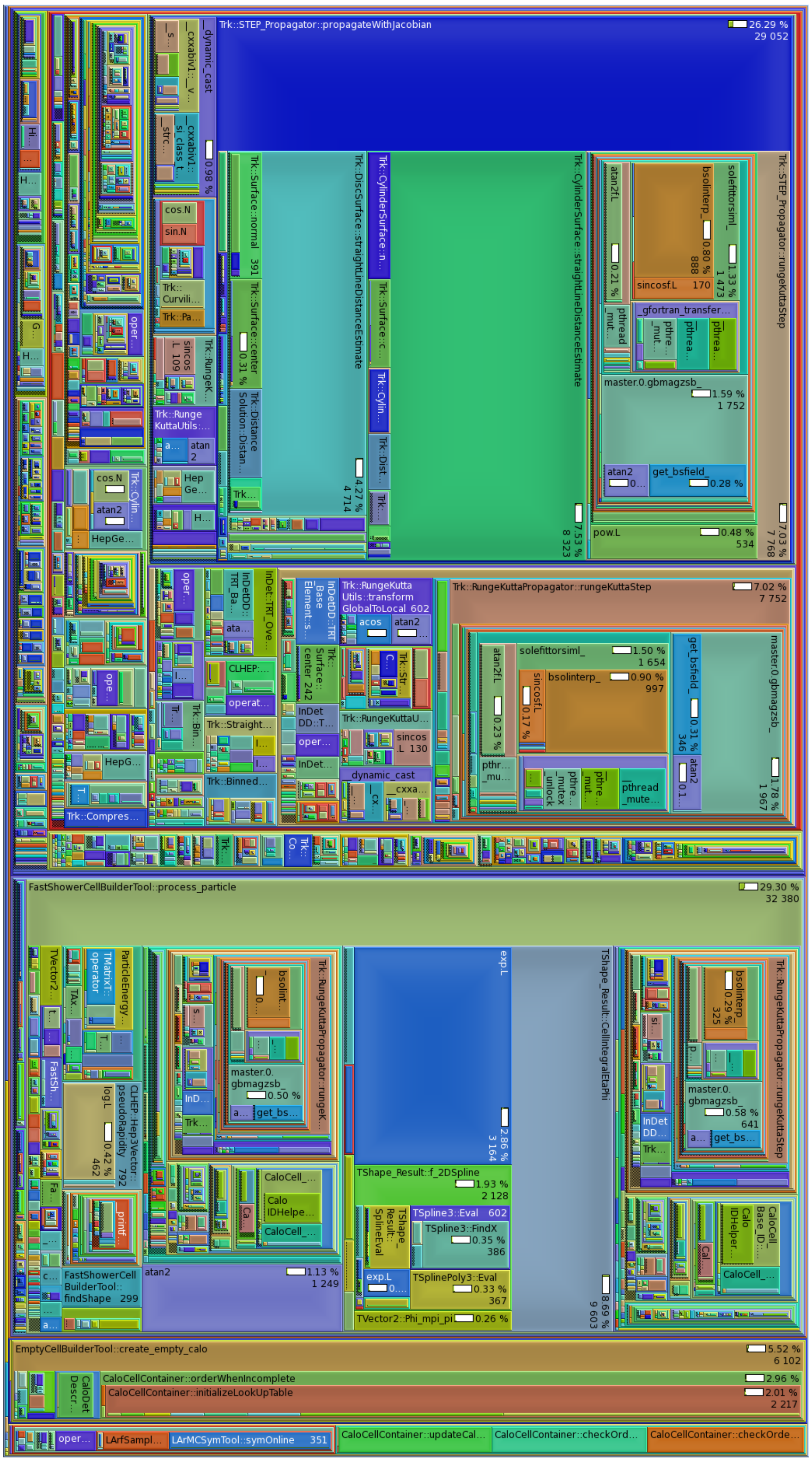


Figure 10.6: CPU profile for ATLFASTIIF detector simulation of 1100 $t\bar{t}$ events with the ISF in ATLAS offline software release 17.7.5.4 The profile is recorded with the GPerftools CPU profiler and visualized with KCachegrind.

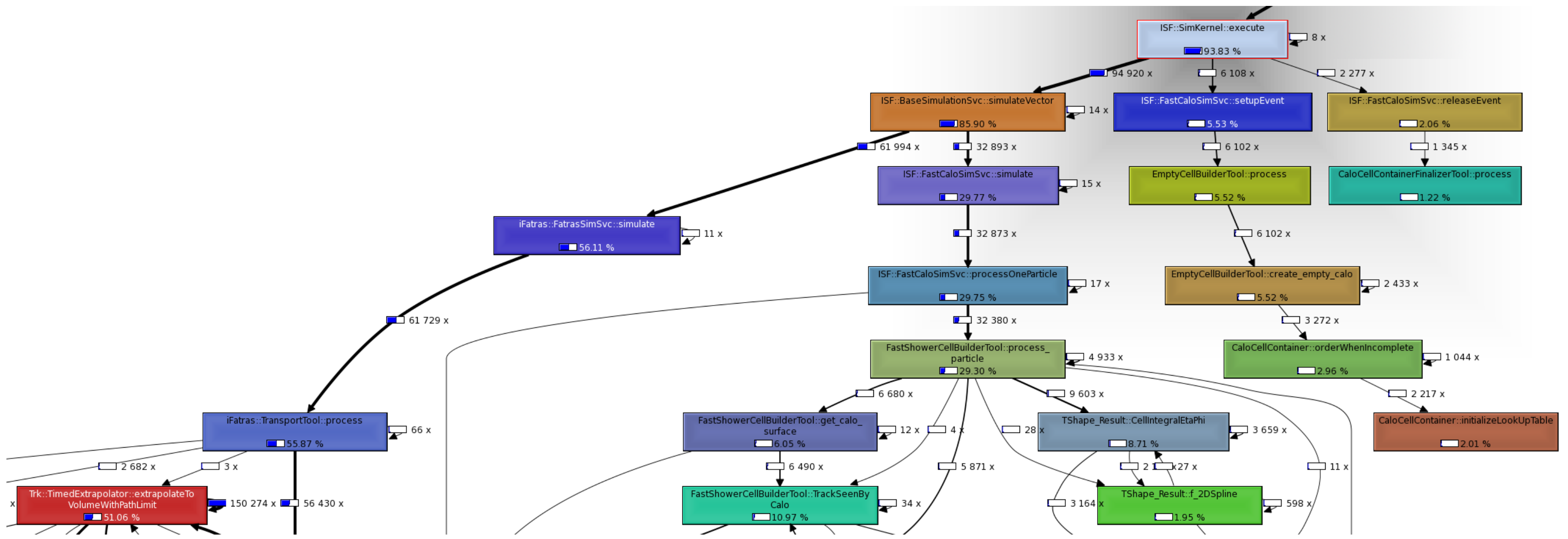

ISF::SimKernel::execute 93.83 %
8 x
94 920 x
6 108 x
2 277 x
ISF::BaseSimulationSvc::simulateVector 85.90 %
14 x
ISF::FastCaloSimSvc::setupEvent 5.53 %
ISF::FastCaloSimSvc::releaseEvent 2.06 %
61 994 x
32 893 x
6 102 x
1 345 x
ISF::FastCaloSimSvc::simulate 29.77 %
15 x
EmptyCellBuilderTool::process 5.52 %
CaloCellContainerFinalizerTool::process 1.22 %
iFatras::FatrasSimSvc::simulate 56.11 %
11 x
32 873 x
6 102 x
ISF::FastCaloSimSvc::processOneParticle 29.75 %
17 x
EmptyCellBuilderTool::create_empty_calo 5.52 %
2 433 x
61 729 x
32 380 x
3 272 x
FastShowerCellBuilderTool::process_particle 29.30 %
4 933 x
CaloCellContainer::orderWhenIncomplete 2.96 %
1 044 x
6 680 x
9 603 x
2 217 x
iFatras::TransportTool::process 55.87 %
66 x
FastShowerCellBuilderTool::get_calo_surface 6.05 %
12 x
4 x
28 x
TShape_Result::CellIntegralEtaPhi 8.71 %
3 659 x
CaloCellContainer::initializeLookUpTable 2.01 %
2 682 x
3 x
6 490 x
5 871 x
2 x
27 x
11 x
Trk::TimedExtrapolator::extrapolateToVolumeWithPathLimit 51.06 %
150 274 x
56 430 x
FastShowerCellBuilderTool::TrackSeenByCalo 10.97 %
34 x
3 164 x
TShape_Result::f_2DSpline 1.95 %
598 x

Figure 10.7: CPU profile and call tree for ATLFASTIIF simulation of 1100 $t\bar{t}$ events with the ISF in the ATLAS offline software release `17.7.5.4`. The CPU profile was generated using GPerftools. The percentage values shown in each individual function are the relative CPU time spent inside this function with respect to the CPU time spent inside the entire Athena algorithm execute step. About 56% of the CPU time is spent inside Fatras and 37% is spent inside FastCaloSim routines. Due to the high simulation speed of ATLFASTIIF, only about 94% of the CPU time is spent inside the ISF SimulationKernel, the remaining 6% are consumed by core components of the Athena framework.

10.3.4 ISF Core Components

One of the main objectives during the development of the ISF was to minimize the computational cost of all core ISF components. Hence, minimizing the computational cost of the entire framework. CPU profiling of various ISF simulation setups shows that each individual component of the ISF use less than 1% of the total CPU time spent in the Athena algorithm execution step (Table 10.3).

The computational cost of the entire ISF framework is dominated by three of its components, namely the GeoIDService (Section 7.3.1), the TruthService (Section 7.4) and the ParticleBroker (Section 7.2). The prior two are called very frequently during ISF detector simulation. This results in a measurable CPU time consumption of the GeoIDService and the TruthService, despite their optimization for high execution speed.

Component	full Geant4	ATLFASTII	ATLFASTIIF
GeoIDService	**0.6%**	**0.3%**	$< 0.1\%$
TruthService	$< 0.1\%$	0.1%	**0.5%**
ParticleBroker	$\sim 0\%$	**0.3%**	0.4%

Table 10.3: The CPU time consumption of different core ISF components relative to the total CPU time spent in the Athena algorithm execution stage, for detector simulation of $t\bar{t}$ events in the ATLAS offline software release 17.7.5.4. The dominating components within each simulation flavour are indicated in bold numbers. GPerftools is used to generate the CPU profiles for the individual ISF detector simulation setups. Between 108000 and 118000 stack traces are recorded with each setup in order to minimize the statistical errors of the GPerftools measurement. This corresponds to different numbers of events being profiled in each simulation: full Geant4 (7 events), ATLFASTII (60 events), ATLFASTIIF (1100 events).

Full Geant4 detector simulation flavour: The GeoIDService is the most expensive ISF component with respect to computing time. Figure 10.8 illustrates the call graph and cost of the ISF GeoIDService in the full Geant4 detector simulation setup. The figure shows that the TrackProcessorUserAction (Section 9.1) is the only method calling the GeoIDService. Though the GeoIDService is optimized for high execution speed, the TrackProcessorUserAction is called in every simulation step a particle makes inside the ISF Geant4 simulator. As this typically corresponds to millions of function calls in the full Geant4 detector simulation, even the costs of fast functions will add up throughout the execution of the detector simulation job. The TrackProcessorUserAction uses the GeoIDService to determine whether a particle crossed a boundary between two ATLAS regions or sub-detectors.

The ParticleBroker is not used during the execution of full Geant4 detector simulation. After sending primary particles to Geant4, all particles are kept within this simulator until the end of the detector simulation of the current event.

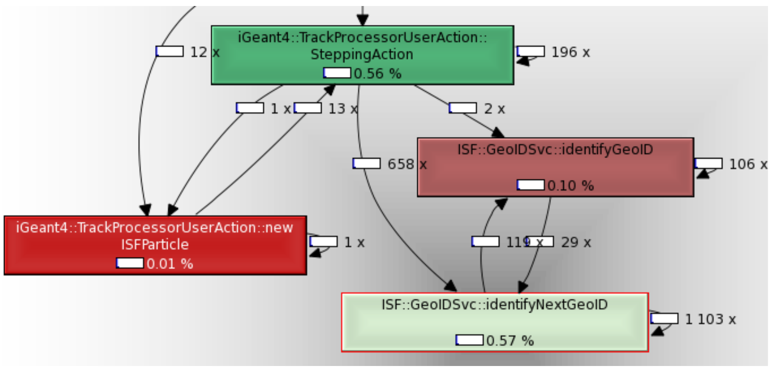

iGeant4::TrackProcessorUserAction::SteppingAction
0.56 %
196 x
12 x
1 x
13 x
2 x
ISF::GeoIDSvc::identifyGeoID
0.10 %
106 x
658 x
iGeant4::TrackProcessorUserAction::new ISFParticle
0.01 %
1 x
119 x
29 x
ISF::GeoIDSvc::identifyNextGeoID
0.57 %
1 103 x

Figure 10.8: The call graph and CPU cost of the ISF GeoIDService in the full Geant4 detector simulation setup in ATLAS offline software release `17.7.5.4`. The GeoIDService appears in 0.57% of all stack traces recorded by GPerftools during the Athena algorithm execution stage. This corresponds to about 0.57% of the CPU time spent inside the GeoIDSvc. In the Geant4 simulator, the GeoIDService is called by the TrackProcessorUserAction to determine whether a particle crossed a boundary between two ATLAS regions or sub-detectors.

ATLFASTII detector simulation flavour : The GeoIDService is called mainly by the ATLAS Geant4 implementation in order to identify whether particles traversed ATLAS sub-detector boundaries. In contrast to full Geant4 simulation, the ParticleBroker is used more frequently in order to re-route particles at sub-detector boundaries. Hence it's relative costs are greater than in full Geant4 simulation. In absolute numbers, the TruthService is called less frequently per event in the ATLFASTII flavour compared to full Geant4. However, due to the higher execution speed of ATLFASTII, the cost of the TruthService increase relative to the CPU time spent inside the entire Athena execution step.

ATLFASTIIF detector simulation flavour : Given the very high execution speed of ATLFASTIIF, individual components of the ISF are expected to show greater relative CPU utilization than in other flavours. The TruthService is called for each inelastic interaction or decay computed by Fatras. All secondary particles generated by Fatras will be returned to the ParticleBroker, in addition to all particles traversing sub-detector boundaries. Hence, the ParticleBroker contributes to the overall simulation time of ATLFASTIIF. The GeoIDService is called to determine whether particles are entered into respective EntryLayer collections (Section 7.4.1). As this operation is of low complexity, the GeoIDService contributes only marginally to the CPU time requirements of the ATLFASTIIF flavour.

Chapter 11

Accuracy of ISF Detector Simulation

The Integrated Simulation Framework allows for the first time to combine various simulators for the simulation of any one particle through the ATLAS detector within one framework. This chapter discusses the consequences of this approach in respect of generating a high quality simulation.

Every simulation method introduces systematic errors in physics analyses due to imperfections in the modelling of the simulated physics processes. A common method for estimating and minimizing these systematic errors is to derive and apply corresponding calibrations. The calibrations aim at minimizing the discrepancies between recorded and simulated data. The most commonly used simulation toolkit for detector simulation in high energy physics experiments is Geant4 (Section 4.3.1). It is considered the most accurate simulator available for ATLAS detector simulation. Hence, calibrations for physics analyses are generally derived for Geant4-based simulation of the ATLAS detector. The aim of combined full and fast detector simulation methods is therefore to reproduce the results obtained by Geant4 simulation. This allows to apply the same (or very similar) calibrations without significantly increasing the systematic errors.

The high accuracy of Geant4 simulation is only guaranteed if it is the sole simulator processing particles through the ATLAS detector for a given simulation sample. If Geant4 were to be combined with any other simulator, the overall accuracy is a combination of the individual accuracies of both simulators. The individual accuracies can not be considered independently in this setup, as one simulator's output may serve as the other simulator's input. In general, the overall simulation accuracy will decrease due to combining multiple detector simulators. The accuracy can be increased, however, if simulators account for the fact that particles are shared with other simulators and actively correct for this. This can be implemented in the form of tuning parameters inside the respective simulator. Separate sets of tuning parameters may be used for each particular simulator combination. In some cases, a sample-specific tuning approach may be employed in order to achieve an optimal agreement of a combined simulation setup with Geant4.

11.1 ATLFASTII

Two FastCaloSim modes are available in the Integrated Simulation Framework. The first one is the traditional or "legacy mode" (see Section 9.3.1 for implementation). It resembles the method with which the parametrization was obtained and hence achieves the most accurate simulation results with respect to Geant4 simulation. In this mode, FastCaloSim excludes the use of other calorimeter simulators within the same detector simulation setup. The second mode is the "ISF mode" (see Section 9.3.2 for implementation), where FastCaloSim can be used in combination with other calorimeter simulators.

The parametrization of FastCaloSim was generated using Geant4 detector simulation of single photon and pion events [80]. The particles were simulated with discrete energies and their corresponding reconstructed calorimeter response was computed. The `HepMC` Monte Carlo truth representation generated during the simulation of the inner detector (ID), plays an important role in the FastCaloSim parametrization. The parametrization describes the relationship between the particle states in the MC truth representation and the reconstructed calorimeter response (including various parameters regarding the shape and position of the energy distribution).

The effects on the calorimeter measurements due to low energy particle interactions occurring in the inner detector volume are intrinsically described by the parametrization, in order to achieve a highly accurate calorimeter simulation. Particles which are generated in decays, radiation or particle-matter interactions inside the ID volume and that have energies of less than 500 MeV, are traced back to common parent particles with energies above this threshold via the MC truth representation. Parent particles are only considered if they do not have any child particle with an energy above the 500 MeV threshold. The parametrization describes the relationship between these parent particles and the respective calorimeter measurements (Figure 11.1). The parametrization also describes the calorimeter response of particles with energies lower than 500 MeV if they were to emerge from the interaction point, however, a less accurate simulation result is expected in this low energy regime. The creation points of the parent particles are within the inner detector volume and therefore do not describe the actual particle states at the point of entering the calorimeter volume. The processes occurring to the parent particles (and their child particles) when travelling from within the ID volume towards the boundary with the calorimeter volume are intrinsically considered in the FastCaloSim parametrization. This includes energy loss processes, decays, radiation and any form of particle-matter interaction. The net effect of these processes is that they lower the total energy which reaches the boundary between the ID and the calorimeter volume.

In the discussion below, the impact of this parametrization to the accuracy of both FastCaloSim modes is compared to Geant4 detector simulation.

11.1.1 Legacy Mode

In the legacy mode of FastCaloSim or in the traditional ATLFASTII simulation flavour, the MC truth representation of the inner detector simulation is used to input the parameterized calorimeter simulation. The same method of reducing the MC truth rep-

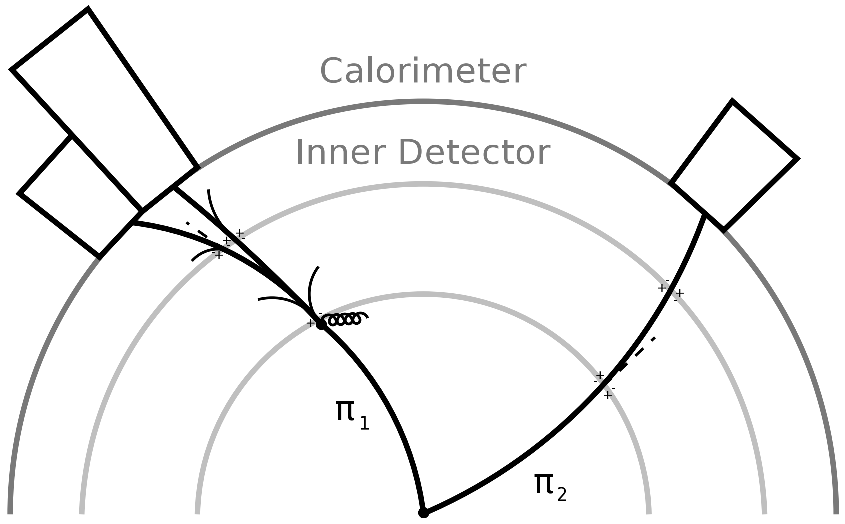


Figure 11.1: The FastCaloSim parametrization describes the relationship between particles in the Monte Carlo truth representation and the resulting energy response in the calorimeter. The parametrization is based on particles which are above an energy threshold of 500 MeV and whose child particles are all below this threshold. Interaction processes occurring to this particle and its child particles while traversing the ATLAS inner detector (ID) are intrinsically described by this parametrization. In this figure both particles, π_1 and π_2 have the same energy and they fulfill the above condition. If provided with a charged pion input particle that has the same energy as π_1 or π_2, FastCaloSim will intrinsically account for the processes that occur to the particle while travelling through the ID towards the calorimeter.

resentation to parent particles with energies just above 500 MeV is applied in order to compute their calorimeter response. Effectively, the FastCaloSim input is reduced to the same level of detail that was used to generate the parametrization in the first place. Consequently, the best agreement with Geant4 simulation is achieved in this mode of FastCaloSim. The legacy mode of FastCaloSim is therefore used to form the ATLFASTII simulation flavour within the ISF.

FastCaloSim calibrations with respect to recorded data were derived. This allows FastCaloSim to reproduce the recorded data more accurately than Geant4 in some cases (Figure 4.7).

The current FastCaloSim parametrization was generated with ATLAS offline release 14 (in the year 2009). It is based on Geant4 release `4.8` and the ATLAS geometry description at that time. The FastCaloSim parametrization has not changed since then. However, the Geant4 software version that is used by the ATLAS collaboration has changed over the years, as well as the detector description. As a result of continuous improvements being made to the Geant4 simulation toolkit, changes in the simulation output are expected between different software versions. The accuracy with which particles at high energies are simulated is not expected to change drastically, since the simulation results for such particles have been validated for many years. The properties of particles in the low energy spectrum, however, are subject to larger changes between Geant4 versions. The dependency of the accuracy of FastCaloSim to the changing Geant4 output is minimized, due to simulating the effects of the high energy particles (more than 500 MeV) in the inner detector on the calorimeter. This ensures that the input to FastCaloSim will not change significantly when using different software versions of Geant4.

Any changes affecting the processes occurring to the particles when travelling through the ID towards the calorimeter boundary will, however, affect the accuracy of FastCaloSim in comparison to Geant4. For example, if changes to the detector material description were made it would result in a discrepancy between the FastCaloSim output and Geant4. In order to minimize such a discrepancy, a re-parametrization of FastCaloSim is required.

11.1.2 ISF Mode

In the ISF, a new mode of FastCaloSim has been implemented, that allows to use the simulator together with other calorimeter simulators in the same event. This mode is enabled only due to the implementation of the Integrated Simulation Framework, hence it is called the "ISF mode". In this setup, FastCaloSim computes the calorimeter response for the individual particles that are traversing the boundary between the inner detector and the calorimeter region.

The same core FastCaloSim implementation is used to compute the calorimeter response in the ISF mode and in the legacy mode. Since FastCaloSim has been parameterized for the legacy mode, a greater discrepancy between Geant4 and FastCaloSim is obtained in the ISF mode (Figure 11.2).

The discrepancy in the ISF mode is due to a double counting of the energy which is lost due to interaction processes occurring to the particles when traversing the inner detector volume towards the ID/calorimeter boundary. The impact of these processes to the calorimeter response is intrinsically described by the FastCaloSim parametrization (discussion above). In the ISF mode, the impact of these processes has already been computed by the inner detector simulation and this is represented in the particle states at the ID/calorimeter boundary. Hence, if the FastCaloSim parametrization is used to generate a calorimeter response for these particles, the energy which is lost due to these processes is implicitly removed again and the energy in the calorimeter will be underestimated.

Two approaches are feasible to increase the accuracy of the FastCaloSim ISF mode:

- Energy scale factors for the particles at the ID/calorimeter boundary can be introduced, in order to correct for the double counting of the energy which is lost due to interaction processes in the ID volume. This has been applied successfully in the application of the ISF to the W boson mass measurement, covered in Chapter 12 in this thesis.

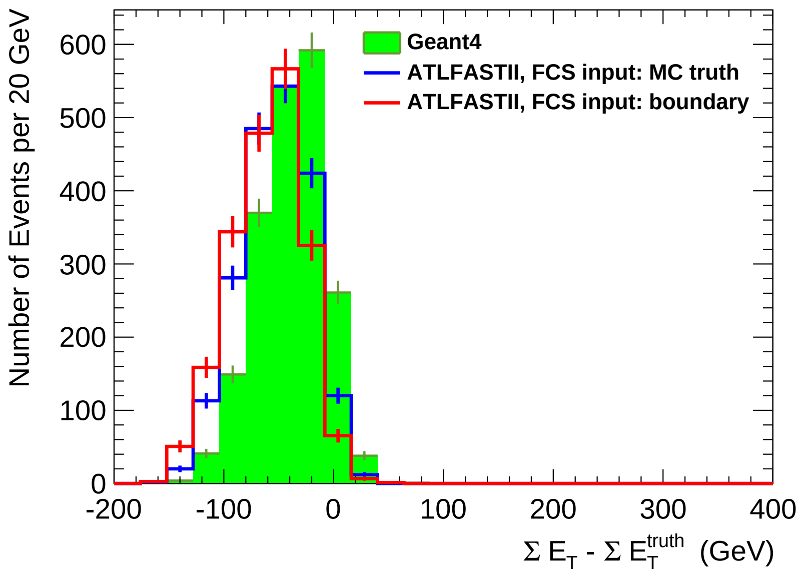


Figure 11.2: The difference between the reconstructed and the MC truth scalar sum of the transverse energy $\sum E_{\mathrm{T}}$ for different ATLFASTII setups compared with Geant4 full detector simulation (green). 2000 $t\bar{t}$ events were simulated in each respective setup with ATLAS offline software release 17.7.4.2. ATLFASTII using FastCaloSim in legacy mode (blue line) is closer to the very accurate Geant4 simulation than ATLFASTII using FastCaloSim in the ISF mode (red line).

- A currently ongoing project is concerned with the re-parametrization of FastCaloSim. The ISF mode will be considered within this new parametrization and hence, an increased accuracy of FastCaloSim with respect to Geant4 is to be expected.

11.2 ATLFASTIIF

The ATLFASTIIF simulation flavour (Section 4.4.2) combines Fatras inner detector simulation with FastCaloSim calorimeter simulation. As mentioned in the previous section, FastCaloSim in the ATLFASTII setup uses the `TruthEvent` Monte Carlo truth collection to compute the calorimeter response, as this provides high simulation accuracy. In the ATLFASTIIF setup, the Monte Carlo truth record reflects the Fatras inner detector simulation output, whereas in the ATLFASTII setup it reflects the Geant4 inner detector simulation output. Fatras, being a fast detector simulation, provides a less accurate description of interaction processes and secondary particles than Geant4 does. Thus, the Monte Carlo truth record generated by Fatras in ATLFASTIIF, will be different from the Monte Carlo truth record generated by Geant4 in ATLFASTII.

The FastCaloSim parametrization was computed based on Geant4 inner detector and calorimeter simulation. Thus, the accuracy of the calorimeter simulation decreases, if FastCaloSim uses the MC truth record created by Fatras (Figure 11.3). This discrepancy can be lowered if FastCaloSim reads the input particles from a well defined geometrical boundary between the inner detector volume and the calorimeter, rather than the `TruthEvent` collection. In other words, the compatibility with full simulation increases if the ISF mode were used in FastCaloSim instead of the legacy mode. A significant difference between ATLFASTII and ATLFASTIIF still remains with this method.

To lower the discrepancy due to combining Fatras with FastCaloSim even further, a FastCaloSim tuning approach specifically for the combination with Fatras is discussed in Section 12.4. In this approach, a set of tuning parameters is used to scale the energy of particles when handed over from Fatras to FastCaloSim at the boundary between the inner detector and the calorimeter regions.

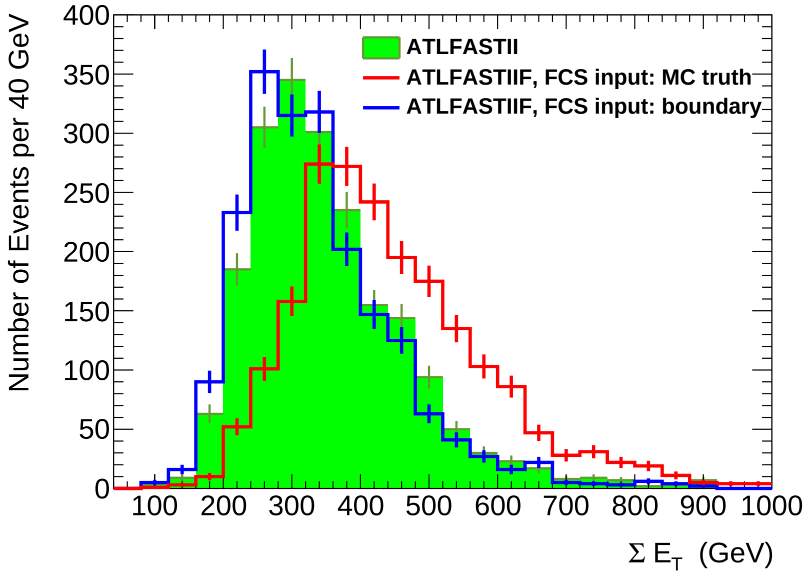


Figure 11.3: The scalar sum of the reconstructed transverse energy $\sum E_{\mathrm{T}}$ of different ATLFASTIIF setups compared with ATLFASTII (green) for 2000 $t\bar{t}$ events simulated in ATLAS offline software release `17.7.4.2`. A significant discrepancy is introduced in ATLFASTIIF relative to ATLFASTII, if the input particles to FastCaloSim are read from the MC truth record, i.e. the legacy mode of FastCaloSim is used (red line). The discrepancy decreases in the ATLFASTIIF setup, if FastCaloSim uses the particles at the geometrical boundary between the inner detector volume and the calorimeter as input, called the ISF mode of FastCaloSim (blue line).

Chapter 12

ISF in the Context of the W Boson Mass Measurement

The measurement of the W boson mass (M_W) allows for a stringent test of the Standard Model (SM) of particle physics. As described in Chapter 2, M_W, the top quark mass (m_t) the Higgs boson mass (M_H) and other experimentally defined parameters can be used to form an overconstrained set of parameters in the electroweak sector of the SM. As the experimental uncertainties of M_W and m_t are the dominating uncertainties in this overconstrained model, any reduction of their uncertainties will enable a more stringent test of the SM.

The overconstrained set of parameters can be used to compute (fit) the W boson mass. This leads to $M_W^{\text{fit}} = 80.358 \pm 0.008$ GeV [38], which differs by more than one standard deviation from the experimentally determined (world average) value of $M_W = 80.385 \pm 0.015$ GeV [17]. The uncertainty of the experimentally determined value is more than twice the size of the error in the fitted W boson mass. Hence, it is of particular interest to measure M_W more accurately in order to determine whether the difference between the computed and the measured W boson mass is caused solely by measurement uncertainties, or whether this indeed points to an inconsistency in the electroweak sector of the SM.

The analysis method applied by the ATLAS collaboration (in the following described in Section 12.1) is expected to achieve a systematic uncertainty of about 7 MeV and a statistical uncertainty of about 2 MeV for the measurement of M_W with 10 fb^{-1} of integrated luminosity collected by the detector. This measurement will contribute significantly to the world average value of the experimentally determined W boson mass and hence, enables a more stringent test of the SM. High-statistics simulation samples are required to study and minimize the experimental uncertainties in the measurement. Specific requirements to the detector simulation (Section 12.2) allow for a dedicated configuration of the ISF to generate the required simulated datasets. The flexibility of the ISF is used to combine highly accurate Geant4 simulation with fast Fatras and FastCaloSim simulation within individual events (Section 12.3). In order to improve the accuracy of the fast detector simulation methods, a dedicated FastCaloSim tuning approach was developed and applied (Section 12.4). A significant speed-up in simulation time and a good physics agreement between ISF fast simulation and full Geant4 simulation is obtained with this approach (Section 12.5).

12.1 Analysis Method

The analysis method and the detector calibration methods discussed in this section have not yet been fully carried out on recorded data. The calibration methods are currently in the process of being refined and implemented for their use with data collected by the ATLAS experiment during the LHC Run 1. The finalization of the detector calibration is a requirement to conducting the W boson mass analysis method described below. The images presented in this section are based on simulated data (and simulated "pseudo data") only, and serve mainly an illustrative purpose.

The final states of W boson decays which can be measured most accurately by the ATLAS experiment are $e\nu_e$ and $\mu\nu_\mu$. In these leptonic decay channels, a significant fraction of the energy of the W boson is contained in the neutrino, making it particularly challenging from an experimental aspect. Both decay channels are studied and Monte Carlo (detector) simulation is used to study and minimize the systematic uncertainty of each measurement [31, 97, 98].

Specific event selection criteria (Table 12.1) are applied to minimize the amount of background processes (such as $W \to \tau\nu$) in the analysis dataset and to increase accuracy of the detector measurements. After applying these selection criteria and considering the corresponding trigger efficiencies, about 47×10^6 $W \to e\nu_e$ decays and 84×10^6 $W \to \mu\nu_\mu$ decays are expected to be measured per 10 fb^{-1} of integrated luminosity collected by the ATLAS detector at the design energy of the LHC. This corresponds to a total efficiency of about 20% for the $W \to e\nu_e$ channel and about 40% for the $W \to \mu\nu_\mu$ channel. The expected statistical precision of the W boson mass measurement is about 2 MeV per channel (with 10 fb^{-1} of integrated luminosity).

M_W can not be reconstructed directly through ATLAS detector measurements, as each of the relevant W boson final states contains one neutrino. The observables for the determination of M_W can be measured in the transverse plane of the ATLAS detector, as the conservation of momentum allows to indirectly determine the transverse momentum of the neutrino through the measurement of the hadronic recoil (Section 12.1.1). Template distributions of the charged lepton transverse momentum and the transverse mass of the W boson are generated through detector simulation with different values of M_W (Section 12.1.2). The template distributions are compared to the corresponding distributions recorded by the detector, from which the measured value of M_W is derived. About 10^9 simulated $W \to e\nu_e$ events were required for the generation of the corresponding template distributions in a comparable measurement of M_W with the D0 detector [7]. About 30 times more events are expected to be measured by the ATLAS detector in this decay channel (given 10 fb^{-1} of integrated luminosity). Together with the additionally studied $W \to \mu\nu_\mu$ channel, the need for simulation statistics in the order of multiple billions of simulated events becomes apparent. The generation of the

Channel	$W \to e\nu_e$	$W \to \mu\nu_\mu$	$Z \to ee$	$Z \to \mu\mu$
Reconstructed Leptons	$p_{\mathrm{T}}^{\ell} > 20$ GeV, $\lvert\eta_\ell\rvert < 2.5$			
Crack region removed	$1.30 < \lvert\eta_\ell\rvert < 1.60$	–	$1.30 < \lvert\eta_\ell\rvert < 1.60$	–
Missing Energy	$\not{E}_T > 20$ GeV		–	–
Events in 10 fb^{-1} (10^6)	47	84	2.1	6.7

Table 12.1: The event selection criteria for the W boson decay channels used for the M_W measurement and the Z decay channels for the respective detector calibration. Electron measurements in regions of the calorimeter with degraded resolution (called "cracks") are excluded from the measurement and calibration. The expected selected number of events per 10 fb^{-1} of integrated luminosity at the design energy of the LHC are illustrated. The table is taken from [97], however, slightly updated values are to be published and are currently available to the ATLAS collaboration in an internal note [99].

required simulation samples with full Geant4 simulation does not seem feasible, as the sample size is of the same order of magnitude as entire ATLAS Monte Carlo production campaigns (Section 1.1). Hence, a fast detector simulation approach is developed.

Detector calibration methods are applied using $Z \to ee$ and $Z \to \mu\mu$ events (Section 12.1.3) in order to minimize the experimental uncertainties of the W boson mass measurement. Extensive detector simulation is carried out in order to facilitate the computation of corresponding calibration terms. It is estimated that about four times more simulated events than recorded (and selected) $Z \to ee$ and $Z \to \mu\mu$ events are required to determine the calibration. This leads to simulation sample sizes in the order of tens of million events for 10 fb^{-1} of integrated luminosity (see Table 12.1). Even with the detector calibration applied in the measurement, the precision of the M_W measurement is dominated by experimental uncertainties (Table 12.2).

A total systematic uncertainty of about 7 MeV is expected for the measurement of M_W per decay channel (with 10 fb^{-1} of integrated luminosity).

12.1.1 W and Z Boson Observables in the Transverse Plane

The conservation of momentum in the transverse plane allows to indirectly measure the transverse momentum of the undetected neutrino and hence allows do derive observables which are crucial for a precise measurement of M_W (discussion below in Section 12.1.2). This section covers the relationships between various observables in the transverse plane (illustrated in Figure 12.1) for $W \to \ell\nu$ and $Z \to \ell\ell$ decays (where ℓ denotes a charged lepton, either e or μ). The latter is particularly important for detector calibration (Section 12.1.3).

The production of W and Z bosons in the LHC is dominated by quark-antiquark annihilation processes. Higher order processes may radiate gluons or quarks which recoil

Source	Effect	$\sigma(M_W)(p_\mathrm{T}^\ell)$ (MeV)	$\sigma(M_W)(m_\mathrm{T}^W)$ (MeV)
Production Model	W width	0.5	1.3
	y^W distribution	1	1
	p_T^W distribution	3	1
	QED radiation	<1	<1
Lepton measurement	Scale, linearity & resolution	4	4
	Efficiency	4.5 (e), <1 (μ)	4.5 (e), <1 (μ)
Recoil measurement	Scale, linearity & resolution	–	5
Backgrounds	$W \to \tau\nu$	2	1.5
	$Z \to \ell(\ell)$	0.3	0.2
	$Z \to \tau\tau$	0.1	0.1
	Jet events	0.5	0.4
Total		~ 7 (e), 6 (μ)	~ 8 (e), 7 (μ)

Table 12.2: Estimated systematic uncertainties affecting the measurement of the W boson mass (M_W) for 10 fb^{-1} of integrated luminosity recorded by the ATLAS detector. The uncertainties for determining M_W through fitting the lepton transverse momentum distribution ($\sigma(M_W)(p_\mathrm{T}^\ell)$) and through fitting the transverse W boson mass distribution ($\sigma(M_W)(m_\mathrm{T}^W)$) are listed individually. The impact of the detector calibration on the lepton and recoil measurements is considered in this table, however, the total systematic uncertainty is still dominated by the experimental uncertainties of these two observables (source: [39, 97]).

against the produced boson and result in a non-zero boson momentum [100]. Hence, the W and Z bosons studied in this physics analysis generally have non-zero transverse momenta ($\vec{p}_\mathrm{T}^{\,W}$ and $\vec{p}_\mathrm{T}^{\,Z}$) [100]. The transverse momentum of the Z boson can be computed through the vector sum of the reconstructed transverse momenta of the two leptons ($\vec{p}_\mathrm{T}^{\,\ell^+}$ and $\vec{p}_\mathrm{T}^{\,\ell^-}$):

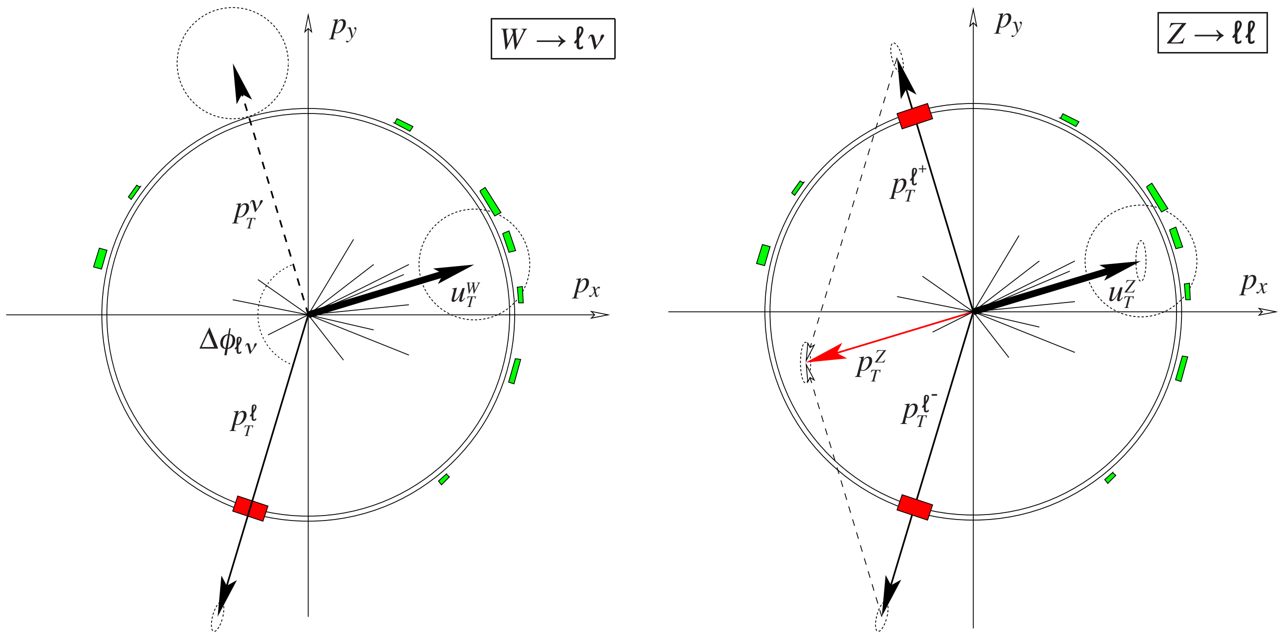
$$\vec{p}_\mathrm{T}^{\,Z} = \vec{p}_\mathrm{T}^{\,\ell^+} + \vec{p}_\mathrm{T}^{\,\ell^-} \; . \tag{12.1}$$

The transverse momentum of the W boson is defined through the vector sum of the reconstructed charged lepton transverse momentum ($\vec{p}_\mathrm{T}^{\,\ell}$) and the transverse momentum of the neutrino ($\vec{p}_\mathrm{T}^{\,\nu}$), where the latter can be indirectly measured as the missing transverse energy ($\vec{\not{E}}_T$):

$$\vec{p}_\mathrm{T}^{\,W} = \vec{p}_\mathrm{T}^{\,\ell} + \vec{p}_\mathrm{T}^{\,\nu} = \vec{p}_\mathrm{T}^{\,\ell} + \vec{\not{E}}_T \; . \tag{12.2}$$

The conservation of momentum in the transverse plane allows for the definition of the hadronic recoil ($\vec{u}_\mathrm{T}^{\,Z}$ and $\vec{u}_\mathrm{T}^{\,W}$), which balances the transverse momentum of the respective boson [7, 100, 101]:

$$0 = \vec{u}_\mathrm{T}^{\,Z/W} + \vec{p}_\mathrm{T}^{\,Z/W} \; . \tag{12.3}$$



(a) (b)

Figure 12.1: (a) $W \to \ell\nu$ and (b) $Z \to \ell\ell$ decays viewed in the transverse plane of the ATLAS detector. The transverse component of the hadronic recoil ($\vec{u}_{\mathrm{T}}{}^{W/Z}$) balances the transverse of momentum of the respective W or Z boson ($\vec{p}_{\mathrm{T}}{}^{Z/W}$) due to the conservation of momentum in the transverse plane. $\vec{u}_{\mathrm{T}}{}^{W}$ is used to estimate the transverse momentum of the undetected neutrino ($\vec{p}_{\mathrm{T}}{}^{\nu}$). The relative azimuthal angle between the lepton and the neutrino ($\Delta\phi_{\ell\nu} = \phi_\ell - \phi_\nu$) is used to compute the transverse mass of the W boson (m_{T}^{W}). The sizes of the dotted ellipses represent the expected resolution of the respective reconstructed objects. A high resolution measurement of $\vec{u}_{\mathrm{T}}{}^{Z}$ is obtained through the precise measurement of the two charged leptons, a lower resolution indirect measurement is obtained through the energy depositions in the calorimeter (green) (image: [39]).

The hadronic recoil is the vector sum of all transverse energy which is not associated with the reconstructed lepton(s) ($\vec{E}_{\mathrm{T}}^{\not\ell}$) [100, 101]:

$$\vec{u}_{\mathrm{T}}{}^{Z/W} = \sum \vec{E}_{\mathrm{T}}^{\not\ell} \; . \tag{12.4}$$

It is measured by computing the vector sum of all transverse energy measurements, excluding the energy measurements which are within a cone (of size ΔR[1] in η-ϕ space) around the respective reconstructed charged lepton(s) (ℓ)

$$\vec{u}_{\mathrm{T}}{}^{Z/W} = \sum_{\substack{\text{outside} \\ \ell \text{ cones}}} \vec{E}_{\mathrm{T}} =: \vec{E}_{\mathrm{T}}{}^{\mathrm{hr}} \; . \tag{12.5}$$

[1] $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$

For the studies presented in this chapter, a cone size of $\Delta R = 0.2$ was chosen.

Combining Equations 12.1 and 12.3 with Equation 12.5, it follows for the Z boson

$$\vec{E}_{\mathrm{T}}^{\mathrm{hr}} = -\left(\vec{p}_{\mathrm{T}}^{\,\ell^+} + \vec{p}_{\mathrm{T}}^{\,\ell^-}\right) \ , \tag{12.6}$$

and for the W boson by combining Equations 12.2 and 12.3 with Equation 12.5

$$\vec{p}_{\mathrm{T}}^{\,\nu} = -\left(\vec{p}_{\mathrm{T}}^{\,\ell} + \vec{E}_{\mathrm{T}}^{\mathrm{hr}}\right) \ . \tag{12.7}$$

The indirect measurement of $\vec{p}_{\mathrm{T}}^{\,\nu}$ (through the relationship given in Equation 12.7) is of particular importance for the determination of the transverse W boson mass which is used to measure M_W (discussed below in Section 12.1.2).

The relationship in Equation 12.6 enables the calibration of the hadronic recoil through $Z \to \ell\ell$ decays (discussed below in Section 12.1.3), which consequently enhances the accuracy of the $\vec{p}_{\mathrm{T}}^{\,\nu}$ measurement.

12.1.2 M_W Measurement

Two observables which are sensitive to M_W are measured, the first one being the reconstructed transverse momentum of the charged lepton (p_{T}^{ℓ}). The second observable is the transverse W boson mass (m_{T}^{W}), which is computed from p_{T}^{ℓ} and the transverse momentum of the neutrino (p_{T}^{ν}), according to

$$m_{\mathrm{T}}^{W} = \sqrt{2\ p_{\mathrm{T}}^{\ell}\ p_{\mathrm{T}}^{\nu}\ \left(1 - \cos\left(\phi_\ell - \phi_\nu\right)\right)} \ , \tag{12.8}$$

where ϕ_ℓ and ϕ_ν are the measured azimuthal angles of the lepton momentum and the neutrino momentum in the ATLAS coordinate system, respectively. The event selection criteria maximize the chance of exactly one neutrino being present in each (simulated or recorded) event. Hence, the measured missing transverse energy is an indirect measurement of the transverse momentum of the neutrino.

Figure 12.2 illustrates the shape of the p_{T}^{ℓ} and m_{T}^{W} distributions after fast detector simulation and with the event selection criteria applied.

MC event simulation and detector simulation are used to generate template distributions of m_{T}^{W} and p_{T}^{ℓ} for different values M_W. The detector simulation output is reconstructed and the same event selection criteria are applied as in the analysis of the recorded data. A least squares parameter fit [102, 103] is carried out in order to determine the optimal value of M_W. χ^2 values are computed to quantify the agreement between each individual template distribution and the corresponding distribution measured in the ATLAS detector. The particular shape of both distributions (also referred to as "Jacobian peak" or "Jacobian edge") results in a high sensitivity of the χ^2 value to the level of agreement between the compared histograms. The effect of the (simulated) ATLAS detector and the reconstruction algorithms to the p_{T}^{ℓ} distribution is illustrated in Figure 12.3. Due to the altered shape of the distribution, in particular the altered slope of the distribution in the vicinity of the peak, the detector simulation plays a crucial role in generating accurate templates for comparison with recorded data. Finally, M_W

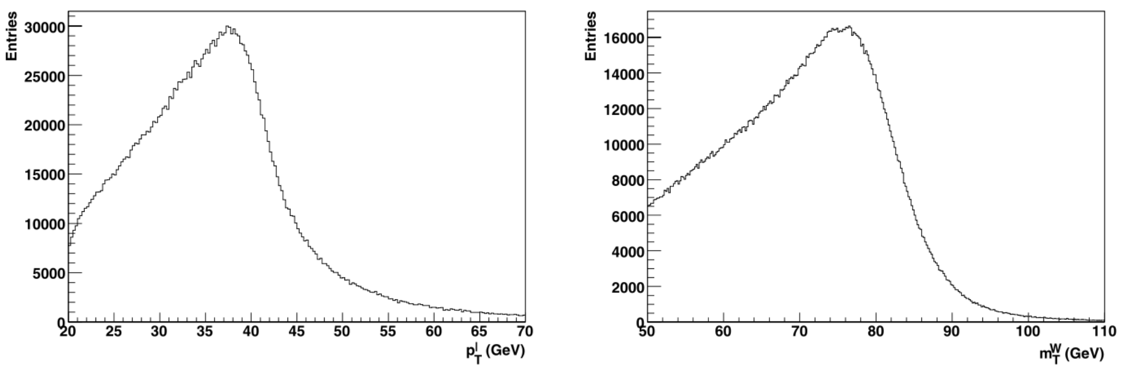


Figure 12.2: Transverse lepton momentum (p_{T}^{ℓ}) and transverse W boson mass (m_{T}^{W}) after fast detector simulation and with the event selection criteria applied. The image illustrates the particular shape ("Jacobian edge" or "Jacobian peak") of both distributions. The steep slope in the distribution results in a high sensitivity of the χ^2 value for the comparison of the simulated and measured distributions (image: [39]).

is determined as the analytic minimum of a parabola which is fitted to the χ^2 values as a function of the simulated values of M_W (Figure 12.4). This method allows to determine M_W with a systematic uncertainty of about 7 MeV for 10 fb^{-1} of integrated luminosity.

The dominating uncertainties for the measurement of M_W are the experimental uncertainties of the lepton and recoil measurements (Table 12.2). Both are affected significantly by the respective uncertainties arising from the energy scale, linearity of the detector response and the detector resolution. The lepton measurement is additionally affected by the reconstruction efficiency, in particular in the electron channel. Muon particles traversing the detector can be identified easily via the ATLAS muon spectrometer, whereas electrons suffer from bremsstrahlung effects that make them particularly prone to misidentification. "Tag and probe" methods are used to determine the corresponding lepton reconstruction efficiencies [39].

In order to minimize the systematic uncertainty of the M_W measurement, a precise calibration of p_{T}^{ℓ} and p_{T}^{ν} is required (discussed below in Section 12.1.3).

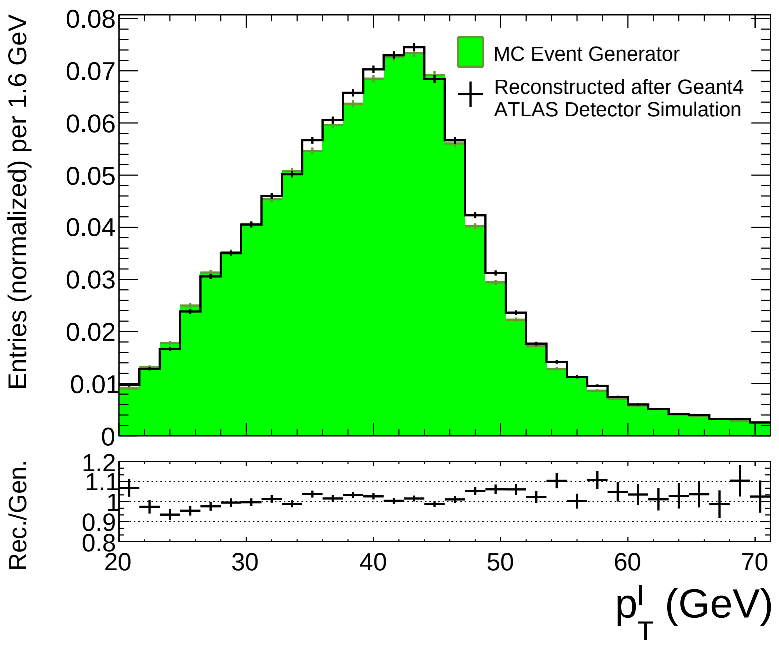


Figure 12.3: A comparison of the lepton transverse momentum (p_{T}^{ℓ}) in simulated $Z \to ee$ events between the MC generator output and the reconstruction output after Geant4 ATLAS detector simulation. The selection criteria for the $Z \to ee$ calibration method were applied on the reconstructed particles. The effect of the detector (simulation) and the reconstruction algorithms is noticeable through a widening of the distribution in the vicinity of the peak. Comparable effects are expected for the simulation of $W \to e\nu$ events.

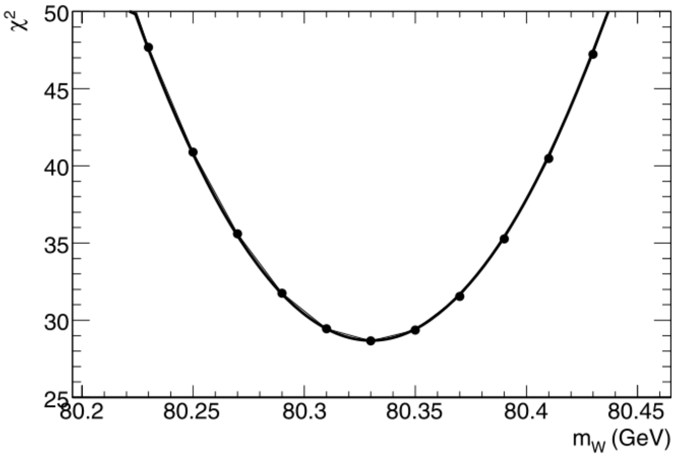


Figure 12.4: The computed χ^2 values as a function of the tested value of M_W (depicted as m_W). Each *dot* represents a comparison between the data measured in the ATLAS detector (simulated "pseudo data" in this case) and the simulated template distribution for a given value of M_W. The *curve* is the fitted parabola (image and caption: [39]).

12.1.3 Detector Calibration

$Z \to ee$ and $Z \to \mu\mu$ decays are used to calibrate the ATLAS detector response to the charged leptons (ℓ) and to calibrate the measurement of the transverse neutrino momentum (p_{T}^{ν}) [39, 104, 105]. The dominating processes generating Z bosons in the LHC are $q\bar{q} \to Z \to \ell\ell$ and $qg \to qZ \to q\ell\ell$, which are Drell-Yan processes [106, 107].

Lepton Calibration

With the precise knowledge of the Z boson mass and width [30], the ATLAS detector response to leptons ℓ (referring to e or μ) is measured in regions of particle energy and pseudorapidity η. Due to the relatively similar mass and the similarity of the processes involved in the creation of the Z boson and the W boson, the resulting leptons ℓ are generated in similar energy regimes.

In order to obtain a precise calibration for the electron energy measurement, an energy scale factor α and an energy resolution parameter a are introduced. Monte Carlo simulation is carried out to generate $Z \to ee$ samples with decalibrated electron energies. The electron energies are scaled with α and the resulting energy is smeared with the resolution parameter a, before the particle is simulated through the ATLAS detector. A histogram of the reconstructed di-electron invariant mass $m_{\ell\ell}$ (in the vicinity of the expected Z mass) is generated for each pair (α_i, a_i), respectively. χ^2 values are computed to quantify the agreement between each individual histogram obtained by simulation and the data recorded in the ATLAS detector. This leads to a set of $\left(\chi_i^2, \alpha_i, a_i\right)$ values (Figure 12.5). A paraboloid $\chi^2 = f\left(\alpha, a\right)$ is fitted through this set in the vicinity of the minimum. The calibration constants α_0 and a_0 are determined as the analytic minimum of the fitted paraboloid.

Calibration constants α_0 and a_0 are determined for different event categories. Individual categories are defined within regions of lepton energy and pseudorapidity. This leads to respective pairs of α_0^j and a_0^j values in bins of electron energy and pseudorapidity.

Recoil Calibration

p_{T}^{ν} is calibrated through using the same simulated and recorded $Z \to \ell\ell$ decays in the ATLAS detector which were previously used to obtain the lepton calibration (discussed above). Accurate transverse momentum measurements of the individual leptons ℓ are required in order to derive the calibration for p_{T}^{ν}. Hence, the previously obtained lepton calibration is applied throughout the method described in this section.

The hadronic recoil in $Z \to \ell\ell$ events is determined accurately through the measurement of the lepton transverse momenta (Equation 12.6). This accurate measurement is used to calibrate the less accurate hadronic recoil measurement which is computed from the vector sum of all transverse energy in the calorimeter, excluding the leptons (Equation 12.5). Corresponding scale (α_{hr}) and resolution (a_{hr}) parameters are derived from comparing the peak position and spread of the $E_{\mathrm{T}}^{\mathrm{hr}}/p_{\mathrm{T}}^{Z}$ distribution between recorded and simulated data [39]. The calibration of the hadronic recoil is then used for the

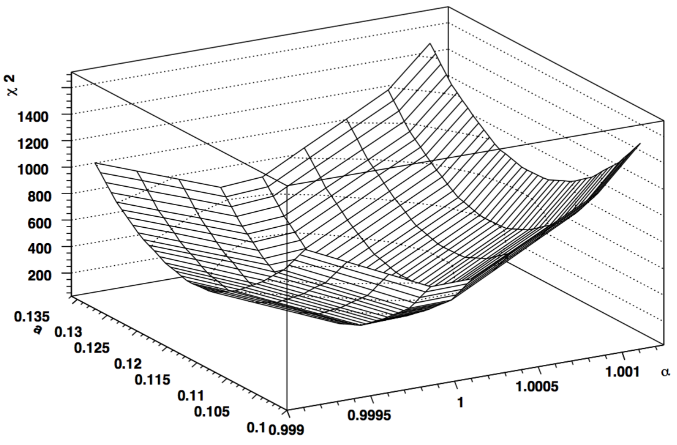


Figure 12.5: χ^2 values are computed to quantify the level of agreement between the simulated M_{ee} histograms with the data recorded in the ATLAS detector. One χ^2 value is obtained for each simulated sample with respective calibration parameters (α_i, a_i). A paraboloid is fitted to the set $\left(\chi_i^2, \alpha_i, a_i\right)$ in order to determine the point α_0 and a_0 with the minimum χ^2 (image: [105]).

accurate measurement of the transverse neutrino momentum in $W \to \ell\nu$ decays (Equation 12.7), which is required to compute the transverse mass (m_{T}^{W}) of the W boson (Equation 12.8).

12.2 Simulation Requirements

High statistics Monte Carlo simulation samples are required for accurate detector calibration and for the measurement of W boson mass. The same detector simulation method is to be used for the generation of both types of samples, as the determined calibration is specific to the chosen simulation setup. A fast simulation setup within the Integrated Simulation Framework is currently being validated for the generation of accurate detector simulation samples for lepton and recoil calibration. The application of the ISF for W boson signal simulation is foreseen and is to be studied in the future.

The level of agreement between a fast simulation setup and Geant4 is required to be of a similar magnitude as the observed level of agreement between Geant4 and recorded data. Small discrepancies between the individual simulators are of little relevance as the calibration method aims at minimizing the discrepancies between either simulation method and recorded data.

In order to calibrate the lepton response with $Z \to ee$ and $Z \to \mu\mu$ decays, a highly accurate detector simulation of the respective signal electrons and muons is required. Considering the level of agreement observed between Geant4 and recorded data in References [108, 109], the required level of agreement between a fast simulation setup and Geant4 is to be within $\lesssim 1\%$ in the respective lepton observables. Accurate simulation of the hadronic recoil ($E_{\mathrm{T}}^{\mathrm{hr}}$) observable is required to allow for the calibration of p_{T}^{ν}. A percent-level agreement ($\lesssim 10$) is required between the fast and the full simulation setups for the hadronic recoil variables. This is based on the level of agreement observed between Geant4 and recorded data in References [110, 111].

12.3 ISF Simulator Configuration

To fulfill the requirements for a accurate simulation of $Z \to ee$ calibration samples (Section 12.2), a dedicated ISF setup (also referred to as simulation flavour) is put in place. Geant4 is used to simulate electrons and positrons stemming from Z decays (including all particles in regions surrounding them), Fatras and FastCaloSim simulate all other particles in the event. Static and semi-dynamic routing rules (Section 6.1) are configured through various SimulationSelectors in the ISF routing chain (Section 6.2 and 7.2). A semi-dynamic ISF SimulationSelector defines cone-shaped regions of interest with a cone size of $\Delta R = 0.4$ in η-ϕ space (Figure 12.6). These ROIs are defined around all electrons and positrons in the simulation input which fulfill the following criteria:

- Their parent particle in the HepMC `GenEvent` structure is a Z boson.
- Their respective transverse momentum in the simulation input is above a threshold of $p_{\mathrm{T}}^{\ell} \geq 15$ GeV/c.

Exactly two such ROIs are expected in each simulated event. All primary particles inside this ROI are selected for Geant4 simulation.

As shown previously, the simulation accuracy potentially decreases if a particle and its secondaries are simulated by more than one simulator (Chapter 11). Therefore, the number of simulators a primary particle and its secondaries are simulated through is kept to a minimum in this ISF simulation flavour. If a primary particle is selected for Geant4 simulation, all its secondaries will be simulated by Geant4 throughout the whole detector ("sticky" Geant4 SimulationSelectors guarantee for that). Geant4 is exclusively used for primary particles in the ROI and their secondaries – there is no ROI requirement for these secondaries. All other particles (primary and secondary) in the ATLAS inner detector (ID) will be simulated by Fatras. Consequently, all secondary particles arising from Fatras simulation in the inner detector region will also be simulated in Fatras. FastCaloSim is used exclusively for calorimeter simulation of particles which were simulated by Fatras through the inner detector. With this simulator configuration, there are only two combinations of simulators a primary particle and its secondaries will encounter:

1. Geant4 for the entire simulation of a primary particle and all its secondaries.
2. Fatras for inner detector simulation and FastCaloSim for calorimeter simulation of a primary particle and all its secondaries.

Due to the low number of simulator combinations which particles encounter in this setup, only one specific simulator tuning is required to increase overall simulation accuracy (discussion below in Section 12.4).

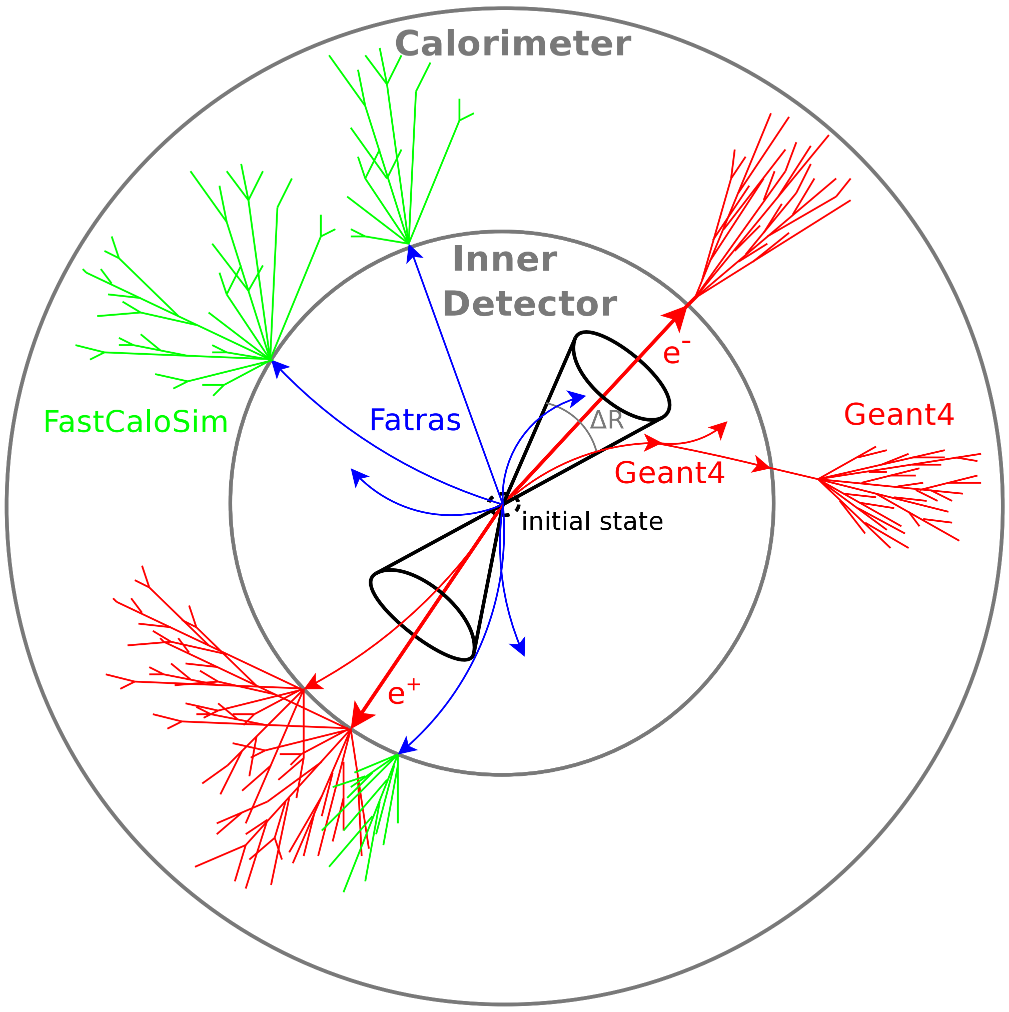


Figure 12.6: The ISF simulator configuration for fast and accurate ATLAS detector simulation of $Z \to ee$ events. The initial states of the Z boson decay products (e^+ and e^-) are used to define cone-shaped regions of interest in the detector simulation. All primary particles which are positioned within either of the ROIs will be simulated by Geant4 (red) throughout the entire detector. Any secondary particle arising from these primary particles will also be simulated by Geant4. The rest of the particles will be simulated by Fatras (blue) inside the inner detector and by FastCaloSim (green) inside the calorimeter.

12.4 Simulator Tuning

As described in the previous section, three different simulators are combined to form the ISF detector simulation setup for $Z \to ee$ event simulation. Any exchange of particles between Geant4 and any of the fast simulators is avoided in this setup. Thus, the only exchange of particles between simulators takes place between Fatras inner detector simulation and FastCaloSim calorimeter simulation.

The ATLAS calorimeter simulation using FastCaloSim is most accurate if the simulation input is formed by the MC truth representation generated by a preceding Geant4 inner detector simulation (Section 11.1). The MC truth representation generated by the $Z \to ee$ ISF fast simulation flavour, however, contains particles which are simulated by Fatras and Geant4 inside the ATLAS inner detector volume. Moreover, the MC truth representation contains both, particles that are to be simulated by Geant4 and particles that are to be simulated by FastCaloSim inside the calorimeter volume. Thus, the MC truth representation is not used as direct input to FastCaloSim calorimeter simulation. Instead, the individual particle states at the boundary between the inner detector and the calorimeter volumes are used to input FastCaloSim. This method of providing input to FastCaloSim is enabled through the implementation of the ISF and thus also called "ISF mode" of FastCaloSim (Section 11.1.2). It has been shown that the discrepancy between FastCaloSim and the full Geant4 simulation increases due to this approach in the ATLFASTIIF setup (Section 11.2). Hence, a specific tuning was developed to improve the agreement between Geant4 and FastCaloSim when using this input method in combination with Fatras inner detector simulation. The specifics of the tuning process and implementation are discussed in this section.

In consequence of providing input to FastCaloSim at the detector boundary, rather than the MC truth representation within the inner detector volume, the energy deposited in the calorimeter is underestimated by FastCaloSim (covered in Chapter 11). This is due to the fact that FastCaloSim intrinsically accounts for energy which is lost in interaction processes that occur to the particles when travelling from the inner detector volume towards the calorimeter. The simulation of this effect is inherent to the parametrization of FastCaloSim, as the simulated energy in the calorimeter is parameterized with respect to particle energies present in the inner detector volume. This effect, however, is already reflected in the particle states (energies) at the boundary between the inner detector volume and the calorimeter. Hence, the FastCaloSim calorimeter simulation using particle states at the detector boundary will account for this lost energy twice and consequently underestimate the energy deposited in the calorimeter.

The use of Fatras for the fast simulation of particles through the ATLAS inner detector volume results in a discrepancy of the total energy that reaches the boundary between the ID volume and the calorimeter with respect to Geant4 full simulation. This is due to higher energy thresholds for the simulation of secondary particles in Fatras. Consequently, less total energy is expected to reach the ID/calorimeter boundary in the fast ISF setup compared to full Geant4 simulation.

In agreement with these expectations, it is found that the FastCaloSim simulator in the ISF fast simulation flavour underestimates the energy which is deposited inside the calorimeter relative to full Geant4 simulation (Section 12.4.1). This discrepancy has a significant impact on the simulation accuracy of the hadronic recoil observable which is critical for the calibration of the p_{T}^{ν} observable in the W boson mass analysis. A local linear relationship between the energy deposited by the calorimeter simulation and the incoming particle energy is assumed. This relationship is assumed to be valid for small changes of the incoming particle energy only. Hence, energy scale factors are introduced which are applied to the particle states at the detector boundary in order to correct for the underestimated calorimeter energy in the FastCaloSim simulator (Section 12.4.2).

12.4.1 Calorimeter Response

To quantify the discrepancy of the energy deposited inside the calorimeter between Geant4 and the ISF mode of FastCaloSim, two types of calorimeter responses (r and r_{calo}) are relevant

$$r = \frac{E_{\text{calo}}}{E_{\text{primary}}}\ , \tag{12.9}$$

$$r_{\text{calo}} = \frac{E_{\text{calo}}}{E_{\text{caloEntry}}}\ . \tag{12.10}$$

E_{calo} is the total deposited energy in the calorimeter as computed by the detector simulation and stored in the HITS file format (Section 4.2). In this study, E_{calo} is determined by either FastCaloSim or Geant4, depending on the specific simulation setup. r is the simulated response of the calorimeter to a particle with the energy E_{primary} when leaving the interaction point. r_{calo} is the simulated response of the calorimeter to a particle with the energy $E_{\text{caloEntry}}$ just before entering the calorimeter sub-system. r combines the effects of inner detector simulation and calorimeter simulation. Thus, r expresses the overall response of the ATLAS detector to a given primary particle. r_{calo}, on the other hand, is a purely calorimeter-based variable. r_{calo} is used in this section to compare the response of calorimeter simulations in different detector simulation setups.

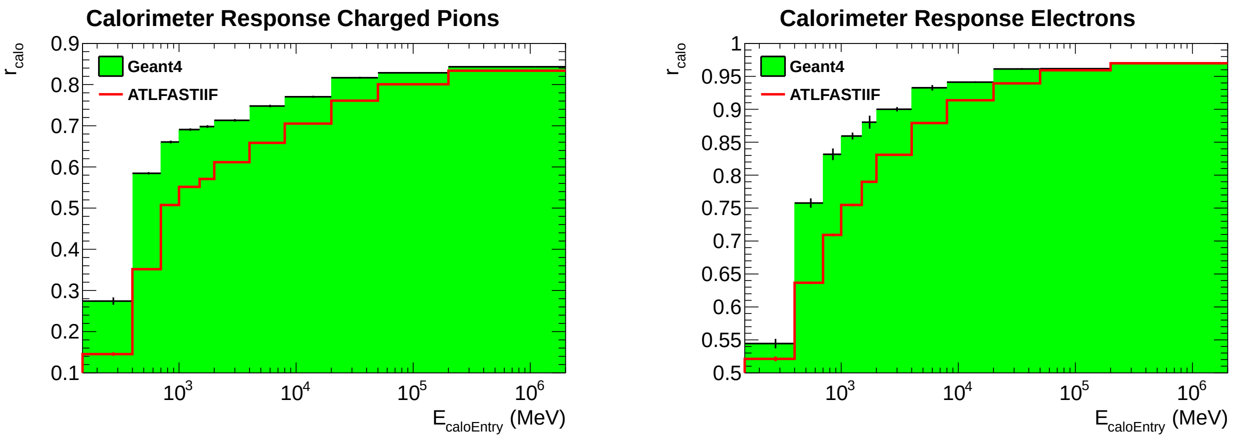
In consequence of using FastCaloSim in the ISF mode, r_{calo} is expected to be lower in comparison with Geant4 simulation, due to the double counting of the energy which is lost due to interactions in the ATLAS inner detector volume by FastCaloSim. A discrepancy of r_{calo} corresponds to an inaccurate simulation of the hadronic recoil which is required for accurate detector calibration.

The primary particle energy E_{primary} is obtained from the `TruthEvent` Monte Carlo truth collection (Section 7.4). For each event, E_{calo} is computed as the sum of the simulated calorimeter hit energies E_{hit} divided (or multiplied) by the respective sampling fraction f:

$$E_{\text{calo}} = \sum_{\text{LAr hits}} \frac{E_{\text{hit}}^{\text{LAr}}}{f^{\text{LAr}}} + \sum_{\text{tile hits}} E_{\text{hit}}^{\text{tile}} \cdot f^{\text{tile}}\ . \tag{12.11}$$

It is important to note that E_{calo} is not equal to the energy measured at the detector readout (after digitization) nor to the reconstructed calorimeter energy. Time thresholds are applied to the simulated calorimeter hits and noise is applied to the detector readout during the digitization step. Individual sensitive detector hits which were generated by Geant4 may be discarded in the digitization step if the respective time threshold is not fulfilled. Hit timing is not simulated by FastCaloSim, hence all sensitive detector hits generated by this simulator will pass the timing thresholds. Only a small fraction of E_{calo} does not pass the timing thresholds after Geant4 simulation, hence E_{calo} is a close approximation of the calorimeter energy represented in the readout signals. The impact of the time thresholds is therefore neglected in the following tuning approach. If deemed necessary, the thresholds might be applied in the computation of E_{calo} for an improved tuning approach in the future.

$E_{\text{caloEntry}}$ is obtained from the `CaloEntryLayer` StoreGate collection, which contains a Monte Carlo truth record of all particles passing the boundary between the ATLAS inner detector and the calorimeter volumes (Section 7.4.1). In ATLFASTIIF and in the $Z \to ee$ ISF simulation configuration, the particle information obtained from the `CaloEntryLayer` collection is equivalent to the simulation input of the ATLAS calorimeter simulation.



(a) (b)

Figure 12.7: The simulated calorimeter response r_{calo} at different particle energies $E_{\text{caloEntry}}$ (a) of single charged pion events (b) and of single electron/positron events. The image shows simulated events within which the respective initial particles do not undergo a significant interaction in the inner detector, i.e. no secondary particles with more than 50 MeV are generated. Geant4 and ATLFASTIIF detector simulation were carried out within ISF in the ATLAS offline software release `17.7.5.4`.

Figure 12.7 illustrates the calorimeter response r_{calo} of Geant4 and ATLFASTIIF simulation for single charged pion ($\pi^{\pm}$) and single electron/positron ($e^{\pm}$) events. Event selection criteria are applied in order to ensure that the simulated calorimeter energy is directly caused by the primary particle entering the calorimeter volume. Hence, events with one or more secondary particle (with energy > 50 MeV) in the ID volume are not considered in the figure and in the following discussion. The ATLFASTIIF simulation flavour is comparable to the ISF setup used for $Z \to ee$ simulation, as both utilize the ISF mode of FastCaloSim. As expected, the figure shows that the calorimeter response in the ATLFASTIIF setup is lower compared to Geant4 detector simulation. It follows that less energy is deposited in the calorimeter by FastCaloSim in ATLFASTIIF. This is a consequence of using FastCaloSim in the ISF mode. The tuning approach discussed in this section focusses on regaining the lack of deposited energy in the calorimeter due the ISF mode of FastCaloSim by introducing particle energy scale factors.

12.4.2 FastCaloSim Energy Scale Factors

Particle energy correction (or scale) factors are applied in the ISF $Z \to ee$ simulation flavour, in order to maximize the level of agreement between Geant4 and FastCaloSim calorimeter simulation. In this ISF setup, all input particles to FastCaloSim are simulated with Fatras through the ATLAS inner detector volume. The correction factors are applied to all fast simulation particles, after leaving the inner detector volume and before entering the calorimeter simulation. The aim is to compensate for the double counting of the energy which is lost due to interaction processes in the ATLAS inner detector volume due to the ISF mode of FastCaloSim.

A local linear relationship between the particle energy at the point of entering the calorimeter and the computed energy deposited inside the calorimeter is assumed. This allows to compute a scale factor for the incoming particle energy, based on a scaling required for the deposited calorimeter energy. This method is expected to be valid as long as the particle energies are scaled by small amounts, otherwise FastCaloSim were to create significantly different energy distributions inside the calorimeter. This would result in a non-linear dependency between the scaled particle energy and the simulated calorimeter energy.

Two types of correction factors are employed. The first type is dependent on the simulated particle type, energy and pseudorapidity. As the dependency is described in bins of the respective parameters, it is referred to as the *binned* energy correction factor. This scale factor is intended to be independent of the specific sample which is being simulated. The aim of the energy dependency of the scale factors is to divide the entire range of particle energies simulated by FastCaloSim into local regions within which a linear response of the calorimeter can be approximated. The η-dependency aims at providing individual scale factors per ATLAS calorimeter region. The second correction factor is applied globally to all particles in the fast simulation before entering the calorimeter. This scale factor is computed specifically for the $Z \to ee$ simulation sample, with the aim of correcting inaccuracies of the Fatras inner detector simulation (e.g. discrepancies in particle multiplicities and the total energy reaching the ID/calorimeter boundary).

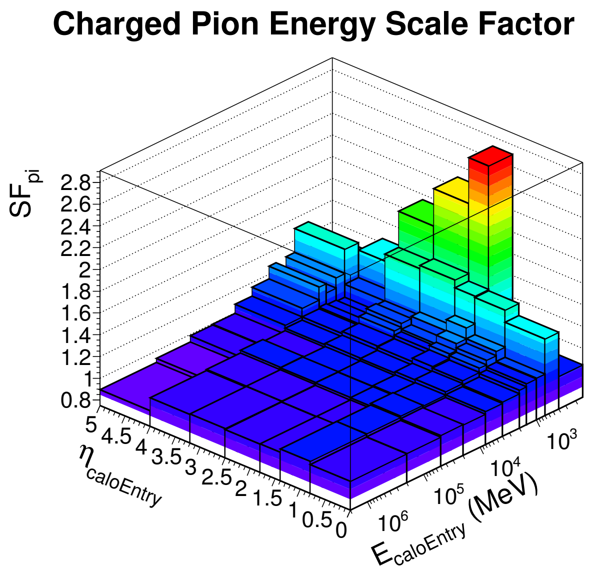
Both types of energy correction factors are discussed below.

Binned FastCaloSim Energy Scale Factor

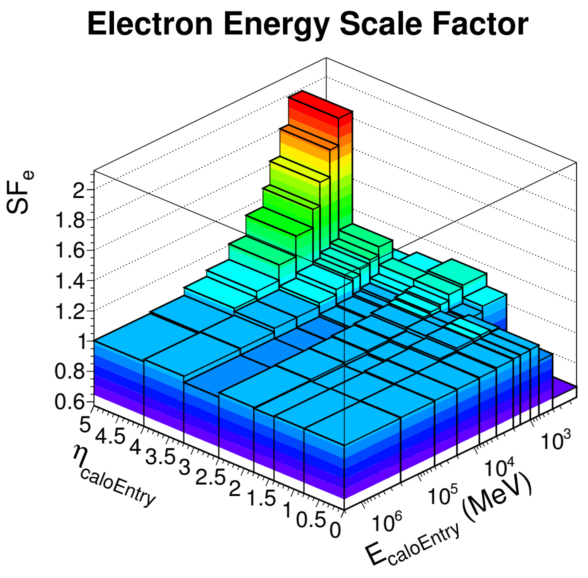
The binned energy correction factor for FastCaloSim in ISF mode (SF) is computed from the ratio between the Geant4 calorimeter response ($r^{\mathrm{G4}}_{\mathrm{calo}}$) and the calorimeter response computed by FastCaloSim in ISF mode ($r_{\text{calo}}^{\text{ISFFCS}}$) for single particle simulation. The error is computed under the assumption of independent and uncorrelated variables $r_{\text{calo}}^{\text{G4}}$ and $r_{\text{calo}}^{\text{ISFFCS}}$:

$$SF = \frac{r_{\text{calo}}^{\text{G4}}}{r_{\text{calo}}^{\text{ISFFCS}}} \,, \tag{12.12}$$

$$\Delta SF = \sqrt{\left(\frac{1}{r_{\text{calo}}^{\text{ISFFCS}}}\right)^2 \left(\Delta r_{\text{calo}}^{\text{G4}}\right)^2 + \left(\frac{r_{\text{calo}}^{\text{G4}}}{\left(r_{\text{calo}}^{\text{ISFFCS}}\right)^2}\right)^2 \left(\Delta r_{\text{calo}}^{\text{ISFFCS}}\right)^2} \,. \tag{12.13}$$



(a)



(b)

Figure 12.8: The FastCaloSim energy scale factors (a) SF_{pi} and (b) SF_{e} in bins of $E_{\text{caloEntry}}$ and $\eta_{\text{caloEntry}}$ ($150\text{ MeV} \leq E_{\text{caloEntry}} < 2\text{ TeV}$ and $|\eta_{\text{caloEntry}}| < 5$). The energy scale factors are computed using Geant4 and FastCaloSim in ISF mode in ATLAS offline software release `17.7.5.4`.

Charged pion ($\pi^{\pm}$) and electron/positron ($e^{\pm}$) single particle events are simulated with Geant4 and FastCaloSim in ISF mode in order to determine the respective energy scale factors. The initial particles originate from the nominal interaction point at $(x, y, z) = (0, 0, 0)$. The initial particle energy (E_{primary}) follows a $1/x$ distribution between 200 MeV and 2 TeV in the case of $\pi^{\pm}$, and between 50 MeV and 2 TeV in the case of $e^{\pm}$. This distribution is chosen to provide sufficiently high statistics in order to allow an accurate modelling of the logarithmic dependency of the calorimeter response to the particle energy (Figure 12.7). Moreover, it approximates the particle energy distribution in minimum bias events.

The detector material in the inner detector volume was removed for the generation of the Geant4 and the FastCaloSim samples. This increases the chance of the primary particle reaching the boundary between the inner detector and the calorimeter, without generating secondary particles due to interactions. In order to measure the response of the calorimeter simulation to individual particles, it is preferred to study events with only one particle entering the calorimeter simulation. The material of the ATLAS beam pipe was not removed in the Geant4 simulation due to technical reasons in the configuration of Geant4 for ATLAS detector simulation. In order to achieve high simulation statistics in due time, Fatras was used to extrapolate the particle tracks from the interaction point to the boundary between the inner detector and the calorimeter for the generation of the FastCaloSim sample. In both simulation setups, the particles are precisely tracked in the magnetic field of the ATLAS detector.

500 000 events are simulated for each respective particle type in each respective simulator. This totals to 2 million events simulated with ISF and Geant, respectively. Due to the simulation of the beam pipe material in the Geant4 setup, selection criteria are applied to reject all events with a significant interaction of the primary particle before it reaches the boundary between the inner detector and the calorimeter volumes. Events are rejected if the primary particle emits a child particle with a kinetic energy of 50 MeV or more before reaching the ID/calorimeter boundary. The scale factors SF_{pi} and SF_{e} are derived from the remaining set of events.

Figure 12.8 illustrates the energy scale factor SF in relation to the pseudorapidity $\eta_{\mathrm{caloEntry}}$ (of the particle position) and the particle energy $E_{\mathrm{caloEntry}}$ in the `CaloEntryLayer`. The precise binning is listed in Table 12.3. A tendency towards higher scale factor values at lower values of $E_{\mathrm{caloEntry}}$ can be observed. This indicates that these particles are significantly impacted by the double counting of the energy which is lost due to interaction processes in the ATLAS inner detector volume, which is inherent to the FastCaloSim parametrization and triggered by the ISF mode of FastCaloSim. Hence, the use of energy scale factors is particularly important for the accurate simulation of particles with low energies. The large scale factors in the region of $E_{\mathrm{caloEntry}} < 400$ MeV (lowest energy bin) illustrate the limitations of FastCaloSim. This is of no concern, as the particle energies are too low to be considered for accurate fast calorimeter simulation. However, this bin is computed separately, in order to factor out possible inaccuracies of the simulator into these dedicated bins.

The SF_{e} and the SF_{pi} scale factors are stored in separate ROOT `TH2` histograms within a ROOT file. The file serves as a look-up table for the FastCaloSim detector simulation (discussed below).

Variable	Number of bins	Bin edges
$\eta_{\text{caloEntry}}$	7	0.0, 0.8, 1.4, 1.8, 2.5, 3.2, 4.0, 5.0
$E_{\text{caloEntry}}$	11	150 MeV, 400 MeV, 700 MeV, 1 GeV, 1.5 GeV, 2.0 GeV, 4.0 GeV, 8.0 GeV, 20 GeV, 200 GeV, 2 TeV

Table 12.3: Bin edges of the energy scale factor (SF) for FastCaloSim in ISF mode. The bin positions and sizes for the $\eta_{\text{caloEntry}}$ are aimed at separating different calorimeter regions and technologies into respective bins. The $_{\text{caloEntry}}$ bin sizes are aimed at covering the logarithmic dependency of the calorimeter response (r_{calo}) on the particle energy.

Global ATLFASTIIF Calorimeter Energy Scale Factor

The use of Fatras to simulate particles in the inner detector volume causes the particles reaching the boundary between the ID and the calorimeter to have different properties compared to Geant4 full simulation. In particular a lower number of particles with low energies is expected to reach the boundary in the fast simulation. This is due to different energy thresholds for the generation of secondary particles inside the respective simulators. In general, the minimal energy a (secondary) particle must have in order to be simulated through the detector, is higher in fast simulations than in full simulations. This results in a lower number of particles being simulated by the fast[2] simulators. The energy carried by these particles is effectively "lost" inside fast simulators, as the energy of the particles that are simulated is modelled accurately. Therefore, less total energy is expected to arrive at the boundary between the inner detector volume and the calorimeter volume in the Fatras simulator compared to Geant4 (assuming the same input). A lack of deposited energy inside the calorimeter is expected as a consequence of this.

The particles simulated by Fatras trough the ATLAS inner detector volume in the fast $Z \to ee$ ISF simulation flavour are subsequently processed by FastCaloSim for calorimeter simulation. FastCaloSim only receives input from the Fatras simulator in this simulation setup. Hence, a (tunable) energy scale factor (SF_{global}) inside FastCaloSim is introduced in order to correct for the expected lack of energy at the ID/calorimeter boundary. The energy of a particle is scaled by this factor prior to its calorimeter simulation with FastCaloSim and after its inner detector simulation with Fatras. As the relative amount of energy which lacks in the fast simulation is expected to be dependent on the specific simulation sample, the optimal value of SF_{global} will depend on specific type of events which are simulated.

The reconstructed hadronic recoil (E_{T}^{hr}) distribution is used to tune SF_{global} in the ISF fast simulation against the Geant4 full simulation reference. The E_{T}^{hr} observable is directly dependent on the simulation performance of the fast simulators in this ISF simulation flavour. Fast simulators are used to simulate all particles outside the cone-

[2]After all, this is one reason for the high simulation speed of fast simulators.

shaped ROIs, which are defined only around the decay particles of the Z boson. The particles forming the hadronic recoil are expected to be mainly outside these ROIs and are therefore simulated by Fatras and FastCaloSim.

ISF fast simulation is carried out with different values of SF_{global}, using a $Z \to ee$ EVNT dataset[3] as input. The previously computed binned energy scale factors for FastCaloSim (previous section) are applied in all ISF simulation samples (Section 12.4.3 describes how the individual scale factors are combined an applied inside FastCaloSim). Standard Geant4 full simulation is carried out using the same input dataset in order to provide a reference for the parameter tuning. 120 000 events are simulated in the Geant4 reference sample and in each ISF sample with $SF_{\text{global}} \in \{0.990, 0.995, 1.000, 1.005, 1.010, 1.015\}$, respectively. 55 000 events are simulated in each respective ISF simulation sample elsewhere. The complete simulation chain is executed for each respective detector simulation sample, which includes: detector digitization, reconstruction and the creation of a flat ROOT n-tuple format called SMWZ D3PD (details regarding this full chain setup see Section 12.5). The specific event selection criteria of the W-mass detector calibration are not applied. Hence, the resulting E_{T}^{hr} distributions combine the effects of Fatras and FastCaloSim for the entire input dataset.

A least squares parameter fit is carried out in order to determine the optimal value of SF_{global}. For this, a χ^2 value is computed to quantify the level of agreement (of weighted histograms [71, 112]) between the individual ISF simulation results, each with a corresponding SF_{global} value, and the Geant4 simulation reference (Figure 12.9). A quadratic polynomial function of the variable SF_{global} is fitted to this set of $\left(SF_{\text{global},i}, \chi_i^2\right)$ values. The analytic minimum of the fitted function is calculated. The corresponding scale factor at the minimum is $SF_{\text{global}} = 1.002 \pm 0.003$.

The global scale factor at the computed minimum differs by less than one percent from the unscaled value. Moreover, it is compatible with the unscaled value within one standard deviation. Due to this, and to avoid sample-specific simulator tuning, the global energy scale factor is set to $SF_{\text{global}} \equiv 1.000$ in all studies presented in the results section of this chapter (Section 12.5). It is found that the tuning with binned energy scale factors (previous section) significantly enhances the level of agreement between fast simulation and Geant4 full simulation for the E_{T}^{hr} observable in the first place (see also Figure 12.12 in the results section).

[3]The EVNT dataset is the same as in the results Section 12.5 in this chapter.

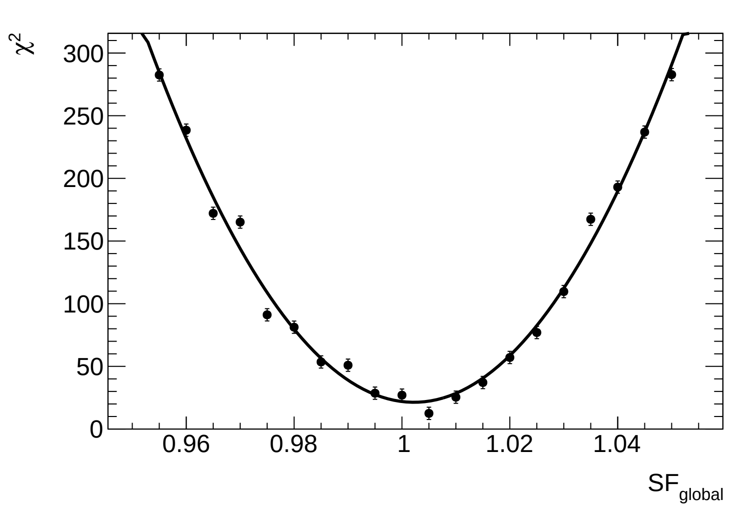


Figure 12.9: χ^2 values obtained by testing the reconstructed hadronic recoil ($E_{\mathrm{T}}^{\mathrm{hr}}$) distribution generated by Geant4 simulation against ISF simulation with different values of SF_{global}. A minimum χ^2 is determined in the fitted quadratic polynomial function at $SF_{\text{global}} = 1.002 \pm 0.003$.

12.4.3 Applying Energy Scale Factors in FastCaloSim

The two separate sets of energy scale factors are applied in the FastCaloSim simulation to all input particles that were simulated by Fatras through the ATLAS inner detector. For each individual input particle to FastCaloSim, the following procedure is followed:

1. The initial energy scale factor for the given particle is determined with the information present in the two-dimensional SF histograms (binned energy scale factors). A modified nearest-neighbour interpolation method is employed[4] to randomly select either the nearest-neighbouring bin or the second closest bin in each histogram dimension.

2. The scale factor is randomly "smeared" within its respective error σ, using a Gaussian probability distribution.

3. The "smeared" scale factor is multiplied by the global energy scale factor (SF_{global}) to form the combined energy scale factor (SF_{comb}).

4. The incoming particle energy ($E \equiv E_{\text{caloEntry}}$) is multiplied by SF_{comb}. The particle momentum (p) is re-computed to guarantee the validity of $E^2 = m^2 + p^2$ with the original rest mass (m) of the respective particle.

[4]This method will be replaced by a linear interpolation method in a future implementation.

5. The standard FastCaloSim parametrization is used to determine the calorimeter response to the modified input particle. The same FastCaloSim parametrization as in the ATLFASTII setup is used. The only difference is that a response is computed for each respective particle which reaches the boundary between the inner detector and the calorimeter, rather than for the entire MC truth collection present after a completed simulation of the ATLAS inner detector volume.

The scale factors are applied to a wider range of particle types than they are obtained from. The charged pion scale factor is applied to the most common types of hadrons that FastCaloSim encounters in a typical detector simulation setup. The electron/positron scale factor is also applied to photons (following the standard procedure of extrapolating e calibration to γ in the calorimeter) and neutral pions:

- SF_{e} is applied to all following FastCaloSim input particles:
  - charged pions (PDG type [113]: ± 211)
  - protons (PDG type: $+2212$)
  - neutrons (PDG type: $+2112$)
- SF_{pi} is applied to all following FastCaloSim input particles:
  - electrons and positrons (PDG type: ± 11)
  - photons (PDG type: $+22$)
  - neutral pions (PDG type: $+111$)

12.5 Results

The fast ISF simulation flavour described in the previous sections is validated for the detector simulation of $Z \to ee$ events[5] by comparison with a reference sample which is simulated entirely by Geant4. The Fatras and FastCaloSim tuning parameters (Section 12.4) are used in the ISF fast simulation flavour in order to increase the accuracy of the fast simulators. To allow for a better interpretation of the simulation results, an additional ISF simulation dataset is generated within which the simulator tuning is not applied.

120 000 events are simulated with either setup. However, the exact number of final events may differ slightly between the individual samples due to technical reasons. To allow for a direct comparison between the different detector simulation approaches, the same digitization and reconstruction steps were performed on either detector simulation sample. Pile-up effects are not simulated. This provides a cleaner picture of the ISF fast simulation effects and therefore allows for better interpretation of the results.

The ISF and Geant4 detector simulation were performed in ATLAS offline software release `17.7.5.4` (using ATLAS geometry version `ATLAS-GEO-20-00-01` and

[5]The following ATLAS event generator (EVNT) dataset is used as detector simulation input: mc12_8TeV.129680.PowhegPythia8_AU2CT10_Zee_DiLeptonFilter.evgen.EVNT.e1861

detector conditions tag `OFLCOND-MC12-SIM-00`), the HITS file merging[6] in release `17.6.51.4`[7], the digitization and reconstruction in release `17.3.12.7` (detector conditions tag `OFLCOND-MC12-SDR-06`) and the generation of flat ROOT n-tuples in the SMWZ D3PD format [114] in release `17.3.11.1.3`[8]. The analysis algorithms with the corresponding event selection criteria (Table 12.1) are executed after the D3PD generation step. The respective D3PD datasets are provided as input to the analysis algorithm.

A correspondence table for the presented observables and the respective variables in the D3PD or W-mass analysis framework is given at the end of the Section (Table 12.4).

12.5.1 Physics Performance

Global event observables and analysis-specific observables are compared between the ISF simulation and the Geant4 simulation datasets in order to determine the accuracy of the ISF $Z \to ee$ fast simulation flavour.

Tracking

In the ISF simulation flavour all particles outside the respective ROIs are simulated using Fatras throughout the ATLAS inner detector volume. The majority of the particles traversing the inner ATLAS tracking detector ("inner detector") in the $Z \to ee$ events is therefore simulated by Fatras. Consequently, event-global tracking observables (Figure 12.10) will represent mainly the performance of the Fatras simulation in this ISF setup.

A good overall agreement is observed for tracking variables between ISF and the Geant4 simulation reference. However, the sum of the transverse momenta of all reconstructed tracks ($\sum p_T$) is slightly shifted towards higher energies in ISF. Moreover, ISF generates slightly less tracks with low energies in the region of 5 GeV/c $\lesssim p_T \lesssim$ 13 GeV/c. The discrepancies are very minor and not critical, thus the precise source is not yet identified. Possible causes could be discrepancies between Fatras and Geant4 in the modelling of particle energy losses, multiple scattering or particle-matter interactions. Fatras implements a fast and custom-made energy-loss model for particles. This model is validated and tuned through single particle simulation at discrete energies with reference to Geant4. The differences in the $\sum p_T$ could be triggered for example due to Fatras underestimating the energy loss of particles in the ATLAS inner detector. Other sources could be discrepancies in the scattering angles of the Fatras multiple scattering model or lower rates of hadronic interactions computed by Fatras. Both could

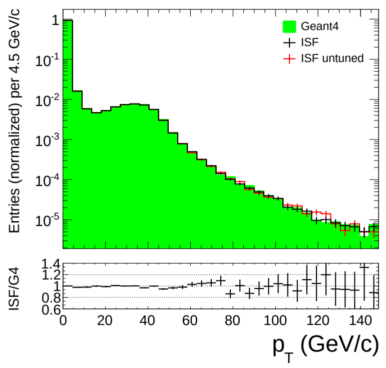
[6] A ROOT file format conversion is carried out during the HITS merging step. The ATLAS geometry version `ATLAS-GEO-20-00-01` and the conditions tag `OFLCOND-MC12-SIM-00` are used for the detector simulation. The detector simulation output which is generated in ATLAS offline software release `17.7.` is made compatible with the subsequent digitization and reconstruction steps in release `17.3.`

[7] This release has a known issue affecting the size of the MC truth collection in the merged HITS file. It is not expected that this issue has any impact on the results presented in this section, as all studied quantities are completely independent of the MC truth representation.

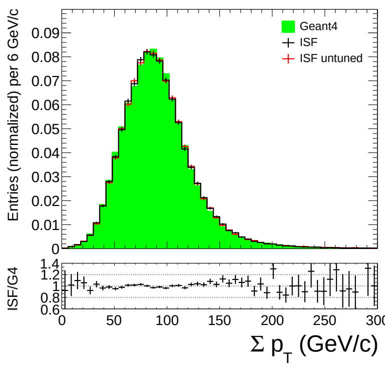
[8] The following software package versions are used in addition to the ATLAS offline software release `17.3.11.1.3`: `D3PDMakerConfig-00-03-58-08`, `McParticleAlgs-00-09-03`, `HadronicRecoil-00-02-05` and `PhysicsD3PDMaker-00-00-68`.

affect the reconstruction efficiency of particles simulated by Fatras and hence impact the event-global tracking variables.

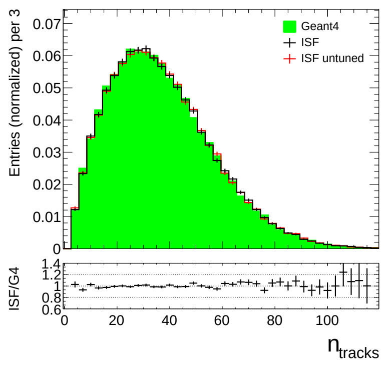
No significant impact of the fast simulator tuning (previous sections) to event-global tracking variables is expected as the tuning only affects the calorimeter simulation. This is consistent with the results presented in this section.



(a)



(b)



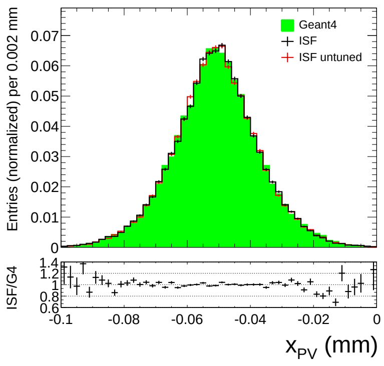
(c)

Figure 12.10: Event-global tracking variables in $Z \to ee$ events simulated by ISF fast simulation and Geant4 full detector simulation. Shown are (a) the transverse momentum distribution of the reconstructed tracks (b) the sum of the transverse momenta of all reconstructed tracks (c) the number of reconstructed tracks per event. Fatras simulates most of the tracks in the ATLAS inner detector in this ISF setup, hence, its performance is crucial for the accuracy of event-global tracking variables.

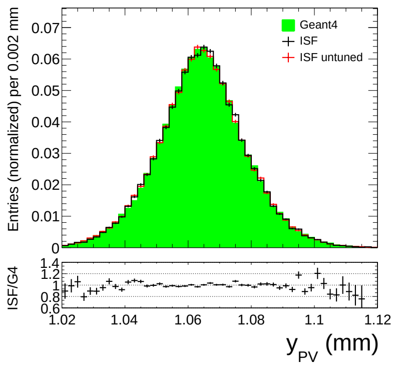
Vertexing

A comparison of the primary vertex positions in the selected $Z \to ee$ events between the ISF simulation setup and the Geant4 full simulation reference is illustrated in Figure 12.11. The primary vertex positions are computed by identifying common vertices from which particle tracks in the ATLAS inner detector originate. As the particles in the ATLAS inner detector are simulated by Fatras and Geant4 in this ISF setup, the simulated accuracy of the primary vertex position depends directly on the performance of these two simulators. A good agreement between ISF and Geant4 is observed.

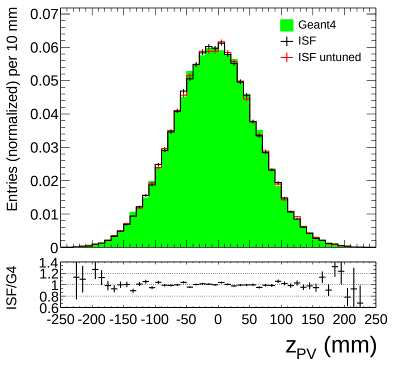
The vertexing algorithms use the results obtained by tracking algorithms (see above), hence, no significant impact of the fast simulator tuning is expected. This is consistent with the results presented in this section.



(a)



(b)



(c)

Figure 12.11: Primary vertex positions in $Z \to ee$ events simulated by ISF and Geant4 full detector simulation: (a) vertex position in x (b) vertex position in y and (c) vertex position in z. In the ISF setup, the performance of both the Fatras and the Geant4 simulator directly impact the accuracy of the illustrated primary vertex positions.

Calorimeter energy and hadronic recoil

In order to calibrate the calorimeter energy response, an accurate simulation of the deposited energy in the calorimeter is required. A comparison of the total energy simulated in the calorimeter is given by the scalar sum of the reconstructed transverse energy ($\sum E_\mathrm{T}$) in the selected $Z \to ee$ events (Figure 12.12a). The hadronic recoil (E_T^hr) is the vector sum of all transverse energy measurements, excluding the energy measurements which are within a cone (of size $\Delta R = 0.2$) around the reconstructed charged lepton(s) (Figure 12.12b). It plays a crucial role in the calibration of the p_T^ν observable, therefore a high simulation accuracy is required.

In this fast ISF simulation flavour, the energy deposited in the calorimeter is composed of the energy deposits generated by FastCaloSim and Geant4 together. Geant4 is used to simulate all signal particles (electrons/positrons from Z decays). Each signal particle represents a significant fraction of the total energy in the event. The particles provided as input to FastCaloSim were simulated by Fatras through the ATLAS inner detector volume in the preceding step. Thus, the accuracy of the simulated energy in the calorimeter depends on the accuracy of all three simulators.

The $\sum E_\mathrm{T}$ distribution generated by ISF shows a slight shift towards higher energies. The mismodelling of energy depositions in the forward calorimeter by FastCaloSim (see below) is likely to cause this shift. Nevertheless, the fast simulator tuning improved the agreement with respect to Geant4.

As all particles outside the cone-shaped ROIs are simulated by fast simulators, the accuracy of the E_T^hr observable depends directly on the accuracy of Fatras and FastCaloSim in the $Z \to ee$ fast ISF simulation setup. An indirect dependency on the simulation accuracy of the Z boson decay products exists, however, this part of the event is accurately simulated with Geant4 (see below). The significant improvement of the hadronic recoil distribution due to the fast simulator tuning underlines the crucial dependency of this variable on the accuracy of FastCaloSim. This also proves the validity and robustness of the applied tuning method. It is worth noting that the tuning approach is derived from single particle simulation and it is therefore not specific to this sample. A sample-dependency of the tuning was averted by choosing a global energy scale factor of $SF_{\text{global}} \equiv 1.000$ for the combination of Fatras and FastCaloSim (Section 12.4.2).

Even though the simulator tuning significantly increases the accuracy of the E_T^{hr} distribution, a discrepancy is observed in the lowest bins of the histogram. This is likely caused by the mismodelling of the forward calorimeter region in FastCaloSim (see below).

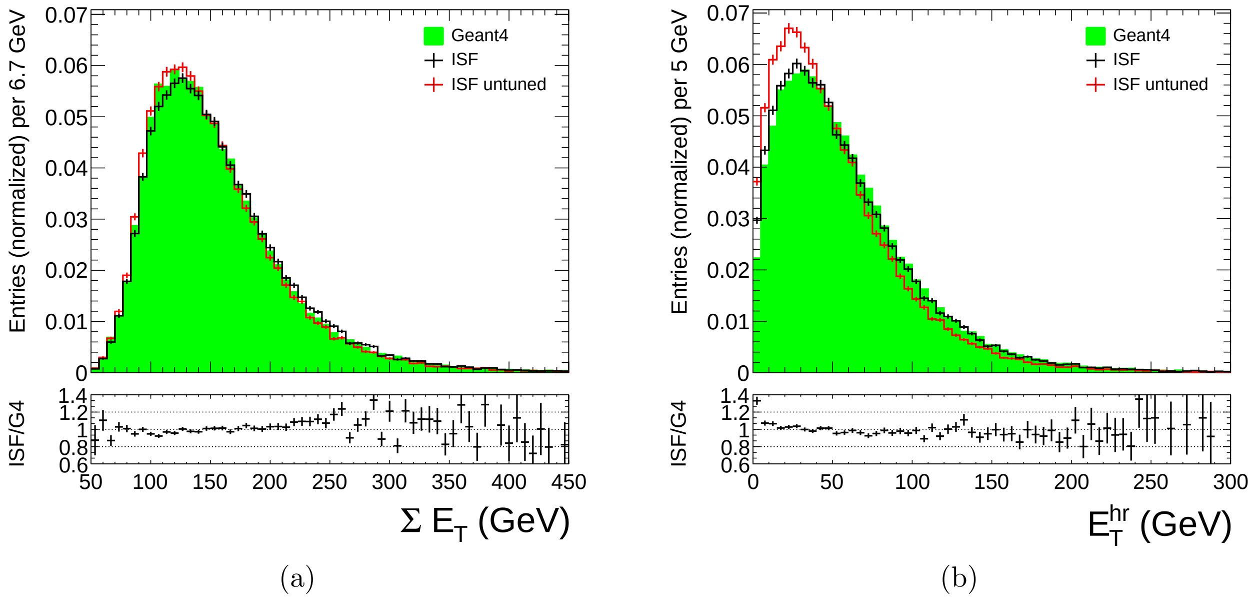
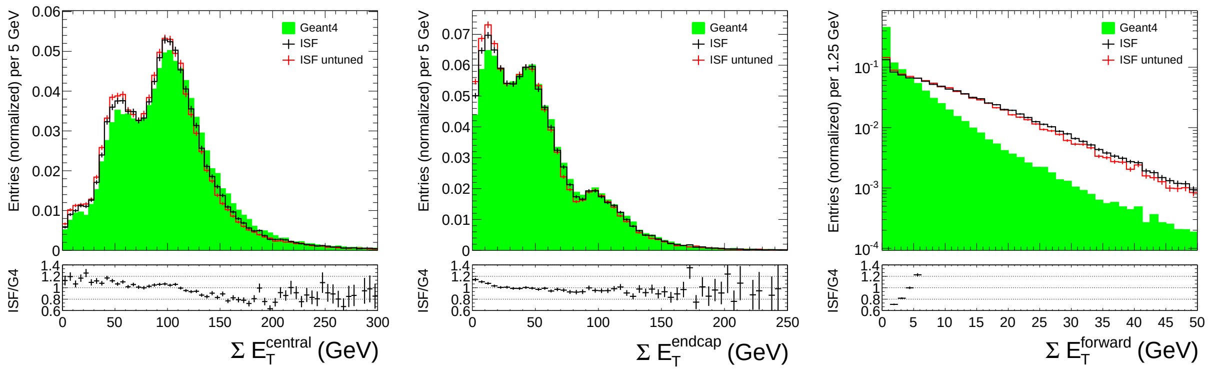


Figure 12.12: The sum of the reconstructed transverse energies in the calorimeter for $Z \to ee$ events simulated by ISF and full Geant4 detector simulation. The (a) scalar sum of the entire transverse energy reconstructed in the ATLAS calorimeter and (b) the computed hadronic recoil are shown. Both quantities are strongly correlated with the Fatras and FastCaloSim simulator tuning.

Calorimeter energy per region

The simulated energy within the different calorimeter regions depends on the performance of all three simulators in the ISF fast simulation flavour.

Even with the FastCaloSim tuning applied in the ISF setup, differences in the reconstructed transverse energy per calorimeter region can be observed (Figure 12.13). The most significant discrepancy appears in the forward calorimeter region ($3.2 < |\eta| < 4.9$). This is due to intrinsic limitations of FastCaloSim, which does not provide a dedicated parametrization for the forward calorimeter (FCAL). The fact that the tuning approach fails to mitigate the discrepancy in the forward region, suggests that this is not caused by a discrepancy of the amount of energy deposited in the calorimeter by the FastCaloSim simulator. Energy scale factors would need to take reconstruction effects into account in order to correct for this discrepancy. In the central ($|\eta| < 1.5$) and end-cap ($1.5 < |\eta| < 3.2$) regions, the level of disagreement is decreased due to the FastCaloSim tuning method. However, a discrepancy still remains between ISF and Geant4, and thus, altered tuning approaches are necessary. The poor simulation accuracy within the FCAL region likely causes an underestimation of the global energy scale factor (SF_{global}) during the tuning process and hence causes the differences in the central and endcap regions. An in-depth interpretation and suggestions on how to improve the agreement of the calorimeter observables are provided in the discussion Section 12.6.



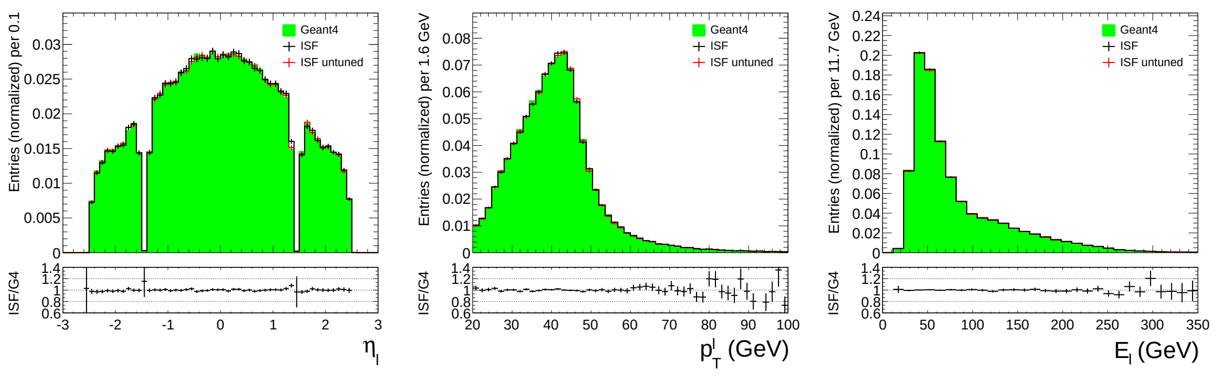
(a) (b) (c)

Figure 12.13: The scalar sum of the reconstructed transverse energies in different parts of the ATLAS calorimeter for $Z \to ee$ events simulated by ISF and full Geant4 detector simulation. The (a) central calorimeter region with $|\eta| < 1.5$ (b) the end-cap region with $1.5 < |\eta| < 3.2$ and (c) the forward calorimeter region with $3.2 < |\eta| < 4.9$ are shown. The discrepancy in (c) is due to internal limitations of FastCaloSim. This might only be resolved by a re-implementation of FastCaloSim. The applied FastCaloSim tuning contributes significantly to the accuracy of ISF in (a) and (b).

Lepton properties

A highly accurate simulation of electrons and positrons (ℓ) from Z boson decays is a requirement to this ISF setup. Hence, these particles are simulated by Geant4 within the ISF. Figure 12.14 illustrates the reconstructed electron/positron properties for the ISF simulation setup and the Geant4 full simulation reference. As expected, a high level of compatibility with the Geant4 full simulation reference is observed.

The fast simulator tuning shows no significant impact on the observables of signal electrons/positions. This is expected, as the measurements of the leptons from Z boson decays depend mainly on the simulation performance of Geant4.



(a) (b) (c)

Figure 12.14: Lepton measurements in $Z \to ee$ events simulated by ISF and Geant4 full detector simulation. The signal particles are simulated by Geant4 in this ISF simulation flavour, thus a good agreement with Geant4 full simulation is expected and observed in all observables: (a) electron pseudorapidity η_ℓ (b) electron transverse momentum p_{T}^{ℓ} (c) and electron energy E_ℓ.

Lepton isolation

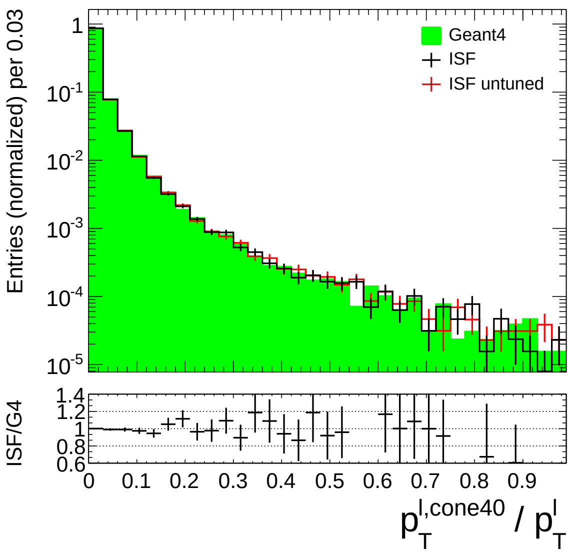
The lepton isolation is studied to determine the accuracy of the detector simulation in the vicinity of the signal electrons/positrons. Two types of lepton isolation variables are studied: track isolation and calorimeter isolation:

$$i_R = p_{\mathrm{T}}^{\ell,\mathrm{cone40}} / p_{\mathrm{T}}^{\ell} \qquad (12.14)$$

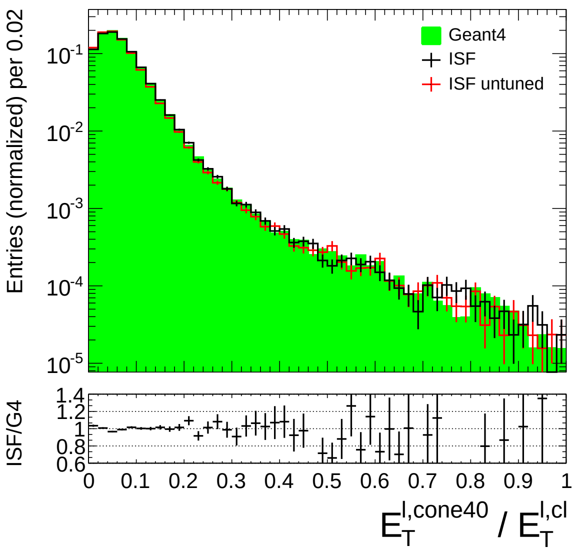
$$I_R = E_{\mathrm{T}}^{\ell,\mathrm{cone40}} / E_{\mathrm{T}}^{\ell,\mathrm{cl}} \qquad (12.15)$$

Here, p_{T}^{ℓ} ($E_{\mathrm{T}}^{\ell,\mathrm{cl}}$) denotes the measured transverse momentum (transverse energy of the associated calorimeter cluster) of the selected lepton and $p_{\mathrm{T}}^{\ell,\mathrm{cone40}}$ ($E_{\mathrm{T}}^{\ell,\mathrm{cone40}}$) denotes sum of the transverse momenta of all tracks (calorimeter energy measurements) within a cone of size $\Delta R = 0.4$ around this lepton. The transverse momentum (transverse cluster energy) of the respective lepton is excluded from this sum. In the ISF setup, these variables are mainly dependent on the Geant4 simulation, as all initial particles inside cones (of size $\Delta R = 0.4$) around the signal leptons are simulated by Geant4. However, particles simulated by Fatras can bend into these cone-shaped ROIs (or come close to them) and thus impact the lepton isolation measurement.

Figure 12.15 shows a good agreement between ISF and Geant4 simulation. An noticeable improvement in the accuracy of the calorimeter isolation is observed due to the fast simulator tuning. This illustrates that particles which are processed by fast simulators do deposit a noticeable amount of energy within cones of size $\Delta R = 0.4$ around the signal leptons. It is important to note that the cone-shaped ROIs in the ISF simulation are also of size $\Delta R = 0.4$ around the signal leptons. It is therefore expected that this effect is less pronounced with smaller isolation cone sizes. This is confirmed in Figure 12.16 which illustrates the track and calorimeter isolation variables for an isolation cone size of $\Delta R = 0.2$. The impact of the fast simulator tuning is far less noticeable with the smaller isolation cone size. This indicated that a smaller fraction of the deposited energy inside the isolation cone is generated by fast simulators.

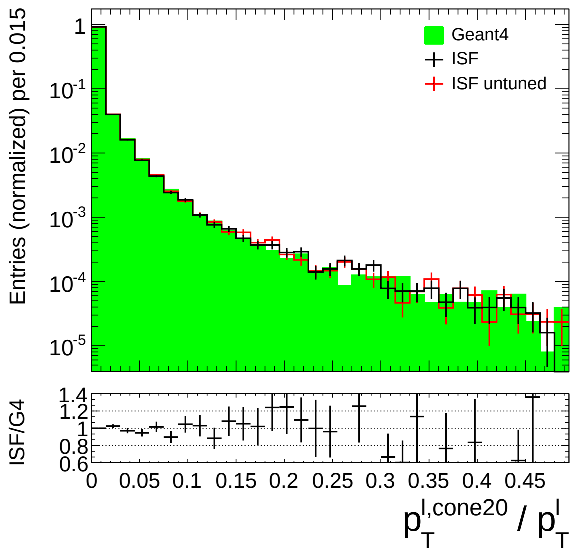


(a)

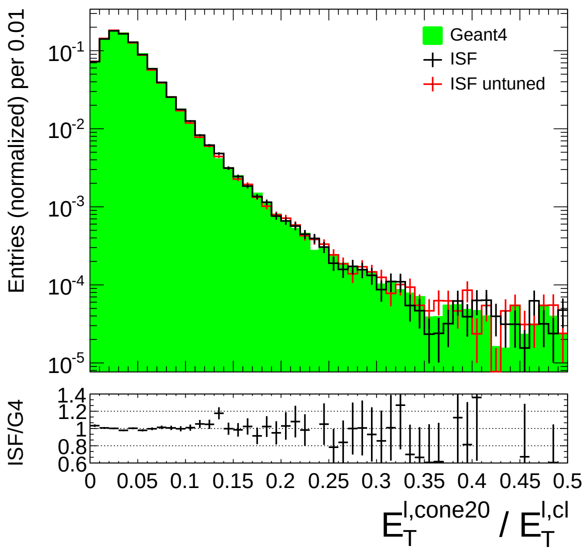


(b)

Figure 12.15: (a) Track isolation and (b) calorimeter isolation of selected leptons in $Z \to ee$ events simulated by ISF and full Geant4 simulation. The level of accuracy of the isolation variables in cones of size $\Delta R = 0.4$ reflects the performance of the combination of all three simulators in this ISF setup: Fatras, FastCaloSim and Geant4. The fast simulator tuning shows a noticeable improvement of the calorimeter isolation variable for this isolation cone size.



(a)

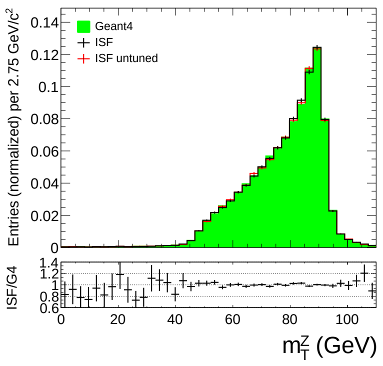


(b)

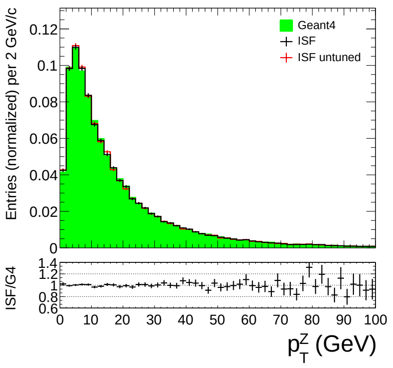
Figure 12.16: (a) Track isolation and (b) calorimeter isolation of selected leptons in $Z \to ee$ events simulated by ISF and full Geant4 simulation. The level of accuracy of the isolation variables in cones of size $\Delta R = 0.2$ reflects the performance of the combination of all three simulators in this ISF setup: Fatras, FastCaloSim and Geant4.

Z boson properties

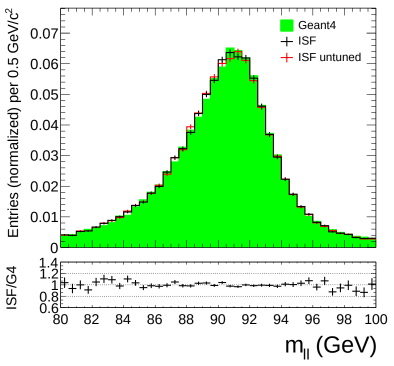
Finally, the simulation accuracy of the reconstructed Z boson properties is studied (Figure 12.17). As the signal electrons in the ISF simulation setup are simulated in Geant4, a high degree of agreement with the full simulation reference is expected and indeed observed. This holds true for all variables which are directly derived from the electron measurements. As expected, the fast simulator tuning does not seem to impact the reconstructed Z boson properties.



(a)



(b)



(c)

Figure 12.17: Z boson measurements in $Z \rightarrow ee$ sample with ISF and full Geant4 detector simulation. The signal particles in this ISF setup are simulated with Geant4, thus a very good agreement between ISF and full Geant4 is expected and observed in all observables (a) the transverse Z boson mass m_{T}^{Z} (b) the Z boson transverse momentum p_{T}^{Z} (c) and the invariant mass of the two-electron system $m_{\ell\ell}$.

Observable	SMWZ D3PD variable	Analysis framework variable
p_{T}	`trk_pt`	-
$\sum p_{\mathrm{T}}$	`MET_Track_sumet`	-
n_{tracks}	`trk_n`	-
x_{PV}	`vxp_x`	-
y_{PV}	`vxp_y`	-
z_{PV}	`vxp_z`	-
$\sum E_{\mathrm{T}}$	`MET_RefFinal_sumet`	-
E_{T}^{hr}	`hr_corrRecoil_20_sumet`	-
$\sum E_{\mathrm{T}}^{\mathrm{central}}$	`MET_RefFinal_sumet_CentralReg`	-
$\sum E_{\mathrm{T}}^{\mathrm{endcap}}$	`MET_RefFinal_sumet_EndcapRegion`	-
$\sum E_{\mathrm{T}}^{\mathrm{forward}}$	`MET_RefFinal_sumet_ForwardReg`	-
η_{ℓ}	-	`selElec_tlv.Eta()`
p_{T}^{ℓ}	-	`selElec_tlv.Pt()`
E_{ℓ}	-	`selElec_tlv.E()`
$p_{\mathrm{T}}^{\ell,\mathrm{cone20}}/p_{\mathrm{T}}^{\ell}$	`el_ptcone20[selElec_index]` / `selElec_tlv.Pt()`	
$p_{\mathrm{T}}^{\ell,\mathrm{cone40}}/p_{\mathrm{T}}^{\ell}$	`el_ptcone40[selElec_index]` / `selElec_tlv.Pt()`	
$E_{\mathrm{T}}^{\ell,\mathrm{cone20}}/E_{\mathrm{T}}^{\ell,\mathrm{cl}}$	`el_Etcone20[selElec_index]` / `selElec_tlv_cl.Et()`	
$E_{\mathrm{T}}^{\ell,\mathrm{cone40}}/E_{\mathrm{T}}^{\ell,\mathrm{cl}}$	`el_Etcone40[selElec_index]` / `selElec_tlv_cl.Et()`	

Table 12.4: Correspondence table for the observables presented in this section and the respective variables in the ATLAS SMWZ D3PD file format or the W-mass analysis framework. Two Lorentz vectors (`selElec_tlv` for the track measurement and `selElec_tlv_cl` for the cluster measurement) and the electron index in the D3PD format (`selElec_index`) are computed by the analysis framework for each selected electron.

12.5.2 Computing Performance

The main goal of the dedicated ISF fast simulation flavour for the W boson mass analysis is to enable the generation of bigger simulation samples within a shorter period of time. Thus, a detailed study of the computing time spent in each individual step of this ISF setup is compared to the corresponding step in a full Geant4 setup.

The MC simulation chain is configured according to the description given at the beginning of Section 12.5 with these two modifications:

- A more recent ATLAS offline software release is used in the HITS file merging step: `17.6.51.5`.
- An updated version of a transient-persistent (TP) converter package is used in the digitization, reconstruction and SMWZ D3PD creation steps, respectively: `GeneratorObjectsTPCnv-00-06-03-01`.

Both modifications are required in order to resolve a known issue in the transient-persistent conversion of the MC truth representation. The issue affects the size of the MC truth collection if fast simulators are used for detector simulation. It would distort all measurements presented in this section. Hence, the measurements are performed with the issue being resolved in each respective MC production step. The issue is not expected to impact the physics performance studies presented in Section 12.5.1.

Table 12.5 shows the significant speedup that is achieved with the ISF detector simulation compared to the Geant4 full simulation for the $Z \to ee$ samples. As expected, the speedup is most prominent in the detector simulation step. The measurements reveal that all subsequent steps in the Monte Carlo production chain (with the exception of ESD to AOD file format conversion) are sped up by the use of the Fatras and FastCaloSim fast detector simulators. This is likely due to the higher energy thresholds implemented in the fast simulators compared to the full Geant4 simulator. Fatras does not simulate particles with energies below 100 MeV, as they are of little relevance for physics analyses. FastCaloSim uses a parametrization model to compute the deposited energy in the calorimeter directly from the state of a particle at the boundary between the inner detector and the calorimeter volumes. It simulates the net effect that a resulting shower of particles has on the sensitive calorimeter elements, without simulating each particle in the shower individually. Consequently, the detector simulation output of the fast simulators contains less sensitive detector hits and a simpler MC truth representation. This speeds up the processing of all subsequent steps and in addition, generates smaller output file sizes than full Geant4 simulation (Table 12.6).

The processing time of the detector simulation step in the ISF fast simulation flavour is found to be dominated by Geant4 detector simulation (Figure 12.18). CPU profiles obtained with GPerftools reveal that 96.8% of the CPU time required by Athena Algorithm execute step is spent inside Geant4 routines, 0.7% inside Fatras routines and 0.7%

[10]The specifications of the computer are: Scientific Linux CERN 6, 64 GiB memory and 4 Intel® Xeon® E5-2650 v2 @ 2.60 GHz processors.

MC Production Step	Geant4 Full Simulation		ISF Fast Simulation		ISF Speedup
	(ms/event)	% of total	(ms/event)	% of total	
Detector Simulation	197000 ± 4000	98	39000 ± 1000	90	$5.0\times$
HITS Merging	85 ± 1	0.04	35.8 ± 0.6	0.08	$2.4\times$
Digitization	930 ± 10	0.5	686 ± 6	1.6	$1.4\times$
Reconstruction	2540 ± 20	1.3	2340 ± 20	5.3	$1.1\times$
ESD to AOD	98.4 ± 0.8	0.05	96.5 ± 0.8	0.2	$1.0\times$
SMWZ D3PD gen.	1360 ± 20	0.7	1280 ± 20	2.9	$1.1\times$
Total	202 ± 4 s/event		44 ± 1 s/event		$4.6\times$

Table 12.5: The average CPU time (in milliseconds if not otherwise noted) required to process one $Z \to ee$ event in ISF and Geant4 full simulation in different MC production steps. 500 events are processed in each simulation setup and MC production step respectively. The PerfMonSD service (Section 10.1.1) generates the respective CPU time measurements. The CPU time required to process the first event is not taken into account. The measurements were carried out one at a time. The corresponding Athena processes were the only high-workload processes scheduled on the computer[10]at the time of the measurement.

inside FastCaloSim routines. Hence, any improvement to the execution speed of Geant4 will significantly impact the execution speed of the ISF fast simulation flavour.

The execution time of the HITS file merging step is dominated by input-output operations and as such depends mainly on the size of the events that are being processed. No computationally demanding algorithms are being executed with the processed data. Hence, a significantly faster processing speed is observed for the ISF simulation sample.

In the digitization step the individual sensitive detector hits are combined and a detector response is computed for the different readout channels. A reduction in the processing time is observed for the ISF sample due to the significantly smaller size of the detector simulation output, i.e. fewer sensitive detector hits. Detector noise is simulated during the detector digitization step and stored in the ROD output data format. This explains why the relative difference of the output files between the ISF and Geant4 simulation samples decreases significantly with the detector digitization step. HITS files which are about a factor of 2.8 smaller in the ISF sample, are processed into ROD files which are only about a factor 1.2 smaller.

The reconstruction speed is slightly higher in the ISF sample due to the lower number of particles that are being simulated by the fast simulators. The resulting lower number of sensitive detector hits and raw data objects (from detector digitization) correspond to a smaller input size to the reconstruction algorithms. As a result, a slightly smaller reconstruction output is generated in the ISF sample

The processing speed of the ESD to AOD file format conversion does not seem affected by the smaller size of the input ESD file. No significant speedup was expected

Format	Geant4 Full Simulation (KB/event)	ISF Fast Simulation (KB/event)	G4/ISF
HITS	674	247	2.7
Merged HITS	675	245	2.8
RDO	1455	1234	1.2
ESD	715	671	1.1
AOD	137	132	1.0
SMWZ D3PD	77	76	1.0
Total	3733	2605	1.4
Total of long-term storage formats (HITS+AOD+D3PD)	889	453	2.0

Table 12.6: The average disk file size (in kilobytes) per stored event for ISF and Geant4 simulation and for different ATLAS MC production file formats. Considering only the data formats which are kept for long-term storage on the ATLAS computing grid, a significant reduction in file size is achieved with ISF fast simulation.

in this step, as it mostly concerned with the filtering and combination of information already present in various StoreGate collections of the ESD data format. The same physics objects are expected to be present in the AOD files of both samples. This is because fast simulators are optimized to generate the same output as Geant4 (or a close resemblance of it) at the analysis-level.

A slightly higher execution speed is observed during the SMWZ D3PD file format creation. This is likely caused by a smaller size of the MC truth representation in the fast simulation.

In both simulation samples, the total processing time is dominated by the respective detector simulation step. As this step is significantly faster in the ISF fast simulation flavour (by about a factor of 5.0) the entire ISF fast simulation production chain is processed significantly faster in consequence (by about a factor of 4.6).

Considering all data formats which are kept for long-term storage by the ATLAS collaboration (merged HITS + AOD + D3PD) a reduction in file size of about a factor 2.0 is achieved with the ISF fast simulation flavour.

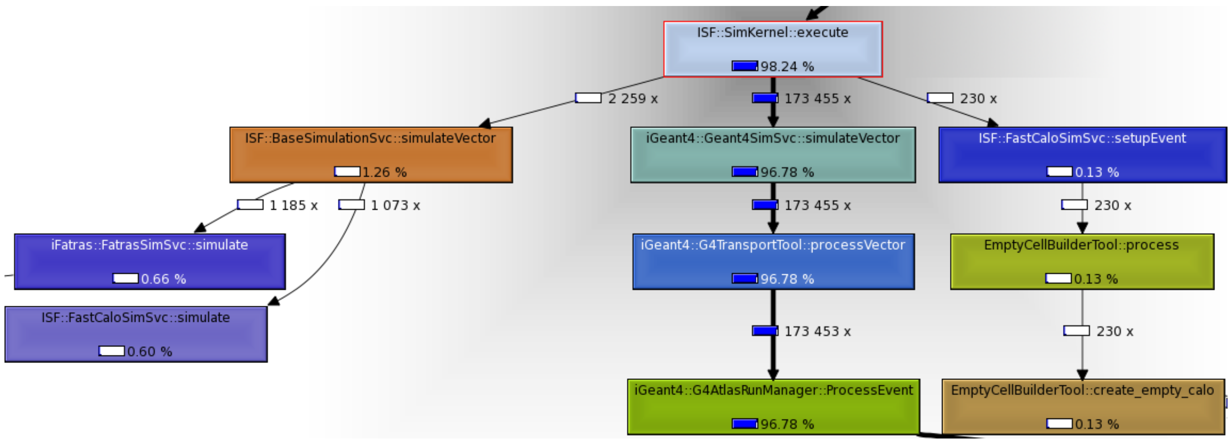

ISF::SimKernel::execute
98.24 %
2 259 x
173 455 x
230 x
ISF::BaseSimulationSvc::simulateVector
1.26 %
iGeant4::Geant4SimSvc::simulateVector
96.78 %
ISF::FastCaloSimSvc::setupEvent
0.13 %
1 185 x
1 073 x
173 455 x
230 x
iFatras::FatrasSimSvc::simulate
0.66 %
iGeant4::G4TransportTool::processVector
96.78 %
EmptyCellBuilderTool::process
0.13 %
ISF::FastCaloSimSvc::simulate
0.60 %
173 453 x
230 x
iGeant4::G4AtlasRunManager::ProcessEvent
96.78 %
EmptyCellBuilderTool::create_empty_calo
0.13 %

Figure 12.18: CPU profile and call tree for ISF $Z \to ee$ fast simulation of 40 events in the ATLAS offline software release `17.7.5.4`. The CPU profile was generated using GPerftools. The percentage values shown in each individual function are the relative CPU time spent inside this function with respect to the CPU time spent inside the entire Athena algorithm execute step. About 96.8% of the CPU time is spent inside Geant4 routines, 0.7% inside Fatras routines and 0.7% inside FastCaloSim routines. The remaining 1.8% of the CPU time are consumed by core components of the Athena framework.

12.6 Discussion

A fast ISF simulation approach was implemented for the detector simulation of $Z \to ee$ events. Cone-shaped regions of interest are created around the individual decay products of the Z boson. Geant4 simulation is used inside these regions, Fatras and FastCaloSim fast simulators are used elsewhere.

Significantly higher execution speeds are achieved with this ISF simulation flavour compared to full Geant4 detector simulation. Smaller simulation output files are generated due to the use of fast simulators. Each individual step in the ATLAS Monte Carlo production chain (with the exception of ESD to AOD file format conversion) executes faster when processing ISF fast detector simulation output compared to Geant4 full detector simulation output. Consequently, the entire ATLAS Monte Carlo production chain executes faster by about a factor of 4.6. The file size of the data formats which are kept for long-term storage by the ATLAS collaboration is reduced by about a factor of 2.0. Both results are outstanding and illustrate that a more efficient use of ATLAS computing resources is enabled due to the use of the ISF.

The simulation accuracy of the fast ISF simulation setup is tested against a Geant4 full simulation reference. Very good consistency is observed in all quantities related to the electrons and positrons from Z boson decays. This indicates that the definition of cone-shaped ROIs (cone size $\Delta R = 0.4$) is sufficient to guarantee an accurate simulation of these signal particles.

A good consistency is observed for all tracking and vertexing variables. This indicates that the Fatras simulator is performing well in comparison with Geant4. Minor differences are observed, which indicate that the sum of all reconstructed track momenta is higher due to the use of the Fatras simulator. These minute differences are of no relevance for the percent-level accuracy required in the fast simulation setup.

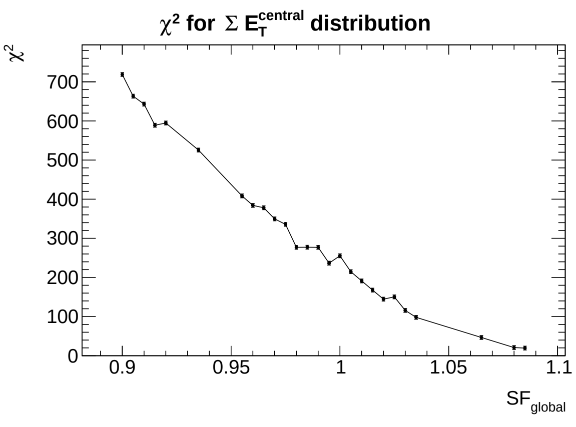
Some differences are observed concerning the simulated energy depositions in the calorimeter. The fast simulator tuning method significantly increases the accuracy of the Fatras and FastCaloSim fast simulators. However, the limitation of FastCaloSim in the forward calorimeter region (Figure 12.13c) is likely to mask the overall lack of simulated calorimeter energy arising from combining Fatras with FastCaloSim simulation (Figure 12.13a and 12.13b). The overestimation of energy in the forward calorimeter (FCAL) shifts the hadronic recoil distribution ($E_{\mathrm{T}}^{\mathrm{hr}}$) in the ISF simulation sample towards higher energies. This distribution is used to tune the global energy scale factor in the fast simulators (SF_{global}). Consequently, the shifted $E_{\mathrm{T}}^{\mathrm{hr}}$ distribution results in an underestimated energy scale factor and therefore in a lack of energy in the central and endcap calorimeter regions. Hence, a revised method must be applied to determine an appropriate value of SF_{global}. The aim is to factor out the impact of the mismodelling in FastCaloSim in the FCAL region.

The following methods are to be studied in order to mitigate this effect:

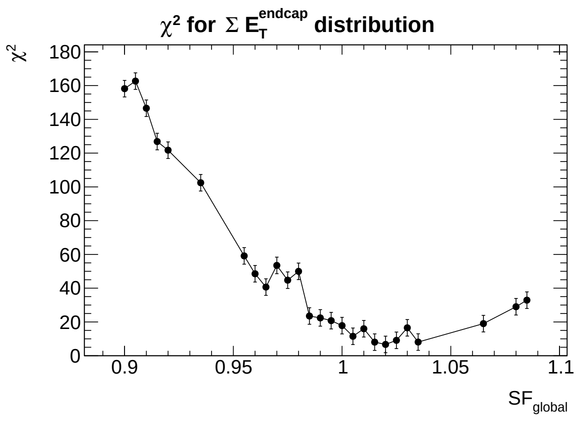
(a-1) The computation of the global energy scale factor SF_{global} is to be based only on the simulation results of the central and endcap calorimeter regions, excluding the forward calorimeter. This will increase the simulation accuracy of the central and endcap regions, but it will not correct the disagreement in the forward region. The high sensitivity of the $E_{\mathrm{T}}^{\mathrm{hr}}$ distribution to the fast simulator tuning makes this a perfect test observable for further tuning (Figure 12.12). Additional event selection criteria must be applied in order to factor out the impact of the mismodelled FCAL region to this observable. Hence, an upper threshold for the energy content in the forward calorimeter may be applied in the tuning process, e.g. $\sum E_{\mathrm{T}}^{\mathrm{forward}} < 1$ GeV.

(a-2) The global energy scale factor is to be split into individual scale factors for different calorimeter regions. Preliminary results reveal that the optimal agreement between ISF fast simulation and Geant4 full simulation is achieved with different values of SF_{global} in different regions of the calorimeter (Figure 12.19). Hence, the $\sum E_{\mathrm{T}}^{\mathrm{central}}$ and $\sum E_{\mathrm{T}}^{\mathrm{endcap}}$ variables can be used to identify the optimal energy scale factor value for each region, respectively.

(b) The simulation of the FCAL volume is to be carried out by Geant4, rather than FastCaloSim. This approach guarantees the most accurate simulation of this detector region. However, it will decrease the simulation speed of the fast ISF simulation flavour. The exact impact on the ISF speedup (Table 12.5) is to be measured.



(a)



(b)

Figure 12.19: First look into χ^2 values for comparing the ISF fast simulation output with full Geant4 simulation in different regions of the detector: (a) scalar sum of the transverse energy in the central calorimeter region (b) scalar sum of the transverse energy in the endcap calorimeter region. 50 000 $Z \to ee$ events are simulated with ISF in each respective sample with a corresponding global energy scale factor (SF_{global}). 110 000 events are simulated in the Geant4 simulation reference sample. The optimal agreement between the fast and full simulation is achieved with different values of SF_{global} in different regions of the detector.

A combination of the methods described above may be employed to achieve an optimal trade-off between simulation speed and accuracy. If the current simulation accuracy in the FCAL volume is determined to be sufficient, only methods (a-1) or (a-2) may be employed in order to increase the simulation accuracy in the central and endcap calorimeter regions. Whereas method (a-2) is expected to provide more accurate simulation results than the application of only method (a-1). If a more accurate simulation of the FCAL is required, a combination of method (b) with (a-1) or (a-2) must be employed. The most accurate simulation results are expected by an implementation of both, method (a-2) and (b).

In either case, an implementation of method (a-2) is recommended. Method (b) is optional and its feasibility is determined by the impact on the detector simulation time, which is to be measured. Independent of whether (b) is implemented, the calibration of the lepton response observable (p_{T}^{ℓ}) seems feasible thus far. The next step is the production of high-statistics validation samples, in order to provide more accurate validation results. This will allow to determine whether the simulation accuracy of the ISF setup is within the required sub-percent level of agreement with respect to full Geant4 simulation for the lepton observables. Without the implementation of (b), a precise detector calibration of the p_{T}^{ν} observable does not seem feasible with the current implementation of FastCaloSim in the fast ISF simulation flavour. The discrepancies in the order of more than 10% between ISF simulation and Geant4 simulation for the calorimeter simulation are larger than the required agreement. However, an ongoing project is concerned with the re-parametrization of FastCaloSim. This new version of FastCaloSim might provide the accuracy required in the FCAL region.

The cone-shaped regions of interest around leptons from Z boson decays have a size of $\Delta R = 0.4$ in the current implementation of the ISF fast simulation flavour. A good accuracy of the lepton isolation variables is achieved with this setting. The impact of different ROI cone sizes on the simulation time and on the simulation accuracy is to be measured. It is expected that smaller cone sizes will increase the simulation speed, but decrease the accuracy of the isolation variables (especially in the calorimeter) and vice versa. The impact on the simulation speed is to be quantified, however, the speed of the current implementation is already a significant improvement over the full Geant4 simulation.

Part IV

Outlook and Conclusions

Chapter 13

Outlook

The Integrated Simulation Framework has proven its success in many ways. It being used for detector simulation in official ATLAS Monte Carlo production campaigns and it has been shown that the framework can significantly reduce the cost of ATLAS MC production. The framework, however, is under constant development and new features are continuously added. The most important updates to the framework and its application in the near future are highlighted in this chapter.

Section 13.1 describes the next steps to be taken in order to further improve the accuracy of the ISF fast simulation flavour in application of the W boson mass analysis. Section 13.2 covers the role of the ISF in the context of a fast ATLAS Monte Carlo production chain which is currently being developed. A prototype for pile-up simulation in the ISF does exist and will be finalized to allow its use in production (Section 13.3). The current implementation of the ISF is a serial algorithm, processing one particle at a time. The modular design of the framework allows to upgrade individual components to support concurrent processing (Section 13.4). A possible update to the framework in order to simplify the integration of the individual simulators into the framework, is covered in Section 13.5.

13.1 Improved ISF Fast Simulation for the W Boson Mass Analysis

The dedicated ISF fast simulation flavour for the detector simulation of $Z \rightarrow ee$ events has shown the full potential of the framework (Chapter 12). Significantly higher simulation speeds are achieved with a simulation accuracy at the same level as full Geant4 simulation for the crucial particles in the even.

However, the accuracy of the fast calorimeter simulation is limited in the forward calorimeter region (FCAL). This intrinsic limitation of FastCaloSim can be overcome through various approaches discussed in detail in Section 12.6. In the most accurate approach, dedicated regions of interest are defined in the detector simulation. These ROIs will the cover regions in the detector within which the fast simulators are least accurate. A functionality intrinsic to the ISF, the routing chain, is used to define these ROIs. Accurate Geant4 simulation is to be used inside these regions in order to mask the localized limitations of the fast simulators. The impact of this approach to the simulation speed and the accuracy is to be measured.

13.2 ISF and the ATLAS Fast Simulation Chain

The current ATLAS Monte Carlo production chain (Section 4.2) requires a number of steps to be executed for the creation of MC samples. Fast implementations of each individual step are currently being developed and validated. Fast ATLAS detector simulation is enabled with the Integrated Simulation Framework. Fast digitization and fast reconstruction algorithms are about to be deployed for production use. With the increase in speed of the respective components in the MC chain, the impact of the input and output processing (i.e. disk operations) to the overall execution speed becomes increasingly significant. Thus, a fast ATLAS MC production chain is currently being developed by the ATLAS collaboration. The aim is to merge the detector simulation, digitization and reconstruction steps into one production step (Figure 13.1). The generator output (EVNT format) serves as input to the fast chain, from which it will directly generate a ROOT-readable data format (xAOD) as the output. To minimize the computational cost of initialization and finalization of Athena framework components, the ultimate goal is to combine all steps into one Athena job or process.

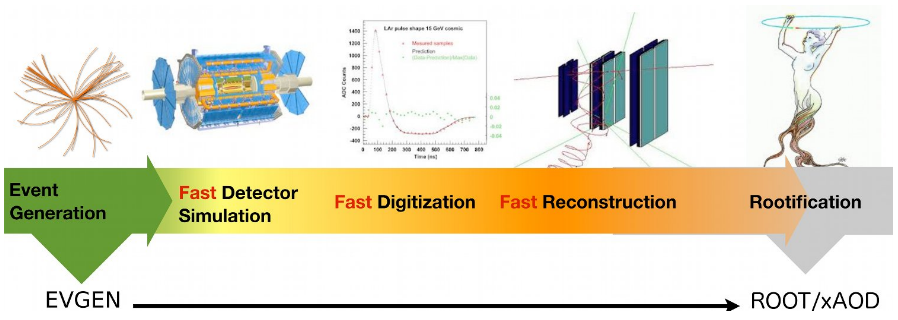


Figure 13.1: The fast ATLAS Monte Carlo production chain. From event generator output (EVNT) to ROOT-readable analysis formats (xAOD) in one step. (image: A. Salzburger).

The role of the ISF in the fast MC production chain is to serve as a framework for fast ATLAS detector simulation. Due to the flexibility of the ISF, fast ATLAS detector simulators can be combined to form an ATLAS detector simulation setup specifically for the needs of subsequent studies and physics analyses. In addition to the simulators currently integrated into the ISF, new fully parameterized simulators may be added. The very high execution speed of such simulators greatly decreases the simulation time required by the detector simulation. Thus, a realistic ISF setup to be applied in the fast

MC chain may consist of a combination of fast and parameterized (very fast) simulators.

The fast simulation chain aims for sufficiently high execution speed in order to allow for private, medium-size Monte Carlo production by individual working groups or institutes. It can be used as a tool to improve the respective analyses prior to requesting high-statistics official Monte Carlo production samples. The resulting requests for official MC production may therefore be better targeted and thus result in a more efficient use of the computing resources.

13.3 ISF Pileup Simulation

Accurate simulation of pile-up effects in the ATLAS detector is essential for physics analyses. In the current ATLAS Monte Carlo production chain (Section 4.2), pile-up events are merged into the detector simulation output of the signal event during the digitization step. The pile-up events are pre-simulated Geant4 detector simulation samples of minimum bias events, which are available in corresponding datasets on the ATLAS computing grid.

This pile-up simulation scheme causes a significant amount of data transfer between grid sites. The datasets must be transferred to all sites on the ATLAS computing grid, which are involved in ATLAS Monte Carlo production. Thus, for fast detector simulation, and in particular for the fast simulation chain covered in the previous section, a faster scheme is to be implemented.

Using the intrinsic flexibility of the ISF particle routing algorithm, the detector simulation of the signal event and of the pile-up even can be carried out within the same detector simulation step inside the ISF. For this, the event generator output (EVNT) of the signal event and the pile-up event(s) are merged into one common simulation input. ISF routing rules can be implemented to select all pile-up particles in the event for fast detector simulation. Due to their high execution speed and accuracy, Fatras and FastCaloSim are ideal simulators for pile-up particles in the ISF. However, not all components of the ATLAS detector have the same sensitivity to pile-up particles. Thus, a second (specifically configured) Fatras instance is used to *transport* particles through regions in the ATLAS detector which are not sensitive to the respective pile-up particles. This Fatras instance will not generate sensitive detector hits for the particles it processes. It is only used to compute the entry point of the particles into regions of the detector which are sensitive to the respective particles. With this combination of Fatras and FastCaloSim simulators, all pile-up particles in the simulated event can be processed through the ATLAS detector and create detector hits in the respective sensitive regions. The signal event is simulated according to the ISF routing rules which are reflecting requirements of the particular physics analysis (full simulation, fast simulation or a combination of both).

A first prototype for pile-up simulation within the ISF is implemented. However, it is not complete yet and further effort is required to extend the current implementation and validate it for production use.

13.4 Concurrent Processing with ISF

Current developments in the CPU market highlight a clear trend towards an increasing number of processor cores per machine – often referred to as the "many-core era". This fundamental change in the computing architecture requires adaptions to the software in order to efficiently use these computing resources. The main challenge for complex applications, like in high energy physics, is the decreasing amount of memory available per processor core due to this development. Thus, various concurrent processing methods are currently being studied for their feasibility in high energy physics [115]. Some studies have resulted in concrete implementations which are used in production.

The ATLAS detector simulation in its current implementation does not directly support concurrent processing within Athena jobs. Individual simulators, such as Geant4, may support concurrent processing, however, this functionality is not exploited for ATLAS detector simulation so far.

The design of the Integrated Simulation Framework allows for future adaptations of the ATLAS detector simulation towards concurrent processing. The individual modules forming the ISF are exchangeable, and specific implementations for concurrent processing may replace some of the currently used components.

Two different types of concurrent processing are feasible with the ISF:

Event-level parallelism : With this type of parallelism, a number of Athena events are processed in parallel. This level of concurrency is steered by the Athena framework and thus, at a higher level than the ISF. Therefore the ISF will most likely not require adaptations for the use of event-level parallelism. A multi-process Athena framework (AthenaMP) has recently been validated for production use by the ATLAS collaboration [116].

Particle-level parallelism : With this type of parallelism, multiple particles from within one event are be processed in parallel. In contrast to the event-level parallelism, the particle-level parallelism is directly enabled due to the ISF but requires adaptations to the framework and simulators. Two different methods seem feasible to realize particle-level parallelism within the ISF. In the first method, each simulator will create multiple threads to process the particles which are assigned to it in parallel. In an alternative approach, the ISF will create multiple threads of each simulator and send sets of particles to the individual threads. In either case, in particular core ISF components (such as TruthService and ParticleBroker) require adaptations in order to accommodate a thread-safe behaviour, as these might be called by multiple threads simultaneously.

The implementation of either of the two methods described above, will enable ATLAS detector simulation to efficiently use many-core computing platforms.

13.5 Simulator Integration and Particle Routing

In the current version of the ISF, a number of interfaces between core ISF components and the simulators are implemented (Chapters 7, 8 and 9). There are two interfaces concerning the flow of information from the simulators into the core ISF components, which the simulators must adhere to. The first one concerns the registration of all incidents where secondary particles are created in a simulator to the central ISF TruthService. Through the second interface simulators are required to return all particles which are passing a sub-detector boundary to the ISF ParticleBroker. These two interfaces function independently, however, a simulator is required to adhere to both, in order for a consistent particle routing and Monte Carlo truth recording within the ISF.

In order to simplify the requirements to simulators, one interface can be formed, which combines the two individual interfaces described above. Simulators would be required to inform the core ISF components through this common interface about incidents where secondary particles are created inside the simulator and when particles cross a sub-detector boundary. Either of these occurrences will be registered to the ISF through a ParticleIncident data format. Similar to the TruthIncident data format, the ParticleIncident format acts a wrapper to allow the core ISF components to access information which is otherwise only available in a simulator-specific data type.

Upon registration of a ParticleIncident, the ISF first creates a Monte Carlo truth record if applicable. Second, if the particle(s) in the ParticleIncident are located on a sub-detector boundary, the ISF will execute the routing algorithm (Chapter 6) to determine the next simulator for the given particle(s). If the next simulator is determined to be the same as the current one generating the ParticleIncident, the ParticleIncident format is used to return information to this simulator, such that the particle ownership stays within this simulator. This will reduce the computational overhead compared to the current implementation, in cases where the simulator does not change for a given particle when crossing from one sub-detector into an other.

Apart from the improved computing efficiency, this implementation will enable new types of routing decisions and particle handovers between simulators. Each registered ParticleIncident may be tested whether it contains relevant particles for subsequent simulators. For instance in the ATLFASTII setup, all ParticleIncidents arising from inner detector simulation can be checked for potential input particles to subsequent FastCaloSim calorimeter simulation. If such a particle is identified, it will be registered for simulation with FastCaloSim and the particle in the ParticleIncident will only be processed until the next detector boundary. The information present to FastCaloSim in this example, is the particle state during its inner detector simulation. This enables energy-level handovers between simulators, in addition to the geometrical handover at sub-detector boundaries. Studies indicate that the accuracy of ATLFASTII simulation will potentially increase with the implementation of this method.

Chapter 14

Conclusions

Monte Carlo detector simulation is an important tool in high energy physics which is used extensively by the ATLAS collaboration in many different areas. It is an essential component to measure and optimize detector performance, improve the significance and accuracy of physics analyses as well to study and develop detector upgrades. Due to its importance, the demand for ATLAS Monte Carlo simulation is higher than what can be generated in a timely manner with the computing resources available through the ATLAS computing grid (Chapters 1 and 3).

The development of the Integrated Simulation Framework is driven by the aim to significantly speed up ATLAS detector simulation (Chapter 4), in order to overcome these limitations and allow for the generation of larger Monte Carlo simulation samples using the same computing resources.

The ISF design and implementation covered in part II of this thesis reaches and even exceeds this goal. The ISF is designed to serve as common framework for all ATLAS detector simulation, while guaranteeing a straight-forward integration into the existing ATLAS Monte Carlo production chain (Chapter 5). The framework has therefore become the standard detector simulation framework for official ATLAS Monte Carlo production campaigns.

The most innovative component of the framework is the ISF routing chain. The routing chain is a novel particle routing algorithm which forms the very core of the framework (Chapter 6). It allows to allocate each particle in the detector simulation to a simulator which is best suited for it. This is enabled due to regions of interest which can be defined specifically for the requirements of performance and physics analyses. Different types of routing rules can be used to define such ROIs. Static routing rules define geometrical ROIs in the ATLAS detector volume or ROIs of kinematic properties of the particles being processed. Semi-dynamic routing rules can define ROIs depending on the particles present in the event generator output. The ISF routing chain algorithm supports either type of routing rules and allows for combinations of both.

Due to the modular design of the framework (Chapters 7 and 8) most of the ATLAS detector simulators existing today are successfully integrated into the ISF (Chapter 9). This enables the ISF to support all traditional full and fast ATLAS detector simulation flavors that existed prior to the implementation of the framework: full Geant4, ATLFASTII and ATLFASTIIF detector simulation. In particular the setup of ATLFASTII has become simpler within the common ISF framework, as it removes one step in the MC production chain which was previously required to execute FastCaloSim.

With the focus on high simulation speed, the ISF core components are designed to introduce very little computational cost in any detector simulation setup. The measurements and CPU profiles presented in Chapter 10 reveal that the computational cost of the core ISF components is less than 1% of the job execution time. This overhead induced due to the framework is negligible compared to the cost of full and even fast detector simulators. In effect, only the Geant4 full detector simulation time slightly increased due to the ISF. All other simulation flavours either had complex and time consuming job setups (ATLFASTII) or were not feasible prior to the implementation of the ISF.

The power of the ISF arises from its unique routing chain algorithm. It enables completely new simulation techniques which were not possible in any preceding ATLAS detector simulation framework. The following new simulation approaches are available to the ATLAS collaboration due to the introduction of the ISF:

Simulator mixing : Full and fast detector simulators can be mixed on a particle-level. This technique is used to form dedicated simulation setups for individual physics analyses with demands for high statistics and high accuracy simulation samples. Accurate simulators (i.e. Geant4) are applied only in regions of interest around crucial particles and fast simulators are applied to the rest of the particles. This increases detector simulation speed significantly, while maintaining the highest possible accuracy for the particles which are relevant for a physics analysis. Chapter 12 of this thesis covers the optimization and validation of this type of ISF simulation, which is developed specifically for $Z \to ee$ detector simulation and the W boson mass measurement.

Partial event simulation : Following the same principles as the simulator mixing approach, the ISF can be configured to only simulate the relevant part of an event and skip all other particles in the detector simulation. This approach provides greatly improved simulation speed (Chapter 10). However, partial event simulation must be allowed within the requirements of a physics analysis, as per definition, it produces strong biases in the simulation output.

The study presented in Chapter 11 indicates that combinations of more than one simulator in the event can decrease the accuracy of the overall detector simulation. The accuracy may be increased, however, if simulators take into account the *history* of a simulated particle. That is to say, a simulator may process a particle differently, depending on the preceding simulator(s) this particle has been processed through. The FastCaloSim detector simulator is designed to process particles which were previously simulated by Geant4 through the inner detector (ATLFASTII). If the Geant4 inner detector simulation were replaced by Fatras (ATLFASTIIF), the accuracy of the calorimeter simulation increases, if FastCaloSim does not adjust to the new type of input. Simulator tuning procedures can be applied to increase the accuracy in these cases.

High simulation accuracy and speed is required in an extensive study presented in Chapter 12 of this thesis. In order to maximize the accuracy of the mass measurement of the W boson, highly accurate $Z \to ee$ detector simulation events are required with high statistics for detector calibration. The full flexibility of the ISF routing chain is employed to form a mixed detector simulation setup, using Geant4, Fatras and FastCaloSim. Semi-dynamic routing rules are used to define cone-shaped regions of interest around the crucial particles (electron and positron) in each event. All primary particles inside the ROIs and all their secondaries are simulated with the most accurate simulator, Geant4. The remaining particles are simulated with the fast simulators Fatras in the inner detector and FastCaloSim in the calorimeter. A dedicated FastCaloSim tuning method was developed and applied. It is found that this method significantly increases the accuracy of the Fatras and FastCaloSim fast simulator combination.

The flexibility of the ISF can be used to further increase the accuracy of the $Z \to ee$ fast simulation flavour. This is achieved by defining ROIs in regions where fast simulators are known to have deficiencies and replace them locally by the Geant4 simulator (e.g. the forward calorimeter region).

The results in Chapter 12 show that a 4.6 times faster execution of the entire MC production chain is possible due to the use of the ISF, while maintaining the same accuracy as full Geant4 detector simulation for the critical observables in the event. The output dataset size (of the long-term stored formats) is smaller by a factor of 2.0, which further reduces the MC production costs for the ISF samples.

Based on these results, the author is confident that the ISF will contribute significantly to the high accuracy achieved by future physics analyses with the ATLAS experiment, in particular the W boson mass measurement.

Part V

Appendix

Appendix A

ISF Core Implementation

An overview of the functionality of the framework and its components was provided in previous Chapters 5, 6, 7 and 8. This chapter covers the implementation details of the various components of the Integrated Simulation Framework. The implementation of the ISF is constantly being improved and is therefore subject to change. The documentation provided in this chapter refers to the implementation present in the ATLAS offline software release `17.7.5.4`. Specific details may differ in other releases.

A.1 ISFParticle

Class Summary	
Class Name	`ISF::ISFParticle`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event
Class Name	`ISF::ISFParticleContainer`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event
Class Name	`ISF::ISFParticleOrderedQueue`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event
Class Name	`ISF::ISFParticleVector`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event

The `ISFParticle` is the C++ type used to describe a particle in the ISF. It contains a set of fundamental properties that are required to describe any physical particle in a detector simulation, these are: particle type, mass, charge, global position, momentum vector and a time stamp. In addition, an ISF particle contains information that relates to particle routing, such as identifiers for the originating, previous and next simulator and detector region.

This data type is used for any transfer of information regarding particles within the ISF or between the ISF and a simulator. Particles which are sent to a simulator for simulation, are provided in the `ISFParticle` data format. Secondary particles generated by a simulator and returned to the ISF, are provided in the same format by a simulator to the ISF ParticleBroker.

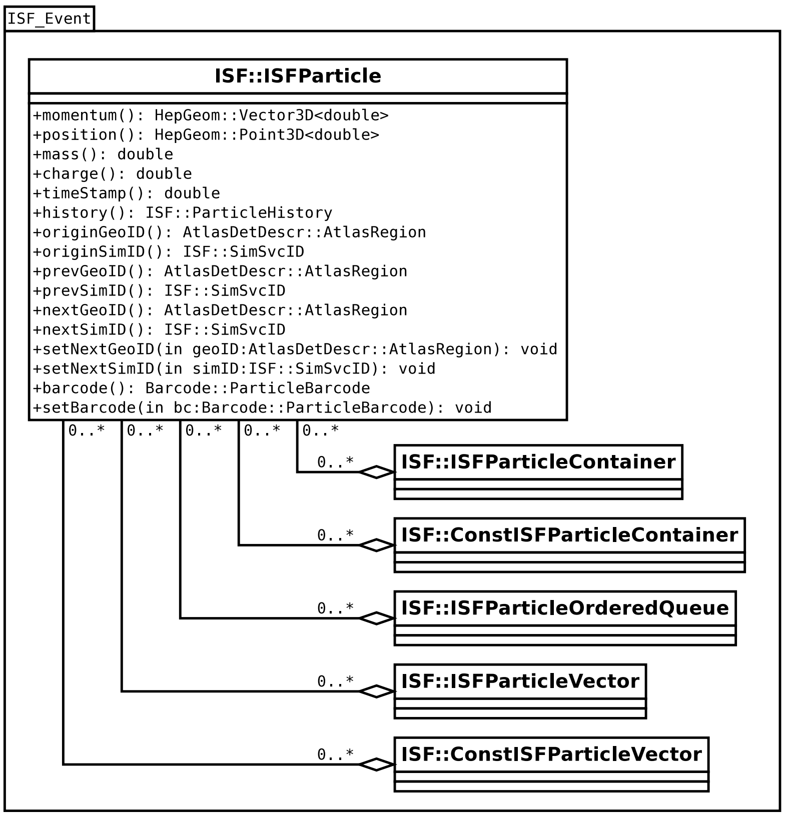

ISF_Event
ISF::ISFParticle
+momentum(): HepGeom::Vector3D<double>
+position(): HepGeom::Point3D<double>
+mass(): double
+charge(): double
+timeStamp(): double
+history(): ISF::ParticleHistory
+originGeoID(): AtlasDetDescr::AtlasRegion
+originSimID(): ISF::SimSvcID
+prevGeoID(): AtlasDetDescr::AtlasRegion
+prevSimID(): ISF::SimSvcID
+nextGeoID(): AtlasDetDescr::AtlasRegion
+nextSimID(): ISF::SimSvcID
+setNextGeoID(in geoID:AtlasDetDescr::AtlasRegion): void
+setNextSimID(in simID:ISF::SimSvcID): void
+barcode(): Barcode::ParticleBarcode
+setBarcode(in bc:Barcode::ParticleBarcode): void
0..*
ISF::ISFParticleContainer
ISF::ConstISFParticleContainer
ISF::ISFParticleOrderedQueue
ISF::ISFParticleVector
ISF::ConstISFParticleVector

Figure A.1: The UML class diagram of the `ISFParticle` C++ class. The `ISFParticle` is the fundamental data type used to transport information regarding physical particles within the ISF.

A number of predefined STL container [117] data types exist for the `ISFParticle` type.

A.2 Simulation Kernel

Class Summary	
`AthAlgorithm` Name	`ISF::SimKernel`
Containing Package	Simulation/ISF/ISF_Core/ISF_Algorithms

The ISF SimulationKernel is the sole `AthAlgorithm` implementation in the ISF, and as such, it takes a central role in the operation of the simulation framework. Its responsibilities range from event initialization, managing the particle loop, performance measurements to event finalization. Figure 8.1 illustrates that the SimulationKernel acts as a central hub for receiving particles from the ParticleBroker and sending them to the individual simulators.

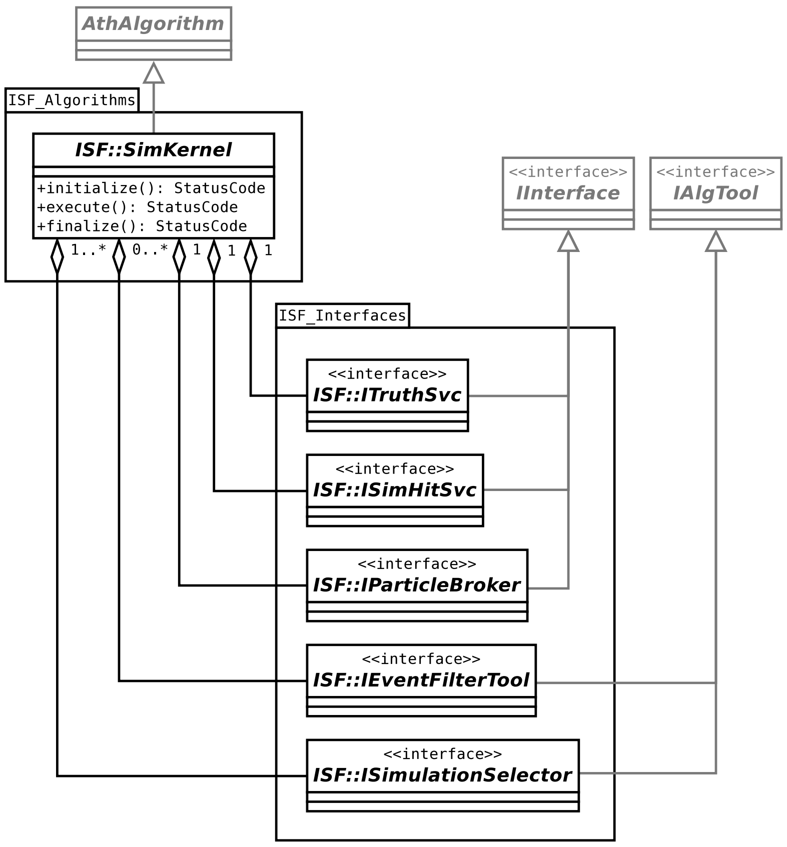


Figure A.2: The UML class diagram of the ISF SimulationKernel including the Athena tools and Athena services it uses.

Figure A.2 shows the UML class diagram of the ISF SimulationKernel implementation. Like any Athena algorithm, the `ISF::SimKernel` implements the three main `AthAlgorithm` methods:

`initialize()` retrieves all `AthAlgTools` and `AthServices` that will be used by the ISF SimulationKernel throughout the execution of the current Athena process. The kernel will forward the SimulationSelectors (Section A.3.1) to the ParticleBroker (Section A.3), as the routing chains are configured with the ISF kernel. The number of simulators present in an ISF job depends on the specific configuration of the framework. During the initialization phase of the SimulationKernel, each registered simulator is dynamically assigned with a unique identifier of type `ISF::SimSvcID`.

Internal to the ISF, each simulator is represented by a unique identifier, which is valid until the end of the current ISF process.

`execute()` is the main method in the ISF SimulationKernel. It initializes the ParticleBroker (Section A.3), the Monte Carlo TruthService (Section A.6), and all simulation services (Section A.4) at the beginning of the event before the event loop commences. In the event loop, the SimulationKernel queries the ParticleBroker for particles which are to be simulated. The kernel retrieves these particles and sends them to the corresponding simulators. The simulators may return secondary particles to the ISF, however, they are not handed back to the SimulationKernel, these particles are directly sent to the ParticleBroker (Section A.4.1 for simulation service requirements). The particle loop ends, if the ParticleBroker does not return any more particles that need to be simulated. At this point, all particles will have been completely simulated through the ISF and the kernel executes the event finalization steps. At this stage, the same services that were previously called for the event initialization are called for event finalization. In addition, event filter tools (Section A.2.1) are evaluated, which determine whether or not the output of the current Athena event will be persistently recorded. For collision event simulation, usually all events are recorded and no event filters are configured. However, in the case of cosmic event simulation, the event filter tools will reduce the simulation output to only a subset of all simulated events which fulfill certain criteria.

`finalize()` collects and prints performance measurements that are collected throughout the execution of the ISF simulation job. In particular CPU time per configured simulator may be of interest to the user.

The `ISF::SimKernel` implementation is used for all flavours of ATLAS detector simulation. This includes full and fast simulation approaches, as well as simulation of cosmic ray events. Each ATLAS detector simulation flavour is simulated with a specifically configured `ISF::SimKernel` instance.

The current `ISF::SimKernel` implementation does not support multi-threaded detector simulation. However, due to the modular design of the ISF, a separate ISF SimulationKernel implementation may be created if multi-threading approaches are to be used.

A.2.1 Event Filter Tool

Class Summary	
Interface Name	`ISF::IEventFilterTool`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::CosmicEventFilterTool`
Containing Package	Simulation/ISF/ISF_Core/ISF_Tools

An event filter tool is an AthAlgTool implementation of the ISF::IEventFilterTool interface. Event filter tools are used to determine whether or not a given Athena event is to be stored persistently.

Figure A.3 shows the UML class diagram of the event filter tool interface and implementation. The ISF SimulationKernel does not use event filter tools for collision event simulation. For cosmic event simulation, however, the ISF::CosmicEventFilterTool is queued by the ISF SimulationKernel in every Athena event. The ISF::CosmicEventFilterTool determines whether particles entered the region of the inner detector during the simulation of the current event. This is done by processing the entries of the CaloEntry TrackRecordCollection (Section A.6.6). If no entries are present in the CaloEntry collection, the simulation output of the current Athena event will not be stored persistently.

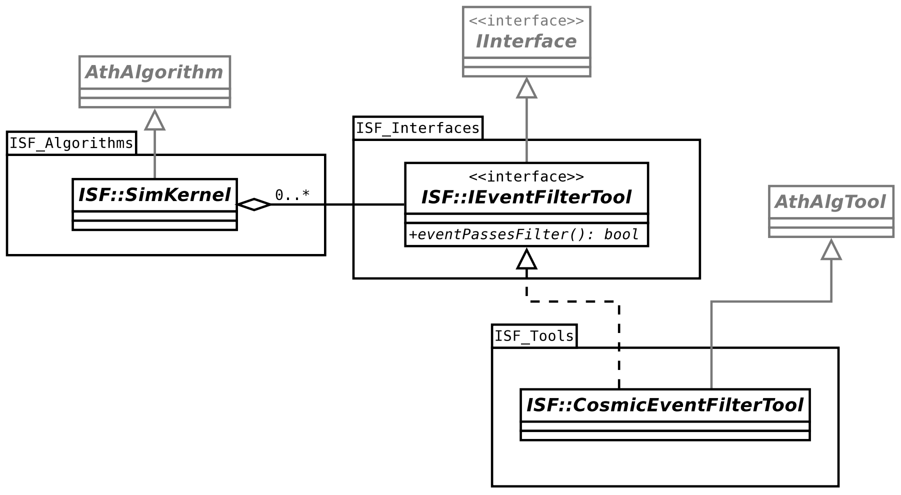


Figure A.3: The UML class diagram of the event filter tool interface and implementation. The ISF::CosmicEventFilterTool is used in ISF cosmic simulation to determine whether the simulation output of a given Athena event will be persistently stored.

A.3 Particle Broker

Class Summary	
Interface Name	ISF::IParticleBroker
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	ISF::ParticleBrokerDynamicOnReadIn
Containing Package	Simulation/ISF/ISF_Core/ISF_Services

The ISF ParticleBroker is an `AthService` handling all `ISFParticles` which are processed within the simulation framework. It is central to the ISF due to its role in assigning simulation engines to individual `ISFParticles`. The ISF ParticleBroker also initiates the preparation of the simulation input by calling a `ISF::IStackFiller::fillStack(..)` method before the execution of the particle loop (Section A.6.1 for a detailed description of the `IStackFiller` interface and its implementation).

The ParticleBroker is accessible via the `ISF::IParticleBroker` interface and currently the `ISF::ParticleBrokerDynamicOnReadIn` is the only implementation of this interface. This ParticleBroker implements the routing algorithm discussed in Section 6.2, which supports static and semi-dynamic routing rules. The ParticleBroker determines the appropriate simulation engine for each `ISFParticle` it receives. Simulation engines may send secondary particles (or particles at sub-detector boundaries) to the ParticleBroker via the `ISF::IParticleBroker::push(..)` interface method. Particles read from the simulation input are also processed through the ParticleBroker. Figure A.4 shows the class diagram of the ISF ParticleBroker interface and implementation.

In order to determine the simulation engine of a given ISF particle, the `ParticleBrokerDynamicOnReadIn` implementation applies a routing chain with semi-dynamic routing rules (Section 6.1.2). The simulator decision made by the routing chain is unique, which is a requirement for a reproducible simulation framework behaviour.

The "joints" in the routing chain correspond to the SimulationSelectors which are implementations of the `ISF::ISimulationSelector` interface (see the following subsection for more details on the implementation of the SimulationSelectors). The ParticleBroker implementation holds a chain of SimulationSelectors for each ATLAS detector region (i.e. for each sub-detector) respectively. To determine the simulation engine for a given `ISFParticle`, the ParticleBroker first uses the GeoIDService (Section A.5.2) to determine the `AtlasDetDescr::AtlasRegion` (Section A.5) within which the next simulation step for this particle will be performed. Subsequently, it calls the `ISF:ISimulationSelector::selfSelect(..)` method of the SimulationSelectors present in the routing chain for the current sub-detector. The simulation engine is then determined as the simulation engine attached to the first SimulationSelector in the corresponding chain which returns `true` for this call.

The SimulationSelectors implementing semi-dynamic routing rules must be provided with the list of initial particles in the detector simulation. Hence, the `ParticleBrokerDynamicOnReadIn` informs all SimulationSelectors of the particles present in the input of the current Athena event. This is carried out immediately after the initial list of `ISFParticles` is prepared by StackFiller, before the start of the event loop and even before the simulation engine is determined for these particles. Updating the SimulationSelectors is carried out by passing each input `ISFParticle` to each registered SimulationSelector via the `ISF::ISimulationSelector::update(..)` method.

Particles that are sent to the `ParticleBrokerDynamicOnReadIn` via the `IParticleBroker::push(..)` method, are processed by the ParticleBroker in the following order:

1. The `ISFParticle` is registered with the EntryLayer tool by calling

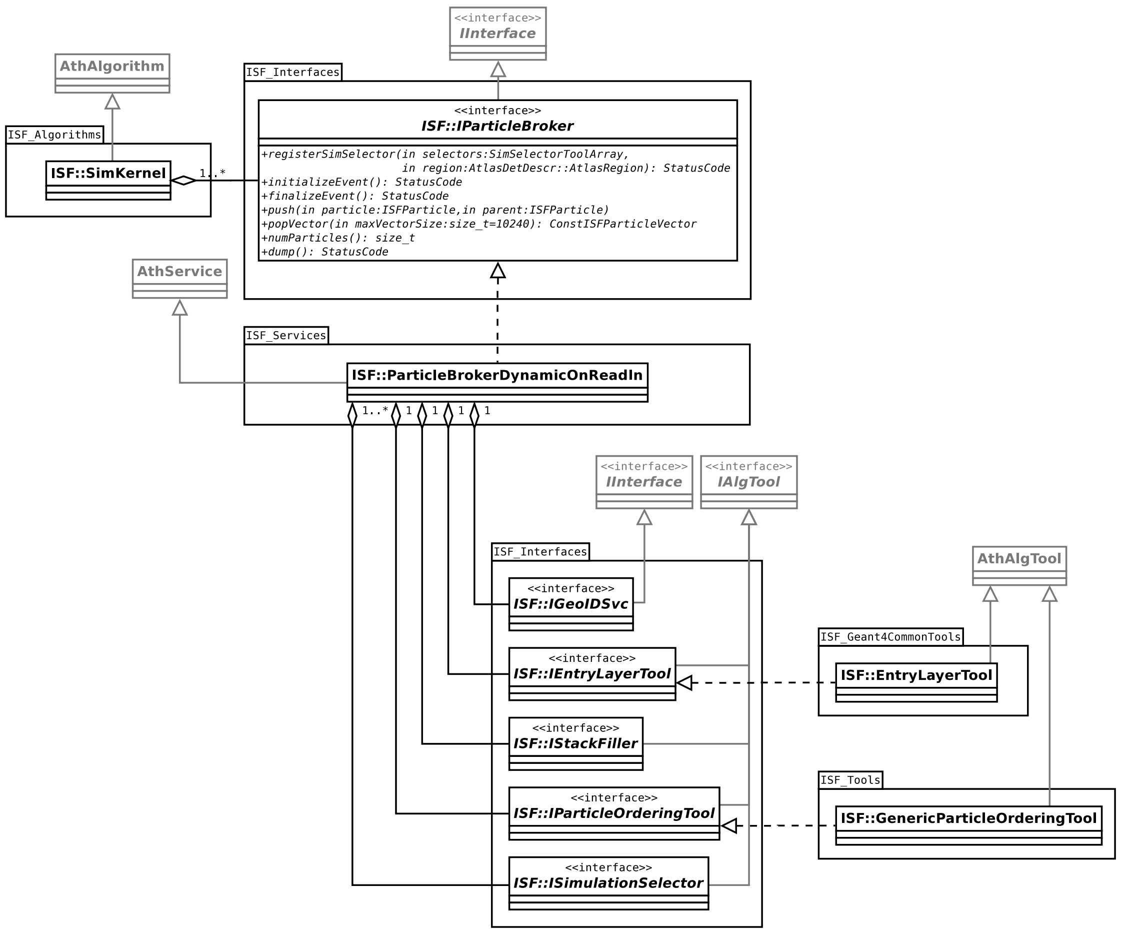


Figure A.4: The UML class diagram of the ISF ParticleBroker interface, its implementation and the tools and services it uses.

`ISF::IEntryLayerTool::registerParticle(..)` (Section A.6.6). If the particle resides on the boundary between two ATLAS sub-detectors, the EntryLayer tool will generate a record in the respective EntryLayer collection.

2. To determine the ATLAS region of an `ISFParticle`, the ParticleBroker reads the region stored in the particle by calling `ISFParticle::nextGeoID()`. If this returns an unset region (`AtlasDetDescr::fUndefinedAtlasRegion`) or if the particle resides on a boundary between two ATLAS detector regions (determined by the EntryLayer tool), the ParticleBroker will use an `ISF::IGeoIDSvc` implementation to resolve the detector region. This is done by calling `IGeoIDSvc::identifyNextGeoID(..)`. Thus, if a simulator returns a particle to the ISF ParticleBroker, and the simulator knows with certainty which detector

region the particle is in, it can the use `ISFParticle::setNextGeoID(..)` method to set the ATLAS region that the particle will be simulated in next.

3. The previously determined ATLAS detector region, is used to access the appropriate routing chain in the ISF ParticleBroker, since the ParticleBroker holds one routing chain per ATLAS detector region.

4. To determine the appropriate simulator for the given particle, the ParticleBroker uses the previously discussed routing algorithm in this routing chain.

5. The ParticleBroker uses a particle ordering tool implementation to set the ordering number of the given particle. Particle ordering numbers are equivalent to the priority with which a particle will be simulated in the single-threaded (sequential) ISF implementation. This is done by calling `IParticleOrderingTool::setOrder(..)`. The ordering tool must generate the same ordering numbers for all particles that are assigned to a common simulator. Upon request by the ISF SimulationKernel, the ParticleBroker will later return all particles at once which share the highest ordering number. These particles will subsequently be sent to the same simulator at once.

6. The particle is added to the `ISFParticleOrderedQueue` which is internal to the ParticleBroker. The `ISFParticleOrderedQueue` is a `std::priority_queue` using the particle ordering number to sort the particles.

A.3.1 Simulation Selector

Class Summary	
Interface Name	`ISF::ISimulationSelector`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::DefaultSimSelector`
Implementation Name	`ISF::ConeSimSelector`
Implementation Name	`ISF::HistorySimSelector`
Implementation Name	`ISF::KinematicSimSelector`
Implementation Name	`ISF::PileupSimSelector`
Implementation Name	`ISF::RandomSimSelector`
Implementation Name	`ISF::TruthAssocSimSelector`
Implementation Name	`ISF::TruthConeSimSelector`
Containing Package	Simulation/ISF/ISF_SimulationSelectors

SimulationSelectors are the ISF equivalent of routing rules, as discussed in Section 6.1. They are the basic building blocks of the ISF routing algorithm, the ISF routing chain (Section 6.2). A SimulationSelector is an `AthAlgTool` which inherits from the `ISimulationSelector` interface. Static and semi-dynamic routing rules can be implemented by using the same `ISimulationSelector` interface. In accordance with the routing rule definition, two properties are essential to every SimulationSelector:

Condition C : The condition of a SimulationSelector must be implemented inside the `passSelectorCuts(..)` method of the specific `ISimulationSelector` implementation. This interfaced method is called when the ISF routing algorithm evaluates individual SimulationSelectors (i.e. routing rules) for a given `ISFParticle`. More specifically, the ISF ParticleBroker calls the `ISimulationSelector::selfSelect(..)` method, which calls the `passSelectorCuts(..)` method internally. The `selfSelect(..)` method can optionally invert the Boolean result that is returned by the `passSelectorCuts(..)` method. This allows, for example, to invert regions of interest in the Python configuration, without the need to change the underlying C++ implementation of the SimulationSelector. It is the return value of the `selfSelect(..)` method, which is directly used by the ISF routing algorithm to determine whether a given particle is selected by a SimulationSelector or not. Thus, if the `selfSelect(..)` method returns `true` for a given `ISFParticle`, this means that the routing rule does select this particle and it will subsequently be sent to the simulator S that is attached to the SimulationSelector.

Simulator S : Each SimulationSelector is associated with one simulation service (Section A.4). The same simulation service might be associated with multiple SimulationSelectors. The simulator S of a SimulationSelector is usually defined at the Python configuration stage of the Athena job setup (Section 4.1). The simulator of a SimulationSelector instance must stay constant throughout the execution of a detector simulation job. Upon being called by the ISF routing algorithm, if the `selfSelect(..)` method of a SimulationSelector instance returns `true` for a given `ISFParticle`, the associated simulator S of this SimulationSelector will be used for the detector simulation of this particle.

Figure A.5 shows the class diagram of the `ISimulationSelector` interface and its current implementations. The most important methods declared in the interface, are:

`simulator()` returns a `ToolHandle` to the `ISimulationSvc` (simulator S) that is associated with the given SimulationSelector instance.

`initializeSelector()` is called once by the ISF ParticleBroker at the beginning of the simulation job.

`beginEvent()` is called once before the particle loop commences inside the ISF SimulationKernel.

`endEvent()` is called once, after the particle loop inside the ISF SimulationKernel has ended. This method is recommended to be used for resetting the condition C in semi-dynamic routing rules. This is required as no information about the current event must influence the simulator decisions of the next event.

`update(..)` is used to update the SimulationSelector with the particles that are in the initial particle list. This method allows semi-dynamic routing rules to define the condition C according to the particles that are in the simulation input event. After the ISF ParticleBroker has prepared the particles from the input file, it will call the `ISimulationSelector::update(..)` method of each SimulationSelector for each particle in the input that will be simulated through the detector.

`selfSelect(..)` is called by the ISF routing algorithm, to determine whether a SimulationSelector selects a given particle or not. The `selfSelect(..)` method calls the `passSelectorCuts(..)` method internally, but may optionally invert the return value of the latter.

`passSelectorCuts(..)` is where the condition C of a routing rule must be implemented and evaluated for a given particle. Note that the return value can be inverted by `selfSelect(..)` method.

There are currently eight implementations of the `ISimulationSelector` interface:

The `DefaultSimSelector` is a static routing rule which selects all particles. This selector is usually applied at the end of routing chains, to set a default simulator for the ATLAS detector region that the respective chain is defined in.

The `ConeSimSelector` is a semi-dynamic routing rule, which registers cone-shaped ROIs around defined types of particles in the simulation input event. It uses the `ConeParticleCuts` class to store the cone-shaped ROIs and to evaluate whether a given particle is within any of the registered cones.

The `HistorySimSelector` is a static routing rule which bases its decision on the previous simulator and detector region of the given particle. This selector might be used to select particles that were previously simulated with a certain simulator in a given ATLAS region. For example, the SimulationSelector might be configured to ensure that all particles (and secondaries) that were at some point simulated by Geant4 will be sent to Geant4 for the remainder of the simulation.

The `KinematicSimSelector` is a static routing rule which bases its decision on static and kinematic properties of the given particle. This selector can be used to define regions of interest, based on a particle's momentum direction, position, charge and type. It uses the `KinematicParticleCuts` class to store and test the condition of the routing rule.

The `PileupSimSelector` is a static routing rule which is used in the ISF prototype for combining the simulation of the hard scatter event and pile-up within the detector simulation step (Section 13.3). This SimulationSelector is designed to select all particles from the pile-up part of the event.

The `RandomSimSelector` is a static routing rule which randomly selects 50% of all particles it is evaluated for. This selector is generally used for testing and development purpose.

The `TruthAssocSimSelector` is a static routing rule which checks whether a given `ISFParticle` is linked to a specific particle type in the `TruthEvent`. The `HepMCHelper` class is used to find relatives of the given `ISFParticle` in the `TruthEvent`. The types of relations that are evaluated are a subset of the elements in the `HepMC::IteratorRange` enum [64]: parents, family, ancestors and relatives.

The `TruthConeSimSelector` is a semi-dynamic routing rule, which registers cone-shaped ROIs around specific particles in the `TruthEvent`. This semi-dynamic routing rule accesses the `TruthEvent` StoreGate collection inside the `beginEvent(..)` method in order to define the condition C. It does not process the particles that are provided through the `ISimulationSelector::update(..)` method. This SimulationSelector is applied in a fast simulation setup relevant for the measurement of the W boson mass, covered in Chapter 12.

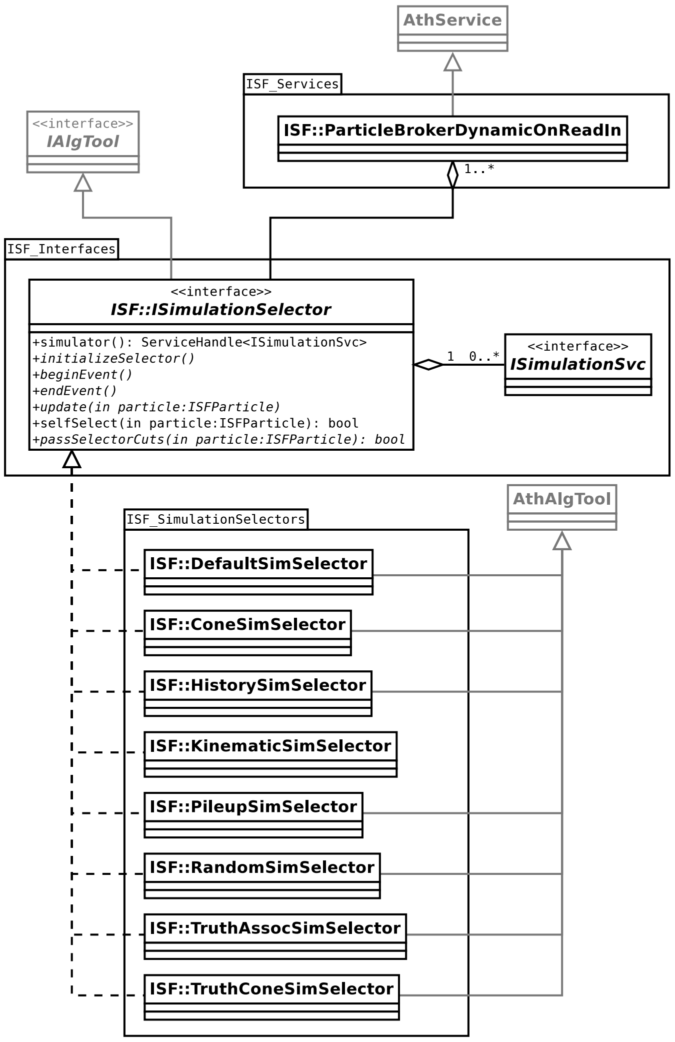


Figure A.5: The UML class diagram of the ISF SimulationSelector interface `ISimulationSelector` and its various implementations. Generally, the ISF ParticleBroker has a number of associations with SimulationSelector instances. Each SimulationSelector has an association with exactly one simulation service. However, a simulation service might be associated with a number of different SimulationSelectors.

A.4 Simulation Services

Class Summary	
Interface Name	`ISF::ISimulationSvc`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::BaseSimulationSvc`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::ParticleKillerSimSvc`
Containing Package	Simulation/ISF/ISF_Core/ISF_Services
Implementation Name	`iGeant4::Geant4SimSvc`
Containing Package	Simulation/ISF/ISF_Geant4/ISF_Geant4Services
Implementation Name	`iFatras::FatrasSimSvc`
Containing Package	Simulation/ISF/ISF_Fatras/ISF_FatrasServices
Implementation Name	`ISF::FastCaloSimSvc`
Containing Package	Simulation/ISF/ISF_FastCaloSim/ISF_FastCaloSimServices

A simulator algorithm fulfilling the necessary requirements (Section 7.1.1 and A.4.1) can be integrated into the ISF. Hence, the ISF is not limited to the simulation engines described in this section.

The following detector simulator implementations are available in the ISF in ATLAS offline software release `17.7.5.4`:

`ISF::BaseSimulationSvc` is the only implementation of the `ISF::ISimulationSvc` interface which does not fulfill the ISF simulator requirements. It adds a number of commonly used functionalities to the otherwise *empty* simulation service interface. The `BaseSimulationSvc` is meant to be used as the actual base class for most simulation service implementations.

`ISF::ParticleKillerSimSvc` is a very simple simulation service. It does not perform any computation with the particles that are provided to it. Thus, any particle sent to the `ParticleKillerSimSvc` will neither produce sensitive detector hits nor secondary particles, nor will the given particle be returned to the ISF ParticleBroker. This equates to inhibiting any further detector simulation of this particle. This service is commonly used if certain detector regions are disabled in the detector simulation, i.e. the simulation of particles through these regions is not required.

`iGeant4::Geant4SimSvc` is the ISF simulation service implementation of the Geant4 toolkit. A detailed description of the Geant4 integration into the ISF is given in Section 9.1.

`iFatras::FatrasSimSvc` is the ISF simulation service implementation of the Fatras fast simulation. A detailed description of the Fatras integration into the ISF is given in Section 9.2.

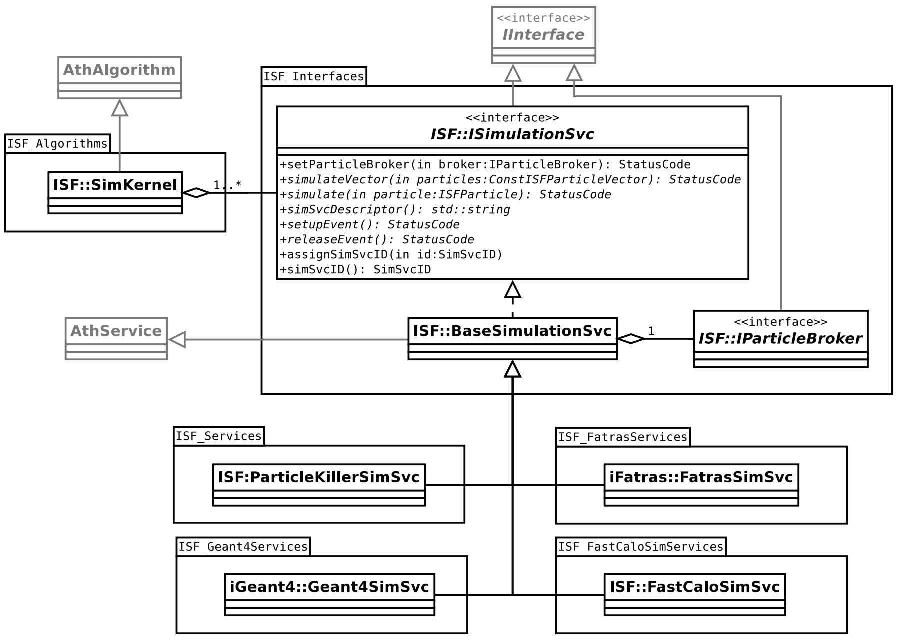


Figure A.6: The class diagram for the `ISF::ISimulationSvc` interface and its implementations. The ISF SimulationKernel accesses the individual simulation engines through `ISF::ISimulationSvc` interface methods. All currently implemented simulators inherit from the generic `BaseSimulationSvc` which implements basic, commonly used functionalities.

`ISF::FastCaloSimSvc` is the ISF simulation service implementation of the FastCaloSim simulation. A detailed description of the FastCaloSim integration into the ISF is given in Section 9.3.

A.4.1 Simulation Service Requirements

This section lists the implementation requirements a simulator must fulfill for the integration into the ISF. This section is an extension of the qualitative list presented earlier in this thesis (Section 7.1.1). As such, it also provides technical details on many of the points mentioned earlier.

Simulation Service Interface

A basic requirement of an ISF simulation service implementation is the inheritance from the `ISF::ISimulationSvc` interface. This interface defines a number of essential methods used by the ISF to send information to and receive information from a particular simulation service implementation:

`setParticleBroker(..)` is called by the ISF kernel to forward the ParticleBroker instance in use by the simulation framework to the simulation service implementation. Since this functionality does generally not depend on the particular simulation service implementation, the `BaseSimulationSvc` implements a generic version of it.

`simulateVector(..)` is called by the ISF kernel to send a set of particles to the simulator implementation. The argument is of type `ConstISFParticleVector` and it contains the `ISFParticles` to be simulated by the simulation service. There is no requirement on the order in which the given particles are to be simulated. Thus, concurrent processing may be used to simulate the particles in the particle vector. However, as mentioned above, the simulator has to be deterministic in its behaviour and provide reproducible results if the same simulation setup is executed multiple times. The simulator must not empty the given `ConstISFParticleVector` nor free the memory of any `ISFParticle` instance given in the vector. The `BaseSimulationSvc` implements a simple version of the `simulateVector(..)` method. It calls the `simulate(..)` method (see below) for each individual `ISFParticle` instance given in the vector of particles.

`simulate(..)` is the single-particle equivalent to the `simulateVector(..)` method described above. The argument is a reference to an `ISFParticle` which is to be processed by the simulation service. This method will not be called by the ISF kernel. However, the `BaseSimulationSvc` may call it if no implementation of the `simulateVector(..)` method is provided in the simulation service implementation. Though, this method is the predecessor of the `simulateVector(..)` method it is still the sole method through which Fatras and FastCaloSim fast simulations receive particles for detector simulation. The reason for this being that the underlying algorithms in these fast simulations are purely sequential and thus any set of particles will need to be processed sequentially.

`simSvcDescriptor()` returns a `std::string` type with a clear text name of the simulation service. The string is used by the ISF to generate messaging output. Thus, it should be concise and allow a user to identify the actual simulation service when encountered in the ISF messaging output. There are no strict requirements for this string to be unique, though for practical reasons each simulator should provide a different string. A few examples of simulation service descriptors currently in use are: ”Geant4”, ”Fatras” and ”FastCaloSim”.

`setupEvent()` is called by the ISF SimulationKernel immediately before the start of the particle loop. The ISF kernel calls this method for each simulation service implementation which is configured in the ISF setup that is being executed. This method can be used by simulators to execute internal event initialization routines.

`releaseEvent()` is called by the ISF SimulationKernel immediately after the end of the particle loop. The ISF kernel calls this method for each simulation service implementation which is configured in the ISF setup that is being executed. This method can be used by simulators to execute internal event finalization routines.

`assignSimSvcID(..)` is called by the ISF SimulationKernel to assign a unique identifier of type `ISF::SimSvcID` to each simulation service registered in the current ISF job. The ID is unique for each simulator within the given ISF process. The very same simulation service implementation may get different IDs in different configurations of the ISF. This method will be called during the initialization phase of the Athena algorithms and it will be called once per simulation service. The simulation service ID is used for ISF internal purpose (e.g. fast array accesses) and thus it must not be modified by the simulation service implementation. Due to these requirements, this method is implemented directly in the `ISF::ISimulationSvc` interface and can not be overloaded in an implementation.

`simSvcID()` is called by core ISF routines to access the identifier of the given simulation service implementation within the current ISF process.

Particle Interactions, Decays and Monte Carlo Truth

Secondary particles resulting from interactions or decays of particles in the detector volume can be passed back to the ISF via the `ISF::IParticleBroker::push(..)` interface method. However, there is no requirement that secondary particles must be returned to the simulation framework. If an interaction which generates additional particles occurs inside a simulator, an `ISF::ITruthIncident` (Section A.6.2) must be created and registered to the ISF TruthService via the `ISF::ITruthSvc::registerTruthIncident(..)` method (Section A.6.3). The ISF TruthService will determine whether the interaction will be written into the Monte Carlo truth output and if necessary it will generate this output.

Figure 8.3 illustrates the information flow if a particle interaction is computed by a simulation service.

Sub-Detector Boundaries

A simulation engine is required to return all particles passing a sub-detector boundary to the ISF via the `ISF::IParticleBroker::push(..)` method. This applies to all primary and secondary particles. When doing so, the affected particles must not be processed any further within the simulation service. The sub-detector boundaries are well defined within the simulation framework and can be retrieved as a list of (r, z) coordinate pairs from the `ISF::ISFEnvelopeDefSvc` (Section A.5.1). In addition, various `ISF::IGeoIDSvc` implementations exist to identify the `AtlasDetDescr::AtlasRegion` of a given `ISFParticle` or a point within the ATLAS detector volume (Section A.5.2).

Charged low energy particles can potentially loop along detector boundaries and may end up being handed back and forth between the ISF and a simulator. Thus, to improve computing performance, particles with very low energies may not be returned to the ISF and can be processed further within the respective simulator. A kinetic energy threshold of 50 keV is applied in the Geant4 simulation service for particles which are to be returned to the ISF ParticleBroker (Section 9.1).

Sending Particles to the ISF Particle Broker

A simulator may send one or many particles to the ISF ParticleBroker via the `ISF::IParticleBroker::push(..)` interface method. Each particle must be instantiated as an `ISFParticle` type beforehand. If available within the simulator, the corresponding parent ISF particle may also be provided to the ParticleBroker interface method.

If the ATLAS region within which a particle is simulated next is known to a simulator, the simulator may assign the corresponding `AtlasDetDescr::AtlasRegion` via the `ISFParticle::setNextGeoID(..)` method. By default, this entry will only be cross-checked by the ISF ParticleBroker if the particle resides on a boundary between two ATLAS sub-detectors. Thus, if the next region is not know with certainty by the simulator, the ATLAS region must be set to `AtlasDetDescr::fUndefinedAtlasRegion`. This is done implicitly by the default `ISFParticle` constructor or it can be done explicitly with the `ISFParticle::setNextGeoID(..)` method. The ParticleBroker will subsequently determine the corresponding ATLAS region if it receives a particle with an unset ATLAS region.

Simulation Output and Sensitive Detector Hits

The output data generated by a simulator must be filled into the respective StoreGate collection directly by the simulator. For example, if a simulator were to generate sensitive detector hits (i.e. energy deposits), the respective SD hit collection must be accessed and filled by this simulator. The relevant StoreGate collections are shared among different simulators. Hence, no one simulator must disallow writing of any one output collection to any of the other simulators. This is particularly important if different simulators are generating the same output format within one ISF process.

A.5 Detector Regions

Various different simulation technologies are available for ATLAS detector simulation, dependent on the particular region of the detector. In particular fast simulation engines are often tailor-made for certain ATLAS sub-detectors only. Thus, as mentioned in Section 6.2, the ISF routing algorithm uses independent routing chains for each ATLAS detector region. These detector regions coincide either with ATLAS sub-detectors, or with generic parts of the detector that are fundamentally different from others.

The different ATLAS detector regions are defined in the `AtlasDetDescr::AtlasRegion` enum. The `AtlasDetDescr::AtlasRegion` enum is not specific to ISF simulation. Thus, it is defined in the DetectorDescription/AtlasDetDescr package in the ATLAS offline software repository. As of ATLAS offline software release `17.7.5.4`, the following regions are defined in this enum:

`fUndefinedAtlasRegion` – undefined region

`fAtlasID` – ATLAS inner detector

`fAtlasForward` – ATLAS forward beampipe structure

`fAtlasCalo` – ATLAS calorimeter

`fAtlasMS` – ATLAS muon spectrometer

`fAtlasCavern` – ATLAS cavern

The geometrical definition and extension of the individual regions is shown in Figure 7.1. The boundaries of the individual ATLAS regions are often called envelopes. Thus, an envelope definition Athena service provides the numerical values of the region boundaries to Athena tools, services or algorithms (Section A.5.1).

Most detector simulation setups will simulate particles originating from a point close to the nominal interaction point in the centre of the ATLAS detector. Thus, particles emerging from this point will generally traverse parts of the physical ATLAS beam pipe before entering the ATLAS inner detector. As the ISF has separate routing chains for each ATLAS detector region, such particles would first encounter the `fAtlasForward` routing chain, before entering the `fAtlasID` region, where a different routing chain will be used. In addition, simulators in the ISF are required to return all particles to the ISF, when they pass boundaries between ATLAS regions (Section A.4.1). This generates unnecessary computational overhead and complexity, since the central beam pipe region can be considered as a part of the ATLAS inner detector region inside the simulators. Thus, the boundaries of inner detector (`fAtlasID`) and the forward beampipe (`fAtlasForward`) regions are slightly modified within the ISF. In the ISF, the `fAtlasID` region includes the portion of the beam pipe which is fully enclosed by the physical ATLAS inner detector. Thus, the ISF inner detector region starts at a radius of $r = 0$ in the cylindrical ATLAS coordinate system. The forward beampipe region is adjusted accordingly.

A.5.1 Envelope Definition Service

Class Summary	
Interface Name	`IEnvelopeDefSvc`
Containing Package	AtlasGeometryCommon/SubDetectorEnvelopes
Implementation Name	`DetDescrDBEnvelopeSvc`
Containing Package	AtlasGeometryCommon/SubDetectorEnvelopes
Implementation Name	`ISF::ISFEnvelopeDefSvc`
Containing Package	Simulation/ISF/ISF_Core/ISF_Services

The envelope definition service is an `AthService` which implements the `IEnvelopeDefSvc` interface. It returns sets of (r, z) coordinate pairs that define the boundaries (envelopes) of the individual ATLAS detector regions in the `AtlasDetDescr::AtlasRegion` enum. An envelope definition service requires a cylindrical symmetry around the z axis for all ATLAS detector regions. A mirror symmetry in the $x - y$ plane may be intrinsically assumed by some envelope definition service implementations.

The most important methods defined in the `IEnvelopeDefSvc` interface are:

`getRZBoundary(..)` returns a vector of `RZPairs` (`RZPairVector`) for a given `AtlasDetDescr::AtlasRegion` element. The vector contains an ordered list of (r, z) coordinates that define the boundary of the given ATLAS region.

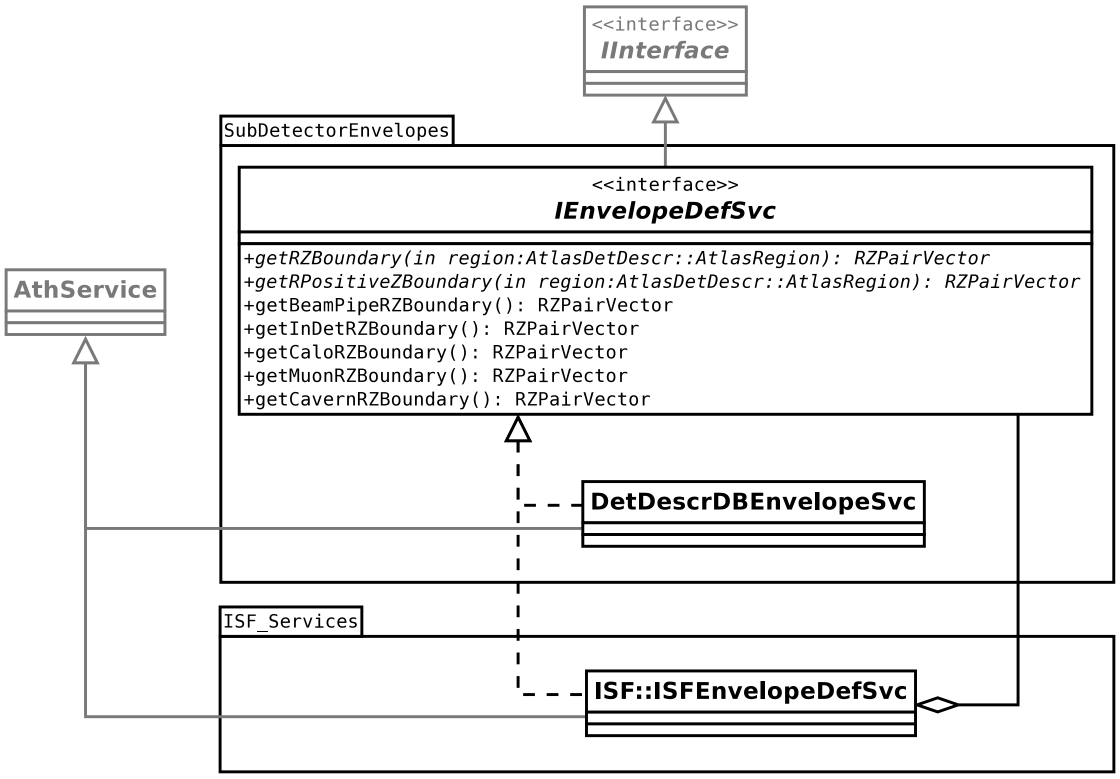
`getRPositiveZBoundary(..)` returns a vector of `RZPairs` (`RZPairVector`) for a given `AtlasDetDescr::AtlasRegion` element. It differs from `getRZBoundary(..)` as it only returns the (r, z) coordinate pairs where $z > 0$.

`getBeamPipeRZBoundary()`, `getInDetRZBoundary()`, `getCaloRZBoundary()`, `getMuonRZBoundary()`, `getCavernRZBoundary()` are wrapper methods of the `getRZBoundary(..)` method for the individual ATLAS regions.

Figure A.7 shows the class diagram of the `IEnvelopeDefSvc` interface and its implementations. As of ATLAS offline software release `17.7.5.4`, two envelope definition service implementations exist:

`DetDescrDBEnvelopeSvc` accesses the detector description database (DDDB) [118, 119] to read the numerical values which define the individual ATLAS region boundaries. Alternatively, it can be configured to read these boundary coordinates from a Python-based configuration. The boundary coordinates in both the DDDB or Python, must only define points with $z > 0$. A symmetry in the $x-y$ plane is intrinsically assumed for all detector regions. Thus the `DetDescrDBEnvelopeSvc` automatically generates the necessary points in the $z < 0$ space. A default (simulation independent) `DetDescrDBEnvelopeSvc` configuration exists, and it can be retrieved through a `ConfiguredFactory` with the name `AtlasGeometry_EnvelopeDefSvc`.

`ISF::ISFEnvelopeDefSvc` returns the ISF definition of the ATLAS detector regions (Section A.5). It reads the global ATLAS region definition from the `DetDescrDBEnvelopeSvc` and adjusts the inner detector volume (`AtlasDetDescr::fAtlasID`) and forward beampipe (`AtlasDetDescr::fAtlasForward`) accordingly. The ISF retrieves its ATLAS detector region definitions directly from this service.



<<interface>>
IInterface
SubDetectorEnvelopes
<<interface>>
IEnvelopeDefSvc
+getRZBoundary(in region:AtlasDetDescr::AtlasRegion): RZPairVector
+getRPositiveZBoundary(in region:AtlasDetDescr::AtlasRegion): RZPairVector
+getBeamPipeRZBoundary(): RZPairVector
+getInDetRZBoundary(): RZPairVector
+getCaloRZBoundary(): RZPairVector
+getMuonRZBoundary(): RZPairVector
+getCavernRZBoundary(): RZPairVector
AthService
DetDescrDBEnvelopeSvc
ISF_Services
ISF::ISFEnvelopeDefSvc

Figure A.7: The UML class diagram of the `IEnvelopeDefSvc` with the `DetDescrDBEnvelopeSvc` and `ISF::ISFEnvelopeDefSvc` implementations. The `DetDescrDBEnvelopeSvc` returns the ATLAS region boundaries as defined in the detector description database or in the Python configuration, both of which are global to ATLAS and not specific to simulation. The `ISF::ISFEnvelopeDefSvc` returns ATLAS region boundaries which are slightly modified from the ATLAS global definition, in order to lower the computational costs in the ISF detector simulation.

A.5.2 GeoID Service

A GeoID service (or GeoIDService) is an `AthService` which implements the `ISF::IGeoIDSvc` interface. It is mainly used to resolve the `AtlasDetDescr::AtlasRegion` corresponding to a given global position in the ATLAS coordinate system. The name GeoID was used in earlier software versions to describe regions in the ATLAS detector, which are now defined by the `AtlasDetDescr::AtlasRegion` enum. A GeoID service implementation is used by the ISF routing algorithm (the ParticleBroker), to identify the appropriating routing chain for each individual particle (Section A.3).

The most important methods defined in the `ISF::IGeoIDSvc` interface are:

`inside(..)` returns whether the given position is inside, outside or on the surface of a given `AtlasDetDescr::AtlasRegion`. The return value is of type `ISF::InsideType`.

`identifyGeoID(..)` returns the `AtlasDetDescr::AtlasRegion` within which the given position is located.

`identifyNextGeoID(..)` predicts the `AtlasDetDescr::AtlasRegion` within which a given particle (or combination of position and momentum vector) will be simulated in the next step. Some implementations approximate this by resolving the ATLAS region for a point which is linearly extrapolated (by a about a millimeter) from the given position along the given momentum.

`identifyAndRegNextGeoID(..)` resolves the next ATLAS region within which the given `ISFParticle` will be simulate and updates the `ISFParticle::nextGeoID()` entry for the particle accordingly. The ATLAS region is resolved by calling `identifyNextGeoID(..)` for the position and momentum direction of the given particle.

The `ISF::IGeoIDSvc` interface contains wrapper methods for the methods described above, accepting position and momentum variables of various types. Common types to containing position and momenta information are: `Amg::Vector3D`, `HepGeom::Point3D<double>`, `HepGeom::Vector3D<double>` or `ISFParticle`.

Figure A.8 shows the UML class diagram of the `ISF::IGeoIDSvc` interface an its implementations. As of ATLAS offline software release `17.7.5.4`, two GeoID service implementations exist:

`ISF::GeoIDSvc` is a `IGeoIDSvc` implementation which is optimized for high speed and minimal computational costs. It is accessed by the ParticleBroker and the Geant4 implementation which is specific to the ISF (Section 9.1). It is used by the prior to determine whether or not a given particle is positioned on a boundary between two ATLAS detector regions and which region the given particle will be simulated in its next step (Section 7.3.1). Approximative methods are used by the `GeoIDSvc`

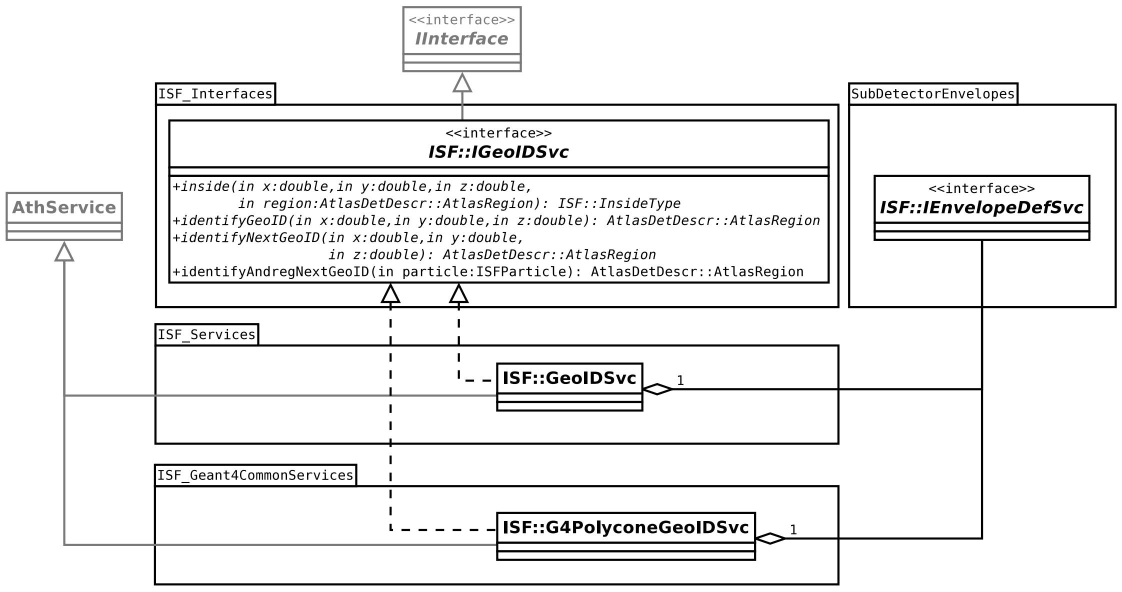

<<interface>>
IInterface
ISF_Interfaces
<<interface>>
ISF::IGeoIDSvc
+inside(in x:double,in y:double,in z:double,
in region:AtlasDetDescr::AtlasRegion): ISF::InsideType
+identifyGeoID(in x:double,in y:double,in z:double): AtlasDetDescr::AtlasRegion
+identifyNextGeoID(in x:double,in y:double,
in z:double): AtlasDetDescr::AtlasRegion
+identifyAndregNextGeoID(in particle:ISFParticle): AtlasDetDescr::AtlasRegion
SubDetectorEnvelopes
<<interface>>
ISF::IEnvelopeDefSvc
AthService
ISF_Services
ISF::GeoIDSvc
1
ISF_Geant4CommonServices
ISF::G4PolyconeGeoIDSvc
1

Figure A.8: The UML class diagram of the `ISF::IGeoIDSvc` interface with the `ISF::GeoIDSvc` and `ISF::G4PolyconeGeoIDSvc` implementations. The `ISF::GeoIDSvc` is optimized for fast processing speed when resolving the ATLAS region of a given position within the ATLAS coordinate system. The slower `ISF::G4PolyconeGeoIDSvc` uses Geant4 classes to provide precise results where the `ISF::GeoIDSvc` uses approximative methods.

in order to guarantee high execution speed. The ISF-specific Geant4 implementation uses the `GeoIDSvc` to determine whether or not a particle has traversed the boundary of an ATLAS detector region in the previous simulation step.

`ISF::G4PolyconeGeoIDSvc` uses `G4Polycone` instances to model the ATLAS detector regions. It uses methods provided by this Geant4 class to determine the return values for the individual `IGeoIDSvc` interface methods. It is slower than the `GeoIDSvc` implementation. However, it uses exact methods to determine whether a given position is on a ATLAS region boundary or not, contrary to the `GeoIDSvc`. This implementation was used by the ParticleBroker before the `GeoIDSvc` was capable of determining whether particles are on sub-detector boundaries or not.

A.6 Monte Carlo Truth

One important component of ATLAS Monte Carlo simulation is the ability to generate a representation of the particles and interactions that were computed by the simulation engines for each simulated event. The ATLAS collaboration uses the HepMC format to store a tree structure of the particles and interactions simulated in the event generation and detector simulation step [59, 63, 64].

The event generator output (which serves as the input for the ISF) is stored in the `GEN_EVENT` collection in the EVNT file format. The ISF copies the `GEN_EVENT` collection from the input EVNT file into the output HITS file. This `GEN_EVENT` collection is one of two MC truth collections in the HITS file. The second collection is named `TruthEvent`. The `TruthEvent` collection contains a copy of the `GEN_EVENT` which is manipulated (moved) according to the given beam spot conditions in the detector simulation. In addition, the ISF appends secondary particles generated by the detector simulation to the corresponding event generator particles in the `TruthEvent`. However, only a subset of all secondary particles and interactions are stored in the `TruthEvent`.

Figure 8.2 shows the interplay of the various ISF `AthAlgTools`, `AthServices` and `AthAlgorithms` involved in the generation of the MC truth representation. The following subsections describe each component and its functionality within the ISF in more detail. Section A.6.1 describes the preparation of the simulation input inside the StackFiller. TruthIncidents (Section A.6.2) are used as wrappers to allow the ISF access to simulator-specific information regarding the simulated particles and physics process. A central TruthService (Section A.6.3) determines the level of MC truth information which is persistently stored in the `TruthEvent` collection in the simulation output. For this, the TruthService uses a set of TruthStrategies (Section A.6.4) which define whether or not a given Truth Incident is significant enough to justify persistent storage. The BarcodeService (Section A.6.5) is used to assign particle barcodes (integer numbers) to secondary particles created by a simulator.

A dedicated ATLAS offline software container was set up within the ISF container to store all packages with dependencies to the (external) HepMC library: Simulation/ISF/ISF_HepMC . As most of the ISF MC truth implementation has dependencies on the HepMC library, the respective classes reside in packages within this container.

A.6.1 Input Processing

Class Summary	
Interface Name	`ISF::IStackFiller`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::GenEventStackFiller`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Tools
Interface Name	`ISF::IGenEventManipulator`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Interfaces
Implementation Name	`ISF::GenEventValidityChecker`, `ISF::GenEventVertexPositioner`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Tools
Interface Name	`ISF::IGenParticleFilter`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Interfaces
Implementation Name	`ISF::GenParticleFinalStateFilter`, `ISF::GenParticlePositionFilter`, `ISF::GenParticleGenericFilter`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Tools
Interface Name	`ISF::ILorentzVectorGenerator`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Interfaces
Implementation Name	`ISF::VertexBeamCondPositioner`, `ISF::LongBeamspotVertexPositioner`, `ISF::VertexPositionFromFile`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Tools

An `IStackFiller` implementation prepares the simulation input for the ATLAS detector simulation within the ISF. It also initializes the MC truth output collection `TruthEvent`. The `IStackFiller` interface is defined in the Simulation/ISF/ISF_-HepMC/ISF_HepMC_Interfaces package in ATLAS offline software repository. Figure A.9 shows a UML class diagram of the implementations and interface classes involved in ISF input processing. The details of which are discussed in this subsection.

The `GenEventStackFiller` is the only implementation of the `IStackFiller` interface. It resides in the Simulation/ISF/ISF_HepMC/ISF_HepMC_-Services package in the ATLAS offline software repository. The name stems from the `HepMC::GenEvent` data format serving as input to the `GenEventStackFiller`. The `ParticleBrokerDynamicOnReadIn` (Section A.3) calls the `IStackFiller::fillStack(..)` method during the event initialization. The responsibility of the `IStackFiller` is to fill the empty `ISFParticleContainer` – provided by the `ParticleBrokerDynamicOnReadIn` – with `ISFParticles` which need to be simulated through the ATLAS detector. For this, the `GenEventStackFiller` creates a new StoreGate collection named `TruthEvent`. Initially it is filled with an exact copy of the `GEN_EVENT` collection. After necessary adjustments to its contents it will serve as the final input collection for the detector simulation. The initial `TruthEvent` is modified by `IGenEventManipulators` which prepare the particles according to the simulation configuration and beam conditions. Two `IGenEventManipulator` implementations exist that can be used by the `GenEventStackFiller`:

`GenEventValidityChecker` checks if the numeric values of the input `HepMC::GenVertex` coordinates are valid, i.e. smaller than infinite.

`GenEventVertexPositioner` is used to position all `HepMC::GenVertex` instances in the simulation input according to the beam conditions. It calls a set of `VertexShifters` (a `ToolHandleArray` of `ILorentzVectorGenerators`) to compute 4-vector shifts which will be added in sequence to all vertex positions (`HepMC::GenVertex::position()`) in the input.

Three `ILorentzVectorGenerator` implementations exist that can be used by the `GenEventVertexPositioner`:

`VertexBeamCondPositioner` accesses the ATLAS `BeamCondSvc` to determine the beam spot position and beam spot size. It computes a total shift for each event, which is applied to all input particle vertices equally by the `GenEventVertexPositioner`. The position is computed randomly with a three-dimensional Gaussian probability function around the provided beam spot position and with widths corresponding to the beam spot widths (in x, y and z coordinates).

`LongBeamspotVertexPositioner` has the same underlying functionality as `VertexBeamCondPositioner` with the extension of generating beam spot positions for HC-LHC [120] studies.

`VertexPositionFromFile` reads the vertex shifts from a separate file rather than the `BeamCondSvc`.

After the `GenEventStackFiller` has processed all `IGenEventManipulators` it will determine a subset of particles in the `TruthEvent` collection which are to be converted into `ISFParticles` for detector simulation. This is done by calling `IGenParticleFilter::pass(..)` for each `HepMC::GenParticle` in the input `HepMC::GenEvent` instance. If more than one `IGenParticleFilter` implementation is configured, the `GenEventStackFiller` will loop over all `IGenEventManipulators` for each `HepMC:GenParticle`. A particle will be converted into an `ISFParticle` (and thus processed by a detector simulation later on) if all its `IGenParticleFilter::pass(..)` calls return `true`. Three `IGenParticleFilter` implementations exist that are commonly used by the `GenEventStackFiller`:

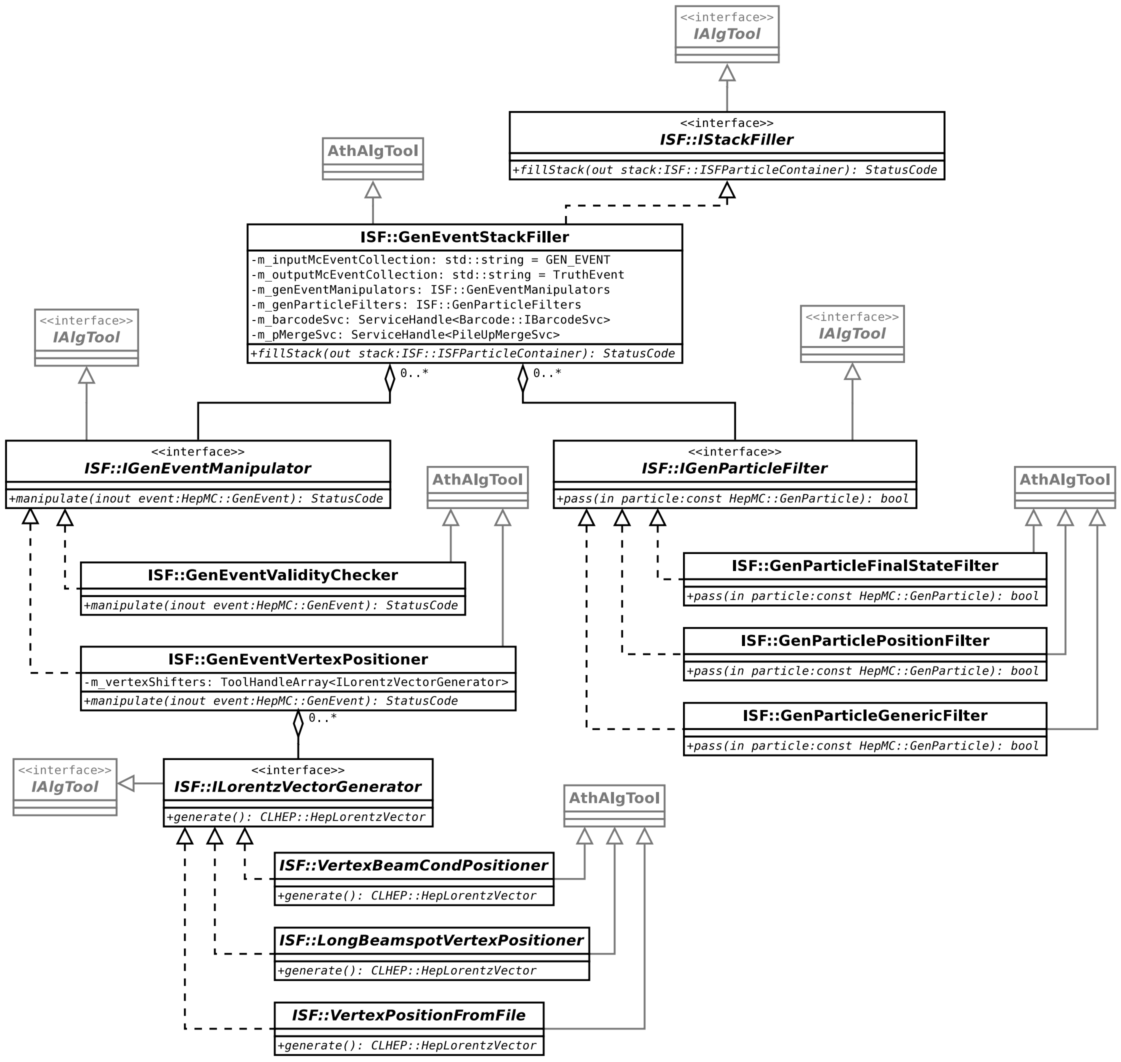
`GenParticleFinalStateFilter` determines whether the given particle is a final state particle or whether it is an intermediate particle produced by the event generator. Only final state particles will pass this filter. ATLAS simulation `TruthHelper` classes are used to determine the whether or not a particle is present in a final state.

`GenParticlePositionFilter` determines whether the particle is inside the ATLAS detector volume.

`GenParticleGenericFilter` is a very flexible filter that can check a particle's PDG type [113] and/or whether a particle's momentum vector is within specified ranges of η and ϕ.

The `GenEventStackFiller` registers the greatest `HepMC::GenParticle` barcode appearing in the `TruthEvent` with the `Barcode::IBarcodeSvc`. This information may be necessary for the `Barcode::IBarcodeSvc` implementation to determine the lowest unique secondary particle barcode it can generate for this event.

After filling and returning the `ISFParticleContainer` from inside the `GenEventStackFiller::fillStack(..)` method, the ISF simulation input processing is completed and the processing continues inside the `ParticleBrokerDynamicOnReadIn` implementation.



<<interface>>
IAlgTool
<<interface>>
ISF::IStackFiller
+fillStack(out stack:ISF::ISFParticleContainer): StatusCode
AthAlgTool
ISF::GenEventStackFiller
-m_inputMcEventCollection: std::string = GEN_EVENT
-m_outputMcEventCollection: std::string = TruthEvent
-m_genEventManipulators: ISF::GenEventManipulators
-m_genParticleFilters: ISF::GenParticleFilters
-m_barcodeSvc: ServiceHandle<Barcode::IBarcodeSvc>
-m_pMergeSvc: ServiceHandle<PileUpMergeSvc>
+fillStack(out stack:ISF::ISFParticleContainer): StatusCode
0..*
0..*
<<interface>>
IAlgTool
<<interface>>
IAlgTool
<<interface>>
ISF::IGenEventManipulator
+manipulate(inout event:HepMC::GenEvent): StatusCode
AthAlgTool
<<interface>>
ISF::IGenParticleFilter
+pass(in particle:const HepMC::GenParticle): bool
AthAlgTool
ISF::GenEventValidityChecker
+manipulate(inout event:HepMC::GenEvent): StatusCode
ISF::GenParticleFinalStateFilter
+pass(in particle:const HepMC::GenParticle): bool
ISF::GenParticlePositionFilter
+pass(in particle:const HepMC::GenParticle): bool
ISF::GenEventVertexPositioner
-m_vertexShifters: ToolHandleArray<ILorentzVectorGenerator>
+manipulate(inout event:HepMC::GenEvent): StatusCode
0..*
ISF::GenParticleGenericFilter
+pass(in particle:const HepMC::GenParticle): bool
<<interface>>
IAlgTool
<<interface>>
ISF::ILorentzVectorGenerator
+generate(): CLHEP::HepLorentzVector
AthAlgTool
ISF::VertexBeamCondPositioner
+generate(): CLHEP::HepLorentzVector
ISF::LongBeamspotVertexPositioner
+generate(): CLHEP::HepLorentzVector
ISF::VertexPositionFromFile
+generate(): CLHEP::HepLorentzVector

Figure A.9: The UML class diagram for the ISF StackFiller. The StackFiller reads the event generator output from the `GEN_EVENT` StoreGate collection. It modifies these input particles with a set of `IGenEventManipulators`. The `IGenParticleFilters` are used to determine a subset of the input particles which will be converted into `ISFParticles` for the detector simulation.

A.6.2 Truth Incident

Class Summary	
Interface Name	`ISF::ITruthIncident`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event
Implementation Name	`ISF::ISFTruthIncident`
Containing Package	Simulation/ISF/ISF_Core/ISF_Event
Implementation Name	`ISF::Geant4TruthIncident`
Containing Package	Simulation/ISF/ISF_Geant4/ISF_Geant4Tools

TruthIncidents are interfaces between individual simulators and the ISF. The `ITruthIncident` interface class allows the ISF MC truth algorithms to access simulator-specific information regarding particle-material interaction processes and particle decays. This layer of abstraction is required as various simulators implement different methods (many times internal) for accessing the information necessary to generate a common MC truth output. For example, in the case of ATLAS Geant4 simulation `G4Steps` are used to gather all relevant information for a consistent MC truth output. In the case of Fatras simulation, the information is present in the form of `ISFParticles` instances and an identifier for the type of interaction occurring. Thus, individual simulator-specific implementations of one common `ITruthIncident` interface minimizes type conversions and improves performance. `ITruthIncident` interface methods are also used to pass information from the ISF back into the simulator.

A `ITruthIncident` instance is created by a simulator ever time it creates one or many new secondary particles due interaction or decay processes. The `ITruthIncident` instance is then provided to the ISF TruthService (Section A.6.3) to determine whether the given interaction is to be recorded persistently in the `TruthEvent` StoreGate collection. For this, the ISF TruthService will gather information about the characteristics of the interaction by calling one or many of the methods defined in the `ITruthIncident` interface. Thus, simulator-specific implementations are required to prepare and provide this information from the objects available within the simulator. The ISF TruthService may return newly generated particle barcodes (for the secondary particles) to the simulator via the `ITruthIncident` interface methods. New particle barcodes are only generated if the ISF TruthService decides to persistently store the given `ITruthIncident`.

Type conversions of primary or secondary particles involved in a truth incident into the `HepMC::GenParticles` format may be requested by the ISF TruthService. For this, the ISF TruthService will call the `ISF::ITruthIncident::primaryParticle(..)` or `ISF::ITruthIncident::secondaryParticle(..)` methods. It is required that the TruthIncident implementation returns the exact same `HepMC::GenParticle` instance upon querying the interface methods multiple times for the same physical particle, even if the particle is queried from within different TruthIncident instances. For example, a newly created *secondary* particle in TruthIncident A may undergo an inelastic interaction later during the simulation and thus become the *primary* particle in TruthIncident *B*. In order to build a consistent MC truth tree structure, it is required that the call `ISF::ITruthIncident::secondaryParticle(..)` for this particular particle in TruthIncident *A* returns exactly the same `HepMC::GenParticle` instance as the call `ISF::ITruthIncident::primaryParticle(..)` for TruthIncident *B*. If the TruthService determines to persistently store the particles contained in a TruthIncident, it will set a "persistency flag" for the respective particles. This information may be required by the TruthIncident implementation, as the ownership of `HepMC::GenParticle` instances which are determined for persistent storage is transferred to the StoreGate service.

The `ITruthIncident` holds an identifier for the ATLAS detector region or sub-detector within which the incident occurred. The identifier is of type `AtlasDetDescr::AtlasRegion`. The simulator creating a TruthIncident instance is required to fill this field correctly, as it will be of crucial importance for the proper function of the service and the consistency of the entire Monte Carlo truth output.

Two `ITruthIncident` implementations are in use by the ISF (Figure A.10):

The `ISFTruthIncident` extracts the information relevant for generating the MC truth output from `ISFParticle` instances. Fatras is the sole client of the `ISFTruthIncident` in the current implementation. This is due to Fatras being the only simulator that uses `ISFParticle` types internally for particle processing. Hence, Fatras creates new `ISFTruthIncident` instances without the need for excessive type conversions. The `ISFTruthIncident` class is implemented in the Simulation/ISF/ISF_Core/ISF_Event package in the ATLAS offline software repository.

`Geant4TruthIncident` instances are created by the Geant4 simulator. It allows the ISF TruthService to access the information present in a given `G4Step` instance. A `SecondaryTracksHelper` instance is used to retrieve all secondary particles produced by Geant4 in the current step. The `TruthHelper` has already been implemented and used prior to the integration of Geant4 into the ISF. The `Geant4TruthIncident` class is implemented in the Simulation/ISF/ISF_Geant4/ISF_Geant4Tools package in the ATLAS offline software repository.

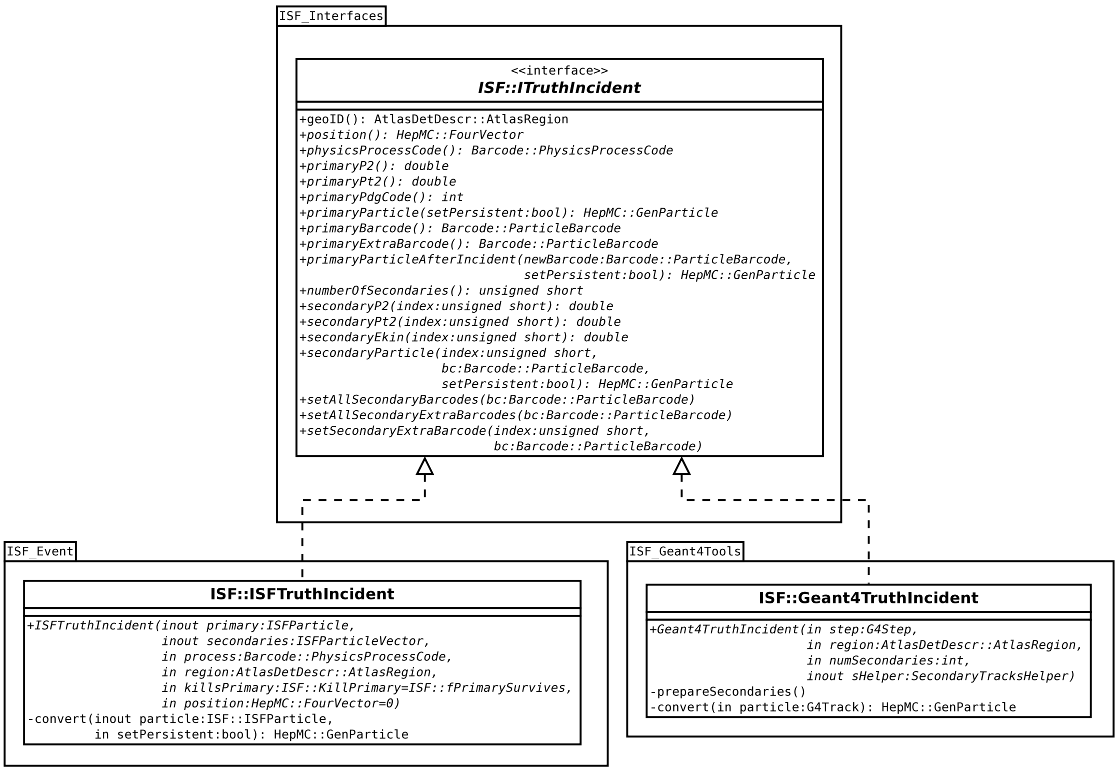

ISF_Interfaces
<<interface>>
ISF::ITruthIncident
+geoID(): AtlasDetDescr::AtlasRegion
+position(): HepMC::FourVector
+physicsProcessCode(): Barcode::PhysicsProcessCode
+primaryP2(): double
+primaryPt2(): double
+primaryPdgCode(): int
+primaryParticle(setPersistent:bool): HepMC::GenParticle
+primaryBarcode(): Barcode::ParticleBarcode
+primaryExtraBarcode(): Barcode::ParticleBarcode
+primaryParticleAfterIncident(newBarcode:Barcode::ParticleBarcode, setPersistent:bool): HepMC::GenParticle
+numberOfSecondaries(): unsigned short
+secondaryP2(index:unsigned short): double
+secondaryPt2(index:unsigned short): double
+secondaryEkin(index:unsigned short): double
+secondaryParticle(index:unsigned short, bc:Barcode::ParticleBarcode, setPersistent:bool): HepMC::GenParticle
+setAllSecondaryBarcodes(bc:Barcode::ParticleBarcode)
+setAllSecondaryExtraBarcodes(bc:Barcode::ParticleBarcode)
+setSecondaryExtraBarcode(index:unsigned short, bc:Barcode::ParticleBarcode)
ISF_Event
ISF::ISFTruthIncident
+ISFTruthIncident(inout primary:ISFParticle, inout secondaries:ISFParticleVector, in process:Barcode::PhysicsProcessCode, in region:AtlasDetDescr::AtlasRegion, in killsPrimary:ISF::KillPrimary=ISF::fPrimarySurvives, in position:HepMC::FourVector=0)
-convert(inout particle:ISF::ISFParticle, in setPersistent:bool): HepMC::GenParticle
ISF_Geant4Tools
ISF::Geant4TruthIncident
+Geant4TruthIncident(in step:G4Step, in region:AtlasDetDescr::AtlasRegion, in numSecondaries:int, inout sHelper:SecondaryTracksHelper)
-prepareSecondaries()
-convert(in particle:G4Track): HepMC::GenParticle

Figure A.10: The `ITruthIncident` interface and its two implementations: `ISFTruthIncident` and `Geant4TruthIncident`. The prior is used by the Fatras simulator and the latter by the Geant4 simulator. Both implementations are fully compatible with the `ITruthIncident` interface and thus are used by the central ISF TruthService. TruthIncidents provide access to simulator-internal information regarding interaction and decay processes that have occurred during the detector simulation. This information is collected by the ISF TruthService which determines whether or not the interaction will be recorded to the `TruthEvent` StoreGate collection.

A.6.3 Truth Service

Class Summary	
Interface Name	`ISF::ITruthSvc`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::HepMC_TruthSvc`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Services

The TruthService generates a consistent Monte Carlo truth representation for each simulated event. It is the central `AthService` within the ISF which receives all TruthIncidents that are created by one or more simulators. It uses TruthStrategies (Section A.6.4) to determine which of the given `ITruthIncident` instances are to be stored persistently in the HITS file.

Figure 8.3 illustrates the flow of information when secondary particles are generated inside a simulator. The simulator wraps the relevant information regarding the interaction into a TruthIncident instance (Section A.6.2). This TruthIncident instance is subsequently registered with the central ISF TruthService by calling `ISF::ITruthSvc::register(..)` from within the simulator. The TruthService uses a `Barcode:IBarcodeSvc` instance (Section A.6.5) to retrieve particle and vertex barcodes for newly created secondaries in the given truth instance.

The `HepMC_TruthSvc` is currently the only MC TruthService implementation in use by the ISF. The implementation resides in the Simulation/ISF/ISF_HepMC/ISF_-HepMC_Services package in the ATLAS offline software repository. The `HepMC_TruthSvc` holds an Athena `ToolHandleArray` of `ISF::ITruthStrategies` for each ATLAS detector region. The detector regions are defined by the `AtlasDetDescr::AtlasRegion` identifier. Upon calling `HepMC_TruthSvc::registerTruthIncident(...)` from within a simulator, the TruthService will iterate through the TruthStrategies configured for the detector region within which the truth incident has occurred. If at least one `ISF::ITruthStrategy::pass(..)` call returns `true`, the given TruthIncident will be recorded to the `TruthEvent` collection on StoreGate. Different detector regions will generally be configured with different sets of TruthStrategies. For example the amount of truth information which is to be recorded in the ATLAS inner detector (`AtlasDetDescr::fAtlasID`) is much more detailed than in the calorimeter region (`AtlasDetDescr::fAtlasCalo`). Thus, it is necessary that the `ISF::ITruthIncident` instance is provided with the correct `AtlasDetDescr::AtlasRegion` identifier (often referred to as `geoID` or GeoID) upon creation inside the simulator.

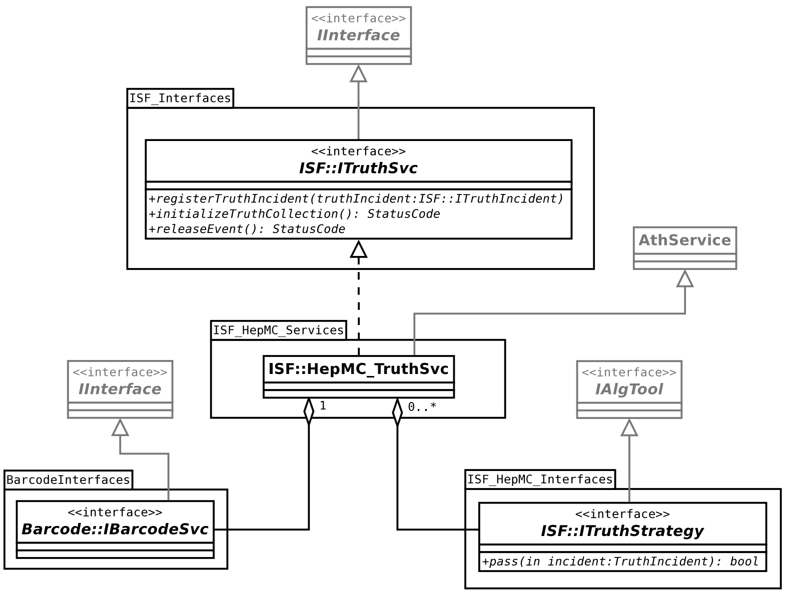


Figure A.11: The class diagram of the ISF TruthService. The `ITruthSvc` interface defines the possible interactions between simulators and the ISF TruthService. The `ITruthSvc::registerTruthIncident(..)` is used by a simulator to communicate to the ISF that an interaction or decay has occurred in which secondary particles are produced. The ISF TruthService will subsequently determine whether this interaction is to be stored persistently. The `HepMC_TruthSvc` is the only TruthService implementation in use by the ISF. It accesses the `Barcode::IBarcodeSvc` BarcodeService interface to generate barcodes for secondary vertices and particles.

A.6.4 Truth Strategies

Class Summary	
Interface Name	`ISF::ITruthStrategy`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Interfaces
Implementation Name	`ISF::GenericTruthStrategy`, `ISF::ValidationTruthStrategy`, `ISF::CylinderVolumeTruthStrategy`
Containing Package	Simulation/ISF/ISF_HepMC/ISF_HepMC_Tools

The ISF TruthStrategies are an essential component of the `HepMC_TruthSvc` ISF TruthService implementation (Section A.6.3). They are required for the creation of a consistent Monte Carlo truth representation of the detector simulation. TruthStrategies are `AthAlgTools` implementations which use a common `ISF::ITruthStrategy` interface. The `ITruthStrategy` is a very lightweight interface which defines only the

`ITruthStrategy::pass(..)` public method. Provided with a `ITruthIncident` instance, the method returns a Boolean type variable. The return value corresponds to whether the TruthStrategy considers the given TruthIncident relevant for persistent storage in the `TruthEvent` StoreGate collection. As mentioned previously, the `HepMC_TruthSvc` holds a `ToolHandleArray` of `ISF::ITruthStrategies` for each ATLAS detector region. Only if at least one TruthStrategy from within the ATLAS region returns `true` (upon calling the `pass(..)` method), the TruthIncident is recorded to the `TruthEvent` StoreGate collection. Thus, ISF will usually run with multiple instances of one or many `ITruthStrategy` implementations. Figure A.12 shows an overview of the TruthStrategy interface and its current implementations.

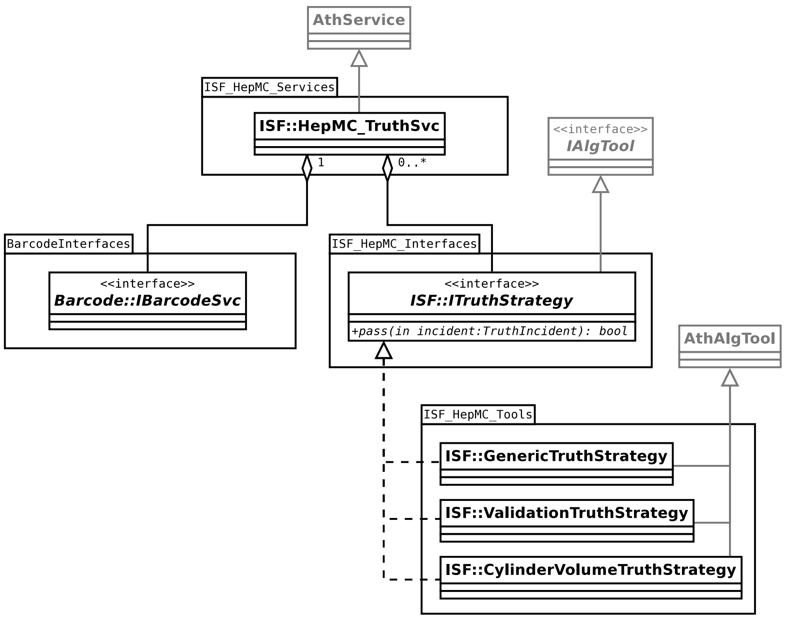


Figure A.12: The class diagram for the `ISF::ITruthStrategy` interface and its implementations. Due to its flexibility, the `GenericTruthStrategy` is the most commonly used TruthStrategy in various ISF production setups.

Three `ITruthStrategy` implementations are currently available for use within the ISF:

The `GenericTruthStrategy` is currently the most commonly used TruthStrategy implementation by the ISF. It offers a number of particle and vertex cuts that can be adjusted (even enabled and disabled) via the Athena Python configuration level. Available cuts are: minimum p_T and kinetic energy of the primary and secondary particles, primary particle PDG type and vertex identifiers. This TruthStrategy implementation is thus very flexible and more than one instance is generally used in a production ISF setup.

The `ValidationTruthStrategy` is a simpler and less flexible version of the `GenericTruthStrategy`. It offers only one parameter, a minimum p_{T} cut for the primary particle. This TruthStrategy is applied when a highly detailed MC truth output is required, e.g. data samples for simulator validation.

The `CylinderVolumeTruthStrategy` is used to select TruthIncidents which have a radial position between a defined inner and outer radius, i.e. $r_{\mathrm{inner}} < r = \sqrt{x^2 + y^2} < r_{\mathrm{outer}}$. One application of this strategy is to record TruthIncidents occurring on specific layers of the ATLAS inner detector geometry.

A.6.5 Barcodes and Barcode Service

Class Summary	
Interface Name	`Barcode::IBarcodeSvc`
Containing Package	Simulation/Barcode/BarcodeInterfaces
Implementation Name	`Barcode::GenericBarcodeSvc`, `Barcode::GlobalBarcodeSvc`, `Barcode::LegacyBarcodeSvc`, `Barcode::ValidationBarcodeSvc`
Containing Package	Simulation/Barcode/BarcodeServices

Particle barcodes play an important role for identifying the simulated particle which causes a given sensitive detector (SD) hit. Along with other properties, a particle's barcode is recorded with each simulated sensitive detector hit it causes. In addition, if the particle is present in the Monte Carlo truth output (i.e. it passes a TruthStrategy), it can be identified with its unique barcode within the `HepMC::GenEvent`. Since the MC truth representation will be present in any subsequent stage after detector simulation, simulated particle properties can be obtained if a particle's barcode is known. However, the amount of information available is limited to the `HepMC::GenParticle` data type, as this is the data format used to represent particles in the MC truth output. In addition, the MC truth output only contains a subset representation of all the particles that were actually simulated. Thus, not all particles creating SD hits will be found in the MC truth output. One approach to storing additional information in sensitive detector hits and in the MC truth representation is to encode such information into particle and vertex barcodes. If a particle is not present in the MC truth representation one may identify its primary particle from decoding the barcode stored in the sensitive detector hit. The actual simulator (fast or full simulation) which created the particle may be encoded into the particle barcode as well. Vertex barcodes may contain information about the physics process of the interaction or decay they represent.

The ISF TruthService (`ISF::HepMC_TruthSvc`) uses a central BarcodeService to generate particle and truth vertex barcodes for secondary interactions inside the ATLAS detector. The `Barcode::IBarcodeSvc` interface introduces a layer of abstraction that allows different actual BarcodeService implementations to exist. Each implementation may encode different kinds of information into the particle and vertex barcodes. Though, only one BarcodeService may be used at one time in an ISF simulation run.

The BarcodeService interface (`Barcode::IBarcodeSvc`) and implementations are independent of the ISF and the ATLAS detector simulation in general. This allows for the use of the very same BarcodeService to encode information into barcodes (during the detector simulation) and decode this information from a given barcode later on (analysis of the simulation output). The barcode classes are contained in the `Barcode` namespace. Particle barcodes are represented by the `Barcode::ParticleBarcode` data type, vertex barcodes are of the `Barcode::VertexBarcode` data type. Both are `typedef`s of the C++ integer data type. The use of the `ParticleBarcode` and `VertexBarcode` types throughout the framework allows to control the actual barcode data type in one central place – the header file where both types are defined. In case greater numerical ranges of barcodes are required or an actual C++ class is introduced for barcodes, the existing source code using barcodes will require minimal to no changes. The BarcodeService interface is defined in the `Barcode::IBarcodeSvc` class.

The `IBarcodeSvc` interface (Figure A.13) declares a number of methods which will be called by the TruthService to request different types of new barcodes:

`newVertex(..)` will return a new unique secondary vertex barcode. Optionally the implementation may use the parent particle barcode and the physics process code of the interaction to generation the new vertex barcode.

`newSecondary(..)` will return a new unique secondary particle barcode. Optionally the implementation may use the parent particle barcode and the physics process code of the interaction to generation the new secondary barcode.

`sharedChildBarcode(..)` will return a secondary particle barcode which will be shared among a number of secondary particles. This functionality is introduced with the ISF for the first time. Shared secondary barcodes are requested by the `ISF::HepMC_TruthSvc` if the secondary particles are not represented in the Monte Carlo truth output. Since unique identification of the particles is not required in this case, a shared barcode may still be used to store a certain amount of information, e.g. regarding the primary particle.

`incrementBarcode(..)` will return a new unique *incremented* barcode based on the given particle barcode. These barcodes are typically requested if a primary particle does not get consumed by the interaction (e.g. bremsstrahlung) and thus an *updated* (incremented) barcode is required.

`registerLargestGenEvtParticleBC(..)` and `registerLargestGenEvtVtxBC(..)` are called during the ISF input processing to inform the BarcodeService implementation of the range of barcodes already in use by the event input.

`hasBitCalculator(..)` and `getBitCalculator(..)` are used to retrieve a ROOT-based barcode encoding/decoding tool from the BarcodeService.

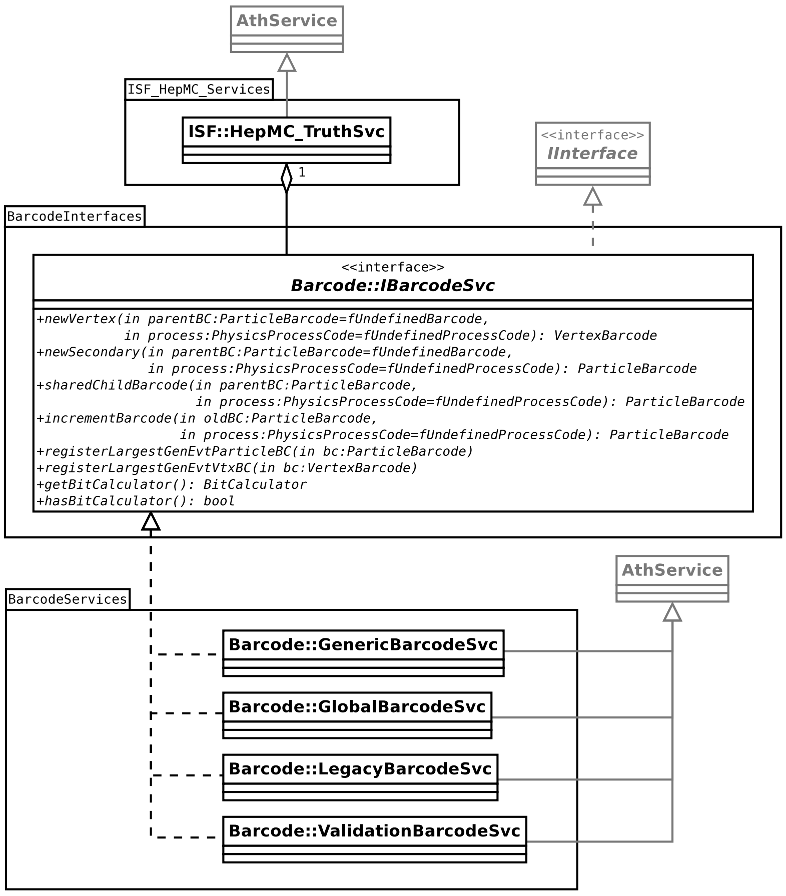
Figure A.13 shows the class diagram of the `Barcode::IBarcodeSvc` interface as well as the four current implementations:

`LegacyBarcodeSvc` is currently the default BarcodeService in the ISF. It provides particle and vertex barcode following the same scheme as in the mc12 Monte Carlo simulation campaign. The secondary particles will be assigned barcodes starting at 200001 which are incremented by 1 for each newly generated particle barcode. The secondary vertices will be assigned barcodes starting at -200001 which are decremented by -1 for each newly generated vertex barcode. Interactions which do not consume the primary particle will increment the primary particle barcode by 1000000 after the interaction.

`GenericBarcodeSvc` is a more generic implementation than the `LegacyBarcodeSvc` and it offers many properties which can be configured in the Athena Python level.

`GlobalBarcodeSvc` is currently in a prototype stage. Its intended use will be for pile-up simulation within the ISF detector simulation (Section 13.3).

`ValidationBarcodeSvc` is used in combination with the `ValidationTruthStrategy` (Section A.6.4) to generate simulator validation samples with highly detailed truth information.



```
+newVertex(in parentBC:ParticleBarcode=fUndefinedBarcode,
           in process:PhysicsProcessCode=fUndefinedProcessCode): VertexBarcode
+newSecondary(in parentBC:ParticleBarcode=fUndefinedBarcode,
              in process:PhysicsProcessCode=fUndefinedProcessCode): ParticleBarcode
+sharedChildBarcode(in parentBC:ParticleBarcode,
                    in process:PhysicsProcessCode=fUndefinedProcessCode): ParticleBarcode
+incrementBarcode(in oldBC:ParticleBarcode,
                  in process:PhysicsProcessCode=fUndefinedProcessCode): ParticleBarcode
+registerLargestGenEvtParticleBC(in bc:ParticleBarcode)
+registerLargestGenEvtVtxBC(in bc:VertexBarcode)
+getBitCalculator(): BitCalculator
+hasBitCalculator(): bool
```

Figure A.13: The class diagram for the `Barcode::BarcodeSvc` interface and its implementations. The `LegacyBarcodeSvc` generates particle and vertex barcodes in mc12-like ISF configurations. The `GenericBarcodeSvc` will become the default barcode service for the mc15 production campaign. The other implementations are used in various test and validation setups, and they are not used in production.

A.6.6 Entry Layer creation with the EntryLayerTool

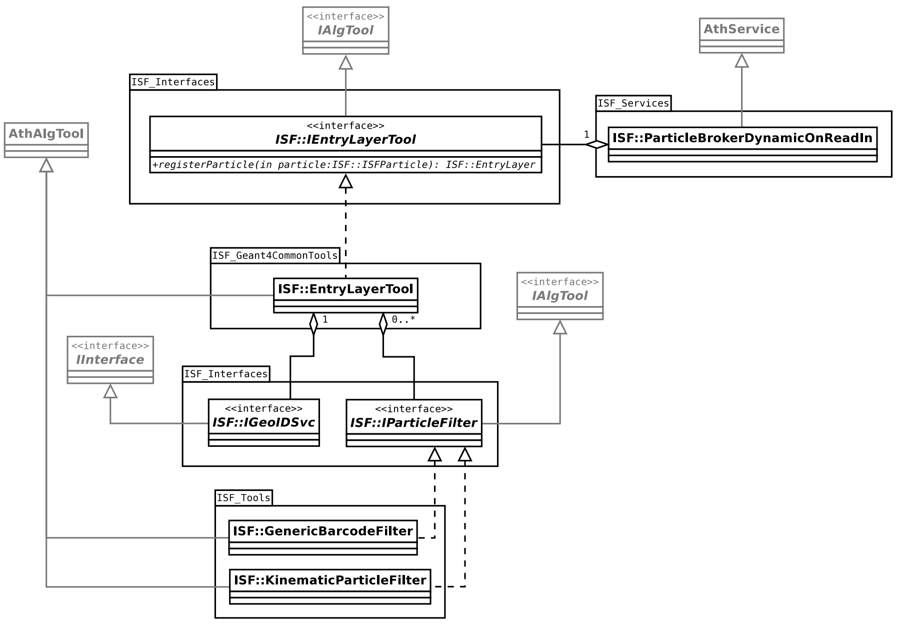
Class Summary	
Interface Name	`ISF::IEntryLayerTool`
Containing Package	Simulation/ISF/ISF_Core/ISF_Interfaces
Implementation Name	`ISF::EntryLayerTool`
Containing Package	Simulation/ISF/ISF_Geant4/ISF_Geant4CommonTools

The EntryLayer records are an important part of the Monte Carlo truth representation in ATLAS detector simulation. They contain MC truth-level information of particles traversing the boundaries between individual ATLAS sub-detectors. In general, only a subset of all particles traversing the sub-detector boundaries are recorded in order to minimize the file size of the simulation output.

StoreGate collections of type `TrackRecordCollection` exist for three EntryLayers respectively: the CaloEntryLayer, the MuonEntryLayer and the MuonExitLayer (Section 7.4.1).

The `ISF::IEntryLayerTool` interface declares the `registerParticle(..)` method which requires one argument of type `ISFParticle`. An implementation of the interface determines whether the given particle is positioned on an EntryLayer, and if so, it will create a representation of the particle in the corresponding `TrackRecordCollection`. This method is called by the ParticleBroker (`ParticleBrokerDynamicOnReadIn` implementation) for all particles which are returned from any simulator to the framework via the `ISF::IParticleBroker::push(..)` method.

The `ISF::EntryLayerTool` is the only implementation of the `IEntryLayerTool` interface. It uses a `ToolHandleArray` of `ISF::IParticleFilters` to determine the subset of particles which are to be recorded persistently. By default, a `ISF::GenericBarcodeFilter` is configured to reject all particles which are not assigned a barcode. Consequently, only particles which are represented in the `TruthEvent` MC truth collection are also recorded into the respective EntryLayer if they traverse a subdetector boundary.



<<interface>> IAlgTool
AthService
ISF_Interfaces
<<interface>> ISF::IEntryLayerTool
+registerParticle(in particle:ISF::ISFParticle): ISF::EntryLayer
ISF_Services
ISF::ParticleBrokerDynamicOnReadIn
1
AthAlgTool
ISF_Geant4CommonTools
ISF::EntryLayerTool
1
0..*
<<interface>> IAlgTool
<<interface>> IInterface
ISF_Interfaces
<<interface>> ISF::IGeoIDSvc
<<interface>> ISF::IParticleFilter
ISF_Tools
ISF::GenericBarcodeFilter
ISF::KinematicParticleFilter

Figure A.14: The class diagram for the `ISF::IEntryLayerTool` interface and the `ISF::EntryLayerTool` implementation. The default configuration of the `ISF::EntryLayerTool` uses the `ISF::GenericBarcodeFilter` to ensure that only particles which are also present in the `TruthEvent` MC truth representation are recorded into the respective EntryLayer collection.

A.7 Framework Configuration

A number of full and fast detector simulation configurations are implemented within the ISF. Any one configuration can be executed through a corresponding string identifier provided to the `Sim_tf.py` job transform script (the MC production job transforms are described in Section 4.2). The entire framework, including all its Athena components, is configured through this string identifier. The identifier is selected through `--simulator` argument of the `Sim_tf.py` job transform.

The Python-level configuration of the ISF is carried out through the Configurable-Factory (also called ConfiguredFactory or ConfGetter) [83] framework. Each Athena component (`AthAlgorithm`, `AthAlgTool` or `AthService`) used in the ISF is configured individually. Each individual configuration of an Athena component is associated with a respective unique string identifier, called ConfGetter identifier. The `--simulator` string identifier provided with the `Sim_tf.py` command line defines the ConfGetter identifier to be used for the ISF SimulationKernel. The configuration of all Athena components in this ISF process is uniquely defined through the configuration of the SimulationKernel, i.e. its ConfGetter identifier.

If an Athena component A accessed another Athena component B (through Athena

`ToolHandles`), the corresponding ConfGetter identifier of component B is provided in the configuration of component A. Multiple configurations of component A exist, each with a respective unique ConfGetter identifier, if it requires differently configured components B in various cases.

Bibliography

[1] The ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. Aug 2008.

[2] Lyndon R Evans and Philip Bryant. LHC Machine. *J. Instrum.*, 3:S08001. 164 p, 2008. This report is an abridged version of the LHC Design Report (CERN-2004-003).

[3] N. Metropolis and S. Ulam. The Monte Carlo Method. *J. Amer. Stat. Assoc.*, 44:335–341, 1949.

[4] N. Metropolis. The Beginning of the Monte Carlo Method. *Los Alamos Science*, 15:125–130, 1987.

[5] The ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, 716(arXiv:1207.7214. CERN-PH-EP-2012-218):1–29. 39 p, Aug 2012. Comments: 24 pages plus author list (38 pages total), 12 figures, 7 tables, revised author list.

[6] The CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B*, 716(arXiv:1207.7235. CMS-HIG-12-028. CERN-PH-EP-2012-220):30–61. 59 p, Jul 2012.

[7] The D0 Collaboration. Measurement of the W boson mass with the D0 detector. *Phys. Rev. D*, 89(arXiv:1310.8628. FERMILAB-PUB-13-489-E):012005. 46 p, Oct 2013. Comments: 46 pages, 61 figures, submitted to Phys. Rev. D.

[8] J Chapman, W Ehrenfeld, J Ferrando, J Garcia Navarro, C Gwenlan, S Mehlhase, V Tsulaia, A Vaniachine, and J Zhong. Challenges of the ATLAS Monte Carlo Production during Run-I. Sep 2014.

[9] J Chapman, W Ehrenfeld, J E Garcia Navarro, C Gwenlan, S Mehlhase, V Tsulaia, and J Zhong. Challenges of the ATLAS Monte Carlo production during run 1 and beyond. Technical Report ATL-SOFT-PROC-2013-045, CERN, Geneva, Nov 2013.

[10] The ATLAS Collaboration. Luminosity Public Results – ATLAS EXPERIMENT Public Results. `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResults`, October 2014.

[11] ECFA High Luminosity LHC Experiments Workshop: Physics and Technology Challenges. 94th Plenary ECFA meeting. Nov 2013.

[12] A Nairz. ATLAS distributed computing: experience and evolution. Technical Report ATL-SOFT-PROC-2013-006, CERN, Geneva, Oct 2013.

[13] The ATLAS Collaboration. ATLAS dashboard Website. `http://dashb-atlas-job.cern.ch/dashboard/request.py/dailysummary`, October 2014.

[14] L Sargsyan, J Andreeva, S Campana, E Karavakis, L Kokoszkiewicz, P Saiz, J Schovancova, and D Tuckett. ATLAS job monitoring in the Dashboard Framework. Technical Report ATL-SOFT-PROC-2012-039, CERN, Geneva, May 2012.

[15] Inc Amazon.com. Amazon Elastic Compute Cloud (EC2). `https://aws.amazon.com/ec2/pricing/`, October 2014.

[16] Yorikiyo Nagashima. *Elementary particle physics*. Wiley, Weinheim, 2010.

[17] K.A. Olive et al. (Particle Data Group). The review of particle physics. *Chin. Phys. C*, 38(090001), 2014.

[18] Yoichiro Nambu. Quasi-particles and gauge invariance in the theory of superconductivity. *Phys. Rev.*, 117:648–663, Feb 1960.

[19] P W Anderson. Plasmons, gauge invariance, and mass. *Phys. Rev.*, 130:439–442, 1963.

[20] F Englert and R Brout. Broken symmetry and the masses of gauge vector mesons. *Phys. Rev. Lett.*, 13:321–323, 1964.

[21] Peter Ware Higgs. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett.*, 13:508–509, 1964.

[22] Peter Ware Higgs. Broken symmetries, massless particles and gauge fields. *Phys. Lett.*, 12:132–133, 1964.

[23] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble. Global conservation laws and massless particles. *Phys. Rev. Lett.*, 13:585–587, Nov 1964.

[24] Dirk Dubbers and Michael G Schmidt. The neutron and its role in cosmology and particle physics. Technical Report arXiv:1105.3694, May 2011. Comments: 91 pages, 30 figures, accepted by Reviews of Modern Physics.

[25] J David Bowman, L J Broussard, S M Clayton, M S Dewey, N Fomin, K B Grammer, G L Greene, P R Huffman, A T Holley, G L Jones, C Y Liu, M Makela, M P Mendenhall, C L Morris, J Mulholland, K M Nollett, R W Pattie, S Penttila, M Ramsey-Musolf, D J Salvat, A Saunders, S J Seestrom, W M Snow, A Steyerl, F E Wietfeldt, A R Young, and A T Yue. Determination of the Free Neutron Lifetime. Technical Report arXiv:1410.5311, Oct 2014.

[26] Wikipedia The Free Encyclopedia. Standard Model. `https://en.wikipedia.org/wiki/Standard_Model`, October 2014.

[27] Frank Fiedler. Precision Measurements of the Top Quark Mass.

[28] Carlo Rubbia. Experimental observation of the intermediate vector bosons W+, W-, and Z0. *Rev. Mod. Phys.*, 57(CERN-OPEN-94-004):699–722, 1994. Nobel lecture.

[29] A. Sirlin. Radiative corrections in the $SU(2)_L \times U(1)$ theory: A simple renormalization framework. *Phys. Rev. D*, 22:971–981, Aug 1980.

[30] ALEPH Collaboration, DELPHI Collaboration, L3 Collaboration, OPAL Collaboration, SLD Collaboration, LEP Electroweak Working Group, SLD Electroweak and Heavy Flavour Groups. Precision Electroweak Measurements on the Z Resonance. *Phys. Rep.*, 427(hep-ex/0509008. CERN-PH-EP-2005-041. SLAC-R-774. CERN-L3-304):257. 302 p, Sep 2005.

[31] Studies of theoretical uncertainties on the measurement of the mass of the W boson at the LHC. Technical Report ATL-PHYS-PUB-2014-015, CERN, Geneva, Oct 2014.

[32] Dimitri Yuri Bardin and Giampiero Passarino. *The Standard Model in the Making: Precision Study of the Electroweak Interactions*. Internat. Ser. Mono. Phys. Clarendon Press, Oxford, 1999.

[33] M Awramik, M Czakon, A Freitas, and Georg Weiglein. Precise Prediction for the W-Boson Mass in the Standard Model. *Phys. Rev. D*, 69(hep-ph/0311148. DCPT-2003-146. DESY-03-184. DESY-2003-184. FERMILAB-Pub-2003-239-T. IPPP-2003-73. 5):053006. 10 p, Nov 2003.

[34] Rym Bouchendira, Pierre Clad, Sada Guellati-Khlifa, Franois Nez, and Franois Biraben. New determination of the fine structure constant and test of the quantum electrodynamics. Technical Report arXiv:1012.3627, Dec 2010.

[35] D. B. Chitwood, T. I. Banks, M. J. Barnes, S. Battu, R. M. Carey, S. Cheekatmalla, S. M. Clayton, J. Crnkovic, K. M. Crowe, P. T. Debevec, S. Dhamija, W. Earle, A. Gafarov, K. Giovanetti, T. P. Gorringe, F. E. Gray, M. Hance, D. W. Hertzog, M. F. Hare, P. Kammel, B. Kiburg, J. Kunkle, B. Lauss, I. Logashenko, K. R. Lynch, R. McNabb, J. P. Miller, F. Mulhauser, C. J. G. Onderwater, C. S. Özben, Q. Peng, C. C. Polly, S. Rath, B. L. Roberts, V. Tishchenko, G. D. Wait, J. Wasserman, D. M. Webber, P. Winter, and P. A. Żołnierczuk. Improved measurement of the positive-muon lifetime and determination of the fermi constant. *Phys. Rev. Lett.*, 99:032001, Jul 2007.

[36] M Baak, M Goebel, J Haller, A Hoecker, D Kennedy, R Kogler, K Moenig, M Schott, and J Stelzer. The Electroweak Fit of the Standard Model after the Discovery of a New Boson at the LHC. *Eur. Phys. J. C*, 72(arXiv:1209.2716. DESY-12-154):2205, Sep 2012. Comments: 11 pages, 5 figures, to be submitted to EPJ-C.

[37] The Gfitter Group. A Generic Fitter Project for HEP Model Testing. `http://gfitter.desy.de/`, October 2014.

[38] Max Baak, Jakub Cuth, Johannes Haller, Andreas Hoecker, Roman Kogler, Klaus Moenig, Matthias Schott, and Joerg Stelzer. The global electroweak fit at NNLO and prospects for the LHC and ILC. Technical Report arXiv:1407.3792. DESY-14-124. CERN-OPEN-2014-038, Jul 2014. Comments: 26 pages, 9 figures.

[39] N Besson, M Boonekamp, E Klinkby, S Mehlhase, and T Petersen. Re-evaluation of the LHC potential for the measurement of m_W. Aug 2007. Accepted as Scientific Note SN-ATLAS-2008-070.

[40] Aamodt et al. The ALICE experiment at the CERN LHC. A Large Ion Collider Experiment. *J. Instrum.*, 3:S08002. 259 p, 2008. Also published by CERN Geneva in 2010.

[41] Chatrchyan et at. The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment. *J. Instrum.*, 3:S08004. 361 p, 2008. Also published by CERN Geneva in 2010.

[42] Alves et al. The LHCb Detector at the LHC. *J. Instrum.*, 3(LHCb-DP-2008-001. CERN-LHCb-DP-2008-001):S08005, 2008. Also published by CERN Geneva in 2010.

[43] J. Goodson. Personal Homepage. `http://www.jetgoodson.com`, February 2014.

[44] The ATLAS Collaboration. Official Website of the ATLAS Experiment. `http://www.atlas.ch/`, August 2010.

[45] A Salzburger. Track Simulation and Reconstruction in the ATLAS experiment. Mar 2008.

[46] C.Y. Wong. *Introduction to high-energy heavy-ion collisions.* World Scientific, 1994.

[47] W W M Allison and P R S Wright. The physics of charged particle identification: dE/dx, Cerenkov and transition radiation. (OUNP-83-35):42 p, Jul 1983.

[48] Vitalii Lazarevich Ginzburg and Vadim Nikolaevich Tsytovich. *Transition radiation and transition scattering. Perekhodnoe izluchenie i perekhodnoe rasseianie.* Adam Hilger series on plasma physics. Hilger, Bristol, 1990.

[49] The ATLAS Collaboration. Standard Model Results – ATLAS EXPERIMENT Public Results. `https://twiki.cern.ch/twiki/bin/view/AtlasPublic/StandardModelPublicResults`, December 2014.

[50] D Adams, D Barberis, C P Bee, R Hawkings, S Jarp, R Jones, D Malon, L Poggioli, G Poulard, D Quarrie, and T Wenaus. The ATLAS Computing Model. Technical Report ATL-SOFT-2004-007. ATL-COM-SOFT-2004-009. CERN-ATL-COM-SOFT-2004-009. CERN-LHCC-2004-037-G-085, CERN, Geneva, Dec 2004.

[51] WLCG Collaboration. Worldwide LHC Computing Grid Homepage. `http://wlcg-public.web.cern.ch/`, May 2014.

[52] Maria Grazia Pia and Georg Weidenspointner. Monte Carlo Simulation for Particle Detectors. Technical Report arXiv:1208.0047, Aug 2012. Comments: CERN Council Open Symposium on European Strategy for Particle Physics, 10 - 12 September 2012, Krakow, Poland.

[53] *ATLAS computing: Technical Design Report.* Technical Design Report ATLAS. CERN, Geneva, 2005. revised version submitted on 2005-06-20 16:33:46.

[54] B Lenzi. The physics analysis tools project for the atlas experiment. Technical Report ATL-COM-SOFT-2009-020, CERN, Geneva, Oct 2009. 23/10/2009.

[55] G Barrand, I Belyaev, P Binko, M Cattaneo, R Chytracek, G Corti, M Frank, G Gracia, J Harvey, Eric Van Herwijnen, B Jost, I Last, P Maley, P Mato, S Probst, F Ranjard, and A Yu Tsaregorodtsev. GAUDI: The software architecture and framework for building LHCb data processing applications. 2000.

[56] G A Stewart, W B Breaden-Madden, H J Maddocks, T Harenberg, M Sandhoff, and B Sarrazin. ATLAS Job Transforms. Technical Report ATL-SOFT-PROC-2013-023, CERN, Geneva, Oct 2013.

[57] P Calafiura, C Leggett, D R Quarrie, H Ma, and S Rajagopalan. The storegate: a data model for the atlas software architecture. Technical Report cs.SE/0306089. ATL-SOFT-2003-009, Lawrence Berkeley Nat. Lab., Berkeley, CA, Jun 2003.

[58] Jan-Philip Gehrcke. ATLAS Software: How to run The Full Chain. `http://http://gehrcke.de/2009/06/atlas-software-how-to-run-the-full-chain/`, July 2011.

[59] The ATLAS Collaboration. The ATLAS Simulation Infrastructure. *The European Physical Journal C - Particles and Fields*, 70:823–874, 2010. 10.1140/epjc/s10052-010-1429-9.

[60] The ATLAS Collaboration. ATLAS TWiki. `https://twiki.cern.ch/twiki/bin/view/Atlas/WebHome`, July 2011.

[61] A et al. Valassi. Persistency Framework Official TWiki. `https://twiki.cern.ch/twiki/bin/view/Persistency`, August 2014.

[62] A Vaniachine. Scalable Database Access Technologies for ATLAS Distributed Computing. Technical Report ATL-COM-SOFT-2009-011. ANL-HEP-CP-09-085, CERN, Geneva, Jul 2009. October 2, 2009.

[63] M Dobbs and J B Hansen. The HepMC C++ Monte Carlo Event Record for High Energy Physics. Technical Report ATL-SOFT-2000-001, CERN, Geneva, Jun 2000. revised version number 1 submitted on 2001-02-27 09:54:32.

[64] M Dobbs, JB Hansen, G Lynn, and L Sonnenschein. HepMC – a C++ Event Record for Monte Carlo Generators. `http://lcgapp.cern.ch/project/simu/HepMC/`, August 2014.

[65] T Sjstrand, S Mrenna, and P Z Skands. Pythia 6.4 physics and manual. *J. High Energy Phys.*, 05(hep-ph/0603175. FERMILAB-Pub-2006-052-CD-T. LU-TP-2006-13):026. 570 p, Mar 2006.

[66] Torbjrn Sjstrand, Stephen Mrenna, and Peter Skands. A Brief Introduction to PYTHIA 8.1. *Comput. Phys. Commun.*, 178(arXiv:0710.3820. CERN-LCGAPP-2007-04. LU TP 07-28. FERMILAB-PUB-07-512-CD-T):852867. 27 p, Oct 2007.

[67] Gennaro Corcella, I G Knowles, G Marchesini, S Moretti, K Odagiri, Peter Richardson, Michael H Seymour, and Bryan R Webber. Herwig 6.5 release note. herwig 6.5. Technical Report hep-ph/0210213. CAVENDISH-HEP-2002-17. CERN-TH-2002-270. DAMTP-2002-124. IPPP-2002-58, CERN, Geneva, Oct 2002.

[68] A. Sherstnev and R.S. Thorne. Parton distributions for lo generators. *The European Physical Journal C*, 55(4):553–575, 2008.

[69] The Geant4 Collaboration. Geant4–a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250 – 303, 2003.

[70] S SNYDER, P VAN GEMMEREN, M NOWAK, T EIFERT, A BUCKLEY, M ELSING, D GILLBERG, E MOYSE, K KOENEKE, and A KRASZNAHORKAY. The Run 2 ATLAS Analysis Event Data Model. Aug 2014.

[71] The ROOT Team. Official ROOT Website. `http://root.cern.ch/`, May 2011.

[72] The ATLAS Collaboration. Charged-particle multiplicities in pp interactions at $\sqrt{s} = 900$ gev measured with the atlas detector at the lhc. *Physics Letters B*, 688(1):21 – 42, 2010.

[73] Matteo Volpi. *Charged particle multiplicities in pp interactions at sqrt(s) = 900 GeV and sqrt(s) = 7 TeV measured with the ATLAS detector at the LHC.. oai:cds.cern.ch:1331501.* PhD thesis, Barcelona, IFAE, Barcelona, 2010. Presented 16 Dec 2010.

[74] T Yamanaka. The atlas calorimeter simulation fastcalosim. Technical Report ATL-SOFT-PROC-2011-021, CERN, Geneva, Jan 2011.

[75] Standard Performance Evaluation Corporation. Geant4 Collaboration Website. `http://geant4.cern.ch/`, April 2014.

[76] K Edmonds, S Fleischmann, T Lenz, C Magass, J Mechnich, and A Salzburger. The Fast ATLAS Track Simulation (FATRAS). Technical Report ATL-SOFT-PUB-2008-001. ATL-COM-SOFT-2008-002, CERN, Geneva, Mar 2008.

[77] S Hamilton, E Kneringer, W Lukas, E Ritsch, A Salzburger, K Sliwa, S Todorova, J Wetter, and S Zimmermann. The atlas fast track simulation project. Technical Report ATL-SOFT-PROC-2011-038, CERN, Geneva, Mar 2011.

[78] Performance of the Fast ATLAS Tracking Simulation (FATRAS) and the ATLAS Fast Calorimeter Simulation (FastCaloSim) with single particles. Technical Report ATL-SOFT-PUB-2014-001, CERN, Geneva, Mar 2014.

[79] A Salzburger, S Todorova, and M Wolter. The atlas tracking geometry description. Technical Report ATL-SOFT-PUB-2007-004. ATL-COM-SOFT-2007-009, CERN, Geneva, Jun 2007.

[80] Michael Duehrssen and Karl Jakobs. *Study of Higgs bosons in the WW final state and development of a fast calorimeter simulation for the ATLAS experiment.* PhD thesis, Freiburg U., Freiburg, 2009. Presented on 21 Dec 2009.

[81] A Arce, M Beckingham, M Duehrssen, E Schmidt, M Shapiro, M Venturi, J Virzi, I Vivarelli, M Werner, S Yamamoto, and T Yamanaka. The simulation principle and performance of the atlas fast calorimeter simulation fastcalosim. Technical Report ATL-COM-PHYS-2010-838, CERN, Geneva, Oct 2010.

[82] The ATLAS Collaboration. Data/MC Comparison for Calorimeter Shower Shapes of High Et Electrons. `http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/EGAMMA/PublicPlots/20111005/ATL-COM-PHYS-2011-1299/index.html`, October 2011.

[83] Martin Woudstra. ConfigurableFactory update + a few related topics, ATLAS Software & Computing Workshop. `https://indico.cern.ch/event/119169/session/17/contribution/117/material/slides/0.pdf`, April 2011.

[84] Roland Jansky, Emmerich Kneringer, and Andreas Salzburger. *Truth Seeded Reconstruction for Fast Simulation in the ATLAS Experiment.* PhD thesis, Innsbruck U., 2013. Presented 30 Sep 2013.

[85] Ritsch E on behalf of the ATLAS Collaboration. Concepts and plans towards fast large scale monte carlo production for the atlas experiment. *Phys.: Conf. Ser.*, 524(012035 `http://dx.doi.org/10.1088/1742-6596/523/1/012035`), 2014.

[86] J.H. Hubbell. Electron & positron pair production by photons: A historical overview. *Radiation Physics and Chemistry*, 75(6):614 – 623, 2006. Pair Production.

[87] Eberhard Haug and Werner Nakel. *The elementary process of Bremsstrahlung*. World Scientific Lecture Notes in Physics. World Scientific, New Jersey, NJ, 2004.

[88] Elmar Ritsch, A Salzburger, and E Kneringer. *Fast Calorimeter Punch-Through Simulation for the ATLAS Experiment*. PhD thesis, Innsbruck U., Innsbruck, 2011. Presented 28 Sep 2011.

[89] D Rousseau, G Dimitrov, I Vukotic, O Aidel, RD Schaffer, and S Albrand. Monitoring of computing resource utilization of the ATLAS experiment. Technical Report ATL-SOFT-PROC-2012-034, CERN, Geneva, May 2012.

[90] N Chauhan, G Kabra, T Kittelmann, R Langenberg, R Mandrysch, A Salzburger, R Seuster, E Ritsch, G Stewart, N van Eldik, and R Vitillo. ATLAS Offline Software Performance Monitoring and Optimization. Technical Report ATL-SOFT-PROC-2013-040, CERN, Geneva, Oct 2013.

[91] gperftools Homepage. `https://code.google.com/p/gperftools/`, September 2014.

[92] J Weidendorfer. kcachegrind Homepage. `http://kcachegrind.sourceforge.net/`, September 2014.

[93] Intel Corporation. Intel® Math Library Official Webpage. `https://software.intel.com/en-us/node/522653`, November 2014.

[94] C. Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

[95] W. Kutta. Beitrag zur naherungsweisen integration von differentialgleichungen. *Z. Math. und Phys.*, 46:435–453, 1901.

[96] Encyclopedia of Mathematics. Runge Kutta Method. `http://www.encyclopediaofmath.org/index.php/Runge%E2%80%93Kutta_method`, November 2014.

[97] Troels Petersen. Measurement of the W mass with the ATLAS detector. Technical Report ATL-PHYS-PROC-2009-040. ATL-COM-PHYS-2008-198, CERN, Geneva, Oct 2008.

[98] N Besson. W mass measurement in the ATLAS experiment. Technical Report ATL-PHYS-PROC-2009-134, CERN, Geneva, Nov 2009.

[99] Nansi Andari, Maarten Boonekamp, Jean-Baptiste Blanchard, and Nenad Vranjes. Measurement of m_W at 7 TeV: Z-based cross check measurements. Technical Report ATL-COM-PHYS-2014-1437, CERN, Geneva, Nov 2014.

[100] V M Abazov. A novel method for modeling the recoil in W boson events at hadron collider. Technical Report arXiv:0907.3713. FERMILAB-PUB-09-363-E, Jul 2009. Comments: 26 pages, 24 figures, submitted to Nuclear Instruments and Method Sect. A.

[101] The D0 Collaboration. Measurement of the W Boson Mass with 1 fb^{-1} Run II Data. Technical Report D0 Note 5893-CONF, May 2009.

[102] John R Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements; 2nd ed.* Books in physics. University Science Books, Sausalito, CA, 1997.

[103] J Wolberg. *Data analysis using the method of least squares: extracting the most information from experiments.* Springer, Heidelberg, 2006.

[104] F Dydak, M W Krasny, and R Voss. The measurement of the W mass at the LHC: shortcuts revisited. Technical Report CERN-LHCC-2009-014. LHCC-I-017, CERN, Geneva, Sep 2009. This LOI has been submitted as well to the SPSC: CERN-SPSC-2009-028 and SPSC-I-239.

[105] N Besson and M Boonekamp. Determination of the Absolute Lepton Scale Using Z Boson Decays: Application to the Measurement of MW. Technical Report ATL-PHYS-PUB-2006-007. ATL-COM-PHYS-2005-072, CERN, Geneva, Nov 2005.

[106] Matthias Schott. *Study of the Z Boson Production at the ATLAS Experiment with First Data.* PhD thesis, Ludwig-Maximilians-Universität München, 2007.

[107] Sidney D. Drell and Tung-Mow Yan. Massive lepton-pair production in hadron-hadron collisions at high energies. *Phys. Rev. Lett.*, 25:316–320, Aug 1970.

[108] The ATLAS Collaboration. Electron and photon energy calibration with the ATLAS detector using LHC Run 1 data. *Eur. Phys. J. C*, (arXiv:1407.5063. CERN-PH-EP-2014-153):74. 51 p, Jul 2014. Comments: 39 pages plus author list + cover pages (51 pages total), 42 figures, 8 tables, published in EPJC, All figures including auxiliary figures are available at http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/PERF-2013-05/.

[109] J-B Blanchard. Performance of the reconstruction, calibration and identification of electrons and photons with the ATLAS detector, and their impact on the ATLAS physics results. Technical Report ATL-PHYS-PROC-2014-207, CERN, Geneva, Oct 2014.

[110] Performance of Missing Transverse Momentum Reconstruction in ATLAS with 2011 Proton-Proton Collisions at $sqrts = 7$ TeV. Technical Report ATLAS-CONF-2012-101, CERN, Geneva, Jul 2012.

[111] Performance of Missing Transverse Momentum Reconstruction in ATLAS studied in Proton-Proton Collisions recorded in 2012 at 8 TeV. Technical Report ATLAS-CONF-2013-082, CERN, Geneva, Aug 2013.

[112] N D Gagunashvili. Comparison of weighted and unweighted histograms. Technical Report physics/0605123, May 2006.

[113] L Garren, I.G. Knowles, S Navas, P Richardson, T Sjöstrand, and T Trippe. Monte carlo particle numbering scheme. Technical report, Jun 2006.

[114] W Bhimji, J Cranshaw, P van Gemmeren, D Malon, R D Schaffer, and I Vukotic. The ATLAS ROOT-based data formats: recent improvements and performance measurements. Technical Report ATL-SOFT-PROC-2012-020, CERN, Geneva, May 2012.

[115] John Apostolakis, Ren Brun, Federico Carminati, and Andrei Gheata. Rethinking particle transport in the many-core era towards GEANT 5. *J. Phys.: Conf. Ser.*, 396:022014, 2012.

[116] D Crooks, P Calafiura, R Harrington, S Purdie, H Severini, S Skipsey, V Tsulaia, and A Washbrook. Multi-core job submission and grid resource scheduling for ATLAS AthenaMP. Technical Report ATL-SOFT-PROC-2012-029, CERN, Geneva, May 2012.

[117] The C++ Resources Network. STL Containers. `http://www.cplusplus.com/reference/stl/`, November 2014.

[118] S Spagnolo, K A Assamagan, J Boudreau, S Baranov, V Tsulaia, A Nairz, I Trigger, C Bourdarios, G Unal, M Lelchuk, B Seligman, J Tth, P Strzenec, D Costanzo, G Gorfine, T H Kittelmann, Y Hasegawa, D Pomarde, A Vaniachine, and A Zalite. The Description of the Atlas Detector. 2005.

[119] A Vaniachine, S Eckmann, D Malon, P Nevski, and T Wenaus. Primary Numbers Database for ATLAS Detector Description Parameters. Technical Report cs.DB/0306103. ANL-HEP-CP-2003-050, Argonne Nat. Lab., Argonne, IL, Jun 2003.

[120] HL-LHC Project. HL-LHC: High Luminosity Large Hadron Collider. `http://hilumilhc.web.cern.ch`, May 2014.

Leopold-Franzens-Universität Innsbruck

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht als Magister-/Master-/Diplomarbeit/Dissertation eingereicht.

10.12.2014	[illegible]
Datum	Unterschrift