

FAST CALORIMETER PUNCH-THROUGH SIMULATION FOR THE ATLAS EXPERIMENT

Diplomarbeit

in der Studienrichtung Physik
zur Erlangung des akademischen Grades
Magister der Naturwissenschaft (Mag.rer.nat.)

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

von
Elmar Ritsch
elmar.ritsch@uibk.ac.at

Betreuer der Diplomarbeit:
Dr. Andreas Salzburger, CERN
Ao.Univ.-Prof. Dr. Emmerich Kneringer, Institut für Astro- und Teilchenphysik

Innsbruck, September 2011



Abstract

This work discusses the parametrization, implementation and validation of a tuneable fast simulation of hadronic leakage in the ATLAS detector. It is dedicated to simulate calorimeter punch-through and decay in flight processes inside the ATLAS calorimeter. Both effects can cause systematic errors in muon reconstruction and identification. Therefore a correct description of these effects is crucial for many physics studies involving muons. The parameterized punch-through simulation is integrated into the fast ATLAS detector simulations Fatras and AtlfastII, respectively. The Fatras based simulation of single pions shows a good agreement with results obtained by the full Geant4 detector simulation – especially in the context of a fast simulation. It is shown that for high energy multi jet events, simulated with the AtlfastII implementation, the muon reconstruction rates show a good agreement with the Geant4 simulated reference.

Acknowledgements – Danksagung

Beginnen möchte ich meine Danksagung bei meinen Betreuern Dr. Andreas Salzburger und Prof. Dr. Emmerich Kneringer. Ihr habt mich seit meinen Anfängen in der Teilchenphysik engagiert unterstützt. Gemeinsam haben wir unzählige Diskussionen geführt, die mir sowohl bei meiner Arbeit von großer Hilfe waren, als auch, mich weit über die Physik hinaus inspiriert haben. Überaus dankbar bin ich auch für euer Engagement bei der Durchsicht dieser Arbeit, ebenso wie für die vielen hilfreichen Kommentare dazu.

Prof. Dr. Dietmar Kuhn gebührt besonderer Dank, für die entgegengebrachte Unterstützung ab der ersten Minute als Sommerstudent und die überaus freundliche Leitung der Teilchenphysik Gruppe in Innsbruck. Ich möchte mich auch bei der gesamten Arbeitsgruppe bedanken, für die herausragende Hilfsbereitschaft und die äußerst angenehme Atmosphäre in der Gruppe.

Meinen Bürokollegen und Papierflieger-Piloten Patrick Jussel, Michael Werner, Jocelin Perez und Klaus Reitberger möchte ich besonders Danken, für die meist heitere Atmosphäre in unserem gemeinsamen Reich.

Speziellen Dank möchte ich auch gegenüber Michael Dührssen und Wolfgang Lukas aussprechen, welche mir in vielerlei Hinsicht bei der Umsetzung meiner Diplomarbeit geholfen haben.

Nicht zuletzt gebührt besonderer Dank meinen Eltern, welche mir das Studium nicht nur nahe legten, sondern mich über die gesamte Dauer meines Studiums tatenreich unterstützten.

Vielen Dank Kathi, dass du auch in sehr arbeitsreichen Zeiten verständnisvoll zu mir gestanden hast.

Contents

Abstract	i
Acknowledgements – Danksagung	iii
1 Introduction and Motivation	1
2 The ATLAS Experiment	3
2.1 Coordinate System	4
2.2 The Inner Detector	5
2.3 The Calorimeter	5
2.4 The Muon Spectrometer or Muon System	6
2.5 Particle Signatures	6
3 Detector Simulation in ATLAS	9
3.1 Simulation Scheme	9
3.2 Full and Fast Detector Simulation	13
3.2.1 Geant4	14
3.2.2 Fatras – Fast ATLAS Track Simulation	14
3.2.3 AtlfastII	16
3.3 The Athena Framework	16
3.3.1 Athena Application Flow	17
3.3.2 StoreGate	18
3.3.3 Data Formats	19
4 Calorimeter Punch-Through – Leakage into the Muon Spectrometer	21
4.1 Calorimeter Punch-Through	21
4.2 Processes in the ATLAS Calorimeter	24
4.2.1 Electromagnetic Showers	24
4.2.2 Muons Traversing Dense Material	25
4.2.3 Hadronic Showers	25
4.3 Geant4 Analysis	27
4.3.1 Simulation Setup and Event Selection	27
4.3.2 Punch-Through Particle Types	28
4.3.3 Punch-Through Occurrence and Number of Particles	30

4.3.4	Particle Type Correlations	31
4.3.5	Particle Energies	33
4.3.6	Deflection Angles $\Delta\phi$ and $\Delta\theta$	34
4.3.7	Particle Momentum Direction ($\Delta\phi_p$ and $\Delta\theta_p$)	36
4.4	Parametrization	38
4.4.1	Fit Quality Measure	38
4.4.2	Parametrization of Punch-Through Particle Quantities and Particle Types	39
4.4.3	Parametrization of Particle Correlations	41
4.4.4	Parametrization of Particle Energy E_{pt}	44
4.4.5	Parametrization of Particle Deflection Angles $\Delta\phi$ and $\Delta\theta$	45
4.4.6	Parametrization of Particle Momentum Direction ($\Delta\phi_p$ and $\Delta\theta_p$)	50
4.5	Parametrized Simulation	52
4.5.1	Simulation Input and Output	53
4.5.2	Simulation Parameters	54
4.5.3	Simulation Scheme	55
5	Implementation	59
5.1	Punch-Through Simulation	59
5.1.1	Number of Punch-Through Particles, Particle Type and Correlations	61
5.1.2	Energy of Punch-Through Particles	61
5.1.3	Deflection Angles $\Delta\phi$ and $\Delta\theta$	62
5.1.4	Particle Position and Momentum Direction	62
5.2	Distributed Random Number Generation	63
5.2.1	Discrete Random Numbers	64
5.2.2	Continuous Random Numbers	64
5.2.3	Parameter Interpolation	64
5.3	Integration into Fatras	66
5.3.1	Fatras Simulation Scheme	66
5.3.2	Fatras Simulation Kernel	69
5.3.3	Track Simulation	69
5.3.4	Particle Transport	70
5.3.5	Particle Extrapolation	70
5.3.6	Fatras Calorimeter Simulation	70
5.4	Integration into AtlfastII	71
6	Results	73
6.1	Comparison to Full Simulation	73
6.1.1	Single Particle Validation	73
6.1.2	High Energy Jet Events	80
6.2	CPU Performance	84
7	Conclusion and Outlook	87

A	The Look-up Table	89
A.1	Input for the <code>Fatras::PDFcreator</code> C++ Class	89
A.2	Particle Type Correlations	90
B	CaloEntry and MuonEntry	93
	Bibliography	95

Chapter 1

Introduction and Motivation

Particle physics might be the most fundamental approach taken in natural science in order to understand the laws of nature. It concerns the study of the *most fundamental* forces and processes acting upon the smallest parts of matter, generally named *particles*. A number of different sciences benefit from the understanding gathered in particle physics and its experiments: from radiation therapy in medicine up to the understanding of the development of the (early) universe.

Numerous particle physics experiments are dedicated to measure particle properties in order to understand the underlying laws of nature. This work concerns the ATLAS experiment at the Large Hadron Collider at CERN. In the LHC, collisions of high energetic protons (or lead ions) result in a high number of newly created particles. Most of these particles subsequently traverse the ATLAS detector, which is dedicated to measure different particle properties. From these, details about the underlying processes involved in the creation of the particles can be reconstructed and compared to theoretical predictions.

Modern particle physics relies as much on computer simulations as on recorded detector measurements. Simulations are used in order to describe the detector output as accurately as possible, based on the best knowledge of the underlying (particle-) physics models and detector description. Theoretical models in particle physics can predict certain properties of particles produced in preceding particle collisions (proton-proton or lead ions in case of the LHC). However, these particles will traverse parts of the surrounding detector, during which they will interact in many different ways with its material, or decay into other particles. Due to the complexity and stochastical behaviour of these numerous different processes and the highly complex design of modern high energy physics detectors, it is nearly impossible to predict the detector output by hand – even if the underlying processes are fully understood. Therefore software tools are used to simulate the most relevant processes and their impact on measurements, when particles traverse such a particle detector.

Some physics studies require a high number of simulated collision events, partly for background studies that often dominate the analyses, or in order to estimate systematic uncertainties due to various mismodelling of the experimental setup. However, full detail

simulations may be very time consuming and they can become a limiting factor when trying to analyse recorded collision data. Therefore different fast simulation approaches are taken, in order to speed up the generation of simulated reference data. This becomes particularly important when trying to *scan* a parameter space (inherent to some theoretical models) in order to find a set of model parameters corresponding with the measurements in the detector.

In this work a tunable fast simulation module is presented which allows to simulate calorimeter punch-through effects. The calorimeter is one major sub-detector of the ATLAS experiment. It measures particle energies, by absorbing different kinds of incident particles. However, due to different processes the confinement of the incident particle energy is not always guaranteed. Therefore shower particles created inside the calorimeter may leave the dense calorimeter material and penetrate surrounding sensitive detector parts – the ATLAS muon spectrometer. Similarly, muons produced in decays taking place inside the calorimeter may penetrate the bulk material and also escape into the muon system. The muon spectrometer plays a crucial role in the measurement and identification of muons traversing the ATLAS detector. Therefore calorimeter punch-through effects will show up as systematic errors in muon measurements. Thus physics studies using properties of identified muon particles will be affected by this very effect. Two such examples would be the study of the Higgs particle decaying into four leptons $H \rightarrow ZZ^{(*)} \rightarrow 4l$ [1] or the measurement of high energy b -jets [2]. In this context, a tunable simulation is of great benefit: it allows for systematic studies on the effect on the analysis with changing properties of the punch-through component.

Chapter 2

The ATLAS Experiment

The ATLAS [3] (A Toroidal LHC ApparatuS) experiment is situated in a cavern, approximately 100 meters underground, at point 1 at the LHC (Large Hadron Collider) at CERN near Geneva, Switzerland. It is a particle detector which measures particles produced by collisions of high-energetic protons or lead ions. It is one of four main experiments along the LHC: ALICE, ATLAS, LHCb and CMS (see figure 2.1). The LHC is installed in the same tunnel which was already used for the LEP (Large Electron Positron Collider) accelerator until the year 2000. The LHC accelerates protons up to energies of 7 TeV which are then brought into collision at the different experiment sites mentioned above.

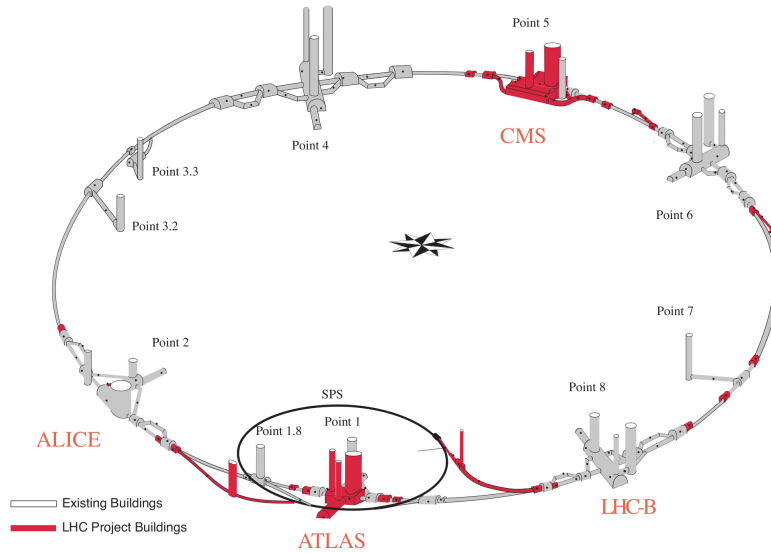


Figure 2.1: Overview of the LHC ring at CERN showing the position of the LHC experiments ALICE, ATLAS, LHCb and CMS. The buildings and tunnels constructed after the LEP era are shown in red [4].

In order to study the physical process of the proton-proton collision, one looks at the particle signatures recorded in the detector. These collisions happen in the center of the ATLAS detector, close to the pixel detector (see figure 2.2). Particles created in the proton-proton (lead ion) collisions traverse the detector and interact with active and passive detector components. The signals created in active detector parts are further referred to as *detector hits* or simply *hits*. In the ATLAS inner detector and muon spectrometer, these signals are used to reconstruct particle trajectories (further called *tracks*) through the detector. The detector hits are also used to reconstruct the energy deposited by the particles traversing the ATLAS calorimeter. From this, particle properties like energy, momentum and charge can be determined, which allow to study the preceding interaction process.

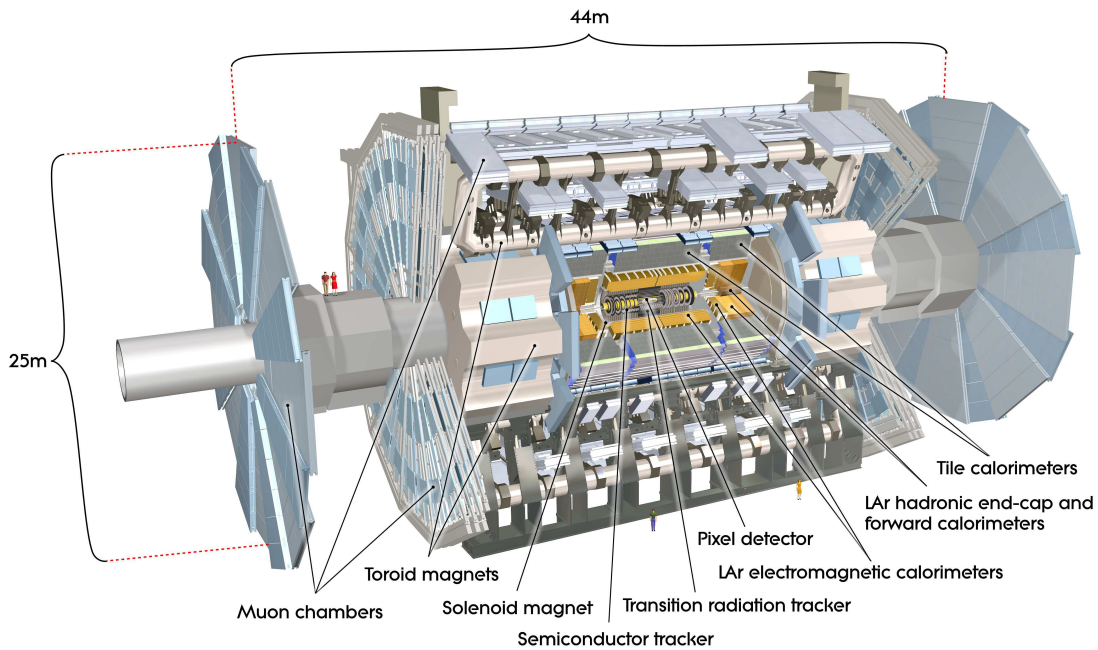


Figure 2.2: Overview of the ATLAS detector and its subdetectors. In order to see the inner structure, the front half of the ATLAS detector is cut away in this image.

2.1 Coordinate System

The global coordinate system [3, 5] applied in the ATLAS experiment is defined to have 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. 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 $\theta = \pi$.

Pseudorapidity η is the rapidity [6] of a particle in the approximation that the particle is massless:

$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right] \quad (2.1)$$

2.2 The Inner Detector

The inner detector (ID) is the innermost layer of the ATLAS experiment and therefore closest to the collision point. Thus particles created in collisions first pass through the ID before they go through any other part of the ATLAS detector.

The main purpose of the ID is to track charged particles in order to estimate their momenta and production vertices. It consists of three subdetectors: a silicon *Pixel Detector*, *Semiconductor Tracker (SCT)* using silicon strip technology and the *Transition Radiation Tracker (TRT)*, which implements a drift tube design.

The ID is embedded in a solenoidal magnetic field with a central strength of 2 Tesla which bends the trajectories of charged particles. From this curvature the particle's charge over momentum-component perpendicular to the magnetic field ratio q/p_T can be measured.

2.3 The Calorimeter

Particles origination at the interaction point enter the ATLAS calorimeter system, after they have passed the ID.

It's main purpose is to measure particle energies. The calorimeter consists of two parts, the (inner) *electromagnetic calorimeter* and the (outer) *hadronic calorimeter*. In order to measure particle energies, incident particles are tried to be *stopped* inside the calorimeter. Therefore the incident particle is provoked to cause a particle shower (see section 4.2), such that the energy deposited by the shower can be measured. In order to do so, the calorimeter is build of very heavy material (such as lead) which enhances the chance to confine most shower particles (shower energy). The deposited energy is used in order to reconstruct the initial particle energy, which implies certain calibration corrections for the energy lost upstream the calorimeter and the fraction of energy actually measured.

The electromagnetic calorimeter measures photon and electron energies, where the hadronic calorimeter measures hadron energies. Different particle types will cause different showers shapes, which are used in order to do particle identification.

In general, muons do not interact much with the calorimeter material. Their interaction is mainly limited to multiple Coulomb scattering and ionization energy loss. Thus,

muons (with energies above approximately 4 GeV) will likely pass the calorimeter and reach the next ATLAS sub-detector, the muon spectrometer.

2.4 The Muon Spectrometer or Muon System

The muon system (MS) is the outermost part of the ATLAS experiment. Like the ATLAS ID it is a particle tracking device aimed to measure particle trajectories. Its name comes from the fact that mostly muons will reach the MS to cause detector hits on sensitive detector parts. In an ideal case, other particle types should either be stopped in the calorimeter (hadrons, electrons, photons) or do not create any signal in the MS (neutrinos). However, calorimeter punch-through and decay in flight effects in the calorimeter do have a significant impact on MS measurements (section 6).

Throughout the muon system a toroidal magnetic field is applied in order to measure each particle's charge over momentum ratio q/p . Combining this measurement with the corresponding ID q/p_T measurement, a higher accuracy on the overall q/p_T is obtained for these particles.

2.5 Particle Signatures

Due to the previously described properties of the individual ATLAS detector components, different particle types will leave distinct *signatures* inside the ATLAS detector. This allows for particle type identification, which is one of the main reasons for the chosen design. Figure 2.3 shows typical signatures for some common particle types.

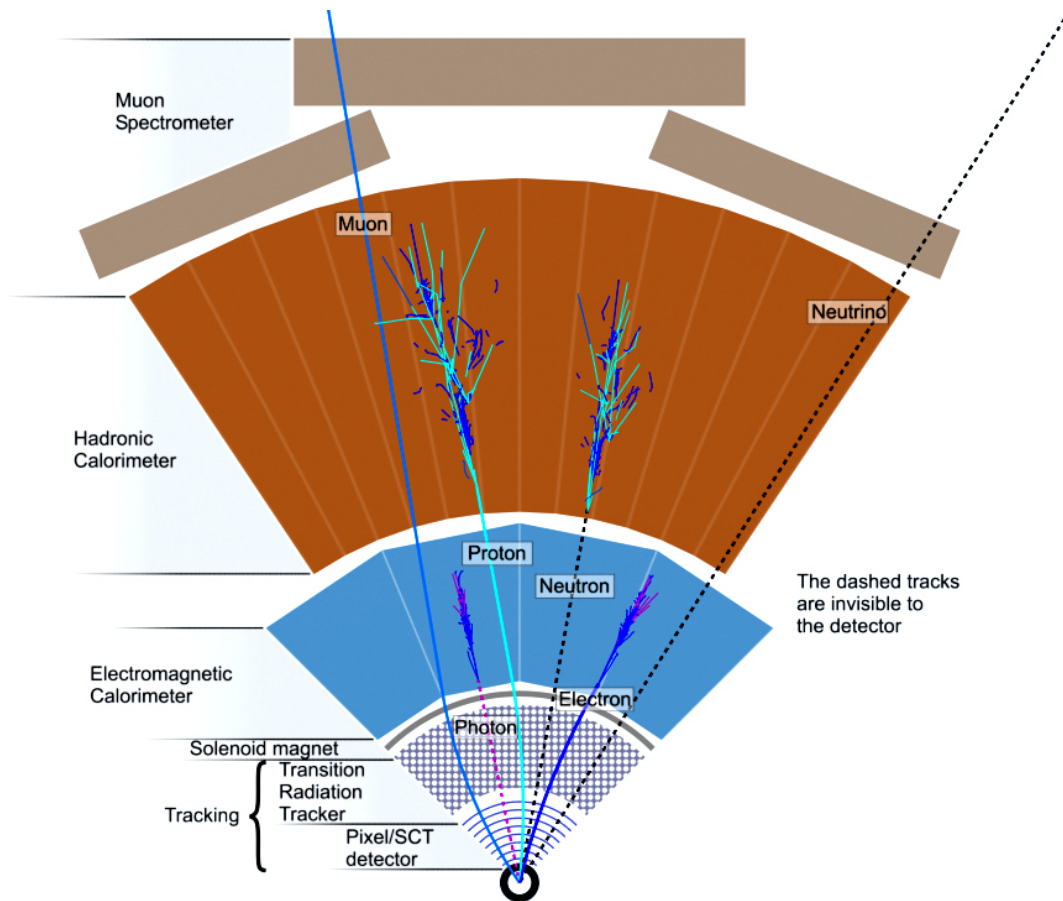


Figure 2.3: Particle signatures for different particle types when traversing the ATLAS detector. The shown particles originate at the interaction point and traverse the ATLAS detector in a radial direction. Charged particle trajectories are bent in the ID and MS due to the magnetic fields present.

Chapter 3

Detector Simulation in ATLAS

In this chapter we discuss how the ATLAS collaboration handles (detector) simulations which is carried out in the software framework Athena. Section 3.1 discusses the typical simulation scheme in ATLAS. In section 3.2 two different simulation concepts are introduced: full and fast detector simulation. Both concepts find their use in the ATLAS collaboration. The concept of fast simulations is particularly important for this work due to the fact that such a simulation is implemented and discussed in later chapters. Section 3.3 gives an overview of the Athena framework, which is used by the ATLAS collaboration for simulation-, digitization- and reconstruction and algorithms. Analysis algorithms may also use the Athena framework, but many examples of standalone analysis algorithms do exist.

3.1 Simulation Scheme

This section describes the different steps necessary for the generation of simulated event data. In order to compare simulated data with recorded data, a common data format is required at a certain stage. For the ATLAS experiment, common data formats are used for the event reconstruction and all following steps. A brief introduction into the ATLAS *full simulation chain* and how to set it up is given in [7]. Further details can be found in [8] and [9].

Figure 3.1 gives an overview of the *standard* full simulation chain used in ATLAS. It is a very detailed full Monte Carlo simulation, based on the Geant4 [11] toolkit (see section 3.2.1). However, any other detector simulation (such as Fatras in section 3.2.2 or AtlfastII in section 3.2.3) also has to apply all points of this scheme. Even though some steps might not use the standard ATLAS algorithms but might be treated internally in either simulation.

The following paragraphs will discuss each simulation step:

Event Generation: The basis of each event simulation is a physics event generator. These event generators create data objects representing particles, which will later be given as input to the detector simulation. The properties of these *particles* are based on

theoretical (and computational) models which are to be tested against data – such as the standard model of particle physics or supersymmetry etc. Yet, due to the complexity of the physical processes occurring in proton proton collision, some event generators apply certain simplifications and must be *tuned* via a set of parameters. Therefore different tunes may exist which will even evolve over time. Changing the settings of an event generator usually leads to a re-run of this first step in the simulation chain which consequently leads to a re-run of *all* subsequent steps as well.

Event generators usually take over a set of input parameters, such as the centre of mass energy of the initial colliding protons or other physical properties.

Event generators typically used in ATLAS are Pythia [12] and Herwig [13].

Detector Simulation: The next step in the simulation chain is the simulation of the detector response to the particles created in the event generation. The fastest detector simulations (such as e.g. Atlfast) use a parametric smearing to model the signal a detector might give to a certain particle. However, in more complex simulation the particles are transported through the detector and the detector response is then emulated. Thereby the simulation computes the paths the particles take while traversing the detector. In order to do so any kind of interaction with the detector material or possible decays of unstable particles are taken into account. In addition to that, the detector simulation computes and stores the particle hits on sensitive detector elements. These detector hits will be stored in an output format known as simulation HITS (see section 3.3.3).

The detector simulation usually ends when all particles are either *stopped* or left the detector volume. In this case stopped means that a particle's energy has dropped below a certain threshold. Particles for which this is the case will not have a significant impact to the overall simulation, and therefore the simulation will not treat this particle any further.

The most common detector simulation used by the ATLAS collaboration is based on Geant4 (see section 3.2.1), but also fast simulation approaches exist.

The implementation carried out in this work is a detector simulation module which may be plugged into any fast ATLAS detector simulation, that does not fully deploy the particle showering in the calorimeter.

Digitization: After the detector simulation step, the simulation hits need to be *translated* into a data format which corresponds to a format retrieved from the detector. For the simulation chain, this translation is carried out by the digitization. Digitized simulation data correspond to bytestream converted data retrieved from the detector. The digitization aims to transform the primary interaction of a particle with the sensitive detector material into measurable quantities, such as the charge drifted to the readout modules in tracking detectors or the energy measured in photomultipliers as present in detector setups. The digitization output format is RDO (RAW Data Objects, see section 3.3.3) which is the exact same data format used to record detector measurements after bytestream conversion. Therefore all further steps are non-simulation-specific and need

to be carried out for recorded events as well.

Besides creating realistic detector output signals, the digitization is responsible for describing event *pile-up*¹ correctly. It does so by overlaying detector simulations of different events and merging them into one common RDO output for a single event.

Reconstruction: The RDO data obtained by the previous simulation step (or from detector measurements) now need to be interpreted in terms of finding particle trajectories and energy depositions. Numerous reconstruction algorithm use different approaches in order find particle tracks (connecting detector hits with realistic particle trajectories), measure particle momenta and energy (track curvature and calorimeter measurements) and do particle identification (various methods using preceding reconstruction results).

The results obtained by the event reconstruction contains all properties needed for subsequent physics analyses. Reconstructed data are stored in an Event Summary Data (ESD) or Analysis Object Data (AOD) data format (see section 3.3.3), depending on the level of detail needed for the subsequent studies. However, AOD is the basis for many physics studies.

¹*pile-up*: in-time and out-of-time inelastic scattering events from additional collisions in the same bunch-crossing and from the previous bunch-crossing

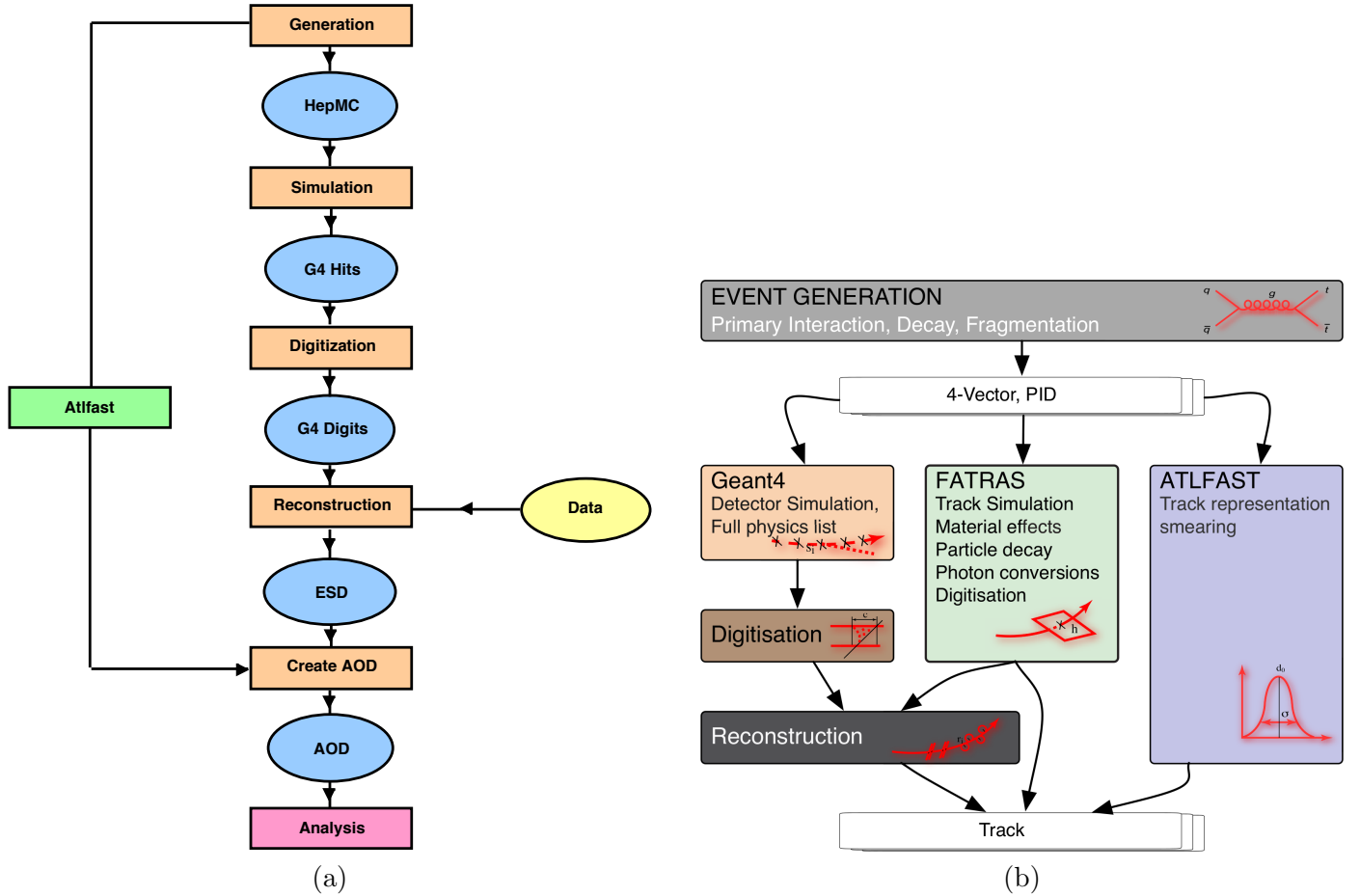


Figure 3.1: Application flow of different simulation chains used in ATLAS: (a) The data flow of the standard Geant4 *full chain* explained in comparison with Atlfast and detector data. Elliptical-shaped boxes represent persistent data objects or -collections whereas rectangular boxes are algorithms [10, 7]. (b) Schematic comparison of the application flow of different ATLAS detector simulation software: Geant4, Fatras and Atlfast [5].

3.2 Full and Fast Detector Simulation

Figure 3.1 (b) compares the application flow of three different detector simulations used in ATLAS: Geant4, Fatras and Atlfast. Geant4 is the the main (most detailed) detector simulation utilised by the ATLAS collaboration. Fatras serves as an alternative fast detector simulation for the ATLAS inner detector and muon spectrometer, and Atlfast was mainly used during the design phase of the ATLAS detector as an ultra fast detector simulation. AtlfastII is a new fast simulation approach combining Geant4 with the fast calorimeter simulation FastCaloSim. Geant4, Fatras and AtlfastII will be discussed in the following sections.

The motivation to develop and use fast detector simulations becomes evident when looking at the simulation time for *single* particle events² in Geant4 based ATLAS detector simulations (figure 3.2). Running the detector simulation with one low energetic electron as initial particle, already takes more than one second on average. Evidently this becomes worse for higher energetic particles or events containing more than one initial particle. As has been shown in minimum bias studies on data at a centre-of-mass energy of $\sqrt{s} = 900$ GeV, a number of up to $n_{ch} \approx 60$ charged particles (with $p_T > 500$ MeV and $|\eta| < 2.5$) were measured per event in the ATLAS inner detector [14]. For higher centre-of-mass energies even more inner detector particles appear: up to $n_{ch} \approx 90$ in $\sqrt{s} = 7$ TeV minimum bias studies [15]. In general, more than 90% of the simulation time is spend inside the calorimeter [16]. Therefore the corresponding detector simulations will spend lots of CPU time on the particle showers created by each of these ID particles when entering the calorimeter. Thus, simulation time becomes a concern for physics studies which require a high number of simulations (e.g. supersymmetry studies which need to scan a parameter space in their physical model). A detailed comparison of simulation time for the different ALTAS detector simulations is given in [8].

²*single particle event*: Events with one initial particle. Detector simulations of such events will usually process more than one particle. This is due to numerous particles created in interaction processes of the initial particle with detector material (e.g. particle showers inside the calorimeter).

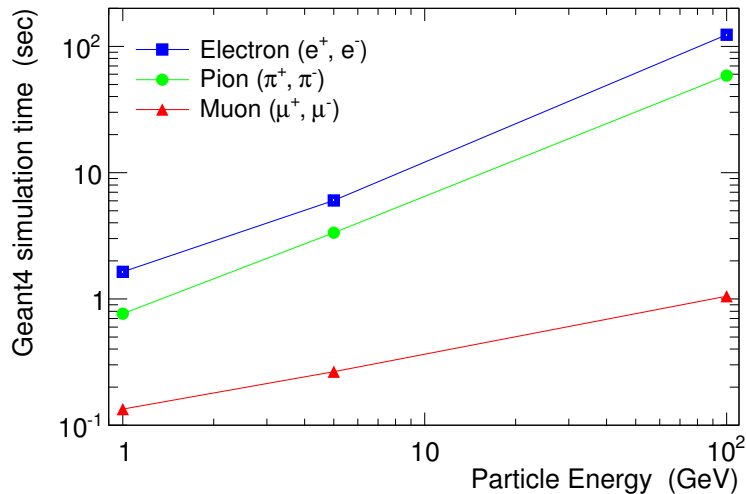


Figure 3.2: Average simulation time in seconds of single particle events using the Geant4 based full ATLAS detector simulation (50k events per data point, ATLAS offline software release 16.0.3.2 with detector description ATLAS-GE0-16-00-00 on a Intel Xeon X5570 @ 2.93 GHz CPU)

3.2.1 Geant4

Geant4 (G4) is a widely used toolkit for the simulation of particles traversing matter. Its main applications are found in detector simulations for high energy physics, radiation simulation in medical sciences and various 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 a number of physics models describing many different kinds of particle-matter interactions. Furthermore, Geant4 also simulates decays of unstable particles. The user is able to create a geometrical description of the material to be simulated and load it into the simulation.

Geant4 provides a number of parameters, which offer the user the ability to fundamentally change the simulation's behaviour and output, such as performance, accuracy, physics models, simulated physics effects (physics list), stepping size, etc.

Due to its long-time operating experience, G4 has become a highly validated and sophisticated simulation for many particle physics experiments. Therefore Geant4 is the main (and most accurate) detector simulation applied by the ATLAS collaboration. This comes, however, with the price of an immense demand for computing resources.

3.2.2 Fatras – Fast ATLAS Track Simulation

Fatras [17] (Fast ATLAS Track Simulation) is a fast ATLAS detector simulation for the inner detector and the muon spectrometer. Particle tracks are simulated by using

standard ATLAS reconstruction tools based on the so-called *reconstruction geometry* [18], which is a simplified detector description. Contrary to the detailed description of the detector material applied in Geant4, the reconstruction geometry describes the ATLAS detector material by a few thin and discrete layers of different materials (see figure 3.3). The layers are arranged in order to reproduce the overall material effects of the detector on the traversing particles. In addition, Fatras uses *averaged* materials which do not necessarily correspond to any real material built into the detector.

Fatras gains up to a factor of 100 in simulation time, in comparison to Geant4 based detector simulations.

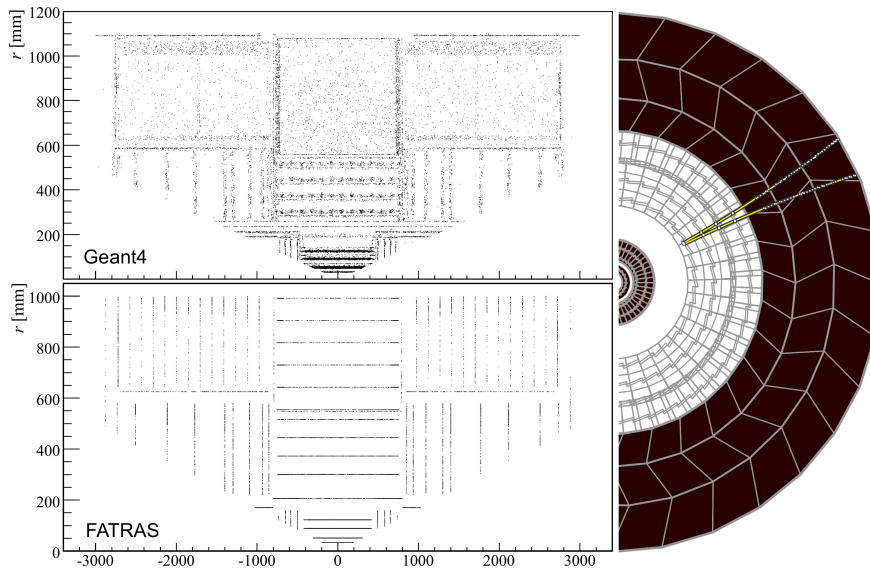


Figure 3.3: Photon conversion points shown in Geant4 and Fatras. Since photon conversion depends on the density of the material, the discrete and thin layers of material used in the Fatras simulation are clearly visible [5]. The right side shows an Atlantis [19] event display view of such an event.

Compared to Geant4, Fatras gains additional computing time, by spending less time on the simulation (creation) of secondary particles produced due to particle-material interactions. Clearly this comes at the price of a less accurate description of the processes occurring in the real detector. However, Fatras still shows good overall agreement with the full detail Geant4 detector simulation and data taken with the detector (see [17, 20]).

For the inner detector Fatras applies its built-in digitization, while the muon spectrometer digitization is done by the standard ATLAS digitization tools. The Fatras digitization output is compatible to the standard digitization output, therefore the standard ATLAS reconstruction tools are called for both, the ID and MS.

Fatras is also capable of directly creating track objects by using its built-in *track refitting* engine. It can do so without using the pattern recognition algorithms which are

part of the standard reconstruction package. When track refitting is enabled, one can study the effects of the pattern recognition algorithms on the resulting track objects.

3.2.3 AtlfastII

AtlfastII is a fast ATLAS detector simulation using the FastCaloSim [21] ATLAS calorimeter simulation. For the ATLAS inner detector and muon spectrometer, the Geant4 simulation is used. This setup results in a simulation ten times faster than the full Geant4 simulation of the whole ATLAS detector [8].

As for the full Geant4 detector simulation, the input particles from the event generator are used as input for the subsequent Geant4 inner detector simulation. Each particle leaving the ID volume is checked for its particle type and only muons are further processed by the Geant4 simulation of the calorimeter. All particles above threshold, however, are handled by the FastCaloSim module, which does not perform a particle transport. Therefore muons are the only particles able to enter the ATLAS muon system in the AtlfastII simulation. Consequently, internal processes in the calorimeter, such as decays and punch-through are not simulated in AtlfastII.

Due to the use of Geant4 simulations in the ID and MS, the standard ATLAS reconstruction tools can be used in order to reconstruct particle information from AtlfastII simulated detector hits in the ID and MS. Moreover, the FastCaloSim output is compatible to Geant4 calorimeter simulation output. Therefore the calorimeter reconstruction can be done by the standard ATLAS tools as well.

AtlfastIIF applies Fatras for the inner detector and the muon spectrometer instead of Geant4 as in AtlfastII. By doing so, a factor of up to 100 in speed is gained compared to the Geant4 full detector simulation.

3.3 The Athena Framework

The Athena framework [9, 22] is widely used by the ATLAS collaboration for most of its computing work. Athena is based on the Gaudi framework which was initially developed by the LHCb collaboration. In the ATLAS experiment it is the common framework which is used for Monte Carlo simulations, data processing and physics analysis of either simulated or recorded data.

User implemented C++ class(es) can be integrated into the Athena framework, within which these classes will have access to given input and output event containers. Athena manages event by event data preparation and it takes care of input and output file handling.

The setup of any Athena run is done via a Python `jobOptions` script (sometimes referred to as `j0` scripts). This script is handed over as parameter to the `athena` shell command when starting an Athena run. This allows the user to fundamentally change the behaviour of any Athena run, by changing the settings inside the `jobOptions`. Within this script, one has the possibility to:

- choose which input dataset to read and which output dataset to write

- choose which C++ (or Python) algorithms should run
- setting and changing (optional) parameters for these algorithms

3.3.1 Athena Application Flow

In order to use C++ libraries within Athena, one has to implement the source code into one or many of the fundamentally different components given by the framework. A complete list of all components is given in [9]. The most relevant components in the context of this work are: *Algorithms*, *Tools* and *Services*. Any user algorithm can be implemented in either of these categories. Figure 3.4 explains the runtime relevance and the difference between Athena algorithms and Athena tools.

Athena algorithms, tools and services can be implemented either by the ATLAS software developers, as a part of the official ATLAS software package, or any user who wants to study certain properties of one or more – simulated or recorded – datasets.

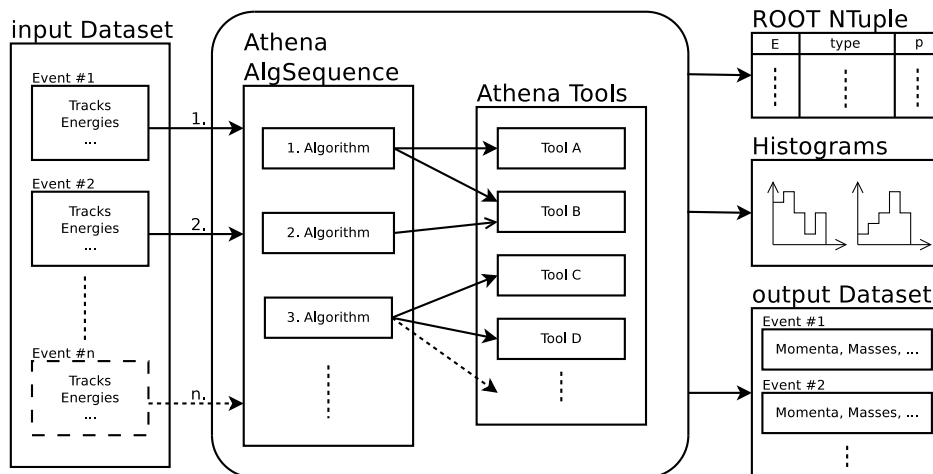


Figure 3.4: Athena application flow when running multiple Athena algorithms which use different Athena tools. The data containers for each respective event are handed over to the Athena algorithms one after the other. This figure gives some examples of possible output types for the Athena algorithms and Athena tools used. The Athena algorithms are always called in the same sequence for each event. This sequence is given by the `AlgSequence` object, which is configured during the initialization phase via the Python `jobOptions`. In most cases an input dataset needs to be given, but there are certain cases where no input dataset is used (eg: physics event generators or Fatras single track simulation)

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. In order to do so, the framework runs in sequence through the events in a certain input dataset and processes them one by one by calling different C++ Athena algorithms on them. An Athena algorithm is a C++ class which inherits from the Athena `AthAlgorithm` class. While processing an event this algorithm only has access to the data relevant to this currently processed event. This carries out the idea of having data which come from independent collision events and study them one by one. After reading the input data to process the current event the Athena algorithms are run in a specified order, which is set by the `AlgSequence` object in the `jobOptions` file. Each of the Athena algorithms specified in this sequence is run only once per event.

Athena Algorithm Tools – `AthAlgTool`: The previously described Athena algorithms may use some common computations or apply some common tasks which will not be implemented in each of the algorithms separately, but only once in corresponding *Athena algorithm tools*. Athena algorithm 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. Other than the Athena algorithms, Athena tools can be called multiple times per event, but they are not necessarily executed in a pre-defined sequence.

Athena Algorithm Services – `AthService`: The Athena services are somewhat similar to the Athena tools, in the sense that they can be run multiple times within a single event. But other than 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 or random number generators. Athena services can be implemented by inheriting from the `AthService` C++ class provided by Athena.

3.3.2 StoreGate

StoreGate (SG) is the central Athena service responsible for handling any kind of transient data object needed by one or many Athena algorithms, tools or services. A detailed description of StoreGate can be found in [23]. Here only the most important features of StoreGate are discussed:

- it takes care of the memory management for any data object registered to it. This implies, for example memory deallocation in case a data object is not needed any more in its transient form.
- it manages the conversion from transient data to persistent data
- the user can access any data object in memory via a built-in dictionary
- nearly any user data type can be managed by StoreGate

Typical objects managed by StoreGate are, for example, `TrackCollections` containing all reconstructed particle tracks, or hits of sensitive detector elements (simulated or recorded ones).

3.3.3 Data Formats

A number of persistent data formats are used in the Athena framework, to store particle and detector information at different stages. In order to allow the use of these data formats on the LHC Computing Grid (LCG), a *POOL* layer is implemented within each of the formats described in this section. Further information about the POOL layer can be found in [9]. The following list only gives the most relevant data formats for this work. More derived formats are also used by the collaboration, but mainly for physics focused analysis, whereas in this work the focus lies on detector simulation and its validation.

Simulation HITS: This data format contains hit information of sensitive detector elements in simulated events. It is the output format of the Geant4 detector simulation (see section 3.2.1). This format is purely simulation based, therefore no counterpart in the data stream from the real detector exists.

RAW Data Objects (RDO): RAW data objects contain voltage or current measurements of sensitive detector parts for each recorded (or simulated) event. The RDO format does not offer any particle information or physical interpretation of these detector measurements. The output of the detector itself is in RDO format. But RDOs might also be generated by any detector simulation, after the *digitization* step (see section 3.1).

Event Summary Data (ESD): Is the output format after running the standard reconstruction algorithms (see section 3.1) on the ATLAS inner detector, calorimeter and muon spectrometer. The (simulated or recorded) detector hits from the previous step are converted into physical objects, like particle tracks or jets.

Analysis Object Data (AOD): AOD is a data format derived from ESD which is commonly used for physics analysis. Therefore AODs contain mostly physical objects. Other than the ESDs, AODs usually do not contain detailed information about the reconstruction step.

Chapter 4

Calorimeter Punch-Through – Leakage into the Muon Spectrometer

This chapter starts by defining the main concern of this work, the *calorimeter punch-through* or *particle leakage* in section 4.1. Next, the basic physical processes and concepts occurring inside the ATLAS calorimeter are described in section 4.2. Based on this, Geant4 studies were conducted and their results are described in section 4.3. Due to the high number of – partly very complex – processes involved in *particle showers* inside the calorimeter, a parametrized approach for a fast calorimeter punch-through has been favored to simplified models, when trying to optimize for CPU performance. Section 4.4 describes such a parametrization model and finally the simulation applying this model is described in section 4.5.

4.1 Calorimeter Punch-Through

The ATLAS calorimeter is built as a hermetic, confining detector (see sections 2.3 and 2.4). Muons should be the only particle type to reach the muon spectrometer and cause detector hits there. Particles entering the calorimeter, will cause a particle shower inside the calorimeter. The shape of this shower and the shower-particle properties in general strongly depend on properties of the initial particle (see figure 4.1 for a qualitative illustration of the energy dependency). Some shower shapes are more elongated (for example for higher energetic initial particles), which enhances the chance of a shower particle to exit the calorimeter and enter the ATLAS muon spectrometer. Such an event is called *calorimeter punch-through* or *particle leakage* into the muon spectrometer and the particles entering the MS are called *punch-through particles*. Depending on the properties of the punch-through particle (particle type, energy, charge, position), the effect to the muon spectrometer will vary significantly: from having no effect in creating clean particle track signatures, to full tracks which will be reconstructed and by chance

be misinterpreted as primary muon tracks by the MS reconstruction. These tracks are further called *fake primary muon* tracks, since they may be used as misidentified muons in physics analyses.

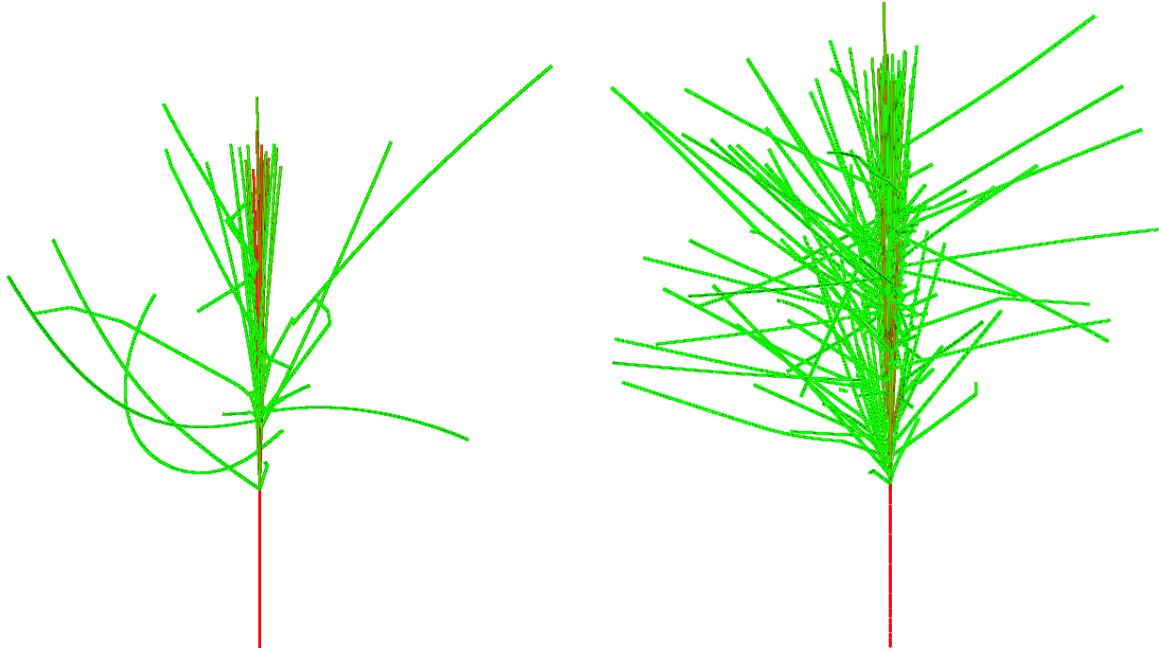
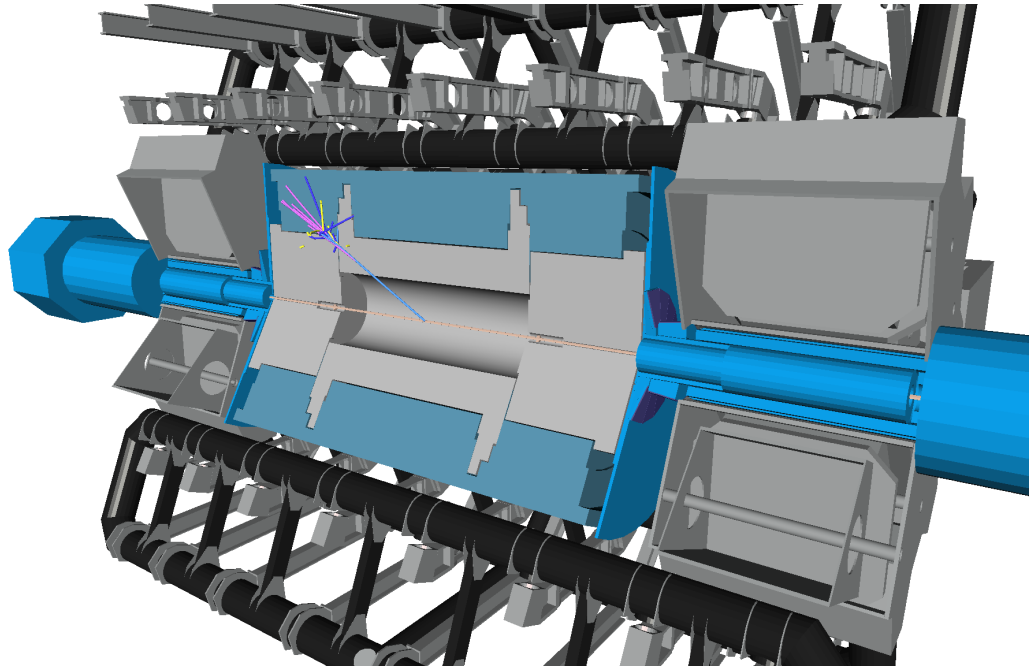
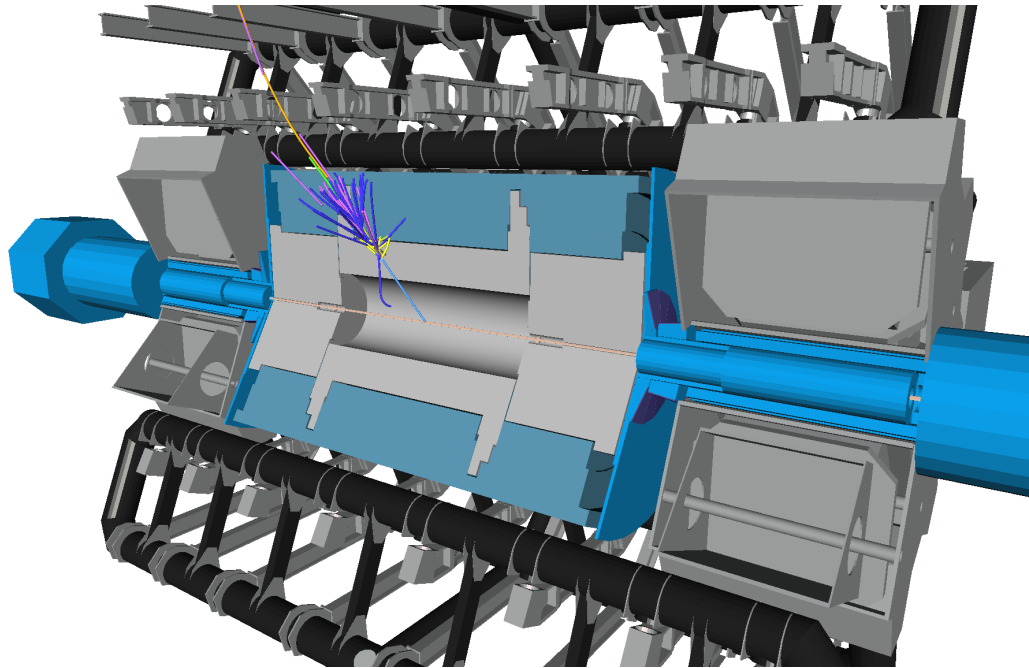


Figure 4.1: A qualitative study of the calorimetric particle shower’s dependency on the initial particle’s energy. The left image shows a particle shower inside the ATLAS hadronic calorimeter caused by a low energetic ($E \approx 100$ GeV) incoming π^- -particle. The right image depicts a shower caused by a high energetic ($E \approx 450$ GeV) π^- -particle. In both images, high energetic particles are colored in red, low energetic ones in green. Only particles with a momentum $p \geq 0.5$ GeV/c are shown. From the images one can deduce that higher energetic particles cause a more spread out particle shower inside the calorimeter – also meaning that a higher number of shower particles is produced.

Figure 4.2 illustrates two Geant4 simulated events of charged π particles entering the ATLAS calorimeter. In the case of the higher energetic particle, a calorimeter punch-through occurs and one of the resulting punch-through particles creates detector hits in the ATLAS muon spectrometer, which subsequently lead to a reconstructed track (orange line) in the MS.



(a)



(b)

Figure 4.2: Virtual Point 1 (VP1) illustration of two Geant4 simulated events of a charged pion as initial particle originating from the interaction point: (a) the calorimetric confinement of the hadron works as expected (low energetic pion). (b) a calorimeter punch-through event (high energetic pion) where one or many particles coming from a particle shower inside the calorimeter, enter the ATLAS muon spectrometer.

4.2 Processes in the ATLAS Calorimeter

In section 2.3 a brief overview of the ATLAS calorimeter was given. This section describes the most frequent electromagnetic and hadronic processes occurring inside a calorimeter. A more extensive view can be found in [24].

Due to particle showers caused by incident particles in dense material, the particle energy is absorbed in a much shorter distance than in a case where no shower develops. Hence, calorimeters provoke shower development in order to confine and measure incident particle energies.

The following subsections discuss the two fundamentally different shower types: electromagnetic and hadronic. The somewhat special role of muons traversing dense material is discussed in section 4.2.2.

4.2.1 Electromagnetic Showers

The processes involved in the development and evolution of electromagnetic showers are rather clear and much easier to describe than those in hadronic showers. Electromagnetic showers are caused by charged particles or photons in dense material.

Low energetic photons will most likely undergo the **photoelectric effect**, in which one bound electron becomes unbound from its atom and the photon is absorbed. **Rayleigh scattering** deviates photon trajectories through material, without them losing any of their energy. In **Compton scattering** effects, photons lose part of their energy, their trajectory gets deviated and one electron is transferred from a bound state into an unbound state. Photons with a high enough energy have the chance to undergo electron-positron **pair production** in the coulomb fields present in the vicinity of nuclei in the dense material. This process transfers the whole photon energy into the electron and the positron. **Photonuclear** reactions become relevant to high energetic photons. However, they play a minor role compared to all other photon interaction processes. Anyhow, nucleons may be freed due to such reactions. In most of these photon interaction processes, at least one charged particle (mostly electron and/or positron) is created. Therefore, charged particle interaction processes need to be understood.

Due to electromagnetic interactions, charged particles may interact in many different ways with dense material when traversing it. **Ionization** of the material along the particle's trajectory already occurs at very low energies. Also atomic or molecular **excitation** may occur. The photons emitted during de-excitation will subsequently undergo any of the photon processes mentioned above. **Cherenkov** light will be emitted by charged particles, in case they traverse a medium with a speed greater than the speed of light in that medium. At sufficiently high energies, charged particles may lose energy by knocking out high energetic electrons (**δ -rays**) of the material structure they traverse. High energetic particles with a low mass (such as electrons and positrons) may lose a significant amount of their energy due to **bremsstrahlung**. **Nuclear** interactions may occur at very high energies. Most of these processes will result in the production of one or many daughter particles, such as photons, electrons or nuclear particles.

There is one fundamental difference between the processes involved in energy loss of

electrons and photons. Whereas electrons lose their energy in a continuous stream of events (e.g. bremsstrahlung), photons may traverse a certain amount of material without losing any energy before one interaction consumes the photon (e.g. pair production).

In electromagnetic showers, one primary particle (such as photon or electron) causes a cascade of the effects mentioned above, in which a large number of particles might be involved. Until a certain shower depth, particle multiplication outweighs particle consumption. However, due to the energy deposited in the material, the average shower particle energy decreases with the shower depth. Thus after the *shower maximum*, the total number of particles involved in the shower decreases. Typical for em-showers is that mostly photons and electrons (positrons) are involved in the particle cascade. For example an electron created (freed) during the shower development, might create one or many photons (e.g. bremsstrahlung), which will again free other electrons, etc. Some of the shower particles may leave the dense material, which is known as calorimeter punch-through (or leakage) as defined in the previous section 4.1.

In the ATLAS calorimeter, the deposited energy is measured with the free ionization electrons and the light emitted from scintillator material inside the calorimeter.

4.2.2 Muons Traversing Dense Material

Due to their charge, the processes occurring when muons traverse matter are the same as mentioned above for electrons and charged particles in general. However, due to the high muon mass, energy loss due to bremsstrahlung is highly suppressed compared to electrons. Thus, muons mainly undergo ionization loss and multiple scattering processes when traversing matter. Therefore it takes a much greater amount of dense material, in order to absorb a muon compared to an electron.

This is the reason for the ATLAS muon spectrometer surrounding the (dense) calorimeter and still being able to detect muon particle tracks in the MS.

Due to the properties mentioned above, muons are often referred to as minimum ionizing particles (MIP).

4.2.3 Hadronic Showers

When high energetic charged hadrons traverse dense material, they will behave at a first glance similarly to muons with the same energy with regard to the electromagnetic processes occurring – i.e. they will continuously lose energy. However, with a certain probability the hadron will strike a nucleus and numerous nuclear and strong interaction processes will take place. For neutral hadrons, this is the only way to lose their energy. Many of these nuclear processes will result in additional shower particles created due to such an event. Such shower particles could be neutrons or protons freed from the hit nucleus, mesons created due to strong interaction and high energy photons due to de-excitation of excited nuclei. Some of these mesons may decay electromagnetically, such as $\pi^0, \eta \rightarrow \gamma\gamma$. Charged particles and photons will interact electromagnetically in the way described in the previous section 4.2.1. Therefore a certain fraction of the initial hadron's energy is deposited in electromagnetic showers. Some mesons may decay

into muons (decay in flight), which is a source of energy leaking from a calorimeter, since muons only interact very weakly with matter. The shower hadrons will interact with other nuclei in the material. Therefore – as for electromagnetic showers – particle multiplication takes place until a certain shower depth. Beyond this depth, the average energy of the shower particles is too low to cause further shower particle multiplication.

For example, a particle shower caused by a 100 GeV pion in lead has $\sim 55\%$ of its energy in the electromagnetic component and the rest in the hadronic shower component [24].

4.3 Geant4 Analysis

In order to implement a fast simulation of particles leaking into the ATLAS muon spectrometer, a detailed study of this very effect was carried out using Geant4 (section 3.2.1) simulated single particle events. Boundary surfaces (*CaloEntryLayer* and *MuonEntryLayer*, see appendix B) are used to define the points where a particle enters and exits the calorimeter, respectively. The following parametrization is therefore based on the definition of these layers. As soon as a particle traverses either of these surface layers during a Geant4 simulation, its current properties are recorded in a so-called **TrackRecordCollection** object (named either *CaloEntryLayer* or *MuonEntryLayer*) via the Athena *StoreGate* ([23] and section 3.3.2) service. All studies described in this- and the following sections only use entries in these two **TrackRecordCollections** in order to analyse properties of punch-through particles.

4.3.1 Simulation Setup and Event Selection

A sample of $\sim 3 \cdot 10^6$ single pion events were simulated to perform these studies: π^+ and π^- particles in equal numbers, all originating at the interaction point. These particles are simulated with a flat distribution in pseudorapidity η between -2.7 and $+2.7$. This range is motivated by the pseudorapidity coverage of the ATLAS muon spectrometer for $|\eta| \leq 2.7$ [3]. The initial energy is distributed uniformly between 150 and 500 GeV. The focus of this study lies on calorimeter and muon system effects and interactions, therefore interactions with any inner detector part (active and passive) were not simulated – in order to save computing time and disk space. If not stated differently, this event sample was used for all further studies and parametrizations.

The Geant4 simulations use ATLAS software release 15.6.12.7 with detector description tag **ATLAS-GE0-16-00-00**.

After simulation, an Athena analysis algorithm was run on the HITS pool files (which were created in the last step) to gain information about punch-through particles and their properties. Two definitions are required to understand the following analysis (see figure 4.3 for a visual illustration of these definitions):

Initial Particle is a π^+ or π^- particle appearing in the *CaloEntryLayer* **TrackRecordCollection** with the same **barcode**¹ as the one particle put into the Geant4 simulation at the interaction point. Additionally the initial particle has to point outwards (towards the calorimeter). Due to technical constraints in the ATLAS Geant4 simulation framework, the beam pipe simulation volume of the beam pipe can not be switched off. In case the starting pion interacts with the beam pipe, it will most likely not arrive at the *CaloEntry* layer and therefore such events are filtered out. If no interaction takes place, the particle gets transported to the *CaloEntryLayer* without simulating any interactions with the ATLAS inner detector material. In all following studies, only events with exactly one initial particle are studied.

¹unique number to identify individual particles within one simulated event

Punch-Through or Output Particle is any particle entry that appears in the MuonEntryLayer TrackRecordCollection and has a momentum pointing outwards (towards the muon spectrometer).

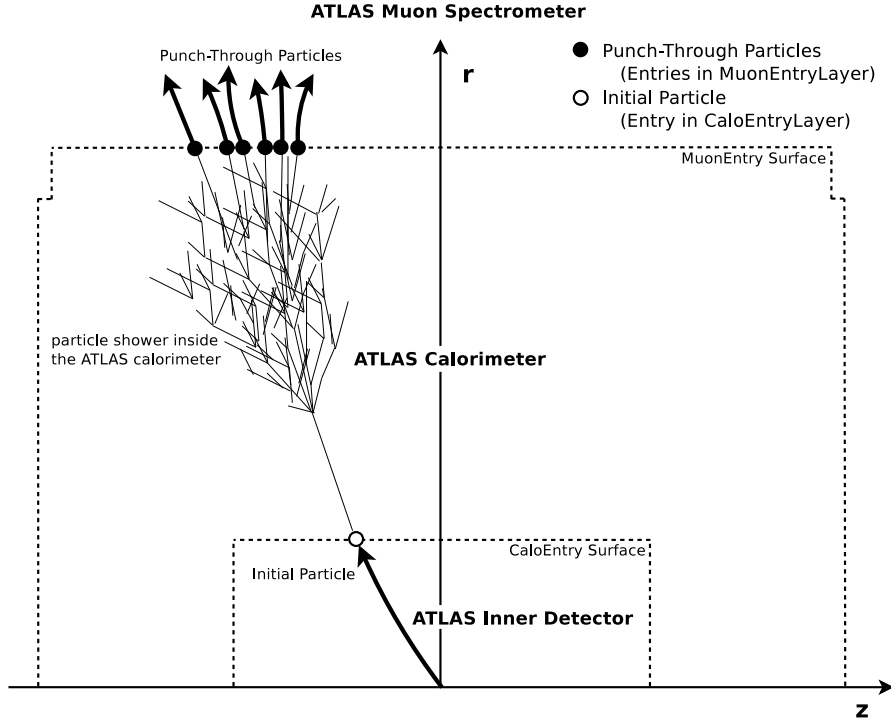


Figure 4.3: Initial particle and the punch-through particles of a typical calorimeter punch-through event. Beside the position, also the particle momentum, the energy and the Monte Carlo particle number can be retrieved for the initial- and punch-through particles, respectively.

4.3.2 Punch-Through Particle Types

The two most fundamental properties that need to be understood, regarding calorimeter punch-through events are:

1. the rate of occurrence of such events
2. the particle types penetrating the ATLAS muon spectrometer most frequently

Different types of particles will have different effects on the sensitive detector elements in the ATLAS muon spectrometer. For example, high energetic, charged particles will

penetrate far into the MS and cause detector hits on their way through the MS. Also particle showers inside the MS can be caused by any high energetic particle. Due to their lack of charge, photons for example will not directly cause detector hits inside the MS, but they can undergo *photon conversion*, where electrons and positrons will be produced (see section 4.2).

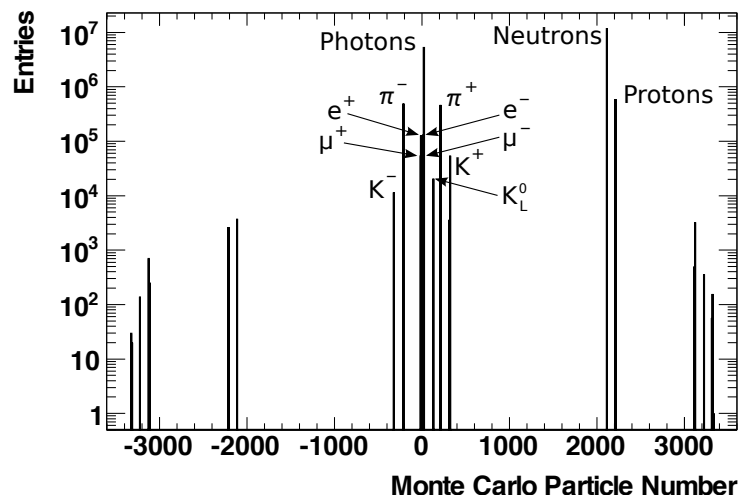


Figure 4.4: Different particle types appearing as punch-through particles. This study is based on Geant4 simulated single $\pi^{+/-}$ events, with energies up to 500 GeV (described in detail in section 4.3.1). The Monte Carlo particle numbering scheme is discussed in [25].

Figure 4.4 shows that the single most frequent punch-through particle type is the neutron. Due to its lack of charge and its weak signal rate in the sensitive muon system detector elements (see [26] and [27] for neutron background studies in the CSC and MDT muon system subdetectors, respectively), neutrons are neglected in any further studies of this work. However, the chosen parametrization approach has been implemented in a flexible way to allow for future extensions, such as e.g. the inclusion of neutrons.

Due to their charge and/or high frequency of appearance, the following particle types were chosen to be most relevant for a later punch-through simulation:

- photons (γ)
- protons (p)
- charged pions (π^+ and π^-)
- electrons and positrons (e^+ and e^-)
- muons (μ^+ and μ^-)

As previously described, the photon is included in the list above due to the fact that there is a chance for it to undergo photon conversion inside the muon spectrometer. Each conversion would produce an electron/positron pair which would subsequently create signals in sensitive detector elements.

In addition, figure 4.4 suggests that for electrons, pions and muons, the corresponding matter like and antimatter like particle types appear at the same rate. Therefore, further studies (and parametrizations) consider cumulative effects of matter like and antimatter like particles for either of these particle types.

A minimum energy of 50 MeV is required for any particle to be considered a relevant punch-through particle. Particles with an energy below this threshold are ignored in any further study or parametrization.

4.3.3 Punch-Through Occurrence and Number of Particles

The rate of punch-through events is crucial for the simulation of this very effect. Clearly, this rate will show a strong dependency on the initial particle's energy. For high energetic initial particles, the particle showers inside the calorimeter will have a more elongated form [24]. This increases the probability to find one or more punch-through particles caused by this initial particle. Figure 4.5 clearly shows this dependency in the Geant4 simulations.

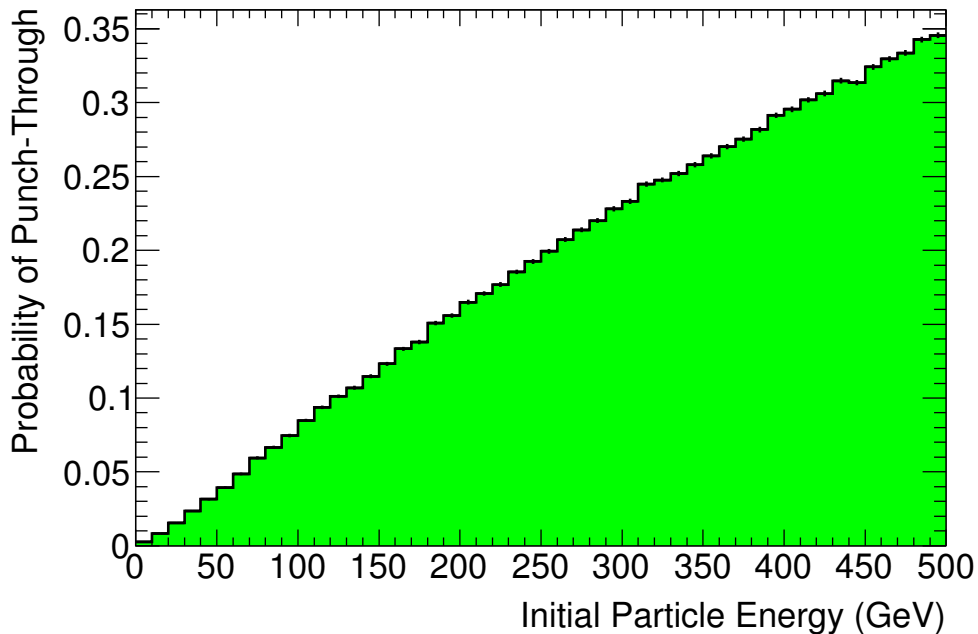


Figure 4.5: Probability of one or more punch-through particles to enter the ATLAS muon spectrometer, based on Geant4 simulations of single charged pion events.

Besides the initial particle energy, there are other parameters which have some impact on the punch-through probability. Most significantly, the pseudorapidity η and thus the corresponding entry position and direction of the particle in the calorimeter have to be taken into account. The reason for this dependency is given by amount of calorimeter material, which strongly depends on η (see figure 4.6(a)). High energetic particles contained in a particle shower inside the calorimeter form a cone around an axis which can be thought of as the extension of the incoming particle's track [24]. Due to low magnetic fields inside the ATLAS calorimeter, a straight line extension along constant η gives a good approximation for particle trajectories traversing the calorimeter. The more calorimeter material the track extension traverses, the greater the chance that the shower particles will be stopped due to material interactions and thus the punch-through probability is low – and vice versa. Figure 4.6(b) shows a clear dependency of the punch-through probability on the incoming particle's η . By comparing this dependency with the calorimeter material budget in figure 4.6(a) the anti-correlation between these two properties becomes evident.

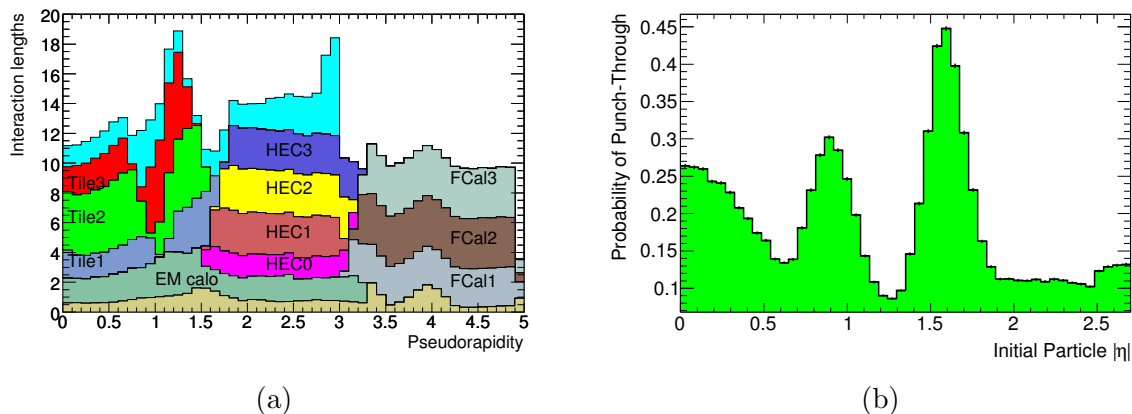


Figure 4.6: (a) Calorimeter material budget (in units of interaction length x_0) versus pseudorapidity $|\eta|$ [3]. (b) Probability to find one or more relevant punch-through particles in a single event vs. the initial particle's $|\eta|$.

4.3.4 Particle Type Correlations

In case of a punch-through event, it is very likely to have more than one punch-through particle entering the muon spectrometer at the same time. A set of particles entering the muon spectrometer can cause a high number of hits and reconstructed particle tracks there. This enhances the chance to create a signal which can be misinterpreted and become a fake primary muon track. Therefore also the distribution for the number of particles is subject to further studies (figure 4.7)

Amongst the particles leaking into the muon spectrometer general correlations must exist, since the particles come from related processes inside the calorimeter and the same initial particle. In addition to the correlations with the initial particle's energy

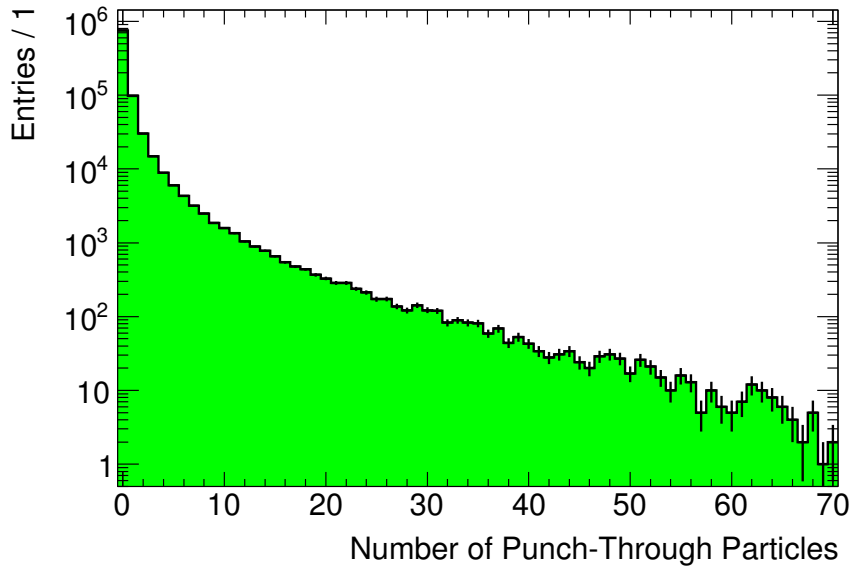


Figure 4.7: Number of punch-through particles with energies above 50 MeV appearing within Geant4 simulated single pion events. The initial pions were simulated with energies up to 500 GeV with pseudorapidities of $|\eta| \leq 2.7$ (details given in section 4.3.1).

and pseudorapidity (mentioned in the last section), one fundamental correlation is the appearance of certain particle types:

Electrons and Photons : $e^{+/-}$ and γ occurrences are strongly coupled due to electromagnetic processes occurring in the calorimeter (see section 4.2). Figure 4.8(a) depicts the correlation seen in the Geant4 simulations.

Pions and Protons : The correlation between $\pi^{+/-}$ and proton occurrence comes from underlying hadronic processes occurring in the calorimeter (section 4.2). Figure 4.8(b) depicts a strong correlation between the two given particle types.

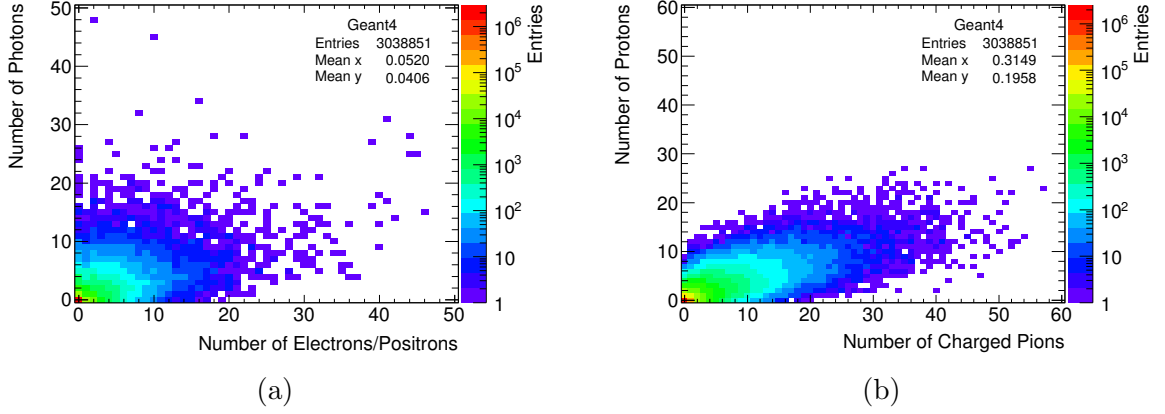


Figure 4.8: Calorimeter punch-through particle correlations observed in Geant4 studies on single pion events: (a) correlation between the number of punch-through photons and electrons/positrons (b) correlation between the number of punch-through protons and charged pions.

4.3.5 Particle Energies

Besides the number of particles entering the muon spectrometer, also the individual particle energies will have a strong impact on the signals recorded in the MS. High energy particles will have a more significant effect to the MS than lower energetic ones. Therefore, the energy spectra of punch-through particles are studied in detail (figure 4.9).

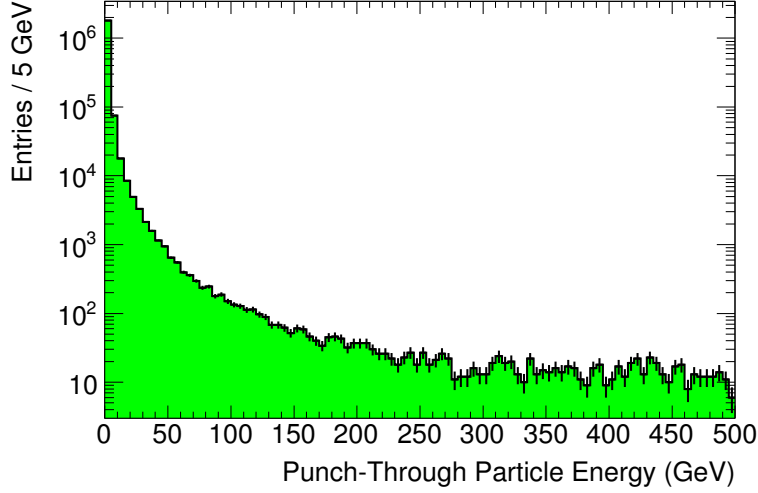


Figure 4.9: Energy spectrum of punch-through particles entering the muon spectrometer, based on Geant4 simulations of single pions.

4.3.6 Deflection Angles $\Delta\phi$ and $\Delta\theta$

As previously described, high energetic shower particles inside the calorimeter form a cone around the direction of the incoming particle. The most significant punch-through particles will mostly be the high energetic shower particles from inside the calorimeter. From a geometric point of view, the punch-through particles can therefore be interpreted as *deflected* incoming particles (apparently with different properties like: energy, particle type, ...). In principal, one deflection angle would be enough, to describe a cone-like deflection of the punch-through particle compared to the incoming particle. But due to the highly complex calorimeter design the punch-through particle deflection is described with **two** deflection angles: $\Delta\phi$ and $\Delta\theta$. The respective angles are the difference between the incoming particle position on the CaloEntry surface and the punch-through particle position (see figure 4.21 for an illustration):

$$\begin{aligned}\Delta\phi &= \phi_{incoming} - \phi_{punch-through} \\ \Delta\theta &= \theta_{incoming} - \theta_{punch-through}\end{aligned}\tag{4.1}$$

The $\Delta\phi$ and $\Delta\theta$ distributions come from randomly distorted particle trajectories through the calorimeter. Therefore both distributions are symmetric around a highest center value at zero and they should be of similar shape. However, the $\Delta\theta$ distribution will show an additional plateau caused by a favoured punch-through pseudorapidity region. Figure 4.10 (a) and (b) show that a certain amount of punch-through particles prefers to enter the muon spectrometer at a pseudorapidity $|\eta|$ between 0.7 and 0.9, for an initial particle's pseudorapidity of $0.7 < |\eta| < 1.5$. This observation becomes clear when recalling the calorimeter material distribution versus $|\eta|$ in figure 4.6(a). Apparently

the calorimeter has less material (or even gaps) in this preferred punch-through region, which clearly enhances the chance for calorimeter shower particles to exit the calorimeter in this region. Therefore the $\Delta\eta$ distribution ($\Delta\eta = \eta_{incoming} - \eta_{punch-through}$) – and consequently also the $\Delta\theta$ distribution – has an additional excentric *bump* in the region $0.7 < |\eta_{incoming}| < 1.5$. This adds up to the plateau seen in figure 4.11 (b) for $0.3 \leq |\Delta\theta| \leq 0.5$. Figure 4.11 (a) shows the corresponding $\Delta\phi$ distribution.

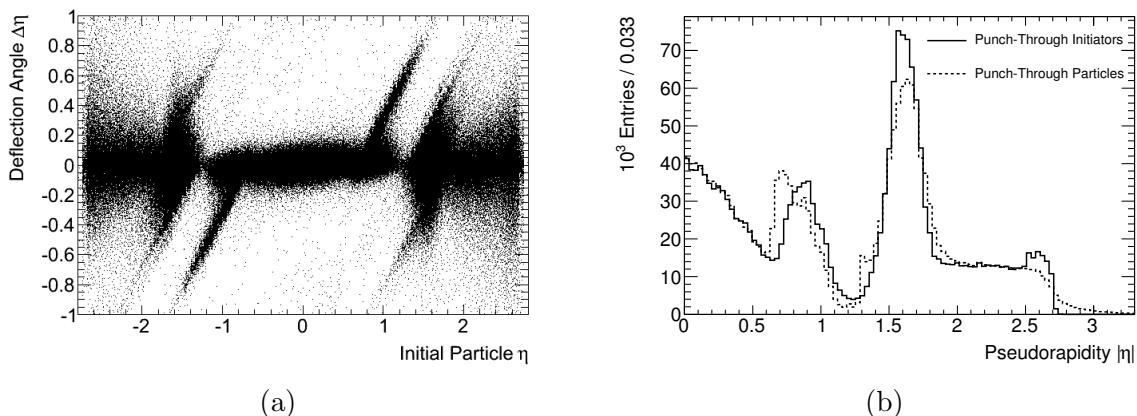


Figure 4.10: Analysis of the preferred punch-through particle $|\eta|$ between 0.7 and 0.9. (a) the distribution of $\Delta\eta$ for all punch-through particles with respect to the initial particle's pseudorapidity η . (b) the pseudorapidity $|\eta|$ distribution of the initial particles causing at least one punch-through particle (continuous line). The dashed line displays the $|\eta|$ distribution of the punch-through particles themselves. The initial particles used in the Geant4 simulation are uniformly distributed in η between -2.7 and $+2.7$ (section 4.3.1), but due to the calorimeter's internal structure, the chance to cause punch-through varies with η (section 4.3.3).

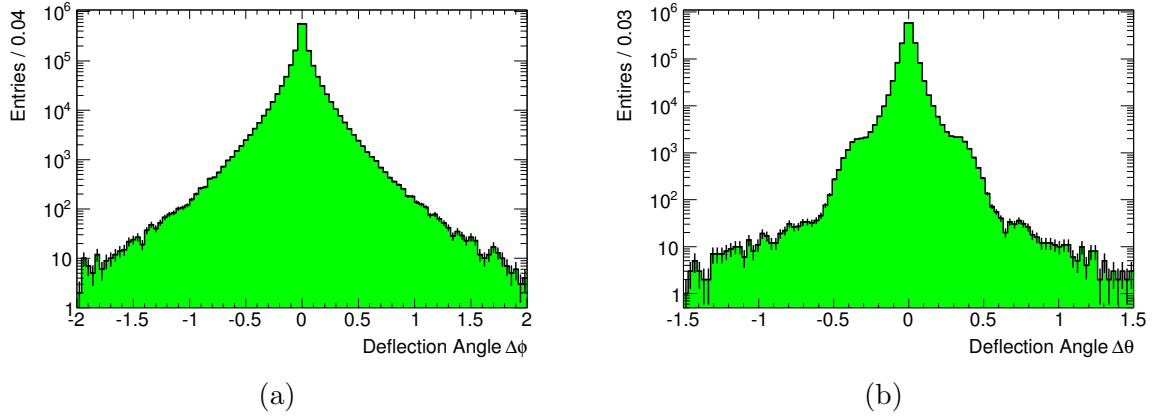


Figure 4.11: Punch-Through particle deflection angles $\Delta\phi$ and $\Delta\theta$ observed in Geant4 simulations.

4.3.7 Particle Momentum Direction ($\Delta\phi_p$ and $\Delta\theta_p$)

The remaining crucial parameter for punch-through particles is the momentum direction when entering the ATLAS muon spectrometer. The momentum direction has a strong impact on the probability to reconstruct particle tracks in the muon spectrometer or even ID/MS combined particle tracks caused by punch-through particles.

Following the idea in the previous section, the *momentum deflection* angles $\Delta\phi_p$ and $\Delta\theta_p$ are defined by:

$$\begin{aligned}\Delta\phi_p &= \phi_{pos} - \phi_{momentum} \\ \Delta\theta_p &= \theta_{pos} - \theta_{momentum}\end{aligned}\tag{4.2}$$

Where ϕ_{pos} and θ_{pos} give the global polar angles of the respective punch-through particle position on the MuonEntry surface. $\phi_{momentum}$ and $\theta_{momentum}$ give the local momentum direction with respect to the punch-through position (see figure 4.21 for an illustration).

Figure 4.12 shows the $\Delta\phi_p$ and $\Delta\theta_p$ distributions in the Geant4 single pion event sample. Due to using the *local* momentum direction in the definition above, different shapes of the $\Delta\phi_p$ and $\Delta\theta_p$ distributions are expected – other than for the $\Delta\phi$ and $\Delta\theta$ distributions in the previous section.

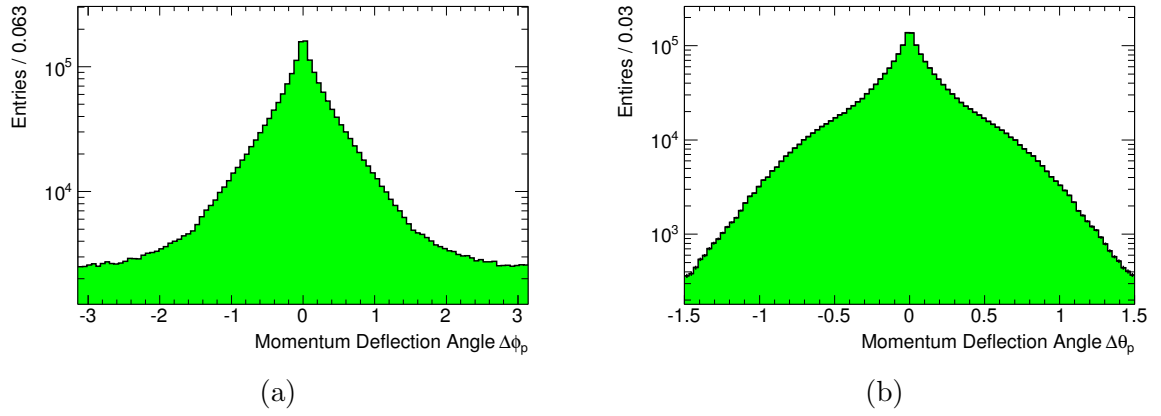


Figure 4.12: Punch-through particle momentum deflection angles $\Delta\phi_p$ and $\Delta\theta_p$ in a Geant4 simulated single pion event sample. $\Delta\phi_p$ and $\Delta\theta_p$ are defined in a local coordinate frame centered at the respective punch-through particle position.

4.4 Parametrization

Due to the high number of – partly very complex – processes occurring inside the calorimeter, the most promising approach for a fast calorimeter punch-through simulation is a parametrized simulation. The advantage of a parametrized simulation is that it is very fast during runtime. This is due to the fact that only the results of particle interaction processes are reproduced (based on a pre-recorded set of parameters and their relations) rather than simulating the physical model behind each process. The input- and output-parameter dependencies and -correlations are stored in a look-up table, which in this particular implementation is a ROOT file. The structure of this ROOT file is explained in appendix A.

In order to describe (and store) these dependencies, the previously mentioned sample of Geant4 simulations (section 4.3) were used to quantitatively obtain certain fit parameters and correlations. This parametrization is carried out for each Geant4 analysis described in sections 4.3.2–4.3.7, respectively.

The whole parametrization process is automated due to the high number of required individual parametrizations, and to give the flexibility to re-parametrize the simulation based on a different reference sample. Therefore a modified χ^2 method (section 4.4.1) is used to measure the quality of each individual fit.

4.4.1 Fit Quality Measure

Many of the following parametrization sections will utilize automated fit procedures in order to describe distributions obtained from Geant4 simulations. The fits are carried out with the ROOT implementation of the binned log-likelihood method. Due to the high number of total fits applied, a measure for the quality of each individual fit was put in place.

In order to do so, this work applies a modified version of the standard χ^2 test [28], which is often used to test the quality of a fit. The standard χ^2 test did not prove to be useful for the fits carried out in this work. Instead of computing the χ^2 value for each fit, a somewhat modified χ_m^2 value is computed:

$$\begin{aligned}\chi^2 &= \sum_i \frac{(O_i - E_i)^2}{\sigma_i^2} \\ \chi_m^2 &= \sum_i (O_i - E_i)^2\end{aligned}\tag{4.3}$$

The sum is taken over all bins i of the respective histogram, O_i are the values obtained from the Geant4 simulations with their corresponding variances σ_i^2 and E_i are the fit function value at the centre of the corresponding bin.

Equation 4.3 shows that the modified χ_m^2 value corresponds to the standard χ^2 value, except for the fact that χ_m^2 uses a standard value of $\sigma_i^2 = 1$ for the variance in each individual bin. This is motivated by the form of the distributions fitted in this work: normalized probability distributions with only a few high probability bins (MPVs,

values between 0.1 and 1), and an exponential decay following these MPVs (down to values of $\sim 10^{-6}$). Therefore it is most important to obtain an accurate description of the (few) most probable values, which will have a strong impact on the overall χ_m^2 value. Due to the exponentially decaying form, slight differences in the less probable values will have less significance for the χ_m^2 value.

This custom test definition has proven to be useful for the kind of distributions fitted in this work. The χ_m^2 value of each automated fit is stored and manually checked after all fits are carried out. Fits with high χ_m^2 values are reviewed by hand and either manually re-fitted with a different set of starting parameters or accepted without any changes, in case the fit seems acceptable.

4.4.2 Parametrization of Punch-Through Particle Quantities and Particle Types

As a first step the particle quantity distributions are reproduced for each individual punch-through particle type (see figure 4.7 for the cumulative distribution). Therefore the distributions are parameterized for each individual punch-through particle type respectively. As previously mentioned in section 4.3.2, corresponding matter like and antimatter like particles appear at approximately the same rate for electrons, muons and pions, respectively. Therefore the parametrization of either of these particle types is based on cumulative properties of the corresponding matter like and antimatter like particles.

Each particle quantity distribution depends strongly on the input particle's pseudorapidity η_{in} and energy E_{in} . The cause of this was already discussed in section 4.3.3. In order to account for this dependency, the distribution for the number of punch-through particles is parametrized for different η_{in} and E_{in} regions (also referred to as *regimes* or *domains*) independently. Due to detector symmetries in η , the parametrization is actually carried out in different domains of the absolute value $|\eta_{in}|$. Table 4.1 gives the numerical values of the respective domains. The parametrization is realized by fitting an empirical function against the distribution. In this case, a function of the following form shows good fit results:

$$f_{num}(x) = P_0 e^{-P_1 x} + P_2 e^{-P_3 x} + P_4 e^{-P_5 x} \quad (4.4)$$

P_i are the fit parameters and x is the number of punch-through particles of a certain particle type.

Figure 4.13 shows some examples of the automatically computed fits for punch-through pions. The exact same procedure (same fit-function) is carried out for each punch-through particle type within the respective incoming particle regimes.

The fit results (fit parameters P_i) and the minimum and maximum value appearing in the distribution are stored in the look-up table (see appendix A).

Particle Type	E_{in} Regions (GeV)	$ \eta_{in} $ Regions	Total Number of Regions
Muons (μ^- and μ^+)	0 – 50, 50 – 100, ... 450 – 500	0.0 – 0.135, 0.135 – 0.27, ... 2.565 – 2.7	$10 \times 20 = 200$
Pions (π^+ and π^-)			
Photons (γ)			
Protons (p)			
Electrons (e^+ and e^-)			

Table 4.1: Discrete regions defined on the incoming particle’s energy E_{in} and pseudo-rapidity $|\eta_{in}|$. Within each of these regions one set of parameters P_i is determined to describe the punch-through particle occurrence of the respective particle type.

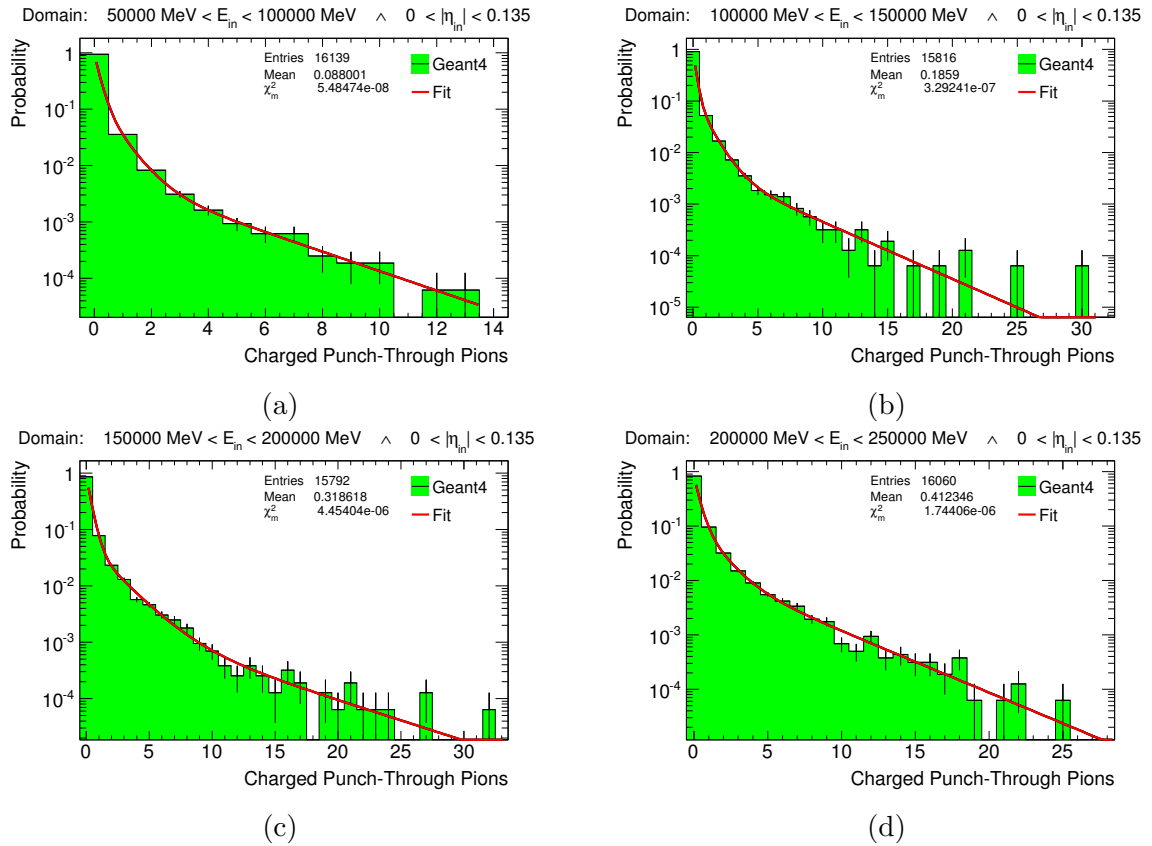


Figure 4.13: Parametrizations of punch-through pion numbers for different $|\eta_{in}|$ and E_{in} regions of the input particle. The underlying Geant4 simulations are single pion events with energies up to 500 GeV and pseudorapidities of $|\eta_{in}| \leq 2.7$. The fit function is a sum of three exponential functions (equation 4.4) and their fit parameters are stored in the look-up table – which will later be used to reproduce these distributions. The shown E_{in} and $|\eta_{in}|$ domains are just examples, a list of all $|\eta_{in}|$ and E_{in} steps is given in table 4.1.

4.4.3 Parametrization of Particle Correlations

The parametrization in the previous section does not take into account correlations between different punch-through particle types. Therefore the correlations are parametrized by an additional procedure.

A correct parametrization (and simulation) of particle quantity distributions (previous section 4.4.2) together with particle correlations, will result in a correct description of the punch-through probabilities (figure 4.5 and 4.6).

Due to the statistical limitations on the Geant4 reference sample (section 4.3.1), the description of the inter-particle correlations are stored in an averaged form, meaning that:

1. the correlation parametrization does not hold any dependency on the initial particle's pseudorapidity η_{in} .
2. only a weak dependency on the initial particle's energy E_{in} is stored by parametrizing within two different energy intervals (*bins*).

Each correlated particle pair is parameterized with the information contained in the correlation plots in figure 4.8. For each E_{in} region, a *normalized* two-dimensional correlation histogram is computed from the Geant4 reference sample. The normalization is carried out along the y-axis, for each respective bin on the x-axis. Thus each x-axis bin contains a probability distribution for the number of punch-through particles of the corresponding y-axis particle type. However, the inverse probability distributions will also be needed during simulation runtime. Therefore the same inter-particle correlation is plotted with mirrored axes and again normalized within each bin of the x-axis, respectively.

These *two* histograms contain the exact same information, but they allow to speed up runtime computation compared to the case where only *one* histogram is used to store each two-particle-correlation. The speed-up is due to the fact the normalizations do not have to be carried out during runtime. These normalized 2D histograms can also be interpreted as matrices and therefore will also be referred to as *correlation matrices*.

In total eight correlation matrices are stored in the look-up table file:

- correlation between number of $\pi^{+/-}$ pions and protons (for E_{in} energy bins 0–200 GeV and 200–500 GeV respectively)
- correlation between number of $e^{+/-}$ and photons (for E_{in} energy bins 0–200 GeV and 200–500 GeV respectively)

Figure 4.14 shows four of these correlation matrices.

Due to the statistical limitation of the underlying Geant4 simulations, the correlation matrices store the exact same statistical fluctuations as were present in the Geant4 event sample. This might become problematic for regions in the correlation matrix where only a few events occurred in the Geant4 simulations – for example for events with more than 40 charged punch-through pions (see figure 4.14 (c)). Due to the low number

of Geant4 simulated events in this region, the probability distributions (for each x-axis bin along the y-axis) mostly only consist of a few filled bins. Clearly this is not an accurate description of the *real* probability distribution for this region. But due to the fact that this region is very rarely addressed in the following simulation, this estimate is a reasonable approximation to start with. However, the validation section (section 6.1.1) does show some effect due to this approximation, but it is regarded as negligible for a fast simulation engine. Finally, a re-parametrization using higher statistics samples can easily be done in the future in order to minimize this effect.

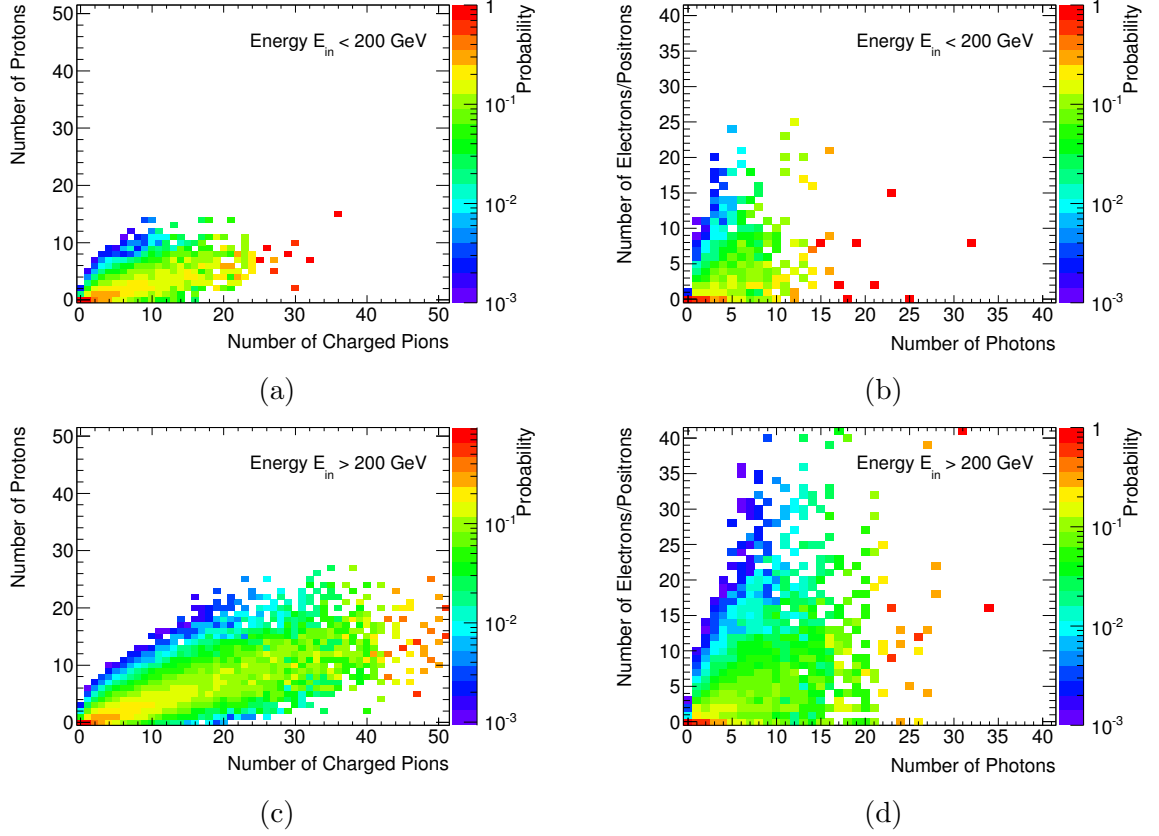


Figure 4.14: Normalized correlation matrices between proton $\leftrightarrow$ charged pion and $\gamma \leftrightarrow e^{+/-}$ punch-through particle types. All 2D-histograms are normalized within each bin of their corresponding x-axis. Therefore, each x-axis bin holds a normalized probability distribution for the corresponding number of punch-through particles of the y-axis type. These plots are used to determine the number of correlated particles, based on a already known number of one type, this is: (a) protons from pions (low initial particle energy E_{in}) (b) electrons/positrons from photons (low E_{in}) (c) protons from pions (high E_{in}) (d) electrons/positrons from photons (high E_{in}).

4.4.4 Parametrization of Particle Energy E_{pt}

The punch-through particle energy E_{pt} (studied in section 4.3.5) is the next important physical quantity which needs to be parametrized. The energy distribution of each relevant punch-through particle type is plotted and parametrized independently. As well as the number of punch-through particles (section 4.4.2), also the punch-through energy distribution strongly depends on the incoming particle's energy E_{in} and pseudorapidity η_{in} , respectively. Therefore the same parametrization approach is used: at first, the incoming particle's E_{in} and η_{in} ranges are divided into distinct regions (domains). Following this approach, also the detector symmetry in η is taken into account, which leads to defining the η_{in} regions by the absolute value $|\eta_{in}|$. The numerical values of these regions are shown in table 4.2. Within each domain, one set of parameters is determined and stored in the look-up table. The parameters are obtained from fitting an empirical function to the distribution. In this case the following function gives good overall fit results:

$$f_E(x) = P_0 \text{landau}_{P_{1,2}}(x) + P_3 \quad (4.5)$$

P_i are the fit parameters and x is the punch-through particle energy for a certain particle type. $\text{landau}_{P_{1,2}}(x)$ is the built-in ROOT TMath::Landau function with two parameters, where P_1 *approximately* corresponds to the most probable value (MPV) and P_2 is given the name *scale parameter* sigma [29].

An example set of automatically carried out fits is shown in figure 4.15. These fits are done for the whole range of the incoming particle's energy E_{in} and η_{in} – and for each punch-through particle type respectively.

Particle Type	E_{in} Regions (GeV)	$ \eta_{in} $ Regions	Total Number of Regions
Muons (μ^- and μ^+)	0 – 62.5, 62.5 – 125, ... 437.5 – 500	0.0 – 0.45,	$8 \times 6 = 48$
Pions (π^+ and π^-)		0.45 – 0.9,	
Photons (γ)		...	
Protons (p)		2.25 – 2.7	
Electrons (e^+ and e^-)			

Table 4.2: Discrete regions defined on the incoming particle's energy E_{in} and $|\eta_{in}|$. Within each of these domains one set of parameters is determined to describe the punch-through particle energies of the respective particle type.

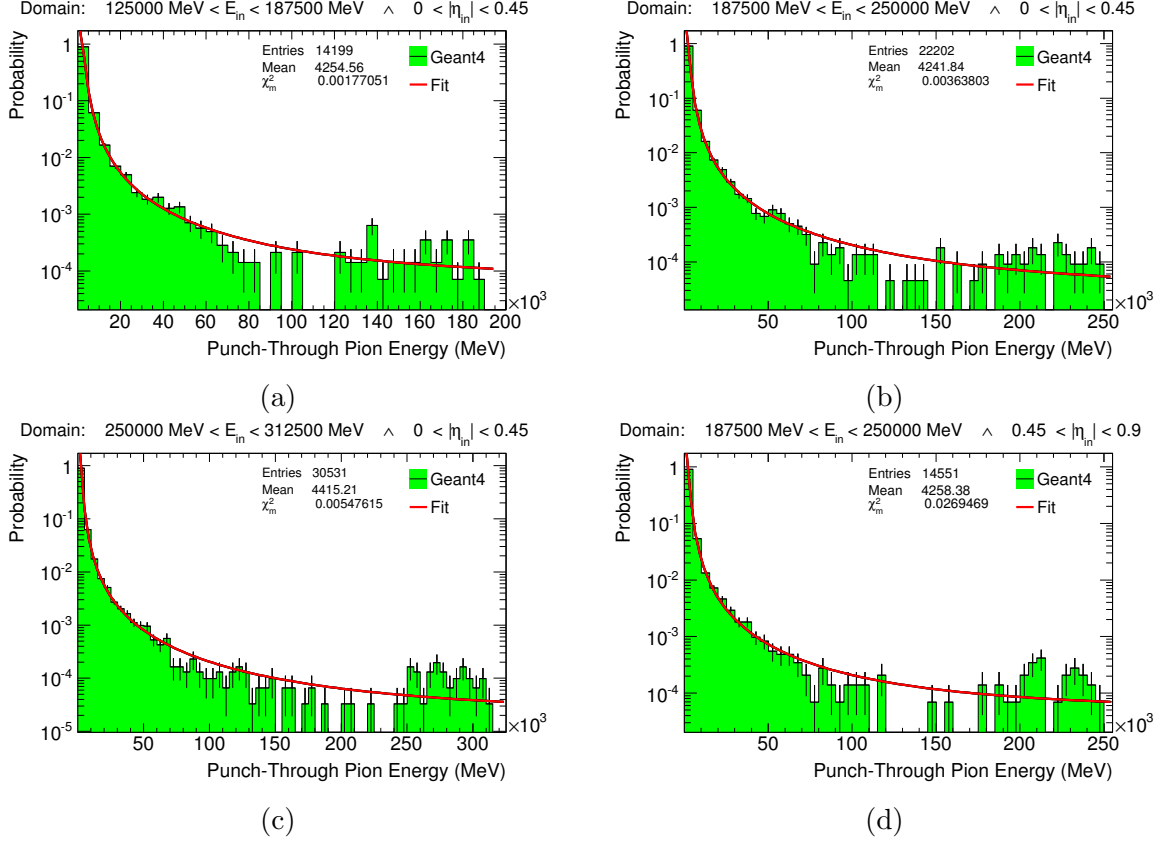


Figure 4.15: Punch-through energy distribution for pions in different $|\eta_{in}|$ and E_{in} regions of the incoming particle. The Geant4 data are obtained from simulated single pion events – details given in section 4.3.1. The shown fits use an empirical function, defined in equation 4.5 and their fit parameters are stored in the look-up table, which will later be used to reproduce these distributions. The shown E_{in} and $|\eta_{in}|$ domains are just examples, a list of all $|\eta_{in}|$ and E_{in} steps is given in table 4.2.

4.4.5 Parametrization of Particle Deflection Angles $\Delta\phi$ and $\Delta\theta$

The punch-through particle deflection angles (relative to the incoming particle) are the next physical properties which need to be parametrized in order to reproduce them in a fast calorimeter punch-through simulation. A deeper understanding of $\Delta\phi$ and $\Delta\theta$ was already given in section 4.3.6 (defined by equation 4.1).

In order to suppress statistical fluctuations, the symmetry of the $\Delta\theta$ and $\Delta\phi$ distributions are taken into account, therefore the parametrization is done with the corresponding absolute value distributions: $|\Delta\phi|$ and $|\Delta\theta|$. However this symmetry does not fully apply to the $|\Delta\theta|$ distribution, due to effects caused by the calorimeter material distribution (gaps) discussed in section 4.3.6. This effect is considered to be negligible for this work and a correct description might be subject to future improvements of this

parametrized simulation.

The parametrization utilizes a similar procedure previously used for particle occurrence (section 4.4.2) and particle energy (section 4.4.4). However, instead of parametrizing within discrete energy and $|\eta_{in}|$ regions of the incoming particle, the domains are now defined by:

1. the absolute value of the incoming particle's pseudorapidity: $|\eta_{in}|$
2. the energy of the corresponding punch-through particle type: E_{pt}

The energy of the simulated punch-through particle E_{pt} is used instead of the initial particle's energy, due to the fact that E_{pt} already describes a physical property of the punch-through particle itself and therefore a correlation with any deflection angle seems more evident than a correlation between the initial particle's energy E_{in} and $|\Delta\phi|$ or $|\Delta\theta|$. Studies on Geant4 simulated events support this conclusion (figure 4.16): the $\Delta\theta$ distribution shows a much stronger dependency on the E_{pt} than on E_{in} .

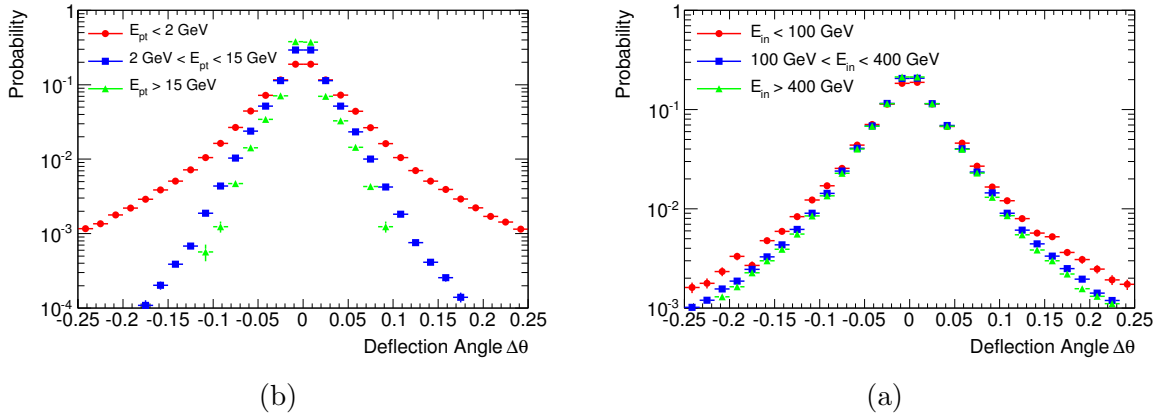


Figure 4.16: The punch-through particle deflection angle $\Delta\theta$ shows a much stronger correlation with the energy of the punch-through particle itself E_{pt} than with the initial particle's energy E_{in} .

The number of fit domains are given in table 4.3. Within each of these regions (and for each particle type, respectively) the $|\Delta\phi|$ distribution obtained from the Geant4 simulations is fitted with the following empirical function:

$$f_{\phi}(x) = \text{gaus}_{P_{0,1,2}}(x) + \text{expo}_{P_{3,4}}(x) = P_0 e^{-\frac{1}{2}\left(\frac{x-P_1}{P_2}\right)^2} + e^{P_3+P_4 x} \quad (4.6)$$

Where $\text{gaus}_{P_{0,1,2}}(x)$ describes the ROOT built-in Gaussian function with three parameters, $\text{expo}_{P_{3,4}}(x)$ is the ROOT built-in exponential function with two parameters, P_i are the parameters determined by the fit method and x is the deflection angle $|\Delta\phi|$ of the respective particle type.

For $|\Delta\theta|$ distributions, the following empirical function is used:

$$\begin{aligned} f_{\theta}(x) &= \text{gaus}_{P_{0,1,2}}(x) + \text{gaus}_{P_{3,4,5}}(x) \\ &= P_0 e^{-\frac{1}{2}\left(\frac{x-P_1}{P_2}\right)^2} + P_3 e^{-\frac{1}{2}\left(\frac{x-P_4}{P_5}\right)^2} \end{aligned} \quad (4.7)$$

Here $\text{gaus}_{P_{a,b,c}}(x)$ describes the ROOT built-in Gaussian function with three parameters, P_i are the fit parameters and x is the deflection angle $|\Delta\theta|$ of the particular particle type.

A set of example fits obtained from the automatic fit procedure is depicted in figure 4.18 for $|\Delta\theta|$ and 4.17 for $|\Delta\phi|$.

Particle Type	Number of E_{pt} Regions	$ \eta_{in} $ Regions	Total Number of Regions
Muons (μ^- and μ^+)	5	0.0 – 0.45, 0.45 – 0.9, ... 2.25 – 2.7	$5 \times 6 = 30$
Pions (π^+ and π^-)	8		$8 \times 6 = 48$
Photons (γ)			
Protons (p)			
Electrons (e^+ and e^-)			

Table 4.3: Discrete fit domains defined on the incoming particle’s pseudorapidity $|\eta_{in}|$ and the respective punch-through particle energy E_{pt} . Within each of these regions two sets of parameters are determined to describe the punch-through particle deflection angles $|\Delta\phi|$ and $|\Delta\theta|$ of the corresponding particle type. The numerical values defining the domain boundaries in E_{pt} depend on the occurring punch-through particle energies in the Geant4 simulations, therefore they are generated dynamically during the automated fit procedure. Thus only the number of regions in E_{pt} is given in this table.

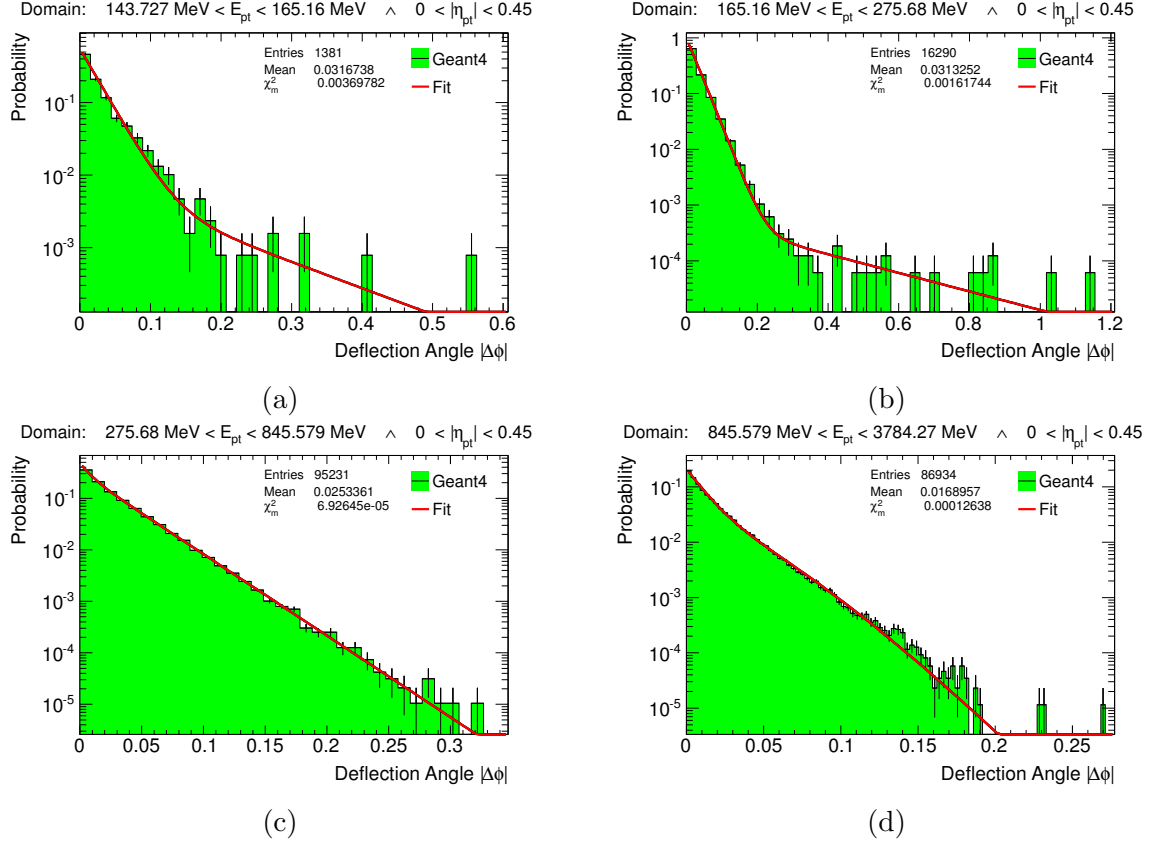


Figure 4.17: An example set of automatically carried out fits of Geant4 punch-through pion $|\Delta\phi|$ distributions for different incoming particle's pseudorapidity $|\eta_{in}|$ and punch-through particle energy E_{pt} regions. The Geant4 data are obtained from simulated single pion events – details given in section 4.3.1. The shown fits use an empirical function given in equation 4.6 and their fit parameters are stored in the look-up table, which will later be used to resimulate these distributions. A list of all $|\eta_{in}|$ and E_{pt} steps used for the $|\Delta\phi|$ parametrization is shown in table 4.3.

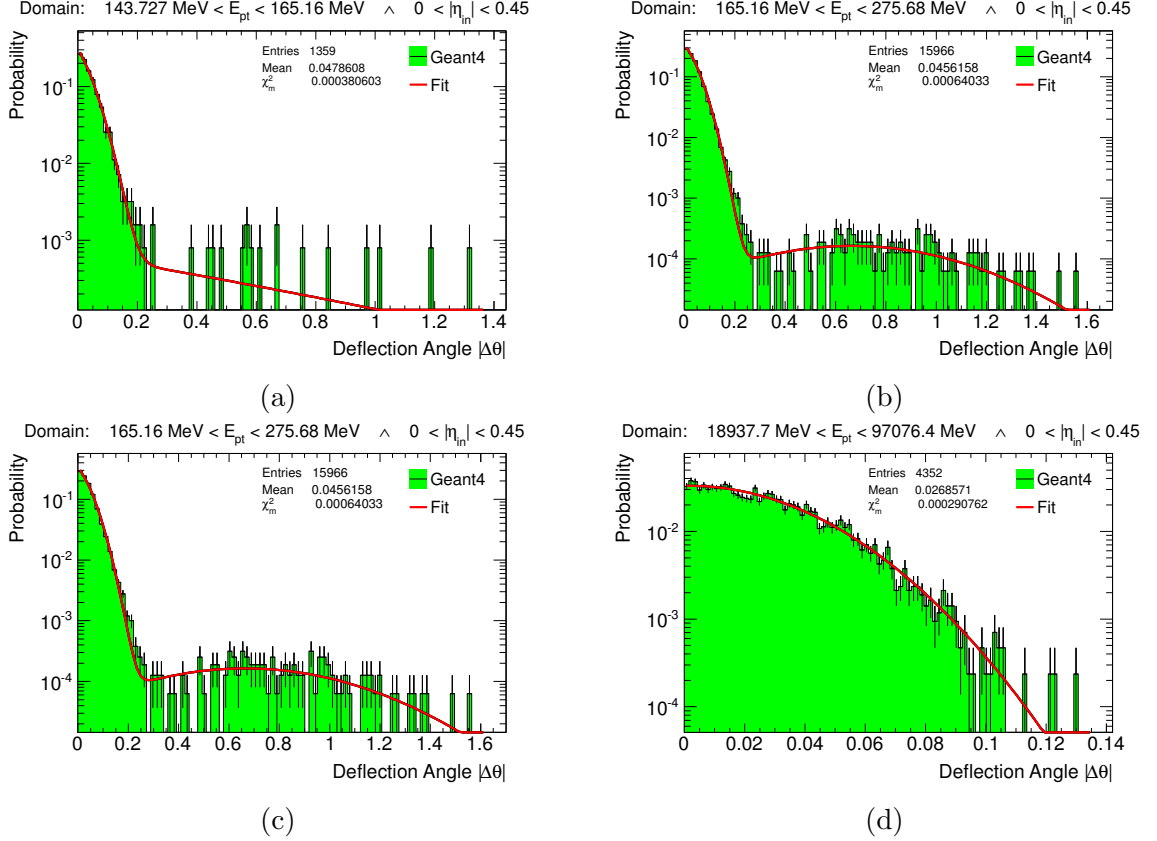


Figure 4.18: An example set of automatically carried out fits of Geant4 punch-through pion $|\Delta\theta|$ distributions for different incoming particle's pseudorapidity $|\eta_{in}|$ and punch-through particle energy E_{pt} regions. The Geant4 data are obtained from simulated single pion events – details given in section 4.3.1. The shown fits use an empirical function, defined in equation 4.7 and their fit parameters are stored in the look-up table, which will later be used to resimulate these distributions. A list of all $|\eta_{in}|$ and E_{pt} steps used for the $|\Delta\theta|$ parametrization is shown in table 4.3.

4.4.6 Parametrization of Particle Momentum Direction ($\Delta\phi_p$ and $\Delta\theta_p$)

The remaining quantities to be parameterized are the angles $\Delta\phi_p$ and $\Delta\theta_p$ (equation 4.2), describing the momentum direction of punch-through particles. In section 4.3.7 the impact of the momentum direction on the reconstruction algorithms in the muon spectrometer was already discussed. Prior to adding this parameter to the simulation, the observed rate of reconstructed tracks due to punch-through particles in the MS was too high.

Due to the symmetry of both distributions, $\Delta\phi_p$ and $\Delta\theta_p$ (figure 4.12), the parametrization is applied on their corresponding absolute value distributions $|\Delta\phi_p|$ and $|\Delta\theta_p|$. Following a similar approach taken in the previous sections, the distributions are parameterized for each single punch-through particle type in different punch-through particle energy E_{pt} and pseudorapidity $|\eta_{pt}|$ regions (see table 4.4). Within each of these regions the distributions are fit with an empirically determined gaussian function:

$$f_\phi(x) = \text{gaus}_{P_{0,1,2}}(x) = P_0 e^{-\frac{1}{2}\left(\frac{x-P_1}{P_2}\right)^2} \quad (4.8)$$

Where $\text{gaus}_{P_{0,1,2}}(x)$ describes the ROOT built-in Gaussian function with three parameters, x is either of the momentum deflection angles $|\Delta\phi_p|$ or $|\Delta\theta_p|$ and P_i are the parameters determined by the fit method and to be stored in the look-up table.

Particle Type	Number of E_{pt} Regions	$ \eta_{pt} $ Regions	Total Number of Regions
Muons (μ^- and μ^+)	5	0.0 – 0.45, 0.45 – 0.9, ... 2.25 – 2.7	$5 \times 6 = 30$
Pions (π^+ and π^-)	8		$8 \times 6 = 48$
Photons (γ)			
Protons (p)			
Electrons (e^+ and e^-)			

Table 4.4: Discrete fit domains defined on punch-through particle’s pseudorapidity $|\eta_{pt}|$ and energy E_{pt} . Within each of these regions two sets of parameters are determined to describe the punch-through particle momentum angles $|\Delta\phi_p|$ and $|\Delta\theta_p|$ of the corresponding particle type. The numerical values defining the domain boundaries in E_{pt} depend on the occurring punch-through particle energies in the Geant4 simulations. Therefore they are generated dynamically during the automated fit procedure with an exponentially growing domain size. Thus only the number of regions in E_{pt} is given in this table.

Figure 4.19 shows some example fits for both distributions.

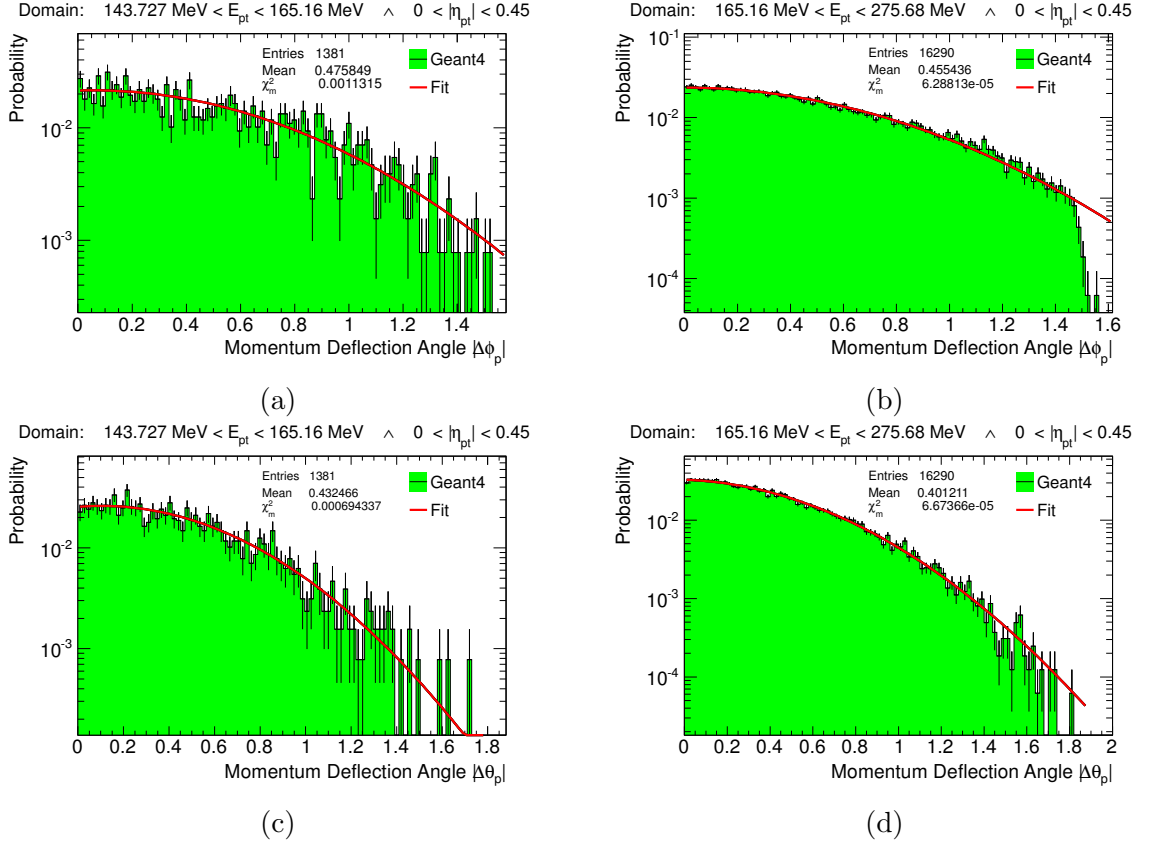


Figure 4.19: An example set of automatically carried out fits of (a,b) $|\Delta\phi_p|$ and (c,d) $|\Delta\theta_p|$ momentum direction distributions for pion punch-through particles. Parametrizations for different regions of punch-through particle's pseudorapidity $|\eta_{pt}|$ and energy E_{pt} are shown. The Geant4 data are obtained from simulated single $\pi^{+/-}$ events with energies up to 500 GeV – details given in section 4.3.1. Each shown fit uses a gaussian function and their fit parameters are stored in the look-up table, which will later be used to re-simulate these distributions. A list of all $|\eta_{pt}|$ and E_{pt} steps used for this parametrization is given in table 4.4.

4.5 Parametrized Simulation

Finally, to reproduce the punch-through particle properties a fast parametrized Monte Carlo simulation was implemented. This simulation is completely implemented within the Athena framework (section 3.3). Thus it can be easily integrated into any kind of ATLAS detector simulation.

The simulation input and output is described in section 4.5.1. Furthermore, the simulation output can be modified (or tuned) to match the user's need, by adjusting the simulation parameters described in section 4.5.2. Finally, the basic application flow is described in section 4.5.3.

In principle this simulation is independent of any particular ATLAS detector simulation, however an implementation is fully integrated into the fast track simulation Fatras (see section 5.3 for the punch-through integration, and section 3.2.2 for details on Fatras). In order to validate the effects on the ATLAS muon spectrometer, an integration into AtlfastII (see section 3.2.3) was carried out as well. Details about the implemented simulation code and the C++ classes are given in chapter 5.

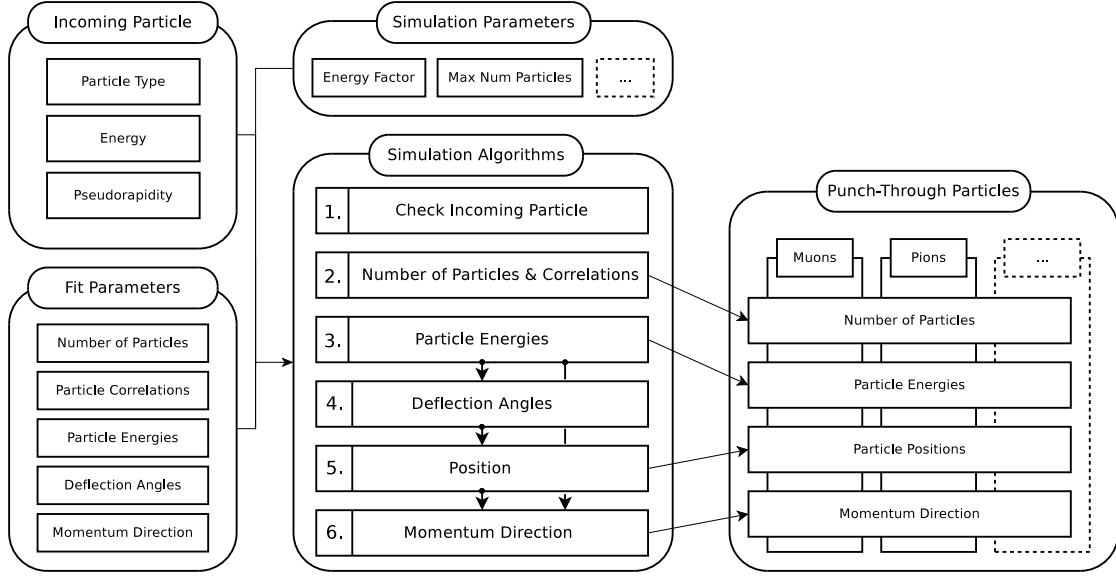


Figure 4.20: A schematic representation of the application flow for the parametrized calorimeter punch-through simulation implemented in this work. The simulation output (the punch-through particles) depends on the properties of the incoming particle, the parametrization (stored in a look-up table file) and the simulation parameters (given via Python `jobOptions`). Before any simulation step is run, the incoming particle is checked for its relevance to the punch-through simulation. In the following first simulation step, the number of particles for each individual punch-through particle type are determined. While doing so, the corresponding particle correlations are taken into account. Furthermore, the individual energies for these particles are computed. Based on the particle energy, the deflection angles are determined and by using them the particle position is computed. Finally, the direction of each particle’s momentum is determined, based on its already determined energy and position.

4.5.1 Simulation Input and Output

The simulation takes one particle at a time as input. From this, a set of punch-through particles are generated according to the parametrized distributions saved in the look-up table file and the parameters set by the user (section 4.5.2). In most cases, no particles will be created at all, which means that the incoming particle is completely absorbed by the ATLAS calorimeter. In simulated signal events more than one particle per event will enter the calorimeter. In this case the punch-through simulation has to be called several times, for each single particle respectively. This implies the reasonable assumption that the particle interaction processes inside the calorimeter are uncorrelated between different initial particles entering the calorimeter.

Due to the definition for a punch-through particle while applying the parametrization (section 4.3.1), the input particle should be positioned on a geometrical surface

called *CaloEntryLayer* (see appendix B). The generated punch-through particles will be positioned on the *MuonEntryLayer*. Both layers do not cross any sensitive detector elements, therefore they were chosen as reference surfaces. In case the input particle's position is not on the *CaloEntryLayer* surface or a given tolerance around the surface, the simulation will not treat the particle to cause any punch-through effects. The error made by particles not positioned *on* the reference surface, but within a reasonable tolerance, is assumed to be negligible for this fast punch-through simulation.

4.5.2 Simulation Parameters

Fundamental simulation properties are easily modifiable through a set of parameters accessible via Python `jobOptions` (see section 3.3). These parameters are provided by the `PunchThroughSimulator` C++ class, explained in detail in section 5.1.

Many parameters (especially scale factors) allow to tune the simulation – e.g. against a given reference sample – or to use this simulation for studies on systematic errors due to punch-through effects.

The following list briefly explains the meaning and effects of these parameters:

Punch-Through Initiators give the incoming particle type for which this simulation applies. The particle type is given according to the Monte Carlo numbering scheme [25]. In Python this parameter is named `PunchThroughInitiators`. The simulation does not differentiate between an incoming matter like or anti-matter like particle type. For both, the exact same punch-through effects will be simulated. Currently only one initiator particle type can be registered to the simulation, but it will be subject for future improvements to add punch-through simulation capability for a variety of initial particle types, each with its own parametrization.

Punch-Through Particle Types have to be registered individually to the punch-through simulator via the parameter `PunchThroughParticles`. Only particle types given in this list are generated as punch-through particles.

Anti-Particle Creation: For each of the particle types given by the previous parameter, a Boolean value will determine if its corresponding anti-particle type will also be generated as punch-through particle. If enabled, the simulation will create matter like and anti-matter like particles according to a same common parametrization. This parameter is accessible as `DoAntiParticles` in Python.

Correlated Particle: Each punch-through particle-type can have up to one corresponding correlated particle type, which has to be given by the parameter `CorrelatedParticle` according to the Monte Carlo numbering scheme.

Minimum Particle Energy: Punch-through particles below a certain energy threshold will not be generated. This value can be modified for each particle type respectively (`MinEnergy`).

Maximum Number of Particles: An upper threshold for the number of punch-through particles for each respective particle-type (`MaxNumParticles`).

Number of Particles Factor: The output of punch-through particles for each respective particle-type according to the parametrization will be multiplied by this factor (`NumParticlesFactor`).

Energy Factor: The particle energies obtained from the parametrization will be multiplied by this factor for each respective particle type (`EnergyFactor`).

Correlation Energy Thresholds: For incoming particles below the `MinCorrelationEnergy` energy threshold, no punch-through particle correlations are taken into account. Above this threshold, the chance to simulate correlations increases linearly until the upper threshold `FullCorrelationEnergy` is reached. For energies above `FullCorrelationEnergy`, full particle correlations are simulated.

Position and Momentum Deflection Angles: The factors given by the parameters `ScalePosDeflectionAngles` and `ScaleMomDeflectionAngles` scale the corresponding deflection angles ($\Delta\phi$ and $\Delta\theta$ or $\Delta\phi_p$ and $\Delta\theta_p$) determined by the simulation.

Tolerance to Reference Surface: The parameterized simulation will be applied to any input particle which has its position inside this tolerance around the `CaloEntryLayer` surface. Any particle outside this tolerance will be rejected by the simulation. The value has to be given in millimeters. (Python name: `ReferenceSurfaceTolerance`)

Barcode Offset: This simulation internal parameter is used to give a starting number for the punch-through particle barcodes, where each particle barcode is unique within a simulated event. The Python name of this parameter is `BarcodeOffset`.

4.5.3 Simulation Scheme

Figure 4.20 shows a schematic overview of the application flow for the parameterized punch-through simulation. The punch-through particle properties are determined in the same sequence as the parametrization is described in section 4.4.

1. **Check Incoming Particle:** Before any of the parametrization is applied, the incoming particle will be checked for its type. If this particle type is not registered to the simulation as a *punch-through initiator* (section 4.5.2), it will be ignored by the simulation and no additional punch-through particles are created – this corresponds to the case where the incoming particle is completely absorbed by the ATLAS calorimeter. Different to the other steps, no *simulation* of particle properties is carried out within this first step.
2. **Number of Punch-Through Particles and Correlations:** After the previous step has been passed, the quantity of punch-through particles for each respective punch-through particle type is determined according to the parametrization. Within this

step, also the registered particle correlations are taken into account. Therefore, for each two-particle correlation, an equally random choice will be made on which particle type to treat first. The quantity of particles for the *first* type will then be determined in exactly the same way as for any uncorrelated particle type: the parametrization depending on the incoming particle's energy E_{in} and pseudorapidity η_{in} (section 4.4.2) is used to do so. Uncorrelated particles are done with this step, but for particle types with correlations, the correlation matrices (section 4.4.3) are used to determine the number of punch-through particles of the corresponding *second* particle type – which has not been determined yet.

The reason for randomly choosing which of the two correlated particle types to treat first, lies in the fact that the parametrization of each independent punch-through particle type correctly describes dependencies in E_{in} and η_{in} , but the correlation matrices are just averaged over the whole E_{in} and η_{in} range. By applying this method, an averaged description of the dependencies on E_{in} and η_{in} together with the correlations is achieved.

- 3. Particle Energies:** At this stage the quantity of punch-through particles is already computed and the energies of each respective particle is determined. Therefore the E_{in} and η_{in} dependent parametrization (section 4.4.4) is applied. This parametrization already takes the result of the previous step into account by applying different parametrizations for different punch-through particle types.
- 4. Deflection Angles $\Delta\phi$ and $\Delta\theta$:** The next physical properties computed by applying parametrization models, are the deflection angles. The angles $\Delta\phi$ and $\Delta\theta$ are treated independently from another. The underlying parametrization is described in section 4.4.5. Therefore the dependency on the incoming particle's pseudorapidity η_{in} and the respective punch-through particle energy E_{pt} will be reproduced. E_{pt} was determined in the previous step and an illustration of its effect on the deflection angles is given in figure 4.16. Due to their symmetry, the deflection angle distributions are only given as absolute value distributions. Thus, for each deflection angle determined via the parametrization, an equally random choice is made for the value to have a positive or negative sign.
- 5. Particle Position:** From the recently computed deflection angles, the exact position on the *MuonEntryLayer* surface is computed for each individual punch-through particle. The generated punch-through particles need to have a position on this surface, simply due to the fact that all parametrizations were carried out under the definition of a punch-through particle to be on this surface (section 4.3.1).

In order to do so, the individual particle positions are determined by calculating the intersection of a straight line (originating at interaction point $(x, y, z) = (0, 0, 0)$) with the *MuonEntryLayer* surface. Hereby the orientation of this straight line is chosen according to the global θ and ϕ values, obtained by adding the deflection angles to the corresponding angles of the incoming particle's position on the

CaloEntryLayer surface:

$$\begin{aligned}\theta &= \theta_{in} + \Delta\theta \\ \phi &= \phi_{in} + \Delta\phi\end{aligned}\tag{4.9}$$

Figure 4.21 gives a geometrical illustration for the process of finding the punch-through particle position.

6. Particle Momentum: The final step of the simulation is to compute the punch-through particle momentum vector $\vec{p}$. Its direction is determined via the angles $\Delta\phi_p$ and $\Delta\theta_p$, which are parameterized depending on the previously computed particle energy E_{pt} and pseudorapidity η_{pt} . The momentum direction in local coordinates (centered at the punch-through particle position) is computed by adding the momentum deflection angles to the respective punch-through particle global coordinates:

$$\begin{aligned}\theta_p &= \theta_{pt} + \Delta\theta_p \\ \phi_p &= \phi_{pt} + \Delta\phi_p\end{aligned}\tag{4.10}$$

Its magnitude is given by the already determined energy E and mass m of the respective particle:

$$|\vec{p}| = \sqrt{p_x^2 + p_y^2 + p_z^2} = \sqrt{E^2 - m^2}\tag{4.11}$$

Figure 4.21 geometrically explains the process of computing the punch-through particle position and momentum vector.

After all necessary properties have been computed for the punch-through particles, the collection of particles is returned by the simulation and handed over to the overlying detector simulation – which might be Fatras, AtlfastII or any other kind of detector simulation, applying a fast punch-through simulation.

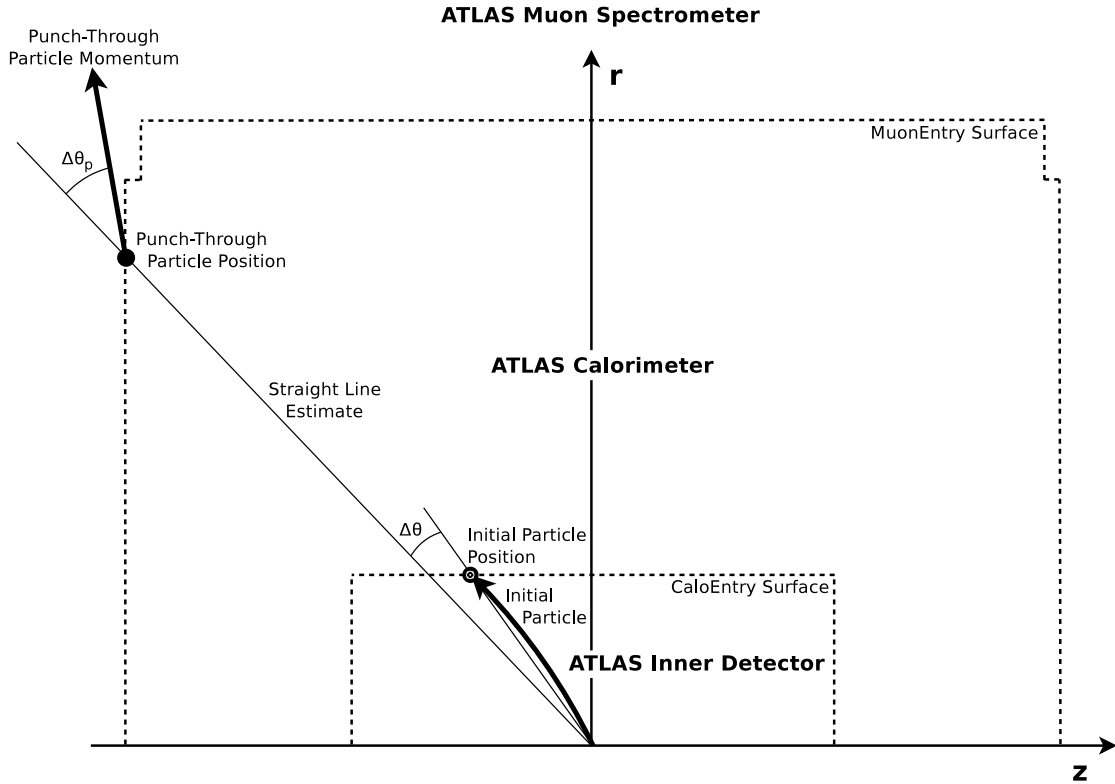


Figure 4.21: The punch-through particle position is determined by intersecting a straight line with the *MuonEntryLayer* surface. This line originates at the interaction point and its orientation is given by the deflection angles $\Delta\phi$ and $\Delta\theta$ relative to the initial particle's position. The momentum is determined relative to the punch-through particle position via the angles $\Delta\phi_p$ and $\Delta\theta_p$.

Chapter 5

Implementation

In section 4.5 an overview of the simulation concepts and its basic application flow was given. This chapter adds a detailed explanation of the underlying C++ implementation. The main punch-through simulator class and its functionality is explained in section 5.1. The method used to generate random numbers with a desired distribution is discussed in section 5.2. Due to the fact that this work was first implemented within Fatras (see section 3.2.2) most of its C++ classes and data types are fit into the Fatras framework. The code resides therefore in the `Simulation/Fatras` container and its subpackages. The Fatras integration is discussed in section 5.3.

Adapting this simulation to other projects, such as the future *integrated simulation framework*, may result in a change of the interface classes and the namespace.

The implementation is compatible to Athena release 17.0.2.6. The C++ source code is fully documented using the documentation generator Doxygen [30].

5.1 Punch-Through Simulation

Class Summary	
C++ Class Name	<code>Fatras::PunchThroughSimulator</code>
Contained in package	<code>FatrasTools</code>
Compatible interface	<code>Fatras::IParticleStateCollectionCreator</code>

As the C++ class name already suggests, the `Fatras::PunchThroughSimulator` Athena tool is the main component of the fast calorimeter punch-through simulation. It uses a parameterized look-up table to reproduce certain parameter relations, distributions and correlations. The content of this look-up table is described in more detail in appendix A.

The `PunchThroughSimulator` takes *one* `Fatras::ParticleState` object at input, which describes one particle at a given time and position in the detector. To run the punch-through simulation for multiple input particles, it has to be called several times – once for each input particle. In Fatras this is carried out by the `Fa-`

`trass::SimulationKernel` (see section 5.3.2). The simulation returns a filled `ParticleStateCollection` in case punch-through particles are created, or a pointer to zero, in case no particles are created.

The parametrization requires the input particle's position to be on the surface of the so called *CaloEntryLayer* (see appendix B). In the Fatras implementation this is ensured by the `ExtrapolationTool` (section 5.3.5). All quantities that are determined by the punch-through simulation depend on at least one of the following parameters (or derived ones) of this input particle:

- particle type described by the Monte Carlo particle numbering scheme [25]
- particle energy E_{in}
- pseudorapidity η_{in} of the particle position when entering the ATLAS calorimeter

With the use of the look-up table and the given input parameters from above, the punch-through simulator uses the `Fatras::PDFcreator` (see section 5.2) to derive the following quantities for the punch-through particles in this order:

1. number of particles and correlations for each particle type
2. particle energies E_{pt}
3. deflection angle $\Delta\phi$
4. deflection angle $\Delta\theta$
5. position of punch-through particle
6. direction of momentum (from $\Delta\phi_p$ and $\Delta\theta_p$)

All punch-through particles created by this simulation module are positioned on the *MuonEntryLayer* (appendix B) (which surrounds the ATLAS calorimeter). Therefore these particles can directly be handed over to any subsequent ATLAS muon spectrometer simulation.

In order to gain maximum flexibility, many core elements of the punch-through simulation can be controlled via (tuning) parameters (section 4.5.2), accessible in the Python `jobOptions` files (section 3.3). The parameters are read during the initialization phase of this Athena tool. Based on these parameters, `Fatras::PDFcreator` objects are created for the different punch-through particle properties for each respective punch-through particle type registered via the `PunchThroughParticles` parameter.

The parameters `PunchThroughParticles`, `DoAntiParticles`, `CorrelatedParticle`, `(MinEnergy, MaxNumParticles, NumParticlesFactor` and `EnergyFactor` are Python vectors, where each entry corresponds to one punch-through particle type. Therefore each of these vectors should have the same length. If this is not the case, the simulation throws a warning message but continues its execution with default values for the missing parameters. Even though the simulation might then be unable to create all desired particle types.

Only if the given input particle type is registered to the `PunchThroughSimulator` (via the `PunchThroughInitiators` Python parameter) the simulation continues in the order of the following subsections. If this is not the case, the simulation ends here and returns a zero value. In which case the Fatras implementation then continues its execution in `Fatras::CaloSimulationTool`.

5.1.1 Number of Punch-Through Particles, Particle Type and Correlations

The `PunchThroughSimulator` produces punch-through particles of each type that is registered via the `PunchThroughParticles` parameter. In case a correlation is registered for two particle types, a random number decides which of the two particle types to treat first (with equal chances). This decision is made for each correlated pair of particle types. Each chosen particle type is then treated in the exact same way as an uncorrelated particle type.

The corresponding `Fatras::PDFcreator` object is used to generate a random number for the quantity of punch-through particles of the currently processed particle type. To do so, the `PDFcreator` uses a generic fit function with a number of fit parameters (section 4.4) obtained from the look-up table file. This function – together with the fit parameters – describes the probability distributions for different punch-through particle types. The `PDFcreator` automatically chooses the correct fit parameters corresponding to the current input particle energy E_{in} and pseudorapidity $|\eta_{in}|$.

For a particle types with no correlation, the process of finding its number of punch-through particles is finished at this point. For correlated particle types, the quantity of particles of the previously not-selected particle type still has to be computed. This is done via correlation matrices (section 4.4.3) stored in the look-up table. Each correlation matrix is averaged and does only depend on one parameter: the already computed quantity of particles for the first particle type. By using this, a discrete probability distribution for the number of particles of the second particle type is obtained. A random number according to this distribution is obtained, by utilizing the *inverse transform sampling* method [31], which is based on uniformly distributed random numbers. This method is utilized for each pair of correlated punch-through particle types.

Clearly, if no punch-through particles appear at all, the punch-through simulation ends at this point. In this case, the `PunchThroughSimulator` returns a zero value.

5.1.2 Energy of Punch-Through Particles

The next step in the punch-through simulation is the computation of the respective punch-through particle energies. Due to the parametrization (section 4.4.4) this is carried out for each punch-through particle type independently. Again, the `PDFcreator` instances are used to compute the corresponding punch-through particle energies. The dependency on the incoming particle's energy E_{in} and pseudorapidity η_{in} is taken into account by doing so.

In order to guarantee energy conservation, a simple test is carried out after each particle energy computation: In case the sum of the individual particle energies exceeds the incoming particle's energy, all following punch-through particles are dropped, even though the previous step (section 5.1.1) would have determined a greater number punch-through particles.

5.1.3 Deflection Angles $\Delta\phi$ and $\Delta\theta$

A number of `Fatras::PDFcreator` instances is used to compute the deflection angles $\Delta\phi$ and $\Delta\theta$ for each respective punch-through particle. The dependencies on the incoming particle's pseudorapidity η_{in} , the respective punch-through particle energy E_{pt} and particle type are taken into account.

5.1.4 Particle Position and Momentum Direction

Each single punch-through particle position is determined by the corresponding angles ϕ and θ (according to equation 4.9). The simulation takes care to keep these angles within their defined range, respectively:

$$\theta \in [0, \pi] \quad (5.1)$$

$$\phi \in [-\pi, \pi] \quad (5.2)$$

These angles are used to compute a corresponding position on the `MuonEntryLayer` surface (figure 4.21), which is build as the intersection of a straight line from the origin to the surface. Mathematically, the line is described by $\vec{R}(l)$, where l is a parameter corresponding to the distance from the origin:

$$\vec{R}(l) = \begin{pmatrix} \cos \phi \sin \theta \\ \sin \phi \sin \theta \\ \cos \theta \end{pmatrix} \cdot l$$

The `MuonEntryLayer` surface consists of individual cylinder- and disc surfaces. They are arranged with cylindrical symmetry around the z-axis. Therefore simple analytical solutions exist to describe the intersection coordinates by the parameter l :

$$l_{disc} = \pm \frac{z_{disc}}{\cos \theta}$$

$$l_{cyl} = \pm \frac{r_{cyl}}{\sin \theta}$$

In this simulation only the positive results are taken. The simplified cylindrical representation used above does not reflect the actual shape of the entry layer surface (see appendix B). Therefore a `Fatras::EntryLayerTool` is used to get ATLAS tracking geometry representations [18] for the corresponding `MuonEntryLayer` parts. Here, only two different surface types are used: `Trk::CylinderSurface` and `Trk::DiscSurface`.

The simulation computes the intersection coordinates on all surfaces, positioned in the same half space ($+z$ or $-z$) where the straight line is pointing to. For each of these surfaces, the tracking geometry built-in `isOnSurface` methods are used to determine if the intersection lies within the respective surface boundaries. By doing so, exactly one valid intersection remains, which is taken as the punch-through particle position.

The punch-through particle momentum is then set to be collinear with the straight line used to find its position on the `MuonEntryLayer` surface.

5.2 Distributed Random Number Generation

Class Summary	
C++ Class Name	<code>Fatras::PDFcreator</code>
Contained in package	<code>FatrasTools</code>

The `PDFcreator` generates discrete or continuous random numbers according to a distribution given by a ROOT TF1 [29] function object. The TF1 object usually depends on a set of settable *function parameters* (fit parameters in this case). The `PDFcreator` determines the TF1 function parameters based on the *input* parameters given as arguments to the `PDFcreator::getRand(...)` method. The *input* parameters do not necessarily correspond to the *function* parameters. An example for input parameters are E_{pt} and η_{in} when computing particle deflection angles (section 5.1.3).

The relation between input parameters and function parameters is given via ROOT TH1 histograms. The channels (bins) of the histogram correspond to intervals of an input parameter value and the content in a certain channel holds the corresponding function parameter value (which is usually a floating point value). Note that the ROOT TH1 1D histogram base class also serves as a base class for both 2D ROOT histograms (TH2) and 3D ROOT histograms. Therefore, a n -dimensional histogram is used, to get one function parameter out of n input parameters. Most of the probability density functions used in this work are described with more than one function parameter. Therefore, a set of TH1 histograms is needed to describe the complete function for a certain set of input parameters.

The use of histograms to store function parameters for different input parameter intervals results in non-continuous transitions between these intervals. This could have negative effects on the physical result of the simulation. Therefore a fast interpolation method was implemented, as described in section 5.2.3.

By calling the `PDFcreator::getRand(...)` method, the function parameters will be determined based on the given set of input parameters. Therefore the TF1 function is then fully defined and a random number is computed according to the the function's distribution. The exact process of computing these random numbers is discussed in subsection 5.2.1 for discrete random numbers and 5.2.2 for continuous random numbers.

All ROOT TF1 functions and ROOT TH1 histograms used by `PDFcreator` instances

inside the `Fatras::PunchThroughSimulator`, are stored in the punch-through parameter look-up table (see appendix A and A.1 in particular).

5.2.1 Discrete Random Numbers

In order to compute discrete random numbers, the `PDFcreator::getRand(...)` utilizes a discrete inverse transform sampling method. For this, the ROOT TF1 function was already fully defined and all function parameters were set by the previously described step. First, a uniformly distributed random number between zero and one is computed. The requested discrete random number is then found by computing the cumulative distribution function until it exceeds the previously computed random number.

5.2.2 Continuous Random Numbers

Continuous random numbers are computed via the *rejection sampling* method [32]. Given this method, multiple attempts might be necessary before the final random number is determined. For each attempt, a pair of random numbers is used.

5.2.3 Parameter Interpolation

As previously mentioned, the function parameters are stored in ROOT TH1 compatible histogram instances. A *nearest neighbor* interpolation is used when reading the histogram content (TF1 function parameters) corresponding to a given input parameter. Therefore the values of these function parameters are discrete and constant within certain input parameter intervals. This results in constant distribution functions within certain input parameter intervals and non-smooth transitions between them. In order to cope with this, a fast multivariate linear interpolation method was implemented inside the `PDFcreator::getRand(...)` C++ method. A full linear interpolation of the whole distribution function would be very time consuming, due to the fact that each interpolated function would need to be re-normalized after interpolation. Therefore the implemented method is based on a weighted random choice between the nearest- and second nearest histogram bin. This random decision is carried out for each input parameter i independently. The weights are given by p_i which describe the probability for choosing the function parameter obtained by nearest neighbor interpolation. $1 - p_i$ gives the probability to use the function parameter from the second closest bin (input parameter interval). p_i is computed by

$$p_i = p(x_i) = 1 - \frac{1}{2} \frac{x_i - c(x_i)}{e(x_i) - c(x_i)} \in [1, 0.5], \quad (5.3)$$

where x_i is the value for input parameter i (which is given), $c(x_i)$ is the center value of the bin to which this parameter value is assigned to (nearest neighbor interpolation) and $e(x_i)$ is the value of the closest bin edge.

Figure 5.1 illustrates $p(x)$ as a function of an input parameter value x for different parameter intervals.

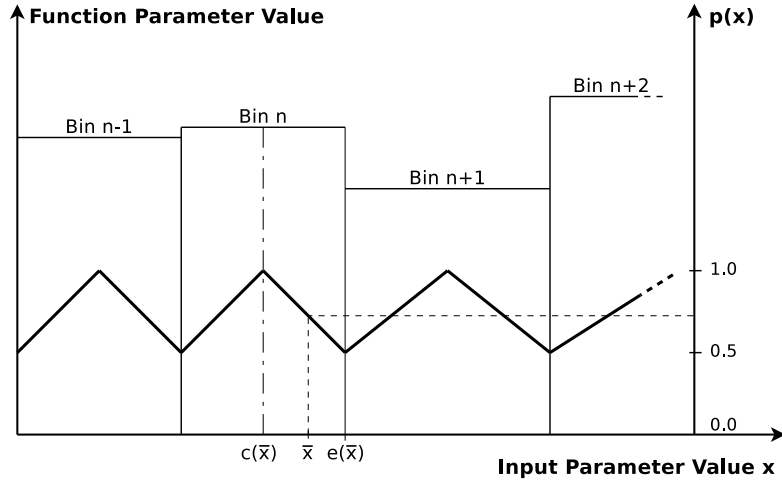


Figure 5.1: Illustration of $p(x)$ plotted against one input parameter value x . An example value $\bar{x}$ is given, together with corresponding closest bin center $c(\bar{x})$ and closest bin edge $e(\bar{x})$. In the center of each bin, $p(x) \rightarrow 1$. This corresponds to a probability close to one for choosing the nearest neighbor interpolation in finding the function parameters. On each bin edge. There is an equal chance of choosing the bin on the lower or upper side. The results obtained by this interpolation method show a very good agreement with the reference data, as shown in the validation section 6.1.1.

The **PDFcreator** utilizes the CLHEP random number generator [33] in combination with p_i to decide from which bin the function parameter will finally be taken from.

5.3 Integration into Fatras

A brief description of Fatras – a fast ATLAS detector simulation – was already given in section 3.2.2. Section 5.3.1 gives an overview of the simulation flow of Fatras, where all following sections give a deeper insight on how different simulation modules and C++ classes in Fatras are involved with the punch-through simulation. Fatras is fully embedded within the Athena framework (section 3.3).

5.3.1 Fatras Simulation Scheme

Figure 5.2 gives an overview of the application- and data flow of Fatras. The simulation consists of three main parts: the simulation input, the simulation kernel and the sub-detector simulations. All of these are described in this section in more detail:

Simulation Input: Fatras takes two possible types of input for the subsequent detector simulation:

1. **Event Generator (EvGen) files** which are provided by a particle physics event generator. This kind of input is standard for any physics analysis (see section 3.1).
2. **Single particles** which are entirely set-up inside the Fatras `jobOptions.py` file that is used to set up a certain simulation job. This input mode is mainly used for Fatras internal validation.

These input particles are added to the Fatras internal *particle stack*, where they will be treated by the Fatras simulation kernel for further processing.

Particle Stack and Simulation Kernel: The particle stack together with the Fatras simulation kernel are the core elements of the Fatras simulation. The stack describes a collection of particles (`ParticleStateCollection`) which are to be processed by the Fatras sub-detector simulations. The simulation kernel goes through all particles in this particle stack respectively and assigns them to the corresponding sub-detector simulation. This design allows a dynamic creation (and subsequent processing) of particles at any stage of the simulation. Also, this design is not limited to any particular particle flow direction. Particles (or their daughter particles) may cross sub-detector boundaries forth and back without any limitation.

The Fatras simulation for one event ends, when all particles in the stack are processed. Then the particle stack is emptied and to be filled with the particles for the next event.

Details about the implementation of the simulation kernel are given in section 5.3.2.

Sub-detector Simulation: Fatras uses different simulation models and -strategies for the three different ATLAS sub-detector parts:

Inner Detector: The Fatras Inner Detector (ID) simulation distinguishes between charged- and uncharged particles. For the former it simulates *detector hits* on sensitive

detector elements, decays and interactions with the detector material. At a later stage, the detector hits will be taken as input for the standard reconstruction software. Uncharged particles will be extrapolated through the ID without causing detector hits, but photon conversions, decays and material interactions are simulated for them as well as for charged particles.

Calorimeter: The calorimeter simulation in Fatras does not fully simulate the calorimeter response to particles. This is subject to a simulation called *FastCaloSim* (see [21]). However, in Fatras the calorimeter simulation is best described by the following three cases:

- If the particle is a muon, it is extrapolated through the calorimeter until it reaches the Muon Spectrometer. For this, the muons have to fulfill certain momentum cuts. Even though the sensitive calorimeter material is not simulated, material interactions of the muons with the calorimeter material are computed.
- If the particle type is set up to cause calorimeter punch-through, Fatras calls the calorimeter punch-through simulation. So far this is only implemented for pions.
- If the particle does not fulfill any of the above conditions, it is dropped and will not be further processed by Fatras.

Muon System: Currently, the Fatras Muon Spectrometer (MS) simulation is only validated for extrapolating muons through the MS. As the ID simulation, it also computes material interactions of the muons with the detector material and simulates detector hits on sensitive MS detector parts.

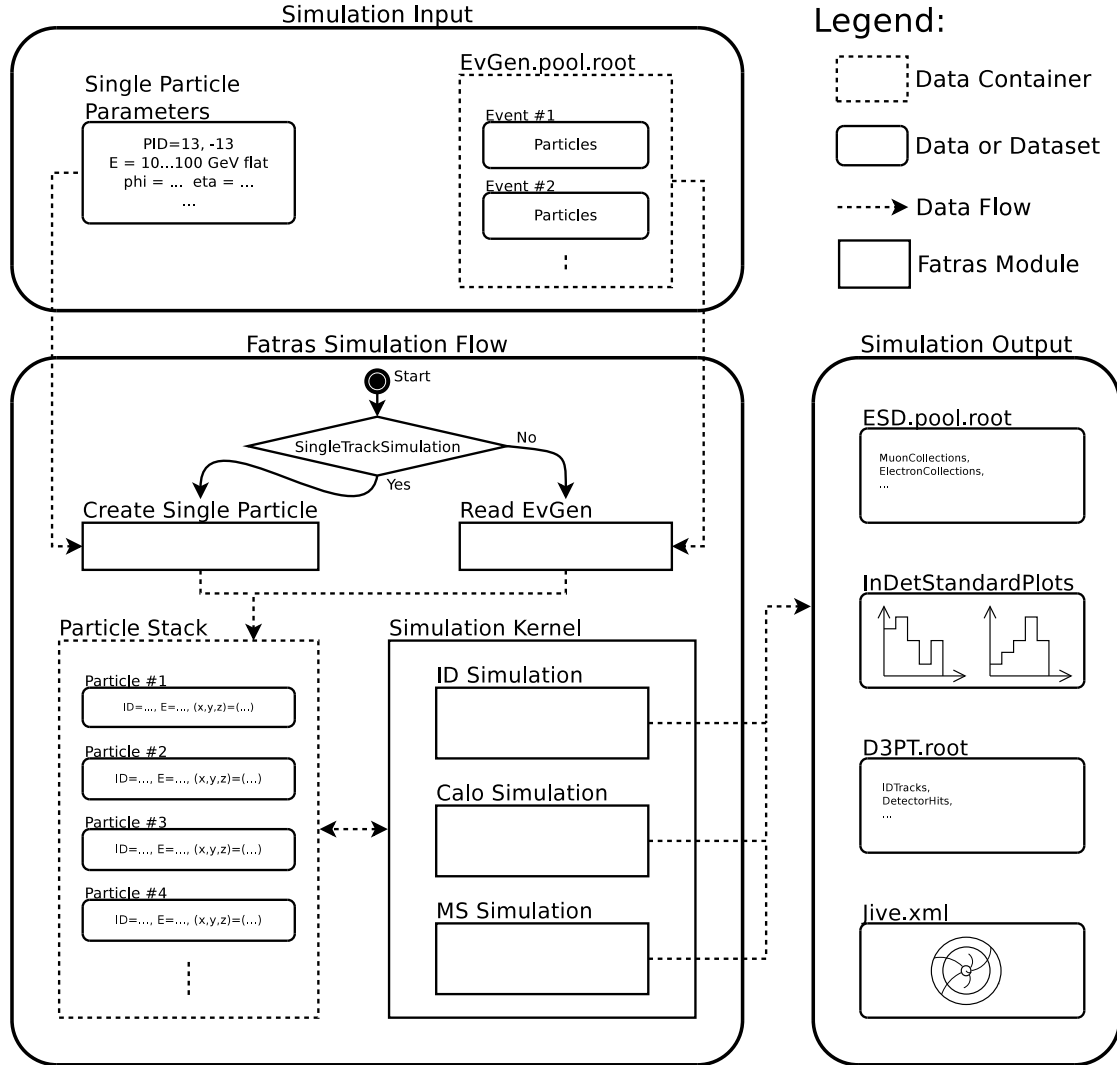


Figure 5.2: Simplified Fatras simulation scheme and data flow. Input particles are added to the simulation particle stack. The simulation kernel processes each particle in the stack and assigns it to the corresponding Fatras sub-detector simulation. Each sub-detector simulation will add newly created particles (e.g. due to hadronic interaction, pair production, ...) to the particle stack. The simulation ends when all particles in the stack are processed.

5.3.2 Fatras Simulation Kernel

Class Summary	
C++ Class Name	<code>Fatras::SimulationKernel</code>
Contained in package	<code>FatrasAlgs</code>

The main simulation loop in Fatras is run within the `Fatras::SimulationKernel`. It runs a stack simulation in which each single particle is represented by a `ParticleState` object. The kernel assigns these `ParticleStates` to the three different Fatras simulation parts: Inner Detector, Calorimeter and Muon System – depending on the respective value of `ParticleState::geometryLabel()`. For the Inner Detector and the Muon Spectrometer simulation respectively, a `ISimulationTool` is called, the calorimeter simulation is done via an `ITransportTool`. By default the Inner Detector and Muon Spectrometer simulations are done via `TrackSimulationTool` (section 5.3.3) instances. In case the calorimeter simulation is enabled, a `CaloSimulationTool` (section 5.3.6) instance is called for all particles inside the ATLAS calorimeter volume. In case the calorimeter simulation is disabled, a `TransportTool` (section 5.3.4) will take care of transporting muons through the calorimeter. The punch-through simulation is called from inside the `CaloSimulationTool`.

The Fatras sub-detector simulations (such as the calorimeter punch-through simulation) may add particles to the simulation stack (via the `ICollectionManager`) for a later treatment by the simulation kernel.

5.3.3 Track Simulation

Class Summary	
C++ Class Name	<code>Fatras::TrackSimulationTool</code>
Contained in package	<code>FatrasTools</code>
Compatible interface:	<code>Fatras::ISimulationTool</code>

`TrackSimulationTools` handle any particle transport for the Inner Detector or the Muon System. In cases where a `Trk::Track` object may potentially be created, it hands the particle over to a `Fatras::ITrackCreator` compatible tool, which by default is a `Fatras::TrackCreator` instance. This is the case when the currently processed particle is charged and may therefore create detector hits directly. If the particle shows no charge, a `Fatras::ITransportTool` is called, because no direct detector hits have to be taken into account for this case, therefore no `Trk::Track` objects will be created. By default, the `Fatras::TransportTool` is used for neutral particle transport.

In any case an `ExtrapolationTool` (section 5.3.5) does the extrapolations behind each particle transport between different (sensitive) detector elements. This ensures

that the entry layers are recorded correctly, which is required for further punch-through simulation and validation.

5.3.4 Particle Transport

Class Summary	
C++ Class Name	Fatras::TransportTools
Contained in package	FatrasTools
Compatible interface:	Fatras::ITransportTool

The transport tool is used within Fatras to do any extrapolation where no `Trk::Track` object is created. The two cases are: extrapolation of neutral particles in the Inner Detector (no detector hits are created by the particle itself) and the extrapolation of muons through the calorimeter.

In any case, the `TransportTool` calls a `Fatras::IExtrapolationTool` (default `ExtrapolationTool`) to do the extrapolation of the given particle. This is necessary in order to fill Calo- and MuonEntry collections correctly. Depending on the current setup, the extrapolator used for this may then produce daughter particles which will be fed into the main simulation stack of the `Fatras::SimulationKernel`.

5.3.5 Particle Extrapolation

Class Summary	
C++ Class Name	Fatras::ExtrapolationTool
Contained in package	FatrasTools
Compatible interface:	Fatras::IExtrapolationTool

The `Fatras::ExtrapolationTool` is basically a wrapper for any `Trk::IExtrapolator` compatible extrapolator. In addition, it automatically fills the Calo- and MuonEntry layer collections (see appendix B) when particles pass through the corresponding surfaces. *All* particle extrapolations in Fatras are done using this tool. It is particularly important for the punch-through simulation, because it assures that the position of the input particles are on the reference surface (CaloEntry layer).

A geometrical description of the Calo- and MuonEntryLayers is retrieved from a `Fatras::EntryLayerTool`, which offers corresponding `Trk::Surfaces` and `Trk::TrackingVolumes`.

5.3.6 Fatras Calorimeter Simulation

The `CaloSimulationTool` is called from the `SimulationKernel` for all particles which are assigned to the Fatras calorimeter simulation – which means they have a `ParticleState::geometryLabel()` equal to `Trk::Calo`. If the currently processed particle is a

Class Summary	
C++ Class Name	<code>Fatras::CaloSimulationTool</code>
Contained in package	<code>FatrasTools</code>
Compatible interface	<code>Fatras::ITransportTool</code>

muon, it will be handed over to a `ITransportTool` (default: `TransportTool`) to be transported through the ATLAS calorimeter. Any other particle type will be handed over to a `IParticleStateCollectionCreator` (default: `PunchThroughSimulator`, section 5.1) which handles the punch-through simulation and returns a `ParticleStateCollection`. This collection contains all particles which will penetrate the Muon Spectrometer after a calorimeter punch-through. The `CaloSimulationTool` adds all particles contained in the collection to the `SimulationKernel`'s particle stack for a later treatment in the Fatras Muon Spectrometer simulation.

5.4 Integration into AtlfastII

As previously described in section 3.2.3, AtlfastII uses full Geant4 detector simulation for the ATLAS inner detector and muon spectrometer, the calorimeter is simulated by FastCaloSim for all particle types but muons. Due to comparatively low CPU-time requirements for muon simulations, muons traversing the ATLAS calorimeter in AtlfastII will be simulated by Geant4. In order to compute punch-through or particle decay in flight effects, the AtlfastII implementation of the parameterized punch-through simulation uses the same Athena tools and C++ classes as previously described in section 5.1. In addition to that, some modifications are done in the AtlfastII `FastID-Killer::SteppingAction(...)` method. In AtlfastII simulations, this method treats all particles leaving the ATLAS inner detector. Its main purpose is to remove these particles from the Geant4 simulation stack, except for muons which are kept in the particle stack.

In order to run the fast punch-through simulation, a few lines of additional code are implemented which call the punch-through simulation for each particle to be removed from the simulation stack. For the currently implemented parameterized simulation, charged pions are the only particle type which will cause any punch-through effect – this check is carried out inside punch-through simulator tool (section 5.1). In case the punch-through simulation returns a set of particles, they will be added to the Geant4 simulation stack from inside the `SteppingAction(...)` method mentioned above.

Chapter 6

Results

Detailed studies were carried out in order to validate the fast ATLAS calorimeter punch-through simulation. In section 6.1, fast simulation results are compared to full Geant4 ATLAS detector simulation results.

Section 6.2 shows the computing time spend in the fast punch-through simulation for different event samples.

6.1 Comparison to Full Simulation

A validation study based on single particle events is described in section 6.1.1. High energy multi-jet events were used in order to validate the punch-through simulation on complex event signatures (section 6.1.2).

6.1.1 Single Particle Validation

The single particle validation is carried out within the Fatras simulation, since Fatras offers a simple way to set-up for single particle simulations. The initial particles are created with the same properties as the event sample used to create the parametrization (section 4.3.1):

- particle creation point is the interaction point $(x, y, z) = (0, 0, 0)$
- initial π^+ and π^- particles to equal number
- energy between $E = 150$ GeV and $E = 500$ GeV (uniformly distributed)
- pseudorapidity $|\eta| \leq 2.7$ (uniformly distributed)
- total number of simulated single particle events in Fatras: $\sim 3 \cdot 10^6$

Additionally the simulation of particle decays and particle interactions with ATLAS inner detector elements has been disabled. This is done in order to apply the same simulation settings as were present in the Geant4 sample used for the underlying parametrization. Therefore the same Geant4 event sample will be used as reference for the validation of the parameterized punch-through simulation (see section 4.3.1).

The Fatras simulations use ATLAS software release 17.0.2.6, detector description tag `ATLAS-GE0-16-00-00` and additional punch-through simulation patches in some software packages.

In a first view, basic punch-through properties, such as the punch-through probability, the number of punch-through particles and their kinematic properties are compared with Geant4 full simulations. This information is accessible through the `CaloEntry-` and `MuonEntryLayers` and does not need event digitization nor reconstruction.

The resulting punch-through probabilities in figure 6.1 show a good agreement with the reference sample. Figure 6.1 (b) shows that strong peaks of punch-through properties become *flattened out* in the fast simulation. These peaks are caused by the calorimeter material distribution which varies with η (figure 4.6). The flattening is due to the fact that the parametrized simulation is statistically limited by its underlying full simulated event sample. As described in section 4.4, this event sample is divided into discrete intervals within each of which the parametrization is carried out. As a consequence, strong variations within one parametrization interval are stored in an averaged form. Therefore the fast punch-through simulation will reproduce these averaged parameters.

In addition, punch-through properties in the fast simulation show the tendency to be *attracted* by the average value, i.e. reference values exceeding the average come out too low in the fast simulation and vice versa. This effect is due to the matrices used to compute particle type correlations. As mentioned in section 4.4.3, two matrices are used to describe each respective two particle type correlation. Each correlation matrix is averaged over approximately half of the total energy range ($0 \leq E \leq 200$ GeV and $200 \leq E \leq 500$ GeV) and over the whole pseudorapidity range ($-2.7 \leq \eta \leq +2.7$). This is the main cause for the differences seen in the punch-through probability- and composition plots in figure 6.1 (a,b) and figure 6.6 respectively.

The punch-through particle frequency is very well described up to high numbers of particles (figure 6.2(a)). Beyond ~ 25 punch-through particles, the parametrization starts to deviate from the Geant4 reference. However, this difference appears in regions with sub-permille probability, and fast simulations do not aim to simulate effects with this sensitivity. Similarly, the punch-through particle energies are described to good detail for the vast majority of events (figure 6.2(b)).

In figure 6.3, full and fast simulation results of the $\Delta\phi$ and $\Delta\theta$ distributions are shown. Finally, the particle momentum angles $\Delta\phi_p$ and $\Delta\theta_p$ are given in figure 6.4. In the context of a fast simulation, the $\Delta\phi_p$ distribution shows a rather significant difference between full and fast simulation. The cause of this could be the use of a local coordinate frame for $\Delta\phi_p$ and $\Delta\theta_p$ (see section 4.4.6). However, this deviation is not expected to have a significant impact on the overall performance of the punch-through simulation and its effect on reconstruction quantities (see section 6.1.2) .

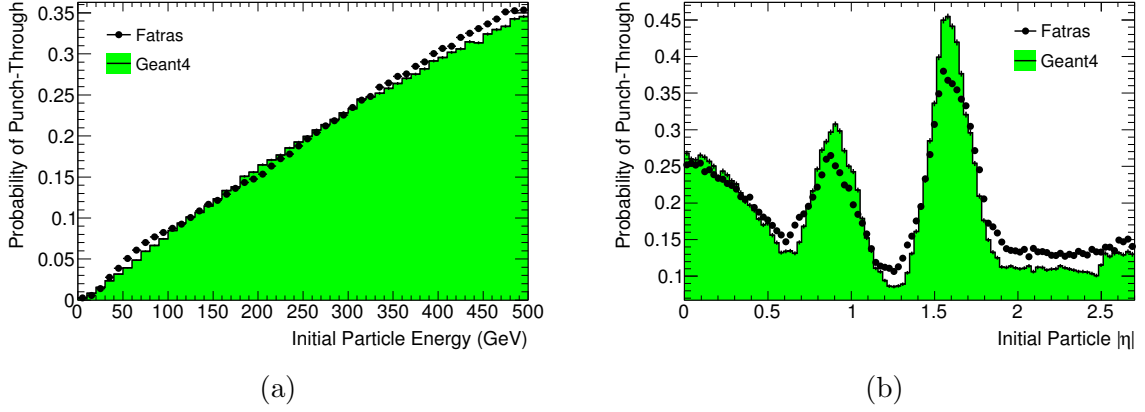


Figure 6.1: Punch-through probability, comparison of fast punch-through simulation (Fatras implementation) and full Geant4 simulation. Due to the underlying parametrization techniques, strong peaks become flattened out in the fast simulation. Simulation setup: single π^+ and π^- in equal numbers with energies up to 500 GeV, pseudorapidity range of $-2.7 \leq \eta \leq 2.7$. Both, Fatras and Geant4 simulations are done on a sample of $\sim 3 \cdot 10^6$ events.

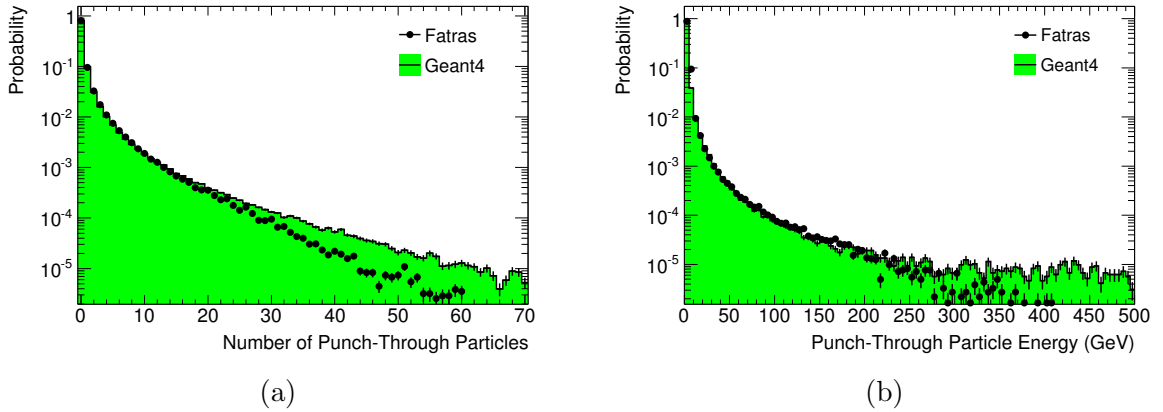


Figure 6.2: Comparison between fast punch-through simulation and full Geant4 simulation for (a) distribution for the number of punch-through particles and (b) punch-through particle energy distribution. The differences in the low probability regions $P < 10^{-4}$ are considered negligible for this fast simulation.

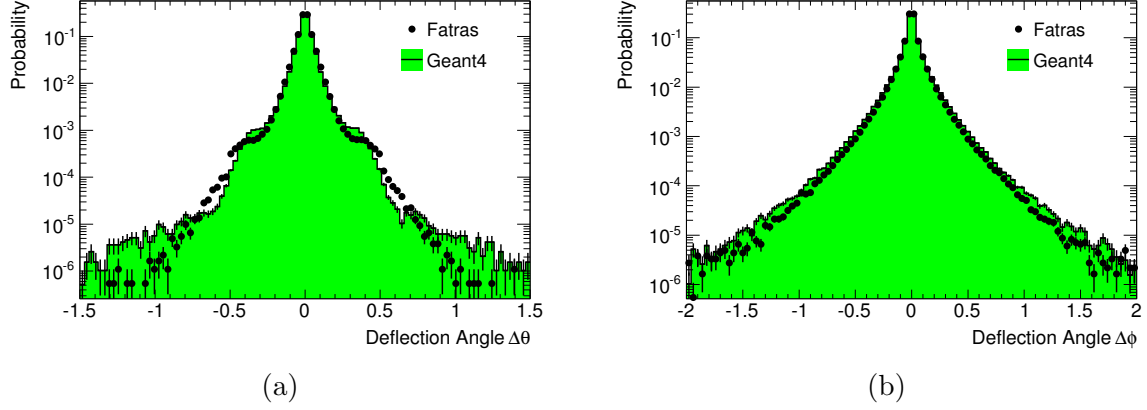


Figure 6.3: Punch-through particle deflection angles in θ and ϕ , compared for Geant4 detector simulation and fast parameterized punch-through simulation. Simulated events: single π^+ and π^- particles.

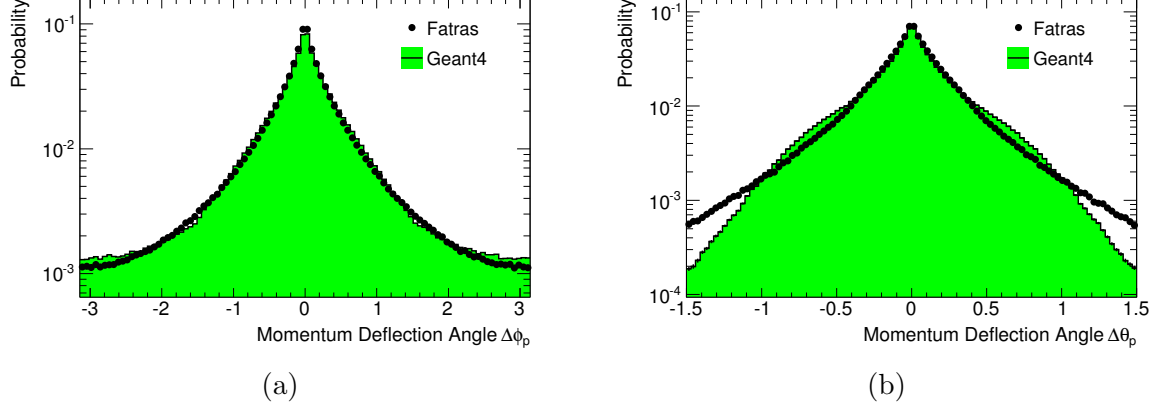


Figure 6.4: Punch-through particle momentum angles in local θ and ϕ coordinates, for full Geant4 and fast parameterized punch-through simulation. The fast punch-through simulation apparently does give a correct description of the $\Delta\phi_p$ distribution. For $\Delta\theta_p$, a good agreement is present up to probabilities of 10^{-2} . However, it is not expected that this deviation will have a significant impact on the overall results. Both distributions satisfy the requirements for a fast simulation approach. Simulated events: single π^+ and π^- primary particles.

Figure 6.5 shows that particle correlations are very well reproduced and full and fast simulation results are within 10% in regard to the respective mean values.

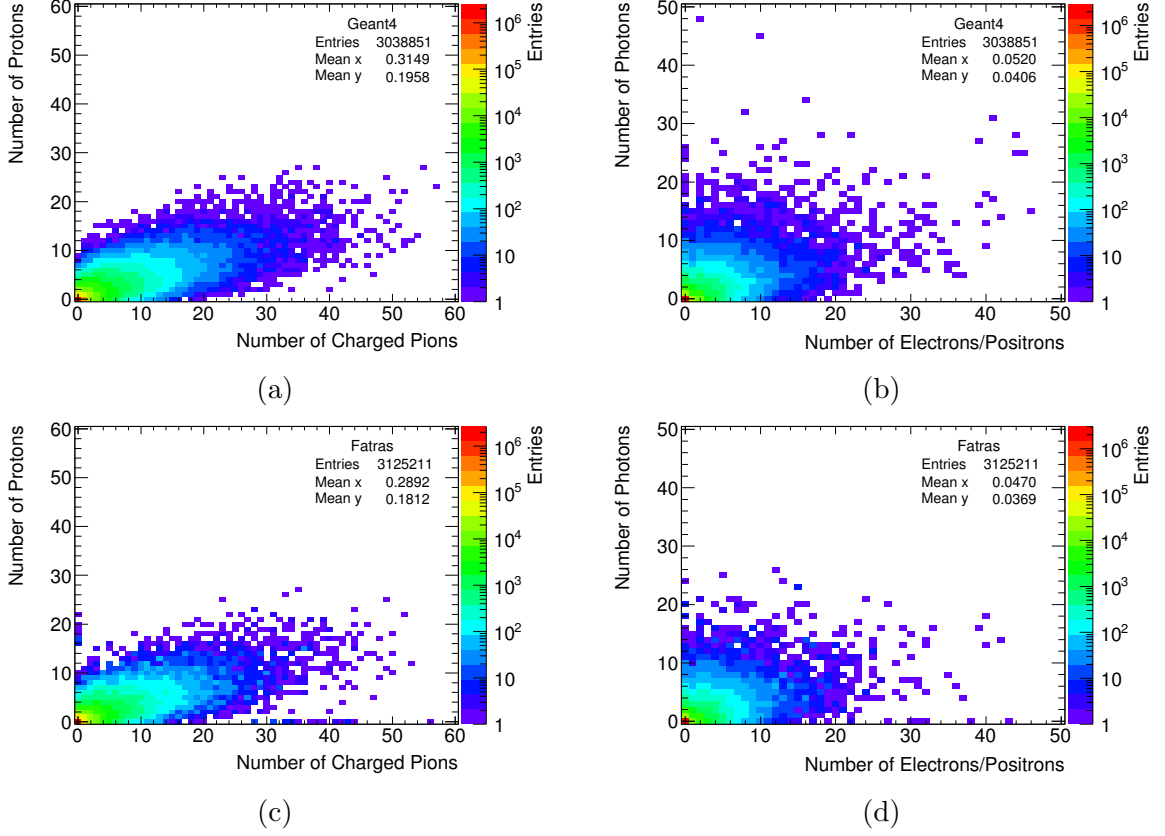


Figure 6.5: Correlations between punch-through particle type appearances, for single pion simulations: (a) protons and pions in Geant4 simulations, (b) electrons/positrons and photons in Geant4, (c) protons and pions in the parameterized simulation, (d) electrons/positrons in the parameterized simulation.

So far only fundamental punch-through properties were validated (taken from the Geant4 analysis section 4.3). As a next step, derived properties are studied: This is the punch-through particle composition in figure 6.6 and the average number of punch-through particles in figure 6.7, both plotted against initial particle energy and pseudo-rapidity η , respectively. As mentioned above, the underlying parametrization causes a flattening effect of strong peaks in either distribution.

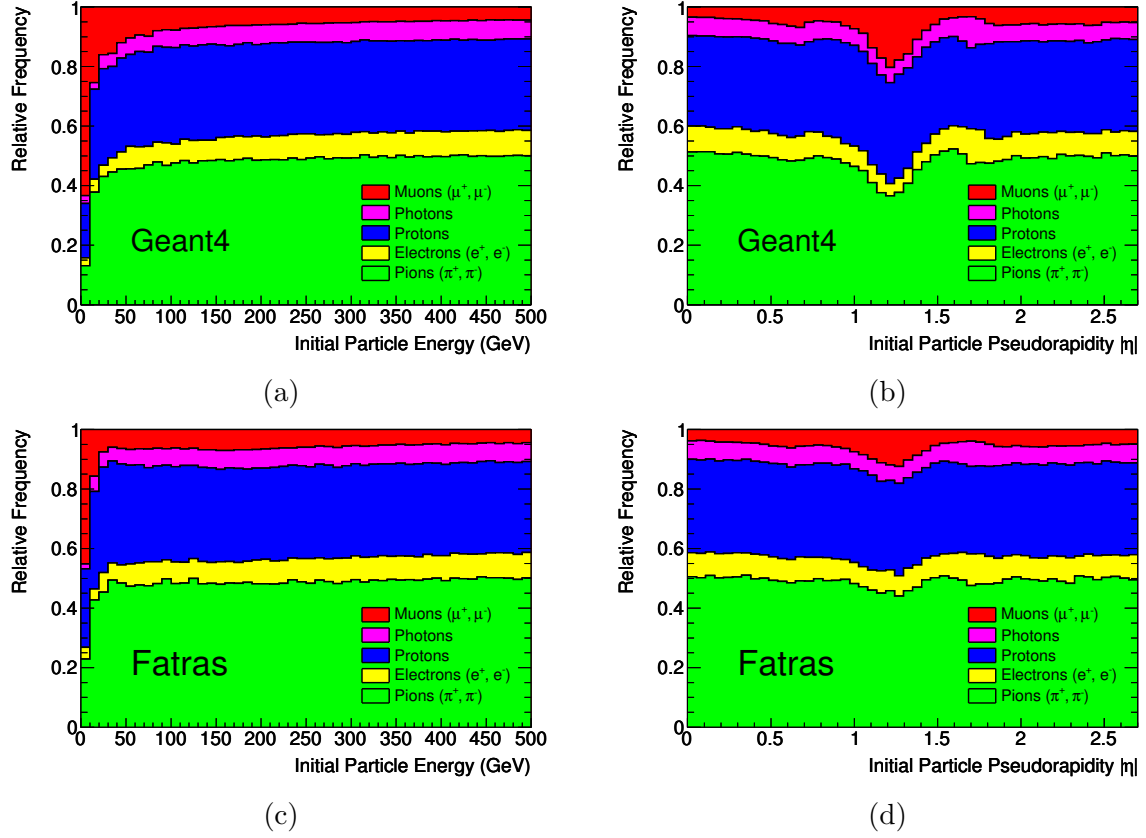


Figure 6.6: Punch-through particle composition compared between Geant4 and fast punch-through simulation (implemented in Fatras). Due to the underlying parametrization model, the fast simulation tends to flatten out strong peaks, compared to the Geant4 simulation.

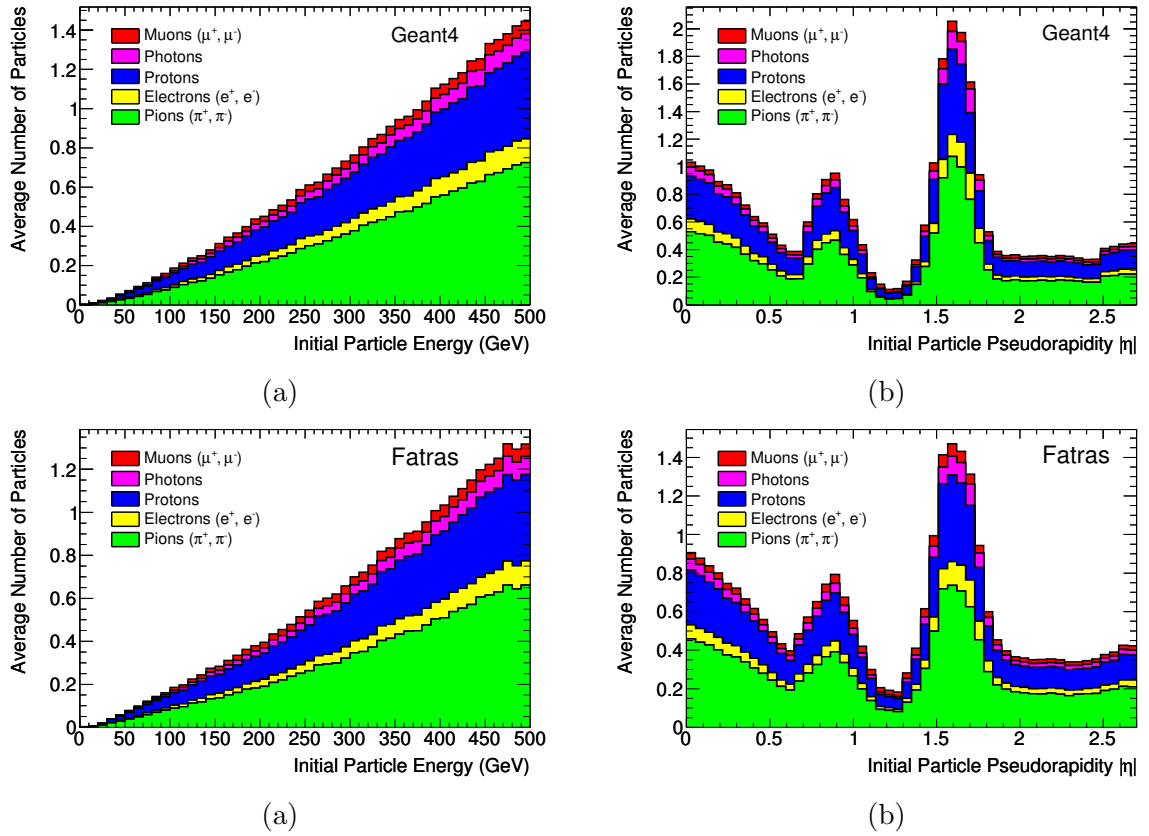


Figure 6.7: Average number of punch-through particles per single pion event. A flattening effect on strong peaks is visible in the fast punch-through simulation, compared to the full Geant4 simulation.

All single particle validation studies show a good agreement of the parametrized punch-through simulation with respect to the Geant4 simulated reference sample – especially in the context of a fast simulation approach. This sets the basis for further validation studies which are carried out in the following section.

6.1.2 High Energy Jet Events

In the previous section, the simulated particle properties generated by full and fast simulation have been validated. The following section concentrates on the validation of reconstructed quantities, such as particle tracks and identified muons. For this, high energy multi-jet events are simulated in full and fast detector simulations.

In order to do so, the parameterized punch-through simulation is embedded into the fast ATLAS detector simulation AtlfastII (see section 3.2.3 and section 5.4). AtlfastII conducts a full Geant4 simulation of the ATLAS inner detector and the muon spectrometer, respectively. Therefore with the punch-through simulation embedded into AtlfastII, calorimeter punch-through effects on reconstructed MS and ID/MS combined objects can be studied in detail, and compared to Geant4 full detector simulations. AtlfastII simulations were done with ATLAS software release 15.6.12.9, Geant4 simulations with release 16.6.3.5, digitization and reconstruction with release 16.6.3.2 (AtlfastII and Geant4).

To focus on effects caused by punch-through or decay in flight in the calorimeter, the reconstructed properties were only compared inside a cone around the respective jet axis. The cone is defined by global η and ϕ coordinates, the jet axis direction and an arbitrarily chosen distance parameter R :

$$\sqrt{(\eta - \eta_{jet})^2 + (\phi - \phi_{jet})^2} = \sqrt{\Delta\eta^2 + \Delta\phi^2} \leq R = 0.6 \quad (6.1)$$

In order to minimize the impact of *jet* reconstruction effects on the punch-through analysis, the corresponding η_{jet} and ϕ_{jet} angles are taken from the **AntiKt4TruthJets** collection. This collection is filled with jet objects reconstructed by the anti-kt algorithm [34] (distance parameter $R = 0.4$) using the *truth information* available in simulations.

The first validation concerns the number of measurements on MS track segments inside each jet cone. MS track segments are straight line connections of close MS detector hits within the same detector chamber [35]. Each MS standalone reconstruction algorithm uses a different set of MS track segments in order to fit particle trajectories through them. This validation uses the results of the Muonboy, Moore MS standalone reconstruction algorithms. An overview of the different techniques applied by the individual MS reconstruction algorithms is given in [35]. Figure 6.8 shows fast and full simulation results for the number measurements on MS track segments in a given jet cone. The AtlfastII punch-through simulation shows a significant improvement in describing the MS activity, compared to AFII without punch-through. However, Muonboy and MuGirl measurements occur with a higher frequency in the AtlfastII punch-through simulation, than in the reference Geant4 full simulation.

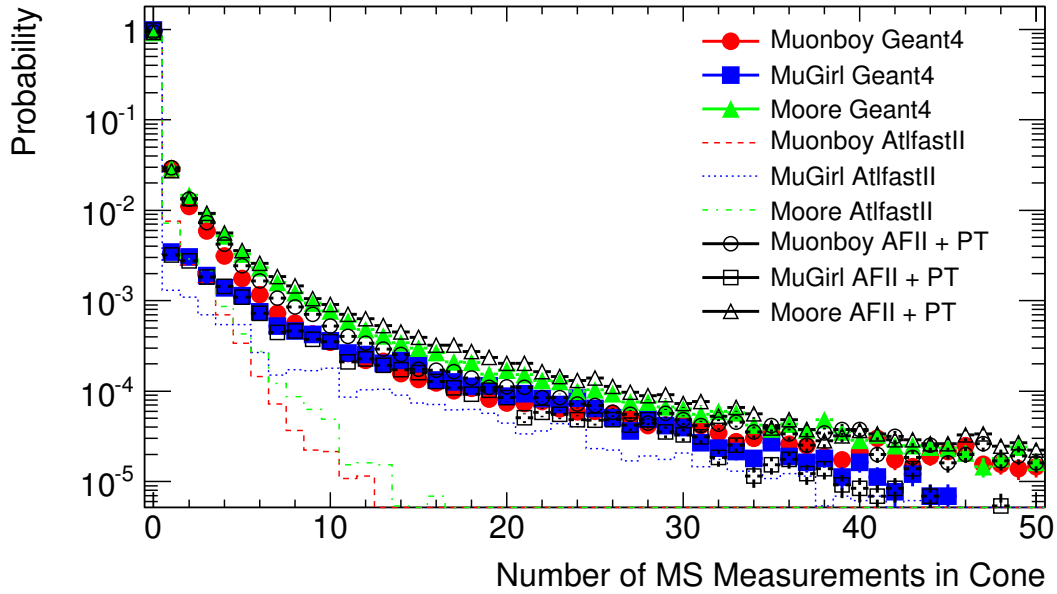


Figure 6.8: Number of measurements on MS track segments in jet cone. In AtlfastII simulations (no punch-through parametrization), all MS measurements are caused by muons leaving the ATLAS inner detector. In the AtlfastII punch-through implementation, a combination of punch-through effects, particle decay in flight and inner detector muons cause the recorded number of MS measurements.

Figure 6.9 shows a good improvement of AFII after implementing the punch-through simulation. However, Moore and Muonboy tracks appear slightly more frequent than in the Geant4 full simulation reference sample.

Combined reconstruction algorithms try to find corresponding ID and MS tracks which fit to the hypothesis of one primary muon causing both tracks. Three combined reconstruction algorithms were used in the following validation: MuID (using Moore standalone MS tracks), Staco (Muonboy standalone MS tracks) and MuTag [35]. Unlike MuID and Staco, MuTag does not require full standalone tracks in the MS. MuTag tries to find ID/MS combined particle tracks for *incomplete* MS standalone tracks. Figure 6.10 shows a significant improvement and a good agreement for the number of combined tracks, comparing fast punch-through simulation with full Geant4 detector simulation.

One additional reconstruction step is carried out in order to identify the muons out of the reconstructed combined tracks. To do so, calorimeter information and combined track information is taken into account. The simulation of muon particles is exactly the same for AtlfastII and full Geant4 detector simulation. Therefore the calorimeter signature of muons is the same for both simulations, whereas the calorimeter signature of any other ID particle is generated by FastCaloSim in case of AtlfastII. Figure 6.11 shows the number of muons from the MuGirlLowBetaCollection, MuIDMuonCollection

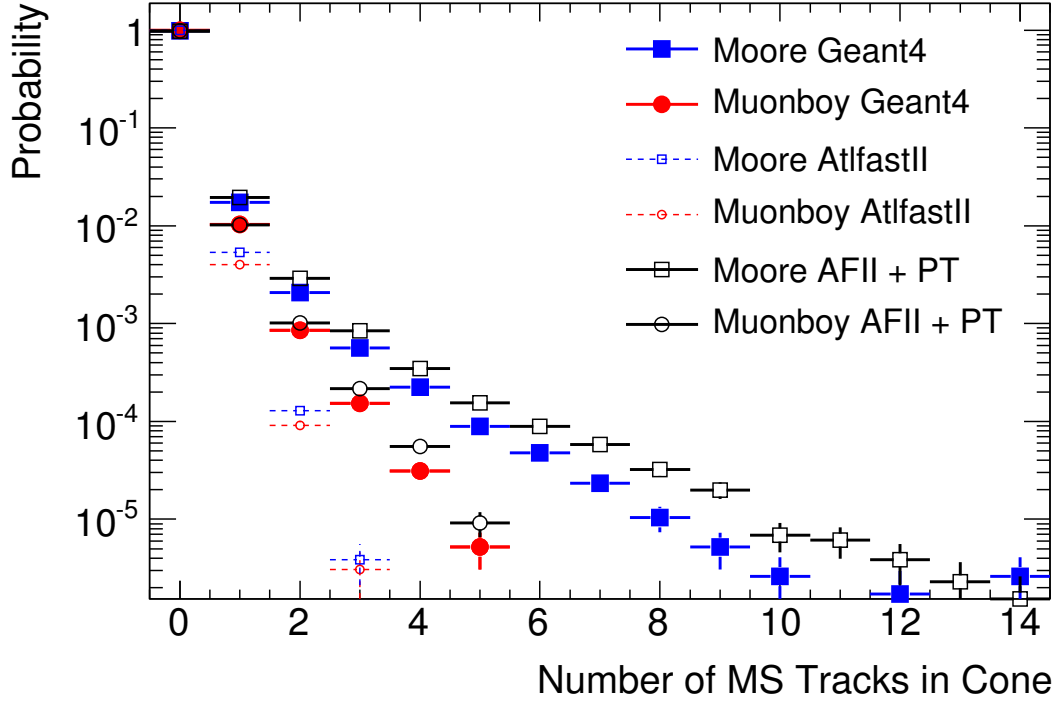


Figure 6.9: Number of standalone muon spectrometer tracks with perigee position inside jet cone. The improvement in AFII after implementing the punch-through simulation is due to its ability to simulate particle decay in flight and calorimeter punch-through effects.

and `StacoMuonCollection` collection (as present in ESD/AOD files). This result is particularly important due to the fact that these muon objects serve as input for any further physics analysis.

The results given throughout this section give a very promising picture of the fast punch-through simulation and show that this simulation is ready for further fast simulation usage on signal events. However, the underestimation of low- β MuGirl muons in the parameterized simulation (figure 6.11) seems to be related to the *slight* excess of the corresponding MuID and Staco muon rates, as well as the excess in MS standalone tracks (figure 6.9). A possible cause for this effect could be over-estimated punch-through particle energies in the parameterized simulation – especially at low punch-through energies. This would lead to more clear track signatures in the MS, in case a low energetic particle enters the MS. Therefore higher rates of MS standalone tracks and lower rates of low energetic muons (low- β) are to be expected. In a detailed look at punch-through particle energies in figure 6.2 (b), a slight excess in the second lowest histogram bin becomes apparent which supports this previous assumption.

Therefore the next *optimization* step to be taken is a detailed study of punch-through

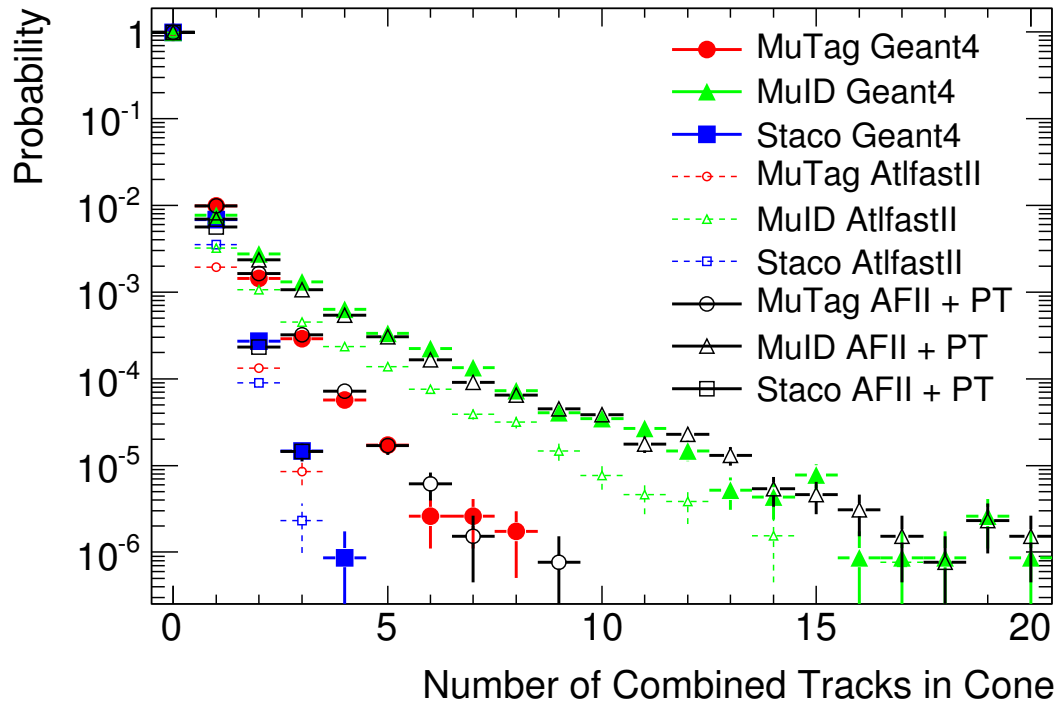


Figure 6.10: Number of ID/MS combined tracks with perigee momentum along jet cone. The three reconstruction algorithms, MuTag, MuID and Staco show compatible results between the AtIfastII punch-through implementation and the full Geant4 detector simulation.

particle energies – with a possible subsequent re-parametrization of the particle energy look-up table.

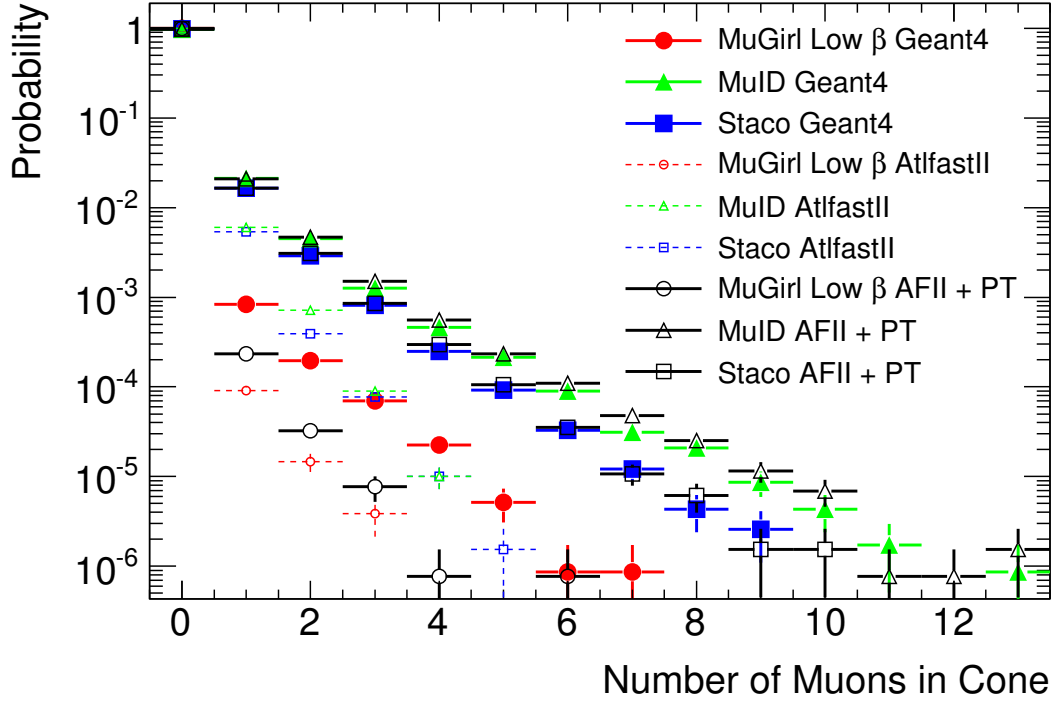


Figure 6.11: Number of reconstructed muons (analysis input objects) with momentuma along jet cone. Muon collections used: `MuGirlLowBetaCollection`, `MuidMuonCollection` and `StacoMuonCollection`. The punch-through simulation underestimates the rate of low- β muons reconstructed by MuGirl. A slight excess in MuID and Staco muon rates can be seen for the AtlfastII punch-through implementation, compared to the Geant4 reference sample.

6.2 CPU Performance

In order to measure the time consumption by the parameterized punch-through simulation, a performance study was conducted on different event samples.

Using the AtlfastII punch-through implementation, single charged pion event samples (with 1, 5 and 100 GeV respectively) and high energy multi-jet events were simulated. From the results given in table 6.1, we see that the time consumption of the fast punch-through simulation is *less* than one per mil of the overall AtlfastII simulation time.

Sample	Geant4 CPU time per event (sec)	AtlfastII			
		total	Punch-Through tools		
		CPU time per event (sec)	CPU time per event (μ s)	CPU time per call (μ s)	calls/event
single $\pi^{+/-}$, 1 GeV	0.668	0.357	52.7	6.89	7.65
single $\pi^{+/-}$, 5 GeV	3.45	0.383	86.1	5.29	16.3
single $\pi^{+/-}$, 100 GeV	61.7	0.805	405	7.74	52.4
high energy multi-jets	690	25.2	20800	5.79	3590

Table 6.1: CPU time of Geant4 and AtlfastII simulations for different event samples. The AtlfastII simulation includes the fast parameterized punch-through simulation. Data for single particle simulations are averaged over 10000 (AtlfastII) and 1000 (Geant4) simulated events, the high energy jet sample is averaged over 1000 (AtlfastII) and 20 (Geant4) events. The CPU used to process these simulations was a Intel(R) Xeon(R) L5640 @ 2.27GHz.

Chapter 7

Conclusion and Outlook

The parameterized simulation approach taken in this work proves its effectiveness and compatibility in almost every aspect. ATLAS calorimeter punch-through and particle decay in flight effects are parameterized based on a set of full Geant4 simulated single pion events (section 4.4). A fast simulation using this parametrization was integrated into the fast ATLAS detector simulation AtlfastII and Fatras (chapter 5). The results obtained from validation studies on single particle events describe the effects from the reference full Geant4 simulation to great detail (chapter 6). However, some deviations can be seen in the fast simulation due to statistical limitations in the reference sample used to compute the parametrization. The currently implemented description of particle correlations also results in some differences between fast and full simulation. Anyhow, at the current stage of this fast simulation, these differences are negligible and they do not show any significant impact on the results obtained in signal events.

The validation carried out on high energy multi-jet events gives a significant improvement of reconstructed muon particles due to the implemented punch-through simulation (section 6.1.2). The number of muons reconstructed by MuID and Staco combined reconstruction algorithms almost match with the Geant4 reference sample. This allows for the first time to include fake muon signatures or muons from decay in flight in the calorimeter in ATLAS fast simulation studies. Thus, it boosts the accuracy with which the MS response is described by fast simulations (figure 6.11). Only low- β MuGirl reconstructed muons are still underestimated in the punch-through simulation. A possible explanation is an insufficient description of low energetic punch-through particles. This is motivated by the observation that the fast simulation overestimates the appearance of standalone MS tracks. Therefore, further improvements of the punch-through simulation might concern a re-parametrization of punch-through particle energies.

Currently the simulation does not take care of correlations with any calorimeter variable. Therefore in one next step, obvious correlations such as the total energy deposited in the calorimeter might be studied, parameterized and implemented into the punch-through simulation.

Automated (fit) procedures were implemented, in order to fill the look-up table with the parameters extracted from a given reference sample. This setup also allows for a

fast re-parametrization of the punch-through simulation and offers an easy way to add new variables to the parametrization (e.g. energy deposited in calorimeter system).

The implemented simulation is independent of any particular ATLAS detector simulation. Hence, the punch-through simulation can easily be implemented into any new simulation framework, such as the integrated simulation framework (ISF) which is currently under development.

Since the comparison to Geant4 shows remarkable results, a comparison of the fast punch-through simulation with data will be the next important step to be taken.

Appendix A

The Look-up Table

The parameters for the Fatras punch-through simulation are stored in one single ROOT file, which is referred to as *look-up table*. The look-up table's structure allows for maximum flexibility regarding further improvements of the simulation – such as adding output parameters or changing the number of input parameters for parameterized distribution functions.

The ROOT file contains a `TDirectory` ROOT class for each respective output parameter. This `TDirectory` holds all relations which are necessary to compute the corresponding output parameter, depending on a set of input parameters. In the current implementation, these `TDirectories` are divided into two categories:

- parametrization input for the `PDFCreator` random number generator (see section 5.2)
- particle type correlations

Each of the two types is discussed in the following sections.

A.1 Input for the `Fatras::PDFcreator C++ Class`

This is the common method to store the parametrization for the fast punch-through simulation. The `TDirectory` content is closely related to the `Fatras::PDFcreator`'s function. The look-up table contains the following set of parametrizations for each respective punch-through particle type:

- number of punch-through particles (named: `NumExitPDG...`)
- punch-through particle energy (named: `ExitEnergyPDG...`)
- deflection angle $\Delta\theta$ (named: `ExitDeltaThetaPDG...`)
- deflection angle $\Delta\phi$ (named: `ExitDeltaPhiPDG...`)
- momentum angle $\Delta\theta_p$ (named: `MomDeltaThetaPDG...`)

- momentum angle $\Delta\phi_p$ (named: `MomDeltaPhiPDG...`)

The trailing full stops will be replaced by the the absolute value of the corresponding Monte Carlo particle number: e.g. electrons (11) and positrons (-11) together use `NumExitPDG11`, `ExitEnergyPDG11`, `ExitDeltaThetaPDG11`, ...

Section 4.4 describes how each individual parameter is determined and section 4.5 explains the runtime access on the corresponding parametrization.

Each directory contains a `TF1::function` instance, which stores the distribution function for the corresponding output parameter. Usually, this function has as a number of function parameters. The numerical values for these parameters are stored in `TH1`, `TH2` or `TH3` compatible ROOT histogram classes, also contained within the same directory. Their name follows the convention: `parameterx`, where 'x' is replaced by the corresponding function parameter number. By using ROOT histogram classes, one can conveniently store different function parameter values for different input parameter intervals. Therefore each input parameter corresponds to one axis in all `parameterx` ROOT histograms. The function parameter value is stored as the histogram content in the respective bin. In other words, each function parameter can be interpreted as a multi-dimensional scalar field, where the *coordinates* are represented by the input parameter values and the function parameter values are the histogram's content at the corresponding coordinates. All `parameterx` histograms within the same `TDirectory` (= of the same output parameter) are required to have the same dimension. Therefore all function parameters will have dependencies on the same set of input parameters.

Additionally `randmin` and `randmax` ROOT histograms are stored within each `TDirectory`. Their dependency on the input parameters is the same as for the `parameterx` histograms. The `Fatras::PDFcreator` ensures that the each determined parameter value lies within the respective range given by the `randmin` and `randmax` histograms.

In the current implementation of the parametrized punch-through simulation, the parametrizations only depend on up to two input parameters. Therefore only two-dimensional `parameterx`, `randmin` and `randmax` histograms are used.

A.2 Particle Type Correlations

The punch-through particle type correlations are stored in the `NumExitCorrelations` directory. The parameterized punch-through simulation is able to compute correlations between pairs of different particle types. Thus, each punch-through particle type can have up to one correlated particle type.

Two dimensional ROOT `TH2` histograms are used to store these pair correlations. These histograms represent, what is described as *correlation matrices* in section 4.4.3. The histograms have to be named according to the scheme: `x_PDGa__y_PDGb__lowE` or `x_PDGa__y_PDGb__highE`. *a* and *b* are replaced by the absolute value of the corresponding Monte Carlo particle number. The trailing `__lowE` or `__highE` string defines whether correlations for either low or high energetic incoming particles are described.

The naming convention motivates the fact that each particle type is represented by

one axis in the histograms. As mentioned in section 4.4.3, each correlation is stored twice, but with mirrored axes.

As an example, the correlation histograms between electrons (11) and photons (22) are named: `x_PDG22__y_PDG11__lowE`, `x_PDG22__y_PDG11__highE`, `x_PDG11__y_PDG22__lowE` and `x_PDG11__y_PDG22__highE`.

Appendix B

CaloEntry and MuonEntry

The CaloEntry and MuonEntry surfaces are widely used throughout this work. The CaloEntry layer surrounds the ATLAS inner detector, whereas the MuonEntry layer encloses the whole ATLAS calorimeter. Furthermore, both layers do not cross any sensitive detector parts. Therefore they are a perfect choice as interface surface between different sub-detector simulations, such as ID/Calo (CaloEntry) and Calo/MS (MuonEntry).

Both are used as reference surfaces for the parametrized punch-through simulation. The simulation's parametrization requires its input particles to be positioned on the CaloEntry layer surface, the resulting simulation output particles will be positioned on the MuonEntry layer.

Figure B.1 gives the dimensions of each layer.

Both, Fatras and the full Geant4 detector simulation create `TrackRecordCollections` of simulated *truth* particles crossing either of the surfaces.

Bibliography

- [1] Bruno Lenzi. Search for the standard model higgs boson decaying to four lepton (muon, electron) final states with the atlas experiment at the lh collider. Technical Report arXiv:0808.0162, Aug 2008. Comments: Poster presented at the Hadron Collider Physics Symposium (HCP2008), Galena, Illinois, USA, May 27-31, 2008; 5 pages, LaTeX, 11 eps figures.
- [2] P Jussel. Inclusive b-jet production in atlas. Technical Report ATL-PHYS-PROC-2011-099, CERN, Geneva, Aug 2011.
- [3] The ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. Aug 2008.
- [4] Jean-Luc Caron. Layout of the lep tunnel including future lh infrastructures. AC Collection. Legacy of AC. Pictures from 1992 to 2002., Feb 1997.
- [5] A Salzburger. Track Simulation and Reconstruction in the ATLAS experiment. Mar 2008.
- [6] C.Y. Wong. *Introduction to high-energy heavy-ion collisions*. World Scientific, 1994.
- [7] 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.
- [8] 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.
- [9] *ATLAS computing: Technical Design Report*. Technical Design Report ATLAS. CERN, Geneva, 2005. revised version submitted on 2005-06-20 16:33:46.
- [10] The ATLAS Collaboration. ATLAS TWiki. <https://twiki.cern.ch/twiki/bin/view/Atlas/WebHome>, July 2011.
- [11] 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.

- [12] T Sjöstrand, 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.
- [13] 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.
- [14] The ATLAS Collaboration. Charged-particle multiplicities in pp interactions at $\sqrt{s} = 900$ gev measured with the atlas detector at the lh. *Physics Letters B*, 688(1):21 – 42, 2010.
- [15] 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.
- [16] T Yamanaka. The atlas calorimeter simulation fastcalosim. Technical Report ATL-SOFT-PROC-2011-021, CERN, Geneva, Jan 2011.
- [17] 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.
- [18] 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.
- [19] The Atlantis Team. Official Atlantis Website. <http://www.hep.ucl.ac.uk/atlas/atlantis/>, July 2011.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.

-
- [24] Richard Wigmans. *Calorimetry – Energy Measurement in Particle Physics*. International Series of Monographs on Physics. Oxford University Press, Oxford, 2000. reprinted 2008.
 - [25] L Garren, I.G. Knowles, S Navas, P Richardson, T Sjöstrand, and T Trippe. Monte carlo particle numbering scheme. Technical report, Jun 2006.
 - [26] Nir Amram and Erez Etzion. *Hough Transform Track Reconstruction in the Cathode Strip Chambers in ATLAS*. *oai:cds.cern.ch:1118033*. PhD thesis, Tel Aviv, Tel Aviv University, Tel Aviv, 2008. Presented on 19 Mar 2008.
 - [27] Daniela Salvatore and Giancarlo Susinno. *Intensive irradiation studies, monitoring and commissioning data analysis on the ATLAS MDT chambers*. *oai:cds.cern.ch:1125845*. PhD thesis, Univ. Calabria, Cosenza, Cosenza, 2007. Presented on 10 Dec 2007.
 - [28] Nikulin M.S. Greenwood P.E. *A Guide to Chi-Squared Testing*. John Wiley & Sons, Chichester, 1996.
 - [29] The ROOT Team. Official ROOT Website. <http://root.cern.ch/>, May 2011.
 - [30] Dimitri van Heesch. Official Doxygen. <http://www.doxygen.org/>, August 2011.
 - [31] M Steyvers. Computational Statistics with Matlab. <http://www.scribd.com/doc/51315212/>, March 2010.
 - [32] Christian P Robert and George Casella. *Monte Carlo statistical methods; 2nd ed.* Springer texts in statistics. Springer, Berlin, 2005.
 - [33] The CLHEP Team. CLHEP - A Class Library for High Energy Physics – Official Website. <http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep/>, July 2011.
 - [34] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. The anti- k_t jet clustering algorithm. *J. High Energy Phys.*, 04(arXiv:0802.1189. LPHE-07-03):063. 12 p, Feb 2008.
 - [35] B Resende. Muon identification algorithms in atlas. Technical Report ATL-PHYS-PROC-2009-113, CERN, Geneva, Sep 2009.