Albert-Ludwigs-Universität
Freiburg

Fakultät für Mathematik und Physik

Lehrstuhl für experimentelle Teilchenphysik
Prof. Dr. Karl Jakobs

Validation and Extension of the Fast Atlas Track Simulation (Fatras) in Preparation for Atlas Inner Tracker Upgrade Simulation Studies

Author:
Jörg Mechnich

Supervisors:
Prof. Dr. Karl Jakobs
Dr. Ulrich Parzefall

April 10, 2008

Contents

Introduction

A long journey is finally coming to an end: 2008 is intended to be the year when the first proton-proton collisions are to be produced by the Large Hadron Collider (LHC) at CERN. Particle physicists from all over the world will be able to start their searches for the Higgs boson and new physics phenomena in a so far unexplored energy domain at the four main experiments at the LHC, namely ALICE, ATLAS, LHCb and CMS.

ATLAS, short for ***A** **T**oroidal **L**HC **A**pparatu**S***, is one of the two general-purpose detectors at the collider and just as large as the collaboration responsible for its creation: over 2000 scientists from 35 countries are currently working on hardware issues or preparing physics analyses.

Although physics runs at the LHC have not even started yet, plans for upgrading the machine and the detectors are already in the making. Around 2015, the central parts of the general purpose detectors ATLAS and CMS will be heavily damaged by irradiation and a replacement will be necessary for the continuation of data taking. This opportunity is seized in order to push the collider parameters to the possible maximum with respect to the dimensions of the existing tunnel. An increase to the ten-fold luminosity of the LHC is the most probable scenario for the upgraded accelerator, referred to as *Super-LHC* (SLHC).

The composition of a new ATLAS Inner Detector has to be different to the current one. The Transition Radiation Tracker (TRT) will no longer be able to perform its task due to a high occupancy. The replacement detector will be consisting of silicon-only sub-detectors. Pixel detectors close to the beamline will deliver high-precision hit information for vertex reconstruction and silicon strip detectors are to be employed in the outer regions in order to reach track parameter resolutions comparable to the current Inner Detector.

Such large-scale detector development projects require a massive simulation effort to understand the key points in physics and detector design and optimise the layout within the constraints posed by the changes to the collider parameters. As the research and development for the current Inner Detector took about fifteen years, it is mandatory to come to well founded decisions soon, in order to exploit the physics potential of the SLHC with an upgraded ATLAS detector well adapted to the SLHC environment.

This thesis describes current upgrade plans and extensions to the ATLAS software framework ATHENA for upgrade simulation studies. A large part of the underlying work is reflected in the validation progress of the ***F**ast **A**TLAS **Tra**ck **S**imulation* (Fatras) which was necessary to eventually enable physics studies. Integral parts of the software were extended and new packages added in order to obtain the required flexibility demanded for the integration of new detector geometries.

The first chapter starts with a brief overview of the LHC and the ATLAS experiment, describing the main parameters of the accelerator and expected physics results of the next few years. The ATLAS Inner Detector is characterised in detail.

Chapter 2 continues with the description of different SLHC upgrade scenarios and the consequences for the ATLAS experiment. New proposals for the necessary changes in the detector layout are discussed.

The third chapter introduces the ATLAS software framework *Athena*. Existing simulation programmes for the ATLAS detector are compared.

In chapter 4, the Fast ATLAS Track Simulation Fatras is presented. The different submodules of the software are discussed in detail in terms of structure and performance.

New components for Fatras and ATHENA created within the framework of this diploma thesis in preparation for upgrade simulation studies are described in chapter 5.

Chapter 6 elaborates upon several studies that have been undertaken in order to validate the fast simulation framework Fatras.

The final chapter contains a comparison of a subset of Inner Detector upgrade geometry layouts. For one selected layout, further performance criteria are studied.

1 The ATLAS Experiment at the Large Hadron Collider (LHC)

1.1 The Large Hadron Collider (LHC)

The Large Hadron Collider (LHC) [1] is a circular accelerator residing in the existing LEP tunnel at CERN in Geneva, Switzerland. With a proton-proton centre-of-mass energy $\sqrt{s}$ of 14 TeV, it pushes the high-energy frontier beyond the Tevatron at Fermilab and LEP.

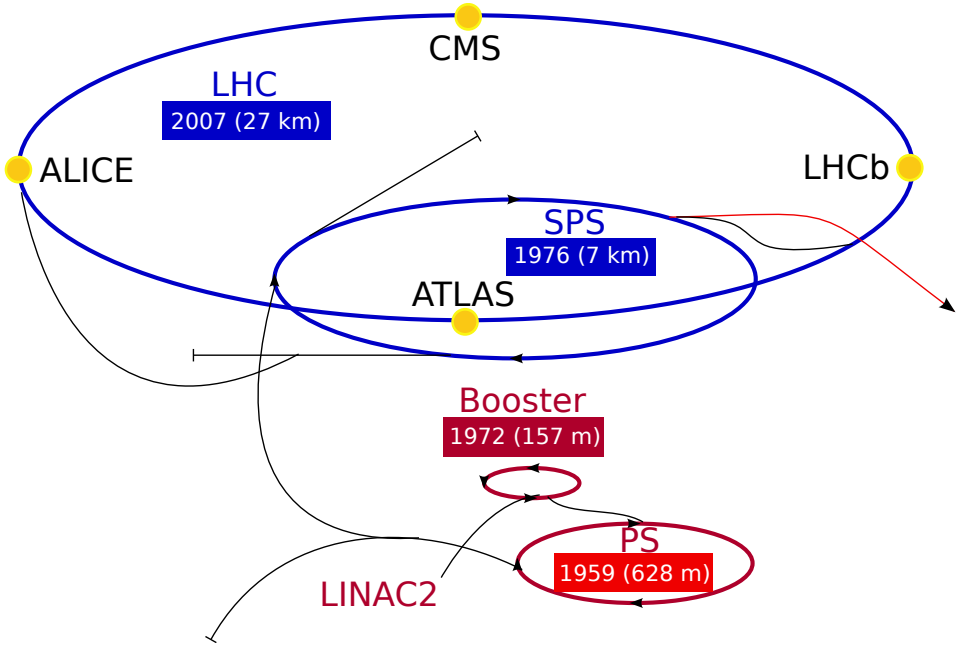


Figure 1.1: The chain of accelerators at CERN which are used in the process of reaching the final centre-of-mass energy of 14 TeV at the LHC [2].

The acceleration of protons starts in a dedicated linear accelerator (LINAC2), which accelerates bunches of 10^{11} protons to an energy of 50 MeV. These bunches are then transferred to the PS Booster (PSB), where the energy is raised to 1.4 GeV. The energy is further increased to 26 GeV by the Proton Synchrotron (PS). The protons are then injected into the Super Proton Synchrotron (SPS) where they are accelerated to 450 GeV. Finally, the SPS injects the protons clockwise and counter-clockwise into the LHC ring, where they reach their final energy of 7 TeV. The cascade of accelerators is displayed in Figure 1.1.

More than 1200 dipole magnets are installed along the LHC beam line to keep the protons on track in the ring. The superconducting dipole magnets provide a magnetic field of up to 9 T. The main parameters of the LHC accelerator are given in Table 1.1.

Parameter	**Value**
Circumference	26659 m
Beam energy	7 TeV
Injection energy	0.45 TeV
Dipole field at 450 GeV	0.535 T
Dipole field at 7 TeV	8.33 T
Helium temperature	1.9 K
Coil aperture	56 mm
Distance between apertures	194 mm
Luminosity	$10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$
Luminosity lifetime	10 h
Bunch spacing	25 ns
Protons per bunch	10^{11}
Bunches per beam	2808

Table 1.1: LHC operation parameters

Like its centre-of-mass energy, the luminosity of the LHC is also unprecedented for a proton collider. The luminosity is defined as the number of protons traversing a plane of unit area per unit time. The higher the luminosity, the more proton-proton interactions will occur. At the LHC's design luminosity of $10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, on average about 25 collisions will take place per bunch crossing, with a bunch spacing of 25 ns. Thus, the number of proton-proton interactions per second will be around 10^9.

Four detectors will be operating at the points where the beams collide (see Fig. 1.2): ALICE [3], ATLAS [4], CMS [5] and LHCb [6].

ATLAS and CMS are general purpose detectors, i.e. they are designed to cover a wide range of physics. Their primary task will be to probe the nature of electroweak symmetry breaking for which presumably the Higgs boson is responsible. In the first few years of data taking, $10\,\mathrm{fb}^{-1}$ of integrated luminosity are expected to be recorded by each experiment, so the Standard Model Higgs boson could be discovered with a significance of $5\,\sigma$ over the full mass range of interest after the combination of results [7]. The two experiments will also explore physics beyond the Standard Model, like supersymmetry, extra dimensions, and even mini black holes. The discovery potential for supersymmetry is essential in the first months of data taking because only $100\,\mathrm{pb}^{-1}$ of integrated luminosity are sufficient to discover squarks or gluinos with masses below about 1.3 TeV. The sensitivity increases to 1.7 TeV for $1\,\mathrm{fb}^{-1}$ and to about 2.2 TeV for $10\,\mathrm{fb}^{-1}$ [7]. The high luminosity and cross sections enable further high precision tests of QCD, electroweak interactions and flavour physics. The top quark will be produced at a rate of a few tens of Hz, providing the opportunity to test its couplings and spin.

The LHCb experiment is dedicated to the study of CP-violation in the B-system, it is therefore optimized for the detection of B-hadrons. LHCb uses a lower luminosity of about $10^{32}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, by defocusing the proton beams near the interaction point. This is needed because the production and decay vertices of the B-mesons are difficult to reconstruct if there is more than one interaction per bunch crossing.

The ALICE experiment focuses on the study of the quark-gluon plasma, by measuring the particles that are produced in heavy ion collisions. The quark-gluon plasma is a phase of QCD where quarks and gluons are not confined in hadrons anymore, but are able to move freely within the plasma. It is expected that the extreme energy densities in heavy ion collisions are sufficient to create this state of matter for fractions of a second.

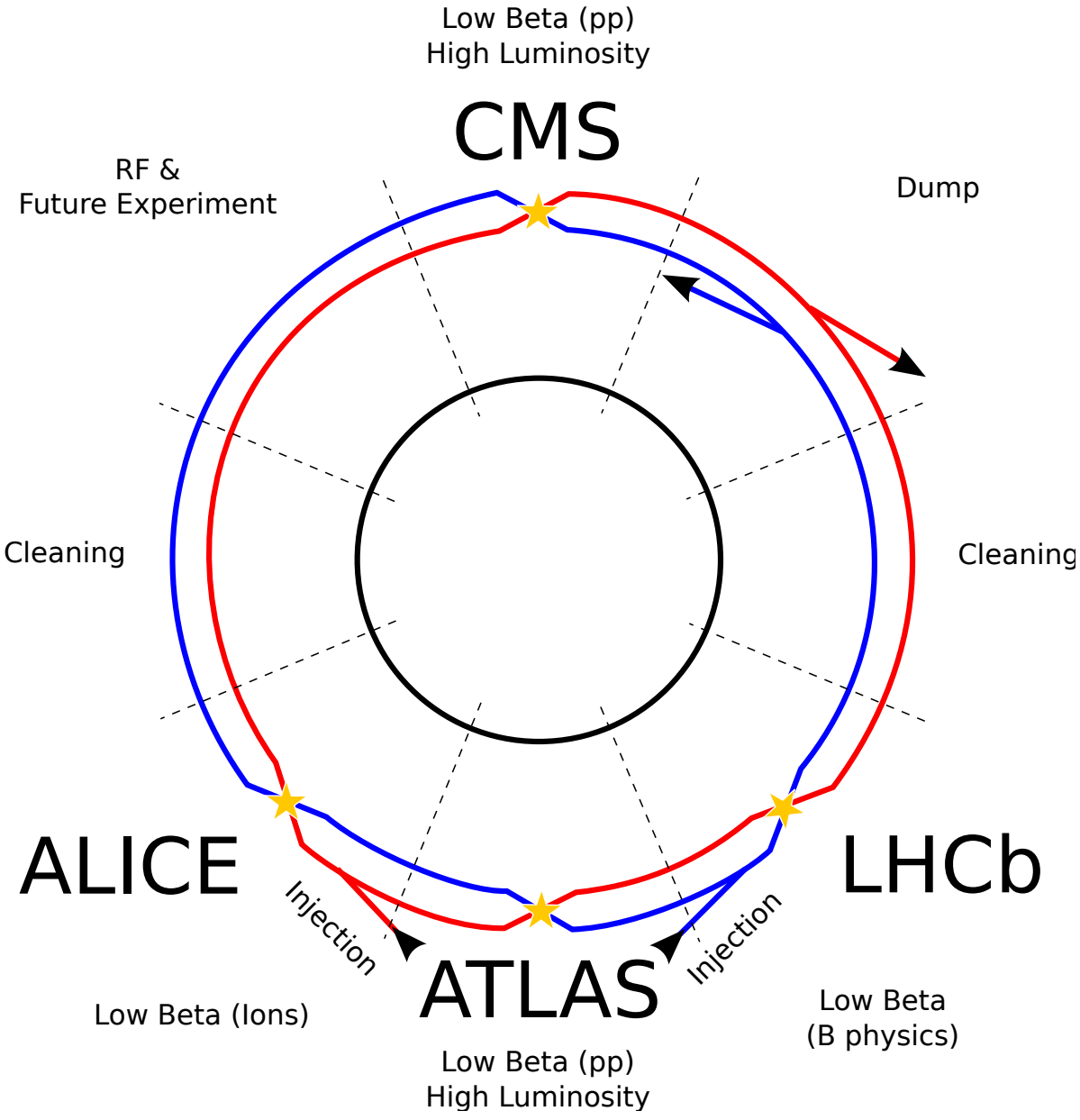


Figure 1.2: Schematic of the experiments and other collider features of the LHC [8].

1.2 The ATLAS Detector

The ATLAS (short for "**A** **T**oroidal **L**HC **A**pparatu**S**") detector (Figure 1.3) is approaching completion at "Point 1", one of the four interaction points of the LHC, just across the main entrance of CERN. The ATLAS detector concept was first presented in a Letter of Intent (LoI) [9] in 1992, followed by a proposal [4] for operation in 1994 and was finally approved for construction in 1995.

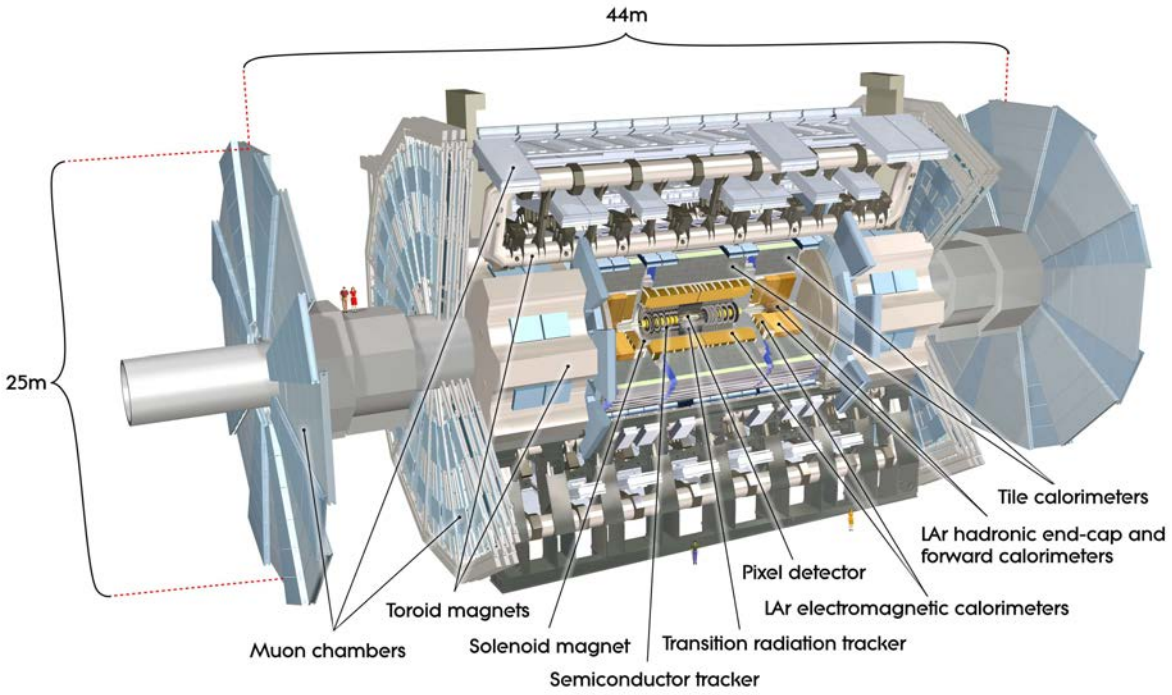


Figure 1.3: Cut-away view of the ATLAS detector [10].

With a weight of 7000 tons, a diameter of 25 m and a length of 44 m, the detector surpasses the dimensions of previous collider experiments by an order of magnitude. This is a direct result of the high beam energy of the LHC. The large size reflects the lever arm needed to reach the necessary momentum resolutions for muons and to sustain all hadrons inside the calorimeters for the measurement of their energy deposition.

ATLAS is shaped like a cylinder consisting of a *barrel* part and two *endcaps*. This is also reflected in every subsystem of the detector. For the description of geometric features it is most useful to use a cylindrical coordinate system with the z-axis collinear to the beamline. The x-y-plane (or also R-plane) is then perpendicular to the main symmetry axis. Angles are usually denoted as θ and ϕ for polar and azimuth, respectively. At hadron-collider experiments, a quantity called *pseudo-rapidity* η is often used instead of the azimuthal angle. It is defined as

$$\eta = -\log(\tan(\frac{\theta}{2})) \tag{1.1}$$

and converges to the relativistic *rapidity* in the massless limit. A convenient property of this quantity is a constant track multiplicity per unit.

The detector has a onion-like structure, starting from the innermost part, the aluminium beampipe. The first sensitive part of the detector is the *Inner Detector* or *Tracker*, responsible for the momentum measurement of charged particle tracks and electron identification. It will be described in detail in the following section 1.3.

The ATLAS Inner Detector is enclosed by the electromagnetic calorimeter covering a pseudo-rapidity range of $|\eta| < 3.2$. It is a lead-liquid argon (LAr) detector with accordion-shaped kapton electrodes and lead absorber plates over its full coverage. The accordion geometry provides complete ϕ symmetry without azimuthal cracks.

The next layer is formed by the hadronic tile calorimeter in the barrel region and the LAr endcap calorimeters.

The muon spectrometer is the outermost part of ATLAS, extending from a radius of 4.25 m around the calorimeters out to the full radius of the detector. Its tremendous size is required to accurately identify and measure the momentum of muons, which penetrate other elements of the detector. The muon trajectories are bend by a toroidal magnetic field of 4 T, in this case produced by eight very large air-core superconducting barrel loops and two endcaps.

1.3 The Inner Detector of ATLAS

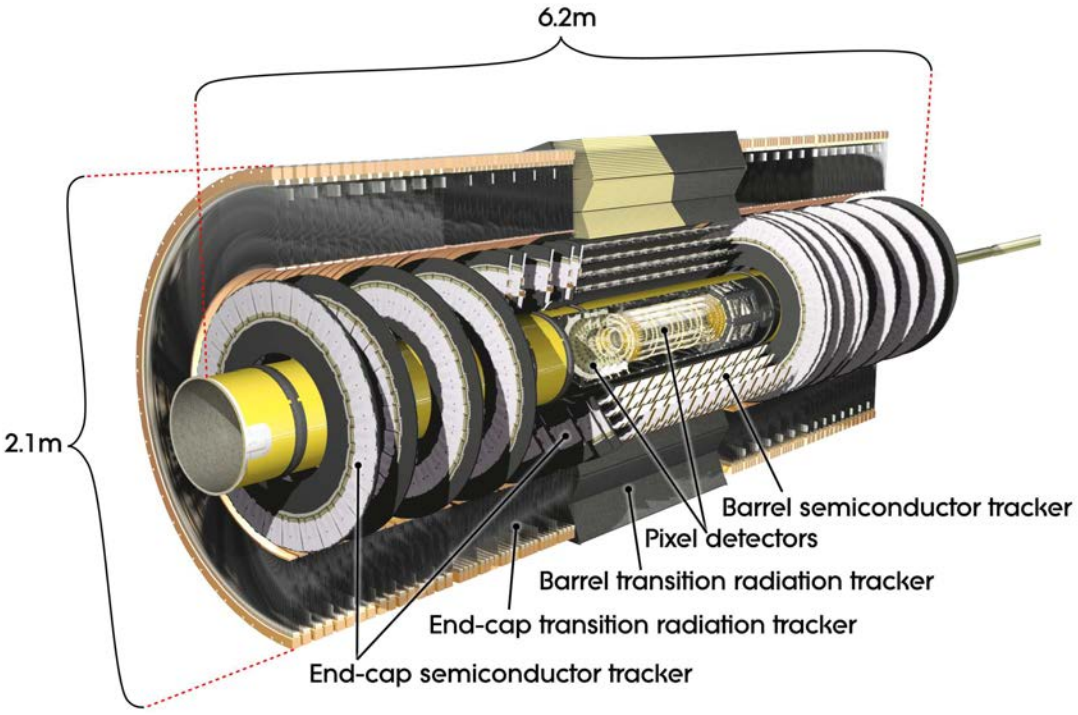


Figure 1.4: Cut-away view of the ATLAS Inner Detector [10].

The ATLAS Inner Detector (ID) [11] (Figure 1.4) is designed to provide hermetic and robust

pattern recognition, both primary and secondary vertex measurements, as well as excellent momentum resolution for charged tracks above a given p_T threshold (nominally 0.5 GeV) and within the pseudorapidity range of $|\eta| < 2.5$. This performance, which is required even at the highest LHC luminosities expected, is at the limit of existing technology. In the following, Inner Detector (ID) and (Inner) Tracker are used synonymously.

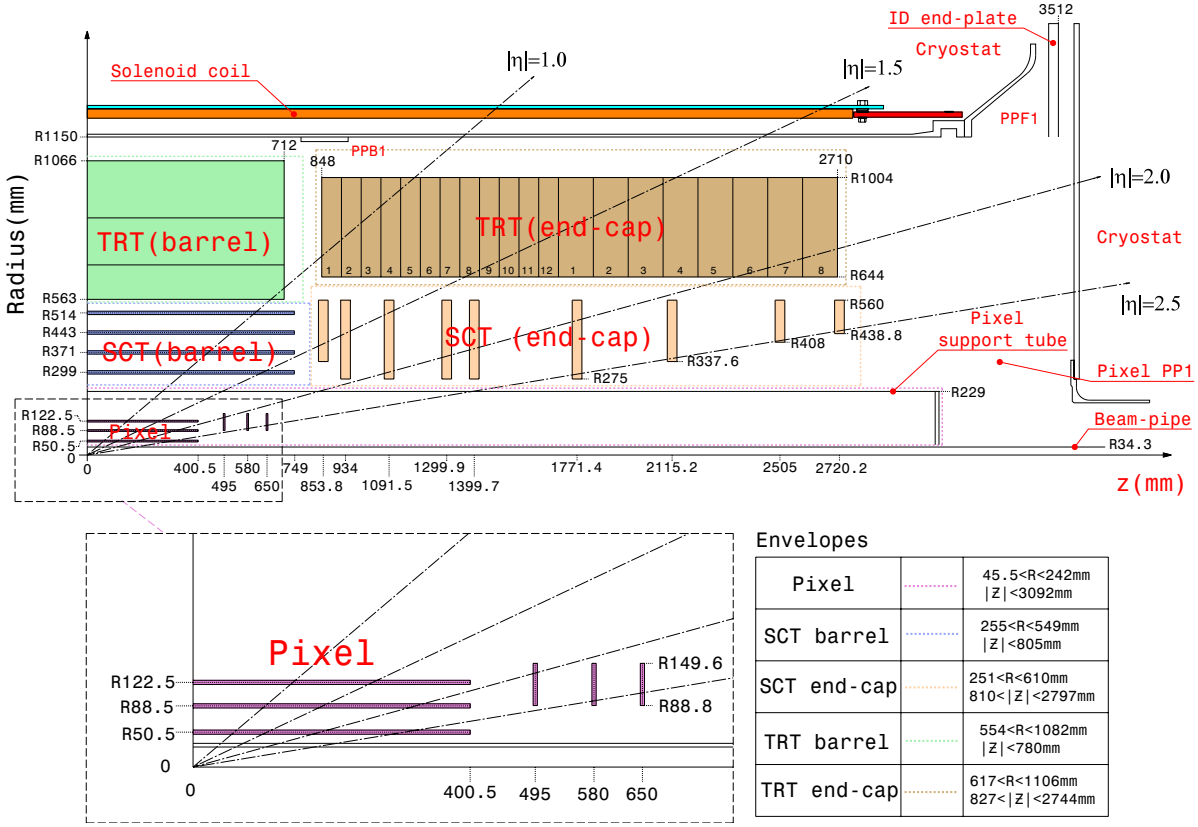


Figure 1.5: Schematic view of a quarter-section of the ATLAS Inner Detector showing each of the major detector elements with its active dimensions and envelopes [10].

The ID consists of three independent but complementary subdetectors. The envelopes of each-subdetector are shown in Figure 1.5. At smaller radii, high-resolution pattern recognition capabilities are available using discrete space-points from silicon pixel layers which are especially important for primary vertex measurements. In the intermediate region, stereo pairs of silicon micro-strip (SCT) detector modules are deployed, producing additional hits for better resolution of charged particle tracks. At larger radii, the transition radiation tracker (TRT) comprises many layers of gaseous straw tube elements interleaved with transition radiation material. With an average of 36 hits per track, it provides continuous tracking to enhance the pattern recognition and improve the momentum resolution up to $|\eta| = 2.0$ and electron identification complementary to that of the calorimeter over a wide range of energies (between 0.5 GeV and 150 GeV). Table 1.3 summarizes the most important performance characteristics of the Inner Detector.

The high-radiation environment imposes stringent conditions on the Inner Detector sensors,

on-detector electronics, mechanical structure and services. The pixel barrel layers and disks were designed to withstand a 1 MeV neutron equivalent fluence F_{neq} of up to $8 \cdot 10^{14}\,\mathrm{cm}^{-2}$. For the pixel inner vertexing layer closest to the beampipe (the *B-layer*), this dose will be reached after about three years of operation at design luminosity. It will therefore have to be replaced after this time period.

The innermost parts of the SCT are designed to tolerate F_{neq} of up to $2 \cdot 10^{14}\,\mathrm{cm}^{-2}$.

Reconstruction efficiency for muons with $p_T = 1\,\mathrm{GeV}$	96.8 %
Reconstruction efficiency for pions with $p_T = 1\,\mathrm{GeV}$	84 %
Reconstruction efficiency for electrons with $p_T = 5\,\mathrm{GeV}$	90 %
Momentum resolution at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 0$	1.3 %
Momentum resolution at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 2.5$	2.0 %
Momentum resolution at $p_T = 100\,\mathrm{GeV}$ and $\eta \approx 0$	3.8 %
Momentum resolution at $p_T = 100\,\mathrm{GeV}$ and $\eta \approx 2.5$	11 %
Transverse impact parameter res. at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 0$	75 μm
Transverse impact parameter res. at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 2.5$	200 μm
Transverse impact parameter res. at $p_T = 1000\,\mathrm{GeV}$ and $\eta \approx 0$	11 μm
Transverse impact parameter res. at $p_T = 1000\,\mathrm{GeV}$ and $\eta \approx 2.5$	11 μm
Longitudinal impact parameter res. at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 0$	150 μm
Longitudinal impact parameter res. at $p_T = 1\,\mathrm{GeV}$ and $\eta \approx 2.5$	900 μm
Longitudinal impact parameter res. at $p_T = 1000\,\mathrm{GeV}$ and $\eta \approx 0$	90 μm
Longitudinal impact parameter res. at $p_T = 1000\,\mathrm{GeV}$ and $\eta \approx 2.5$	190 μm

Table 1.2: Performance characteristics of the ATLAS Tracker [7].

1.4 The Inner Detector Sensors

This section describes the sensors of the pixel and SCT silicon sensor modules. Details on the TRT sub-system will be neglected.

As discussed in Section 1.3, the sensors are subject to large integrated radiation doses. They have therefore been developed and tested to withstand the expected irradiation, with a safety factor of approximately two.

1.4.1 Pixel and SCT Detector Sensors

The pixel [12] and SCT [13] sensors are required to maintain adequate signal-to-noise ratios over the detector lifetime of ten years at design luminosity (with the exception of the pixel vertexing layer, as discussed above). The integrated radiation dose has important consequences for the sensors of both detectors. In particular the required voltage to fully deplete the detector substrate, determined by the effective doping concentration, depends on both the irradiation and the subsequent temperature-sensitive annealing. The sensor leakage current also increases

linearly with the integrated radiation dose. The n-type bulk material effectively becomes p-type after a fluence F_{neq} of $2 \cdot 10^{13}\,\mathrm{cm}^{-2}$. This effect is called *type inversion*. After that, the effective doping concentration grows with time in a temperature-dependent way which severely reduces charge collection efficiency.

The pixel sensors required the most leading-edge and novel technology to meet the very stringent specifications on radiation hardness, resolution and occupancy in the innermost layers. The sensors are 250 μm thick detectors, using oxygenated n-type wafers with readout pixels on the n^+-implanted side of the detector. All of the 1744 pixel sensors share the same rectangular shape of $19 \times 63\,\mathrm{mm}^2$. The minimum pixel size is $50 \times 400\,\mu\mathrm{m}^2$, dictated by the readout pitch of the front-end electronics. As shown in Figure 1.5, the pixel modules are arranged in three barrel layers and two end-caps with three disk layers each.

For reasons of cost and reliability, the 15912 sensors of the SCT use a classic single-sided p-in-n technology with AC-coupled readout strips. The sensors will initially operate at $\approx 150\,\mathrm{V}$ bias voltage, but operating voltages between 250 and 450 V will be required for good charge collection efficiency after ten years of operation, depending on the sensor radius, the integrated luminosity and the length of warm-up periods. The sensor thickness of $285 \pm 15\,\mu$m is a compromise between the implied operating voltage, the primary signal ionisation and the simplicity of fabrication. The strip pitch was determined by the required digitising precision, granularity, particle occupancy and noise performance. A strip pitch of 80 μm with two 6 cm-long sensors daisy-chained was chosen for the rectangular barrel sensors. Radial strips of constant azimuth with mean pitch of $\approx 80\,\mu$m were chosen for the trapezoidal end-cap sensors. There are a total of 768 active strips of up to 12 cm length per sensor, plus two strips at bias potential to define the sensor edge.

2 Upgrade Scenarios for the LHC and ATLAS

2.1 Motivation for the Upgrade

The main motivation for an upgrade of the LHC is the maximum exploitation of the existing tunnel, machine and detectors. Several scenarios were considered, to be more precise the upgrade to a higher centre-of-mass energy of $\sqrt{s} \approx 28\,\text{TeV}$ and/or an increase of the instantaneous luminosity. This upgrade is commonly called *Super-LHC* or *SLHC*. Prospects for the physics potential of different upgrade scenarios are to be found in [14], whereas details of the impact on the collider are explained in detail in [15].

An energy upgrade would render most of the current machine's magnets useless as they would have to be replaced by magnets with a much higher bending power. A magnetic field of about 15 T would be required if the LEP/LHC tunnel should be reused. At the moment, it is not possible to produce field strengths as high as this with existing collider technology. Furthermore, higher beam energies lead to a massive increase of synchrotron radiation which is hard to extract and demands heavy shielding. However, an energy upgrade would be necessary to extend the mass reach of the LHC significantly and would be directly exploitable by the experiments (i.e. without major changes to detector designs).

On the other hand, a luminosity upgrade would primarily leave the machine unaffected. Depending on the scenario, only the focus quadrupoles near the interaction regions and parts managing the bunch insertion from the SPS would have to be exchanged. Additionally, a luminosity upgrade could proceed in several steps, avoiding large downtimes of the machine and detectors. Considering realistic modifications, an increase in luminosity of about an order of magnitude could be achieved.

Thus, a decision for upgrading the luminosity is the most probable scenario at the moment for it leaves the majority of LHC parts untouched. Additionally, for scenarios which would imply changes to the interaction regions, modifications are to be made in their forward regions which will have to be exchanged due to irradiation damage anyway or be removed for the replacement of inner detector parts.

It has to be noted that a luminosity upgrade of the LHC is mainly useful to improve the physics performance of the two high-luminosity experiments ATLAS and CMS. The B-physics studies at LHCb are already performed at a luminosity which is an order of magnitude smaller than at the general-purpose experiments during LHC runs. In addition to that, their physics programme is intended to be completed at the end of the planned LHC runs. Also, the benefits of a higher instantaneous luminosity would be reduced for heavy ion collisions due to very large nuclear cross-sections, leading to a reduction of beam lifetime and therefore of the integrated luminosity.

From a physics point of view, even if the existence of supersymmetry or large extra dimensions were to be discovered by the LHC, full understanding of the theory behind could be extended by additional measurements with a complementary linear collider (e.g. search for the

reason of supersymmetry breaking or what fixes the sizes of extra dimensions). It is clear that the true gain of an upgrade will heavily depend on the results to be obtained at the LHC. However, it is possible to make some prospects for the performance advantage of an upgraded accelerator over the LHC with respect to the standard performance criteria used in the LHC design. A few details on the mass reach for new physics phenomena is given in Table 2.1.

To summarize, an energy upgrade of the LHC is technically and financially unfeasible. The major extension of the mass reach of the LHC will be reserved to a future collider of a larger radius, such as the VLHC if it will be ever built at all. The only realistic option for the near future will be a luminosity upgrade which will be discussed in detail in the following section.

Process	**LHC** $L \approx 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ $\mathscr{L} \approx 100\,\mathrm{fb}^{-1}$	**SLHC** $L \approx 10^{35}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ $\mathscr{L} \approx 1000\,\mathrm{fb}^{-1}$
squarks	2.5 TeV	3 TeV
Z'	5.4 TeV	6.5 TeV
q^*	6.5 TeV	7.5 TeV
Two extra dimensions	9 TeV	12 TeV
Triple Gauge Coupling (95%)	$\lambda_\gamma = 0.0014$	$\lambda_\gamma = 0.0006$
Λ compos.	35 TeV	50 TeV

Table 2.1: Comparison of physics discovery reaches between the LHC and the SLHC at integrated luminosities of $100\,\mathrm{fb}^{-1}$ and $1000\,\mathrm{fb}^{-1}$ respectively, corresponding to one full year of running at nominal luminosity ([16] and [17])

2.2 Luminosity Upgrade Scenarios for the LHC

When assuming round beams, the instantaneous luminosity at the main LHC interaction points IP1 (ATLAS) and IP5 (CMS) is roughly described by

$$L \approx n_b \frac{\gamma f_{rev}}{2 r_p} \frac{1}{\beta^*} N_b \, \Delta Q_{bb} \, F_{profile} \, F_{hg}, \tag{2.1}$$

where ΔQ_{bb} denotes the total beam-beam tune shift (limited to ≈ 0.01 which was derived from previous hadron collider experiments), f_{rev} the revolution frequency of the bunches, $F_{profile}$ a form factor depending on the longitudinal profile of a bunch (1 for a Gaussian and $\sqrt{2}$ for a uniform profile) and F_{hg} the reduction factor due to the hourglass effect, which is relevant for bunch lengths comparable to, or smaller than, the IP beta function β^*.

The values for these parameters at the LHC and different SLHC upgrade scenarios are summarized in Table 2.2.

A luminosity upgrade could happen in different stages, beginning from the *nominal* luminosity settings. Without hardware changes, the performance can be pushed to *ultimate* LHC performance by increasing the number of protons per bunch and maxing out the magnetic field strength of the collider dipoles to its design limit of 9 T. In following phases, the implementation of hardware modifications to machines and detectors are to be made.

Parameter	Symbol	Nominal	Ultimate	old	ES	LPA
number of bunches	n_b	2808	2808	5616	2808	1404
protons per bunch	N_b [10^{11}]	1.15	1.7	1.7	1.7	4.9
bunch spacing	Δt_{sep} [ns]	25	25	**12.5**	25	**50**
average current	I [A]	0.58	0.86	1.72	0.86	1.22
normalized transverse emittance	$\gamma\epsilon$ [μm]	3.75	3.75	3.75	3.75	3.75
longitudinal profile		Gaussian	Gaussian	Gaussian	Gaussian	uniform
rms bunch length	σ_z [cm]	7.55	7.55	3.78	7.55	11.8
beta function at IP1&5	β^* [m]	0.55	0.5	0.25	0.08	0.25
(effective) crossing angle	θ_c [μrad]	285	315	445	0	381
Piwinski angle	ϕ	0.4	0.75	0.75	0	2.01
hourglass factor	F_{hg}	1.00	1.00	1.00	0.86	0.99
peak luminosity	$\hat{L}$ [$10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$]	1.0	2.3	9.2	15.5	10.6
events per crossing		19	44	88	294	**403**
rms length of luminous region	σ_{lum} [mm]	45	43	21	53	37
initial luminosity lifetime	τ_L [h]	22.2	14.3	7.2	2.2	4.5
average luminosity (T_{ta} = 10 h)	L_{av} [$10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$]	0.5	0.9	2.7	2.4	2.5
optimum run time (T_{ta} = 10 h)	T_{run} [h]	21.2	17.0	12.0	6.6	9.5
average luminosity (T_{ta} = 5 h)	L_{av} [$10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$]	0.6	1.2	3.7	3.6	3.5
optimum run time (T_{ta} = 5 h)	T_{run} [h]	15.0	12.0	8.5	4.6	6.7
e-cloud heat load for δ_{max} = 1.4	P_{ec} [W/m]	1.07	1.04	**13.3**	1.0	0.4
e-cloud heat load for δ_{max} = 1.3	P_{ec} [W/m]	0.44	0.6	7.9	0.6	0.1
SR heat load	P_{SR} [W/m]	0.17	0.25	0.5	0.25	0.36
image-current heat load	P_{ic} [W/m]	0.15	0.33	1.85	0.33	0.70

Table 2.2: Collider parameters for the (1) nominal and (2) ultimate LHC compared to three upgrade scenarios with (3) shorter bunches at 12.5 ns spacing [old baseline], (4) more strongly focused ultimate bunches with early separation at 25 ns spacing [ES] and (5) longer flat bunches at 50 ns spacing in a regime of large Piwinski angle [LPA] [18].

In the last few years, collider experts have been proposing several options for upgrade modifications to the LHC. The first approach was to half the bunch spacing to 12.5 ns. However, it became clear that is virtually impossible to extract the large heat loads from electron cloud effects (at least an order of magnitude higher than at the LHC). It therefore seems that this scenario is ruled out. Two different approaches for the upgrade are currently discussed among collider experts and the detector collaborations.

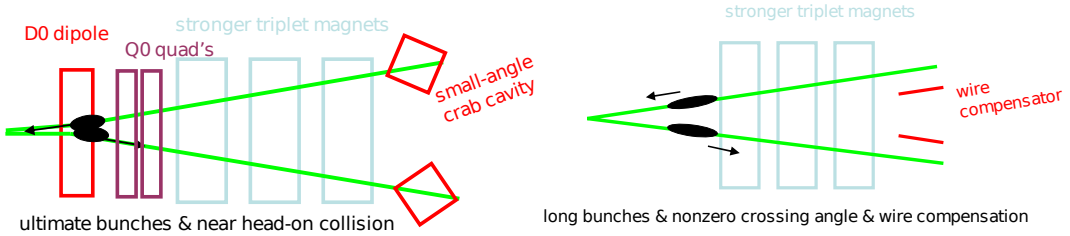
The first is the *early separation* (ES) scenario (see Figure 2.1a) which preserves the ultimate LHC bunch structure but implies modifications to the detectors. Dipole magnets would have to be inserted in the forward regions, approx. 3 m from the interaction point and possibly additional quadrupole magnets a few meters downstream. This scheme also introduces *crab cavities* for the first time to hadron colliders in order to reduce β^* to 10 cm.

The *large Piwinski angle* (LPA) scenario shown in Figure 2.1b is an alternative which leaves the detectors untouched. The Piwinski angle is defined as

$$\phi = \theta_c \sigma_z / (2\sigma^*), \tag{2.2}$$

where θ_c denotes the beam crossing angle and σ_z and σ^* characterise the longitudinal and transverse bunch size, respectively. The bunch spacing is to be doubled to 50 ns whereas the bunch sizes and proton number are to be increased. The effective crossing angle can thus be much larger to gain a similar luminosity as in the ES scenario which is why only minor modifications would have to be made to the LHC. As an additional downside, the instantaneous luminosity would be even higher than in the early separation scenario which has consequences

for the occupancy of the hardware and makes event reconstruction more difficult because of higher in-time pile-up.



(a) Early separation scheme (ES) (b) Large Piwinski angle scheme (LPA)

Figure 2.1: Interaction region layouts for the two main upgrade scenarios [18].

The pile-up of minimum bias events has a severe impact on the ability to use forward jet tagging and central jet vetoing as a tool to enhance signal to background ratios at SLHC luminosity [17]. It is crucial to study the impact on the reconstruction efficiencies, not only for the tracking detectors but also for the calorimeters, especially in the forward region where the density of low p_T tracks will be much larger than for the LHC. Also the muon wheels could be affected by an increased luminosity. In the barrel region however, the calorimeters and the muon system should remain largely unaffected.

The Physics Opportunities and Future Proton Accelerators (POFPA) committee recommended the 50 ns bunch spacing configuration as the SLHC baseline [19]. The collaborations of ATLAS and CMS will have to carefully study if this option is feasible for physics at their particular experiments.

2.3 Consequences for the ATLAS Inner Detector

A luminosity upgrade of the LHC will happen at a time when the ATLAS Inner Detector has reached the end of its life-cycle. The innermost layer of the Pixel Detector, the so called *B-Layer* will have been already replaced once, probably after about four years of service in 2012 because of irradiation damage. In the end, the complete silicon detector layers are to be removed.

Also the TRT can not maintain its former performance due to aging effects. In addition to that, it is no longer feasible to have TRT-like technology in place in a high luminosity environment. Due to long dead times of the gas-type detector elements, enormous occupancy problems will arise.

It is therefore mandatory to build fast and irradiation resistant detectors for the upgrade which will very probably be based on silicon pixel and silicon strip technology only. A complete replacement of the Inner Detector is therefore necessary to maintain the required performance for b-quark and hadronic tau tagging [14].

Increased granularity will help reduce detector occupancy and thus preserve momentum and displaced vertex resolution. To reduce the neutron background in the Inner Detector region, the material budget has to be kept small, even reduced in comparison to the detector area and

services of the current layout. As an example, serial powering of the detector modules is a very attractive option to reduce the fraction of the material budget contributed to by cabling.

The omission of the TRT will have another important consequence: the lack of its neutron-absorbing properties will result in high radiation doses in the outer regions of the ATLAS Tracker where secondary particles returning from the calorimeters pose a threat to silicon detectors and electronics [20]. It is therefore planned to line the cryostat with a 5 cm polyethylene moderator.

To meet all the requirements for the upgrade, new detector technologies have to be researched and developed. The most notable proposal maybe is the employment of three-dimensional silicon sensors [21] which could be used to increase radiation hardness in the pixel and inner strip detector layers. Prototypes of p-type 3D silicon strip detectors have already been successfully produced and tested after irradiation [22]. Switching from n-doped sensor bulk material to p-type silicon avoids the problem of type-inversion mentioned in section 1.4.1. The charge collection efficiency is improved in comparison to the presently employed p-in-n detectors due to n-side readout. This means collection of electrons, which have a higher mobility and hence lower susceptibility to charge trapping. Trapping is one of the main consequences of irradiation damage at SLHC fluences.

2.4 Design Proposals for the ATLAS Inner Detector Upgrade

Independent of the detector technology, decisions have to be made at which radii a certain granularity is needed to guarantee the necessary performance for charged particle tracking.

Basically, three different types of silicon detectors are planned to be used in the Tracker upgrade. Like in the current layout, pixel detectors are to be used for the inner layers. In the mid-range, *short-strip (SS)* silicon detectors are intended to be employed, whereas in the outermost region *long-strip (LS)* modules comparable to the current SCT modules could be deployed. The schematics in this sections and pictures throughout this thesis show the three module types in green, blue and red colors, respectively. For the sake of comprehensibility, the silicon strip detector will be referred to as SCT as a whole just like in the current detector. When the short- and long-strip subsystems are addressed, it will be indicated by the use of the according abbreviations.

In the barrel region, proposals have converged to 4 cylindrical layers of pixel detectors, followed by 3 SS layers, which are enclosed by 2 LS layers.

In the endcap regions, the innermost discs are built from 3 layers of pixel modules. The SCT disc layers are not uniformly divided into SS and LS modules but change strip length depending on η in their constituent rings. Different proposals how the structural components of the discs could be designed exist.

As a starting point for layout studies, the *Strawman* layout has been defined with general parameters for barrel (Table 2.3) and endcap (Table 2.4) layer sizes and positions. The microstructure of the different layers (i.e. module types and geometries) are not fixed yet because they heavily depend on the used technology, not only for the sensors themselves but also for the services.

The layout geometry defined by these parameters is also called the *quasi-projective* Strawman layout due to the staggered barrels ending in about the same region of pseudo-rapidity η.

Layer	**Technology**	**Radius** (mm)	**z-Extent** ($\pm$ mm)	**Pitch** (μm)	**Length** ($\pm$ mm)	ϕ **sectors**
b	Pixel	50	400	50	0.2	not yet defined
1	Pixel	120	400	50	0.2	not yet defined
2	Pixel	180	400	50	0.4	not yet defined
3	Pixel	240	400	50	0.4	not yet defined
4	SS	320	1000	80	35	not yet defined
5	SS	460	1000	80	35	40
6	SS	600	1000	80	35	48
7	LS	750	1900	100	90	27
8	LS	950	1900	100	90	40

Table 2.3: Barrel parameters for the Strawman layout [23].

Disc	**Technology**	**z-Position** (mm)	**Inner-r** (mm)	**Outer-r** (mm)	**Pitch** (μm)	**Length** ($\pm$ mm)
1	Pixel	500	66	280	50	0.4
2	Pixel	625	87	280	50	0.4
3	Pixel	750	107	280	50	0.4
4	SS & LS	1200	182	600	80	35-100
5	SS & LS	1650	256	600	80	35-100
6	SS & LS	2100	331	600	80	35-100
7	SS & LS	2600	413	950	80	35-100
8	SS & LS	3200	512	695	80	35-100

Table 2.4: Endcap parameters for the Strawman layout [23].

Variants of the layout exist and are shown in the following subsections.

2.4.1 The Quasi-Projective Layout

The barrel length increases with radius throughout the different subdetectors. This option, as shown in Figure 2.2, is most similar to the current layout.

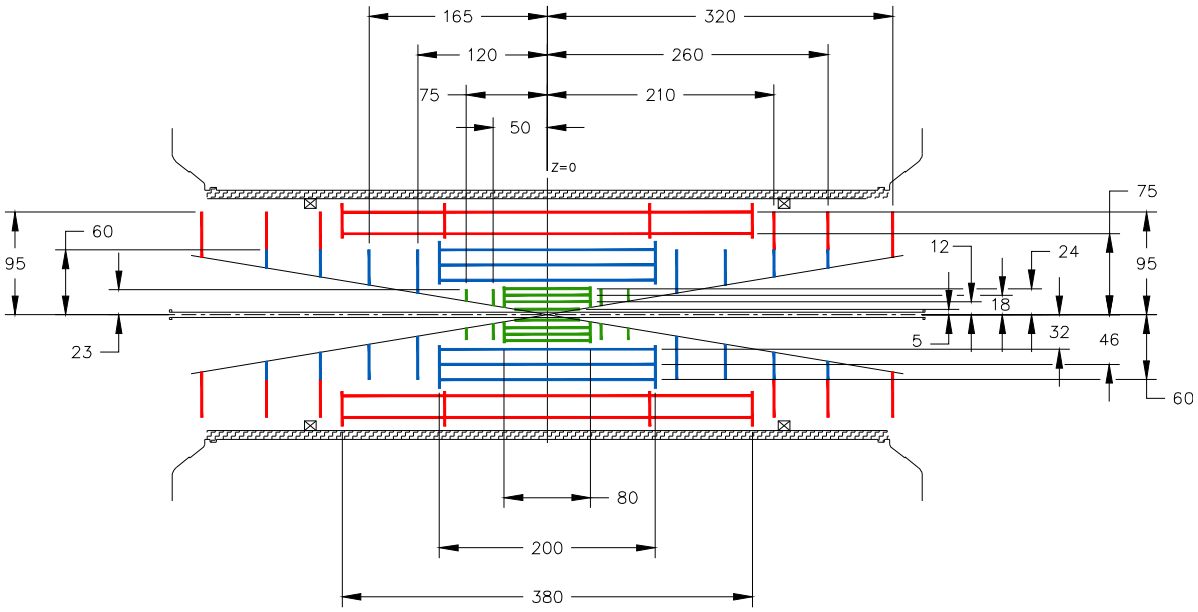


Figure 2.2: Sectional drawing of the "quasi-projective" strawman layout [23].

2.4.2 The Equal-Length SCT Barrel Layout

In this layout (Figure 2.3), the routing of services is simplified by choosing SCT barrels of equal length. Also, insertion and removal of the endcaps is much easier.

2.4.3 The Conical Layout

The barrel layers of this layout (Figure 2.4) are identical to the one in the previous section. The SCT endcaps are tilted to form cones. In this way, a higher stiffness in comparison to flat discs is reached. The layout is only shown for completeness and is not further studied in the framework of this thesis.

2.5 Agenda for Simulation Studies

For every layout, the precise positions of the layers have to be optimised by simulation. For the strip layers, it has to be determined, in which regions double-sided/stereo modules would improve the performance of the detector significantly and justify the additional amount of material. In addition to that, a double B-layer scenario for all layouts would probably improve vertexing precision enormously but would also suffer from irradiation problems. A cost-benefit

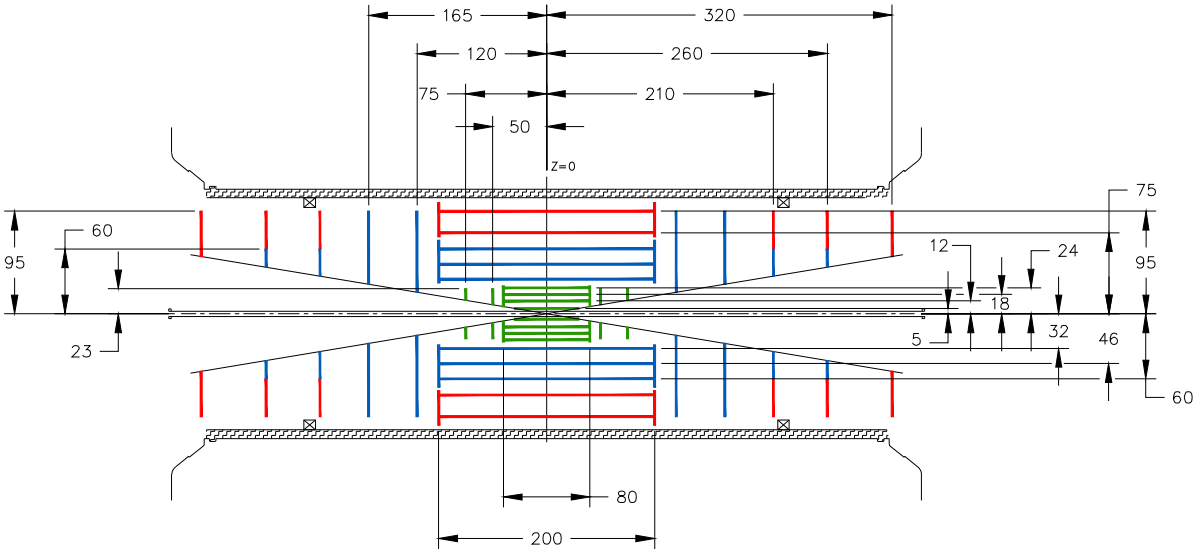


Figure 2.3: Sectional drawing of the strawman layout with strip barrels of equal length [23].

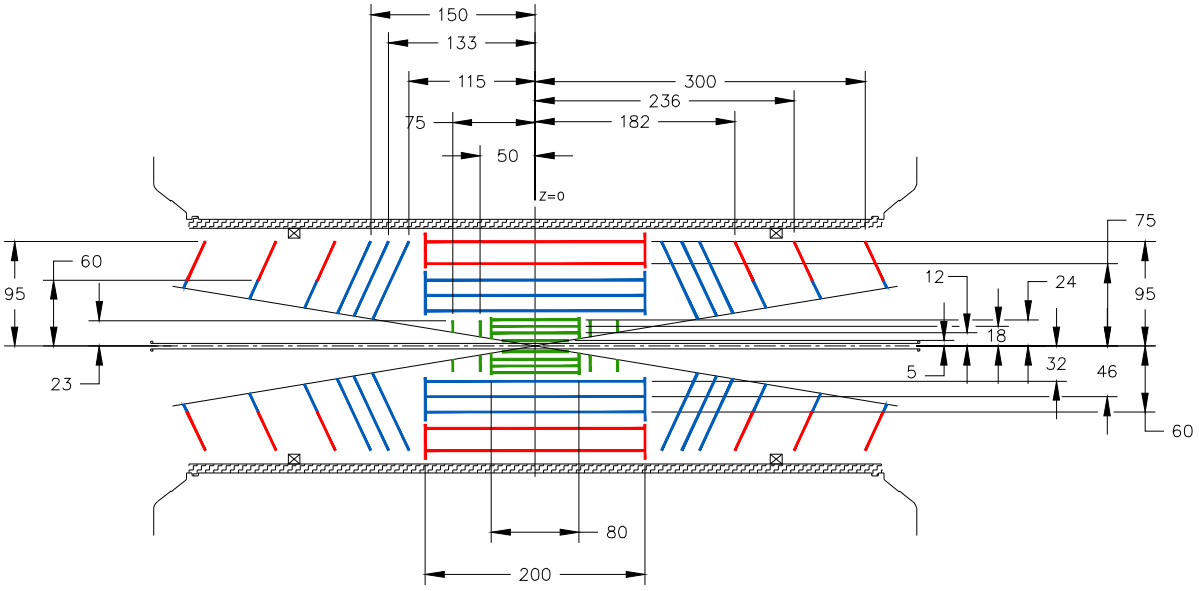


Figure 2.4: Sectional drawing of the conical strawman layout [23].

analysis here would greatly help and early results could influence the B-layer upgrade for the current ATLAS Tracker.

The proposed layouts will have to be evaluated with respect to basic performance param-

eters like track parameter resolutions, primary vertex reconstruction performance, b-tagging performance, as well as to benchmark physics processes. The latter will be chosen according to prospects of signatures to be encountered at the SLHC. Some of these studies will also depend on calorimeter information (i.e. b-tagging).

A complete list of proposed benchmark studies is to be found in [23]. Only a few relevant to this thesis are listed here.

Track resolutions can be derived from single electron and muon tracks of a p_T spectrum spanning from 1 GeV up to possibly a few hundred GeV. However, it suffices to probe samples up to 100 GeV because performance should not change very much above this value.

For vertexing performance measurements, it is most convenient to use $t\bar{t}$ event samples, possibly merged with a different number of pile-up events to test luminosity (and therefore occupancy) scaling as well as the algorithmic stability of the vertexing reconstruction modules.

3 Detector Simulation within the ATLAS Software Framework

3.1 The ATLAS Software Framework ATHENA

In this section, a basic introduction to the design of the ATLAS Software Framework *ATHENA* will be given and the details of the event reconstruction software will be elaborated upon. An extensive presentation of the ATLAS computing model can be found in the Computing Technical Design Report [24].

At today's high-energy physics experiments, computers are indispensable tools to aid during the planning, construction and operation phases. Huge amounts of data have to be processed at each of these stages, which is especially true at the LHC and ATLAS.

The software which handles these tasks has to be fast, flexible and modular. The ATLAS software framework ATHENA was designed to meet all of these requirements.

The ATHENA framework is an enhanced version of the Gaudi framework [25] that was originally developed by the LHCb collaboration, but is now a common ATLAS-LHCb project and is in use by several other experiments including GLAST [26] and HARP [27]. ATHENA and Gaudi are concrete realizations of a component-based architecture (also called Gaudi) which was designed for a wide range of physics data-processing applications. The fact that it is component-based allows for flexibility in developing both a range of shared components and, where appropriate, components that are specific to the particular experiment and better meet its specific requirements.

To be fast, the software packages are basically written in C++, a high-level programming language, which matches the performance of C or FORTRAN (at least when used properly), but allows to create computer programs in a classic object-oriented approach. Also more recent "state-of-the-art" software engineering paradigms can be realised with C++ (i.e. generic programming).

Flexibility is achieved by the generation of dynamic loadable libraries which are assembled at run time by Python scripts. The ATHENA runtime basically extends a standard Python interpreter (and can also be used interactively). The setup for each run of the software is done in a top-level Python script, commonly called *topOptions*, or more generally *jobOptions* file.

External software like common high-energy physics libraries (e.g. CLHEP [28]), event generators and physics simulation frameworks are used extensively and made available through wrapper packages.

3.2 Monte Carlo Detector Simulation

The so-called Monte Carlo simulation of physics events in particle collisions and the induced detector response is an essential technique in high energy physics. During the preparation phase

of an experiment this is the only source of predictions for the effectiveness of a given detector setup with respect to various types of physics events. Once data taking has started, it is the foundation of tests of theoretical models against the real detector response. Furthermore, since most of the readout and event reconstruction software is developed in parallel to the detector installation, simulated data is — besides data taken in test beam setups and commissioning runs using cosmic rays — the only input available for the testing and performance validation of the reconstruction software.

The event simulation process can be divided into two different stages. The primary physics event is usually produced by common high energy physics programs such as PYTHIA [29] or HERWIG [30], generally referred to as *Monte Carlo event generators*. Their output contains the result of the hard scattering process which took place at the interaction point of the simulated experiment and is represented by the four-momentum vectors and identity of the particles.

In a second step, this information is used for the simulation of the detector response, which depends on the experimental setup. It has to account for the peculiarities of the detector geometry and the integrated detection technologies.

In the following, details of the calorimeter simulation will be neglected for only the simulation of tracks in the ATLAS Inner Detector is of interest within the scope of this thesis. The simulation of trajectories of charged particles followed by hit creation, as well as the derivation of *tracks* from the hit information by the *track reconstruction software*, is both referred to as *tracking*. However, it should be evident from the context, which interpretation is valid.

In the most sophisticated detector simulation — in the following referred to as *full simulation* — these components are realised by a very detailed geometrical model of the detector and an accurate description of the particle interaction with the sensitive detector material, followed by a realistic hit cluster creation. In fast simulation techniques based on parameter smearing, on the other hand, both components are respected intrinsically by smearing functions that are obtained from full simulation results.

These two track simulation techniques are extensively used in ATLAS: the detailed full detector simulation that is based on the well known *Geant4* simulation toolkit [31], and a fast track simulation (as a part of the *ATLFAST* [32] program) that works on the basis of four-momentum vector smearing.

Recently, a new fast simulation has emerged: the *Fast ATLAS Track Simulation (Fatras)* realises a full Monte Carlo simulation approach, but gains a significant speed advantage by using a simplified geometry model, which was originally used by the ATLAS track reconstruction software only.

Figure 3.1 presents an overview on the track simulation frameworks available in ATLAS.

3.2.1 Full Detector Simulation with Geant4

The physically most accurate simulation framework for the modelling of physics events within the ATLAS software framework is Geant4. It is a complete rewrite of the FORTRAN-based GEANT [33] (**G**eneration of **E**vents **AN**d **T**racks) simulation toolkit in C++ with a modern object-oriented design. Facilities of Geant4 used in the full simulation of ATLAS events include the handling of the *geometry*, particle *tracking* and the generation of a *detector response* which is explained in the next paragraph.

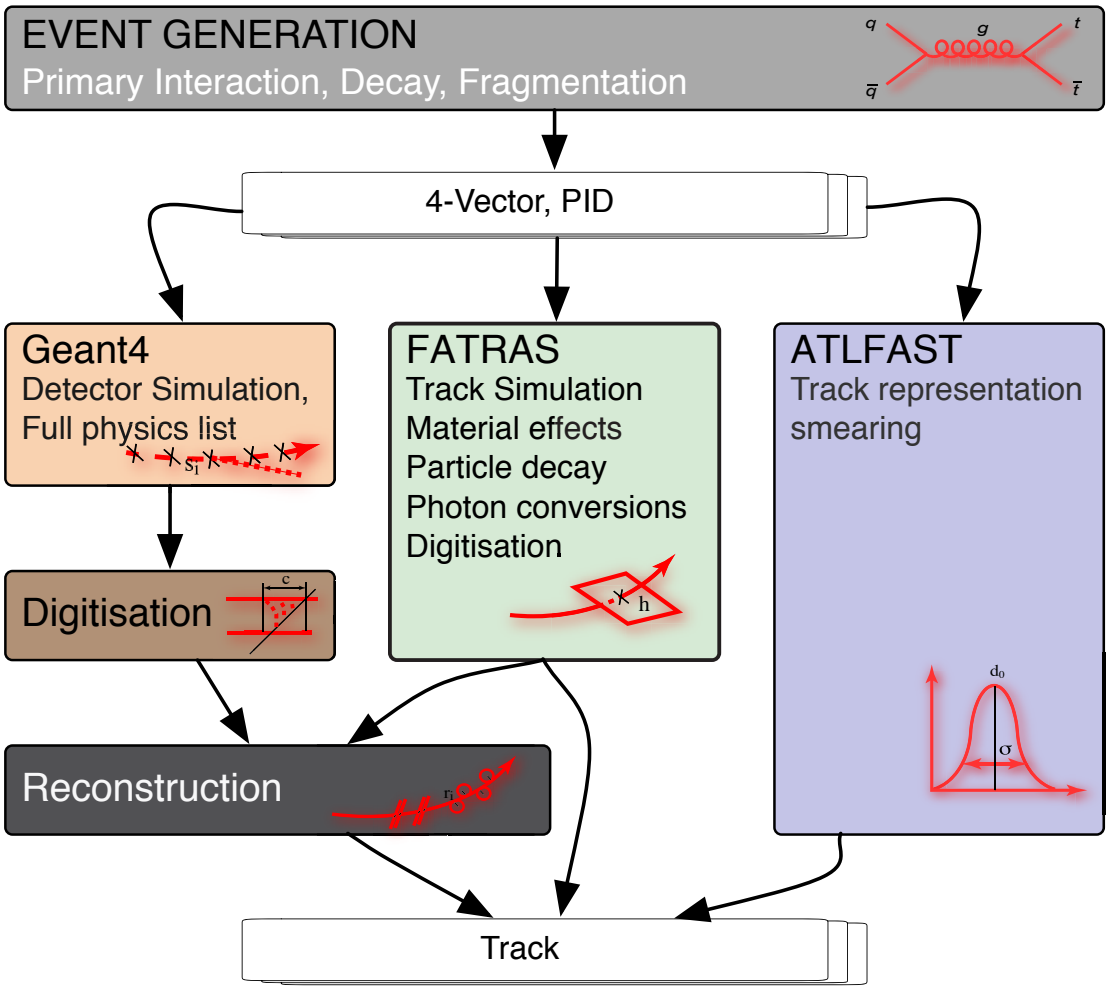


Figure 3.1: Schematic overview of the different simulation frameworks in ATLAS.

The *geometry* is a representation of the physical layout of the experiment, including detectors, absorbers which affect the path of particles. *Tracking* is the simulation of the passage of a particle through the geometry. This involves considering possible interactions and decay processes. The simulation of the *detector response*, when a particle traverses active sensor materials, is the final step.

In ATLAS, the full detector simulation propagates the particle through the a complex geometry model and simulates the interaction of the particle with the sensitive and non-sensitive detector material. Many different physics processes, such as particle decay or electromagnetic and hadronic interactions of the particle with the detector material are performed. Some of these processes produce new particles, which are added to the stack of particles to be processed. This procedure is iterated until the child particle fails to pass a certain energy threshold.

After the simulation, readout signals are generated from hits in the sensitive detector parts and further processed in a *digitisation* module that prepares the simulated information for the reconstruction algorithms. The digitisation is not part of the Geant4 simulation toolkit, but carried out by a dedicated module that is integrated into the software framework of the experiment.

The reconstruction software, a so-called *online* application for the event triggering and the *offline* part for the final event reconstruction and analysis, is executed subsequently to the detector simulation and digitisation. It yields the final resolutions and efficiencies for the reconstructed tracks. During the track reconstruction process, a different detector geometry is used in comparison to the simulation step in Geant4: only the active layers of the detectors are modeled as it is adequate to represent the positions of the sensitive detector areas. This detector description is commonly called the *reconstruction geometry* [34].

For convenience, the compound of full detector simulation, digitisation and offline reconstruction will be in the following referred to as the *offline chain*.

3.2.2 Fast Track Simulation with Atlfast

The ATLFAST simulation bypasses the trajectory building, hit creation, digitisation and reconstruction by applying a smearing function directly on the kinematic parameters of the generated particle. The smearing approach attempts to tune the track parameters to the results of the offline track reconstruction, which is only valid in a purely statistical manner. The smearing functions are obtained from track parameter resolutions that originate from the full simulation and reconstruction chain. Dedicated smearing functions have to be found for different particle types, momentum ranges and vertex radii. The parameter smearing also has to accumulate all aspects of the entire simulation and reconstruction chain (including the detector layout, the material budget, the digitisation and cluster creation, as well as the track reconstruction performance). The smearing functions have to be, in principle, updated if any of the involved components changes substantially. Many physics studies have been performed in the past using the ATLFAST simulation. However, in particular for tracking performance studies, it is not suitable since no hit information is generated.

3.2.3 Fast Track Simulation with Fatras

Fatras is based on the reconstruction tools and uses the common tracking event data model (EDM) [35] natively[1]. During the simulation of the trajectory of a particle, the reconstruction geometry is used to simplify the entire process. Fatras propagates the particle using the ATLAS extrapolation engine [36] for the transport of the track parameters and the inert navigation schema of the reconstruction geometry representation for the trajectory building. Material effects are applied according to the amount of traversed material and physics processes such as bremsstrahlung, photon conversions and the decay of non-stable particles are supported. Initially developed as a validation tool that has been extensively used during the recent development of the track reconstruction components [37], it became a powerful tool for broader purpose including a (limited) usage in the simulation of physics events. Fatras is able

[1] Both, the Geant4 simulation and ATLFAST incorporate their own internal event data model that is optimised for their specific needs.

to produce hits along a trajectory and supports track fitting, vertex reconstruction or even the input creation for the standard offline reconstruction chain. It allows large scale tracking and vertex finding, as well as flavour and lifetime tagging studies (in combination with calorimetric measurements) while guaranteeing low execution times.

Fatras is aimed to be combined with the dedicated fast shower simulation *FastCaloSim* [38] to a fast simulation of the whole ATLAS detector, called *ATLFAST-II*. A very brief outlook and discussion of the current status of such a combined simulation can be found in [39].

Fatras is currently limited to the ATLAS Inner Detector, mainly because the reconstruction geometry description of the second device used for track reconstruction, the muon spectrometer (MS), is still in a prototype version. A future extension of Fatras which includes the MS is one challenging part for the further development of the Fatras project.

Figure 3.2 shows the same $t\bar{t}$ production event simulated with the full simulation and Fatras.

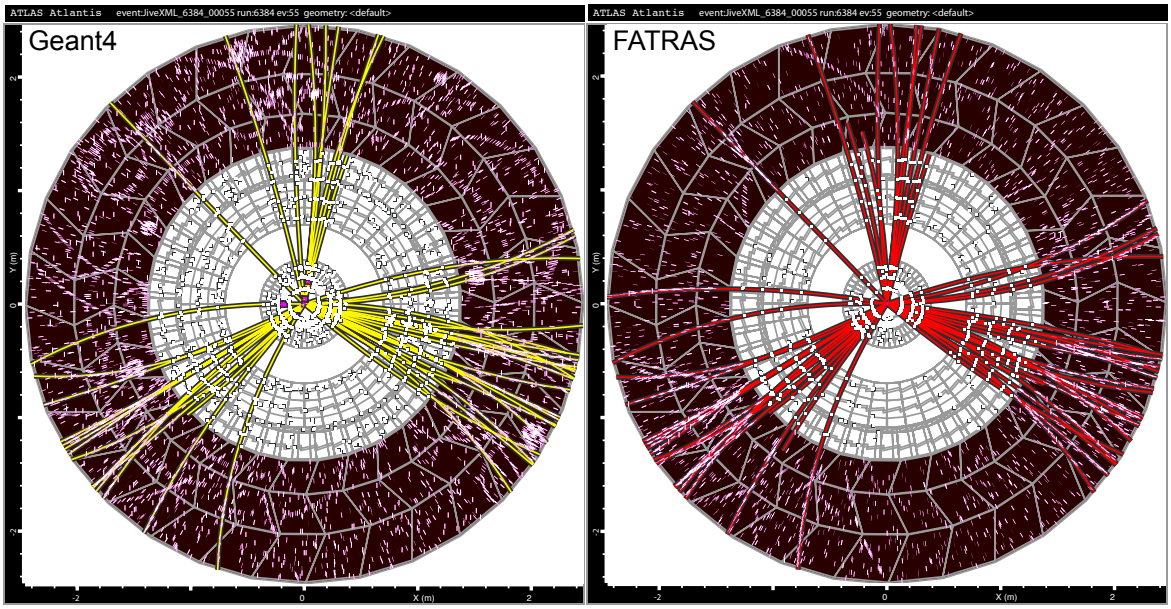


Figure 3.2: The same hard proton-proton scattering event leading to the production of a $t\bar{t}$ pair simulated with the full detector simulation and the Fatras fast simulation in the ATLAS Inner Detector. In both cases, the standard ATLAS offline track reconstruction is performed, the tracks found are also displayed. The visualisation was done with the ATLAS event display ATLANTIS [40].

3.3 The Logical Structure of an ATHENA Run

From here on, names of C++ and Python classes are printed in a typewriter font (e.g. `Algorithm`). Names of ATHENA packages are set in a sans-serif font (e.g. FatrasExample).

Every execution of the ATHENA software framework follows the same pattern. In the topOptions file, the components to be run are set up. After the parsing of this file, the configured libraries are loaded and execution is passed to the C++ runtime. The most important

entity here is the `EventSelector` which is responsible for the setup of the data for each event (e.g. the output of an Monte Carlo event generator). This data is available through data *collections* which are retrieved from the global `StoreGateSvc`. This service is the interface between the running code and files containing persistent data from earlier runs or real data coming directly from the detector.

For each event a sequence of `Algorithm`s is executed. Usually, an `Algorithm` encapsulates a logical stage of data processing. Depending on the configuration, the results of the run can be recorded per event to plain ROOT [41] ntuples or other, ATLAS-specific file types. It makes use of `AlgTool`s, which are acting on the data preselected by the parent `Algorithm`.

This structure will become clearer in the next chapter where the data flow in Fatras is explained.

4 The Fast ATLAS Track Simulation (Fatras)

Development for Fatras, the Fast ATLAS Track Simulation, was started in 2005 for the validation of the track reconstruction tools within ATHENA. With time, it has grown to a competitive fast simulation for the ATLAS Inner Detector which is able to produce results that are very similar to the full simulation with Geant4. Only by the employment of fast simulation methods it is possible to produce large background data samples that are needed for the analysis of physics channels with small signal cross sections and poor signal-to-background ratios.

The author entered the team of core developers during the writing of this diploma thesis and is one of the authors of a corresponding internal ATLAS note ([39]), which was published recently.

This chapter covers the design of the simulation and description of its components. The used calculation methods are explained in detail where necessary to illustrate differences to other simulation models.

4.1 Concepts and Modules of Fatras

Fatras re-uses modules and resources of the offline track reconstruction to a large extent, while only few dedicated components replace standard offline algorithms and tools. The main benefits of this — besides the pure performance aspects — can be summarised as follows:

- **maximum compatibility with the full simulation** to guarantee client/analysis code to run independently of the chosen simulation strategy. Additionally, this makes Fatras a fast development alternative for future analyses, before the final analysis can be performed on fully simulated or real data.
- **automatic adaption to changed detector conditions** through the `TrackingGeometry` and the reconstruction modules used.
- **easy expansion and modification** through the component model.

The remaining part of this section will give a brief overview of the high level modules used in Fatras. A more detailed description of the individual components can be found in Section 4.2.

4.1.1 Module Sequence and Data Flow

The default Fatras simulation sequence consists of six different modules, each of which are realised as an ATHENA `Algorithm` class. Figure 4.1 illustrates the execution sequence in Fatras in a simplified diagram.

These `Algorithm` classes build the simulation chain of Fatras and are in the simplest configuration executed in the sequence as described below. However, the component pattern design

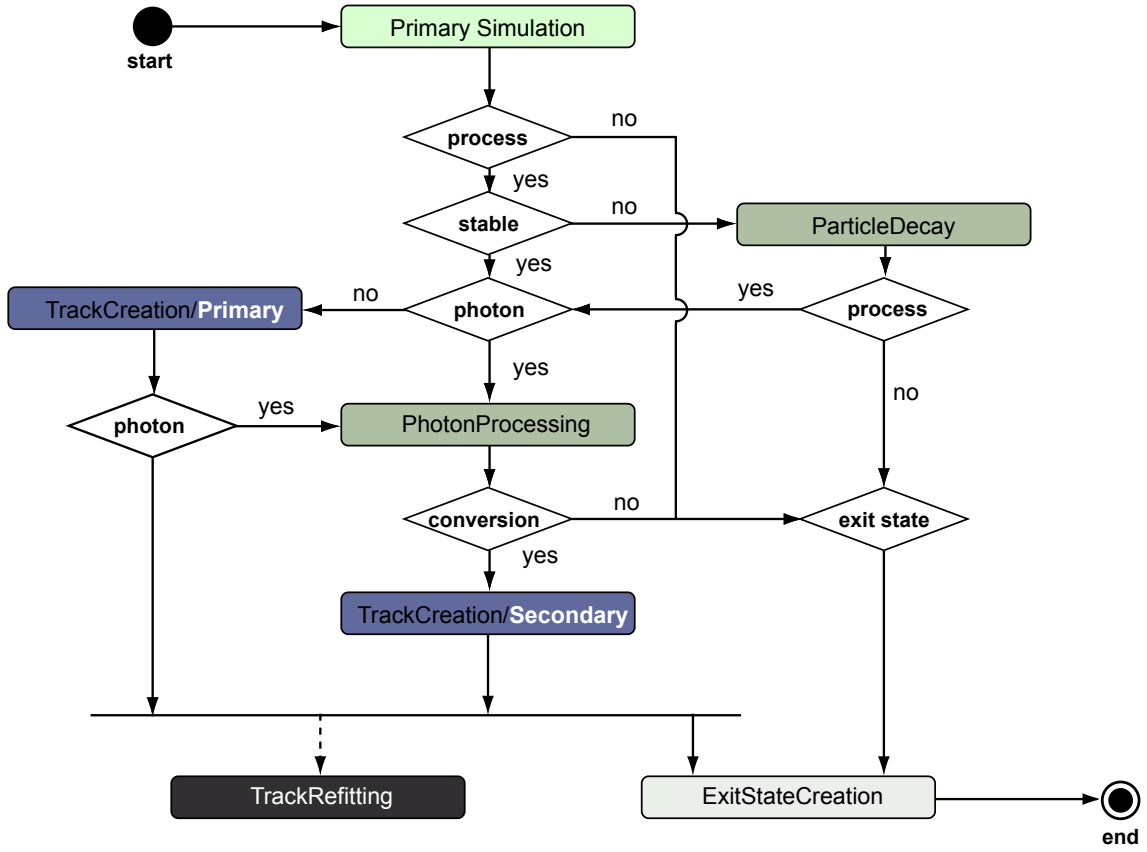


Figure 4.1: UML activity diagram showing the six different modules that build the Fatras simulation. The particles in a given input event collection are processed by the indicated algorithms.

and in particular the application of a central data store instead of direct dependencies between the acting modules allows to modify and extend the algorithmic sequence in a flexible way, e.g. to allow for the insertion of an additional iteration on conversion or decay products. The latter is necessary to correctly handle secondary particles that are produced for example in hadronic interactions.

The following list presents a brief overview of the different Fatras modules. A more detailed description is to be found in section 4.2.

- **Event Preprocessing:** the (optional) `McEventPreProcessing Algorithm` is responsible for the smearing of primary vertices in an input event collection. In its default configuration, it adds a random shift to the nominal vertex (i.e. the origin of the detector coordinate system) which is identical to the estimated uncertainty of the ATLAS beam spot position also used by the offline simulation framework.

- **Primary Simulation:** the primary simulation module is realised either as the simple `SingleTrackSimulation` or the `GenEventSimulation Algorithm`.

The `SingleTrackSimulation` provides simulated single track events mainly targeted at validating the fast track simulation itself. It also creates a fast and convenient framework for the validation of the offline reconstruction chain. The `GenEventSimulation Algorithm`, on the other hand, is designed to process the input provided by event generators.

- **Particle Decay:** particles that are not flagged as stable by the generator are filtered into a dedicated container by the primary simulation.

  Stable decay products are added to the collection of particles for the (primary) track creation, photons are filled into a dedicated collection and particles that do not interact with the detector are scheduled for exiting the detector volume. Following the common Fatras design, these actions are outsourced to `AlgTool`s implementing the `IParticleDecayer` interface.

- **Track Creation:** the track creation `Algorithm` is the core of Fatras. Several instances are executed in the Fatras simulation sequence to allow for an iterative treatment of secondary particles induced by hadronic shower reactions or photon conversions. At this place, *truth tracks* are created from the hit information derived from the intersection of the trajectories of charged particles with sensitive detector layers.

- **Photon Processing:** photons from the Monte Carlo generator (i.e. final state radiation), as well as hard bremsstrahlung originating from the transport of electrons through the detector, are further handled by the dedicated `PhotonProcessing Algorithm`. They are extrapolated through the detector and the conversion probability is calculated as a function of the traversed material. Also pair production is performed during this step. Tracks originating from photon conversions are created by a secondary track creation instance and may again lead to hard photon emission. In the default Fatras configuration, the emitted photons are not processed multiple times for performance reasons, although this would be technically possible.

- **Track (Re)fitting:** this optional algorithm is able to produce tracks with comparable parameter resolutions to tracks produced by the standard offline reconstruction software. The *truth tracks* derived from the simulated trajectories are refitted after a random smearing of the input to remove the bias from the initial track parameters.

- **Exit State Creation:** the last step in the Fatras sequence, the `ExitStateCreation Algorithm`, has no direct implication on the performance of Fatras, but prepares the input for follow up algorithms such as e.g. fast shower parametrisations of the calorimeter. Simulated tracks, neutral particles and photons that did not produce conversions in the inner detector volume are extrapolated to the exit surfaces of the tracker volume.

- **Post Processing:** the hit post-processing module is independent from the main algorithmic sequence in Fatras, but necessary for the preparation of the hit collections for the standard offline reconstruction chain. The simulated hits originating from Fatras tracks and additionally created noise hits are filled into the dedicated hit collections that are used in the ATLAS offline reconstruction.

4.1.2 Modes and Reconstruction Feeding

While the pure track simulation is sufficient as fast but accurate input for a more detailed calorimeter shower simulation, any tracking based study has to at least use the refitted Fatras tracks and — if pattern recognition effects are of importance — finally the reconstructed tracks. These different modes will be in the following referred to as *simulation*, *refit* and *reconstruction mode*, respectively[1]. For the *refit* mode, the simulated track as a whole is passed to a track fitting module, while for the *reconstruction* mode, only the simulated hits are — together with noise hits — filled into the standard hit collections, such that the standard track reconstruction can be performed. The latter includes pattern recognition and track fitting, see Figure 4.2.

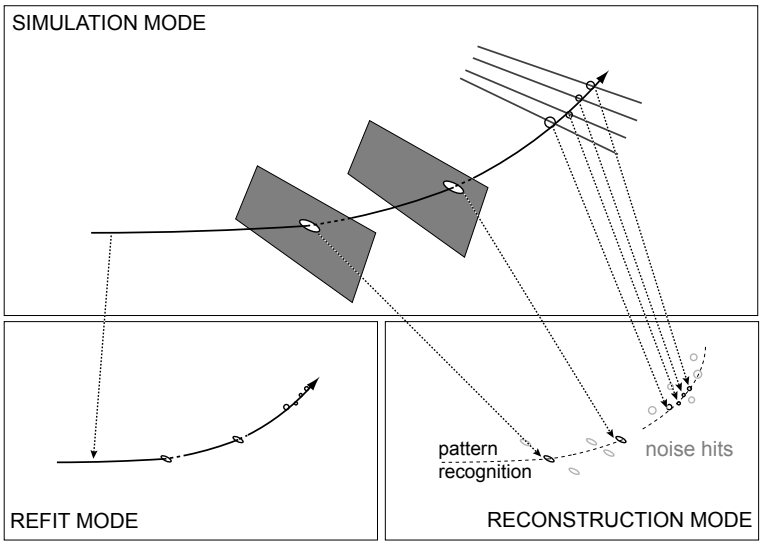


Figure 4.2: Simplified illustration of the three different Fatras modes: the *simulation* mode creates only the truth tracks that are then further processed in the *refit* and *reconstruction* mode, respectively.

4.1.3 The Fatras Event Data Model

The Fatras Event Data Model is identical to a large extent with the tracking EDM of the ATLAS offline reconstruction. Non-calibrated and calibrated measurements are expressed by the associated offline EDM classes. The use of the extrapolation engine for the trajectory creation and the common track fitting tools produces track objects that are compatible to those from the offline reconstruction. This aspect has several advantages for the further event analysis: many standard validation tools, but also the ATLAS event display applications, such as ATLANTIS or VP1 [42] are able to use output from Fatras directly.

EDM Extensions The polymorphic structure of the ATLAS tracking EDM allows to create generic hit cluster objects to be used by the Fatras simulation. This is in particular interesting

[1]The default keys used for retrieving these collections from the transient event store are *FatrasTracks*, *RefittedFatrasTracks* and *ReconstructedFatrasTracks*, respectively.

when using Fatras in design studies for future upgrade scenarios of the ATLAS tracker. A generic silicon cluster class that inherits from the common ATLAS `PrepRawData` base class and the associated calibrated version that implements the `RIO_OnTrack` interface can be found in the FatrasEvent package. The integration of these custom classes as direct extensions of the ATLAS tracking EDM allows for the use of common ATLAS tracking tools, such as track and vertex fitters on tracks that originate from modified detector setups without any intervention on the standard ATLAS reconstruction chain. The full extension of Fatras to satisfy the needs of tracker upgrade studies require also an updated detector model. This is made possible by the design of the `TrackingGeometry` that provides generic geometry classes independent from any given detector technology.

4.2 The Fatras Simulation Modules

In Section 4.1 of this document, a brief overview of the Fatras concepts and modules has been presented, including a description of the main algorithmic sequence that builds the simulation chain. In this section, details on the individual modules are given and set in context with the full detector simulation.

4.2.1 Primary Simulation

The primary simulation is the first module in the Fatras `Algorithm` chain. It is realised by one of two different `Algorithm` classes that can be chosen in the job configuration:

SingleTrackSimulation

This `Algorithm` offers the possibility to simulate user defined single track events with different particle types and kinematic input variables. In general, randomly generated input parameters are used, but the `SingleTrackSimulation` can also be executed in a scan mode as well as with fully configured track parameters for debugging single tracks which show unexpected or faulty behaviour in the offline reconstruction or the track fitting modules.

GenEventSimulation

The `GenEventSimulation Algorithm` interfaces the output from any given Monte Carlo generator to the fast track simulation and thus enables Fatras to be used in more complex physics event studies. The generated particles are hereby sorted into stable particles, decaying particles, photons or neutral particles — each type filled into a dedicated collection for the further processing in follow-up modules of the fast track simulation.

4.2.2 Particle Decay

In general, most of the relevant particle decays which are contributing to the final state signature of an ATLAS event happen inside the beampipe volume and are determined by the event generator. However, a non-negligible amount of mesons will produce a decay vertex inside the inner detector and can therefore be a source for additional leptons and light-quark jets faking

b-jets. These decays have to be handled by detector simulation frameworks as they give rise to additional tracks in the ATLAS Inner Detector and energy deposition in the calorimeters.

FATRAS provides two algorithms with dedicated `AlgTool`s to take care of particle decays. They both share the main principles of how the lifetime and therefore the path-length of the decaying particle's trajectory are calculated.

Based on the lifetime τ of the particle, the decay length λ is simulated by throwing a uniformly distributed random number $\xi \in [0, 1)$, according to

$$\lambda = c \cdot (\beta\gamma) \cdot (-\log \xi), \tag{4.1}$$

where c denotes the speed of light and $\beta\gamma$ yields the boost back into the lab system.

The trajectory of a charged particle is approximated by a helix to obtain the decay vertex position. If it is outside the inner detector volume, the particle is added to the collection to be processed by the primary track creation algorithm later on. Neutral particles are presumed to take a non-bended path through the detector. In case they are long-lived enough to decay outside the tracker, they are directly handed over to the exit state creation.

Further processing of the remaining particles is performed by the dedicated `AlgTool`s implementing the `IParticleDecay` interface. This is where the actual decay happens and its kinematics are determined. Details are discussed in the context of the algorithms using them which are discussed below. All tracks are created by an `AlgTool` implementing the `ITrackCreator` interface. By default, it is taken from the primary track simulation. It is used to extrapolate the trajectory to the decay vertex and to create a `Trk::Track` object from simulated detector hits.

ParticleDecay

The `ParticleDecay Algorithm` is part of the FatrasAlgs package. Only a limited number of particles and decay channels are supported, focusing on processes that are important for tracking studies (i.e. π_0, $K^{\pm}$ and $K^0_{S/L}$). The `ParticleDecayCreator AlgTool` provided by the FatrasTools package is handling the decays of the particles already mentioned.

G4ParticleDecay

The `G4ParticleDecay Algorithm` is located inside the FatrasG4Algs package. It provides an interface to the Geant4 particle decay classes which are able to handle all particles of the standard model by default. The `PDGToG4Particle AlgTool` is in charge of storing information such as the mean lifetime, charge and branching ratios of decay channels which is also used by the `G4ParticleDecayCreator AlgTool` to select a decay mode and calculate the corresponding kinematics.

4.2.3 Track Creation

The track creation in Fatras is done in two steps: the first one marks the trajectory building and is carried out by the extrapolation engine using the reconstruction geometry. The ATLAS `TrackingGeometry` is characterised by a navigation model using neighbouring volumes that are attached at shared surfaces. The confining surfaces of the `TrackingVolume` class (the

main components of the `TrackingGeometry`) extend the common surface description that the ATLAS Event Data Model is based on and can therefore be used with the extrapolation engine directly. In this way, the trajectory can be followed through the detector, since every boundary surface leads directly to the next detector volume traversed by the particle when being intersected. The various volumes contain layer objects that carry a material description and/or a sub-array of sensitive detection surfaces. A simple binning scheme links the intersection position with a layer to the associated detector element and consequently, the trajectory of hits can be built by passing one layer after the other — always guided from one volume to the next by the internal navigation tree. Material effects, such as ionisation loss, radiation loss or multiple Coulomb scattering are taken into account during the trajectory building. This is sped up by exchanging the stochastic material effects treatment as used in the track reconstruction with custom Monte Carlo-based algorithms. A detailed description of the material effects integration can be found in a dedicated subsection below.

The second part of the track creation is the conversion of the given trajectory into a track object. This involves cluster creation on the one hand and applying efficiency tuning on the other hand. The Fatras hit cluster creation model is described in more detail later on.

Material Effects integration

The simulation of interactions between the traversing particle and the detector material is essential for any track simulation engine. This could be done at several complexity levels which quickly lead to a high calculation complexity[2]. As a general rule, a sequential simulation is always more accurate than a fully parametric one. The individual step intervals in which an iterative simulation is performed regulates both accuracy and computational complexity. For a fast track simulation it is thus of particular interest to find a good compromise between a reasonably accurate description and the time spent for simulating it. In Fatras, the material effects integration has been optimised to be coherent with structures in main detector components (such as silicon layers, support structures). Within such a component, the corrections to the trajectory are applied as a single action. In the reconstruction geometry, all of these detector components are described as layer objects with associated material descriptions, this update mechanism is thus often called *layer-based* or *point-like*.

In Figure 4.3 a comparison of the overall material distribution for the simulation geometry based on Geant4 and the `TrackingGeometry` that is used by Fatras is shown.

Simulation of Multiple Scattering When a particle traverses detector material it is subject to multiple small angle deflections caused by the electromagnetic field of the nuclei inside the detector material. Following the central limit theorem, the distribution of the sum of these small deflections — the multiple scattering process — can be approximated by a Gaussian probability density function (PDF). However, single large angle scattering processes disturb the purely Gaussian character of the scattering distribution. In Fatras, multiple scattering can be applied in two ways that are based on two methods provided by the `MultipleScatteringUpdator` that is part of the extrapolation package: the first possibility is a purely Gaussian approximation

[2]The most realistic description of these effects would require the simulation of single atom interactions with the particle and can not be carried out in any high energy physics simulation engine.

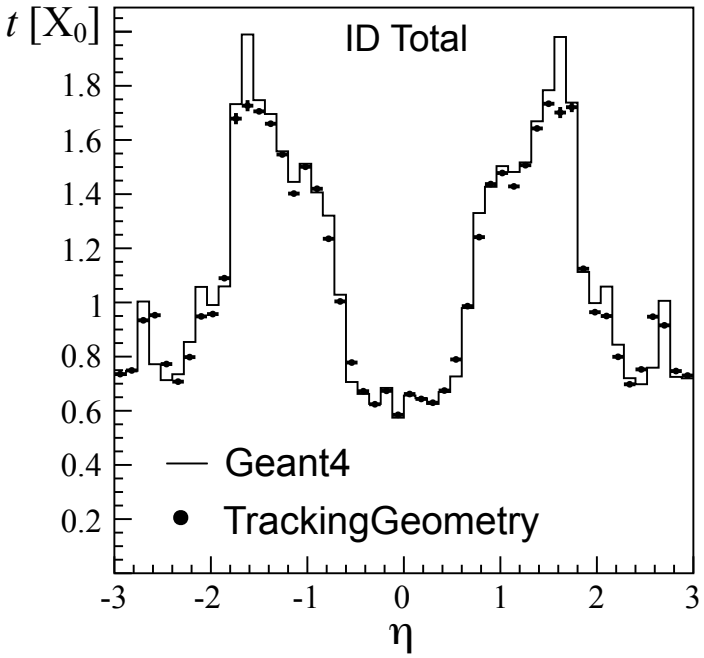


Figure 4.3: Comparison of the material budget described by the Geant4 simulation geometry and the `TrackingGeometry` description in terms of total path length in units of radiation lengths.

expressed by the Gluckstern formula (see [43], also [44])

$$\sigma_{ms}^{proj} = \frac{13.6\,\mathrm{MeV}}{\beta cp}\sqrt{t}\,(1 + 0.038\ln t), \tag{4.2}$$

with t as the covered path length in units of X_0. $\beta = \frac{v}{c}$ as usual. For many applications — and in particular for usage in a fast track simulation to validate the reconstruction software — it is of particular interest to simulate tail effects. Fatras therefore offers a second model for the integration of multiple scattering in the trajectory building process that is able to simulate parts of the tail distribution. This is done by using a Gaussian mixture model (see [39] for details).

Energy Loss Simulation The energy loss of particles traversing detector material is mainly caused by electromagnetic processes. For particles heavier than electrons, the energy loss ΔE due to ionisation is the by far dominating process. It is described by the theory of Bethe-Bloch [45] and follows a asymmetric probability density function defined by the integral equation of the Landau distribution (see also [46])

$$\rho(\lambda) = \frac{1}{2i\pi}\int_{c-i\infty}^{c+i\infty} e^{s\ln s+(\lambda)s}ds = \frac{1}{\pi}\int_{0}^{\infty} e^{-s\ln s-\lambda s}sin(\pi s)ds. \tag{4.3}$$

c is any positive real number and λ is a dimensionless number proportional to the energy loss ΔE. It can be shown that the maximum of this function, the most probable energy loss, can be written as

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\gamma^2}{I} + \ln \frac{\xi}{I} - 0.8 + 4.447 \right]. \tag{4.4} $$

with the Landau parameter $\xi = ZN_a \frac{k}{\beta^2} t$. t denotes the thickness of the traversed material, here in μm. In this context, I is the logarithmic average excitation energy and Z the atomic number of the target material. N_a is Avogadro's number and k is used for the constant of the Rutherford cross section ($k = 2.55 \times 10^{-19}$ eV per atom for single charged particles). m is the mass of the interacting particle and $\gamma = \sqrt{1-\beta^2}$.

In Fatras, the energy loss of heavy particles is sampled by a Monte Carlo-based Landau distribution using the most probable value ${}_L\Delta_p$ as given in Eq. (4.4) and a parametrised width of the distribution that has been obtained using the Geant4 simulation toolkit. Figure 4.4 shows a comparison of the energy loss distributions as given by Geant4 and Fatras for 2 GeV muons in a 250 μm silicon layer.

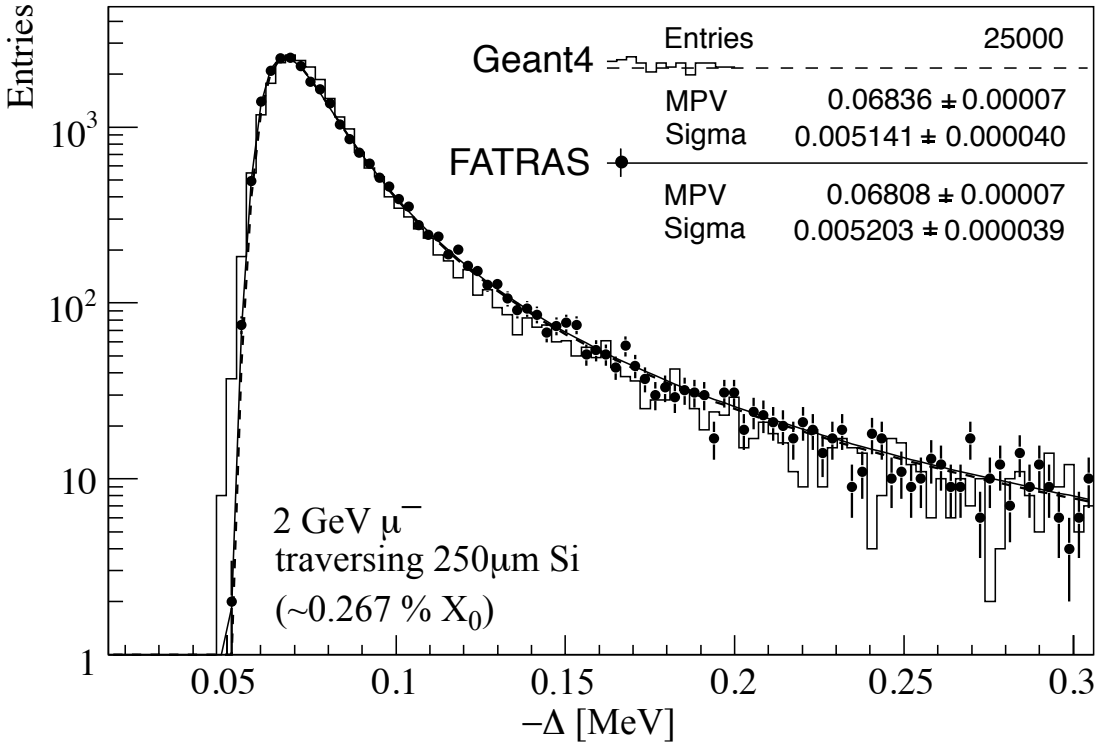


Figure 4.4: Energy loss distribution for 2 GeV muons traversing 250 μm of silicon, showing the full Geant4 simulation in comparison to the Fatras energy loss implementation. Fatras uses the Landau formula for the determination of the most probable energy loss value (MPV) and a parametrised width of the distribution that has been determined from the Geant4 simulation.

Passing through detector material, especially electrons lose in addition to the ionisation

process a significant amount of energy due to radiation loss (*bremsstrahlung*). The resulting distribution is in general a mixture between a Landau distribution due to ionisation loss and a highly asymmetric radiative contribution that results in a long tail towards high energy loss. The theory for the bremsstrahlung loss has been developed by Bethe and Heitler [47] and is only briefly described here. In the following, z denotes the ratio between the final energy E_f and the initial energy E_i. The PDF for the fraction $z \in (0, 1)$ can then be written as

$$\rho(z) = \frac{[-\ln z]^{c-1}}{\Gamma(c)}, \tag{4.5}$$

where c denotes $c = t/\ln 2$. The factor z is sampled in Fatras, which is done by a dedicated energy loss module in the extrapolation engine. Again, this is only possible since the extrapolation engine follows a strict component software pattern. Two different strategies can be chosen: the default implementation is a Monte Carlo-based sampling of the Bethe-Heitler distribution as given in Eq. (4.5), while the `GSFPDF AlgTool` models the Bethe-Heitler distribution as a sum of weighted single Gaussian distributions. It has been developed for validating the *Gaussian Sum Filter* (GSF) [48].

Photon Emission Significant loss of energy due to bremsstrahlung makes it important to take another aspect into account: the emitted high energetic photon has to be tracked through the detector volume, since it can influence the event morphology in several ways. When interacting with the detector material, this can result in lepton pair creation (mainly electron-positron) — in the following also referred to as *photon conversion* — and thus lead to additional tracks in the detector volume. If no conversion takes place, the additional photons influence the cluster signatures in the calorimeter. The handling of these photons, covering both effects — the conversion and the transport to the calorimeter volume — is in detail described in Section 4.2.4 of this document.

The Fatras bremsstrahlung model describes the emission of hard photons and their respective angle to the initial electron. The photon energy hereby corresponds to the energy loss fraction according to the Bethe-Heitler theory as given in Eq. (4.5). It can be shown that the angle of the emitted photon w.r.t. the parental electron direction is proportional to m_e/E_e. Electrons relevant for the track reconstruction in the ATLAS Inner Detector have typical momentum values that are significantly higher than the electron mass. Thus, the emitted photons are almost collinear to the original electron trajectory. Figure 4.5 shows a comparison of emitted high-energetic photons simulated by Geant4 and Fatras for electrons with a transverse momentum p_T of 15 GeV within the ID tracking acceptance region $|\eta| < 2.5$. The full detector simulation usually produces a higher number of emitted photons, mainly due to the fact that Fatras is restricted to only one iteration of photon conversion, while in Geant4 a full cascade simulation is performed. This leads to an underestimation of bremsstrahlung photons in the low momentum spectrum, which could be cured by the addition of further iterations of the algorithm. However, this has not been integrated into the current Fatras setup yet.

Hadronic Interactions The simulation of nuclear interaction between hadrons and the detector material is currently limited in Fatras. The reason is, that the reconstruction geometry does not yet provide any information about the nuclear interaction probability. An updated

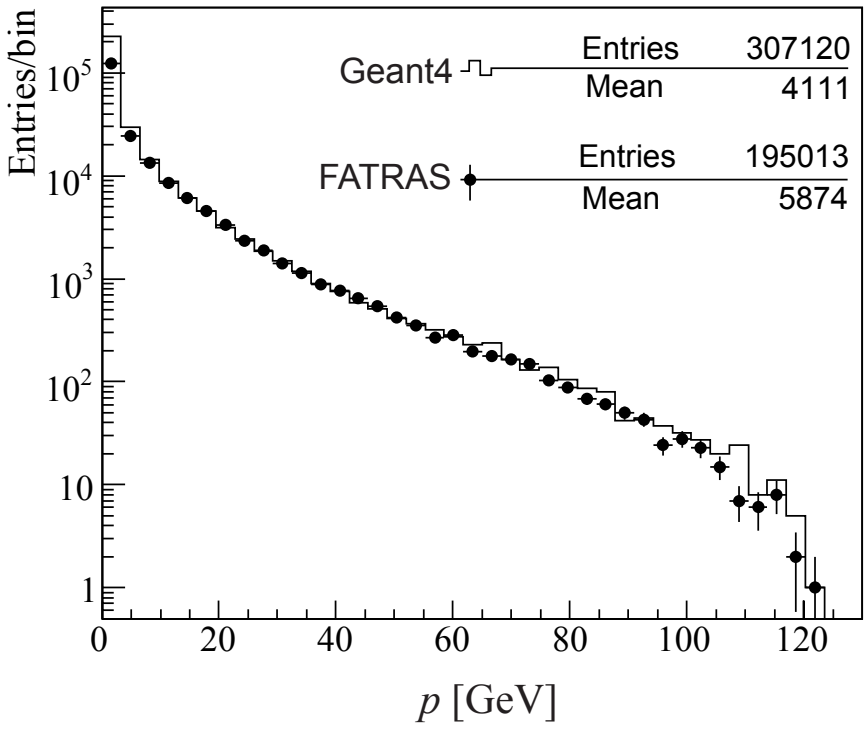


Figure 4.5: The momentum distribution of hard photons that are emitted from electrons simulated by Geant4 and Fatras. The identical input sample of 50000 single electron tracks with transverse momenta of $p_T = 15$ GeV is used. Only tracks inside $|\eta| < 2.5$ are taken into account. When restricting the electron energy to a momentum higher than 5 GeV and accounting only for photons with $p > 1$ GeV, the ratio of photons produced from Geant4 to Fatras drops in the given example from about 1.5 to 1.13. The ratio of the mean value changes from 1.42 to about 1.07.

model for this is planned to be integrated into the ATLAS software release 14.0.0. For a fast track simulation, two aspects are particularly interesting in the context of hadronic interactions: on the one hand, the dominant hadronic shower process leads very often to effectively shorter track lengths (or even no clear hadron trace in the detector at all), which influences both the track parameter resolutions and the track reconstruction efficiency. On the other hand, hadronic shower particles can penetrate into the detector and need to be followed for a successive calorimeter simulation. While the decay process ($\pi^{\pm} \rightarrow \mu^{\pm}\nu$) and the scattering process ($\pi^{\pm} \rightarrow \pi^{\pm}$) are included in the particle decay module and the multiple scattering update mechanism of Fatras, respectively, the hadronic cascade (and hence the destruction of the initial pion) has to be taken into account in a separate step. The hadronic shower model is carried out in a simplified way and is parametrised from data that has been simulated with Geant4. It includes several fit parameters and restricting assumptions which are described more detailed in [39].

However, it should be mentioned here shortly that as long as the hadronic interaction length

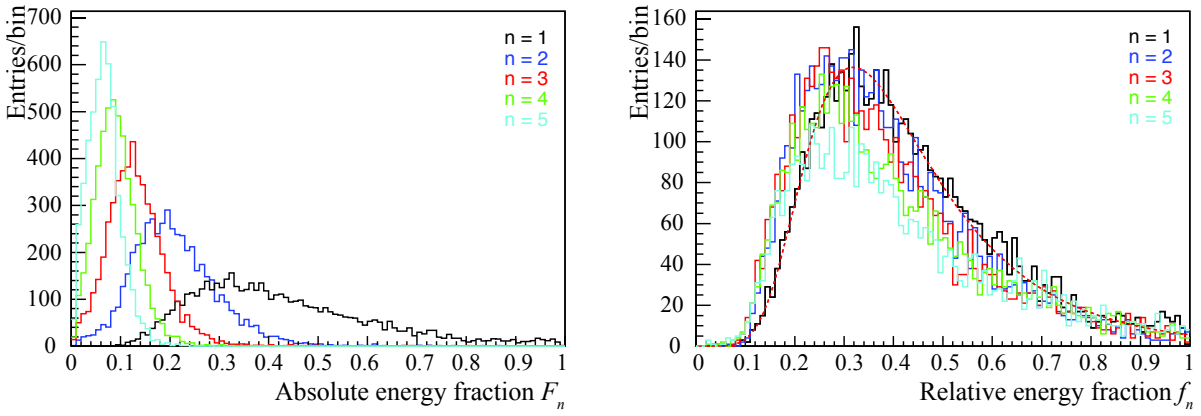


Figure 4.6: Absolute (w.r.t. the incoming particle) and relative (w.r.t. the rest energy) energy fractions for the five most-energetic particles in the hadronic cascade obtained with Geant4.

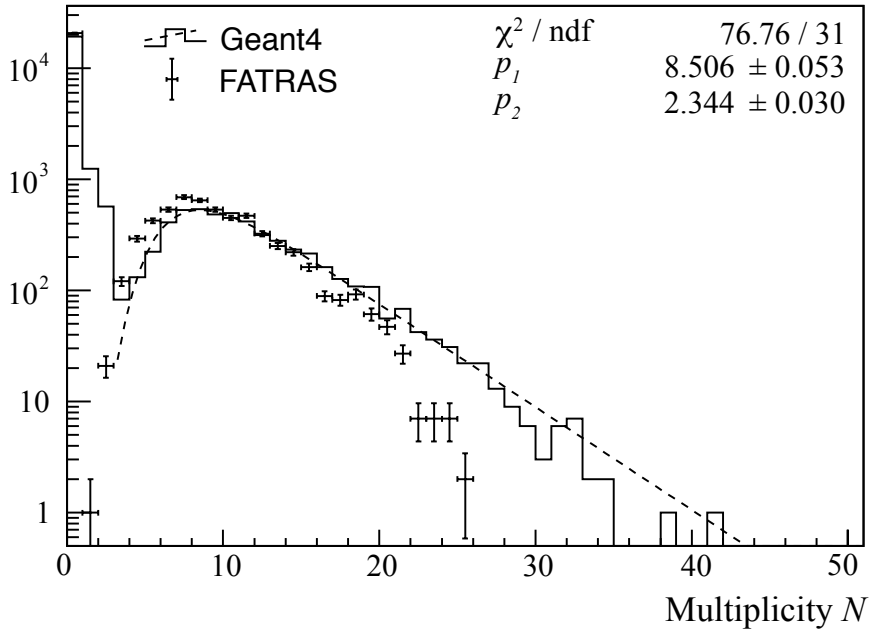


Figure 4.7: Particle multiplicity N in hadronic showers generated with Geant4 and Fatras. The fit function which is the basis for the Fatras hadronic shower model is also shown in the histogram

λ is not introduced as a material parameter in the `TrackingGeometry` description, it is ap-

proximated by

$$\lambda = 0.37 \cdot s_{had} \cdot \bar{Z} \cdot \mathrm{X}_0, \tag{4.6}$$

with s_{had} as a tunable scaling factor, $\bar{Z}$ for the average atomic number and X_0 for the radiation length of the material. This parameter is used for determining the probability of a hadronic interaction following an exponential decay law.

Finally, the particle content of the shower has to be simulated. The shower contains many different types of particles, but mainly charged and neutral pions, protons and neutrons. Only these particles are created with fractions of 33% π^0, 25% π^+, 25% π^-, 10% neutrons and 7% protons. This composition is adjusted to the shower content generated by Geant4 and can be modified as tuning parameters. In Figure 4.6, the energy distribution of the five most-energetic particles is shown as simulated by Geant4.

Figure 4.7 illustrates the particle multiplicity in hadronic showers from nuclear interactions of the primary particle with detector material. It shows the distribution obtained with Geant4, the generic fit function used for the parametrised level and the corresponding distribution produced with Fatras.

Figures 4.8 (a) to (d) show comparisons of the absolute energy fractions for the first four most energetic particles between the Geant4 simulation and Fatras.

In general, good agreement in the distributions generated by the parametrised shower in Fatras and the full simulation can be observed.

Track Creation from a given Trajectory

The extrapolation engine is not capable of producing simulated tracks in the ATLAS reconstruction geometry, it only provides a trajectory of track parameters on sensitive detector elements. This trajectory has to be transformed into a `Track` object of the ATLAS tracking EDM. The `TrackCreator AlgTool` is responsible for performing this task. It calls the extrapolation engine and dissolves the returned trajectory according to the single surface intersections. Since the `Surface` object of the ATLAS reconstruction geometry is linked to the underlying readout element of the full ATLAS detector description, it is possible to identify the detector type and apply the appropriate hit cluster creation and smearing. The track parameter information is used to create standard ATLAS tracking EDM objects, the non-calibrated `PrepRawData` and the calibrated `RIO_OnTrack` objects representing the simulated detector measurements. Inefficiencies can be applied by introducing efficiency parameters for the individual subdetectors.

Hit Cluster Creation and Simplified Smearing A correct cluster creation description is essential for obtaining realistic track parameter resolutions. It can be shown, that the impact parameter resolutions are dominated by the measurements in the innermost detection layers[3], while the momentum resolution depends on the entire hit collection of the track. The model for cluster creation in the pixel detector has therefore to be the most accurate, since its influence — in particular the cluster sizes and shapes of the B-layer measurement — on the track parametrisation is the most significant one.

[3]This can easily be shown by assuming a simplified two-layer detector model.

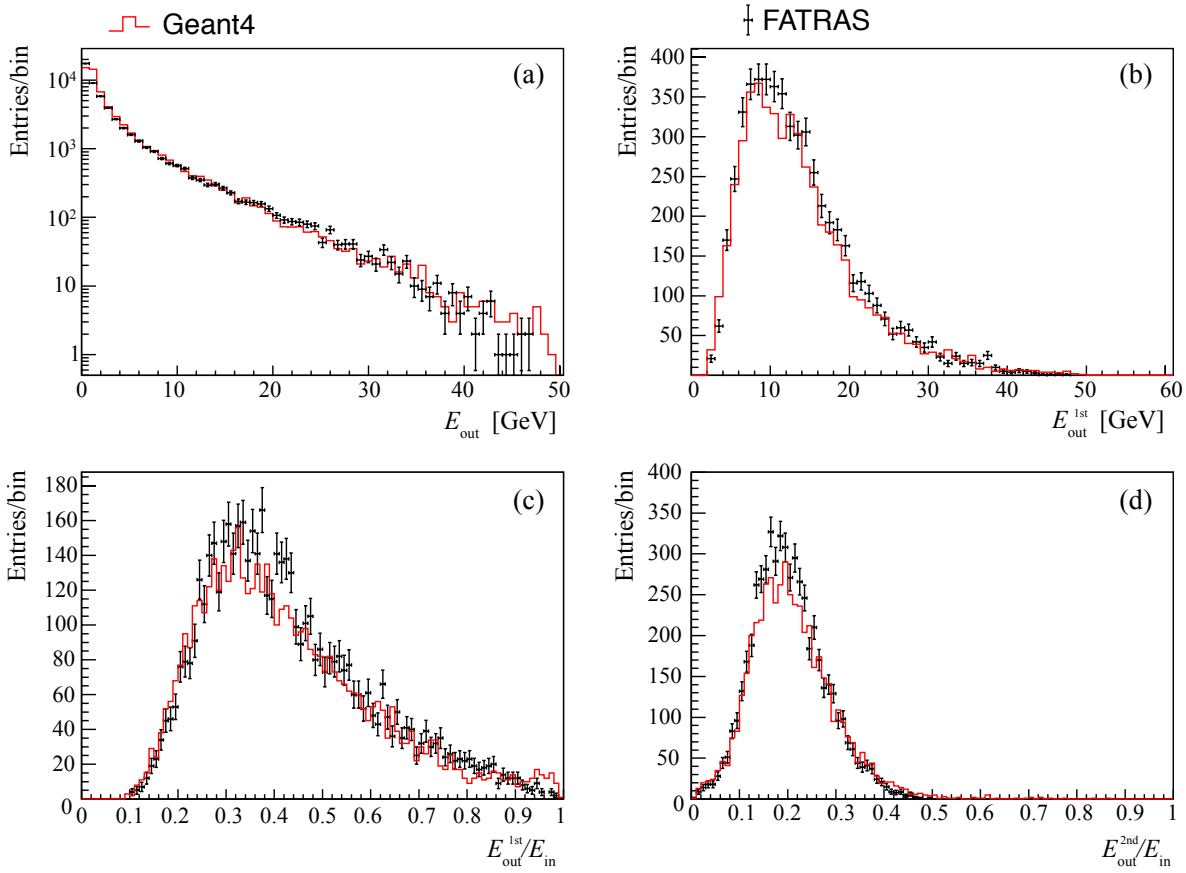


Figure 4.8: Comparison of hadronic shower particle energies in Fatras and Geant4: (a) energy of all shower particles, (b) energy of of the most energetic shower particles, (c) and (d) relative energy fractions of the first and second most energetic shower particles.

In the ATLAS offline reconstruction, the pixel cluster creation algorithm uses the analog readout information to determine the cluster position with the help of the time-over-threshold (ToT) information. This technique yields a higher resolution than the intrinsic single pixel resolution, but requires the charge deposition distribution to be known, which is either provided through the readout when reconstructing real data or simulated in the full detector simulation. In Fatras, the charge deposition in sensitive detector material is not simulated down to this level of detail, since only energy loss is applied to the particle when traversing detector material. A geometrical approach to cluster creation is used instead to create similar cluster sizes and shapes as provided by the full detector simulation and digitisation. The geometrical cluster creation uses the path length of the track in the individual intersected pixels to calculate the cluster position from the centre positions of the associated pixel cells. Given n geometrically intersected pixels with s_i being the respective path length in pixel i, the cluster position $\vec{p}$ is

calculated as

$$\vec{p} = \frac{1}{\sum_{i=0}^{n} \Theta(s_i - s_{cut}) \cdot s_i} \sum_{i=0}^{n} \Theta(s_i - s_{cut}) \cdot s_i \cdot \vec{p}_i, \tag{4.7}$$

where the $\vec{p}_i$ denote the individual centre positions of the intersected pixels and the Heaviside function Θ is inserted to demand a minimal path length of the track within each silicon pixel, which is equivalent to a minimal charge deposition per pixel. s_{cut} denotes the only model parameter of the pixel cluster creation. Figure 4.9 shows an illustration of the geometrical cluster creation approach.

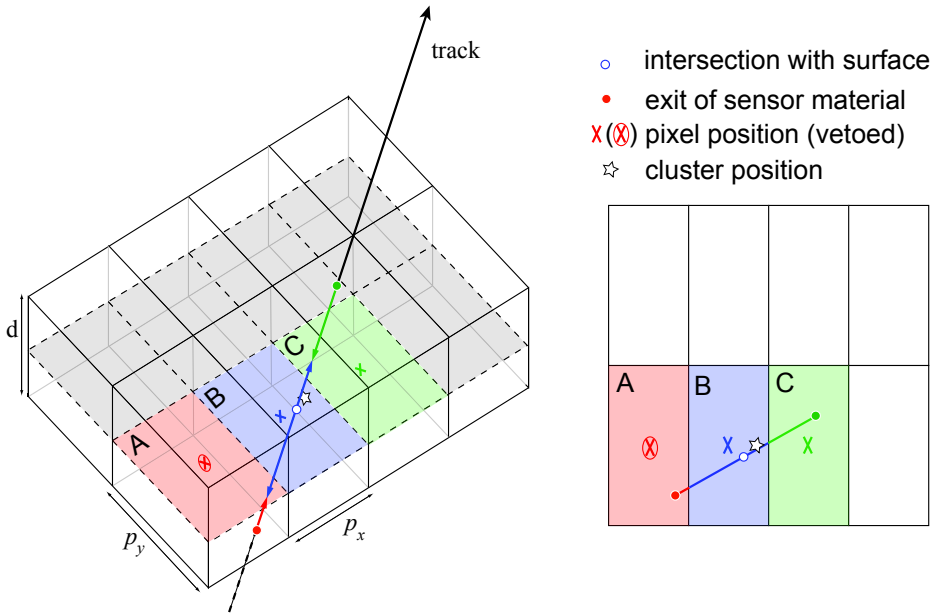


Figure 4.9: The geometrical cluster creation model for the pixel detector that is used in the Fatras simulation. Analog cluster creation is hereby performed by weighting the centre positions of intersected pixels with the track distance inside the pixel. Silicon pixels that are traversed by the track, but do not host a sufficiently long path length for the pixel to detect a signal, are vetoed for the cluster forming process (pixel A).

The cluster creation for the silicon strip detector is for both the offline realisation and in the simplified Fatras model based on a digital approach, accounting for only the intersected strips to build the final cluster position. This results in one-, two- and three-strip clusters, without using any charge deposition information. For the Transition Radiation Tracker, the hit creation in Fatras is simply done by Gaussian smearing of the intersection path, since the cluster shapes in the TRT are in general not very complex, but roughly compatible with a Gaussian distribution of the drift radius. Since the drift radius error varies with the drift time, this information is retrieved from the offline drift function tools. An additional tail contribution can be added to account for non-Gaussian effects and a scaling parameter is available to regulate the momentum resolution.

The hit creation — carried out by the `ClusterCreator AlgTool` — can also be performed with purely Gaussian smearing for all detector technologies. This does not lead to track

parameter resolutions that are comparable to the tracks found by the offline reconstruction, but is very useful for using Fatras as a validation tool, in particular when testing the track fitting modules.

4.2.4 Photon Processing

The transport of electrons through the detector causes in many cases the emission of a high-energetic photon. Since these photons create either an electromagnetic shower in the calorimeter or convert into (mainly) electron-positron pairs when interacting with detector material, it is important to follow these processes in the track simulation. In Fatras, the photons from final state radiation or emitted by electrons are collected in one `ParticleState` container and further processed by a dedicated algorithm, the `PhotonProcessing`. Each photon is propagated through the detector, following a straight line model. A specially configured instance of the `Extrapolator` is used to perform this operation. A specific `IMaterialEffectsUpdator` implementation, the `McConversionCreator`, simulates the conversion of the photon into an electron-positron pair, depending on the amount of material when a layer is crossed[4].

Figure 4.10 shows an example photon conversion event simulated with Fatras and displayed with the ATLANTIS event display. It presents in addition a conversion vertices map in the $R - z$ projection of the Inner Detector using Geant4 simulation and Fatras. The simplified geometry used in Fatras is mirrored by the the discrete distribution of conversion points in the TRT detector, which is modeled as a simple set of several cylinders and discs in the `TrackingGeometry`, while a continuous material distribution in the full detector geometry used by Geant4 can be identified.

Figure 4.11 shows a comparison of simulated electron energies for particles originating from photon conversions in the ATLAS ID and the dependence of the mean electron energy on the transverse photon momentum.

When comparing a full physics event, Fatras underestimates the number of bremsstrahlung photons and photon conversions since only a limited number of iterations of the entire cascade are carried out. For the simplest possible Fatras configuration that only has one iteration, this means that an electron which comes from a pair production process can radiate a high energetic photon. This photon is, however, not tracked further in terms of conversion creation. It is, together with photons that are not converted into a child pair at first place, filled into the appropriate containers for the exit state creation (see Section 4.2.6).

4.2.5 Track Refitting

The track refitting module has to be performed only when Fatras is used in *refit* mode. In this case, the refit of the simulated track is necessary to remove the truth bias and to result in an adequate track resolution compared to the full simulation and offline reconstruction chain. The simulated track is hereby kept as a reference or truth track for validation purposes. The refit of an entire track is in general performed such that the perigee representation is used as the starting point of the track. Since in Fatras the perigee associated to a track is the true origin of the trajectory, this would introduce a strong bias towards artificially high track resolution

[4]The simulation of $\mu^-\mu^+$ pair production is omitted in Fatras since it is by orders of magnitudes less probable than for the electron-positron case.

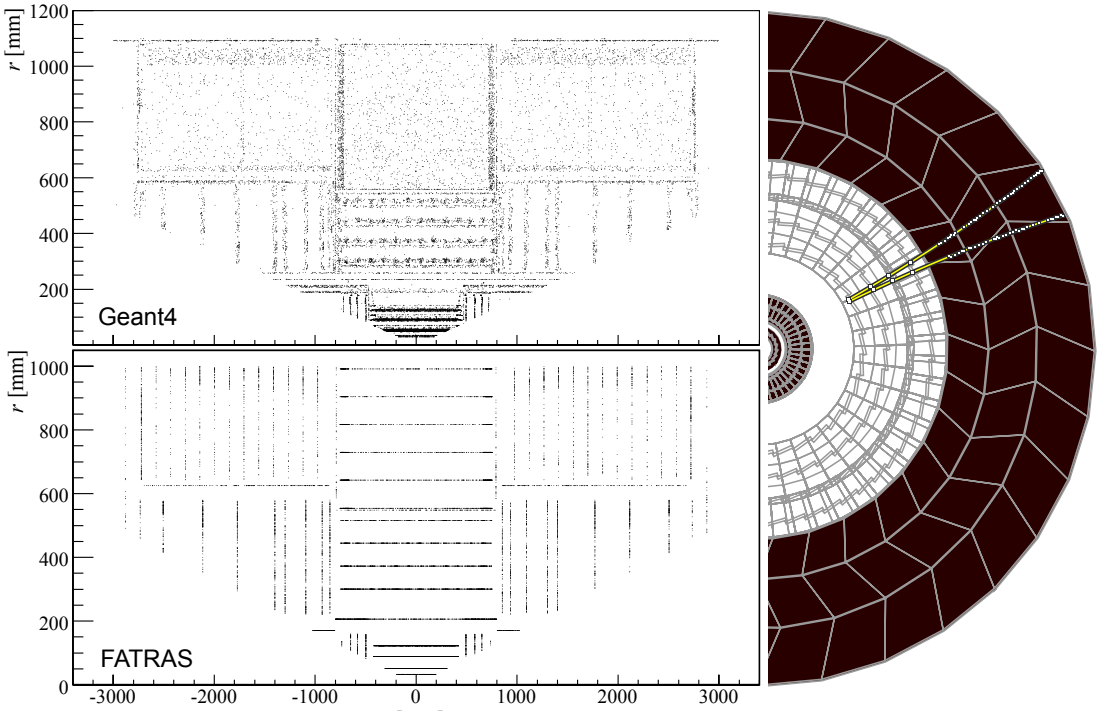


Figure 4.10: Photon conversions in the ATLAS Inner Detector, simulated with Geant4 and Fatras. The simplified simulation geometry of Fatras can be seen, which is limited to several discrete layers, while the Geant4 simulation geometry is more detailed. The picture to the right shows a photon conversion before the first SCT barrel layer, simulated with Fatras and shown with the ATLAS event display ATLANTIS.

on the first measurement layer. The special `TrackRefitting Algorithm` that is provided by the FatrasAlgs package allows therefore to smear the initial perigee representation.

4.2.6 Exit State Creation

In Fatras, every particle — independent of its origin or creation process — that has not been decayed within the detector volume is transported to the boundary of the volume for further processing in a subsequent detector part, e.g. for future integration with both the full or fast calorimeter simulation, as well as a future extension of Fatras for the Muon Spectrometer.

4.2.7 Post Processing and Noise Level Creation

An additional level of post processing is necessary in Fatras to achieve full compatibility with the offline reconstruction. In the pure refit mode, the tracks have to be processed before refitting to simulate pattern recognition effects such as holes on a track or missed extensions from one subdetector to another[5]. On the other hand, when running in reconstruction mode,

[5]The standard ATLAS reconstruction employs an inside-out procedure for the pattern recognition that starts off in the innermost silicon layers and extends a successful silicon track segment to the TRT straw detector

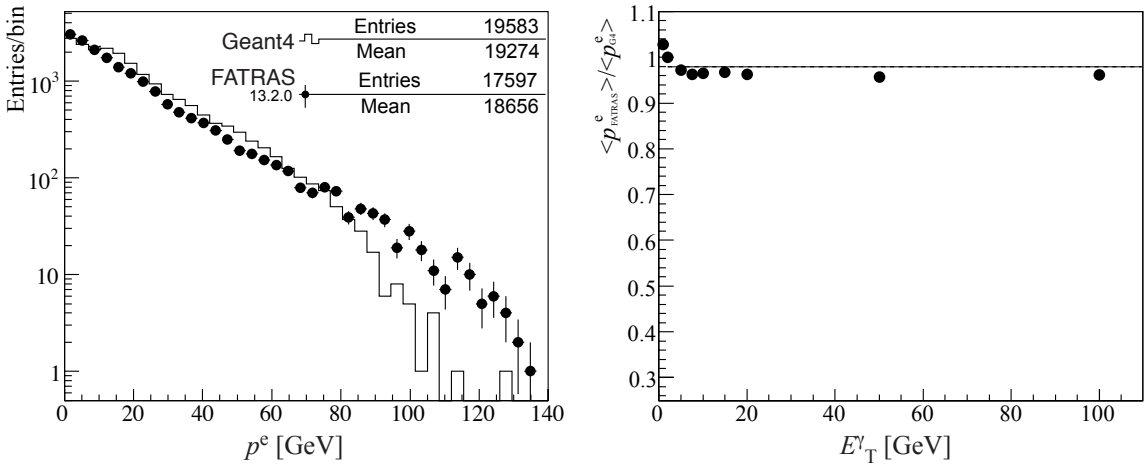


Figure 4.11: Comparison (Geant4/Fatras) of the electron energy distribution originating from photon conversions in the ATLAS ID for photons with an initial transverse energy of $E_T^\gamma = 15$ GeV (left). The right plot shows the ratio of the mean child electron momentum $< p^e_{Fatras} > / < p^e_{G4} >$ from photon conversions in the ID for photons with various fixed transverse momenta.

a special post processing in terms of hit correction is not necessary. The different topologies originating from effects in the pattern recognition process are produced in the reconstruction mode just naturally. Another aspect has to be considered though, namely the creation of *fake* hits that originate in reality from electronic noise or low energetic secondary particles that are created when the primary particles traverse the detector material. Finally, hit containers and noise creation have to be handled: during the hit post processing, hits are stripped from the simulated tracks and filled into the same hit collections that are used in the offline reconstruction chain. During this operation, noise hits can be added independently to every subdetector data collection and the noise levels can be adjusted freely by the user.

at larger radii.

5 Key Extensions for Upgrade Simulation Studies

This chapter contains an extensive description of the modules which were designed and implemented within the framework of this diploma thesis in order to render upgrade simulation studies possible with Fatras.

5.1 Fatras G4ParticleDecay AlgTool

The `G4ParticleDecay Algorithm` supersedes the formerly used `ParticleDecay Algorithm` which implements provisional decays for pions and kaons only. In contrast to that, the `G4ParticleDecay` class uses a more generic interface to particle descriptions and decay properties from the Geant4 simulation framework [1]. The advantage of this approach is that there already exist definitions for the most common unstable particles (e.g π and K mesons, Λ and Σ baryons). Additionally, predefined templates for various decay types can be used (e.g. mass-dependent phase-space decay, form-factor dependent kaon-style decay etc.) in order to define further decay channels, also for newly defined particles, i.e. SUSY particles. All this information is stored by an instance of the `PDGToG4Particle AlgTool`.

When an unstable particle is encountered in Fatras, it is put in a dedicated collection of particles, the `FatrasDecayingStates` collection. The primary particle decay `Algorithm` is executed after the Monte Carlo event input collection is either read from a file or generated on the fly by the `SingleTrackSimulation Algorithm`. One by one, the particles to be decayed are read from the collection and an iteration is started. It is important to note, that the input states are pushed on a stack container where also the decay products are stored after a decay. For this reason, particles from the stack are checked again for stability and forwarded to the primary track creation if this is the case. After some basic checks on the validity of the input, the average lifetime is retrieved from the local `PDGToG4Particle` instance and used for the determination of the decay length (as was described in Section 4.2.2). Depending on the charge of the particle, a discrimination is made between neutral particles, whose trajectories are extrapolated in a straight line, and charged particles. The trajectory of the latter in a magnetic field can be expressed by a helix to approximate the decay position.

A helix can be described by five parameters with respect to a pivotal point. In ATLAS, the closest point of approach to the beamline is most commonly used which is called the *perigee*. The helix parameters are directly linked to the five perigee parameters which also completely define a track and are defined in Figure 5.1:

$$\vec{q} = (d_0, z_0, \phi, \theta, q/p)^T. \tag{5.1}$$

In the ATLAS detector the field lines of the tracking magnetic field are fundamentally collinear to the beamline. For this reason, only the x and y components of a charged particle's

[1] It is important to stress that only particle properties and specialized containers from Geant4 are used, not the simulation itself.

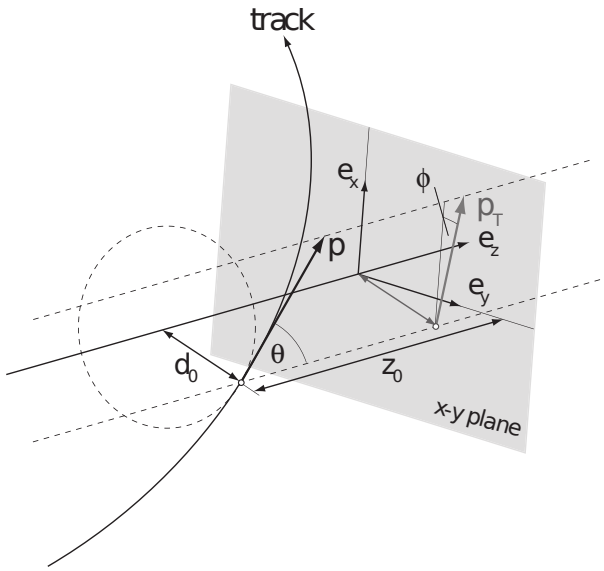


Figure 5.1: The point of closest approach to the beamline of a track, also called the *perigee*, with its defining parameters [36].

momentum vector are changing over time while it traverses the Inner Detector volume. It is possible to derive the helix parameters from the perigee and the magnetic field vector, which is done by the class `TrkExHelix` from the TrkExtrapolation/TrkExUtils package.

In this way it is determined whether the estimated decay vertex of the particles lies inside the Tracker volume and a decay has to be computed, or the particle leaves the Inner Detector without decaying. In any case, the `Extrapolator` from the Fatras primary track creation is used, which means that on each material layer, the default Fatras material interactions are considered.

The actual decay is calculated by the `G4ParticleDecayCreator AlgTool`. The parent particle is looked up in the same particle property table as was used in the `Algorithm` for the retrieval of decay channels. A list of the preconfigured decays is to be found in Appendix A.

5.2 The Fatras GenericGeometry Extension

To permit loading a custom geometry into the Inner Detector simulation, a `TrackingGeometry` has to be generated from geometric data. Normally, this is done by requesting the details of the ATLAS geometry from a remote database at CERN which contains details on material composition, position and alignment of all detector parts.

For the Tracker upgrade, this is neither possible nor feasible: the different layout proposals are far from fixed and details about the detector layers (e.g. material distributions on a mesoscopic level) are largely unknown. The `GenericGeometry` extension enables the user to specify a layout on a general level. Only two types of detector layers are currently supported: barrel-shaped configurations for the central part of the detector and disk-like structures for the endcaps. Each layer can be assembled from detector modules, for which rectangular and trapezoidal shapes are available.

5.2.1 Geometry Definition

The geometry description is defined in a Python script file which is loaded at Athena start-up and specified by a Fatras `JobProperty` in a loaded `jobOptions` script. As an example, the geometry file for the quasi-projective Strawman layout can be found in Appendix B.

The geometry description file normally starts with the definition of the materials to be used for each detector layer. The thickness, radiation length X_0, mass number A, atomic number Z and density ρ are mandatory parameters, combined in the `GenericMaterialDescriptor`.

Sensor modules are the basic building blocks of the tracking geometry. They come in two flavours described by the `GenericRectangularModule` and `GenericTrapezoidalModule` structures. In the respective Figures 5.2a and 5.2b, the defining quantities are shown. The *thickness* parameter, which is especially relevant for the hit creation, is not displayed in the figures.

A distinction between pixel-type and strip-type sensors is made by passing either `GenericPixel` or `GenericStrip` as the integral *clusterType* argument. It has to be noted that strips do not have to be as long as the whole module. By choosing a strip length smaller then the module pitch in the corresponding direction, a segmentation will be automatically conducted. This is especially useful when combining several short strips to one module.

After the modules have been defined, the endcap rings can be instantiated. Specific to each instance of the class `GenericEndcapRing` is the *module* type, the number of sections in ϕ (*sectorsPhi*), the inner and outer radii (*innerR* and *outerR*) and a *rotation* parameter. The last argument is necessary to align the rings correctly across the entire disc.

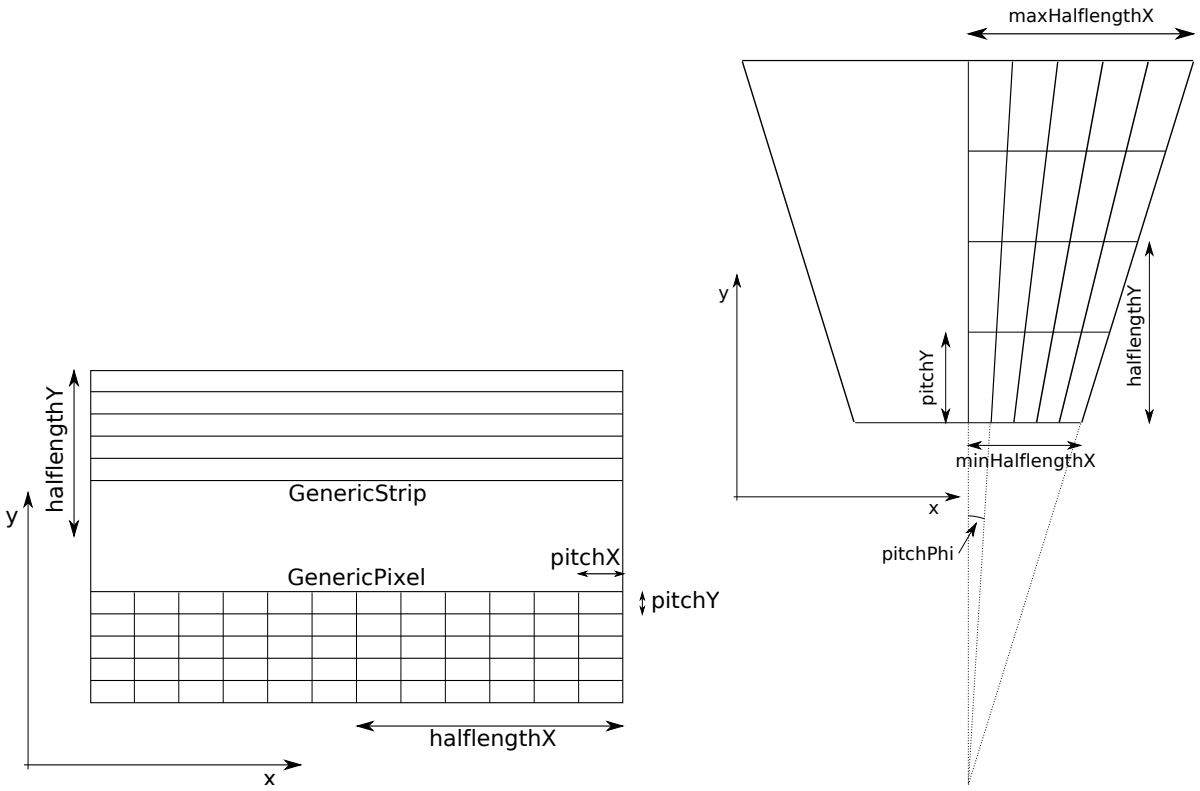
Now, that all the basic entities necessary to build the different detector layers have been created, the `GenericBarrelLayer` and the `GenericEndcapDisc` can be formed at specific positions (*radius* and *zPos* respectively). Both of them take a *material* argument (i.e. a `GenericMaterialDescriptor` instance created earlier). Due to the different structure of barrel and endcap layers, the remaining steps differ largely.

For the `GenericBarrelLayer`, the *halflength* in z-direction and the segmentations in ϕ and z (*sectorsPhi* and *sectorsZ*) have to be given. The modules can be tilted in ϕ by specifying a non-zero *tiltAngle*. Otherwise, their orientation is tangential to the cylinder formed by the geometric layer parameters.

The endcap discs are finalized by adding existing `GenericEndcapRing` objects with the *addRing()* function.

In a last step, the final geometry is assembled. The logical detector volumes are created by instances of type `GenericDetectorVolume`, as well as a `GenericBeamPipe` object. After adding the layers with *addBarrelLayer()* and *addEndcapDisc()* to the volumes, the latter are joined together with the beampipe to a top-level `GenericGeometry`. With the call of the *registerGeometry()* function, the structure of the geometry and all parameters are forwarded to corresponding C++ `AlgTool`s.

After the initialisation phase of the ATHENA run has finished, the actual creation of the reconstruction geometry is started by dedicated C++ classes provided by the FatrasDetDescrTools package.



(a) A schematic GenericRectangularModule (b) A schematic GenericTrapezoidalModule

Figure 5.2: The two types of modules available for custom geometry building: the (a) GenericRectangularModule and the (b) GenericTrapezoidalModule. The two available cluster types `GenericPixel` and `GenericStrip` are only shown for the rectangular module type but are as well available for the trapezoidal shape.

5.2.2 Geometry Building

The steering of the geometry building process is handled by the `GenericGeometryBuilder` class. It implements the `Trk::IGeometryBuilder` interface and replaces the default reconstruction geometry building mechanism. By calling the *trackingGeometry()* function, the assembly of the `TrackingGeometry` is initiated.

Depending on the settings, the beampipe and the subdetectors are built. For the former, the same facilities as for the standard reconstruction geometry are used. For the latter, again, the component design of ATHENA is exploited: the `GenericLayerBuilder` class uses the `Trk::ILayerBuilder` interface to provide the necessary functions for the retrieval of the cylindrical layers for the barrel and the disc layers for the endcaps confined in each subdetector volume. Each `GenericLayerBuilder` C++ object has a one-to-one correspondence to the `GenericDetectorVolume` objects on the Python side.

The remaining components necessary in the build process are the C++ representations of the previously set materials, modules and layers which are loaded by the `GenericLayerBuilder` instances for each subdetector volume.

The actual geometry creation proceeds from the beamline to the outside for each subdetector volume separately. In this process, all or some endcaps also can be omitted to build e.g. a barrel-only geometry.

A complete overview of the dependencies between the individual geometry part representations in Python and C++ is presented in Figure 5.3.

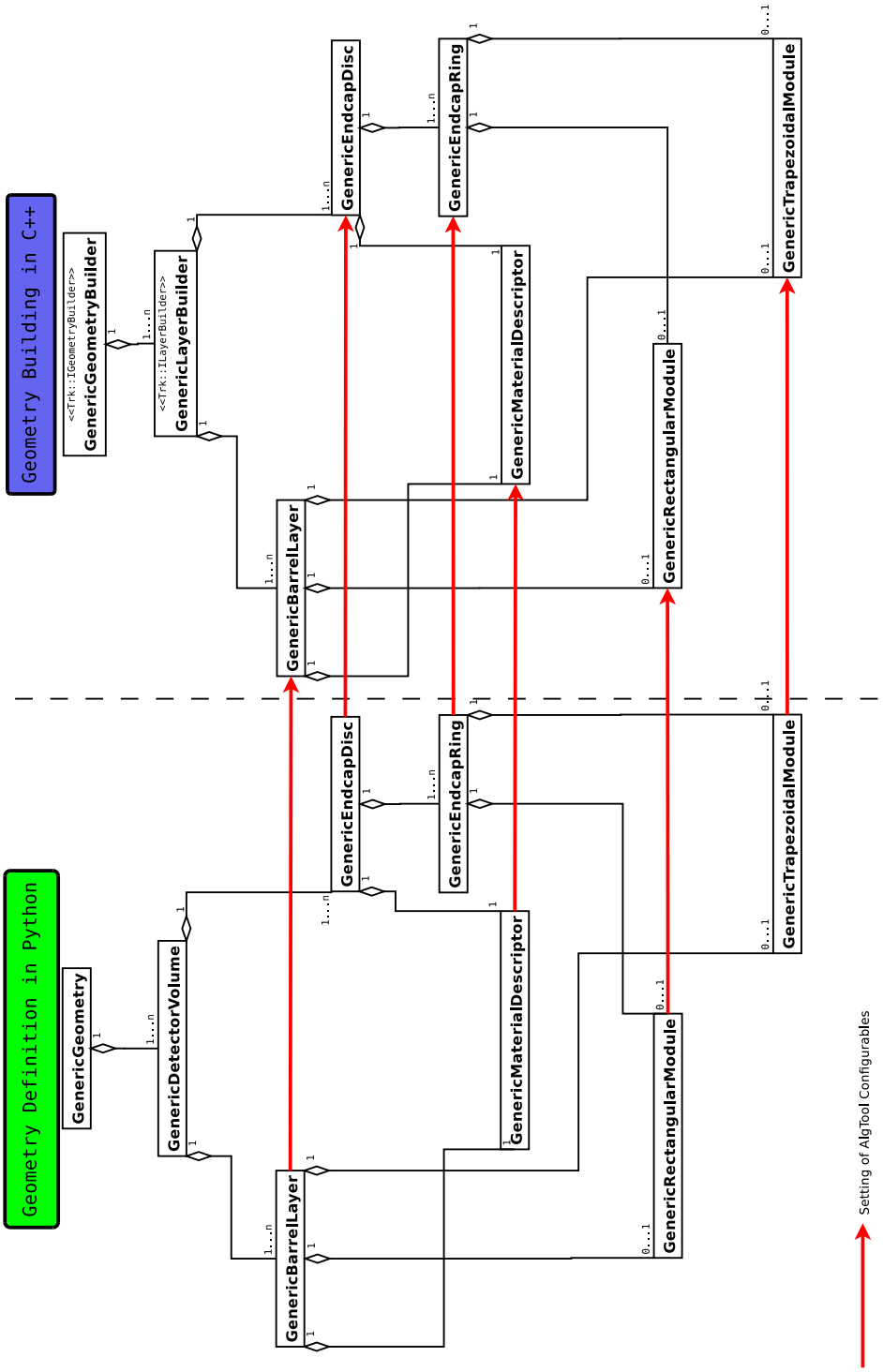


Figure 5.3: Class diagram of the `GenericGeometry` extension for Fatras. The Python classes on the left are used to set the geometry description. On the right, the structure responsible for the actual building of the geometry is shown. The data flow during the initialisation phase of ATHENA is indicated by red arrows.

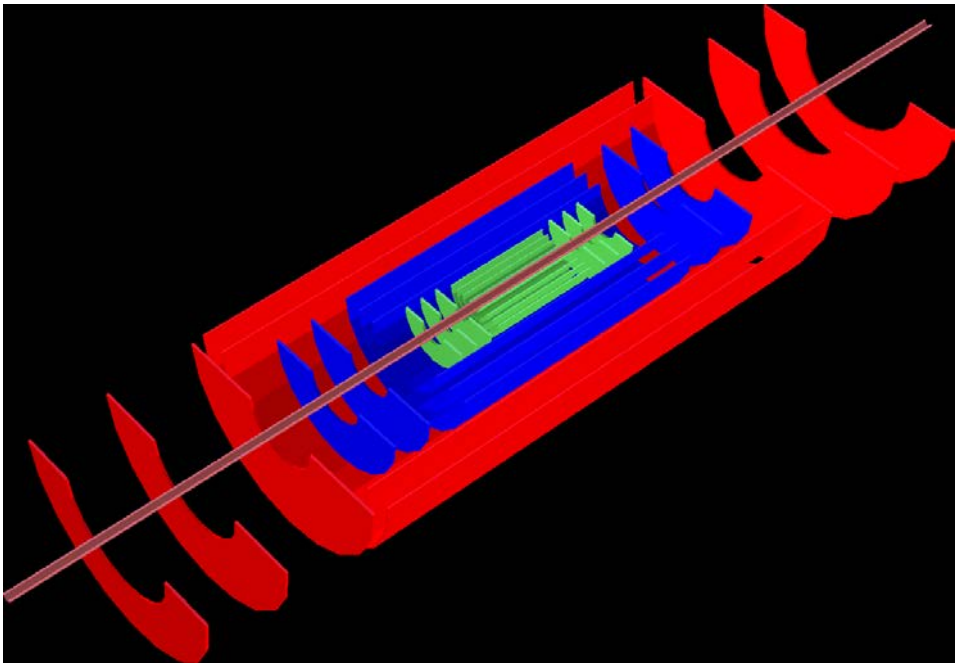
5.2.3 Geometry Validation

To test the `GenericGeometry` extension and to conduct first studies of upgrade layouts, two of the Strawman proposals have been implemented: the quasi-projective layout (Figure 2.2) and the equal-length SCT barrel layout (Figure 2.3). The files containing the geometry descriptions are provided by the FatrasDetDescrExample package.

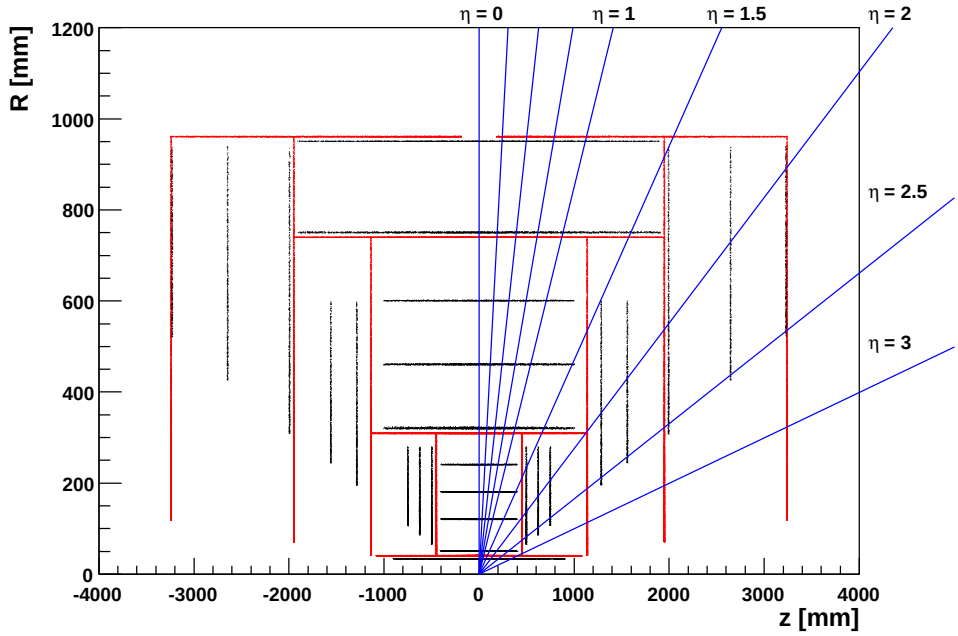
During the development of the software and the creation of the geometries, visualisation tools are indispensable. The TrkDetDescrExample package contains scripts for displaying the reconstruction geometry on volume, layer and module level separately, in a three-dimensional ROOT viewer. These files have been modified to enable the loading of the geometry description files.

While the graphical presentation of the geometry is helpful during the modelling phase, it is even more important to test its fitness in a primitive simulation setup to validate the complicated internal structure of the `TrackingGeometry`. Also here, existing scripts for the validation of the standard reconstruction geometry could be re-used and extended for the production of ROOT ntuples containing data from layer navigation tests.

In Figures 5.4 and 5.5, examples of the output from these validation facilities is displayed.

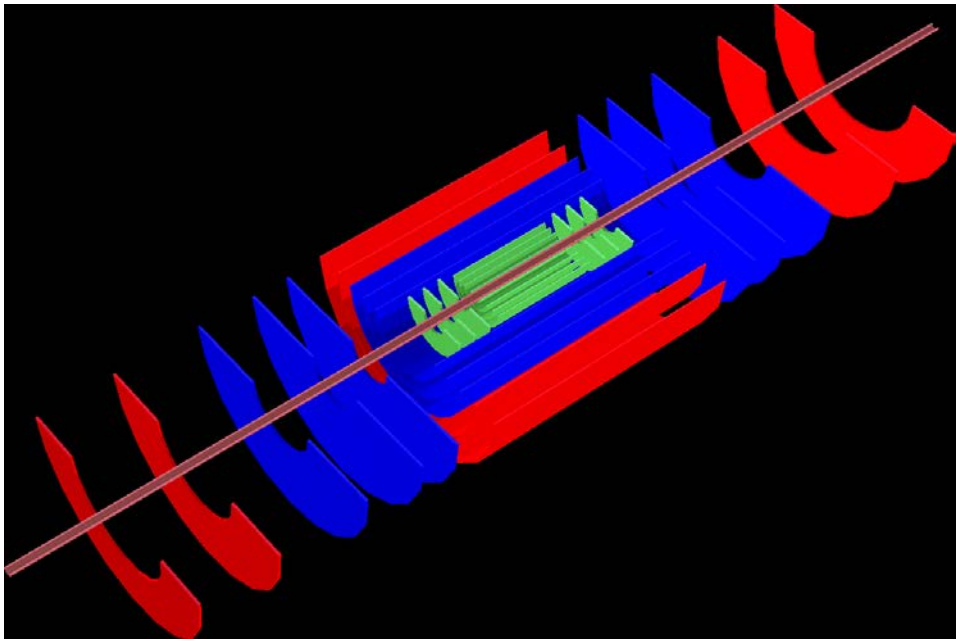


(a)

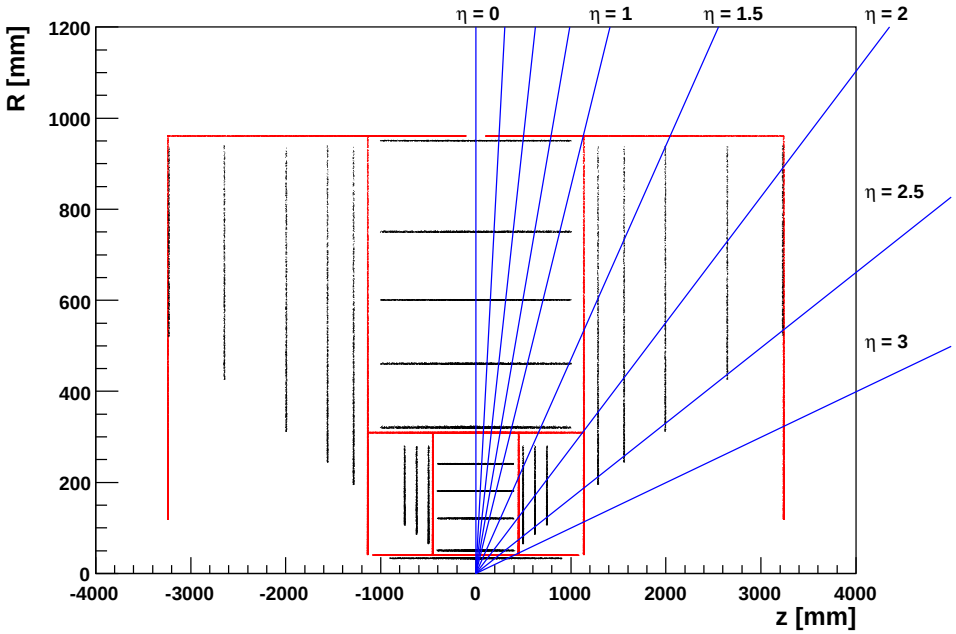


(b)

Figure 5.4: (a) 3D rendering of the projective Strawman layout in ROOT and (b) a layer hit map produced with scripts from the TrkDetDescrExample package.



(a)



(b)

Figure 5.5: (a) 3D rendering of the equal-length barrel Strawman layout in ROOT and (b) a layer hit map produced with scripts from the TrkDetDescrExample package.

5.3 Extensions to the Virtual Point 1 Event Display (VP1)

VP1, or Virtual Point 1, is a new event display for ATLAS which is aimed to provide a tool useful for the understanding of physics events and to help with debugging of software and analysis [42].

The software is fully embedded in the ATLAS offline software framework ATHENA, and thus provides direct access to the same data and algorithms that are used in e.g. the reconstruction of physics events.

VP1 is based on the C++ GUI toolkit Qt 4 [49] and relies on Coin3D/OpenGL [50] for 3D graphics. It utilises a flexible plugin architecture for the actual interaction with ATLAS data and algorithms, allowing for easy parallel development by multiple contributors.

The extensions to Fatras described in Sections 5.1 and 5.2 were integrated into plugins for VP1 to support debugging and validation.

5.3.1 VP1 TrackingGeometry Plugin

The `VP1TrackingGeometryPlugin` package makes it possible to display both the standard ATLAS reconstruction geometry and the `TrackingGeometry` created by the `GenericGeometry` extension in VP1. The navigation volumes and layers, as well as the active detector surfaces, can be translated, rotated and zoomed in three-dimensional space. Figure 5.6 shows the VP1 window configured with the plugin. In Figures 5.7 and 5.8, the ATLAS reconstruction and the projective Strawman geometry are displayed, respectively.

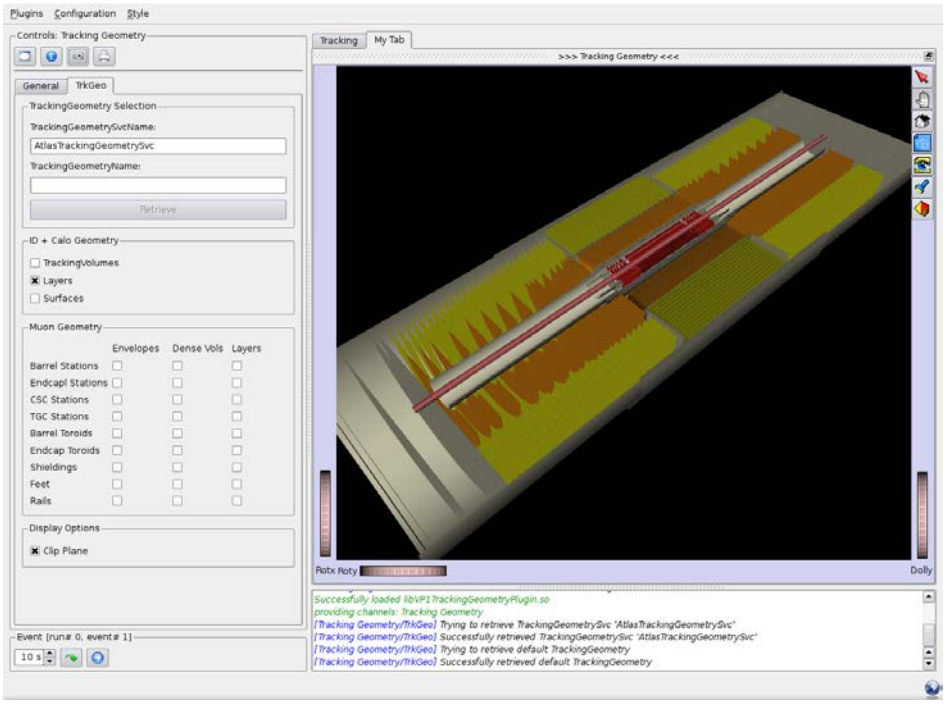


Figure 5.6: The VP1TrackingGeometryPlugin showing the navigation layers of the standard ATLAS reconstruction geometry. The geometry is clipped along the x-z plane.

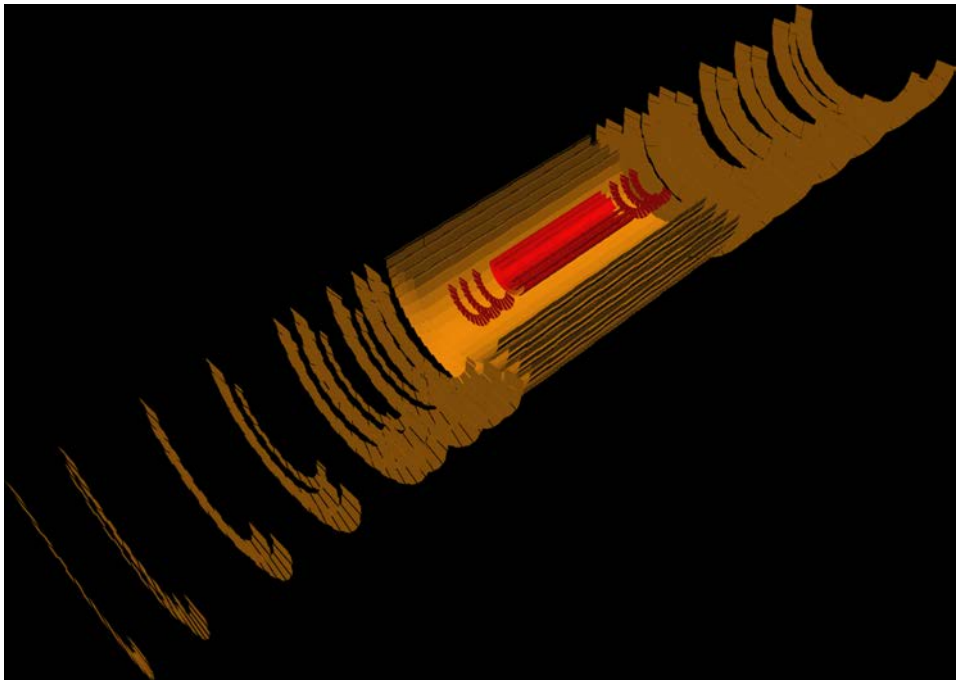


Figure 5.7: The active detector surfaces of the standard ATLAS reconstruction geometry shown in VP1. The geometry is clipped along the x-z plane.

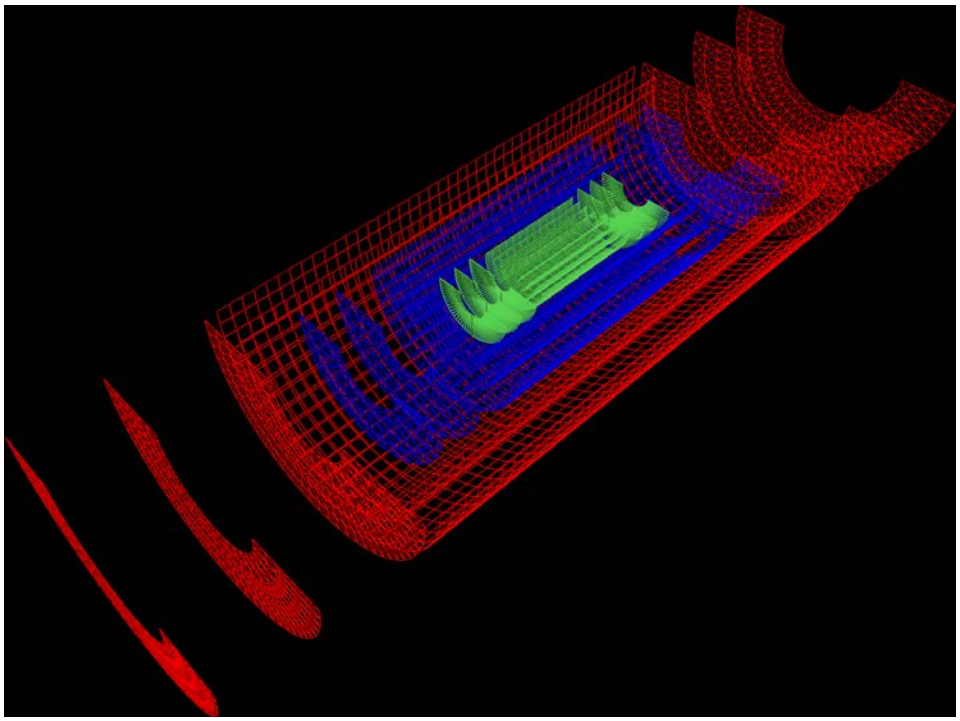


Figure 5.8: The geometry of the projective Strawman layout displayed in a wireframe mode with VP1. The geometry is clipped along the x-z plane.

5.3.2 VP1 Fatras Mode

The `VP1FatrasSystems` package adds an instance of Fatras configured for single track simulation to VP1. Events can be simulated in real-time by selecting a particle from a list. After the simulation step, the created track information is displayed along with the geometry. In this way, the correct behaviour of the `G4ParticleDecay Algorithm` has been tested. An example of a K_S^0 decaying to to two pions is shown in Figure 5.9.

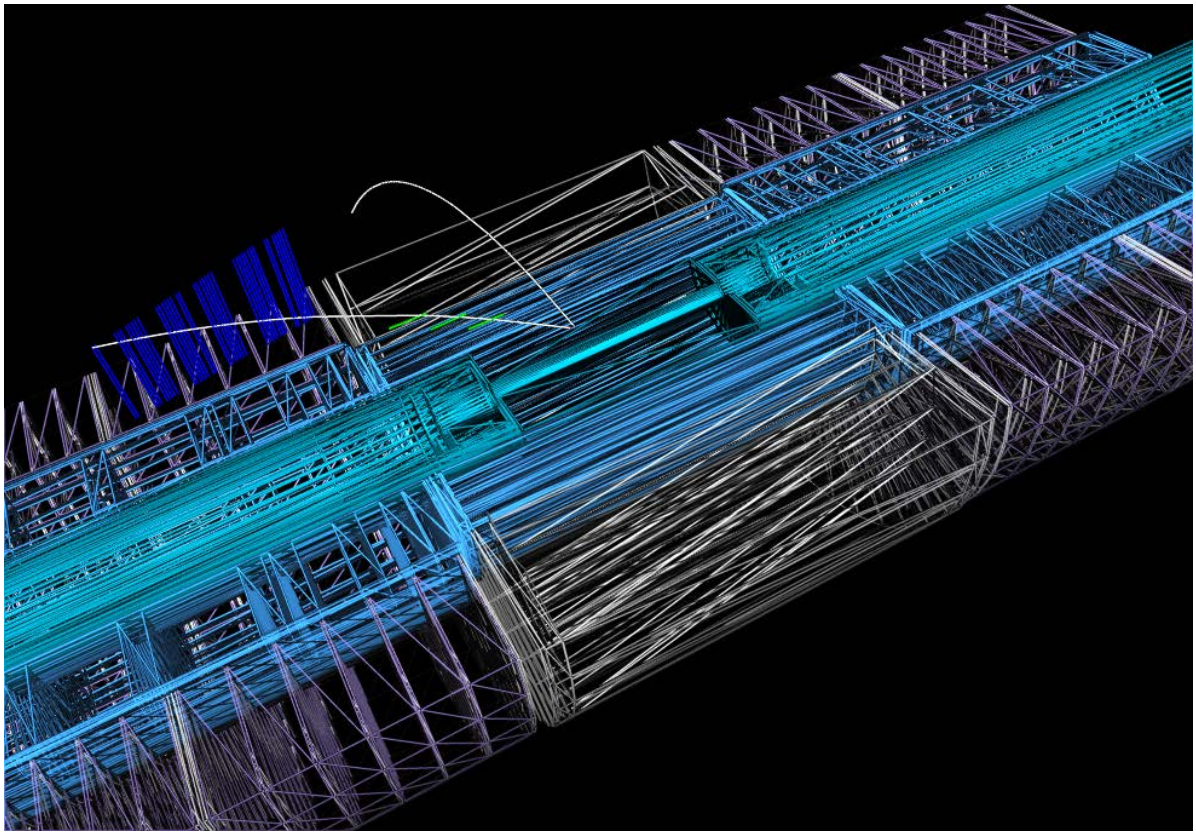


Figure 5.9: Decay of a K_S^0 to two pions in a cut-away view of the ATLAS Inner Detector in VP1. SCT hits of the left track are indicated by green, TRT hits by blue colored detector elements.

6 Fatras Validation Studies

In this chapter, Fatras is compared to the full simulation by means of typical benchmark studies that only depend on the ATLAS Inner Detector. Single track validation has already been started in the context of the creation of the individual algorithms in Fatras. A systematic comparison to the offline chain has still to be conducted to prove the applicability of Fatras as a fast simulation method complementary to Geant4 for huge amounts of data samples.

6.1 General Setup

As mentioned earlier in this text, Fatras has historically been created for tracking code validation. For this reason, the code base develops very quickly in sync with the changes made to other parts of ATHENA, that are connected to the track reconstruction chain. This gives rise to a negative side-effect: new contributions to Fatras are almost always merged into the most recent version that is only compatible with ATHENA development releases called *developer nightlies*.

This is why Fatras had to be run on a pre-14.0.0 developer nightly version of ATHENA. Unfortunately, in the time frame of the validation studies, it was not feasible to simulate full physics events for the full Inner Detector simulation. Thus, older event samples had to be used, simulated with ATHENA release 13.0.40. For studies using single lepton samples, the full simulation was run locally as this is possible to do on a small time scale.

6.2 Track Reconstruction Efficiencies

Measuring the track reconstruction efficiencies for several types of particles is a very basic but also one of the most important benchmark studies when estimating the performance of a detector geometry and the detector simulation framework, as well as the whole reconstruction chain.

The efficiency is defined by the equation

$$\epsilon = \frac{n_{reco}}{n_{true}}, \tag{6.1}$$

where n_{reco} and n_{true} is the number of reconstructed and simulated tracks, respectively. Here, only tracks within the tracker design acceptance of $|\eta| < 2.5$ and a transverse momentum $p_T > 500\,\text{MeV}$ are considered for all particle types. For reconstructed tracks, a match to a simulated track has to be present.

6.2.1 Electron and Muon Efficiencies

A comparison of the reconstruction efficiencies for muons and electrons in offline simulation and Fatras can be seen in Figure 6.1.

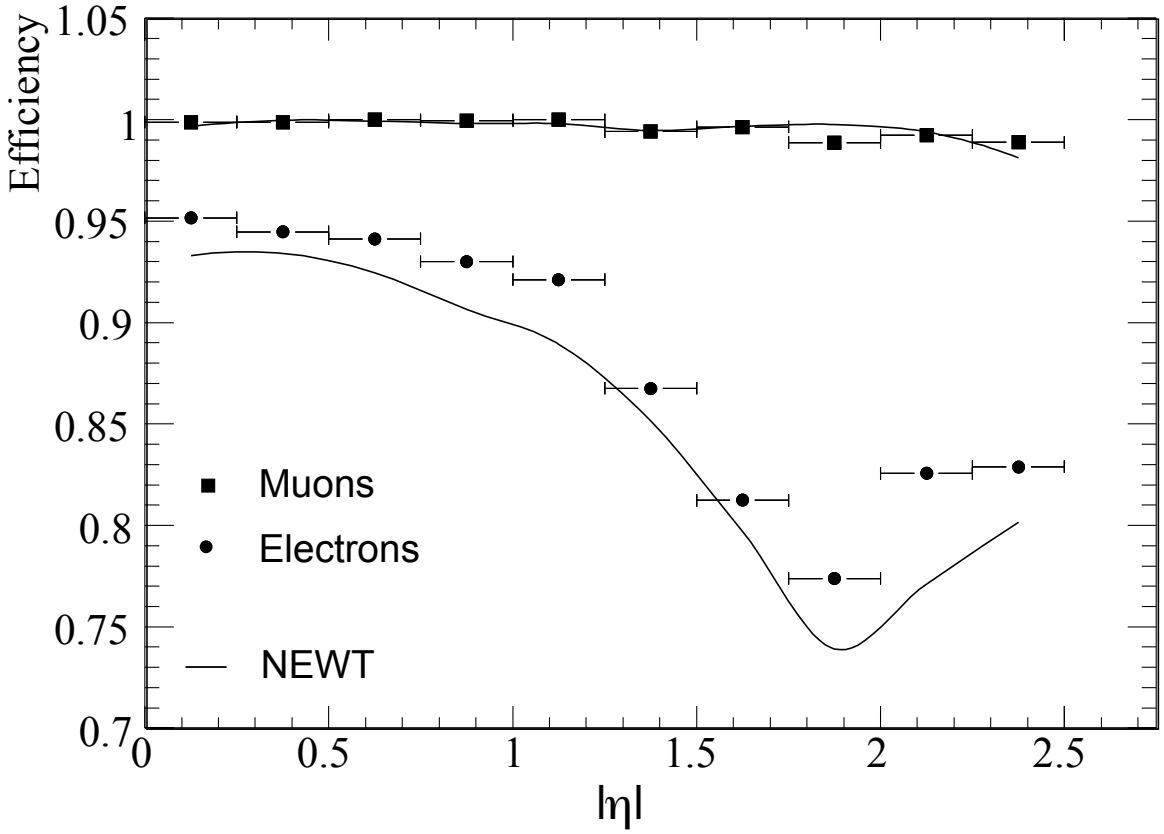


Figure 6.1: Comparison of reconstruction efficiencies for single electron and muon tracks with transverse momentum of $p_T = 5$ GeV as a function of $|\eta|$. The markers display the results obtained with Fatras whereas the results from Geant4 with New Tracking (NEWT) are shown by a continuous line.

The excellent reconstruction efficiency for muons is due to their almost negligible interaction probability with detector material. For this reason, both simulation frameworks show very good agreement for all values of η.

Electrons, on the other hand, are more subjected to bremsstrahlung which leads to a smaller reconstruction efficiency. Due to the simplified material interaction within the fast simulation, the efficiencies are slightly higher in Fatras but the difference is always less than 5%. The large drop in efficiency at $|\eta| = 1.9$ is caused by the additional material in the transition region from the barrel to the endcap detectors of the Inner Tracker.

6.2.2 Pion Efficiencies

In Section 4.2.3, it has been pointed out that the fast simulation of hadronic interactions with detector material in Fatras is still limited. However, a comparison of the simple scaling model is presented in Figure 6.2.

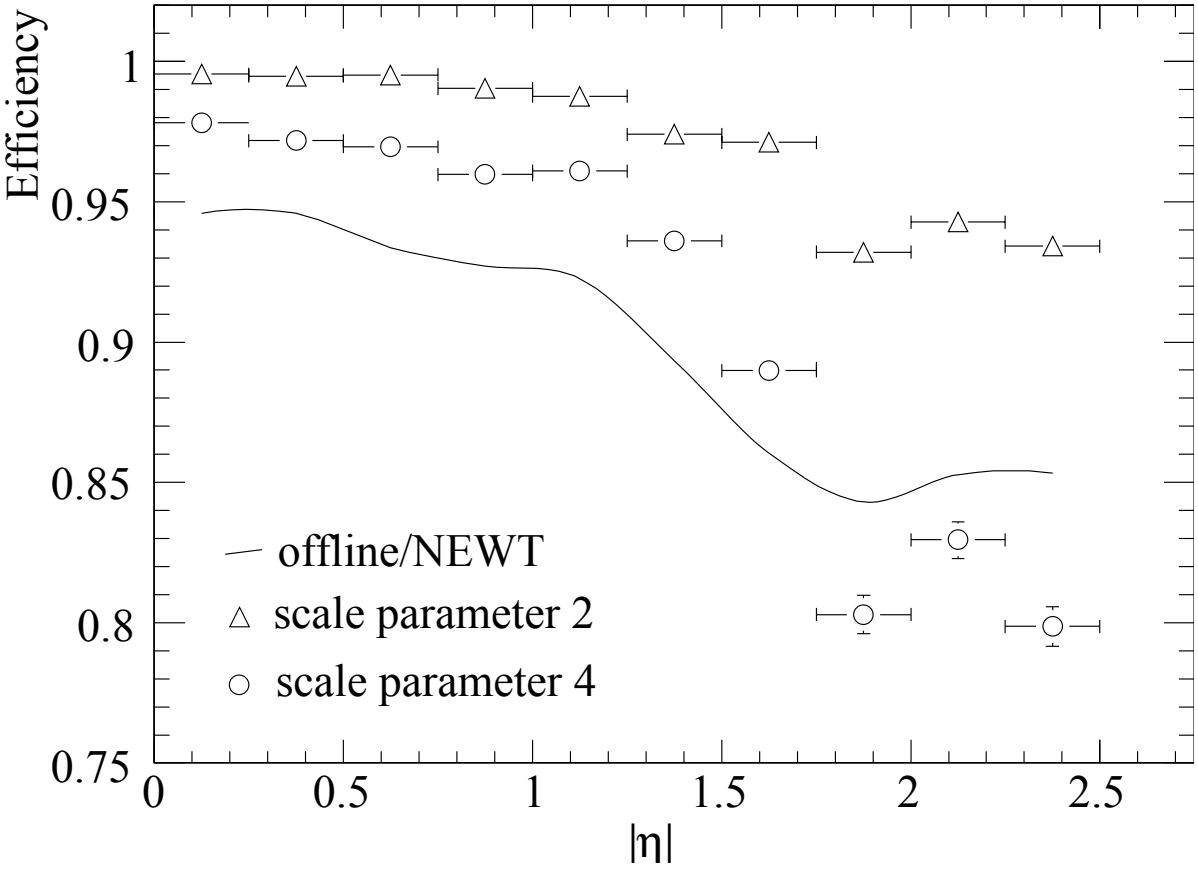


Figure 6.2: Comparison of reconstruction efficiencies for single pion tracks with transverse momentum of $p_T = 5$ GeV for two different values of the scale parameter in the current hadronic interaction model of Fatras (see 4.2.3). The markers display the results obtained with Fatras whereas the results from Geant4 with New Tracking (NEWT) are shown by a continuous line.

The results derived from full simulation show a decrease in efficiency for large values of $|\eta|$ very similar as for electrons. It is obvious to see that the shape of the simulated data from Geant4 is reproduced very poorly by the current hadronic interaction model in Fatras. The scaling of the radiation length X_0 has to be replaced by a proper description of the hadronic interaction length.

6.3 Vertex Reconstruction

The reconstruction of the primary vertex position is of critical importance at the LHC, because it is the main tool available to distinguish particles coming from the hard scattering event under study (signal) from particles arising from additional proton-proton interactions taking place in the same bunch crossing (pile-up). This is very important for the general track reconstruction and the determination of the event topology. Even if the position of such interaction vertices is constrained in the x-y plane by the beam spot size of 15 μm, so that signal and pile-up vertex positions cannot be distinguished in the transverse plane. In the z-direction the beam profile is very large in size, so the vertices distribute approximately according to a Gaussian distribution with a width of 56 mm.

Two main phases of the vertex reconstruction can be distinguished: in the vertex *finding* process, possible vertex candidates are derived from histogramming the z positions of the perigee of the tracks. During the *fitting* stage, the positions and the covariance matrices of these vertices is estimated.

The default fitting method in ATLAS as of Athena release 13 is the *Multi-adaptive vertex fitter* [51]. In this method, different vertex candidates compete for the tracks. This is accomplished by assigning adaptive weights to the tracks in order to find an optimal solution.

As a benchmark event type for testing vertex reconstruction efficiency, $t\bar{t}$-events are often used because of their high track multiplicity.

Figures 6.3 to 6.8 show the resolutions and pulls of the x,y and z coordinates of the fitted primary vertex in both simulations. A double-Gaussian fit has been applied to be able to describe the core and the tails separately. A single Gaussian fit does not suffice to accurately describe both the peak and the tails of these distributions. The histogrammed data is actually a convolution of many individual Gaussian distributions. However, a fit of a double-Gaussian

$$p(x_1, x_2) = \frac{c_1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x_1-\overline{x}_1)^2}{\sigma_1^2}} + \frac{c_2}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x_2-\overline{x}_2)^2}{\sigma_2^2}} \tag{6.2}$$

works well enough to result in a reasonable fit quality.

The fit parameters *Const1*, *Mean1* and *Sigma1* shown in the diagrams correspond to c_1, $\overline{x_1}$ and σ_1 respectively, similarly for the second Gaussian.

The resolutions and pulls for the vertex position in the plane transverse to the beamline in Figures 6.3 to Figures 6.6 show very similar results. For the full simulation, the RMS of the x and y resolution distributions is about 11 μm and agrees with results found e.g. in [10], p. 330f. Results for Fatras deliver a RMS of about 10 μm. The slightly better resolutions in the fast simulation are cause by the broader tails of the distributions which is reflected in the fit parameters. This conclusion can not be drawn from the knowledge of the RMS only as it does not account for the tails.

The pull distributions deliver similar results.

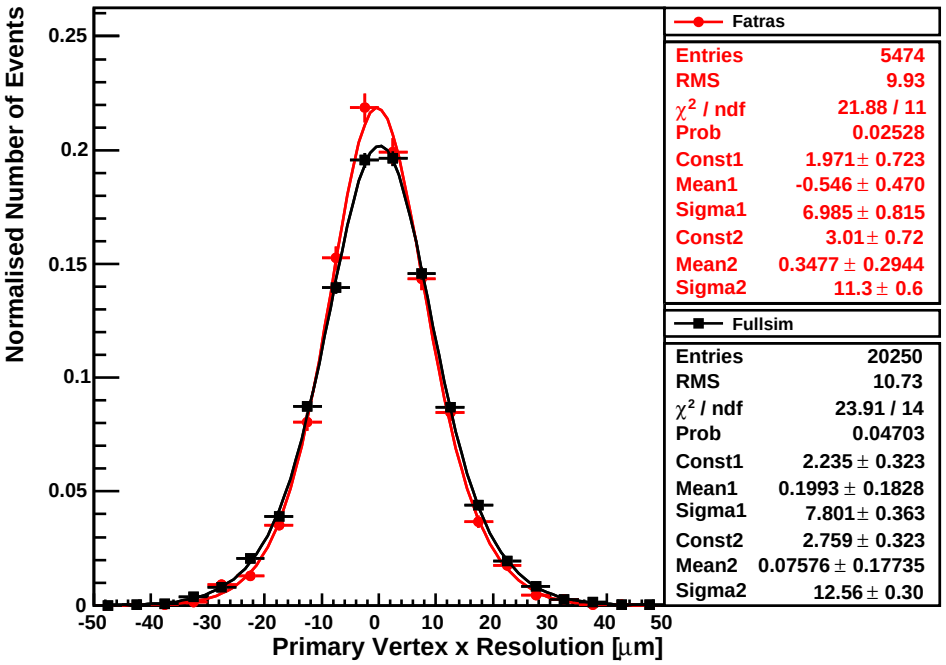


Figure 6.3: Resolution in x direction of the reconstructed primary vertex position.

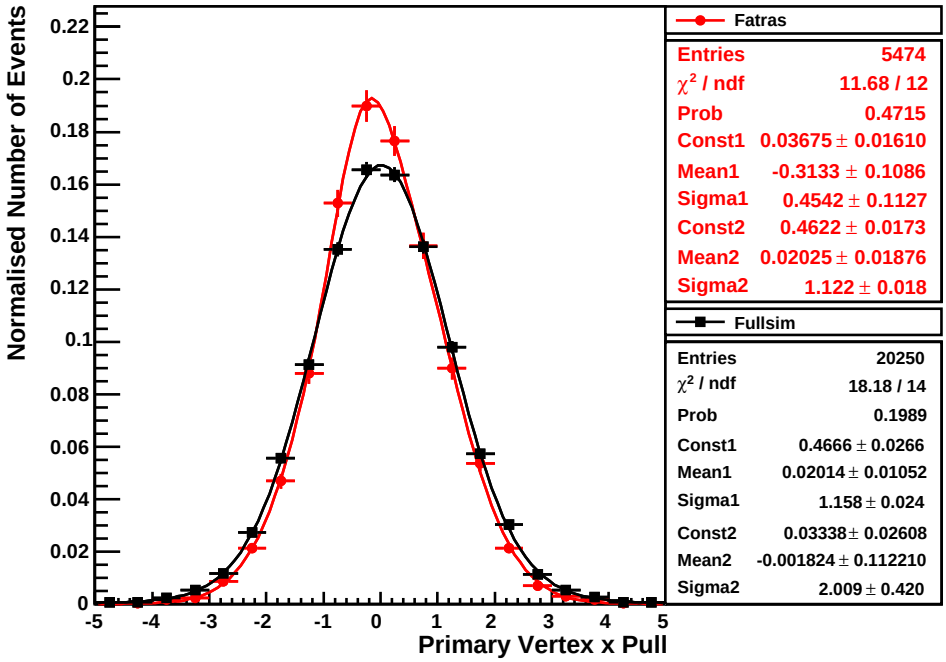


Figure 6.4: Pull in x direction of the reconstructed primary vertex position.

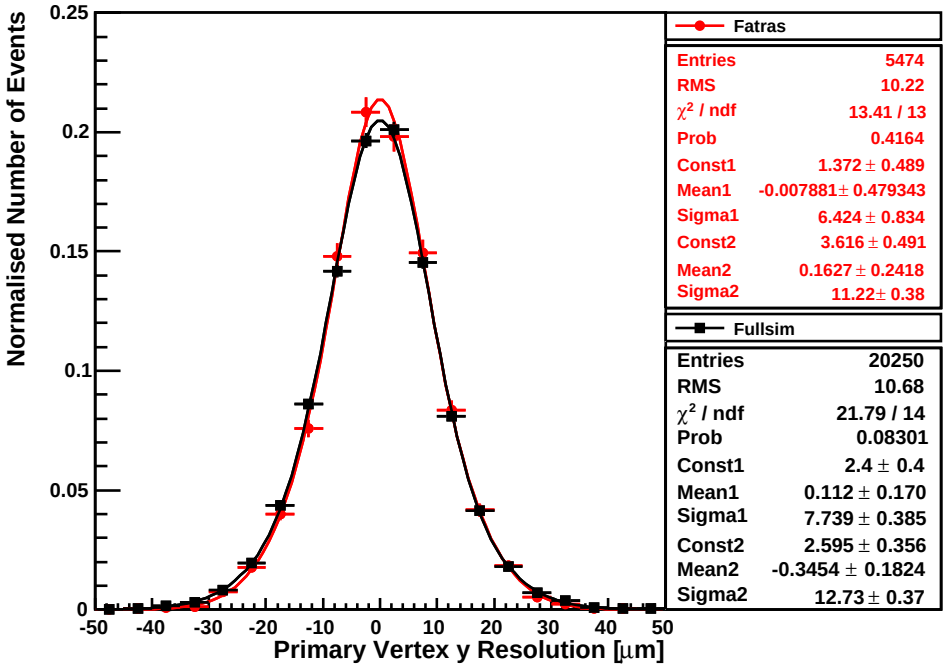


Figure 6.5: Resolution in y direction of the reconstructed primary vertex position.

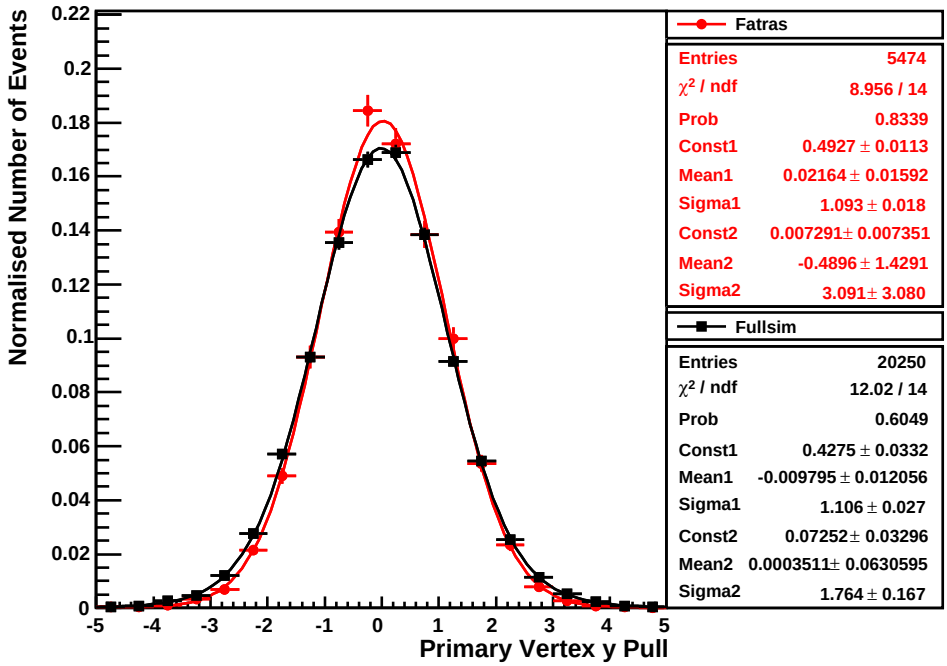


Figure 6.6: Pull in y direction of the reconstructed primary vertex position.

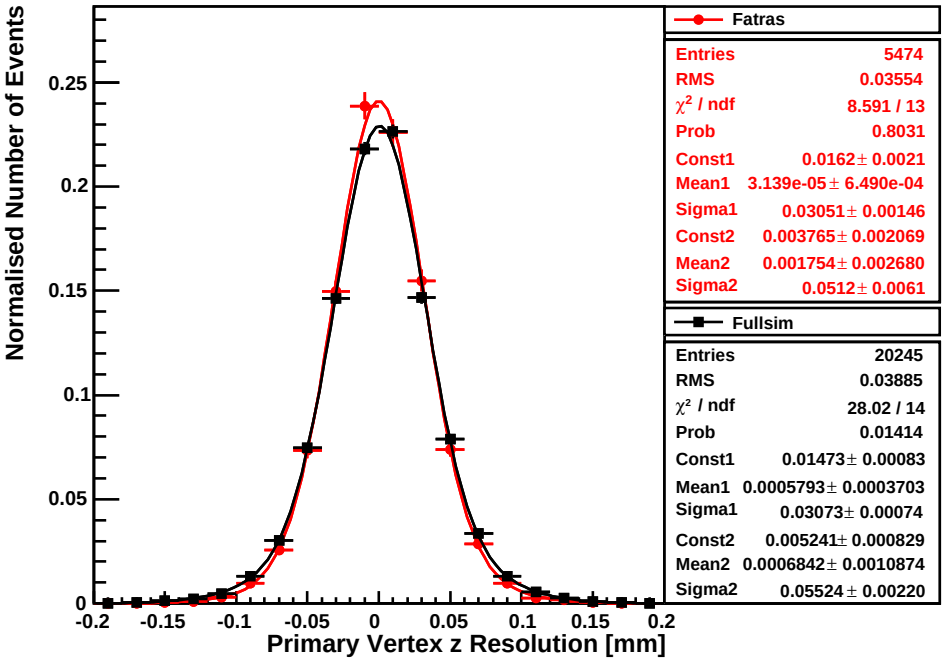


Figure 6.7: Resolution in z direction of the reconstructed primary vertex.

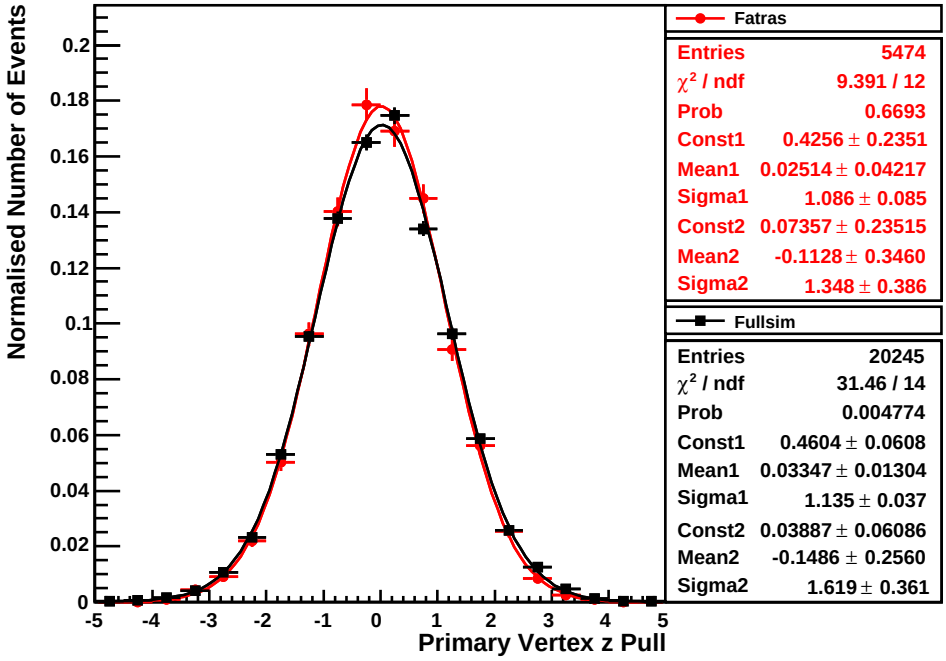


Figure 6.8: Pull in z direction of the reconstructed primary vertex.

The resolution in the z direction shows an RMS of about 39 μm for the full simulation and 36 μm for Fatras, as shown in Figure 6.7. The same arguments as for the x and y resolutions hold here. The pull distributions (Figure 6.8) agree nicely.

In all plots, a slight overestimation of the vertex position precision can be noted in Fatras, but only to an extent that is statistically negligible. This is reflected in an overall very good agreement of the distribution widths.

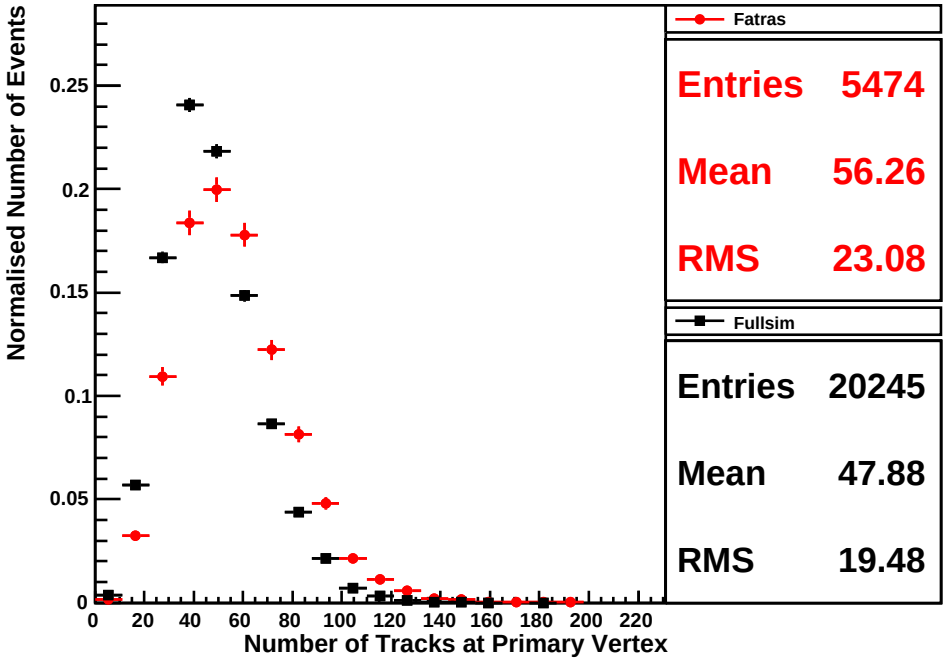


Figure 6.9: Number of tracks originating from the reconstructed primary vertex.

In Figure 6.9, the number of reconstructed tracks fitted to the primary vertex is shown. In average, eight more tracks are observed for the fast simulation in comparison to the full simulation. This large discrepancy does obviously not affect the vertex position resolutions very much. A possible explanation is that additional tracks of very low-p_T are considered in the vertex fit but only contribute with small weights. Usually this is the case for tracks with poor impact parameter resolutions caused by heavy multiple scattering processes.

6.4 $Z^0 \rightarrow 2l$ Mass Reconstruction

To get an impression on how a difference in track reconstruction efficiencies and track parameter resolutions is reflected in the analysis of a physics event, the reconstruction of the invariant lepton pair mass in Z^0 decays is studied in the following.

Two types of samples were chosen: a Z^0 decaying to two muons should show a very distinct shape of the Z^0 mass peak in the lepton invariant mass distribution as muons are much less subjected to bremsstrahlung in comparison to electrons. The latter can loose a large fraction of their energy when radiating photons while passing through detector material. A shift and asymmetric broadening of the Z^0 mass peak is therefore to be expected in the distribution of the invariant electron mass. Thus, a comparison of $Z^0 \rightarrow e^+e^-$-events in fast and full simulation is of great interest to probe on how accurate the energy loss simulation of Fatras was implemented and tuned.

The truth data is used to distinguish the types of particles that are part of the event. Only tracks with a transverse momentum greater than 25 GeV are considered for the analysis. After that, the two leptons with calculated invariant mass closest to the Z^0 mass are selected and assumed to originate from the Z^0 decay. Even though this is a quite artificial method, it should suffice for the comparison of the two simulation types.

6.4.1 $Z^0 \rightarrow \mu^+\mu^-$ Mass Reconstruction

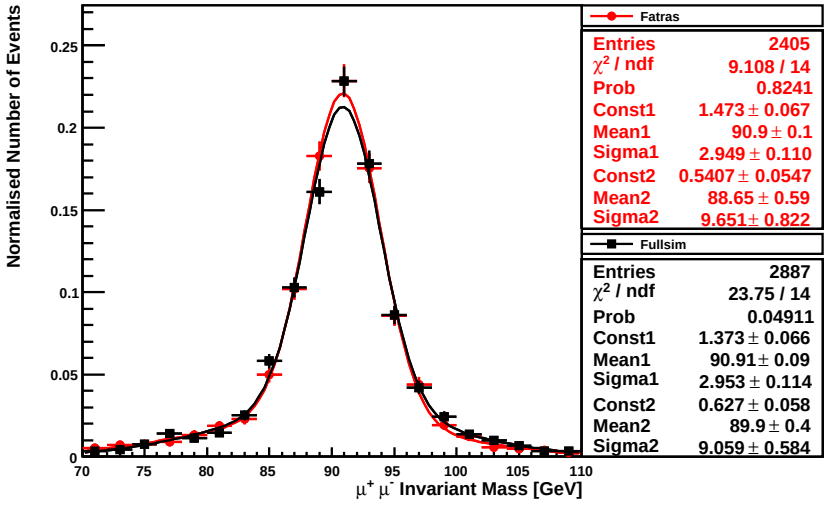


Figure 6.10: Comparison of the dimuon invariant mass in fast and full simulation with double-Gaussian fit functions.

Figure 6.10 shows a fit of a double-Gaussian function (see Equation 6.2) to each of the data samples to separate the peak from the tails of the distributions. The simulated data is in fact a convolution of a Breit-Wigner distribution from the Z^0 decay and a Gaussian distribution reflecting the resolution of the detector. The fit function used in the plots delivers an adequate description which results in a low χ^2/ndf value. The two Gaussian fits to the peaks

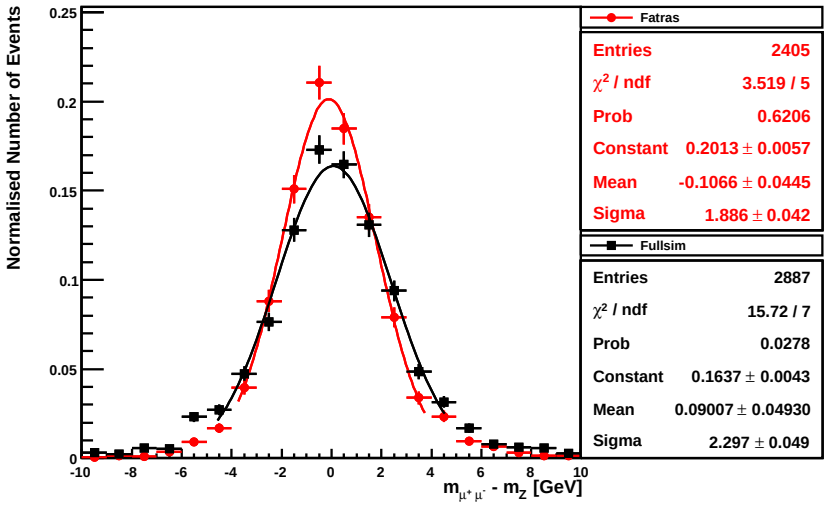


Figure 6.11: Comparison of the difference between the dimuon invariant mass and the true Z^0 mass per event in fast and full simulation with Gaussian fit functions.

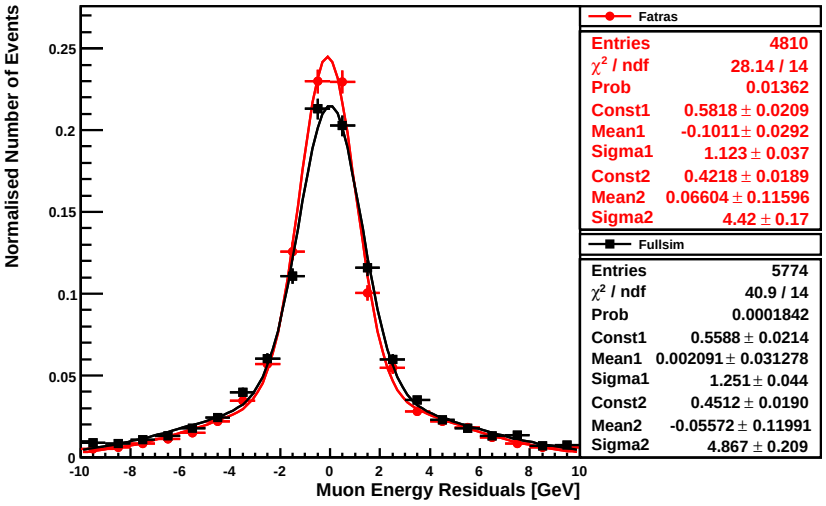


Figure 6.12: Comparison of the muon energy resolutions in fast and full simulation with double-Gaussian fit functions.

are described by the parameters Const1, Mean1 and Sigma1 and show excellent agreement. However, it is observable that the fast simulation data is a bit narrower.

This observation is confirmed by the distributions of the difference between the dimuon invariant mass and the true Z^0 mass per event (Figure 6.11). After the subtraction of the Z^0 mass, the pure detector resolutions are derived. The data points should follow a Gaussian distribution. A fit shows indeed, that the resolution for the Z^0 mass in Fatras is about 20% better than in the full simulation.

For the energy resolution (Figure 6.12), the two simulation frameworks agree well within the uncertainties.

6.4.2 $Z^0 \rightarrow e^+e^-$ Mass Reconstruction

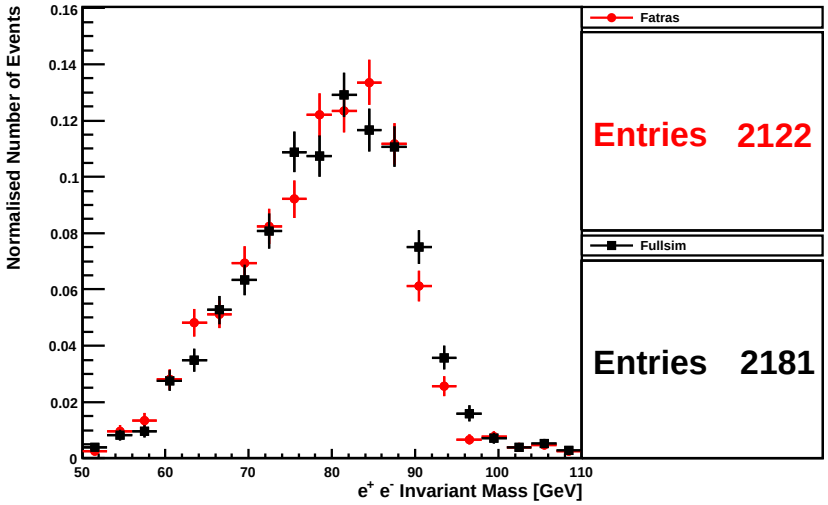


Figure 6.13: Comparison of the dielectron invariant mass in fast and full simulation.

The two distributions in Figure 6.13 show the invariant mass of two electrons in $Z^0 \rightarrow e^+e^-$ events. The data shows a very broad tail towards smaller energies which is caused by bremsstrahlung of the electrons in the material of the Inner Detector.

For the measurement of the Z^0 mass, calorimeter information is usually used for the isolation of energy clusters produced by photons near the impact point of the electrons. In order to compare the two simulation frameworks and to estimate the quality of the description of bremsstrahlung in Fatras, this correction is not performed.

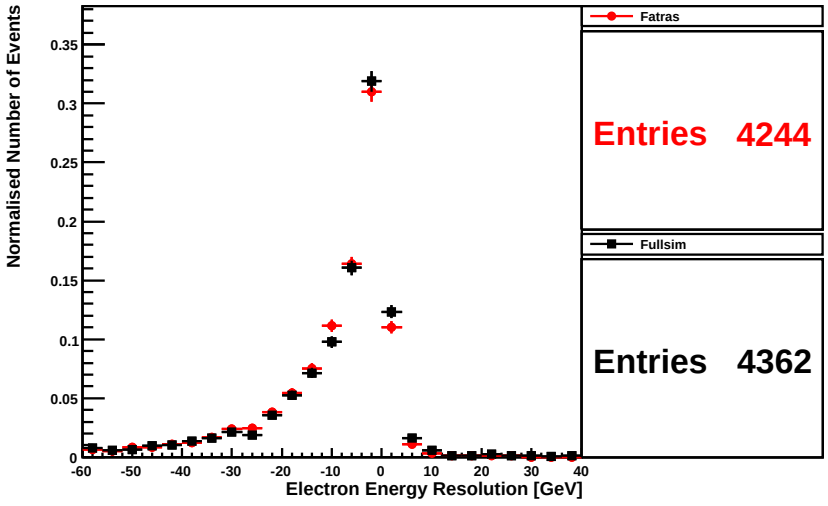


Figure 6.14: Comparison of the electron energy resolutions in fast and full simulation.

In fact, even if Fatras only does one iteration of bremsstrahlung per particle, the distributions

agree well within the statistical uncertainties. The energy residual plot in Figure 6.14 also reflects the energy loss by bremsstrahlung photons in the large tail in the negative region of the x-axis.

7 Performance Studies with Fatras for the SLHC Upgrade

In this chapter, the `GenericGeometry` extension is used to simulate the implementations of the two main Inner Detector upgrade layout proposals introduced in Section 2.4. The following section features a general comparison of the projective and the equal-length Strawman layout. A detailed analysis of basic track parameters and hit occupancies as a function of luminosity for the projective Strawman layout follows.

7.1 Layout Comparison

7.1.1 Material vs. Pseudorapidity η

This section describes the material distributions for the beampipe and the silicon strip detectors in both geometries. The amount of material which is traversed in a straight line by imaginary neutral particles is plotted as a function of η. Data from the projective Strawman is drawn as a solid-line histogram whereas dots represent the equal-length barrel Strawman layout.

The beampipe used in upgrade simulation studies is assumed to be identical to the one employed by the `TrackingGeometry` of the current detector. For this reason, the distribution in Figure 7.1 is identical to the corresponding plot shown in [34].

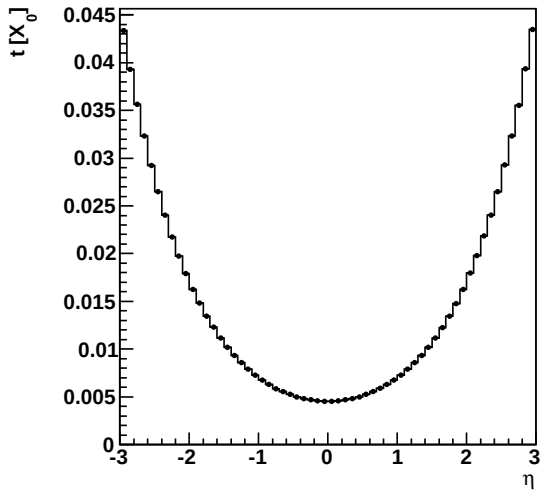
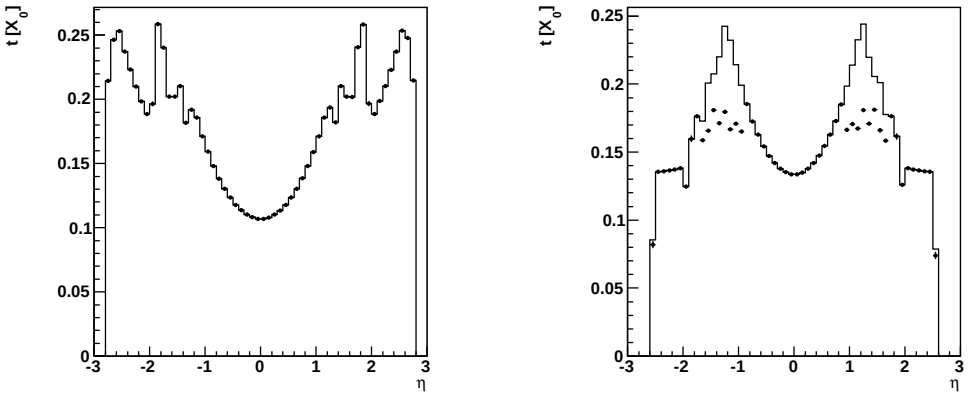


Figure 7.1: Beampipe material in units of radiation length X_0 as a function of η.

Figures 7.2a and 7.2b show the pixel detector material and the integrated material in short-strip and long-strip detectors, respectively. In the implemented layouts, every silicon detector

layer consists of plain 250 μm silicon without accounting for the service material, like cabling, cooling and the readout electronics. Thus, the amount of material is underestimated and has to be tuned in future studies. However, in close approximation the total material is expected to be proportional to the silicon detector area, hence the relative proportions are assumed to be realistic.

The two layouts can be clearly distinguished in the region of $1 < |\eta| < 2$, where the barrel-endcap transition regions of the strip detectors feature very different accumulated thicknesses for the material. On the first glance, the smaller amount of material for the equal-length barrel layout in this region could lead to the assumption that this specific geometry poses the better solution with respect to the material distribution. Huge amounts of cabling coming out of the Inner Detector will be situated in this region, though, and the cabling is not taken into account by this study. For this reason, a final conclusion can not be drawn yet.



(a) Pixel detector

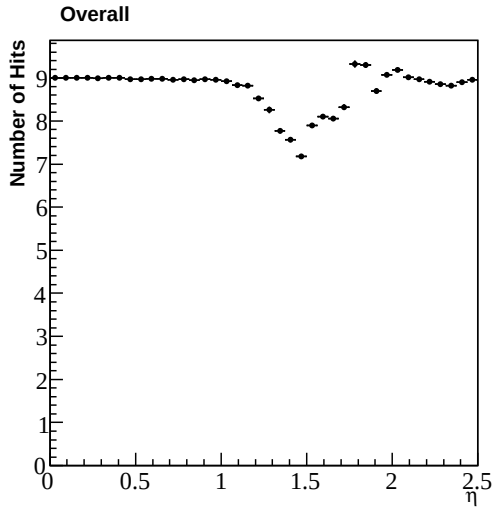
(b) Short-strip and long-strip detectors

Figure 7.2: The material distributions of the silicon detectors are shown in (a) for the pixel detector and in (b) for the strip detectors in units of radiation length X_0 as a function of η. Data from the projective Strawman is drawn as a solid-line histogram whereas the dots represent the equal-length barrel Strawman layout.

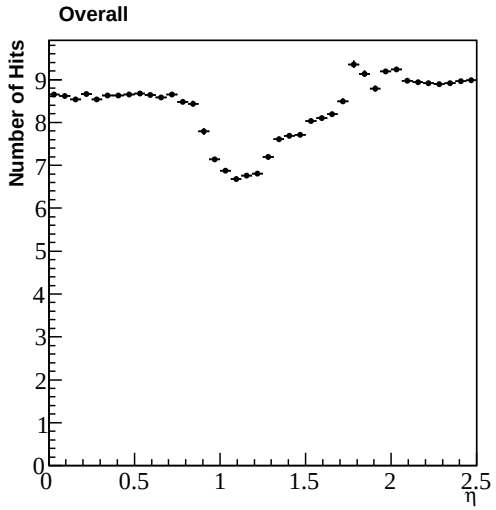
7.1.2 Hits vs. Pseudorapidity η

The number of hits along a track is a critical benchmark when estimating the accuracy of track parameter resolutions. Usually, a certain number of hits is required by the track reconstruction software to assure the quality of the track fit. In the ATLAS detector paper [10] for example, a track only passes the quality cuts if it has a minimum of seven precision (i.e. silicon layer) hits.

In Figures 7.3a and 7.3b, the number of hits in sensitive detector material are shown for the two main upgrade layouts. The simulation, again, uses neutral particles with unbent trajectories. Given the current quality cuts of a minimum of seven silicon hits for the present ATLAS Inner Detector, the projective Strawman layout would be preferred to the proposal with equal-length barrels as the average number of hits is higher and stays above seven for the whole range in η.



(a) Number of hits vs. η for the projective Strawman layout.



(b) Number of hits vs. η for the equal-length barrel Strawman layout.

Figure 7.3: Number of hits vs. η for the (a) projective and the (b) equal-length barrel Strawman layout.

7.2 Additional Studies for the Projective Strawman Layout

For getting a first glimpse on the performance of the projective Strawman layout, track parameter resolutions and hit occupancies for several instantaneous luminosities have been studied.

7.2.1 Track Parameters

A sample of 10000 muons with a momentum of 50 GeV has been simulated with Fatras to give an estimation on the track parameter resolutions for the projective Strawman layout.

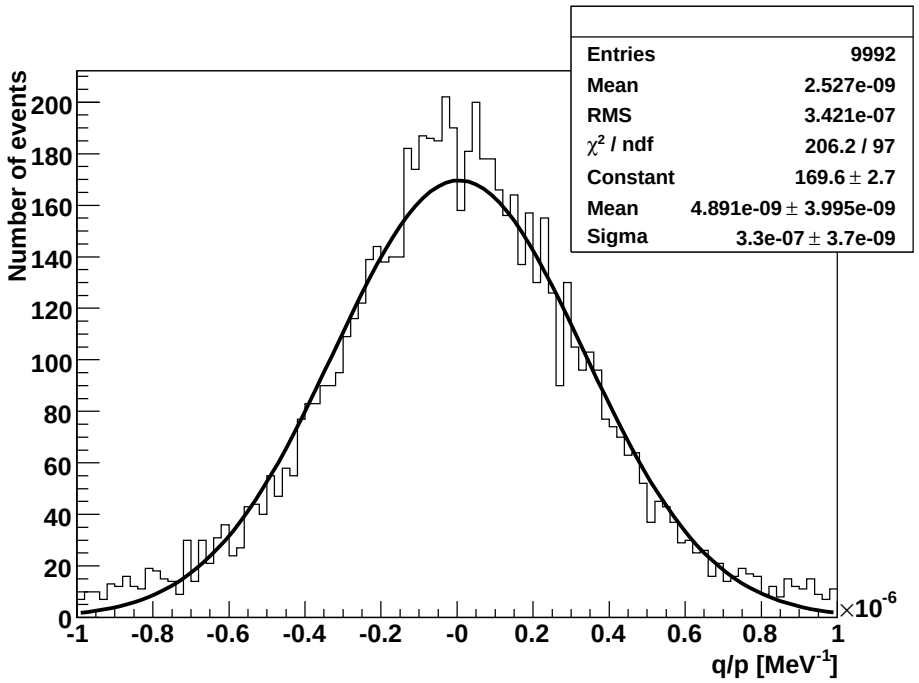


Figure 7.4: Resolution of the $\frac{q}{p}$ track parameter for 50 GeV muons simulated with the projective Strawman layout.

Figure 7.4 features a Gaussian fit to the distribution of the q/p track parameter. The mean momentum resolution of the tracks can be derived by the equation

$$\delta(p) = p^2 \cdot \delta(\frac{q}{p}), \tag{7.1}$$

with $\delta(\frac{q}{p})$ corresponding to the width of the Gaussian fit.

For the given momentum of 50 GeV, equation 7.1 translates to 825 MeV, or 1.65% of the original momentum. This result is compatible with the requirements for the Inner Detector performance in [11].

The resolutions of the remaining track parameters are shown in Figure 7.5 for completeness, corresponding pull distributions are given in Figure 7.6.

The Gaussian fits to the pulls reproduce the shapes satisfyingly well. For the $\delta(\frac{q}{p})$ parameter, a large non-Gaussian tail has to be noted.

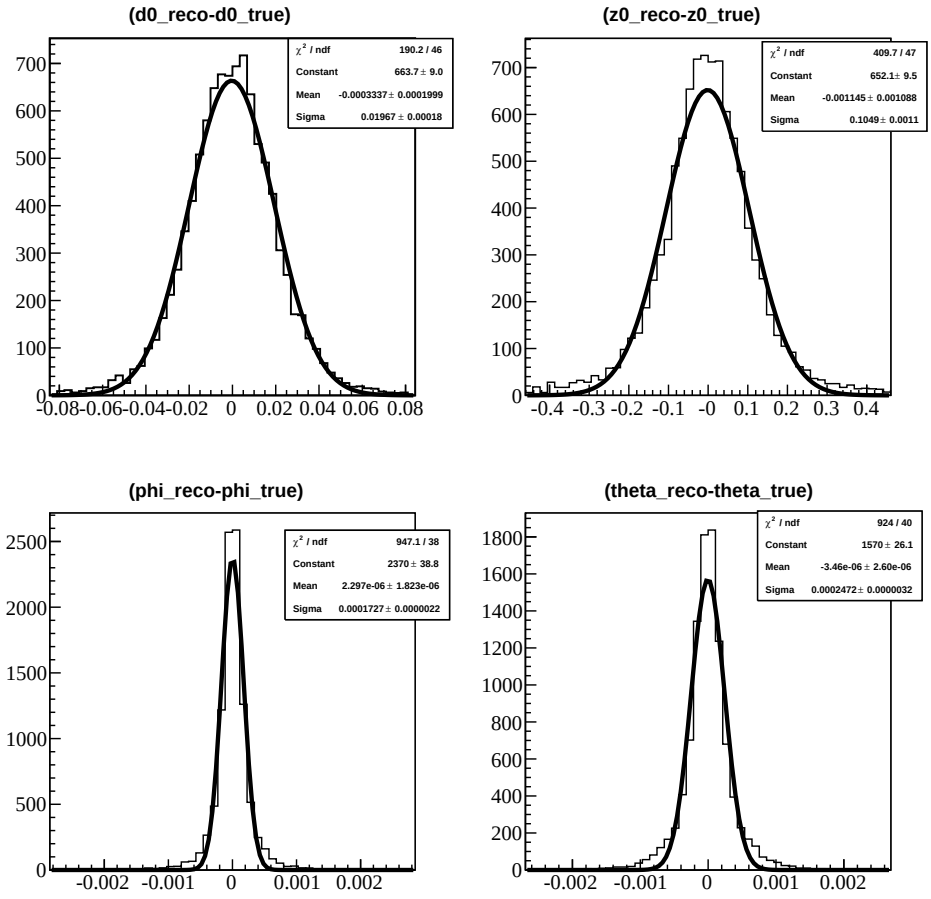


Figure 7.5: Resolutions of the d_0, z_0, ϕ and θ track parameters for 50 GeV muons simulated with the projective Strawman layout.

7.2.2 Hit Occupancy vs. Number of Pile-up Events

For the hit occupancy study, a $t\bar{t}$ signal event has been merged with a number of minimum bias events taken from a Poissonian distribution. The mean of this distribution is dictated by the expected number of pile-up events per bunch crossing. For the LHC design parameters, about 23 pile-up events are expected. As the exact value for the integrated luminosity at the SLHC is still unknown, the scaling of the hit occupancy over a range from 2.3 up to 400 pile-up has been studied.

The hit occupancy is measured per layer: the number of hits in each layer is divided by the number of readout channels. A proper hit cluster creation for the `GenericGeometry` is not available for this study, which results in an underestimation of the cluster sizes, depending on the sensor type. With a realistic clustering, the cluster sizes vary from about one to three hit channels for pixels and 1-2 hit channels for strip sensors. To account for this, the results have to be scaled by corresponding factors. In the following figures, this correction is not included.

Figure 7.7 displays the occupancies in the pixel barrel layers, indicated by black circles for

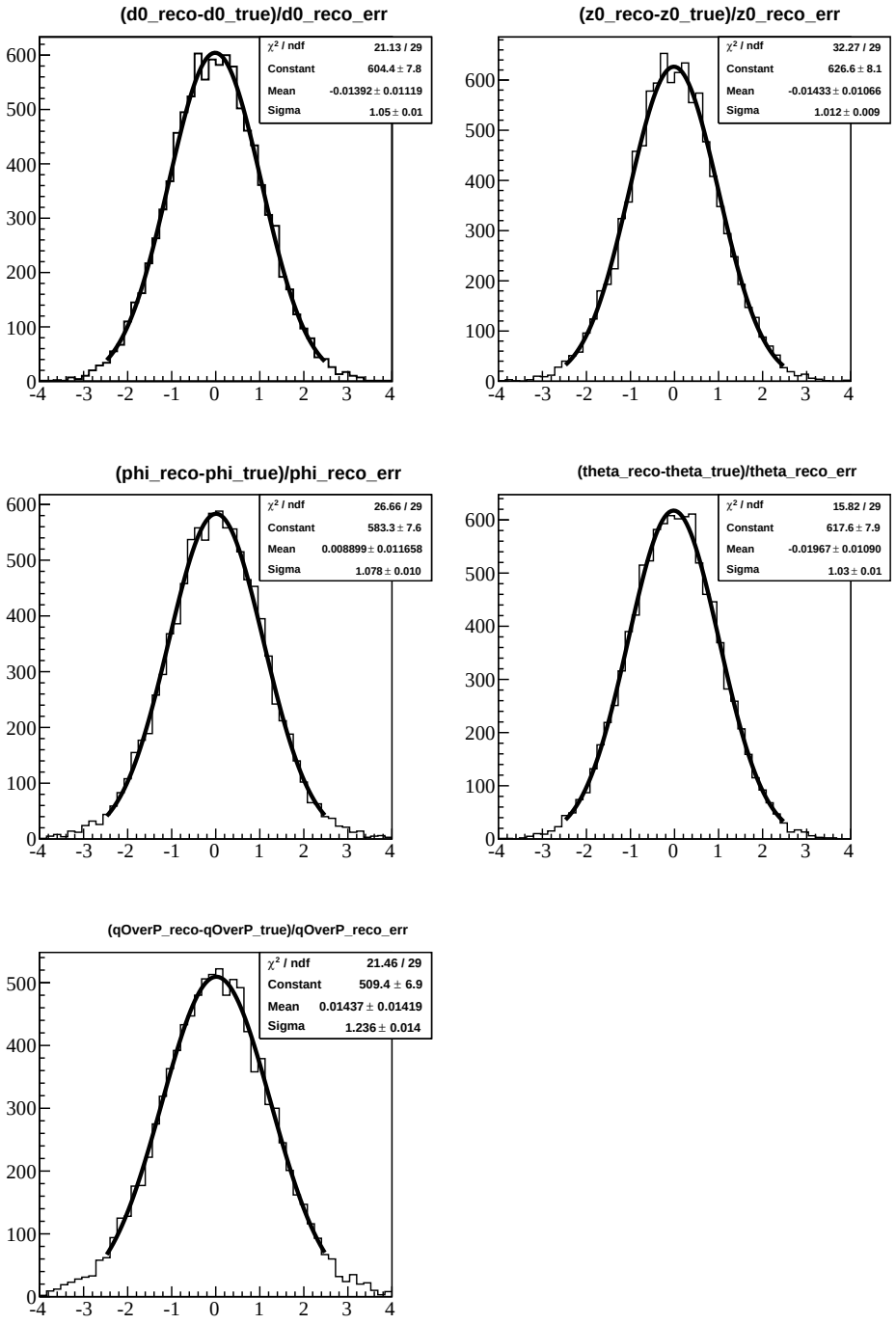


Figure 7.6: Pulls of the d_0, z_0, ϕ, θ and $\frac{q}{p}$ track parameters for 50 GeV muons simulated with the projective Strawman layout.

the B-layer, red squares for the second pixel layer, green pyramids for the third pixel layer and blue triangles for the fourth pixel layer. The occupancy scales linearly with the number of pile-up events, as expected with the constraints given above.

An interesting feature of this plot is the higher occupancy in the third layer in comparison to the second one. This is caused by the larger pixel sizes of $50 \times 400\,\mu\text{m}^2$ in the outermost barrel layers.

The distribution for the pixel endcap (Figure 7.8) shows the data points for the first, second and third disc by black circles, red squares and green pyramids, respectively. As the granularity is the same for all sensors, the hit occupancies decrease with the distance from the interaction region. In Figure 7.9 to 7.12, similar results for the short-strip and long-strip subdetectors are shown.

The maximum hit occupancies are summarized in Table 7.1. At least for the strip detectors, which are designed to achieve a maximum hit occupancy of about 2% at SLHC luminosity, the simulation results look already very good. For the pixel detectors, the simulation should be repeated once a realistic clustering algorithm has been implemented as the estimated occupancies seem to be too small.

Subdetector Part	**Max. Hit Occupancy** 23 pile-up	**Max. Hit Occupancy** 400 pile-up
Pixel Barrel	$2.14 \cdot 10^{-5}$	$3.03 \cdot 10^{-4}$
Pixel Endcap	$8.36 \cdot 10^{-6}$	$1.21 \cdot 10^{-4}$
SS SCT Barrel	$2.91 \cdot 10^{-4}$	$3.88 \cdot 10^{-3}$
SS SCT Endcap	$1.29 \cdot 10^{-3}$	$1.87 \cdot 10^{-2}$
LS SCT Barrel	$1.52 \cdot 10^{-4}$	$2.05 \cdot 10^{-3}$
LS SCT Endcap	$1.09 \cdot 10^{-3}$	$1.56 \cdot 10^{-2}$

Table 7.1: Maximum hit occupancies per subdetector part for LHC design luminosity and maximum SLHC luminosity.

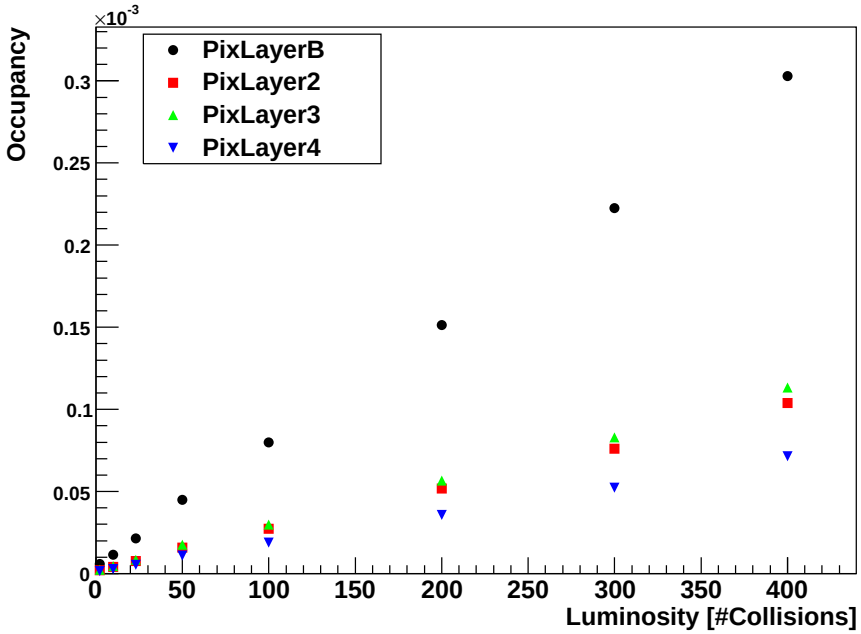


Figure 7.7: Hit occupancy as a function of pile-up in the four pixel barrel layers.

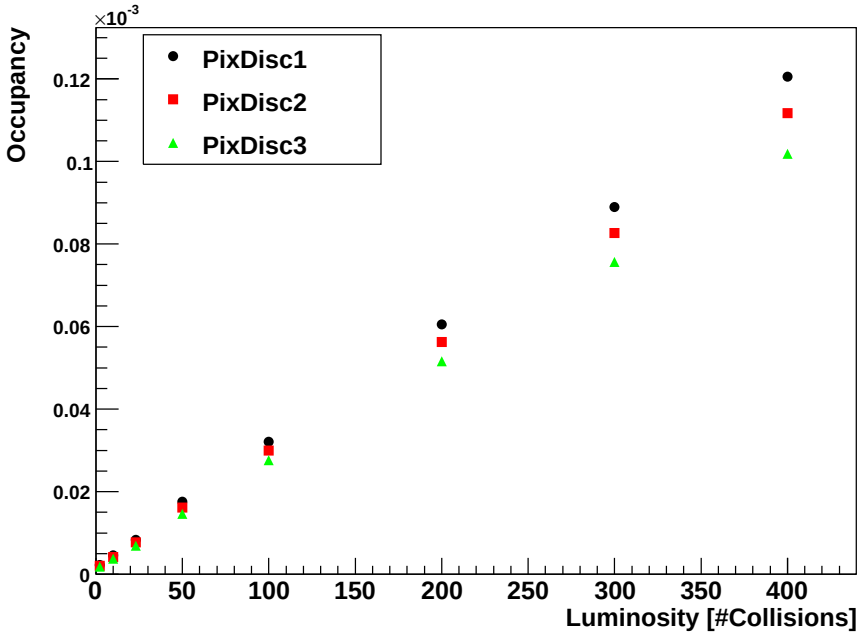


Figure 7.8: Hit occupancy as a function of pile-up in the three pixel endcap discs.

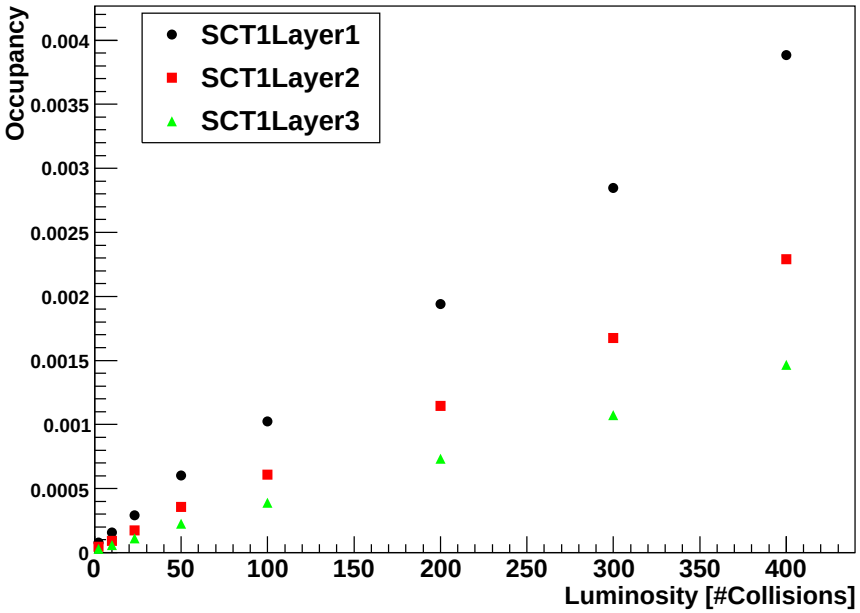


Figure 7.9: Hit occupancy as a function of pile-up in the three short-strip SCT barrel layers.

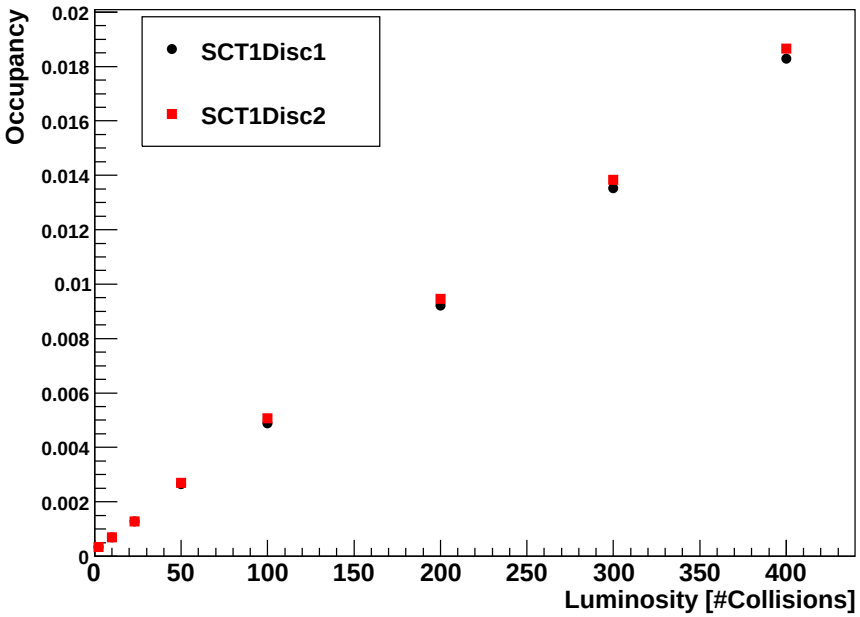


Figure 7.10: Hit occupancy as a function of pile-up in the two short-strip SCT endcap discs.

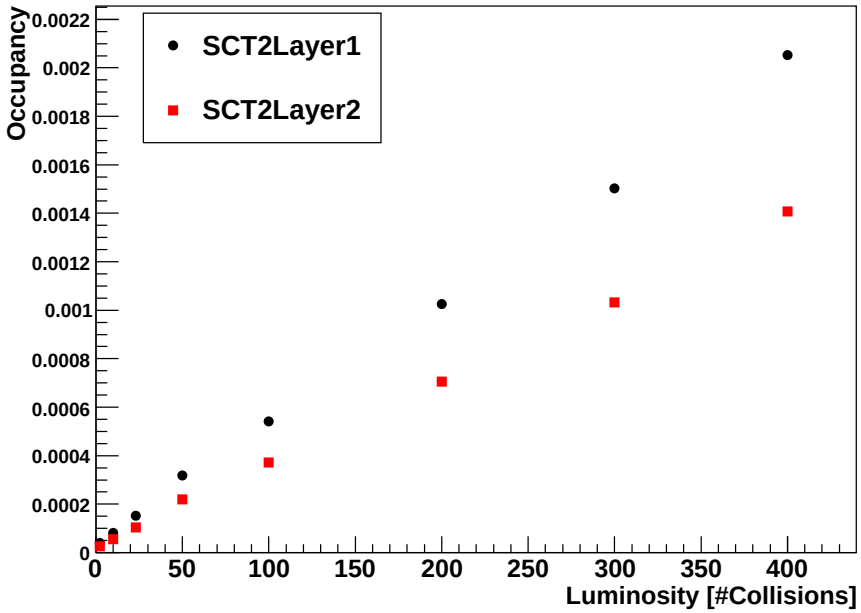


Figure 7.11: Hit occupancy as a function of pile-up in the two long-strip SCT barrel layers.

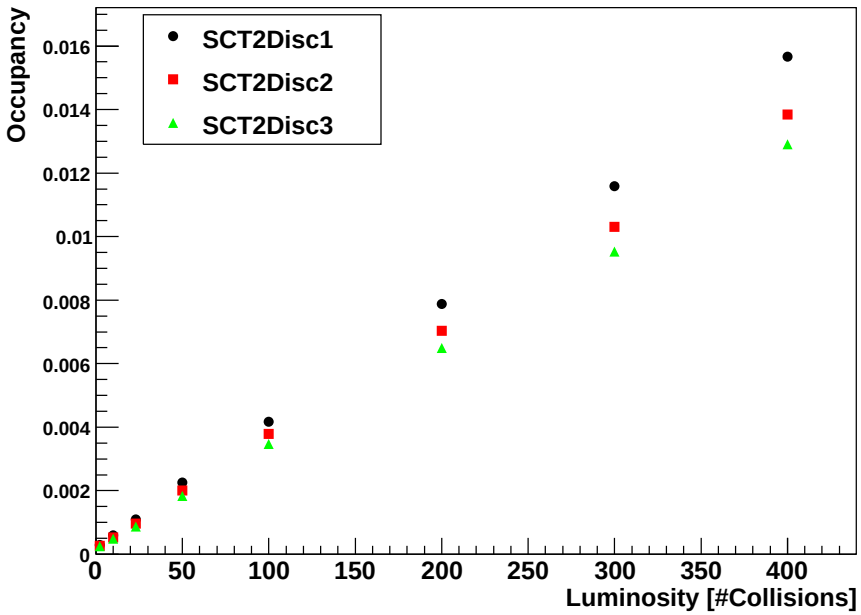


Figure 7.12: Hit occupancy as a function of pile-up in the three long-strip SCT endcap discs.

8 Summary

In this diploma thesis, several upgrade scenarios for the LHC collider and the ATLAS detector have been motivated and discussed. In order to be able to make decisions on the upgrade strategy, the Fast ATLAS Track Simulation software (Fatras) has been modified to enable the simulation of several layouts for the ATLAS Inner Tracker. Fatras has been described with a focus on recently added extensions. It represents a valuable tool for the comparison of the proposed silicon-only tracker upgrade layouts.

The validation of Fatras is an important step towards the completion of a fast simulation framework for the whole ATLAS detector. At the moment, the full simulation with Geant4 is still used for the ATLAS Tracker. Only by the employment of fast simulation methods for all parts of the detector, it will be possible to eventually produce sufficiently large background data samples for the analysis of many physics channels.

It has been shown that Fatras is performing well in the simulation of events with single electron and muon tracks in comparison to the full simulation with Geant4. The description of material interactions for hadrons still needs further improvement and tuning. This has been demonstrated with an exemplary study of single pion events within this thesis. More complex studies such as primary vertex reconstruction and the analysis of Z^0 decays indicate a reasonably good agreement with the full detector simulation for vertex resolutions and invariant mass spectra.

Significant contributions to the software development of Fatras have been made during the implementation of new detector geometry layouts, as well as to the new three-dimensional event visualisation tool for ATLAS, Virtual Point 1 (VP1).

For the very high luminosity environment of the Super-LHC, it is of vital importance to design a tracking device which is not only able to withstand the hostile radiation background in the vicinity of the interaction region, but is also able to provide very accurate tracking information to the event reconstruction.

The studies performed in this thesis were the first to simulate the complete silicon tracking detector for the proposed geometry layouts of the planned new ATLAS Inner Detector. It was possible to compare several layouts and draw preliminary conclusions in order to give advice on future research directions and layout modifications.

Two different tracker upgrade geometry layout proposals were studied in detail, namely the quasi-projective and the equal-length barrel “Strawman” layout. The results obtained indicate a preference for the quasi-projective geometry. Its main advantage is a higher average number of detector hits over the full acceptance range.

A Decay Tables for the `G4ParticleDecayCreator`

The following tables contain the decay channels active by default when running Fatras with the `G4ParticleDecay Algorithm`.

Name	Code	#Chan	Ratio	Products
μ^-	13	1		
			1.0	$e^-\ \overline{\nu}_\mu\ \nu_e$
τ^-	15	6		
			0.2541	$\pi^0\ \pi^-\ \nu_\tau$
			0.1784	$e^-\ \overline{\nu}_e\ \nu_\tau$
			0.1736	$\mu^-\ \overline{\nu}_\mu\ \nu_\tau$
			0.1106	$\pi^-\ \nu_\tau$
			0.0946	$\pi^-\ \pi^-\ \pi^+\ \nu_\tau$
			0.0917	$\pi^0\ \pi^0\ \pi^-\ \nu_\tau$

Table A.1: Preconfigured **lepton** decays produced by the `G4ParticleDecayCreator`.

Name	Code	#Chan	Ratio	Products
π^0	111	2		
			0.988	$\gamma\gamma$
			0.012	e^+e^-
π^+	211	1		
			1.0	$\mu^+\ \overline{\nu}_\mu$
η	221	4		
			0.3942	$\gamma\gamma$
			0.3256	$\pi^0\ \pi^0\ \pi^0$
			0.226	$\pi^0\ \pi^+\ \pi^-$
			0.0468	$\gamma\ \pi^+\ \pi^-$
η'	331	3		
			0.437	$\eta\ \pi^+\ \pi^-$
			0.302	$\rho^0\ \gamma$
			0.208	$\pi^0\ \pi^0$

Table A.2: Preconfigured **light meson** decays produced by the `G4ParticleDecayCreator`.

Name	Code	#Chan	Ratio	Products
K_L^0	130	6		
			0.2020	$\pi^-\ e^+\ \nu_e$
			0.2020	$\pi^+\ e^-\ \overline{\nu}_e$
			0.1983	$\pi^0\ \pi^0\ \pi^0$
			0.1348	$\pi^-\ \mu^+\ \nu_\mu$
			0.1348	$\pi^+\ \mu^-\ \overline{\nu}_\mu$
			0.1247	$\pi^0\ \pi^+\ \pi^-$
K_S^0	310	2		
			0.6895	$\pi^+\ \pi^-$
			0.3105	$\pi^0\ \pi^0$
K^0	311	2		
			0.5	K_L^0
			0.5	K_S^0
K^+	321	6		
			0.6339	$\mu^+\ \nu_\mu$
			0.2103	$\pi^+\ \pi^0$
			0.0559	$\pi^+\ \pi^+\ \pi^0$
			0.0493	$\pi^0\ e^+\ \nu_e$
			0.0330	$\pi^0\ \mu^+\ \nu_\mu$
			0.01757	$\pi^+\ \pi^0\ \pi^0$

Table A.3: Preconfigured **strange meson** decays produced by the G4ParticleDecayCreator.

Name	Code	#Chan	Ratio	Products
n	2112	1		
			1.0	p $e^-\ \overline{\nu}_e$
Σ^-	3112	1		
			1.0	n π^-
Λ	3122	2		
			0.639	p π^-
			0.358	n π^0
Σ^0	3212	1		
			1.0	$\Lambda\ \gamma$
Σ^+	3222	2		
			0.516	p π^0
			0.483	n π^+
Ξ^-	3312	1		
			1.0	$\Lambda\ \pi^-$
Ξ^0	3322	1		
			1.0	$\Lambda\ \pi^0$
Ω^-	3334	3		
			0.678	$\Lambda\ K^-$
			0.236	$\Xi^0\ \pi^-$
			0.086	$\Xi^-\ \pi^0$

Table A.4: Preconfigured **baryon** decays produced by the G4ParticleDecayCreator.

B Geometry Definition of the "Strawman" Layout

Listing B.1: FatrasDetDescrExample/SLHC_UpdateGeometry.py

```
################################################################################
#
# The geometry for the first inner tracker upgrade strawman layout
#
################################################################################

# import GenericGeometry
from FatrasDetDescrExample.GenericGeometry import *

# import SystemOfUnits and add useful shortcuts
from AthenaCommon.SystemOfUnits import *
um = micrometer
urad = mrad*1e-3

################################################################################
# Define materials
################################################################################
siMat = GenericMaterialDescriptor( name = "Silicon",
                                   thickness = 0.250,
                                   x0=9.36,
                                   A=28.0855,
                                   Z=14,
                                   rho=2.33*10e-3)

################################################################################
# Define module types
################################################################################

# the small pixel barrel module (320x320 chips)
innerPixBarrelMod = GenericRectangularModule( name = "innerPixBarrelMod",
                                              halflengthX = 8 * mm,
                                              halflengthY = 32 * mm,
                                              pitchX      = 50 * um,
                                              pitchY      = 0.2 * mm,
                                              thickness   = 1.2 * mm,
                                              clusterType = GenericPixel)

# the large pixel barrel module (320x160 chips)
outerPixBarrelMod = GenericRectangularModule( name = "outerPixBarrelMod",
                                              halflengthX = 8 * mm,
                                              halflengthY = 32 * mm,
                                              pitchX      = 50 * um,
                                              pitchY      = 0.4 * mm,
                                              thickness   = 1.2 * mm,
                                              clusterType = GenericPixel)
# the pixel endcap modules
pixEndcapMod1 = GenericTrapezoidalModule( name = "pixEndcapMod1",
                                          minHalflengthX = 2 * mm,
                                          maxHalflengthX = 8.5 * mm,
                                          halflengthY    = 107 * mm,
                                          pitchPhi       = 100*urad,
                                          pitchY         = 0.4 * mm,
                                          thickness      = 1.2 * mm,
```

```
                                        clusterType    = GenericPixel)

pixEndcapMod2 = GenericTrapezoidalModule( name = "pixEndcapMod2",
                                        minHalflengthX  = 2.5 * mm,
                                        maxHalflengthX  = 8 * mm,
                                        halflengthY     = 96.5 * mm,
                                        pitchPhi        = 100*urad,
                                        pitchY          = 0.4 * mm,
                                        thickness       = 1.2 * mm,
                                        clusterType     = GenericPixel)

pixEndcapMod3 = GenericTrapezoidalModule( name = "pixEndcapMod3",
                                        minHalflengthX = 4 * mm,
                                        maxHalflengthX = 10.5 * mm,
                                        halflengthY    = 86.5 * mm,
                                        pitchPhi       = 100*urad,
                                        pitchY         = 0.4 * mm,
                                        thickness      = 1.2 * mm,
                                        clusterType    = GenericPixel)

# the sct shortstrip barrel module
sct1BarrelMod = GenericRectangularModule( name = "sct1BarrelMod",
                                        halflengthX = 43 * mm,
                                        halflengthY = 17.5 * mm,
                                        pitchX      = 80 * um,
                                        pitchY      = 35 * mm,
                                        thickness   = 1.2 * mm,
                                        clusterType = GenericStrip)


# the sct longstrip barrel module
sct2BarrelMod = GenericRectangularModule( name = "sct2BarrelMod4",
                                        halflengthX = 51.5 * mm,
                                        halflengthY = 45 * mm,
                                        pitchX      = 80 * um,
                                        pitchY      = 90 * mm,
                                        thickness   = 1.2 * mm,
                                        clusterType = GenericStrip)


# SS endcap modules
sctEndcapMod1 = GenericTrapezoidalModule( name = "sctEndcapMod1",
                                        minHalflengthX = 7.7751/2. * cm,
                                        maxHalflengthX = 9.7224/2. * cm,
                                        halflengthY    = 4.8947/2. * cm,
                                        pitchPhi       = 161.5 * urad,
                                        pitchY         = 4.8947*cm,
                                        thickness      = 1.2 * mm,
                                        clusterType    = GenericStrip)

sctEndcapMod2 = GenericTrapezoidalModule( name = "sctEndcapMod2",
                                        minHalflengthX = 9.6030/2. * cm,
                                        maxHalflengthX = 12.2599/2. * cm,
                                        halflengthY    = 6.6786/2. * cm,
                                        pitchPhi       = 161.5 * urad,
                                        pitchY         = 6.6786 * cm,
                                        thickness      = 1.2 * mm,
                                        clusterType    = GenericStrip)

sctEndcapMod3 = GenericTrapezoidalModule( name = "sctEndcapMod3",
                                        minHalflengthX = 6.1608/2. * cm,
                                        maxHalflengthX = 8.3657/2. * cm,
                                        halflengthY    = 11.1935/2. * cm,
```

```
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 11.1935 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

sctEndcapMod4 = GenericTrapezoidalModule( name = "sctEndcapMod4",
                                          minHalflengthX = 8.3066/2. * cm,
                                          maxHalflengthX = 10.2611/2. * cm,
                                          halflengthY    = 9.9223/2. * cm,
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 9.9223 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

sctEndcapMod5 = GenericTrapezoidalModule( name = "sctEndcapMod5",
                                          minHalflengthX = 10.2020/2. * cm,
                                          maxHalflengthX = 11.8190/2. * cm,
                                          halflengthY    = 8.2085/2. * cm,
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 8.2085 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

# LS endcap modules
sctEndcapMod6 = GenericTrapezoidalModule( name = "sctEndcapMod6",
                                          minHalflengthX = 5.9166/2. * cm,
                                          maxHalflengthX = 7.1101/2. * cm,
                                          halflengthY    = 12.1473/2. * cm,
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 12.1473 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

sctEndcapMod7 = GenericTrapezoidalModule( name = "sctEndcapMod7",
                                          minHalflengthX = 7.0807/2. * cm,
                                          maxHalflengthX = 8.2076/2. * cm,
                                          halflengthY    = 11.4695/2. * cm,
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 11.4695 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

sctEndcapMod8 = GenericTrapezoidalModule( name = "sctEndcapMod8",
                                          minHalflengthX = 8.1781/2. * cm,
                                          maxHalflengthX = 9.2282/2. * cm,
                                          halflengthY    = 10.6881/2. * cm,
                                          pitchPhi       = 161.5 * urad,
                                          pitchY         = 10.6881 * cm,
                                          thickness      = 1.2 * mm,
                                          clusterType    = GenericStrip)

##############################################################################
# Define endcap rings
##############################################################################

# Pixels
pixelRing1 = GenericEndcapRing( "PixRing1",
                                module = pixEndcapMod1,
                                sectorsPhi = 104,
                                innerR = 66 * mm,
                                outerR = 280 * mm)

pixelRing2 = GenericEndcapRing( "PixRing2",
```

```
                                module = pixEndcapMod2,
                                sectorsPhi = 110,
                                innerR = 87 * mm,
                                outerR = 280 * mm)

pixelRing3 = GenericEndcapRing( "PixRing3",
                                module = pixEndcapMod3,
                                sectorsPhi = 84,
                                innerR = 107 * mm,
                                outerR = 280 * mm)

# SCT
sctRing1 = GenericEndcapRing( "SCTRing1",
                              module = sctEndcapMod1,
                              sectorsPhi = 16,
                              innerR     = 24.4389 * cm - 2*sctEndcapMod1.halflengthY,
                              outerR     = 24.4389 * cm)

sctRing2 = GenericEndcapRing( "SCTRing2",
                              module = sctEndcapMod2,
                              sectorsPhi = 16,
                              innerR = sctRing1.outerR,
                              outerR = 30.8175 * cm)

sctRing3 = GenericEndcapRing( "SCTRing3",
                              module = sctEndcapMod3,
                              sectorsPhi = 32,
                              innerR = sctRing2.outerR,
                              outerR = 42.4692 * cm,
                              rotation = math.pi / 32)

sctRing4 = GenericEndcapRing( "SCTRing4",
                              module = sctEndcapMod4,
                              sectorsPhi = 32,
                              innerR = sctRing3.outerR,
                              outerR = 52.0915 * cm,
                              rotation = math.pi / 32)

sctRing5 = GenericEndcapRing( "SCTRing5",
                              module = sctEndcapMod5,
                              sectorsPhi = 32,
                              innerR = sctRing4.outerR,
                              outerR = 60. * cm,
                              rotation = math.pi / 32)

sctRing6 = GenericEndcapRing( "SCTRing6",
                              module = sctEndcapMod6,
                              sectorsPhi = 64,
                              innerR = sctRing5.outerR,
                              outerR = 72.3650 * cm,
                              rotation = math.pi / 64)

sctRing7 = GenericEndcapRing( "SCTRing7",
                              module = sctEndcapMod7,
                              sectorsPhi = 64,
                              innerR = sctRing6.outerR,
                              outerR = 83.5345 * cm,
                              rotation = math.pi / 64)

sctRing8 = GenericEndcapRing( "SCTRing8",
                              module = sctEndcapMod8,
                              sectorsPhi = 64,
                              innerR = sctRing7.outerR,
```

```
                                outerR = 93.9226 * cm,
                                rotation = math.pi / 64)


################################################################################
# Build pixel barrel layers
################################################################################

# the b layer
pixLayerB = GenericBarrelLayer( name       = "PixLayerB",
                                radius     = 50 * mm,
                                halflength = 400 * mm,
                                module     = innerPixBarrelMod,
                                material   = siMat,
                                sectorsPhi = 20,
                                sectorsZ   = 13,
                                tiltAngle  = 0.174533 * rad,)

# pixel layer 1
pixLayer1 = GenericBarrelLayer( name       = "PixLayer1",
                                radius     = 120 * mm,
                                halflength = 400 * mm,
                                module     = innerPixBarrelMod,
                                material   = siMat,
                                sectorsPhi = 48,
                                sectorsZ   = 13,
                                tiltAngle  = 0.174533 * rad)

# pixel layer 2
pixLayer2 = GenericBarrelLayer( name       = "PixLayer2",
                                radius     = 180 * mm,
                                halflength = 400 * mm,
                                module     = outerPixBarrelMod,
                                material   = siMat,
                                sectorsPhi = 72,
                                sectorsZ   = 13,
                                tiltAngle  = 0.174533 * rad)

# pixel layer 3
pixLayer3 = GenericBarrelLayer( name       = "PixLayer3",
                                radius     = 240 * mm,
                                halflength = 400 * mm,
                                module     = outerPixBarrelMod,
                                material   = siMat,
                                sectorsPhi = 96,
                                sectorsZ   = 13,
                                tiltAngle  = 0.174533 * rad)

################################################################################
# Build pixel endcap layers
################################################################################

# pixel disc 1
pixDisc1 = GenericEndcapDisc( name     = "PixDisc1",
                              material = siMat,
                              zPos     = 500*mm)
pixDisc1.addRing( pixelRing1)

# pixel disc 2
pixDisc2 = GenericEndcapDisc( name     = "PixDisc2",
                              material = siMat,
                              zPos     = 625*mm)
pixDisc2.addRing( pixelRing2)
```

```
# pixel disc 3
pixDisc3 = GenericEndcapDisc( name     = "PixDisc3",
                              material = siMat,
                              zPos     = 750*mm)

pixDisc3.addRing( pixelRing3)

##############################################################################
# Build short-strip barrel layers
##############################################################################

# inner sct layer 1
sct1Layer1 = GenericBarrelLayer( name       = "SCT1Layer1",
                                 radius     = 320 * mm,
                                 halflength = 1000 * mm,
                                 module     = sct1BarrelMod,
                                 material   = siMat,
                                 sectorsPhi = 24,
                                 sectorsZ   = 58,
                                 tiltAngle  = 0.174533 * rad)

# inner sct layer 2
sct1Layer2 = GenericBarrelLayer( name       = "SCT1Layer2",
                                 radius     = 460 * mm,
                                 halflength = 1000 * mm,
                                 module     = sct1BarrelMod,
                                 material   = siMat,
                                 sectorsPhi = 34,
                                 sectorsZ   = 58,
                                 tiltAngle  = 0.174533 * rad)

# inner sct layer 3
sct1Layer3 = GenericBarrelLayer( name       = "SCT1Layer3",
                                 radius     = 600 * mm,
                                 halflength = 1000 * mm,
                                 module     = sct1BarrelMod,
                                 material   = siMat,
                                 sectorsPhi = 46,
                                 sectorsZ   = 58,
                                 tiltAngle  = 0.174533 * rad)

##############################################################################
# Build short-strip endcap layers
##############################################################################

# inner sct endcap 1
sct1Disc1 = GenericEndcapDisc( name     = "SCT1Disc1",
                               material = siMat,
                               zPos     = 128.25 * cm)

sct1Disc1.addRing( sctRing1)
sct1Disc1.addRing( sctRing2)
sct1Disc1.addRing( sctRing3)
sct1Disc1.addRing( sctRing4)
sct1Disc1.addRing( sctRing5)

# inner sct endcap 2
sct1Disc2 = GenericEndcapDisc( name     = "SCT1Disc2",
                               material = siMat,
                               zPos     = 156.04 * cm)

sct1Disc2.addRing( sctRing2)
```

```
sct1Disc2.addRing( sctRing3)
sct1Disc2.addRing( sctRing4)
sct1Disc2.addRing( sctRing5)

################################################################################
# Build long-strip barrel layers
################################################################################

# outer sct layer 1
sct2Layer1 = GenericBarrelLayer( name       = "SCT2Layer1",
                                 radius     = 750 * mm,
                                 halflength = 1900 * mm,
                                 module     = sct2BarrelMod,
                                 material   = siMat,
                                 sectorsPhi = 46,
                                 sectorsZ   = 43,
                                 tiltAngle  = 0.174533 * rad)

# outer sct layer 2
sct2Layer2 = GenericBarrelLayer( name       = "SCT2Layer2",
                                 radius     = 950 * mm,
                                 halflength = 1900 * mm,
                                 module     = sct2BarrelMod,
                                 material   = siMat,
                                 sectorsPhi = 58,
                                 sectorsZ   = 43,
                                 tiltAngle  = 0.174533 * rad)

################################################################################
# Build long-strip endcap layers
################################################################################

# outer sct disc 1
sct2Disc1 = GenericEndcapDisc( name     = "SCT2Disc1",
                               material = siMat,
                               zPos     = 199.22 * cm)

sct2Disc1.addRing( sctRing3)
sct2Disc1.addRing( sctRing4)
sct2Disc1.addRing( sctRing5)
sct2Disc1.addRing( sctRing6)
sct2Disc1.addRing( sctRing7)
sct2Disc1.addRing( sctRing8)

# outer sct disc 2
sct2Disc2 = GenericEndcapDisc( name     = "SCT2Disc2",
                               material = siMat,
                               zPos     = 265.13 * cm)
sct2Disc2.addRing( sctRing4)
sct2Disc2.addRing( sctRing5)
sct2Disc2.addRing( sctRing6)
sct2Disc2.addRing( sctRing7)
sct2Disc2.addRing( sctRing8)

# outer sct disc 3
sct2Disc3 = GenericEndcapDisc( name     = "SCT2Disc3",
                               material = siMat,
                               zPos     = 323.35 * cm)
sct2Disc3.addRing( sctRing5)
sct2Disc3.addRing( sctRing6)
sct2Disc3.addRing( sctRing7)
sct2Disc3.addRing( sctRing8)
```

```
################################################################################
# Define volumes
################################################################################

# the beampipe
beampipe = GenericBeamPipe()

# the pixel detector
pixDet = GenericDetectorVolume( name  = "Pixel",
                                color = 8)

# the inner sct detector (mostly short strips)
sct1Det = GenericDetectorVolume( name  = "SCT1",
                                 color = 2)

# the outer sct detector (mostly long strips)
sct2Det = GenericDetectorVolume( name  = "SCT2",
                                 color = 4)

################################################################################
# Build geometry
################################################################################

# combine pixel detector
pixDet.addBarrelLayer( pixLayerB)
pixDet.addBarrelLayer( pixLayer1)
pixDet.addBarrelLayer( pixLayer2)
pixDet.addBarrelLayer( pixLayer3)
pixDet.addEndcapDisc( pixDisc1)
pixDet.addEndcapDisc( pixDisc2)
pixDet.addEndcapDisc( pixDisc3)

# combine first SCT
sct1Det.addBarrelLayer( sct1Layer1)
sct1Det.addBarrelLayer( sct1Layer2)
sct1Det.addBarrelLayer( sct1Layer3)
sct1Det.addEndcapDisc( sct1Disc1)
sct1Det.addEndcapDisc( sct1Disc2)

# combine second SCT
sct2Det.addBarrelLayer( sct2Layer1)
sct2Det.addBarrelLayer( sct2Layer2)
sct2Det.addEndcapDisc( sct2Disc1)
sct2Det.addEndcapDisc( sct2Disc2)
sct2Det.addEndcapDisc( sct2Disc3)

# create the top level geometry object
strawman = GenericGeometry( "Strawman1")

# combine volumes to geometry
strawman.addBeampipe( beampipe)
strawman.addVolume( pixDet)
strawman.addVolume( sct1Det)
strawman.addVolume( sct2Det)

# c++ transition if athena is running
if 'AthenaCommon' in dir():
    strawman.registerGeometry()
```

References

[1] P. Lefèvre and T. Pettersson, The Large Hadron Collider: conceptual study, 1995, CERN/AC/95-05.

[2] O. Brüning and P. Collier, *Nature* **448**, 285 (2007).

[3] N. Ahmad *et al.*, A Large Ion Collider Experiment - Technical Proposal, 1995, CERN-LHCC-95-71.

[4] The ATLAS Collaboration, ATLAS: Technical proposal for a general-purpose p p experiment at the Large Hadron Collider at CERN, 1994, CERN-LHCC-94-43.

[5] The CMS Collaboration, Technical Proposal, 1994, CERN-LHCC-94-38.

[6] S. Amato *et al.*, LHCb Technical Proposal, 1998, CERN-LHCC-98-4.

[7] D. Froidevaux and P. Sphicas, *Annu. Rev. Nucl. Part. Sci.* **56**, 375 (2006).

[8] LHC Machine Outreach, `http://http://lhc-machine-outreach.web.cern.ch/`.

[9] The ATLAS Collaboration, *ATLAS Letter of Intent for a General-Purpose pp Experiment at the Large Hadron Collider at CERN*, Letter of Intent, CERN, Geneva, 1992, CERN/LHCC/92–4, LHCC/I2.

[10] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, 2008, submitted to JINST.

[11] The ATLAS Inner Detector Collaboration, ATLAS Inner Detector Technical Design Report, 1997, ATLAS TDR–4.

[12] M. Gilchriese (editor), *JINST* **A88**, 888 (2007).

[13] A. Ahmad *et al.*, *Nucl. Instr. Meth.* **A578**, 98 (2007).

[14] S. Abdullin *et al.*, Physics Potential and Experimental Challenges of the LHC Luminosity Upgrade, 2002, CERN-TH/2002-078, hep-ph/0204087.

[15] O. Brüning *et al.*, LHC Luminosity and Energy Upgrade: A Feasibility Study, 2003, LHC-Project-Report-626, CERN.

[16] A. De Roeck, J. Ellis, and F. Gianotti, Physics Motivations for Future CERN Accelerators, 2001, CERN-TH/2001-023, hep-ex/0112004.

[17] G. Azuelos, *J. Phys. G* **28**, 2453 (2002).

[18] F. Zimmermann, LHC Upgrade Scenarios, 2007, CARE-Conf-07-020-HHH.

[19] R. Garoby and J. Ellis, Re: Towards a preferred bunch-spacing scenario for the SLHC, 2006, memo to R. Aymar and J. Engelen, available from `http://pofpa.web.cern.ch/pofpa/LHC_bunch_spacing.doc`.

[20] I. Dawson, Radiation predictions at the SLHC and irradiation facilities, 2006, Talk given at ATLAS Tracker Upgrade Workshop, Liverpool.

[21] S. I. Parker, C. J. Kenney, and J. Segal, *Nucl. Instr. Meth.* **A395**, 328 (1997).

[22] S. Eckert *et al.*, *Nucl. Instr. Meth.* **A581**, 322 (2007).

[23] N. Hessey and J. Tseng, Layout Requirements and Options for a new Inner Tracker for the ATLAS Upgrade, `https://edms.cern.ch/document/809071/2/`, 2006, Rev. F, as of April 10, 2008.

[24] ATLAS Computing Group, ATLAS Computing Technical Design Report, Technical report, CERN-LHCC-2005-022, 2005, ATLAS TDR–017.

[25] G. Barrand *et al.*, *Comp. Phys. Comm.* **140**, 45 (2001).

[26] GLAST - The Gamma-ray Large Area Space Telescope, `http://harp.web.cern.ch/harp/`, as of February, 25 2008.

[27] HARP (PS214) - The Hadron Production Experiment at the PS, `http://harp.web.cern.ch/harp/`, as of February, 25 2008.

[28] CLHEP - A Class Library for High Energy Physics, `http://proj-clhep.web.cern.ch/proj-clhep/`, as of February, 25 2008.

[29] T. Sjostrand, S. Mrenna, and P. Skands, *JHEP* **0605**, 026 (2006).

[30] G. Corcella *et al.*, HERWIG 6.5 Release Note, 2002, hep-ph/0210213.

[31] S. Agostinelli *et al.*, *Nucl. Instr. Meth.* **A506**, 250 (2003).

[32] Atlfast, `http://www.hep.ucl.ac.uk/atlas/atlfast/`, as of April, 10 2008.

[33] GEANT - Detector Description and Simulation Tool, http://wwwasd.web.cern.ch/wwwasd/geant/, as of April, 10 2008.

[34] A. Salzburger, The ATLAS Tracking Geometry Description, 2007, ATLAS Note, ATL-COM-SOFT-2007-009.

[35] P. F. Akesson *et al.*, ATLAS Tracking Event Data Model, 2006, ATLAS Public Note, ATL-SOFT-PUB-2006-004.

[36] A. Salzburger, The ATLAS Extrapolation package, 2007, ATLAS Note, ATL-COM-SOFT-2007-01.

[37] A. Salzburger, The New Fast ATLAS Track Simulation (FATRAS), talk given at CHEP, 2006.

[38] M. Dührssen, The Method of the Fast Calorimeter Simulation FastCaloSim, ATLAS Note in preparation.

[39] 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, 2008.

[40] Atlantis, `http://www.cern.ch/atlantis/`, as of April, 10 2008.

[41] ROOT, `http://root.cern.ch/`, as of April, 10 2008.

[42] Virtual Point 1 (VP1), `http://atlas-vp1.web.cern.ch/atlas-vp1/`, as of April, 10 2008.

[43] V. L. Highland, *Nucl. Instr. Meth.* **129**, 497 (1975).

[44] W.-M. Yao *et al.*, *Journal of Physics G: Nuclear and Particle Physics* **33**, 1 (2006).

[45] L. Landau, *J. Phys. USSR* **8** (1944).

[46] W. Leo, *Techniques for Nuclear and Particle Physics Experiments*, Springer, 1994.

[47] H. Bethe and W. Heitler, *Proc. Roy. Soc. Lond.* **A146**, 83 (1934).

[48] R. Frühwirth and A. Strandlie, Track finding and fitting with the Gaussian-sum Filter, Proc. of CHEP, 1998, `http://www.hep.net/chep98/PDF/10.pdf`.

[49] Qt - Trolltech, `http://trolltech.com/products/qt/`, as of April, 10 2008.

[50] Coin3D - 3D Graphics Development Tools, `http://www.coin3d.org/`, as of April, 10 2008.

[51] G. Piacquadio, K. Prokofiev, and A. Wildauer, Primary vertex reconstruction in the ATLAS experiment at LHC, Proc. of CHEP, 2007, paper in preparation.

List of Figures

List of Tables

Danksagung

An erster Stelle möchte ich allen danken, die mich beim Erstellen dieser Arbeit unterstützt haben, im Besonderen meinen Eltern und meiner Familie.

Mein Dank gilt Prof. Dr. Karl Jakobs für die Ermöglichung der Arbeit über dieses interessante Thema.

Meinem Betreuer Dr. Ulrich Parzefall danke ich besonders für die Begleitung auf dem Weg zur Vollendung der Diplomarbeit.

Ganz besonderen Dank möchte ich Dr. Ralf Bernhard für interessante Diskussionen sowohl über fachliche als auch private Themen aussprechen.

Ein grosser Dank gebührt Andreas Salzburger und den anderen Fatras-Entwicklern, die durch die Bereitstellung der Simulationssoftware viel zur Entstehung dieser Arbeit beigetragen haben.

Danken möchte ich Dr. Christian Weiser und Nicola Giacinto Piacquadio, unseren lokalen Trackingspezialisten, für die fachmännische Beratung und Einführung in die (Un-)Tiefen der Vertex Rekonstruktion.

Simon Eckert und Ingo Torchiani danke ich für die Gesellschaft im Lehrstuhl an einsamen Wochenenden und Nächten, sowie meinen Büronachbarn Henrik Nilsen und Sascha Thoma während des Rests der Woche.

Michael Dührssen und im besonderen Inga Ludwig danke ich für das Korrekturlesen.

Ich bedanke mich bei Susanne Kühn und Evelyn Schmidt für die moralische Unterstützung und die Bereitstellung von Süssigkeiten.

Allen anderen, ungenannten Freunden aus meiner Vergangenheit und Gegenwart danke ich an dieser Stelle für die schönen Zeiten, die wir gemeinsam verbracht haben.

Zusammenfassung

Für die geplante Modifikation des Large Hadron Collider (LHC) am CERN, Genf für hohe Luminositäten, die für Mitte des kommenden Jahrzehnts geplant ist, muss der Innere Detektor des ATLAS Experiments gänzlich ausgetauscht werden. Gegenwärtige Pläne sehen einen ausschliesslichen Einsatz von Siliziumsensoren vor. Für die detaillierte Implementation der Detektorgeometrie existieren mehrere Vorschläge, zwischen denen es sich zu entscheiden gilt.

Die *Fast ATLAS Track Simulation* (Fatras) wurde im Rahmen der vorliegenden Diplomarbeit erweitert. Dadurch wurden auch Untersuchungen verschiedener Detektorgeometrien für das Upgrade des Inneren Detektors von ATLAS möglich.

Die Leistungsfähigkeit der Simulation wurde anhand des existierenden Detektors überprüft. Darüber hinaus wurden zwei Vorschläge für eine neue Detektorgeometrie untersucht und erste Ergebnisse präsentiert.

Erklärung

Hiermit erkläre ich, dass ich die vorgelegte Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Freiburg, den 10. April 2008

Jörg Mechnich