Track Simulation and Reconstruction in the ATLAS Experiment

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

an der

Fakultät für Mathematik, Informatik und Physik

der

Leopold-Franzens-Universität Innsbruck

eingereicht von

Mag. rer. nat. Andreas Salzburger

im März 2008

Track Simulation and Reconstruction in the ATLAS Experiment

Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

an der

Fakultät für Mathematik, Informatik und Physik

der

Leopold-Franzens-Universität Innsbruck

eingereicht von

Mag. rer. nat. Andreas Salzburger

im März 2008

Abstract - *Zusammenfassung*

The reconstruction and simulation of particle trajectories is an inevitable part of the analysis strategies for data taken with the ATLAS detector. Many aspects and necessary parts of a high-quality track reconstruction will be presented and discussed in this work. At first, the technical realisation of the data model and the reconstruction geometry will be given; the reconstruction geometry is characterised by a newly developed navigation model and an automated procedure for the synchronisation of the detailed simulation geometry description with the simplified reconstruction geometry model, which allows a precise description of the tracker material in track reconstruction. Both components help the coherent and fast integration of material effects in a newly established track extrapolation package, that is discussed in the following. The extrapolation engine enables a highly precise transport of the track parameterisation and the associated covariances through the complex magnetic field and the detector material. As a direct consequence, the track parameter resolution could be improved — in particular at the low momentum regime where material effects are most pronounced. This will be shown in the context of the new track reconstruction chain that is presented together with an extensive validation section.

The intrinsic navigation model and modular design of the extrapolation engine are also key aspects of a new fast Monte Carlo track simulation engine that is described in the second part of this thesis. It includes the most relevant physics processes, such as the integration of random multiple scattering and energy loss effects when the particle traverses the detector; also particle decay and the simulation of photon conversions are included in the new fast simulation. It reaches high compatibility with the full simulation, while decreasing the execution time per event by about 2 orders of magnitudes.

Die Rekonstruktion und Simulation von Teilchenspuren ist ein unumgänglicher Bestandteil der Analyse von Ereignissen im ATLAS Detektor. Viele Aspekte und notwendige Bestandteile für eine hochqualitative Spurenrekonstruktion werden in dieser Arbeit vorgestellt und ausgiebig diskutiert. Ausgehend von der technischen Realisierung des Datenmodells und der Rekonstruktionsgeometrie, wird ein Extrapolationspacket beschrieben das einen Grundbaustein der gesamten neuen Spurenrekonstruktionssoftware des ATLAS Detektors darstellt. Im Zusammenhang mit der Rekonstruktionsgeometrie wird ein neuentwickeltes Navigationsmodell und ein Verfahren zur Synchronisierung der Materialbeschreibung des Detektors zwischen der sehr detailgenauen Simulations- und der vereinfachten Rekonstruktionsgeometrie dargelegt. Diese beiden Komponenten verhelfen zu einem schnellen aber dennoch höchst präzisen Trasport der Spurparameter und deren Fehlerkomonenten, dessen Validierung ebenfalls in der vorgelegten Arbeit enthalten ist. Eine direkte Konsequenz der verbesserten Materialbeschreibung ist eine erhöhte Spurenauflösung für niederenergetische Teilchen, was zusammen mit einer generellen Beschreibung der Integration dieser neuen Komponenten in die Rekonstruktionssoftware dargelegt wird.

Ebenfalls aufbauend auf dem Navigationsmodell der Rekonstruktionsgeometrie und dem Extrapolationsmodul wurde eine vollkommen neue schnelle Simulation entwickelt, die ausführlich im letzten Teil der vorgelegten Arbeit beschrieben wird. Diese Monte Carlo Simulation integriert die meistrelevanten physikalischen Prozesse, wie z. B. Vielfachstreuung, Energieverlust durch Ionisierung und Bremstrahlung, aber auch Photonkonversionen und Teilchenzerfall. Sie zeichnet sich bei etwa hundertfach gesteigerter Simulationsfrequenz durch eine äusserst genau Übereinstimmung mit Daten von der vollen Simulation aus.

Acknowledgements - *Danksagung*

I would like to thank at first place my two supervisors Prof. Dr. Dietmar Kuhn in Innsbruck, and my supervisor at CERN, Dr. Markus Elsing. I appreciate their great support and advice over the years. I am in particular thankful for their confidence to grant me enough freedom for developing my own ideas, while keeping an eye on my work at the same time.

I had the chance to work closely with many interesting persons on the various projects that are presented in this thesis, and it is impossible to name everyone independently. Some of their names appear on the presented documents or are given within the context; it was a great pleasure to learn from all of them and to have many fruitful discussions.

Dr. Edward Moyse, however, should be mentioned here individually not only for his great technical support and the many ways out of the C++ jungle he has taught me, but also for sharing an office and the great time we had. Dr. Wolfgang Liebig should be mentioned as well for his careful reading of many draft versions of the presented ATLAS notes; his comments have been useful guidelines to improve the documents and his remarks have been welcome input to many parts of the presented projects. Are Strandlie should not be forgotten, he has been a big supporter and always a critical mind in the best sense.

The former and present members of the high energy physics group at the university of Innsbruck should also find their place here; although having spent the last years at CERN, there have been always strong bounds with my home university and the people of my institute.

Den Dank an meine Eltern kann ich schlecht in Worte fassen; wie kann man für all das danken? Vielleicht einzig mit Demut und dem Stolz, Euer Sohn zu sein. Auch meine Brüder will ich hier nicht vergessen, zu denen beiden ich gerne aufblicke.

Meine rote Anna Blume! Danke, dass du in meinem Garten blühst.

Geneva, 20^{th} of March, 2008

Contents

And so they are
like fatal pillars
of Hercules to the sciences;
for they are not stirred
by the desire or hope
of going further.

Francis Bacon -
The New Oragnon

Chapter 1

Introduction

Exploring the unknown has always been a driving force of mankind, a mirror of our curiosity. In ancient greece, the rock of Gibraltar, which marks the most southern tip of the peninsula — at present time still a source of conflict between the United Kingdom and Spain — and the Monte Hacho in Ceuta[1], North of Morocco, have been widely regarded as the end of the known world. Also called *The Pillars of Hercules*, this natural barrier between the Mediterranean Sea and the Atlantic Ocean became a symbol for the gate to the unexplored land, the open door for only the curious ones; to pass for those who want to look further. On a clear day — and given the right instruments — one might be able to see through this gate from one of the many tops of the ATLAS mountain massive, that covers half of north Africa, spreading over Tunisia, Algeria and Morocco itself.

Nowadays, when satellites are able to map every inch of our planet, and great explorations have changed from the discoveries of new lands to fundamental research in many different scientific aspects, the ATLAS experiment will serve us as our instrument to look through this virtual gate; in the sense of the explorers of the past, driven by curiosity and the human thirst of finding answers to our questions, it is built to push this border further away to the next questions that will certainly arise from our findings. Although our way of exploration might not be as thrilling as a ship full of argonauts setting off from their homeland to unknown adventures, the discoveries should pay back in excitement for whatsoever effort has been put into this project.

1.1 Structure of this Thesis

The reader will soon notice that main parts of this thesis are included as ATLAS public notes, that are available from the *CERN Document Server* (CDS) [1.1]. This is in the full spirit of working as a part of a big collaboration like the ATLAS experiment, where the communication and documentation of the individual contributions became a necessary, but sometimes burdensome task. The software applications, a substantial part of the presented work, require in particular a detailed documentation on several levels, mainly due to the huge number of clients and users that interact on a daily basis with the software framework of the experiment. In addition, it should also facilitate further developments and modifications — preparing it to achieve maximum performance for the future measurements with the ATLAS detector. The reader will also learn that the presented work expands into various fields and that some of the described modules have been an outcome of a collaborated

[1]There is, however, a discussion amongst historians whether this attribute has to be granted to the Mount Sidi or some other rock formation along the northern coast of Morocco.

effort. To avoid ambiguities, a small summary of the author's individual contribution to the presented topics will be given adherently to the included documents, either as an explicit list of contributions or in form of acknowledgments to the co-contributors.

The ATLAS experiment is amongst the most challenging projects of modern high energy physics, but also expands into various other fields that do not seem to be obviously related at first sight. This document presents the challenges and some of their solutions for the ATLAS experiment with dedicated focus on one particular aspect: the reconstruction and simulation of particle trajectories in the tracking devices of the ATLAS detector. Track reconstruction is indispensable for any event analysis, not only for the final determination of the event topology and the kinematic quantities of the involved particles, but already at earlier stages of the event triggering. Finding the trajectories of the particles is the only way to determine their production origin (i.e. in the terminology of high energy physics, their *vertex*), reconstruct decay chains or jet axes from the fragmentation process. It also plays an important role in particle identification, either through the trajectory itself such as in combined muon reconstruction or electron reconstruction, or through decay vertex information in heavy quark tagging.

Chapter 2 presents — starting from the pure scientific intention, the search for the fundamental structure of our world and the relation between the smallest constituents that build (to our current knowledge) everything around us — a short review of the most common theories in high energy physics in conjunction with some extended models and predicted measurements. It also describes the main features of the LHC accelerator that will open the kinematic window for these future searches and measurements with the ATLAS detector. It discusses the consequences of both, the event signatures of desired measurements and the constraints imposed by the LHC machine on track reconstruction; this is done on two levels: the pure algorithmic challenge to cope with the huge amount of input data, but also the requirements for a functional reconstruction framework.

Chapter 3 describes the instrumentation of the complex ATLAS detector, in other words, the focus from the physics challenge will be drawn onto the technical and experimental realisation of the project. In addition, in Sec. 3.2, the attention will be put on the processing of the data taken with the given detector setup. This includes the (partly software based) trigger that is aimed to select *interesting* events from overwhelming background, the actual event reconstruction and the analysis of the reconstructed data. The processing of the ATLAS event data will be also described in the context of grid computing to complete the full picture of the experimental aspects of modern high energy physics.

Chapter 4 will give a review of track reconstruction techniques and will in particular emphasise on track fitting algorithms. In the following, Chap. 5 to Chap. 7 will describe the realisation of some main modules of track reconstruction: the geometry, the event data model and the track extrapolation engine in the ATLAS offline reconstruction software. The complexity of modern track reconstruction will be hereby shown in a remarkable way: single equations that have been introduced and discussed in the context of Chap. 4 turn out to require complex technical solutions when being deployed in the actual reconstruction software chain, where many additional aspects such as reliability, execution speed or disk storage and memory usage have to be considered.

Chapter 8 will then combine the presented modules in the context of a newly established modular track reconstruction chain. A small review of the ATLAS software framework ATHENA is embedded in this section to emphasise some of the additional constraints mentioned earlier. An overview on the performance of the new track reconstruction is also given in this section, a work that has been carried out for the recently finished document that describes the deployed ATLAS detector and its performance in a final review before

data taking starts in 2008 [1.2].

In Chap. 9, a new fast track simulation program will be presented. The establishment of such a fast track simulation technique has been completely new in the ATLAS experiment, and has only be possible through the precedent careful design of the reconstruction geometry, the event data model and the extrapolation engine. It is able to create look-alike input for the standard track reconstruction and combines the work described in the previous Chaps. 5 to 8 in a — to the eyes of the author — beautiful way. An extensive comparison of the new fast track simulation to the existing simulation programs will be included in this chapter.

Finally, Chap. 10 will conclude the document and give an outlook on further modifications and adaptions of the presented work. Latter is, because it is of no doubt for the author that the simulation and reconstruction applications are moving targets and are matter of such necessary modifications. The start of the data taking period at the end of this year will build the most stringent test for the deployed modules and there is a good chance that some individual parts of data preparation process will fail. Another aspect in this scope is an eventual change of the physics targets or even a future upgrade of the ATLAS detector, with both making a modification of the reconstruction or simulation software inevitable. The author, however, hopes that at this stage the reader will not only have learned about the new powerful track reconstruction and simulation modules that have been put in place, but also that exactly this aspect of flexibility has been targeted with extensive care.

The Appendix aims to provide some additional aspects that are either useful for the understanding of the context, or referred to within this document for the convenience of the reader.

Bibliography

[1.1] CERN Document Server (CDS). Website. http://cds.cern.ch.

[1.2] The ATLAS Collaboration. The ATLAS Experiment at CERN. *ATLAS Communication, to be published in: Journal of Instrumentation*, ATL-COM-PHYS-2007-102, 2007.

Those who reject new theory are in the grip of an older theory.

Richard Powers - Galatea 2.2

Chapter 2

The LHC and the Window to new Physics

Collision experiments are the most prominent technique for fundamental research in particle physics. Following a common spirit that sometimes you have to destroy in order to create something new, they allow us to create conditions that are similar to the state of universe *relatively* short in time after the big bang. In fact, the *Large Hadron Collider* (LHC) [2.1], which will be described in more detail in the following section, will reach back in time as close as 10^{-14} seconds after the creation of the universe. Given the rapidity of processes that took place in the early stages of the universe, this is, however, still fairly late.

2.1 The Large Hadron Collider at CERN

The LHC will be the future main accelerator at CERN, Geneva. It is currently under completion in the facilities that have been already used for the *Large Electron Proton* (LEP) [2.2] collider, which has been decommissioned in 2000. The LHC, where two proton beams will be collided with a maximum collision energy of 14 TeV, is at the end of a multi step acceleration chain. It starts from a linear proton accelerator that brings protons to an initial energy of 50 MeV. Bunches of 10^{11} single protons are then injected into the so-called *PS Booster*, a pre-acceleration complex to 1.4 GeV, from where they are inserted into the *Proton Synchrotron* (PS) and successively — after reaching 26 GeV — into the *Super Proton Synchrotron* (SPS). The SPS is an accelerator hosted in an about seven kilometers long tunnel that is placed roughly 30 meters underground; it has been already used for several experiments at CERN. The proton beam is accelerated in the SPS up to an energy of 450 GeV before being bi-directional injected into the LHC machine. The LHC is build in a tunnel located 100 meters underground on average, hosting 1232 superconducting dipole magnets that force the particles on a circle with a radius of about 27 kilometers. Magnetic lenses collimate the proton beams that are collided head-to-heat at four points along the ring that mark the four main experiments operated at the LHC: the two general purpose experiments ATLAS and CMS [2.3], located on opposite sides of the LHC tunnel, the LHCb [2.4] experiment that will focus on b-physics studies (such as CP violation studies on the B-meson) and the ALICE experiment [2.5], that has been designed to investigate collisions, mainly to study the quark-gluon plasma. A fifth experiment, the TOTEM experiment [2.6] aimed to measure very forward particles to focus on physics that is not accessible to the general-purpose experiments will be installed as vacuum chambers attached to the beam pipe and close the CMS interaction point. Figure 2.1 shows an illustration of the SPS and LHC with the locations of the experimental facilities.

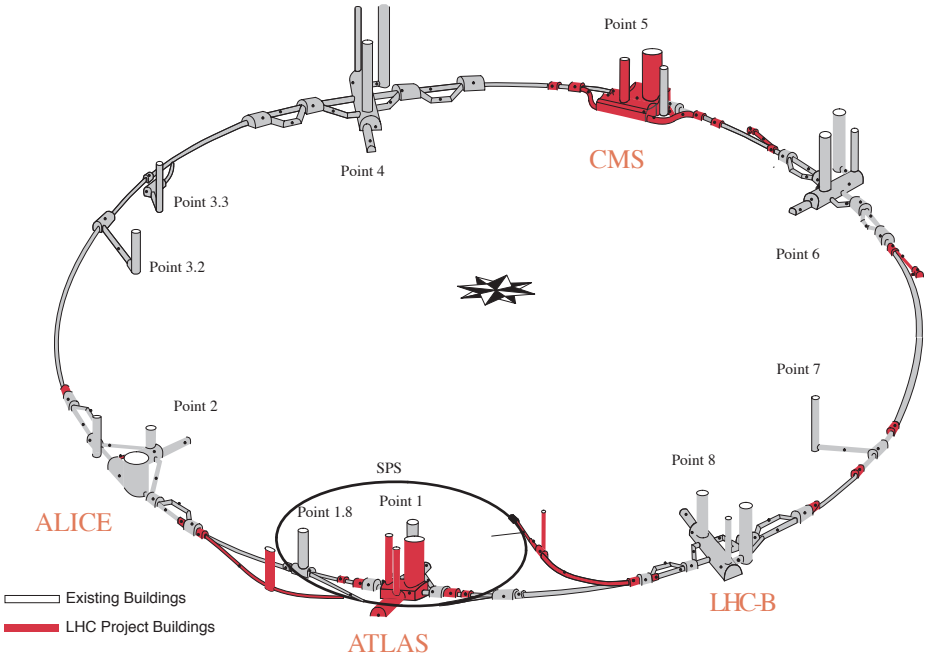


Figure 2.1: The experimental areas at the Large Hadron Collider that is located at CERN. The illustration shows the positioning of the four LHC experiments along the 27 kilometer long tunnel that has previously hosted the Large Electron Proton Collider (LEP). While for the two huge general purpose experiments, ATLAS and CMS, new caverns had to be constructed, ALICE and LHCb could use existing facilities of the prior LEP experiments. The picture includes also the Super Proton Synchrotron (SPS) with a circumference of about seven kilometers that is used as an injector to the LHC.

Luminosity and Cross Section

The proton bunches will be collided at a rate of every 25 nanoseconds, producing about 24 primary interactions at each bunch crossing. This leads to about 10^9 proton-proton interactions per second. Such a high collision rate is necessary, since many of the interesting physics interactions to be studied occur with very small probability; in terms of high energy physics, this is described by the *cross section* parameter σ and is as such expressed in units of area.

In collider physics the potential interaction rate is, in general, expressed through the *luminosity* parameter that is defined by the number of particles that are accelerated divided by a unit area and unit time. Consequently, the number of expected events N_i per second for a specific process can be expressed through its cross section σ_i and the machine luminosity L as

$$N_i = L \cdot \sigma_i. \tag{2.1}$$

The LHC has a design luminosity of $10^{34}\text{cm}^{-2}\text{s}^{-1}$, but will be operated at lower luminosity in the startup phase. The luminosity is obviously not constant during a physics run; it decays due to the degradation of the beam intensity caused by the already happened interactions and beam emissions. It is thus more a machine parameter than a good measure for the available event statistics. Latter is therefore often expressed through the integrated luminosity $\mathcal{L}$ that can be directly related to the number of events in the integrated time window. For doing so, also the duty cycle of the machine that accounts for the actual fraction of

beam operation has to be considered.

2.2 The Physics Challenge

The impulse for the fundamental searches of high energy physics could come out of a child's mouth: *What is the world made of?* And like so often, the simpler the question, the broader the answer to be given. The theory most referred to in this context is the *Standard Model* (SM). Although being tested to very high precision in previous high energy physics experiments [2.7,2.8] it has one draw back: it is incomplete, and - in the worst case it is even wrong. Regardless of the fact that so far no experimental data exists that contradicts with predictions of the SM would a failure in the search for its last missing piece, the Higgs boson, be regarded as a general failure of the SM theory as we know it now. Complete reviews of the SM are given in many standard textbooks, such as [2.9], and only a very short overview will be presented in the following.

2.2.1 The Standard Model

The Standard Model includes at present two relativistic quantum field theories, the *Quantum Chromo Dynamics* (QCD) that describes the strong interaction, and the electro-weak theory, which successfully unifies the *Quantum Electro Dynamics* (QED) with the theory of weak interactions. Both theories are gauge theories, i.e. it is assumed that the fundamental interactions are invariant under local gauge symmetry transformation. Applying this requirement — in the formalism of the Lagrangian field theory through the introduction of gauge fields — results in a field coupling term that can be interpreted as *messenger* particles with spin 1 (*bosons*) of the interaction. Following the theorem of *Noether* [2.10] symmetry is always connected with conservation quantities, and it is a remarkable success of the SM that it also describes conservation laws to great satisfaction. In terms of the Standard Model, the application of local gauge symmetry is even more dramatic: the symmetry group does not only yield conservation laws and thus the conserved quantities of the theory, but determines also the dynamics of the system through the introduction of the the messenger particles. This mechanism could be first demonstrated for invariance under local phase rotations of the QED $U(1)$ symmetry group. It has later been expanded to more general symmetry groups by Yang and Mills [2.11], making them the most successful techniques in modern physics. The resulting theories are called *Yang-Mills theories*.

Ironically, when one believes the episode that the falling apple on Sir Isaac Newton's head inspired him to his *Principia* and could thus be regarded as the inauguration of modern physics, gravity is still the least known interaction in modern times. In many senses, it is still as mysterious to us as it might have been for Newton himself. Several attempts have been carried out to unify the the fundamental interactions in one theoretical framework ($\rightarrow$ *Grand Unified Theory* (GUT), see e.g. [2.12]), which requires in the current understanding a quantised field theory of the gravitational force. For this reason, and according to the gauge principle that will be briefly described in the following sections, a gauge boson for the gravitation (the so-called *graviton*) is often postulated, but no experimental evidence could so far strengthen this approach.

Constituents of the Standard Model

A complete theory of our world has to include both, the elementary constituents of matter and the mechanisms of how they interact. In the history of mankind, the picture of the

elementary units that build our world have changed very often: from the four elementary constituents *earth*, *fire*, *water* and *air* in the ancient greek times, to Democrite's unsplittable *atomos* and Mendeleev's *periodic table*, all of which have been motivated to describe the most elementary units and how they are assembled to build the world we see. Often, experimental results forced to reveal the current theory, amongst which is the Rutherford's scattering experiment of α nuclei targeted at a thin gold foil that marked the exploration of the atomic sub-structure. In this pure principle, the LHC experiments follow Rutherford's footprints and perform yet another scattering experiment, even though on quite another energy scale.

The SM describes the fundamental constituents of matter as 12 different types of particles with spin $1/2$ (*fermions*) that are presented in Tab. 2.1; they can be divided into two groups: those that experience the strong interaction (*quarks*), and *leptons* that do not couple to the QCD gauge bosons. The different behavior of the two groups under influence of the strong interaction has a dramatic effect: since the strong interaction increases with the interaction distance, quarks can not be observed as single particles, but only in a confined compound. These *quark-atoms* are called *hadrons*. Leptons, on the other hand, mark prominent final state particles of physics events to be studied at the LHC and can be observed as isolated, single particles. Both types quarks and leptons can be classified in three generations, while the known world that we can experience with our human senses is entirely built from the first one[1]. For each of the fundamental fermions an according anti-particle exists with (mainly) inverted quantum numbers.

Table 2.1: The fundamental fermions ordered into three generations in the Standard Model. Nowadays, experimental evidence has been found for the existence of all of these fermions.

Fermions	**Generation**			**Charge**
	I	II	II	
quarks (q)	$\begin{pmatrix} u \\ d \end{pmatrix}$	$\begin{pmatrix} c \\ s \end{pmatrix}$	$\begin{pmatrix} t \\ b \end{pmatrix}$	$\frac{2}{3}$ $-\frac{1}{3}$
leptons (ℓ)	$\begin{pmatrix} \nu_e \\ e \end{pmatrix}$	$\begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix}$	$\begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}$	0 -1

The electro-weak Interaction

In the previous section it has been briefly mentioned (without giving proof) that the gauge symmetry formalism yields *messenger* particles of the interaction. For the electro-weak theory these are four gauge bosons, the mass-less photon (γ), which has been the first gauge boson to be discovered, and the massive vector bosons: the neutral Z^0 and the two charged $W^\pm$. The number of gauge bosons is hereby not arbitrary, but results from the symmetry in which the gauge theory is embedded (for the electro-weak theory, this is the $SU(2)_L \times U(1)_Y$ group of the left-handed isospin and the hypercharge Y)[2]. The number of gauge bosons corresponds to the number of generators of the underlying symmetry group. The initial formulation of the electro-weak interaction as a classical Yang-Mills theory, however, leaves the

[1] During the LEP area at CERN, overwhelming evidence could be found that the number of families is restricted to three, see [2.8].

[2] A crucial phenomenolocigal input from experimental observation of $\beta-$decay [2.13] was that the weak interaction does not respect reflection symmetry, which restricts the symmetry group to the left-handed isospin sector.

problem of massless mediator bosons, a defect that has been overcome by the introduction of spontaneous symmetry breaking that will be explained in the following. Weak interaction is felt by all fermions, either as a neutral weak current through Z^0, or as a charged current trough coupling with the W bosons. Since the photon couples only to the electrical charge, QED is restricted to charged particles and thus neutrinos (ν) are not affected.

The Strong Interaction

The strong interaction is described through the theoretical framework of the QCD. In the fermion sector, it is restricted to the quarks, since only those carry the charge of the strong interaction, the so-called *color charge*; three types of color charge exist: *green*, *red*, and *blue*. The basic theory of the QCD resembles the mechanism of local gauge invariance that has been briefly discussed before. The symmetry group of the QCD is the non-Abelian group $SU(3)_c$, that needs 8 independent generators. Consequently, eight gauge fields need to be introduced; in the particle interpretation they are called *gluons* and act as the messenger particle of the strong force. The QCD symmetry is in contrast to the electro-weak sector not spontaneously broken, and the gluons remain therefore massless. The non-Abelian character of the $SU(3)_c$ group (i.e. the generators of the group do not commute) has a strong consequence: the gauge bosons of the strong interaction have color charge and can thus interact with each other (*self coupling*). The self coupling is the reason for a property of the dynamical behavior of the strong interaction that is in contrast to the electro-weak interaction: the *confinement*, that encloses bound state quarks (*partons*) in the size of about a nucleus. The partons show by contrast a quasi-free movement at small distances, which is usually referred to as *asymptotic freedom*. This behavior is reflected in the theory of an energy dependent coupling constant caused by the renormalisation of the theory through loop corrections. Quarks can therefore not be observed, only colorless compounds exist: mesons $(q\bar{q})$, barions (qqq) and anti-barions $(\bar{q}\bar{q}\bar{q})$.

Table 2.2 summarises the fundamental interactions, their associated gauge bosons and relative strength parameters.

Table 2.2: The fundamental interactions and the associated gauge bosons, their relative strength and range.

Interaction	**Gauge bosons**	**Range** [m]	**Rel. Strength**
strong	8 gluons	10^{-15}	1
electromagnetic	γ	∞	1/137
weak	Z, W	10^{-18}	10^{-6}
gravitational	graviton (postul.)	∞	10^{-39}

The Standard Model Higgs Boson

One part of the Standard Model is not yet well established experimentally: we do not know what causes the fundamental particles to acquire masses, nor is a tested theory known that would explain the different values of these masses. The most prominent theory in this context is the Higgs mechanism [2.14]. It is based on a spontaneous breaking of the (electro-weak) symmetry, which introduces a massive scalar particle, the so-called *Higgs boson*.

In the previous section, the vector bosons that mediate the electro-weak interaction have

been introduced through the principle of local gauge theory of the $SU(2)_L \times U(1)_Y$ symmetry group. However, when simply introducing the gauge fields in the Yang-Mills formalism, the vector bosons remain massless, which is clearly in strong disagreement to any experimental evidence. On the other hand, when introducing mass terms for the Z and $W^{\pm}$ *by hand* into the Lagrangian, the local gauge invariance would immediately be violated. One would be left with the choice of giving up the fundamental theory (which so nicely reproduces the observed particles, many of their properties and conservation laws) or neglecting the fact that the mediator bosons are massive. Latter is, however, difficult to do since the masses of the Z and $W^{\pm}$ bosons are indeed needed to explain the relatively short range of the weak interaction. An elegant way out of this disaster has been given through the Higgs mechanism, see e.g. [2.15]. An $SU(2)$ doublet of complex scaler fields Φ with three degrees of freedom is introduced with a non-zero vacuum expectation value (VEV). The VEV violates the local $SU(2)_L \times U(1)_Y$ symmetry, leaving only the electromagnetic $U(1)_Q$ symmetry group intact. In common terminology, one speaks of the electro-weak symmetry group to be spontaneously broken, which leads to - following the *Goldstone Theorem* [2.16] - the appearance of massless bosons, the so-called *Goldstone bosons*. The spontaneous symmetry breaking of the electro weaksector generates three Goldstone bosons, respecting the three broken generators of the initial gauge group $SU(2)_L \times U(1)_Y$. After applying an effective field rotation (i.e. transforming to *unitary gauge*), these massless bosons are absorbed as new longitudinal components of the rotated gauge fields Z^0, W^+ and W^-, which turns them into massive particles. In several references this process is casually spoken referred to as the Goldstone boson being *eaten up* by the gauge boson. The photon, on the other hand, remains mass-less, since the generator of the $U(1)_Q$ symmetry (a sub-symmetry of the hypercharge) remains unbroken. Also the $SU(3)_c$ charge symmetry of QCD remains intact and the according gauge bosons — the gluons — massless particles.

Fermion Masses The fermion masses can also be generated with the introduced scalar field Φ. Fermions spontaneously acquire mass through a so-called Yukawa coupling term, that can be introduced without breaking $SU(2)_L \times U(1)_Y$ symmetry. This type of interaction is not a gauge interaction and the coupling strength is not determined by the electro-weak theory. In fact, the couplings of the Higgs boson with the fermions are arbitrary parameters of the SM, but direct proportional to the fermion masses. Latter is one of the motivations for the LHC that expands the reachable kinematic phase-space in comparison to prior collision experiments in the hope to reach an energy scale where the Higgs boson is more likely to be observed.

2.2.2 Beyond the Standard Model

The Higgs sector of the Standard Model asses one of the most striking arguments for an extension of the model itself; commonly known as the *hierarchy problem*, it addresses the internal consistency of the theory. The *uncertainty principle* of quantum mechanics [2.17] allows pairs of short lived so-called *virtual* particles to *appear* from the vacuum and then disappear again. These virtual particles lead to single or multi-loop corrections to the effective Higgs mass $\mathrm{m_H}$, and — unfortunately for the Standard Model — the correction to the mass grow with the energy of the virtual particles. Since arbitrarily large energies are not excluded by the theory, $\mathrm{m_H}$ could be arbitrarily large as well. However, the Higgs mass is constrained to the electro-weak scale and the hierarchy problem could thus be concentrated in one question: *Why is the Higgs mass so small?* The theory of *Supersymmetry* (SUSY) provides an elegant solution to this problem. It is based on the fact, that the corrections to the effective mass dif-

fers in a relative minus sign for fermion and boson loop corrections. If every fundamental particle had a *supersymmetric* partner that differs by an absolute spin value of $1/2$, one could achieve a systematic cancellation of the loop corrections. In fact, this is the (little simplified) principle of the *Minimal Supersymmetric Extension of the Standard Model* (MSSM) [2.18]. The bosonic partners of the fundamental fermions would then be consequently called *sleptons* and *squarks* (for *scalar* leptons and *scalar* quarks, respectively) and must have higher masses than their leptonic counter-parts. To agree with the non-observation of super-symmetric particles in prior high energy experiments, SUSY must be broken the supersymmetric particles heavy.

There are many more theories and models that go far beyond the SM, some of which include almost futuristic ideas that involve extra dimensions (that could not be seen so far) and a huge theoretical background that is commonly referred to as *String Theory*, which replaces the concept of point-like particles by a theory of vibrating superstrings. It seems foolish to even try to summarise a great deal of them and, in addition, it is not necessary for the understanding of the presented work. As a physicist the author looks forward with great expectations to test the predictions of these models with great care when data taken starts with the ATLAS detector.

2.2.3 Event Topologies and Consequences for Track Reconstruction and Simulation

Precise and efficient track reconstruction is indispensable for the analysis of many phenomena presented in the previous section. Since with the LHC, high energetic protons will be collided head-on, the primary process will be a fundamental QCD interaction that leads to an overwhelming background of multi-jet events. Efficient identification of electrons and muons final states can be used to separate new phenomena from the QCD background. This requires excellent track reconstruction and dedicated electron track fitting techniques.

The ATLAS detector has been designed to cover a large spectrum of possible event signatures in the LHC environment, but the Higgs search and SUSY models have marked the major guidelines for the detector requirements. The huge QCD-jet background prevents from detecting the Higgs boson through purely hadronic modes, unless the decay channel can be clearly identified. The dominant decay of the Higgs boson in the lower mass sector is $H \rightarrow b\bar{b}$. Unfortunately, it is completely shadowed by QCD background and thus a direct Higgs production with the Higgs boson decaying to $b\bar{b}$ can not be measured. In associate production modes, however, this decay channel can be separated clearly enough if the jet flavor can be determined. This requires an excellent vertex resolution (and thus track resolution) to identify b-quarks due to their large decay radii (b-tagging). A similar situation is present for the Higgs decay into a $\tau^+\tau^-$-pair. Most other decay channels involve high p_{T} leptonic final states, and henceforth high quality electron and (combined) muon reconstruction is needed.

And furthermore, Z and W bosons are identified through their leptonic decays and will be used for both to search for new physics phenomena and for precision measurements of the Standard Model parameters. Additional neutral gauge bosons predicted in many new physics scenarios — see Sec. 2.2.2 — can be identified through their decay into muons and electrons in events selected by muon or electron based triggers. Typical SUSY event signatures are m jets, n leptons and missing energy from escaping neutralinos that can not be measured[3], where m and n can have different values (even zero), which lead to different

[3] In hadron colliders, only the missing transverse energy can be estimated, that is why these measurements are often called E_T-measurements.

background scenarios.

Most of these phenomena require not only a well functioning track reconstruction, but also rely on powerful triggering mechanism that allow to distinguish these events from other event topologies. The trigger strategy in ATLAS is a multi level combination of both hardware and software components that have to be optimised in execution time; a more detailed description of the deployed techniques and algorithms is given in Sec. 3.2.1.

Hit Occupancy and Track Reconstruction in Jets Final state leptons are not the most challenging tracking relevant signatures for future measurements with the ATLAS detector. In contrary, they provide relatively clean traces in the detector and are often easy to isolate. Track reconstruction in jets, however, puts stringent requirements onto the pattern recognition: the high occupancy caused by the track density in jets clearly degrades the track finding efficiency and increases the probability of finding ghost tracks (→*fake tracks*, see Chap. 4). Efficient and precise track reconstruction inside jets will be needed in particular for heavy quark tagging.

Event Simulation The most unambiguous signal for a Higgs boson is a peak in the invariant mass spectrum of its decay products. This measurement can be done without any Monte Carlo (MC) simulation for the background. However, for the event selection and the study of properties just like couplings and spin parity, large MC background simulations are needed to determine the cross sections and the various characteristic distributions for both, signal and background events. This does not only require the most precise theoretical models that can be implemented in the event simulation, but also fast but precise simulation techniques such that the desired event statistic will be available. A solution for a new, fast and precise simulation strategy for the innermost tracking device of the ATLAS detector is presented in Chap. 9 of this document.

Bibliography

[2.1] LHC. *LHC Technical Design Report*, volume I, II, III. CERN, 2000.

[2.2] S. N. Ganguli. The Story of Large Electron Positron Collider. *RESONANCE*, October 2002.

[2.3] The CMS Collaboration. *CMS Physics Technical Design Report*. Technical Design Report CMS. CERN-LHCC-2006-001, Geneva, 2006.

[2.4] The LHCb Collaboration. *LHCb reoptimized detector design and performance - Technical Design Report*. Technical Design Report LHCb. CERN-LHCC-2003-030, Geneva, 2003.

[2.5] The ALICE Collaboration. *ALICE Technical Proposal*. Technical Proposal ALICE. CERN-LHCC-95–71, Geneva, 1995.

[2.6] The TOTEM Collaboration. *TOTEM Technical Design Reoport - Total Cross Section, Elastic. Scattering and Diffraction Dissociation at the LHC*. Technical Design Report TOTEM. CERN-LHCC-04–002, Geneva, 2004.

[2.7] B. Mele. Precision Tests of the Standard Model at LEP. *arXiv.org:hep-ph/9312285*, 1993.

[2.8] The Particle Data Group. Review of Particle Physics. *Phys. Lett. B*, 592, 2004.

[2.9] F. Halzen and A. D. Martin. *Quarks and Leptons: An Introductory Course in Modern Particle Physics*. Wiley, NY, 1984.

[2.10] E. Noether and M. A. Tavel. Invariant Variation Problems. *arXiv.org:physics/050306*, 2005.

[2.11] G. 't Hooft, editor. *Fifty years of Yang-Mills theory*. Hackensack, USA: World Scientific, 2005.

[2.12] C. S. Aulakh. The minimal supersymmetric grand unified theory. *Phys. Lett.*, B588:196–202, 2004.

[2.13] C. S. Wu, E. Ambler, R. W. Hayward, D. D. Hoppes, and R. P. Hudson. Experimental Test of Parity Conservation in Beta Decay. *Phys. Rev.*, 105(4):1413–1415, Feb 1957.

[2.14] P. W. Higgs. Broken symmetries, massless particles and gauge fields. *Phys. Lett.*, 12:132–133, 1964.

[2.15] A. Djouadi. The Anatomy of Electro-Weak Symmetry Breaking. I: The Higgs boson in the Standard Model. *arXiv.org:hep-ph/0503172*, 2005.

[2.16] J. Goldstone, A. Salam, and S. Weinberg. Broken Symmetries. *Phys. Rev.*, 127:965–970, 1962.

[2.17] W. Heisenberg. *Die Physikalischen Prinzipien der Quantenmechanik*. Hirze, Leipzig, 1930.

[2.18] S. P. Martin. A Supersymmetry Primer. *arXiv.org:hep-ph/9709356*, 1997.

Anything one man
can imagine,
other men can
make real.

Jules Verne

Chapter 3

The ATLAS Experiment

A particle detector has in principle two tasks: to provide measurements for the identification of a particle and to determine its kinematic properties; this is usually achieved by *measuring* both, the particle energy and the particle trajectory through the detector. These tasks are, in general, performed by several different detector technologies: tracking detectors for the estimation of the trajectory and calorimetry devices for energy measurements.

Modern particle detectors used in collision experiments are almost entirely of cylindrical design. Barrel cylinders at growing radii are located around the nominal interaction point, followed by disc structures that cover regions further along and upwards the beam direction. Tracking devices are located close to the primary interaction point for measuring the particle trajectory and to enable primary and secondary vertex reconstruction. Usually, calorimetric devices follow at larger radii and are designed to measure the particle energy. In general, all particles except the muon and neutrino are stopped within the calorimetric devices, simply for the fact that the calorimeter embodies enough material, such that the interaction probability of the particle with the detector material is high enough to result in a complete deposition of its kinematic energy. In fact, this is how the energy measurement is usually performed. There is little one can do about the neutrinos since their interaction probability with material is too low to build a reasonable small detector[1]. Thus their *measurement* as final state particles has to be done through their absence. There are also new physics models that predict massive neutral noninteracting particles that could leave the detector and also require a *missing energy* measurement. For this particular aspect a hermetic closure of the calorimetric devices is of great importance. Finally, muons have a far lower interaction probability than all other particles — except the neutrino obviously — and can thus penetrate the calorimetry devices. Most particle detectors incorporate therefore dedicated muon detectors that enclose all other sub-detectors. Final state leptons with high momentum are often very clean indications for fundamental physics processes. The dedicated muon chambers therefore often play an additional role as trigger chambers for selecting *interesting* events.

3.1 The ATLAS Detector

There exist many detailed documents that describe the setup of the ATLAS detector, starting from a technical proposal [3.1] document, the technical design reports — see [3.2–3.6] — to

[1]There are, of course, recent experiments that are designed to measure the induced decay products of interactions caused by neutrinos with detector material. The big interaction length, however, forces to use huge detector devices, few of them are even of artificial source. Usually, the polar ice fields, the water of an ocean or the earth itself are used as the effective detector body in such experiments.

a recently published document [3.7] that gives a detailed description of the ATLAS detector setup in its actually installed form in the experimental cavern. These documents also reflect the evolution of the experiment from a pure concept study to its final realisation. For convenience, the following description of the ATLAS detector will stick to the most recent detector characteristics, while trying to focus on relevant aspects for the presented work.

3.1.1 Axes Definitions and Conventions

The nominal interaction point of the two proton beams defines the origin of the axis system. The y-axis points to the surface, the x-axis to the center of the LHC tunnel and the z-axis is (almost) collinear to the beam line so that the axis system is mathematical positive defined. The polar angle θ is measured from the positive z-axis covering the range $\theta \in [0, \pi)$, the azimuth angle ϕ is measured in the $(x - y)$-plane, such that the positive x-axis has an azimuthal angle of $\phi = 0$ and the positive y-axis an azimuthal angle of $\phi = \pi/2$; ϕ covers the range $\phi \in [-\pi, \pi)$. The azimuthal and polar angles can be expressed through the momentum components using the relations

$$\tan\phi = p_y/p_x \quad \text{and} \quad \cot\theta = p_z/p_T, \tag{3.1}$$

where p_x, p_y and p_z denote the components of the momentum corresponding to the axis system and $p_T = \sqrt{p_x^2 + p_y^2}$ is defined as the transverse momentum with respect to the beam axis.

The pseudorapidity η is the rapidity of a massless particle and depends on θ as

$$\eta = -\log\tan(\theta/2). \tag{3.2}$$

The pseudorapidity is often used instead of the polar angle, since it transforms by just a constant shift under application of a Lorentz boost. High luminosity indicates in this document a luminosity of 10^{34} cm^{-2}s^{-1}, while low luminosity corresponds to 10^{33} cm^{-2}s^{-1}.

3.1.2 Global Parameters and Overview

The ATLAS detector is a classical particle detector in the sense of its sub-detector arrangement. At innermost regions a tracking or vertexing detector, the *Inner Detector* (ID), is installed that provides accurate particle localisation, such that both the initial momentum and the particle's production vertex can be estimated with very high precision[2]. A more detailed discussion of the ID can be found in Sec. 3.1.3. The ID is enclosed in a superconducting solenoid that provides the magnetic field for momentum measurements, which are performed by determining the trajectories of charged particles under influence of the Lorentz force. The ID is followed by calorimetric devices that are designed to measure the particle energy through its electromagnetic or hadronic interactions with the detector material. Consequently, the sub-detectors of the ATLAS calorimeter are called *electromagnetic* and *hadronic* calorimeters, respectively. Both consist of several different components that are presented in Sec. 3.1.4. For muons that penetrate the calorimeter due to their low interaction probability, a dedicated tracking detector has been built: the *Muon Spectrometer* (MS). It is the largest part of the ATLAS detector and allows the trajectory measurement of muons with very high precision to a momentum range of about 1 TeV, which is enhanced through a complex toroidal

[2]The production vertex of a single particle can, in general, not be determined, since only the trajectory of the particle is reconstructed. Vertex reconstruction has to be therefore performed on a collection of trajectories that are found in the tracking detector.

magnetic field setup and will be described in more detail in Sec. 3.1.5. Table 3.1 presents and overview of the global properties of the ATLAS detector, while Fig. 3.1 shows an illustration of the ATLAS detector and identifies the main components.

Table 3.1: Overall dimensions and properties of the ATLAS detector.

Parameters	
Total length	≈ 40 m
Overall diameter	≈ 22 m
Total weight	7000 Tons
η coverage for track reconstruction	$\eta < \|2.7\|$
η coverage for calorimetry	$\eta < \|5.0\|$

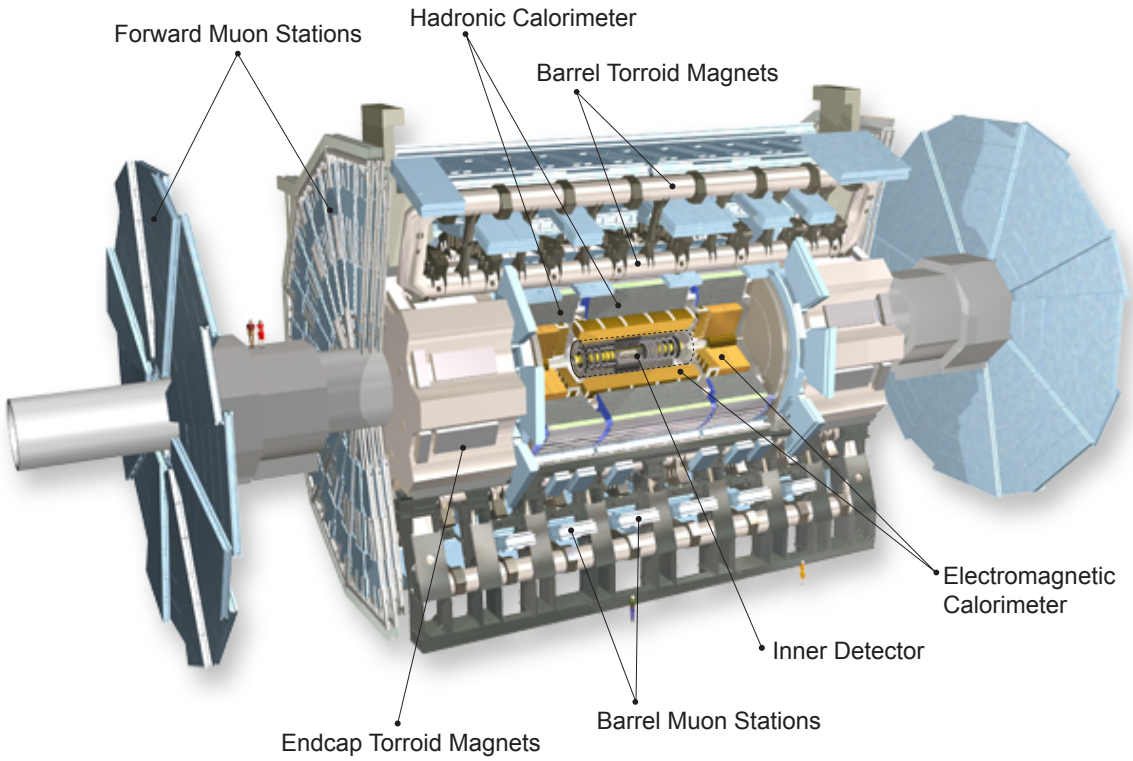


Figure 3.1: Overview of the ATLAS Detector. The central most tracking device, the Inner Detector, is highlighted with a dedicated cylinder volume.

Design Considerations

The design concepts of the ATLAS detector have been clearly driven by the required sensitivity for finding new physics final-state signatures that are reachable with the LHC (→

Sec. 2.2.3). While the detector setup has undergone several iterations since the technical proposal, the *wishlist* for the ATLAS detector performance has been unchanged since then:

- very good electromagnetic calorimetry for electron and photon identification and measurements, complemented by hermetic jet and missing energy calorimetry;
- efficient tracking at high luminosity for lepton momentum measurements, b-quark tagging, and for enhanced electron and photon identification. As well as high quality track reconstruction for τ-lepton identification and the reconstruction capability of some B decay final states at lower luminosity;
- stand-alone, precision, muon-momentum measurements up to highest luminosity, and very low-p_{T} trigger capability at lower luminosity.

And, *in order to maximize the physics reach and to optimize the exploitation of the LHC it is also important to achieve* [3.1]:

- large acceptance in $|\eta|$ coverage;
- triggering and measurements of particles at low-p_T thresholds.

Building a detector that is capable of fulfilling all this is itself a challenging task. In addition, readout electronics, trigger systems and finally the event reconstruction software has to be deployed in parallel for processing the taken event data[3]. Section 3.1.3 to Sec. 3.1.5 will focus on the pure technical realisation of the ATLAS detector, while Sec. 3.2 will subsequently draw the readers attention on the second aspect, the trigger system and the reconstruction software challenge in particular.

3.1.3 The Inner Detector

The ID is contained within a cylindrical envelope of length ± 3.4 m and of radius 1.15 m. It is placed inside a solenoidal magnetic field and covers a pseudorapidity range of $|\eta| < 2.7$. The ID consists of the following parts:

- the high-granulated silicon pixel detector closest to the interaction point;
- a silicon strip detector (SCT, *Semiconductor Tracker*) that surrounds the pixel detector;
- the *Transition Radiation Tracker* (TRT), a straw detector that is mainly designed to provide many measurements aimed at increasing the momentum resolution and providing additional particle identification potential via the transition radiation probability.

Full tracking acceptance can be achieved within a pseudorapidity region of $|\eta| < 2.5$, including parameter measurements and vertexing for heavy-flavour and τ tagging. An overview of the ID including a brief description of the main components and the number of readout channels, can be seen in Fig. 3.2. The high-radiation environment of the ID close to the beam line and interaction point imposes stringent conditions on all of its sub-parts during the first planed ten years of operation[4]: the sensors, the support and cooling structures and the onboard electronics.

[3] It seems of little sense to build a highly granulated detector, when the processing readout system or reconstruction software is not capable of dealing with the data taken with such a device.

[4] The innermost pixel layer is planned to be replaced after three years of operation, while a general upgrade of the ID tracker that corresponds to an according machine update of the LHC is scheduled to take place after ten years. A brief outlook on the simulation of future layouts of the ATLAS detector is presented in Chap. 9 of this document.

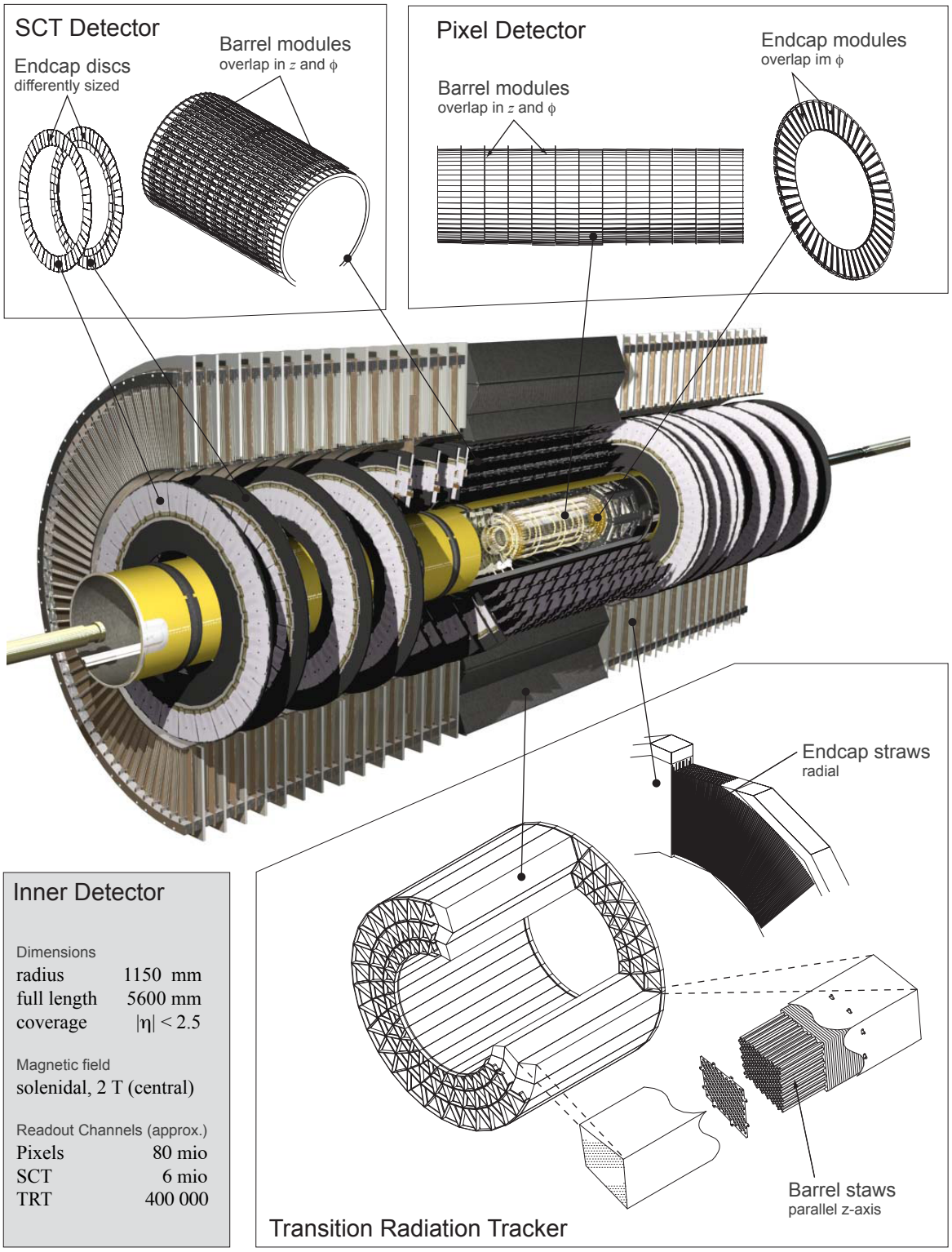


Figure 3.2: Overview of the Inner Detector and its components. Although the geometrical coverage of the ID expands to a pseudorapidity of about $|\eta| < 2.7$, track reconstruction is limited to $|\eta| < 2.5$, since a minimal required number of modules has to be intersected to allow the trajectory finding and constrain the track fit.

The ID Magnet System

The ID is placed inside a superconducting NbTi/Cu solenoid magnet that provides a central field strength of about 2 Tesla. The solenoid has a radius of 1.25 meters and a length of 5.3 meters and is thus shorter than the ID itself. This results in an inhomogeneous magnetic field, in particular in the forward region. Figure 3.3 shows the longitudinal and transverse components of the magnetic field strength in the ID. A discussion of the influences on track reconstruction (and in particular in the context of track extrapolation) caused by the magnetic field setup can be found in Chap. 7.

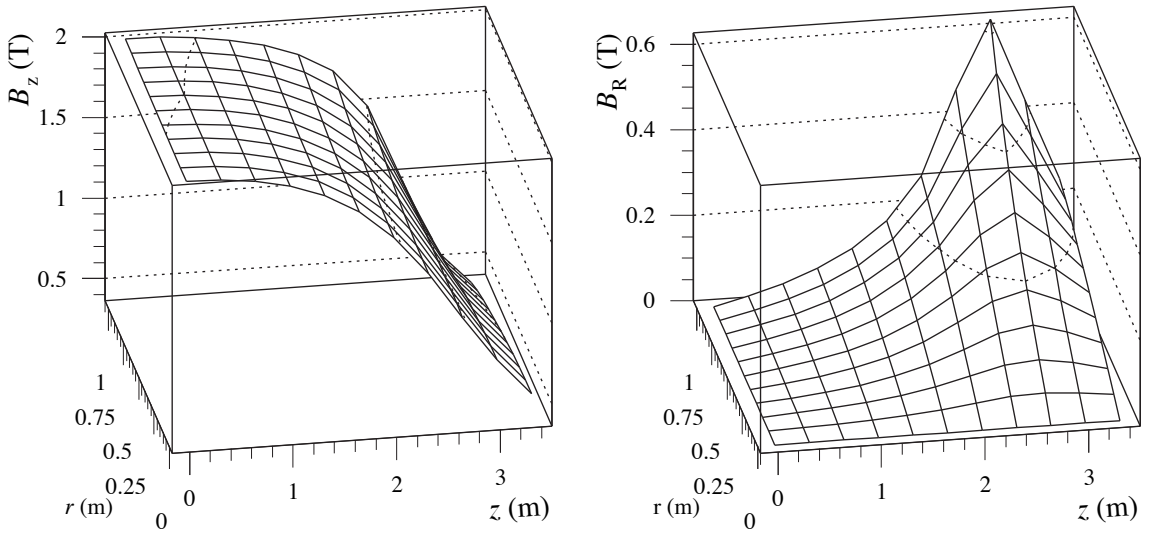


Figure 3.3: Inhomogeneity of the solenoidal magnetic field in the Inner Detector, showing the dependency of the transverse magnetic field B_r and the longitudinal magnetic field B_z as a function of z and the radius r.

The Pixel Detector

The innermost of the tracking detectors is realised as a silicon pixel detector to achieve highest granularity around the interaction point. The pixel detector — and in particular the innermost layer — dominates the impact parameter (and henceforth also the vertex) resolution in track reconstruction[5]. There are in total 1744 identical pixel modules that are assembled to three barrel layers at radii 50.5, 88, and 120 mm. Along and opposite the beam line there are three endcap discs at each side, located within $495 < |z| < 650$ mm. One single pixel module has the dimension of 24.4 $\times$ 63.4 mm^2 (with an active area of 16.4 $\times$ 60.8 mm^2), and is approximately 250 μm thick; it is built of 16 identical silicon sensors with 18×164 pixels each, which leads to about 47 000 pixels per module and 80 million readout channels for the pixel detector in total. Most pixels are 50 $\times$ 400 μm^2 large and yield — improved through an analog method that uses charge interpolation — an intrinsic resolution of about 10 μm in $r\phi$ and 115 μm in z, respectively. The areas where the single sensors are put together are characterised by larger pixel sizes (50 $\times$ 600 μm^2) and a shared readout system that leads to so-called *ganged* pixels [3.8]. In track reconstruction, these ambiguous pixels have to be dealt with dedicated care, either by assigning a larger cluster error or by resolving the actual pixel position. The mechanism of a silicon detector is straight-forward: each

[5] A brief discussion of this aspect, including the fundamental relations, can be found in App. A.1 of this document.

pixel (or strip) works like a single diode, with a voltage applied such that the border area is depleted and the resistance is high. A traversing charged particle would ionise the material and create pairs of electrons and corresponding holes along the trace. The opposite charges then drift apart under the voltage and can be registered as a hit. One single pixel hit yields a resolution of

$$\sigma_{1x} = \frac{w}{\sqrt{12}}, \tag{3.3}$$

which corresponds to the uncertainty of a uniform distribution over the pixel width w. The resolution can be improved when more than one pixel contribute to the *cluster*. This is, because the individual pixel position can be weighted according to the deposited charge. The charge is in the ATLAS pixel detector estimated via a time over threshold measurement. In track reconstruction, this process of determining the track intersection through the associated pixels is called *clusterisation*. A broader discussion of this aspect in the context of the fast track simulation will be given in Chap. 9.

The pixel detector is inserted into the *Pixel Support Tube* (PST) with a total length in z of 6.9 meters, an inner radius of $R_{min} = 4.2$ centimeters and an outer radius of $R_{max} = 25$ centimeters. The PST can be removed and installed almost independently from other detector structures. It provides the ability to replace the pixel elements for example in case of major radiation damages.

The Semiconductor Tracker

The SCT is a silicon strip detector with double-layered silicon modules. The 2112 barrel SCT modules have a 80 μm pitch micro-strip structure, two modules are mounted with a stereo angle of ± 20 mrad with respect to another to build one *sandwich* base element. The stereo angle between the modules allows to measure both detector coordinates, yielding an intrinsic precision of 17 μm in the $r\phi$ direction and a resolution of about 960 μm in the longitudinal direction. This is in particular important for the pattern recognition process, when the double-sided sandwich structure — together with the constraints given from the module surface and a beam spot assumption — are used to build three-dimensional space points that mark the input objects for the seed finding process (→ Chap. 8). The barrel modules are assembled to four cylindrical layers that expand ± 746 mm in z around the nominal interaction point an are build with radii between 230 and 559 mm.

While all SCT barrel modules are identical requires the more complicated detector setup of the SCT endcaps differently shaped modules. The modules are of trapezoidal shape and the single strips are placed in a fan-like structure and point to the beam-axis. Modules of different sizes are installed in various radial rings which are mounted on 9 disks at each side of the detector within the range $847.5 < |z| < 2727$ mm. The endcap modules also follow a sandwich structure with a stereo angle to constrain a two-dimensional local measurement per intersected disc; a similar resolution as in the SCT barrel can hereby be achieved.

Both, the barrel and endcap readout is purely binary, i.e. in contrary to the pixel detector where the charge deposition characteristics is used to perform an analog clustering approach, the SCT clustering model is purely digital.

The Transition Radiation Tracker

The silicon tracking devices are enclosed in the so-called *Transition Radiation Tracker* (TRT), which is a drift tube detector (see also Sec. 4.1). There are about 50000 straws in the barrel region that are mounted in a 32-fold geometry, and 320000 straws in the endcap regions

that span out in a fan structure pointing along the radial direction. One single straw has an internal diameter of 0.4 cm and a 85 μm thick Kapton ($C_5H_4O_2$) wall. The central wire is made of gold-platted Tungsten-Rhenium (W-Re) with a radius of 15.5 μm and the straw is supposed to be filled with a non-flammable gas mixture (70 % Xe, 27 % CO_2, 2 % O_2). In total, the TRT has 420000 readout channels with a drift-time measurement for each channel; a track in the TRT causes on average 36 hits in the TRT. The spatial resolution $\sigma_{r\phi}$ of a single straw is on average 140 μm for zero luminosity; at high luminosity the spatial resolution increases to 250 μm due to pile-up effects[6]. The intrinsic resolution is not constant, but depends strongly on the drift radius. In general, the closer the track approaches the wire, the less precise can the drift time be measured. This is a composition of two reasons: on the one hand is the effect of discrete ionisation cells less pronounced at larger drift radii (and consequently at smaller path lengths in the individual tube), and on the other hand is the drift velocity higher at closer radii, which also leads to an increased measurement error.

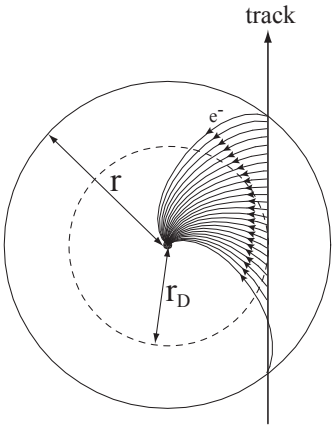


Figure 3.4: Illustration of a single measurement with a TRT straw of radius $r = 2$ mm. The traversing particle ionises the confined gas and causes an electron shower towards the central wire (anode). The bending of the electron trajectories towards the wire are hereby caused by the magnetic field. From the shower arrival time distribution a drift time measurement can be performed which defines a circle around the wire. Finding the actual closest point of approach is then subject of track reconstruction and can not be performed through the measurement.

The TRT provides a high number of hits per track, and thus contributes significantly to the momentum resolution of the ID. Furthermore, it hosts a strong potential for additional particle identification. The volumes between the single straw detectors are filled with materials that have rapidly changing dielectric constants, resulting in transition radiation when ultra relativistic particles traverse these detector areas. The emitted photons, typically within the X-ray spectrum of a few keV, are usually collinear to the particle direction and absorbed mainly by the Xe gas — the main component of the TRT gas mixture. This yields a higher charge deposition and consequently a higher charge collection on the wire, flagged by the readout electronics as a *high threshold* hit. It can be shown that the transition radiation is highly proportional to the relativistic γ factor of the traversing particle. The TRT layout has been optimised for the identification of electrons that have due to their lower mass a relatively high γ value. Figure 3.5 shows the probability of a high threshold hit p_{HT} in the TRT for different particles obtained with test beam data and fitted with a generic fit function.

Both, an adequate drift radius description and a simplified high threshold hit simulation are included in the fast track simulation and explained in some more detail in Chap. 9.

[6] In Sec. 4.3.2, a dedicated fitting technique will be presented that can minimise wrong hit assignments caused by the high occupancy in pile-up environment. This dedicated filter has been deployed in the ATLAS reconstruction framework in both, pattern recognition and track fitting, which will be described in Chap. 8 of this document.

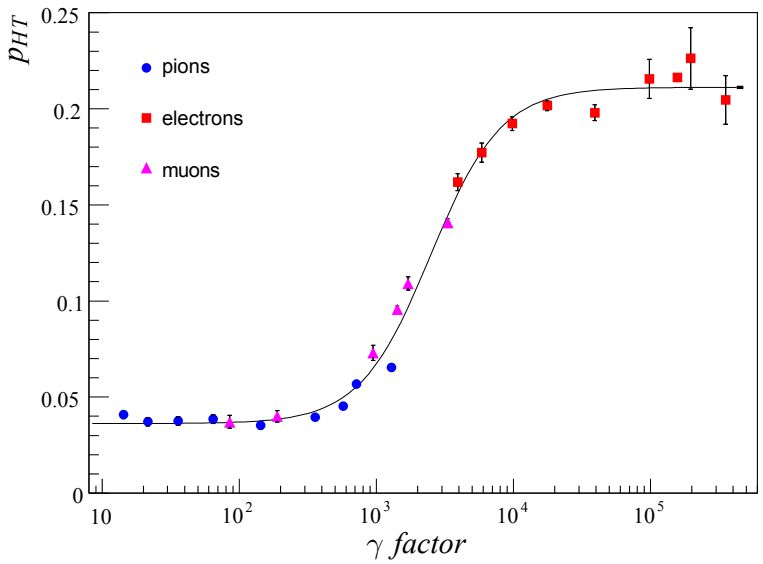


Figure 3.5: The probability of a high threshold hit in the TRT obtained from testbeam data for different particle types. Source: [3.9]

Material Budget of the ATLAS Inner Detector

For a tracking detector it is very important that the amount of inert material is kept at the lowest level. This is, because the interaction of the traversing particle with the detector material decreases obviously the overall resolution of the tracking devices since it introduces random disturbances that have to be taken into account. The detailed knowledge of the material distribution is hereby crucial for the integration into both simulation and track reconstruction algorithms, since the single effects obviously scale with the material budget. The theoretical background for the integration of material effects into track fitting is described in Chap. 4, and the current implementation of these models in the track extrapolation is further on presented in Chap. 7. The simulation of multiple scattering and energy loss effects is described in the context of a new fast track simulation engine in Chap. 9.

The inert material of the ID also influences all successive detectors further distant from the interaction point. In ATLAS, this effect is negligible for the hadronic calorimeter and the outermost stand-alone muon tracker, but has to be considered for the electromagnetic calorimeter. The correction of the ID material for the electromagnetic calorimeter measurement is partly done through the insertion of a dedicated presampler layer, see Sec. 3.1.4, and partly using dedicated scale factors obtained from testbeam runs.

Figure 3.6 shows the material distribution as given by the simulation geometry in terms of radiation length X_0 depending on the pseudorapidity η for the ATLAS Inner Detector.

It is almost impossible to provide a precise *a priori* picture of the material distribution that represents the actually built detector. A big effort has been put recently [3.10] to complete the understanding of the single contributions of the various detector parts to a very detailed level and reflect them in the simulation geometry. Unfortunately, every step closer to the realistic detector setup increased the overall budget, the published material distributions of the *Technical Proposal* [3.1] and the *Inner Detector Technical Design Report* [3.4] in comparison a recent simulation layout can also be seen in Fig. 3.6

Estimation of the Tracker Material

The final determination of the actual material in the ID will only be possible once the detector construction is completed. This is, because many components of the detector such as

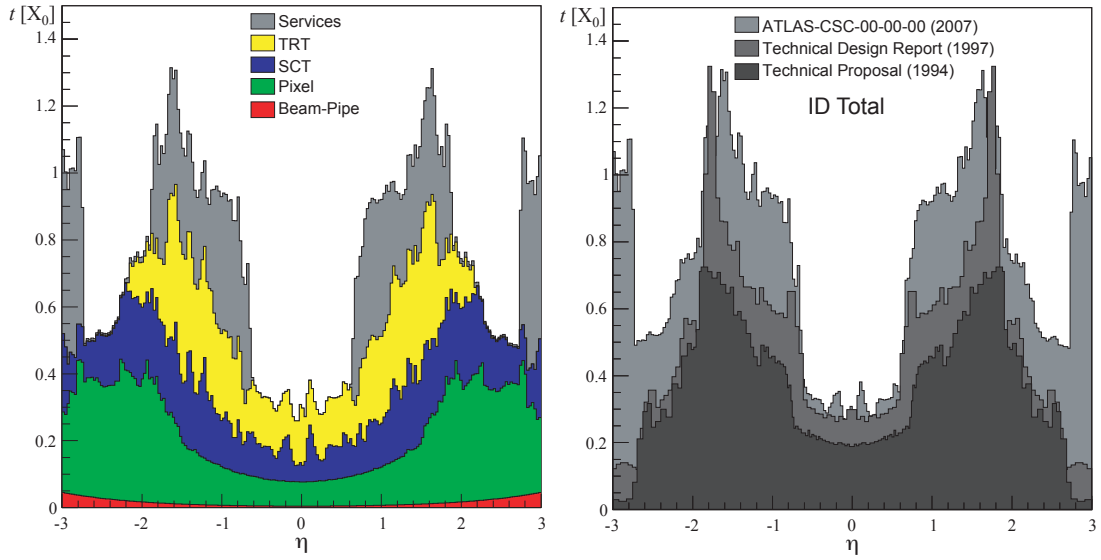


Figure 3.6: The material budget in terms of radiation length X_0 showing the cumulative contributions of the sub-detectors and detector services in the Inner Detector to the left. The right plot shows the evolution of the total material budget integrated into the detector simulation within the last ten years.

cables, screws, and additional tape or glue material are not trivial to be described through a detector model or are simply not precisely enough known. A very common technique to determine the material of tracking detectors is a detailed analysis of photon conversions within the detector volume, which are well described by theory [3.11]. However, this approach puts stringent requirements onto the reconstruction software; it has to be able of finding the conversion products (i.e. mainly electron tracks) and performing a successful conversion vertex fit. Yet another material-dependent process can be used for the material calibration in conjunction with electrons: the emission of hard brem photons. In contrast to the conversion finding, this technique does not require a vertex fit, but a precise determination of the electron trajectory. In particular, the momentum before and after such a hard emission of bremsstrahlung has to be reconstructed. A recently developed technique that is based on an extension of the standard track fitting algorithms ($\rightarrow$ *Gaussian Sum Filter*, see Sec. 4.3.2) has shown a great potential to be used in such an approach. A third possibility to constrain the material distribution of the tracking detector is the mass-determination of narrow particles through purely leptonic decay channels, such as $K_s \rightarrow \mu^+\mu^-$ or the leptonic J/Ψ decays. These methods, however, depend strongly on a precise knowledge of the magnetic field and the detector alignment.

From previous experiments [3.12] it is known that the necessary calibration of the material has exceeded 20%. For ATLAST, an anticipated average description of the tracker material to a level of $\approx 1\%$ is targeted to comply with the needs of SM precision measurements [3.13].

A coherent framework for updating the material description of the sub-detectors will thus be necessary for the starting phase of the ATLAS experiment. An appropriate schema should be able to cover two different aspects: at first — since the detector simulation and reconstruction software use different geometry models — an automatic synchronisation between these two descriptions is necessary that guarantees consistent material sources for analyses based on Monte Carlo simulated data. Such a mechanism is implemented in the newly developed reconstruction geometry, where a mapping mechanism of the simulation material onto the reconstruction geometry model is presented; it is in detail described in Chap. 5. On

the other hand, a facilitated scaling mechanism of the given material layers may be used to provide fast feedback to the simulation geometry, in particular when being used with the fast track simulation that is presented in Chap. 9.

3.1.4 Calorimetry

The ATLAS calorimetry system builds a hermetically closed structure that is subdivided into an *Electromagnetic Calorimeter* (ECAL) and a *Hadronic Calorimeter* (HCAL). Both subsystems consist of a barrel and two end-caps sections. For calorimetry it is of particular interest to cover a very large $|\eta|$ region to enhance a precise missing energy estimation. Good missing energy measurements are crucial, since this marks predicted final states signatures of many new physics scenarios or is simply needed for (mostly hadronic) recoil measurements. The ATLAS calorimeter therefore covers the detector region up to $|\eta| < 4.9$, the very forward regions are equipped with a dedicated *Forward-Calorimeter* (FCAL). An overview of the ATLAS calorimeter system is shown in Fig. 3.7.

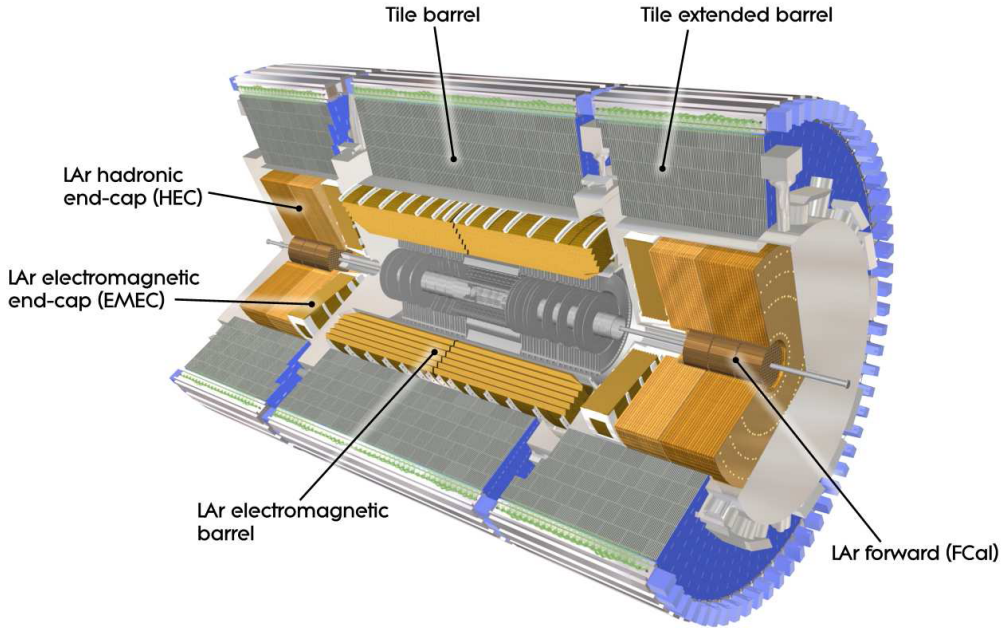


Figure 3.7: Cut-away view of the ATLAS calorimetry devices. The illustration contains the central Inner Detector, the LAr calorimeter, the Tile barrels and forward calorimeters.

The relative energy resolution of calorimetric devices is often parameterised from experimental measurements. It is then given after noise subtraction in the form of

$$\frac{\sigma(E)}{E} = \frac{a}{\sqrt{E}} \oplus b, \tag{3.4}$$

where a is the stochastic term and b the constant term reflecting local non-uniformities in the response of the calorimeter.

The Electromagnetic Calorimeter

The ECAL is a lead/liquid-argon detector with accordion-shaped geometry and is often referred to as *Liquid Argon* (LAr) calorimeter. The LAr is divided into a barrel part that covers

$|\eta| < 1.475$ and two endcaps ($1.375 < |\eta| < 3.2$). The accordion-shaped kapton electrodes and lead absorber plates ensure hermetic cluster measurement over the entire pseudorapidity range, the rapidity segmentation $\Delta\eta$ varies hereby from 0.003 to 0.1, while the azimuthal segmentation varies from 0.025 to 0.1. The thickness exceeds 24 radiation lengths in the barrel region, and 26 radiation lengths in the endcaps. It also includes a few radiation lengths for the pre-sampler that is restricted to the region $|\eta| < 1.8$. The presampler consists of an active LAr layer, and is used to correct for the energy lost by electrons and photons upstream of the calorimeter (i.e. in the ID and the solenoid, which accounts for approximately 2.3 X_0). The huge material budget of the calorimeter devices has a great influence on combined muon reconstruction, an appropriate description of the material distribution is therefore needed within the tracking framework. Figure 3.8 shows the material contribution in the LAr calorimeter, a comparison of the total material description in the simulation and reconstruction geometry can be found in Chap. 5. The energy resolution of the LAr is $\sigma(E)/E = 10\%/\sqrt{E} \oplus 0.2\%$.

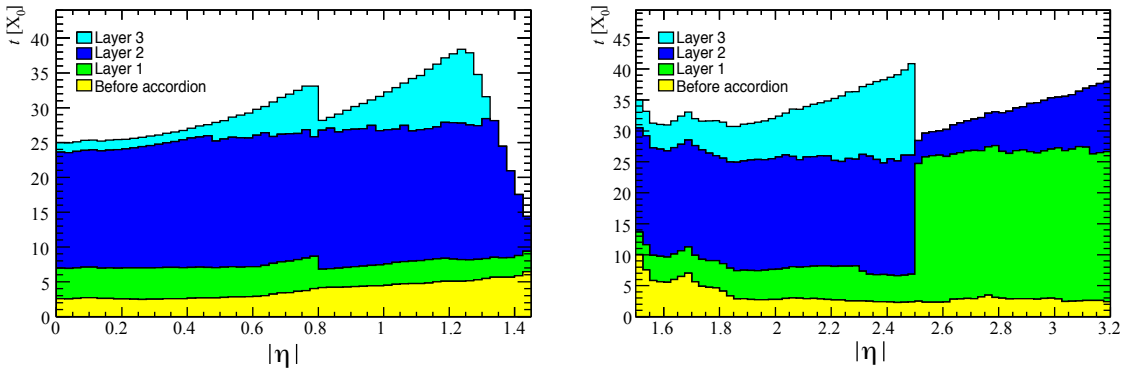


Figure 3.8: The material budget in terms of radiation length X_0 for the LAr calorimeter divided in the the contributions before the accordion structure (including the ID and pre-sampler) and the different accordion layers.

The Hadronic Calorimeter

The HCAL is placed behind the LAr calorimeter and consists of two parts: an iron tile calorimeter and LAr hadronic endcap calorimeters (HEC). The larger part, the hadronic barrel calorimeter, is a sampling calorimeter that uses iron as the absorber and scintillating tiles as the active material. The tiles are placed radially and staggered in depth, a structure which is periodic along z. The *Tile* calorimeter consists of a central barrel structure and two identical extended barrels, reaching in total up to $|\eta| < 1.7$. In regions of higher pseudorapidity, a dedicated hadronic endcap calorimeter is located. Is is covered by the extended Tile barrels and uses liquid argon as the sensitive material, while mainly lead is used as an absorber material. The full acceptance region of the HCAL can thus be expanded to $|\eta| < 3$.

One crucial design parameter for the HCAL has been its desired thickness to build a good containment for high energetic hadronic jets. This is not only necessary for the precise jet energy estimation, but also inevitable for the Muon Spectrometer, since a contamination of the MS with jet constituents would dramatically decrease the resolution of the outermost tracking detector.

The fractional energy resolution of the HCAL yields according to Eq. (3.4) a stochastic term of about 56 % and a constant term of 5.5 % in the barrel region, but is strongly dependent on

the pseudorapidity and again different for the HEC.

The Forward Calorimeter

In the very forward region ($3 < |\eta| < 4.9$), a dedicated forward calorimeter is deployed to ensure a maximum coverage which is required for a good E_T^{miss} *measurement*. The FCAL is located about 4.5 m from the interaction point and consists of three parts: the first part is made of lead, while the following two parts use tungsten as the absorber material. Each section of the forward calorimeter consists of a metal matrix with regularly spaced longitudinal channels that carry the readout channels and are filled with liquid argon as the active material. For the FCAL, a relative energy resolution of about $\sigma(E)/E = 28\%/\sqrt{E} \oplus 3.5\%$ for electrons, and $\sigma(E)/E = 70\%/\sqrt{E} \oplus 3\%$ for pions can be achieved, respectively.

Luminosity Monitor and Very Forward Detectors

In the extreme forward region of ATLAS three small detector systems are installed and complete the coverage of the ATLAS detector. These are closely connected to the luminosity determination in ATLAS, but are also foreseen to study forward physics. If ordered according to their distance from the ATLAS interaction point, the first system is the main luminosity monitor of ATLAS (LUCID) located at a distance of 17 m in both directions from the interaction point. The second system is a zero degree calorimeter (ZDC), which is located at a distance of $\pm$ 140 m from the interaction point. This corresponds to the location, where the LHC beam pipe is divided in two and the ZDC is located between the beam pipes just after the split inside an absorber. The most remote detector is the ALFA system at a distance of $\pm$ 240 m from the ATLAS interaction point, it is assembled with the same technology as the TOTEM detector. The main purpose of LUCID is to detect inelastic pp scattering in the forward direction, both in order to measure the integrated luminosity of the ATLAS runs and to monitor luminosity and beam conditions. While LUCID will only provide a relative luminosity measurement, ALFA on the other hand provides an absolute luminosity measurement by measuring the trajectory of protons elastically scattered at very small angles.

3.1.5 The Muon Spectrometer

The ATLAS *Muon Spectrometer* (also referred to as *Muon System*) is the outermost part of the ATLAS detector and builds a high precision stand-alone tracking detector. It follows an eight-fold azimuthal symmetry given by the incorporated toroid barrel and endcap magnets. It expands over a pseudorapidity range of $|\eta| < 2.7$ and exceeds hereby slightly the acceptance region of the inner detector. The complex MS magnet system is the most eye-caching part of the ATLAS detector, and it marks also one of its most revolutionary approaches. It builds together with precise tracking chambers and the mostly air-filled structure of the MS that incorporates a relatively low material budget a high precision tracking detector and fast muon trigger that includes a momentum measurement.

The MS Magnet System

Three toroidal structures, a large barrel toroid and two endcap structures of each side of the detector build the complex magnetic field of the Muon Spectrometer. The barrel toroid consists of eight coils that are contained in stainless-steel vacuum vessels with a tube diameter of 1.1 m, a length of 25.3 m and a radial width of about 5.4 m. The three layers of barrel

chambers are partly embedded, partly outside the toroid coil structure. The barrel toroid covers also the entire calorimeter volume and expands over the two smaller endcap toroid magnets, which creates a highly inhomogeneous magnetic field that puts stringent conditions to the extrapolation software, see Chap. 7. The inhomogeneity of the magnetic field in the Muon Spectrometer in both azimuthal and polar direction is illustrated as a simplified field map in Fig. 3.9

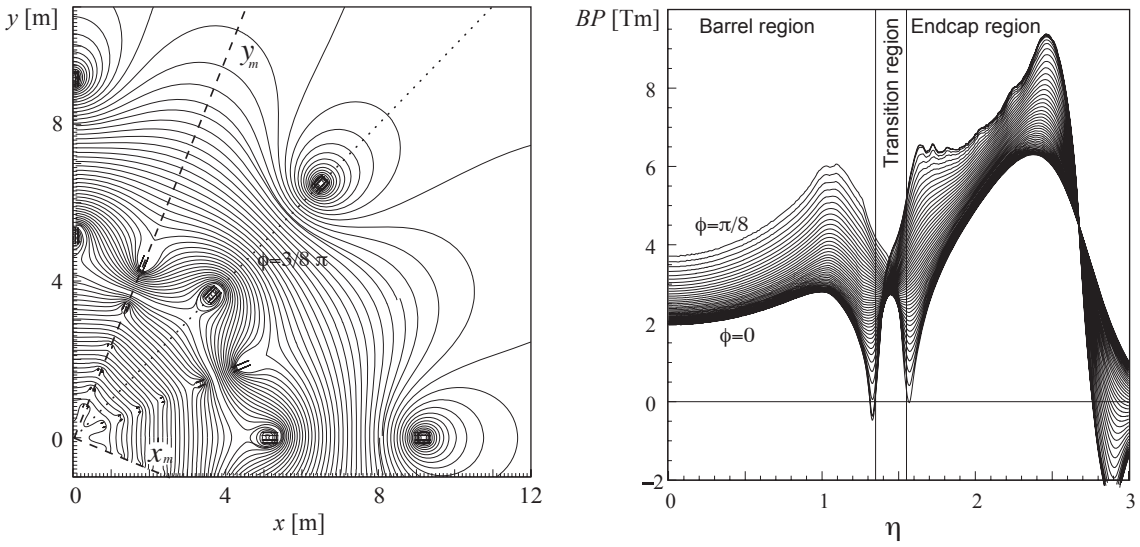


Figure 3.9: The magnetic field in the ATLAS Muon System: one quarter of the barrel magnetic field with its eight-fold symmetry is shown to the left. The right plot shows a the dependency of the magnetic field along the beam direction, including the highly inhomogeneous transition region between the barrel and one endcap toroid.

The MS combines precision chambers with trigger chamber technologies, leading to high precision measurements while providing fast muon trigger possibilities at the same time. The basic components of precision measurements are *Monitored Drift Tubes* (MTDs), i.e. pressurised drift tube with a diameter of 30 mm, operating with Ar/CO_2 gas (in a mixture of 93:7) at 3 bar. At higher pseudorapidity ranges (2 to 2.7), the MDTs are replaced by *Cathode Strip Chambers* (CSCs), that are taken because of their finer granularity and smaller time resolution to cope with the higher track densities.

The Muon System hosts in addition to the precision chambers two trigger chamber technologies that provide fast information of muon tracks to be used in first level trigger logics. *Resistive Plate Chambers* (RPCs) and *Thin Gap Chambers* (CSCs) provide in the barrel and endcap, respectively, discrimination on muon transverse momentum, bunch crossing identification, and also a fast but coarse tracking information for the region of interest building to be used in the higher level trigger stages. RPC and TGC measurements are not only used during the triggering phase, but provide also missing coordinate measurements for the precision chambers.

An overview of the four chamber technologies integrated in the Muon System can be seen in Fig. 3.10, Tab. 3.2 summarises the main parameters of the integrated technologies.

The combination of all sub systems of the MS allows to perform momentum measurements with a precision ranging from 2-3 % for muons between 10 GeV and 200 GeV, increasing to about 10 % for 1 TeV muons and exceeds hereby the achieved momentum resolution for high energetic muons in the ID. In general, a combined muon reconstruction of the two

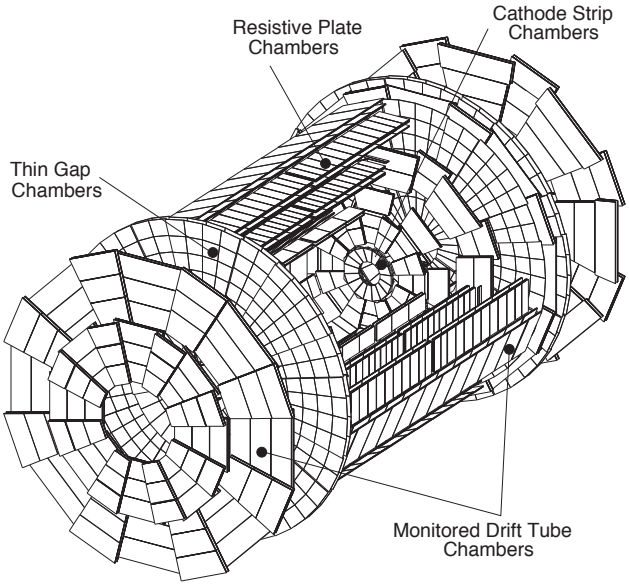


Figure 3.10: Overview of the four different chamber technologies integrated in the ATLAS Muon System. The precision chambers (Monitored Drift Tubes at lower η and Cathode Strip Chambers in the forward regions) are accomplished by two trigger system chambers, the Resistive Plate Chambers in the barrel and Thin Gap Chambers in the endcap region.

Table 3.2: Parameters in the four subsystems of the muon detector. Numbers in bracket are for the initial phase of the experiment.

	Resolution			**Hits/track**		**Numbers of**	
	η	ϕ	time	barrel	endcap	chambers	channels
Precision Ch.							
MDT	35 μm		750 ns	20	20	1172 (1108)	354k (339k)
CSC	40 μm	5 mm	4 ns		8 (4)	64 (32)	61.4k (30.7k)
Trigger Ch.							
RPC	3 cm	3 cm	20 ns	6		622 (560)	373k (359k)
TGC	0.7-3.6 cm	2-3 cm	20 ns		8	3588	318k

tracking devices in the ATLAS detector is highly desirable, which should also be reflected in the reconstruction software realisation, see Chap. 8.

3.2 The Trigger and Software Challenge

3.2.1 Triggering and Event Selection

Storing the detector data at a given bunch crossing rate of 40 MHz would be an impossible task for the ATLAS experiment: on the one hand, the readout systems and the writing of the data to disk can not be operated at such a high frequency, and on the other hand the disk storage which would be necessary for such a recording rate is simply not available. When assuming an anticipated raw data size of about 1.5 Megabyte per event, a 40 MHz recording rate would correspond to the writing of 60 Terabyte per second. ATLAS deploys therefore a complex trigger and data acquisition system, which must reduce the event rate from the bunch crossing rate of 40 MHz to around 200 Hz for recording onto mass storage. By doing this, dedicated focus is drawn on *interesting* events to be recorded: inelastic proton-proton collisions have for example a cross section of 70 mb, whereas the cross section for typical Higgs discovery channels are in the order of 0.1 pb, and therefor 700 million times less

likely.
The ATLAS trigger system [3.14, 3.15] consists of a three-level structure:

- **Level 1** (LVL1): a synchronous system using dedicated hardware and readout system and working on calorimeter and muon data only, searching for high p_{T} muons, electrons/photons, jets and τ-leptons that decay into hadrons. In addition, a dedicated trigger searches for large missing transverse energy in the event. The LVL1 trigger delivers a yes/no signal for each bunch crossing after around 2 microseconds. During the processing time, all data has to be stored in data pipelines to be able to read the full detector information whenever a positive LVL1 trigger decision has been drawn. The LVL1 has to reduce the event rate to maximal 100 kHz.
- **Level 2** (LVL2): an asynchronous system using programmable processors, looking in more detail at *regions of interest* (ROI) found by the LVL1 trigger. Data from all detectors (calorimeter, muons, tracking) is used at this stage, which reduces the trigger rate from around 100 kHz to below 3.5 kHz. The LVL2 trigger has an average processing time of 40 ms.
- **Event Filter** (EF): an event building and reconstruction system using farms of commercial processors, reducing the event rate to around 200 Hz. The EF plays a prominent role in the context of track reconstruction: it is effectively identical to the new track reconstruction chain (*offline* reconstruction), that will be in more detailed in Chap. 8 of this document.

The main difference between the offline reconstruction and LVL2/EF application is that latter is executed on only a limited set of the ID input data, that is given by the refined ROI information provided by the LVL1 trigger. This requires a smart on-demand unpacking of the bytestream data that is read out from the detector. As yet another consequence, the main pattern recognition modules have to run in a ROI-seeded mode and on the full event input collection and the processing time of the involved modules has to be kept as low as possible. Since the new track reconstruction has been realised in a modular component pattern design it facilitates the exchange of individual modules, in case e.g. they proof to be an unacceptable CPU time overhead for the EF.

3.2.2 Offline Event Processing and Grid Computing

The successive step to the event filtering is the offline reconstruction. It builds a major part of the ATLAS software suite and is thus subject to stringent requirements, see e.g. [3.16]. A main reason for this is the complexity and scale of the ATLAS experiment, that is evidently also reflected in the software applications. These have to be highly modular and robust to facilitate an adaption to changed physics goals and even different hardware instrumentations[7]. In addition having a modular software framework seems to be a suitable working model for a collaboration that is spread all over the world. The specific needs of such a huge and geometrically disperse collaboration are also reflected in the ATLAS computing model: to facilitate coherent software development while assuring high stability at the same time, regular nightly builds of the entire software suite are done between the various release

[7] It is of no doubt that both, the physics goals and the hardware instrumentations of the ATLAS detector will change during the lifetime of the experiment. For latter, i.e. eventual upgrade scenarios in compound with an according upgrade of the accelerator, research and planning has already started. In Chap. 9, a small discussion of such an upgraded geometry will be given in context with the presented fast track simulation program.

cycles. Final, production releases are put together regularly and deployed on world-wide computing sites.

The ATLAS software framework used for simulation, reconstruction and event analysis is called ATHENA and is to more detail described in Chap. 8, Sec. 8.2, of this document. In this context, a broader review of the deployed component model and abstract interface structure will be given.

Grid Computing

In a full year of operation, ATLAS is going to take 2 PB of raw data that have to be processed. In addition, large scale Monte Carlo simulation (and successive reconstruction) has to be performed. It is impossible that one central computing facility provides sufficient computing power and disk storage to all attached institutes and collaborative members. In the past years [3.17–3.19] ATLAS has gained experience in grid computing with a dedicated hierarchy system that consists of Tier sites on different levels:

- the **Tier-0** is the highest node in the grid hierarchy; it marks the interface of the experimental online system with the entire grid system (i.e. it is the site at which raw data is taken). For the LHC experiments, CERN is the Tier-0 facility;
- the **Tier-1** centers are the next level in the grid hierarchy. At Tier-1 nodes the reprocessing of raw data takes place, and they provide disk and tape storage for raw data, *Event Summary Data* (ESD) and *Analysis Object Data* (AOD);
- **Tier-2** are typically regional computing facilities at universities; they are foreseen for Monte Carlo production, provide disk storage for AOD and computing units for centralised analyses;
- the lowest recognised computing facility in ATLAS will be **Tier-3** sites that put a dedicated focus on end-user analysis that is purely based on AOD and *Derived Physics Data* (DPD). In current figures, a dedicated *CERN Analysis Facility* is planned to serve about 100 persons, while the number of users that need computing access for data analysis is by more than a magnitude higher.

A direct consequence of a world-wide computing system is the need for a smart and reliable data transfer and cataloging infrastructure. This collaboration specific applications have to be interfaced by an abstract layer with actual computing sites, such that resources can be accessed remotely by client software without needing to know the system configurations. This *middle* layer is often referred to *grid middleware*. In ATLAS, many grid tools that allow distributed analysis and remote data access has been developed and deployed by ATLAS in cooperation with the *LHC Computing Grid* (LCG) project and with the middleware providers of the three large grid infrastructures in use: EGEE [3.20], OSG [3.21] and NorduGrid [3.22]. A detailed discussion of this topic would go far beyond the scope of this thesis.

Bibliography

[3.1] The ATLAS Collaboration. *ATLAS Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN*. CERN/LHCC/94-34, 1994.

[3.2] The ATLAS Collaboration. *Liquid Argon Calorimeter Technical Design Report*. CERN-LHCC-96-41, 1996.

[3.3] The ATLAS Collaboration. *Tile Calorimeter Technical Design Report*. CERN-LHCC-96-42, 1996.

[3.4] The ATLAS Collaboration. *Inner Detector Technical Design Report - Volume I*. CERN-LHCC-97-16, 1997.

[3.5] The ATLAS Collaboration. *Inner Detector Technical Design Report - Volume II*. CERN-LHCC-97-17, 1997.

[3.6] The ATLAS Collaboration. *Muon Spectrometer Instrumentation Technical Design Report*. CERN-LHCC-97-22, 1997.

[3.7] The ATLAS Collaboration. The ATLAS Experiment at CERN. *ATLAS Communication, to be published in: Journal of Instrumentation*, ATL-COM-PHYS-2007-102, 2007.

[3.8] M. Mass. *ATLAS Pixel Module - der Aufbau und deren Tests im Labor und im Pionenstrahl*. PhD Thesis, Universitat Dortmund, 2005.

[3.9] S. Mehlhase and T. Petersen. A probability based approach to PID in the TRT detector of ATLAS. *ATLAS Communication*, ATL-COM-INDET-2006-017, 2007.

[3.10] G. Gorfine. Detector Description Versioning. ATLAS Twiki Website, 2007. https://twiki.cern.ch/twiki/bin/view/Atlas/InDetGeometryVersions.

[3.11] H. A. Bethe and L. C. Maximon. Theory of Bremsstrahlung and Pair Production. I. Differential Cross Section. *Physical Review*, 93, 1954.

[3.12] A. Korn. *Measurement of B-hadron Masses at CDF Run II*. PhD thesis, Massachusetts Institute of Technology, 2004.

[3.13] F Gianotti. Precision physics at LHC. *Public ATLAS Note*, ATL-PHYS-99-001, 1999.

[3.14] The ATLAS Collaboration. *ATLAS first Level Trigger, Technical Design Report*. CERN-LHCC-98-14, 1998.

[3.15] The ATLAS Collaboration. *ATLAS higher Level Triggers and DAQ, Technical Design Report*. CERN-LHCC-2003-022, 2003.

[3.16] The ALTAS Collaboration. *ATLAS Computing, Technical Design Report*. CERN-LHCC-2005-022, 2005.

[3.17] A. Putzer. A Step Towards A computing Grid For The LHC Experiments : ATLAS Data Challenge 1. *CERN Report*, CERN-PH-EP-2004-028, 2004.

[3.18] The ATLAS DC group. Atlas data challenge 2 : A massive monte carlo production on the grid. *Max Planck Society - eDocument Server:*, 2005. http://edoc.mpg.de/312203.

[3.19] D. Quarrie. Computing System Commissioning. ATLAS Twiki Website, 2008. https://twiki.cern.ch/twiki/bin/view/Atlas/ComputingSystemCommissioning.

[3.20] Egee homepage. Website. http://public.eu-egee.org.

[3.21] Open science grid homepage. Website. http://www.opensciencegrid.org.

[3.22] Nordugrid homepage. Website. http://www.nordugrid.org.

Searchin' high,
searchin' low
Searchin' everywhere I know

Bob Dylan - Dignity

Chapter 4

Track Reconstruction

Track reconstruction is one fundamental part of the event reconstruction in high energy physics experiments. In a naive picture it can be divided into the procedure of finding track candidates, the *pattern recognition*, and the estimation of the parameters that describe the particle trajectory, the *track fit*. A charged particle — when traversing through the detector — leaves a trace through electronic signals (*hits*) on sensitive detector elements. In pattern recognition, the collection of hits has to be found that has been caused by one single particle. Misidentified hits (in the following called *fake hits*) need to be avoided, since they usually decrease the final track resolution or may even lead to wrong track signatures and — consequently — sometimes to a wrong particle identification. The collection of hits are then further processed in the track fit to estimate the associate trajectory parameters. In the track fit, some level of discrimination can be applied that helps to eliminate or flag fake hits by their unproportional contribution to fit quality measures, such as the χ^2 of the fit. A relatively clean input sample provided by the pattern recognition that is characterised by high efficiency while containing only few *ghost tracks*[1] is, however, required not only for an optimisation of the computing time, but also for the stability of track fitting process.

The classical picture of separating the pattern recognition from the track fit became outdated in modern track reconstruction algorithms, in particular the boundary between local pattern recognition and track fitting vanishes in various applications. Quality checks have to be performed whether a found hit is compatible with others under a single track hypothesis, thus very often a simplified track fit is already performed at (local) pattern recognition level.

4.1 Tracking Detectors

In general, the trajectory of a particle can not be directly measured, but only a localisation (with a given uncertainty) of the particle at several discrete points in the detector volume can be done. In ATLAS, two conceptually different types of tracking devices are deployed: *cluster* measurements on planar surfaces and *drift* measurements through drift tube detectors. While clusters can describe both one-dimensional and two-dimensional local measurements and can be directly integrated into the track fit, the localisation through drift tube measurements is usually ambiguous and can only be resolved through a hit collection, or by using an already well defined track candidate seed. This is, because the measurement given by a drift tube is the drift time, i.e. the time which the electron shower needs to drift to the centrally positioned straw, and thus only a circle around the straw center can be defined. This has

[1] Collection of hits that can not be associated with one single particle

indeed implications on both track finding and track fitting, where the missing coordinate measurement along the straw often increases the number of track candidates significantly. In addition, fits may remain remain unconstraint in one to several parameters on the other hand[2]. The latter is in particular a problem for fitting techniques that rely on a measurement prediction, since the transport of undefined parameters can not be done.

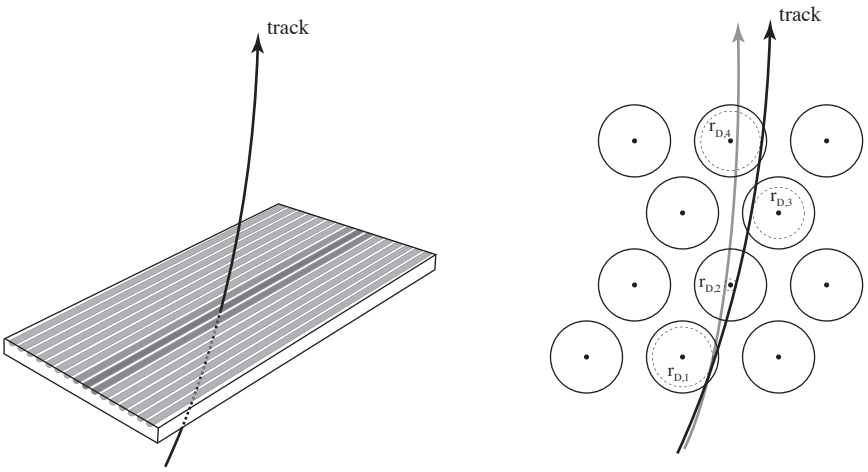


Figure 4.1: Schematic illustration of cluster and drift circle measurements. The left picture shows the principle of a cluster measurement through a segmented planar surface. The resolution that can be achieved is often higher than the segmentation of the measurement device since an interpolation of several neighboring cells can be done. The illustration to the right shows several drift circle measurements: the correct track candidate can hereby be only found though a collection of drift circle measurements or a well defined track seed. The illustration shows how a possible second track candidate that is still compatible with the drift circle measurements in tube 1 and 2 is ruled out by drift measurements in the layers 3 and 4.

Cluster measurements, on the other hand, lead usually directly to measurements of the local coordinates: a segmented detection device (this can be a silicon pixel or strip structure as in the Inner Detector, or Cathode Strip Chambers in the Muon Spectrometer) is used and the cells that are activated by a traversing track are used to build a measurement. This process (in Chap. 3 introduced as clusterisation) leads usually to a resolution higher than the intrinsic segmentation of the measurement device, since an interpolation of the several read out cells can be performed. Figure 4.1 shows a schematic illustration of both cluster and drift circle measurements. An example of a geometrical clusterisation model used in the context of a newly established fast simulation application is described in Chap. 9 of this document.

The ultimate goal of a tracking detector is to provide measurements that determine the trajectory in such a way, that both the particle origin (vertex) and the initial momentum can be estimated. Latter is usually carried out through a curvature measurement in magnetic field. In any case, the influence of the tracking detector on the particle trajectory has to be minimised. In track reconstruction — since the disturbance of the original trajectory is mainly due to the interaction of the particle with the traversed detector material — this is equivalent to minimising the material budget of both detection devices and support structures.

[2]In Chap. 6 the technique of introducing *pseudo-measurements* will be discussed briefly that are only defined to constrain fits such that convergence can be achieved.

4.2 Track Finding

The quality of a track reconstruction application can be judged on two parameters: the track reconstruction efficiency and the track parameter resolutions. They reflect — in a classical picture — the two different tasks of track finding and track fitting, latter is further discussed in Sec. 4.3.

There exist many different techniques that can be applied in the pure pattern recognition process, most of the classical methods incorporate a global approach[3]. The task of pattern recognition is to find subsets of data objects that *belong* together, i.e. they can be classified based on an *a priori* knowledge or on statistical information that can be extracted from the pattern itself. Both methods are implemented in the ATLAS track reconstruction software; the first technique is realised in the road building process based on track seeds in the very first stage of track finding. The second one is e.g. used in the TRT standalone segment finding, where the technique of a *Hough transformation* that has been initially developed for analysing bubble chamber photographs [4.1] is applied. Pattern recognition is always very specific to the given detector setup and a general discussion would go far beyond the scope of this document, the author will thus restrict the discussion to the specific implementations and methods used in the new ATLAS track reconstruction chain, that can be found in Chap. 8. The interested reader may, however, find a broad review of this topic in [4.2].

4.3 Track Fitting

Track fitting describes the estimation of the track parameters from a given set of measurements. In reality, the measurements can only be given with an uncertainty that originates from the intrinsic detector resolution and the positioning uncertainties of the detector elements due to *misalignment*. It is also necessary to define an underlying track model that is, in general, the solution of the transport equation in the given detector setup. In idealised cases, e.g. in absence of magnetic field or in a homogeneous magnetic field, an analytical track model can be applied. For the ATLAS experiment with its complex magnetic field configuration this is not possible. In most applications, the track model is thus given through the appropriate solution of the transport equation using numerical methods. Since the particle undergoes successive interactions with the detector material, these effects have to be taken into account when performing the track fit. Conceptually, the track fit therefore combines the geometrical setup of the detector, the measurements and the algorithmic modules that provide the parameter transport through the magnetic field setup. In the ATLAS reconstruction software, these modules can be identified by the reconstruction geometry description, see Chap. 5, the event data model as presented in Chap. 6 and the track extrapolation engine, Chap. 7.

The following sections focus on the two most common fitting techniques, the *Least Squares Method* (also referred to as *Global χ^2 fit*), and the *Kalman Filter*. The integration of both techniques into the ATLAS offline reconstruction software, can be found in Chap. 8.

We will denote the state vector of the particle, i.e. the parameterisation of the trajectory at a given point or time as

$$\mathbf{x} = \mathbf{x}(s), \tag{4.1}$$

and use a parameterisation along the arc length s which is most suitable for track reconstruc-

[3] A non-classical approach would be the use of so-called *neural networks* that are trained to find specific patterns within an overwhelming data sample.

tion in collision experiments[4]. In track reconstruction, the state vector can — in general — not be measured directly, but only a localisation of the track at discrete measurement points in the detector is presented. These discrete measurements are in most cases given by the intersection of the trajectory with a detection module. Hence, the state vector $\mathbf{x}_i$ should be regarded as the track parameterisation at the ith measurement device. The measurement can then be described as a function of the state vector

$$\mathbf{m}_i = h_i(\mathbf{x}_i) + \epsilon_i, \tag{4.2}$$

where the function h_i — in the following referred to as *measurement mapping function* — describes the specific function of the detector device i that maps the trajectory intersection to an actual measurement (such as the local coordinates on a planar surface); ϵ_i is a vector of unknown random variables and describes the intrinsic measurement error of the detection device. In Chap. 6, Sec. 6.2, it will be shown that the ATLAS track parameterisation has been optimised such that the functions h_i for all detector devices are simply projection matrices that reduce the five parameters of the full state vector to the actually measured ones. In other words, the *measured* parameters are identical with the local coordinates on the surface. It is also possible to include entire track segments into the track fit, and thus also the directional parameters that are not subject of a direct measurement have to be included as *measured* parameters[5].

The exact track intersection with the module can usually not be measured. The measurement quality is limited by the intrinsic detector resolution: hence, there is an unknown actual displacement ϵ between the measured and the real intersection position. In the following, it is assumed that the measurement error is unbiased and has a finite variance, i.e.

$$< \epsilon_i >= 0, \tag{4.3}$$

and, respectively,

$$\mathbf{var}(\epsilon_i) = \sigma_i < \infty. \tag{4.4}$$

The covariance matrix of the measurement error will be in the following denoted as

$$\mathbf{cov}(\epsilon_i) = \mathbf{V}_i. \tag{4.5}$$

In addition we define the evolution of the state vector as

$$\mathbf{x}(s_i) = \mathbf{x}_i = f_{i-1}(\mathbf{x}_{i-1}) + \delta_{i-1}. \tag{4.6}$$

Equation (4.6) is referred to as *system evolution*, where f_{i-1} is called the extrapolation of the track state $\mathbf{x}_{i-1}$ to the state $\mathbf{x}_i$. In a first assumption, we restrict the track extrapolation to be a linear function. A detailed discussion about the consequences of a non-linear track propagator for the track fitting will be given in Sec. 4.3.1 and Sec. 4.3.2, respectively. The transport of the state vector through the detector introduces a random disturbance (δ_{i-1}) caused by interactions of the particle with the detector material. Following the terminology of filter theory these distortions are also called *process noise*. The random noise disturbance is omitted in the purely deterministic evolution of the state vector, simply as it is unknown and can not be determined. Omitting the δ_i, however, demands that the process noise is unbiased. We will denote the covariance matrix of the process noise as

$$\mathbf{cov}(\delta_i) = \mathbf{Q}_i, \tag{4.7}$$

[4] A small motivation for the choice of the free parameter can be found in Chap. 7, Sec. 7.2

[5] The realisation of such a generic model will be presented in Chap. 6 of this document. It allows completely abstract fitter implementations that are able to operate on any subset of the full track parameterisation.

and assume in addition that the process noise has finite variance. In Sec. 7.2 it will be demonstrated that not all aspects of the particle interaction with the detector material will be accounted as process noise. The average loss of kinematic energy will not be treated as process noise, but is integrated as a deterministic process into the equation of motion, while the uncertainties (*straggling*) of the applied energy loss correction indeed contribute to the process noise.

The task of track fitting is to find the set of initial parameters $\tilde{\mathbf{x}}_0$ (in track reconstruction of high energy physics experiment usually the track parameters expressed at the production vertex) that is — together with the provided track model — best compatible with the provided measurement set. This has to be done without introducing a bias to the initial parameters and should provide an initial track state that is characerised by the smallest possible uncertainties.

4.3.1 The Least Squares Method

The least squares method is a global fitting method, i.e. all measurements are fitted at once under a track model hypothesis. A *predicted* measurement $\mathbf{m}^{pred}$ can be expressed through an initial parameter set $\mathbf{x}_0$ by the system evolution, defined in Eq. (4.6), as

$$\mathbf{m}_i^{pred} = h_i(\mathbf{x}_i) = h_i(f_i(\mathbf{x}_0)). \tag{4.8}$$

In the global track fit, the optimal estimate $\tilde{\mathbf{x}}_0$ of the initial state vector $\mathbf{x}_0$ is determined by minimising the residuals

$$\mathbf{r}_i = \mathbf{m}_i - \mathbf{m}_i^{pred} \tag{4.9}$$

for each contributing measurement. This is done under consideration of the according measurement errors, i.e. the residuals are weighted by the intrinsic measurement error of the detection module. To quantify the single contributions per measurement to the global fit quality, we define the so-called χ^2 function of the track fit

$$\chi^2 = \sum_i \frac{\left[\mathbf{m}_i - h_i(f_i(\mathbf{x}_0))\right]^2}{\sigma_i^2}, \tag{4.10}$$

or — using Eq. (4.5) — in matrix notation

$$\chi^2 = \mathbf{r}^T \mathbf{V}^{-1} \mathbf{r}, \tag{4.11}$$

where the system evolution is assumed to be linear and the global residual vector can be written as

$$\mathbf{r} = \mathbf{m} - \mathbf{H}(\mathbf{F} \cdot \mathbf{x}_0). \tag{4.12}$$

Minimising Eq. (4.10) yields the fitted parameters $\tilde{\mathbf{x}}_0$; we will assume that both the system evolution f_i and the measurement mapping functions h_i are linear. If the measurement error complies with Eqs. (4.3) and (4.4), $\tilde{\mathbf{x}}_0$ is the optimal estimate of the system, i.e. it has *minimum variance*.

In track reconstruction, when a particle is propagated through magnetic field and the transport equation is the solution of a second order differential equation, the system evolution is generally non-linear[6]. The transport function of the system equation, Eq. (4.6), is therefore

[6]The measurement mapping functions h_i on the other hand can usually be expressed as linear function by an appropriate choice of the track parameterisation.

often linearised through an Taylor expansion to first order at an approximative solution that is given through the seed parameters of the fit $\mathbf{x}_{0,I}$, yielding

$$f(\mathbf{x}_0) = f(\mathbf{x}_{0,I}) + \frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_{0,I}}(\mathbf{x} - \mathbf{x}_{0,I}) + \mathcal{O}((\mathbf{x} - \mathbf{x}_{0,I})^2). \tag{4.13}$$

Material Effects Integration in the global Track Fit

In Eq. (4.8) the system evolution has been used without the random disturbances due to interaction of the particle with the detector material. While the deterministic treatment of energy loss (→ Chap. 7) is accounted for in the track extrapolation process, the stochastic behavior of the multiple scattering (and in some applications also energy loss effects) has to dealt with separately in the global χ^2 function. This is done by introducing the deflection angle θ_j as an additional parameter to the fit. The global χ^2 function then becomes

$$\chi^2 = \sum_i \frac{[\mathbf{m}_i - h_i(f_i(\mathbf{x}_0, \{\theta_j\}_i))]^2}{\sigma_i^2} + \sum_j \frac{\theta_j^2}{\sigma_{MS,j}^2}. \tag{4.14}$$

The contribution of the fitted scattering angle to the global χ^2 function has to be evidently regulated by the expected range of the scattering process due to the traversed material. In the broadest sense, the scattering contribution is therefore also dependent on the initial state vector $\mathbf{x}_0$, since the traversed material is clearly a function of the trajectory[7]. In some applications, such as dedicated electron fitting of in the presence of large amounts of material, also the energy loss can be introduced as a fit parameter. The according straggling error is then, in general, difficult to describe theoretically and is often taken from a parameterisation.

4.3.2 The Kalman Filter

While global fitting techniques have been amongst the first methods that have been used in the reconstruction of high energy physics events, iterative fitting methods have been only introduced to event reconstruction during the era of LEP data taking between 1989 and 2001.

The first *progressive* methods have been proposed by Billoir [4.3], still being — in the beginning — of limited performance in comparison to the widely spread least squares method. Only the realisation [4.4] that the progressive fit is identical to the first part of the well known Kalman filter application, a technique that has been used for many years in signal processing and radar tracking, and thus the integration of the full Kalman filter mechanism evoked its big success in track reconstruction of high energy physics experiments. A notation in analogy to [4.4] is in the following chosen for the convenience of the reader.

The Kalman filter is a recursive procedure for estimating the state parameters of a discrete, linear dynamic system. In track fitting, this system is given by the measurements $\{\mathbf{m}_i\}$ that are caused by one particle when traversing the detector while being disturbed through interactions with the detector material.

We will, as a first step, limit the discussion to the linear Kalman filter, i.e. we assume that the track extrapolation in the system evolution is a linear function of the state vector $\mathbf{x}$. The extension of the Kalman filter to a non-linear system will be discussed subsequently.

[7] It is worth mentioning that all successive predicted measurements $\{\mathbf{m}_i^{pred}\}$ to the scattering surface at s_j with $i > j$ have to take the scattered direction into account when evaluating the system evolution.

Kalman Filter Procedure The Kalman filter includes the measurement progressively into the fit and follows a three-step procedure:

- The **prediction** describes the state vector at *future* stage, taking all previous measurements into account. The predicted state vector will in the following be denoted as $\mathbf{x}_i^{i-1}$ and is given by system evolution

$$\mathbf{x}_i^{i-1} = \mathbf{F}_{i-1}\mathbf{x}_{i-1}. \tag{4.15}$$

  As discussed earlier, only the deterministic part of the effects originating due to interaction of the particle with detector material are taken into account in the parameter prediction. The stochastic noise — mainly respecting the multiple coulomb scattering of the particle, but also including the straggling function of the applied energy loss correction — is superimposed as additional uncertainties $\mathbf{Q}_{i-1}$ to the transported (i.e. *predicted*) covariance matrix

$$\mathbf{C}_i^{i-1} = \mathbf{F}_{i-1}\mathbf{C}_{i-1}\mathbf{F}_{i-1}^T + \mathbf{Q}_{i-1}. \tag{4.16}$$

- The **filtering** is the estimation of the *present* state vector, that takes all previous and the actual measurement into account. It follows directly to the prediction and yields the — with the actual measurement — *updated* predicted state vector

$$\mathbf{x}_i = \mathbf{x}_i^{i-1} + \mathbf{K}_i(\mathbf{m}_i - \mathbf{H}_i\mathbf{x}_i^{i-1}), \tag{4.17}$$

  where $\mathbf{K}_i$ is called the *Kalman gain matrix*[8] and represents the modification of the predicted state vector $\mathbf{x}_i^{i-1}$ by taking the measurement $\mathbf{m}_i$ into account. Commonly spoken, the updated state vector is created by building the weighted mean of the measurement prediction that includes all prior measurements and the measurement $\mathbf{m}_i$, where the according inverse covariances represent the weights of the single components, such that

$$\mathbf{K}_i = \frac{\mathbf{C}_i^{i-1}\mathbf{H}_i^T}{\mathbf{V}_i + \mathbf{H}_i\mathbf{C}_i^{i-1}\mathbf{H}_i^T}. \tag{4.18}$$

  The gain matrix can also be expressed as

$$\mathbf{K}_i = \mathbf{C}_i\mathbf{H}_i^T\mathbf{G}_i, \tag{4.19}$$

  when introducing the updated or filtered covariance matrix $\mathbf{C}_i$ that can be, similarly to the updated state vector, directly gained from the predicted covariance matrix

$$\mathbf{C}_i = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{C}_i^{i-1}. \tag{4.20}$$

  The filtered residual at the present surface is given by the predicted residual $\mathbf{r}_i^{i-1}$, the measurement mapping function and the Kalman gain matrix

$$\mathbf{r}_i = \mathbf{m}_i - \mathbf{H}_i\mathbf{x}_i = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{r}_i^{i-1}, \tag{4.21}$$

  and the covariance matrix of the filtered residual is then

$$\mathbf{R}_i = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{V}_i. \tag{4.22}$$

[8] We restrict the discussion of Kalman Filter to the gain matrix formalism since this is used by default in the ATLAS track reconstruction. The reader will find a full description of the both the gain matrix and the mathematical equivalent weight matrix formalism in the quoted references.

It should be mentioned that in Chap. 8, Sec. 8.3, where a general performance review of the newly established ATLAS track reconstruction chain is presented, so-called *unbiased* residuals for hit resolutions and pull distributions are used. The unbiased residuals describe the difference between the track parameterisation at a detection device i to the actual measurement, without taking the current measurement into account for the track parameterisation. Using the Kalman filter formalism, these unbiased residuals can be easily calculated by applying an *inverse* filtering step at any later stage of the track reconstruciton process. As a single requirement both the measurement error and the filtered covariance need to be accessible, which is supported by the reconstruction event data model, see Chap. 6.

In the Kalman formalism, the global χ^2 of the fit is incremented with each inclusion of an additional measurement, a feature which is in particular helpful to detect outliers or falsely assigned measurements to the track. It reflects the fact that the filtered residual vectors are uncorrelated (and, in case of Gaussian measurement errors and process noise even independent). The increment χ^2_+ is hereby defined by the squared filtered residuals weighted by the filtered covariance of the residuals

$$\chi^2_+ = \mathbf{r}_i^T \mathbf{R}_i^{-1} \mathbf{r}_i. \tag{4.23}$$

- The **smoothing** completes the Kalman filter formalism and represents the estimation of the state vector when regarding all measurements of the track. Clearly, the smoothing can only be done, after all measurements have been included through the filtering procedure. The filtering is therefore often referred to as *forward* Kalman filter, while the smoothing is done *backwards*. Only the inclusion of the smoothing procedure (which was missing in the first progressive methods as proposed by Billoir) turn the Kalman filter to be mathematically equivalent to the least squares method[9].

  When regarding a track with n measurements, the smoothed state vector $\mathbf{x}_i^n$ can be expressed as

  $$\mathbf{x}_i^n = \mathbf{x}_i + \mathbf{A}_i(\mathbf{x}_{i+1}^n - \mathbf{x}_{i+1}^i), \tag{4.24}$$

  using the difference between the predicted state vector $\mathbf{x}_{i+1}^i$ and the smoothed state vector $\mathbf{x}_{i+1}^n$ on the successive surface, such as the smoothing gain matrix

  $$\mathbf{A}_i = \mathbf{C}_i \mathbf{F}_i^T (\mathbf{C}_{i+1}^i)^{-1}. \tag{4.25}$$

  In absence of process noise, the smoothing is equivalent to a backward extrapolation.

The progressive nature of the Kalman filter supports its biggest benefits: on the one hand, since measurements can be added one-by-one in contrast to the global track fit, it is a powerful instrument to be used already at pattern recognition stage (→ *Combinatorial Kalman Filter*, see also [4.2]). The modular filtering concept facilitates multi-variant extensions, two specific concepts will be discussed further along. The Kalman filter requires in addition less CPU resources than global fitting methods. This is, because the matrices to be dealt with in the forward and backward filtering are of a maximal dimension equal to the track parameterisation (i.e., usually, five-dimensional), while the matrix to be inverted in the global fit grows successively with every single parameter that is included. As a consequence (and a rule of thumb), the CPU time increases linearly for the Kalman filter with every additional

[9] It seems intuitive, that only when respecting the contribution of all other measurements at each single filtering step, the sequential fitter yields the same performance as a global treatment of the whole measurement set at once.

hit that is added to the track, while the computing cost needed for matrix inversion in the global χ^2 minimisation increases rapidly with higher dimensions[10].

Extension and Seeding of the Kalman Fit

From Eqs. (4.15) to (4.25) one can learn that the the transported state vector is used as a whole, i.e. it is not necessary that the system evolution (track propagation) is a linear function of the state vector. In reality, the system evolution is given by the solution of the equations of motion. For a charged particle that traverses a magnetic field, this is a second order differential equation, given in Chap. 7. One can thus without restriction of generality replace Eq. (4.15) with the more general expression in Eq. (4.6) and give up the requirement of f_{i-1} to be linear. However, for the Kalman filter formalism also the transport of the covariance matrix has to be performed. In case of a non-linear propagation of the state vector, this is a non-trivial problem. The solution to this problem is to linearise the transport equation by using a Taylor expansion as given in Eq. (4.13). The transport of the track state covariances is then yielded by the jacobian matrix

$$(\mathbf{J}_{i,i+1})_{\mu\nu} = \frac{\partial(\mathbf{x}_{i+1})_\mu}{\partial(\mathbf{x}_i)_\nu}. \tag{4.26}$$

The evolution of the covariance matrix from measurement i to the measurement surface $i+1$ can then be denoted as

$$\mathbf{C}_{i+1} = \mathbf{J}_{i,i+1}^T \mathbf{C}_i \mathbf{J}_{i,i+1}. \tag{4.27}$$

Finding this jacobian matrix is not always a trivial task; an analytical track model would allow a fully analytical calculation of the transport jacobian matrix, while evidently a numerical solution of the system equation of the state vector would consequently require a numerical jacobian transport as well. Both solutions are realised in the ATLAS track extrapolation software and discussed in very detail in Chap. 7. When linearising the system evolution for the covariance transport (or any other non-linear component of the Kalman filter) one usually speaks of the *Extended Kalman Filter* (EKF). The EKF is restricted to the regime where a (local) linear approximation is valid to some extend. This has, in general, little impact on the fit itself, with the single consequence that a qualitative seed is needed to initiate the filtering procedure. Since the Kalman filter follows then very closely the measurements, the linear approximation has to be valid only in small ranges.

In the last years, the standard Kalman filter formalism has been extended to suite particular situation in the track reconstruction, two of which will be described briefly in the following, since they have been deployed in the ATLAS reconstruction software.

The Deterministic Annealing Filter In an environment such as the LHC, track fitting suffers mainly from wrong or ambiguous hit assignment due to the high hit occupancy in jets and pile-up events. Also, a significant noise fraction is expected that can be — in case of δ-electrons — even correlated to the track. The *Deterministic Annealing Filter* (DAF) [4.6] is an adaptive extension of the Kalman filter that has been optimised for such situations. It integrates a robust model of fuzzy hit assignment that is resolved only within the track fit. Per measurement surface, more than one measurement can be taken into account, each of which weighted to a compound of measurements by the according residual to the predicted

[10] The most common LU decomposition method requires — when being fully deployed — N^3 loop operations for the inversion of a $N \times N$ matrix, where the exponent can be slightly decreased to $\log_2 7$ when using more speed-optimised methods [4.5].

track state. These weights will in the following be denoted as p_i^k and will be used to calculate the assignment probability of measurement k on a measurement surface i to build the *effective measurement*. Following the notation introduced in the previous section, the effective measurement $\tilde{\mathbf{m}}_i$ and the effective covariance $\tilde{\mathbf{V}}_i$ are written as

$$(\tilde{\mathbf{V}}_i)^{-1} = \sum_k p_i^k (\mathbf{V}_i^k)^{-1} \tag{4.28}$$

$$\tilde{\mathbf{m}}_i = \tilde{\mathbf{V}}_i \cdot \left(\sum_k p_i^k (\mathbf{V}_i^k)^{-1} \mathbf{m}_i^k \right), \tag{4.29}$$

when the index k runs over all measurements that are included on the layer i.

The introduced effective measurements and covariances can be used in the standard Kalman filter formulae without any change. However, there is an additional computational aspect to be considered: the weighting of the individual measurements is done using the inverse covariance matrix (i.e. the weight matrix). To save unnecessary matrix inversions that are in general CPU time intensive it is beneficial for the DAF to rely on a Kalman weight updating formalism. A short discussion of this topic will be included in Chap. 8 of this document, where the integration of the DAF into the ATLAS track reconstruction is presented. Since in a final evaluation one of the measurements is assigned with highest weight to the track, the individual measurements are often called *competing measurements*, the integration of such an event data model schema can be found in Chap. 6. The remaining part of the DAF (and indeed the most important one) is the annealing mechanism: it describes the estimation of the assignment probabilities using the competing measurements. It has to be done in an iterative way, because the track information at an initial state is not accurate enough to correctly discriminate between correctly and falsely assigned hits. After a first turn of forward filtering and smoothing, however, the combined track states include the information of all hits and are precise enough to evaluate the assignment probabilities. Latter is done by superimposing a deterministic annealing schema on top of the filtering step. It is carried out by inflating the measurement covariance matrices with the use of an effective *system temperature* T that is iteratively cooled down, while the assignment probabilities are re-evaluated. The annealing schema is necessary to avoid local optima.

As a first step the system temperature $T = 1/\beta$ (with $\beta \in (0, 1]$) is introduced to inflate the covariances of the individual measurements, substituting $\mathbf{V}_i^k \to T\mathbf{V}_i^k$. The inflated covariances are now used to calculate the non-normalised assignment probabilities Φ_i^k, when assuming the measurement PDF to be proportional to a multi-Gaussian distribution and requiring Eqs. (4.3) and (4.4)

$$\Phi_i^k \propto \frac{1}{\sqrt{T\mathbf{det}(\mathbf{V}_i^k)}} \cdot \exp\left[-\frac{1}{2T} (\hat{\mathbf{r}}_i^k)^T \mathbf{V}_i^k \hat{\mathbf{r}}_i^k \right], \tag{4.30}$$

with $\hat{\mathbf{r}}_i^k$ being the smoothed unbiased residual for the measurement m_i^k. In Eq. (4.30) a technique from statistical mechanics is applied by introducing the assignment probability through a Maxwell-Boltzmann function. The effective energy of the system is hereby given through a quadratic potential of the unbiased residuum normalised by the measurement error, the thermal energy obviously regulated by T. The weights are then normalised to

$$p^k = \frac{\Phi_i^k}{\sum_j (\Lambda_j^k + \Phi_j^k)}, \tag{4.31}$$

when introducing an implicit cut-off value λ for the single measurements through

$$\Lambda_j^k \propto \frac{1}{\sqrt{T\mathbf{det}(\mathbf{V}_j^k)}} e^{-(\beta\lambda/2)}. \tag{4.32}$$

At low temperatures λ turns into an equivalent of a χ^2 cut.
The iteration is started with a high system temperature T that allows almost all measurements to comply with the given cut value, but is then cooled down to a thermal equilibrium that will favor one of the assigned hits. It can be shown that the best performance can be achieved if the system is not *cooled* down to the absolute 0 value, but some small level of fuzzy hit assignment remains allowed.

The Gaussian Sum Filter In all previous considerations it has been assumed, that the process noise and the measurement error are Gaussian and unbiased. In reality, however, purely Gaussian noise assumptions are always simplification of a different, more inconvenient truth. In Chap. 7 the Gaussian error modeling for the process noise in track reconstruction, the effects caused by interactions of the particle with detector material, and its validity range will be discussed. In particular electrons show strong non-Gaussian behavior due to radiative energy loss and a dedicated fitter that can cope with such changed input parameters is highly desired. The *Gaussian Sum Filter* (GSF) [4.7] includes the treatment of non-Gaussian noise (and measurement errors). It is based on the Kalman filter and follows the same procedure of progressive forward filtering and backward smoothing. In its most general form, both the measurement error and the process noise, are modeled as a mixture of multi-Gaussian distributions. In this case (and using the nomenclature introduced earlier), the predicted distribution of the state $\mathbf{x}_i$ at a surface i that depends on all previous observations $\mathbf{Y}_{i-1} = \{\mathbf{y}_1 ... \mathbf{y}_{i-1}\}$ is given through a Gaussian mixture with N_{i-1} components

$$p(\mathbf{x}_i|\mathbf{Y}_{i-1}) = \sum_{j=1}^{N_{j-1}} \pi_i^j \phi(\mathbf{x}_i; (\mathbf{x}_i^{i-1}))^j, (\mathbf{C}_i^{i-1})^j), \tag{4.33}$$

where $\phi(\mathbf{x}; \mu, \mathbf{V})$ is a multi-variant Gaussian distribution int $\mathbf{x}$ with mean μ and covariance $\mathbf{V}$. The single weights π_i^j have to fulfill the normalisation requirement $\sum_{j=1}^{N_{i-1}} \pi_i^j = 1$.
The distribution of measurement errors ϵ_i are also modeled as a Gaussian mixture $p(\epsilon_i)$ with E_i unbiased components

$$p(\epsilon_i) = \sum_{j=1}^{E_i} \lambda_j \phi(\epsilon_i; 0, \mathbf{V}_j), \tag{4.34}$$

with $\sum_{j=1}^{E_i} \lambda_j = 1$. The same multi-component description can be applied to the process noise, describing the scattering distribution at a certain intersected layer i depending on the effective material thickness $t_i(\phi, \theta)$ as

$$p(t_i) = \sum_{j=1}^{M_i} \alpha_j \phi(t_i; 0, \mathbf{Q}_j), \tag{4.35}$$

when α_j denotes the single weights of the components that fulfill normalisation, and M_i is the number of components that describe the scattering distribution at the layer i. The state arriving at the scattering layer i already is N-component state that follows Eq. (4.33). After applying the multiple scattering update, the two multi-variant probability density functions (PDFs) get convoluted

$$p(\mathbf{x}) = \sum_{j=1}^{N} \sum_{k=1}^{M_i} \pi_j \alpha_k \phi(\mathbf{x}; \mathbf{x}_j, \mathbf{C}_j + \mathbf{H}_{MS}^T (Q)_j \mathbf{H}_{MS}), \tag{4.36}$$

where $\mathbf{H}_{MS}$ describes the mapping function of the scattering parameters onto the track parameterisation. The application of energy loss is very similar to the multiple scattering

mechanism in the GSF. Energy loss, however, is usually treated as an effect with non-zero mean, i.e. a mean or most probable energy loss is directly applied to the state vector, or — in terms of the GSF — to the according component of the state vector. Equation (4.36), when being adapted for energy loss treatment, would then read

$$p(\mathbf{x}) = \sum_{j=1}^{N} \sum_{k=1}^{M_i} \pi_j \alpha_k \phi(\mathbf{x}; \mathbf{x}_j + \mathbf{H}_{el}\Delta E_j, \mathbf{C}_j + \mathbf{H}_{el}^T \mathbf{var} < \Delta E_j > \mathbf{H}_{el}), \tag{4.37}$$

again, with $\mathbf{H}_{el}$ denoting the mapping function for an explicit energy loss value to the track parameterisation. In reality, Eqs. (4.36) and (4.37) are often combined, since multiple scattering and energy loss are in general applied on the same detector layers. The multi-component state are at first processed in a parallel way, which creates the need of a multi component system evolution[11]. However, the successive convolution of multi-covariant PDFs would lead to an exponential increase of components. Component merging is therefore one additional important aspect of the GSF.

Bibliography

[4.1] V. C. Hough. Machine analysis of bubble chamber pictures. *Proc. Int. Conf. High Energy Accelerators and Instrumentation*, 1959.

[4.2] R. Mankel. Pattern recognition and event reconstruction in particle physics experiments. *Rept. Prog. Phys.*, 67:553, 2004.

[4.3] P. Billoir. Track Fitting with multiple scatterin: a new method. *Nucl. Inst. Meth.*, A225:352–366, 1984.

[4.4] R. Frühwirth. Application of Kalman filtering to track and vertex fitting. *Nucl. Inst. Meth.*, A262, 1987.

[4.5] W. H. Press, Vetterling. W. T., S. A. Teukolsky, and B. P. Flannergy. *Numerical Recipes in C++, Second Edition*. Cambridge Univeristy Press, 2003.

[4.6] R. Frühwirth, A. Strandlie, M. Winkler, and T. Todorov. Recent results on adaptive track and multitrack fitting. *Nucl. Instrum. Meth.*, A502:702–704, 2003.

[4.7] R. Frühwirth. Track fitting with non-gaussian noise. *Comput. Phys. Commun.*, 100:1–16, 1997.

[11] In the ATLAS reconstruction software, this requirement led to the development of a dedicated track extrapolation unit, that uses most underlying tools of the standard extrapolation engine, see Chap. 7, but could deal with a multi-component track parameterisation. The event data model had to be adapted for multi-component track states accordingly, see Chap. 6.

You know of course that a mathematical line, a line of thickness NIL, has no existance. [...] Neither has a mathematical plane. These things are mere abstractions.

H. G. Wells - The Time Machine

Chapter 5

The Reconstruction Geometry

5.1 Introduction

In Chap. 4, the importance of a geometry model for track reconstruction has been briefly mentioned, probably without notice by the reader: it builds the binding module between the algorithmic reconstruction parts and the event data model. Since tracking detectors are usually realised to provide discrete localisations of the particle trajectory, it is at first means necessary to have surface descriptions that allow to parameterise the track with respect to them. In ATLAS, dedicated `Surface` objects have been created that build a central part of the reconstruction geometry description and the event data model, which is further described in Chap. 6. The reconstruction geometry has — besides building the link between geometry information and the linear algebraic operations in track reconstruction — another important task: it provides the material description for the integration of material effects in the track fitting process. The correct estimation of the detector material will be crucial for many tracking-related studies, in particular in combination with electron tracking.

One particular aspect of the reconstruction geometry description is that is has to be capable of dealing with the realistic detector setup: this includes a coherent integration into the ATLAS alignment algorithms on the on hand, and a strategy to to cope with distorted or deformed surfaces on the other hand. The high precession of the ATLAS tracking devices requires an exact knowledge of the position and possible module distortion (due to mounting tension, gravitational effects or production uncertainties) in order not to limit the tracking performance.

The following section will present the new ATLAS reconstruction geometry model that has been deployed in the last three years. Although initially started as a description of the ID, it has been designed in a generic way, such that it has been easily expanded to describe the overall detector setup. Dedicated geometry descriptions for the combined testbeam run in 2004 and the various detector commissioning phases using comic rays have been created. Even a first proposed upgrade model of the ATLAS ID has been realised with the reconstruction geometry and used together with the new fast track simulation that will be presented in Chap. 9.

Recently [5.1] a new flexible model has been added that allows a custom integration of material effects (from other sources than the reconstruction material description) directly through the reconstruction geometry. In such a way it provides a framework for the inclusion of energy deposition measurements from the calorimeter to the extrapolation package, which is further described in Chap. 7.

5.2 The ATLAS Reconstruction Geometry Description

ATL-SOFT-PUB-2007-004, 31 p.

Acknowledgements

The author would like to thank Marcin Wolter and Sharka Todorova for their great contribution to the Muon Tracking Geometry and the detached volume schema.

Bibliography

[5.1] D. Lopez Mateos, E. W. Hughes, and A. Salzburger. A Parameterization of the Energy Loss of Muons in the ATLAS Tracking Geometry. *Public ATLAS Note*, ATL-MUON-PUB-2008-001, 2008.

The ATLAS Tracking Geometry Description

A. Salzburger*
Leopold Franzens Universität Innsbruck, Austria & CERN
S. Todorova
Tufts University, USA
M. Wolter
Institute of Nuclear Physics, PAN, Krakow, Poland

December 6, 2007

ATL-SOFT-PUB-2007-004
13 December 2007

Abstract

Track reconstruction requires a detector geometry description for the usage in track extrapolation processes and material effects integration during track finding and track fitting. Since, in general, the more realistic detector description used in full detector simulation causes an unacceptable increase of CPU time consumption when being used in track reconstruction, the reconstruction geometry is realised as a simplified description of the actual detector layout. This documents presents the data classes of the newly developed ATLAS reconstruction geometry and describes its building process for the ATLAS CSC detector layouts. Additionally a comparison of the material budget described by the reconstruction geometry with one used in full detector simulation will be presented for the Inner Detector and the calorimeter devices.

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*

*corresponding author: Andreas.Salzburger@cern.ch

1 Introduction

The abstraction of the complex detector geometry to a simplified version for the use within track finding and fitting is a common technique in track reconstruction of high energy physics experiments. A dedicated reconstruction geometry is not only necessary for the definition of the measurement surfaces, but also for the description of the detector material for the integration during track extrapolation processes and track fitting. In track reconstruction, the interaction of a particle with the traversed detector material is, in general, described by Gaussian or multi-Gaussian noise addition to the measurements to account for multiple scattering effects and deterministic mean or most probable energy loss with a small variance. The stochastic description together with the applied simplifications in the description of the material effects allow to use a simplified geometry model with an averaged material description. However, an accurate description of the material in both weight and spatial position within the detector is essential for high quality track reconstruction and hence for various measurements aimed to be done with the ATLAS detector. Evidently, a good balance between CPU time consumption (which directly corresponds to the level of abstraction) and physics performance has to be found.

During the redesign of the ATLAS offline reconstruction software, initially invoked by the *Final Report of the Reconstruction Task Force* (RTF) [1], a new detector description has been deployed to serve the needs of modern track reconstruction techniques. This reconstruction geometry will be henceforth referred to as `TrackingGeometry`. It is fully integrated in the object oriented C++ based ATLAS software framework ATHENA [2] — an enhanced version of the LHCb GAUDI framework [3] — and has been developed in coherence with the ATLAS tracking event data model (EDM) [4], respecting ATLAS coding standards [5].

This document concentrates on the ATLAS-CSC-00-00-00 detector description version, which is the base version for all layouts used within the ATLAS *Computer System Commissioning* (CSC). CSC layouts incorporate misalignment, realistic magnetic field and material distortions, enhancing detector performance studies as close as possible to the conditions to be found at startup of the experiment. The ATLAS `TrackingGeometry` and associated material descriptions also exist for legacy layouts to guarantee backward compatibility. Table 1 summarises the supported ATLAS layout versions. This document is based on the ATLAS software release 12.0.6.

Table 1: Supported ATLAS detector layouts by the `TrackingGeometry`. For each of the supported layouts, the material description file can be retrieved from the ATLAS conditions database (COOL). Since some of the layouts differ only slightly in terms of material budget, material files have been mapped solely for root layout types. All other supported layout tags are linked to these root layouts. Although the ATLAS-CSC-01-02-00 layout introduces distortions in the material distribution, it is per default described through the non-distorted root layout, since this layout type is dedicated for evaluating the effect of a wrong material description — and in the best cast — for developing a material calibration technique.

Layout	**Comments**	**Base Layout for Material**
ATLAS-Rome-Initial-01	-	ATLAS-DC3-02
ATLAS-Rome-Initial-02	used for Rome production	ATLAS-DC3-02
ATLAS-DC3-02	production layout of 11.0.X	-
ATLAS-DC3-04	-	ATLAS-DC3-05
ATLAS-DC3-05	Inner Detector material update	-
ATLAS-DC3-06	-	ATLAS-DC3-07
ATLAS-DC3-07	TRT material update	-
ATLAS-CSC-00-00-00	-	ATLAS-CSC-00-00-00
ATLAS-CSC-00-01-00	misalignment (MA)	ATLAS-CSC-00-00-00
ATLAS-CSC-01-00-00	realistic magnetic field (RMF), no MA	ATLAS-CSC-00-00-00
ATLAS-CSC-01-01-00	RMF, MA	ATLAS-CSC-00-00-00
ATLAS-CSC-01-02-00	RMF, MA , material distortions (MD)	ATLAS-CSC-00-00-00

1.1 Design Principles and Document Structure

Two main design principles have been followed while developing the new `TrackingGeometry`: a rigorous separation of mathematical objects and physical or logical entities and a sub-detector technology independent implementation to enhance its usage in common tracking tools that do not acess detector specific information. `Surface` and `Volume` classes are designed as purely geometrical constructs and are located in the TrkSurfaces and TrkVolumes packages of the Tracking software repository, respectively; the contained classes of these packages are described in detail in Sec. 2 and Sec. 3 of this document. The purely geometrical objects are extended to `Layer`, `TrackingVolume` and `DetachedTrackingVolume` classes that add material and/or magnetic field information. These extended classes are contained by the TrkGeometry package and are further described in Sec. 4. The `Layer` and `TrackingVolume` classes provide the general pattern for the internal navigation schema used during track extrapolation processes. The existence of two different volume representations reflects the strategies that are followed in describing the sub-detectors: while the *Inner Detector* and the calorimeter volume are described by a full connective setup, the *Muon System* with its very complex geometry and volume shapes requires a different concept for material description the navigation. The process of building the sub-detector geometries and the combination to one final reconstruction geometry is described in Sec. 5. Since a full description of the ATLAS detector would go far beyond the scope of this document, the reader is invited to find further details about the ATLAS detector geometry, technologies and setup in [6]. Sec. 5, however, assumes that the reader is familiar with the main concepts of the ATLAS detector. Section 6.1 covers the material association to the layers and volumes of the ATLAS `TrackingGeometry` and gives a comparison of the material budget with the amount of material given by the geometry description of the full ATLAS detector simulation, which is based on the Geant4 simulation toolkit [7]. Section 7 refers to prototype extensions of the TrkDetDescr package and Sec. 6 describes the automatic testing of `TrackingGeometry` building and visualization methods for debugging.

The appendix of this document explains nomenclature, the typesetting within this document and gives a brief description of the ATLAS software framework components used within this context.

2 The Surface Description

The `Surface` classes are the main components of all further geometrical objects defined by in the new `TrackingGeometry`. Volumes and layers are designed as composites respectively extensions of the `Surface` class. Furthermore, surfaces play a key role in the structure of the common ATLAS tracking EDM. They serve as the input to track extrapolation and give therefore the natural reference frames for track representations. The expression of a track with respect to a given surface type is realised through according track parameter classes. Additionally, all *local-to-global* transformations in track finding and fitting are delegated to the surface to guarantee consistency between the used coordinate systems. Finally, the `Surface` class also builds the interface of reconstruction algorithms to the common ATLAS detector description GeoModel [8]: in ATLAS, a common geometry database is used and a GeoModel detector description is built from this source. In full detector simulation (performed with the Geant4 toolkit) this description is converted into a Geant4 description that is understood by the simulation program[1].

Since for reconstruction a direct translation from GeoModel, or - even more - a direct use of either GeoModel or Geant4 would result in an too complicated description, a simplification of the geometry has to be done. However, the positions and extends of the sensitive detector elements are required to the same detail as in the simulation process. Thus, surfaces representing actual measurement surfaces are directly bound via GeoModel to the ATLAS detector description database. This mechanism also guarantees the correct alignment of the surfaces respecting the latest conditions data.

Respecting the ATLAS detector layout and the requirements imposed by global and local track fitting algorithms, five different surface types have been implemented in the new geometry description. The different surface types exist as individual classes extending a common `Surface` base class. The base class concentrates the minimal required information for the definition of a surface:

- a center position and a rotation with respect to the global frame, implemented as an instance

[1] This is performed by a dedicated `Geo2G4` converter.

of the CLHEP [9] object `HepTransfrom3D`

- a boundary shape description, implemented as an inherited object of the class `SurfaceBounds`

Figure 1 shows the UML inheritance diagram for the various `Surface` classes.

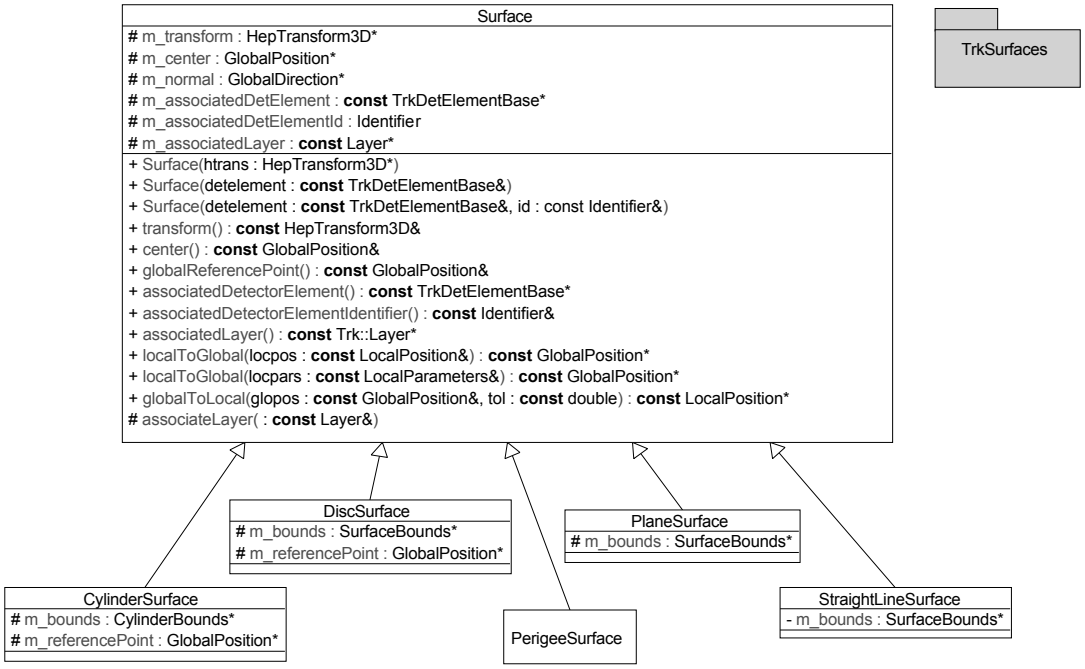


Figure 1: Simplified UML diagram for the surface classes of the ATLAS (tracking) detector description. The base class constructors indicate the two different surface types **private** and **public** : `Surface` instances associated to a `TrkDetElementBase` are **public** surfaces. They are owned and managed by the central ATLAS detector description GeoModel. **Private** surfaces can either be constituents of the `TrackingVolume` instances or reference surfaces for EDM objects that are not bound to sensitive detector elements. The diagram also shows the main local to global transformation methods provided by the surface interface. These transformations are overloaded by the extended child classes respecting the internal or natural surface coordinate system.

2.1 Local Frame Definitions

The choice of different surface representation has been based, in general, on the existence of a unique intrinsic local coordinate system. This is important to establish a coherent definition of the track parameterisation with respect to the measured coordinates given by the detection device. A planar surface of rectangular shape is therefore expressed by the same class as a planar surface of trapezoidal shape as both are characterised by a local cartesian coordinate frame. The two instances of type `PlaneSurface` would then differ simply by diverse `SurfaceBounds` types. A disc-like planar surface on the other hand is represented by a dedicated `DiscSurface` class to respect the intrinsic polar coordinate frame. This concept is broken for performance reasons by the introduction of the `PerigeeSurface` class, which is conceptually identical with the `StraightLineSurface` class, but restricts the orientation of the surface to be parallel to the nominal z axis. Figure 2 shows an illustration of the existing different surface and the main boundary types. Table 2.1 lists the different surface descriptions and the associated local frame definitions.

The local frame is not necessarily a cartesian coordinate frame, however, each local frame is based on a three-dimensional cartesian helper frame, $H = (\mathbf{h}_x\,, \mathbf{h}_y\,, \mathbf{h}_z\,)$, that defines the position and rotation of the surface with respect to the global frame. Cylindrical, pseudo-cylindrical[2] and polar coordinate

[2]The system of a `PerigeeSurface` or `StraightLineSurface`, where the local coordinates describe the transverse respectively longitudinal closest approach is called pseudo-cylindrical in this context.

systems referred to as *local coordinate frames* are deduced from this helper frame; the following conventions have been imposed:

- the normal vector of planar surfaces points along the $\mathbf{h}_z$ direction
- the symmetry axis for cylinders or line surfaces points along the $\mathbf{h}_z$ direction

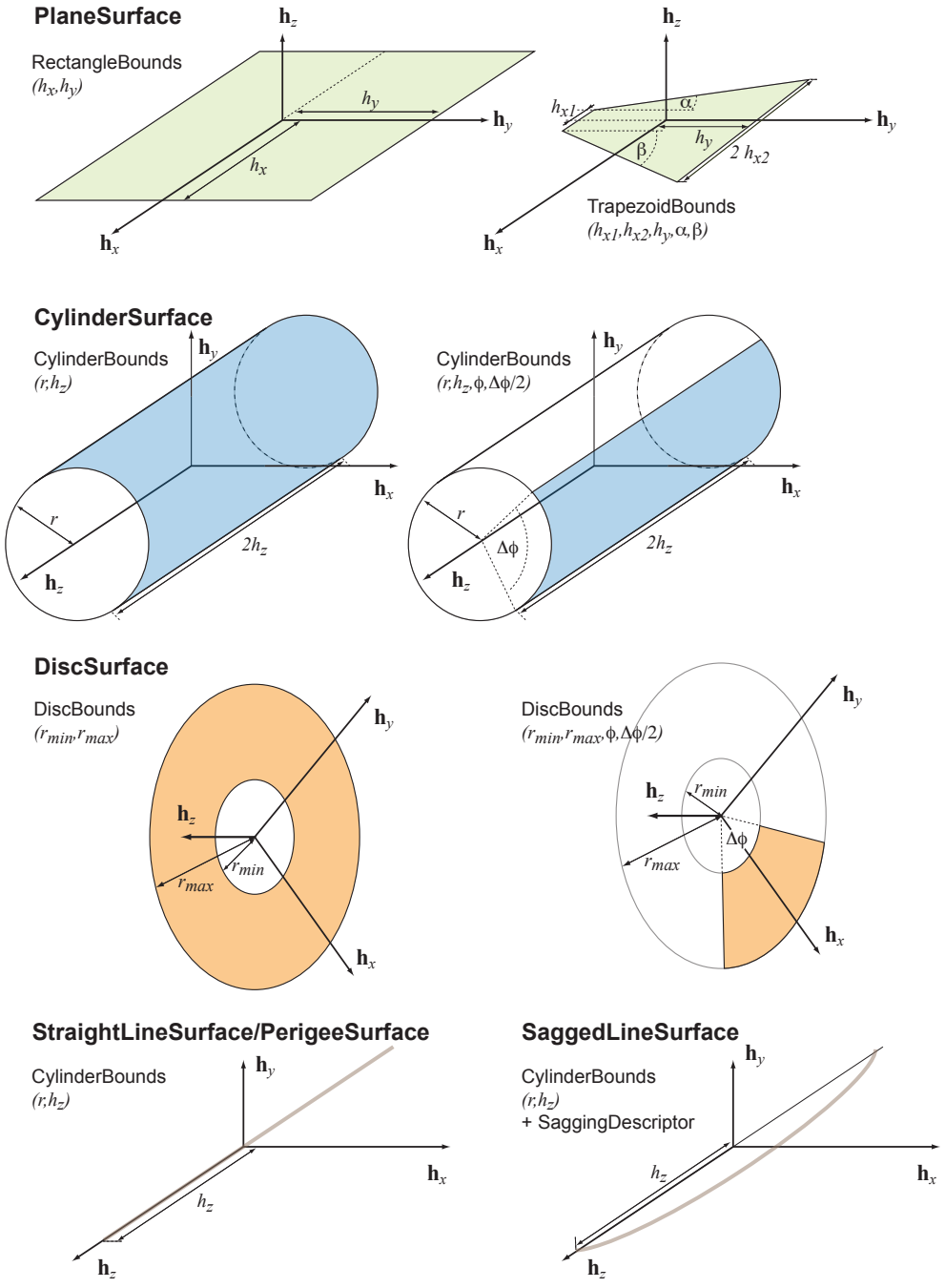


Figure 2: The different surface types and the main boundary shapes as used in the EDM and implemented in the TrkSurfaces package. In general, different surface types where chosen for different intrinsic coordinate systems. This concept is broken for performance reasons by the introduction of the `PerigeeSurface` type that is a specialisation of the `StraightLineSurface` type. The picture also includes the prototype of a `SaggedLineSurface` which has been designed to describe the sagging of wires due to gravitation, a more detailed description of this aspect can be found in Sec. 7.

The conventions remain the only restrictions that have to be met for defining additional surface and bounds classes.

`Surface` classes for volume definitions In Sec. 3 it will be shown that `Surface` objects are used to confine all further volumes in the ATLAS `TrackingGeometry`. Many different volume shapes have to be constructed, also including volumes that have been created through a boolean operation (such as subtraction) including two basic volume types. This results in the need of more complex `Surface` class definitions to handle the boolean behavior and to allow subtracted shape definitions. This additional classes, the `SubtractedPlaneSurface` and the `SubtractedCylinderSurface` still extend the common `Surface` base class, and are located — together with dedicated `SurfaceBounds` definitions — in the TrkGeometrySurfaces CVS repository[3].

Table 2: The five main surface types defined in the TrkSurfaces package, possible boundary shapes and the local coordinate frame

Surface Type	Boundary Shapes	Natural intrinsic Frame
`CylinderSurface`	cylindrical, opt. sectoral	cylinder coordinates $(r\phi, l_z)$
`DiscSurface`	disc-like, opt. sectoral, elliptical	polar coordinates (r, ϕ)
`PlaneSurface`	rectangular, trapezoidal, diamond	cartesian coordinates (l_x, l_y)
`PerigeeSurface`	pseudo-cylindrical	impact coordinates (d_0, l_z)
`StraightLineSurface`	pseudo-cylindrical	drift coordinates (r_d, l_z)

2.2 Memory Management and Object Persistency

Object persistency denotes the writing of transient EDM classes to disk, which is one of the core foundations of the ATLAS computing model. It allows reconstruction and data processing to be separated into different steps and guarantees re-processing of already recorded event data. In ATLAS, the persistency process is realised by the abstract POOL [10] interface, ROOT [11] provides the concrete implementation for input and output streaming.

Two different kinds of `Surface` types exist in terms of memory management and object persistency:

- **Public Surfaces:** surfaces that represent tracking detector elements are constructed and owned by such; The detector element (which is part of the GeoModel detector description) is consequently responsible for the deletion and freeing of the associated memory for the particular `Surface` instance during the cleanup of the reconstruction job. In this configuration the `Surface` class acts simply as a proxy for the according detector element described by GeoModel. When creating a track representation on such a surface, the `Surface` object itself is not copied, but a pointer to the existing instance is stored within the track parameters class to guarantee consistent local to global (and *vice versa*) parameter transformations. Due to the large number of surfaces that are associated to detector elements these objects are not persistified when the event information is written to POOL[4]. Only the identifier of the detector element is stored and the surfaces are recreated when track parameters are read in from a persistent data file. Storing the identifier (a small class encapsulating a bitwise encoded integer) reduces the size of the surface on disk substantially.

- **Private Surfaces:** surfaces that are not owned by detector elements are copied by object rather than by pointer. This situation can be given through `Surface` instances within the `TrackingGeometry` description, or for `Surface` objects that build reference frames for EDM objects. Most of the EDM objects are expressed with respect to public `Surface` instances, however, the EDM design allows to express tracks or track segments with respect to any given

[3]Since these type of surface classes are not part of the EDM, but of the geometry description, it has been chosen to outcast them into a dedicated location.

[4]In the ATLAS Tracking EDM the `Track` class is realised as a container class for `TrackStateOnSurface` objects. In general, only few `TrackStateOnSurface` instances on a track are not associated to sensitive detector elements.

surface. Consequently these surface instances are then owned by the holding object and are persistified together with the track expressions when the event information is written to the persistent store.

The distinction between the possible *private* or *public* behaviour of a surface can be easily done by calling the `associatedDetElement()` method: any result different from 0 indicates that the surface belongs to a detector element. In general, the public surfaces outnumber private ones by orders of magnitudes in a standard reconstruction application.

3 The Abstract Volume Description

The ATLAS `TrackingGeometry` provides an abstract volume representation; the `Volume` class concentrates differently shaped objects within a single implementation. Logical volumes or magnetic volumes inherit from the abstract volume description. The positioning and relative rotation of the volume with respect to the global frame is enhanced similarly to the `Surface` class by an instance of the CLHEP `HepTransform3D` class. A `Volume` is mainly characterised by a set of confining boundary surfaces (class `BoundarySurface`). Three `BoundarySurface` child classes extend in a double inheritance structure the corresponding mathematical `Surface` classes — which are described in Sec. 2 — and the `BoundarySurface` base class. The usage of derived classes from the TrkSurfaces package provides full compatibility of the `BoundarySurface` classes with the common tracking structure and enables boundary surfaces as input surfaces to track extrapolation processes. This full integration into the extrapolation package turns the `BoundarySurface` objects into key components of the navigation process between different volume entities: since the `BoundarySurface` class extends the `Surface` class with pointers or arrays of pointers to attached `Volume` objects, the uniquely defined normal vector of the surface can be used as an immediate indicator to the attached objects on both sides. This relation is used in the navigation process of track extrapolation between the volumes of the reconstruction geometry, see Sec. 4.

The `BoundarySurface` class itself is implemented as a class template, such that the association to an attached volume of derived type can be done without unnecessary `dynamic_cast` operations. Table 3 lists the child classes of the `BoundarySurface` base class.

Table 3: The defined child classes that extend both the common `Surface` base class for the geometrical integration and the `BoundarySurface` interface for the `Volume` confinement.

Class	Geometrical Base	Volume Shapes
`BoundaryCylinderSurface`	`CylinderSurface`	cylinder cover of tubes, full cylinder volumes
`BoundaryDiscSurface`	`DiscSurface`	faces of tubes, full cylinder volumes
`BoundaryPlaneSurface`	`PlaneSurface`	faces of boxes, sectorial faces for tubes
`BoundarySubtractedCylinderSurface`	`SubtractedCylinderSurface`	for subtracted volumes
`BoundarySubtractedPlaneSurface`	`SubtractedPlaneSurface`	for subtracted volumes

The implementations of `BoundarySurface` classes enhance the creation of basic volume shapes, which are realised through the `VolumeBounds` class. The main shapes — excluding the boolean shapes – are illustrated in Fig. 3, Fig. 4 shows the decomposition of a sample cylindrical volume in its boundary surfaces and the description of the normal vector to the boundary surfaces. The UML class diagram of the `Volume` class with its ownership relation to the possible `VolumeBounds` is displayed in Fig. 5.

3.1 Fast `BoundarySurface` Access in Navigation Processes

In the navigation process of a track extrapolation it is useful to have fast access to the `BoundarySurface` instance through which the track is expected to enter or exit a volume. A linear approximation of

the track direction is in more than 99 % of the cases sufficient for this estimation, for cases where the linear approximation fails to describe the exit surface of a volume the next probable exit surface has to be taken. The `VolumeBounds` class together with a dedicated `ObjectAccessor` class performs this fast search and provides an ordered list of `BoundarySurface` objects depending on their likeliness to be intersected by the track. In the `Navigator` — a central part of the ATLAS extrapolation package — the `BoundarySurface` objects can then be successively intersected with the use of an propagation `AlgTool` to find the entry, respectively exit surfaces of volume entities.

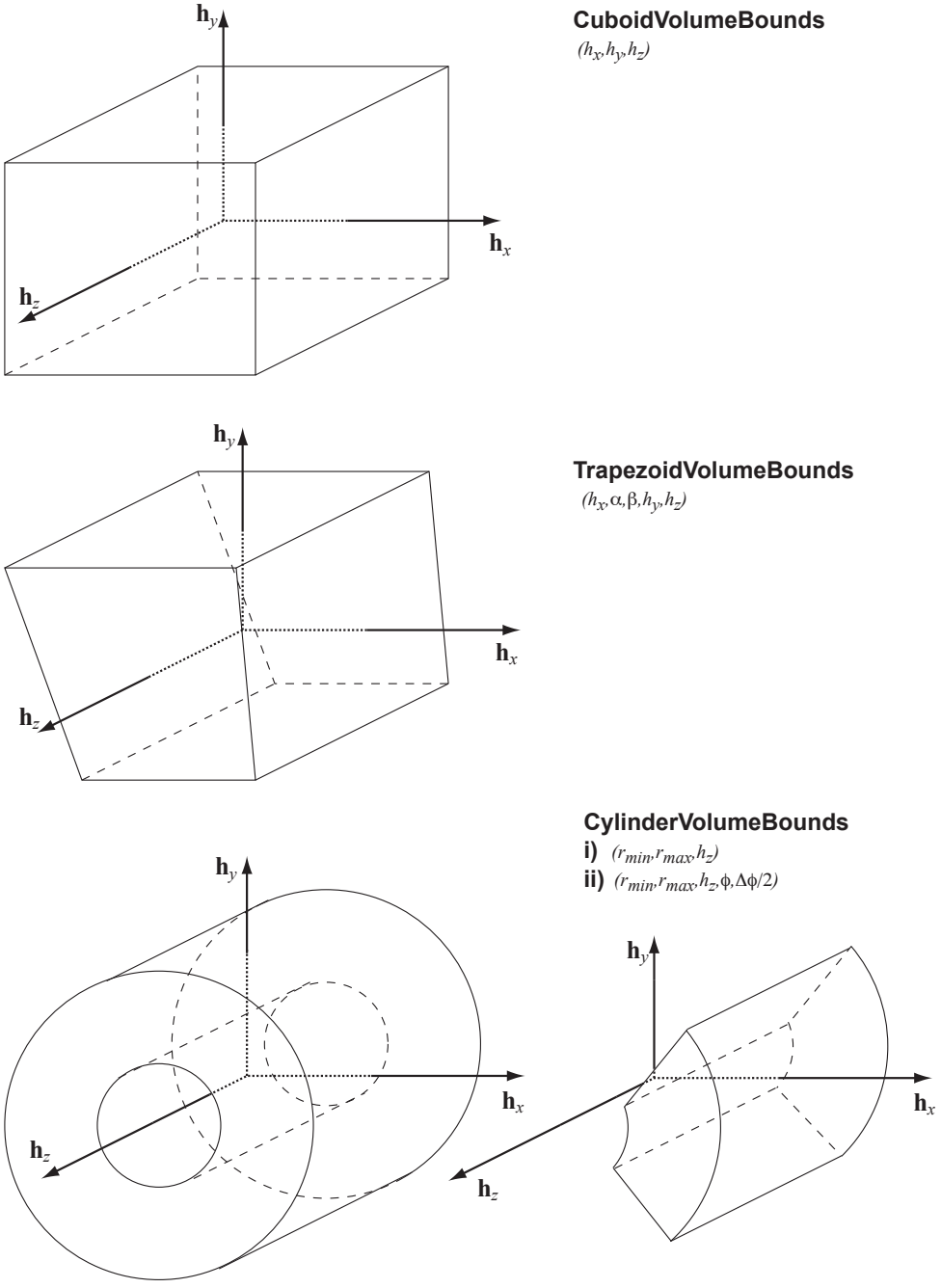


Figure 3: The three main volume types given by the different volume boundary classes. Volumes are mainly characterised by the set of confining boundary surfaces that are handled and owned by the `VolumeBounds` class. More complex volume shapes such as e.g. the bevelled cylinder shape can also be found in the TrkVolumes repository, they are mainly used for the construction of the complex `TrackingGeometry` of the Muon System.

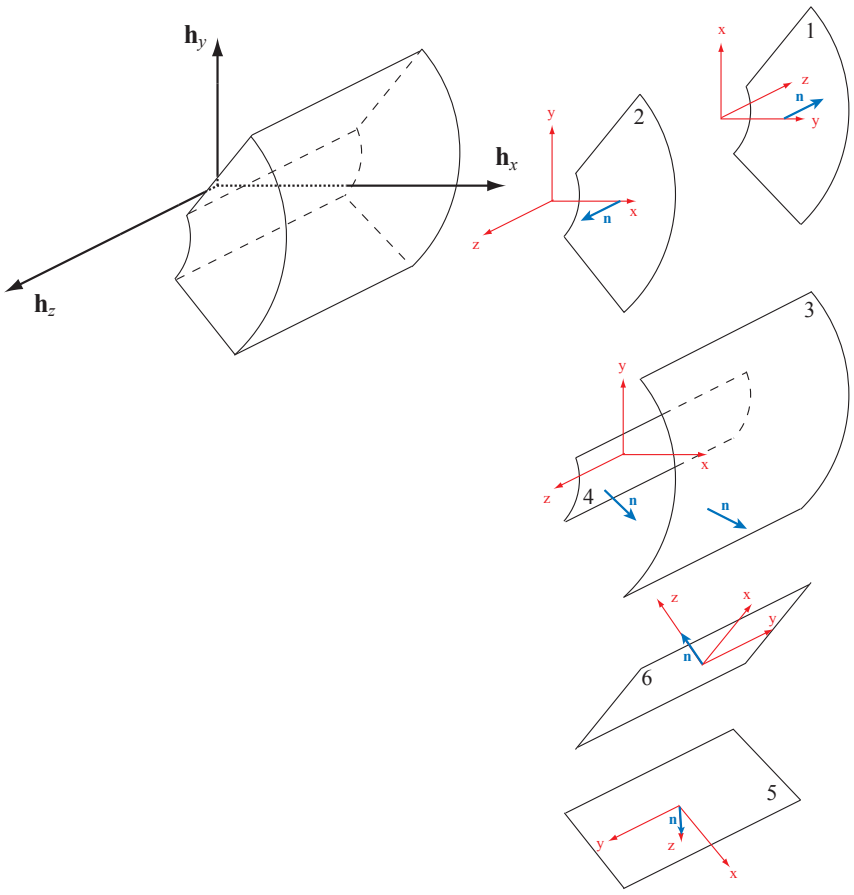


Figure 4: A (sectorial) cylindrical volume shape and its decomposition into boundary surfaces; the uniquely defined normal vectors of the boundary surfaces enhance the pointing to the inside respectively outside neighbor and serve as the key for the navigation between volumes.

4 The Geometry Description

Surfaces and volumes are geometrical entities, but can not handle the description of physical or logical objects in the tracking detector model. Relevant information about the material is missing for the use in track reconstruction. In addition, the reference to the magnetic field through the volume is of advantage when using a parameterised field description for a decrease of CPU time consumption in algorithms with repeating magnetic field look-ups[5].

4.1 The `Layer` Class

A `Layer` class combines the surface information together with material description to be mainly used in track fitting. It is located in the TrkGeometry package and provides additionally the possibility to order sub-surfaces representing sensitive detector elements onto the layer surface. This schema, together with the newly designed extrapolation engine results in a powerful geometry that enhances a predictive navigation to be used e.g. for *a posteri* holes-on-track search and the new *Fast Atlas Track Simulation* (FATRAS) [12].

The derived layer classes `PlaneLayer`, `DiscLayer` and `CylinderLayer` follow a double inheritance structure from the corresponding `Surface` classes and a common `Layer` base class, Fig. 6 shows a simplified UML diagram for a planar layer.

Similarly to the boundary surfaces, the extension of the common `Surface` base class enhances `Layer`

[5]For most pattern recognition algorithms the simplified parametric field description is sufficient for the required tracking quality during track finding.

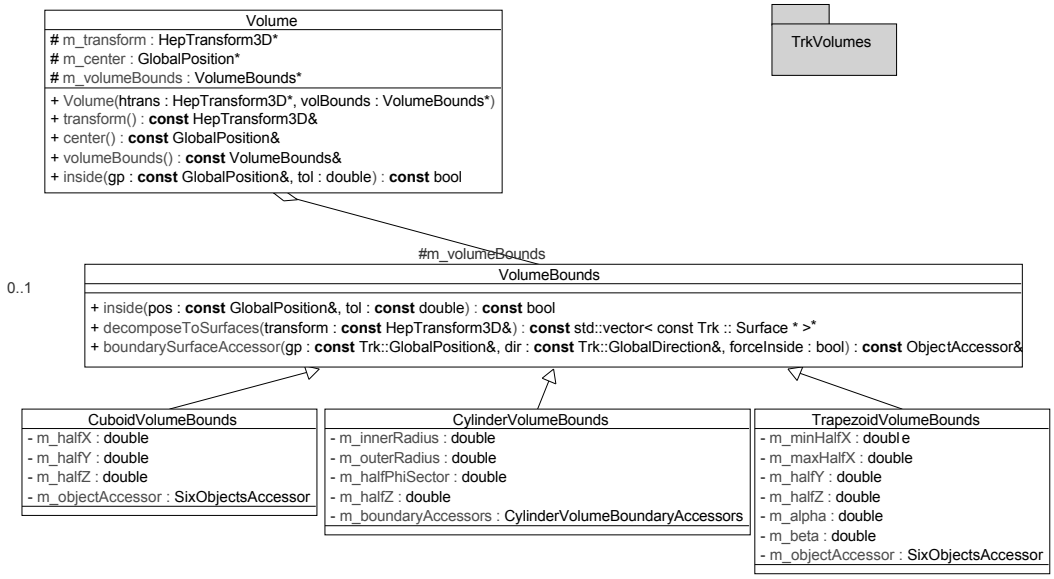


Figure 5: Simplified UML of the Volume class in the ATLAS `TrackingGeometry`. A `Volume` is mainly characterised by its confining boundary surfaces. The contained `BoundarySurface` class and its extensions is also shown in this diagram.

objects to be naturally used within the extrapolation package.

In a static setup, the `Layer` class includes the possibility to point to the next or previous `Layer` object with respect to the enclosing volume frame for inter-layer navigation. A dedicated `NavigationLayer` class that does not represent any material has been created to respect material-free regions in the simplified detector model while preserving the unique layer association of every point in the detector. This is necessary to enhance fast navigation between layers in the reconstruction process. Figure 7 shows an illustration of an example volume with a contained static configuration of navigation and material layers.

4.1.1 The `LayerMaterialProperties` Class

Material on a layer is described by classes extending a `LayerMaterialProperties` base class. Two different child classes, the simple `HomogenousLayerMaterial` that describes a constant material distribution and the more complex `BinnedLayerMaterial` extend this base class. The `BinnedLayerMaterial` enhances material description with a binned structure, both one- and two-dimensional. The same binning technique as for the ordering of sub volumes within the `TrackingGeometry` hierarchy is used, realised by dedicated `BinUtility` classes. A more detailed description of these and further utility classes used from TrkDetDescr package can be found in the Appendix, see Sec. A.3.

The material distribution on a layer can be chosen with different update directives. In some subdetectors the layer based material update may be done at different stages in the extrapolation process: some of the layers require the material update to be done before respectively after the measurement is taken into account during the track fit, simply for the more accurate representation of the actual spatial placement of the material in the detector. The concept of applying the material update into two steps, one part before including the track measurement on track (henceforth called *pre-update*), one part after inclusion of the hit on the consecutive extrapolation to the next hit module (*post-update*) is realised by the interface design of the layer material description. For layers traversed during track extrapolation without a hit, a full update is done taking the entire layer material into account[6].

[6]These three concepts of pre-update, post-update and full update are also reflected in the interface of the `IMaterialEffectsUpdator` class in the new track extrapolation package.

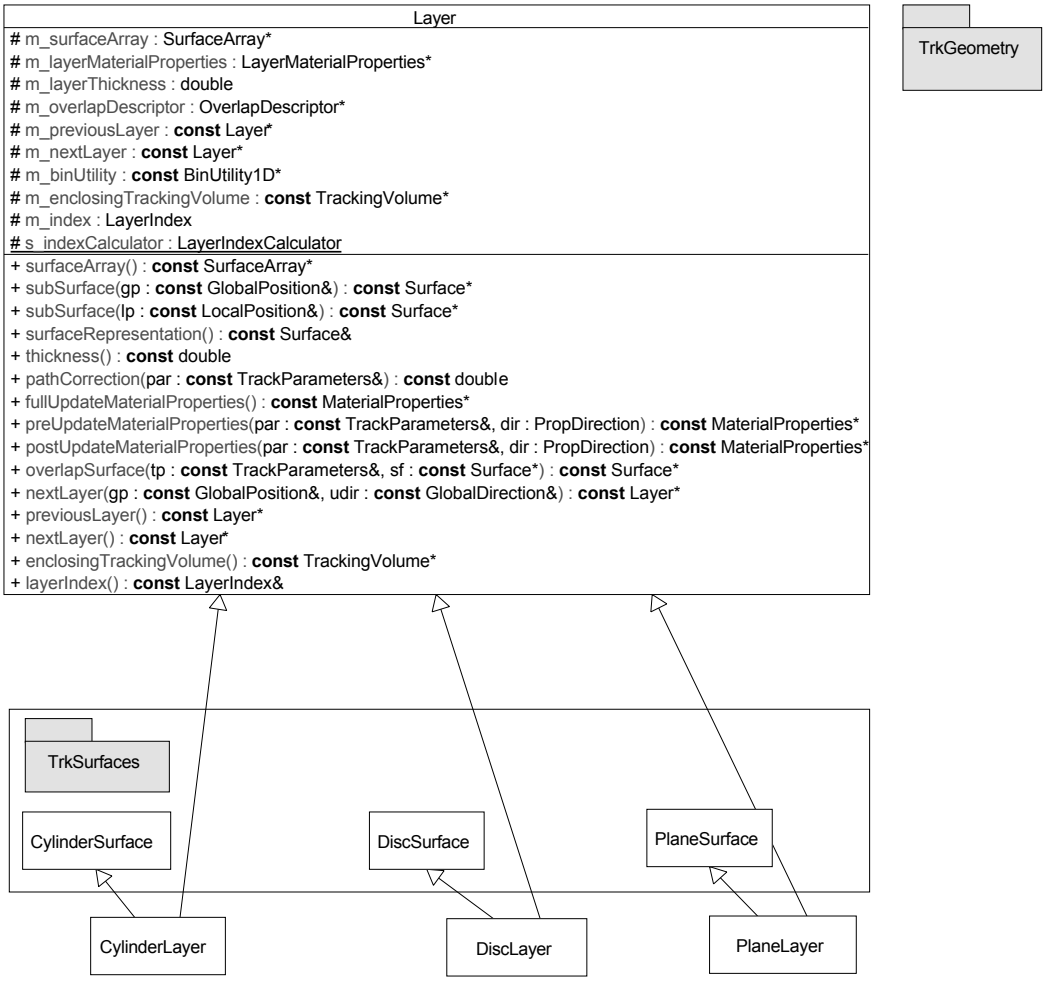


Figure 6: Simplified UML diagram for the layer class in the TrkGeometry package. The double inheritence of a layer from its dedicated surface representation and a common base class is shown. For convenience constructors and destructors are ommitted in this diagram.

Reference Material The average description of detector material using layers with binned material, however, can not guarantee the accuracy needed for various benchmark measurements aimed for with the ATLAS detector. Therefore, a dedicated `ReferenceMaterial` class has been added to the `LayerMaterialProperties` that can concentrate a very accurate description of a particular subset of the layer. The application of this method in the context of the Pixel and SCT detector is described further detail in Sec. 5.

4.2 The `TrackingVolume` Class and `DetachedTrackingVolume` Class

In the same way the `Layer` class extends the `Surface` base class, the `Volume` class is extended to represent logical entities in the reconstruction geometry. The derived classes, i.e. the `TrackingVolume` and the `DetachedTrackingVolume` classes, extend both, a material and a magnetic field property class. This design has been motivated by the following requirements:

- Deriving the `TrackingVolume` and `DetachedTrackingVolume` from the `MaterialProperties` class enables the modeling of detector volumes with dense material such that the newly developed STEP propagation algorithm [13], which takes material effects during the propagation continuously into account can be used.
- The inheritance from the magnetic field property object turns the `TrackingVolume` into the

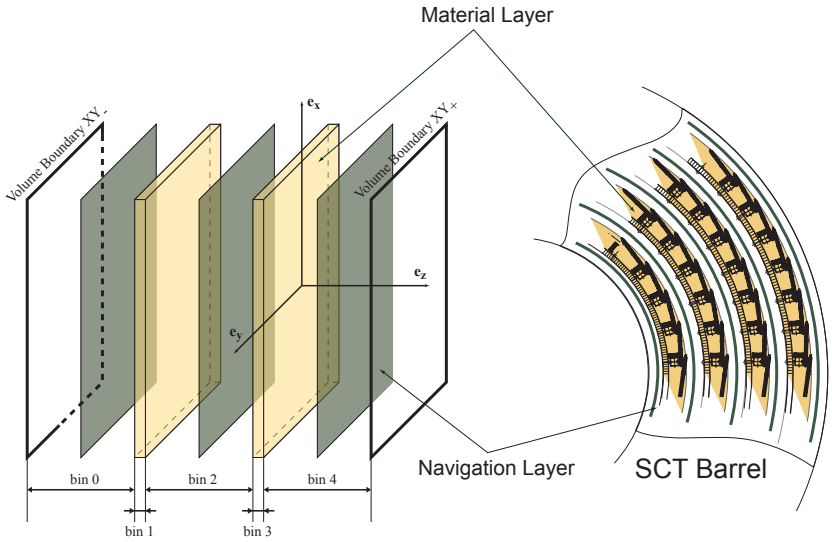


Figure 7: Illustration of an example volume (left) containing three material layers (bright) and four navigation layers (dark). The navigation layers are place holders with the single purpose to point to the next respectively previous material layers. The second illustrations shows the realisation of this strategy for the SCT Barrel layers.

interface to the magnetic field access during the propagation and helps therefore to synchronise magnetic volumes with detector entities.

The existence of two separate classes, the `TrackingVolume` and the `DetachedTrackingVolume` is due to their different role during the navigation process. `TrackingVolume` objects require a fully connected setup, i.e. the `BoundarySurface` objects are shared between volumes respectively point to the attached volumes to guide the navigation from one volume to the other. Figure 8 shows the simplified UML inheritance tree for the `TrackingVolume` class; the integration of the `TrackingVolume` class into the `TrackingGeometry` and its part in the inter-volume navigation is described in Sec. 4.3. The `DetachedTrackingVolume` is designed to be used in a non-connective setup which results in a different navigation strategy in track extrapolation processes: the entry and exit surface of a track to such a volume is not *a priori* known and can not easily be found by a linear track approximation. It has to be found by multiple propagations to detect the nearest intersection with the various `BoundarySurface` objects. The `DetachedTrackingVolume` can be regarded as a movable *floating* body that is loosely (or not) connected to the surrounding environment. It is constructed, in general, from a prototype `TrackingVolume` and contains methods for cloning and repositioning to facilitate the construction of multiple objects. The sub-structure of the `DetachedTrackingVolume` is fully connective and the navigation can fall back to the standard navigation mechanism used with `TrackingVolume` instances.

4.2.1 Confined Layers and Volumes

A `TrackingVolume` can contain a subset[7] of confined layers (illustrated in Fig. 7) or a subset of confined volumes, respectively. Whereas the first point enables the positioning and navigation of layers within a detector volume, the second concept creates a hierarchy structure for the entire `TrackingGeometry`. Evidently one `TrackingVolume` instance must not contain both types of subsets at once, which is regulated by the available constructors that are defined for the `TrackingVolume` objects. Thus, two categories of `TrackingVolume` instances exist in the reconstruction geometry description:

- volumes on **container level** that contain other `TrackingVolume` instances;
- volumes on **navigation level** which contain `Layer` objects and are at the lowest brach of the volume hierarchy;

[7]The subsets are not simple STL container objects, but dedicated templated container classes that enable a binned ordering of objects in an array structure, see Sec. A.3.

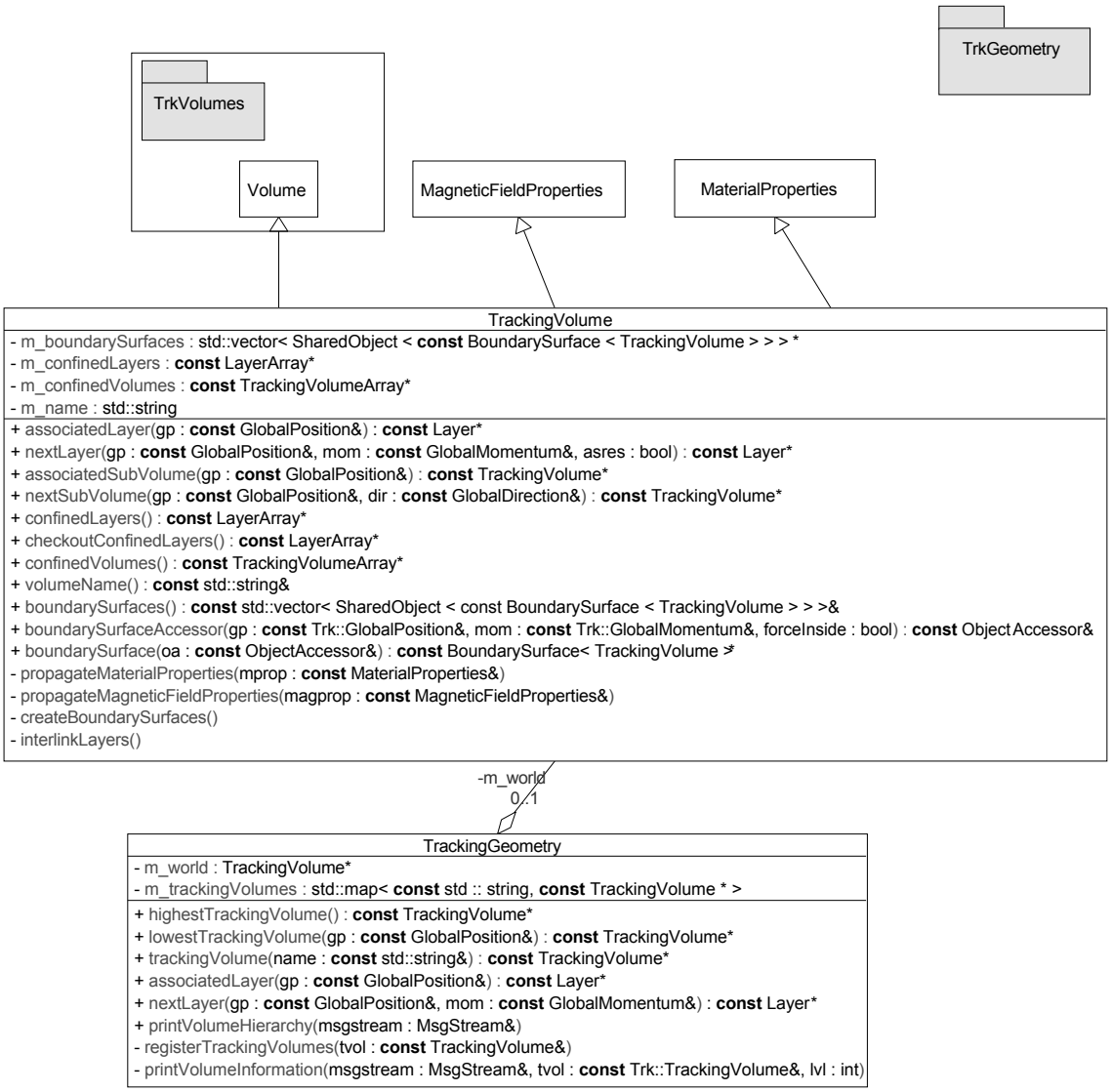


Figure 8: Simplified UML inheritance diagram for the `TrackingVolume` class in the new TrkGeometry package. The triple inheritance from the abstract volume class, such as the classes representing material respectively magnetic field properties. For convenience constructors and destructors are omitted in this illustration.

All navigation actions taken inside the `TrackingGeometry` are done on **navigation level**, hence volumes on **container level** do not need to be included in the navigation, material and magnetic field schema. The confined layers of a `TrackingVolume` on **navigation level** can exist in three ways: an ordered static setup, an unordered static setup and a dynamic setup. The ordered static layer setup (as illustrated in Fig. 7) is characterised through an alternating sequence of `NavigationLayer` objects and layers with material description, ordered along the main traversing direction of tracks within the `TrackingVolume`. The `NavigationLayer` fills artificially the entire space between two `Layer` objects, such that every point in the volume can be uniquely assigned to either a navigation object or to a `Layer` itself. The static setup is for instance realised in the standard Inner Detector `TrackingGeometry` setup, see Sec. 5.1. In case that the volume contains a set of layers which cannot be suitably ordered in one or two dimensions, the concept of association of every space point to a `Layer` is abandoned (in general, it would require filling of the *empty* parts of a volume with complex spacing objects). In this case, the `TrackingVolume` or `DetachedTrackingVolume` can hold an unordered vector

of `Layer` objects and the navigation has to be performed similarely to the `DetachedTrackingVolume`: by searching the next layer intersection through propagation.

The alternative layer description in a `TrackingVolume` is a dynamic setup when `Layer` objects are created on demand when being needed for material information during the track extrapolation. This is done by registering an `IDynamicLayerCreator` to the `TrackingVolume` at creation time. The `IDynamicLayerCreator` interface allows various concrete implementations with different sources for the material description. When traversing a `TrackingVolume` with a dynamic layer setup, the request for the traversed material is forwarded to the registered concrete implementation of the `IDynamicLayerCreator AlgTool`. Given the global position for the start point and the end point (destination surface impact or exit of the volume at a boundary surface), the material is collected by the `AlgTool` and modeled in a number of n new layers in between. Hence, the client algorithm is responsible for memory cleanup associated to the dynamic `Layer` objects.

4.3 The `TrackingGeometry` Class

The `TrackingGeometry` class is the top class in the geometry hierarchy. It holds the `TrackingVolume` of highest hierarchical order which represents the full detector geometry through its internal hierarchy tree. This allows the stepping down to the lowest volume (i.e. the volume on navigation level) for any given point in the ATLAS detector. All `TrackingVolume` objects of the geometry are registered with their name to the `TrackingGeometry` to allow the search of specific volumes e.g. for the creation of reference expressions at well defined entry or exit points of sub-detector volumes. The `TrackingVolume` class inherits the entire templated `BoundarySurface` schema from the `Volume` base class and builds therefore a fully connected set. Full connectivity in this sense should indicate that each single volume on navigation level — except the outermost boundary volumes of the entire detector description — is fully attached to neighbor volumes at all sides via the `BoundarySurface` mechanism. Evidently gaps between volumes have to be filled with dedicated gap volumes at navigation level. The mechanism of glueing volumes together requires that `BoundarySurface` instances can be shared between `Volume` objects, i.e. in software terns that the `BoundarySurface` object becomes a member of both `Volume` objects. Therefore `BoundarySurface` objects must be reference counted to guarantee a clean memory management, while optimising memory consumption by not creating unnecessary copies of identical objects. The `SharedObject` class is in charge for the reference counting mechanism and is located in the TrkDetDescrUtils package, which is in further detail described in Sec. A.3.

Recursive Property Setting The hierarchy tree structure allows the modification of the entire `TrackingGeometry` after the building process, e.g. a total scaling of all material objects by a global factor or the redefinition of the used magnetic field configuration. As a concrete example a parameterised field description can be applied to all volumes of lower hierarchy level than the Inner Detector if the accuracy needed for the specific application allows to simplify the details field description. An example of the hierarchical tree structure of the `TrackingGeometry` volumes is illustrated in Fig. 20.

4.4 Connectivity and Navigation through the `TrackingGeometry`

The hierarchy schema enables a unique association of any global position with a `TrackingVolume` at navigation level. This is achieved by stepping down the hierarchy tree to the lowest branch associated with the given space point. Decisions between `Volume` objects on the same level are achieved by the binned ordering of the fully connective `TrackingVolume` objects within a container. However, global search methods are CPU time consuming and since most of the navigation sequences within track extrapolation processes follow a trajectory through various volume using the connectivity of the `TrackingGeometry` serves as a faster navigation model. Since the full understanding of the navigation requires a deep knowledge about the structure of the ATLAS track extrapolation engine, the reader is encouraged to find a further description of the ATLAS extrapolation package in [14]. Figure 9 illustrates a sample navigation sequence following a trajectory through a toy detector.

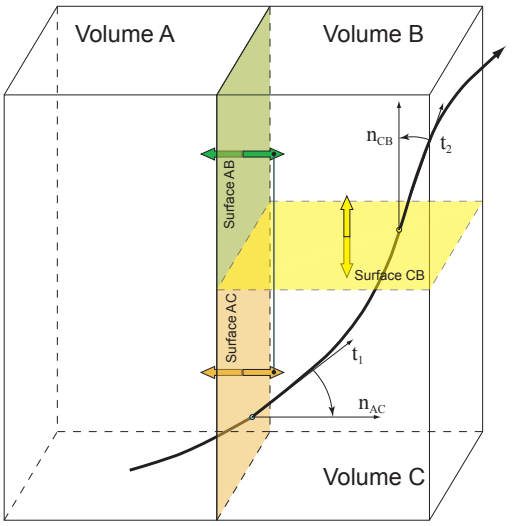


Figure 9: Illustration of a sample navigation following a particle trajectory through three fully connected volumes A, C and B. The volume boundary surfaces hold information about the attached volumes respectively volume arrays, such that a simple projection onto the surface normal vector enhances a step by step navigation scheme.

5 The ATLAS `TrackingGeometry`

The building of the ATLAS `TrackingGeometry` requires the parsing of the full ATLAS detector description GeoModel. Specific information about detector structures has to be accessed for this taks, therefore the sub-detector constructions introduce dependencies on these associated software repositories. The new ATLAS tracking realm, on the other hand, has been designed to be subsystem independent and the same concept is also applied to the content of the TrkDetDescr container package. To integrate the `TrackingGeometry` into this software structure, the building of the sub-detector `TrackingGeometry` instances has been outsourced into the associated detector repositories, while still using only common classes from the Tracking repository that do not refer to specific detector technologies. Various different concrete implementations of an `IGeometryBuilder` interface class, each for one sub-detector or for different detector setups, are retrieved at run-time and are steered by a central `AlgTool`, the so-called `GeometryBuilder` (located in the TrkDetDescrTools package). Details of the building process and the structure of the sub-detector geometries are described in the following sections.

5.1 The Inner Detector `TrackingGeometry`

The reconstruction geometry for the Inner Detector is created by the `InDetTrackingGeometry-Builder AlgTool` that makes use of several other `AlgTool` classes for the creation of `Layer` and `Volume` objects that are contained by the ID. The building process is evoked by calling the `PixelLayerBuilder` and the `SCT_LayerBuilder`, respectively, that parse the associated sensitive GeoModel detector description source for the pixel detector and the silicon strip detector (SCT). The overall dimensions of the silicon detector are determined and `Layer` objects created, while the sensitive detector elements are sorted in binned arrays with a fast access mechanism, Figure 10 illustrates this simplified model for a SCT endcap disk. The ID `TrackingGeometry` automatically adapts to different layouts[8] and misalignment configurations. The `Layer` objects for the *Transition Radiation Tracking* (TRT) are, in general, not built by parsing the sensitive detector elements. This is due to the fact that the material in the TRT is almost continously distributed and can be — for performance reasons — simplified to a few layers in the reconstruction geometry. Modeled layers for condensed material information are inserted in the corresponding TRT volumes to represent the inert material of the TRT detector[9].

A UML sequence diagram for the creation of the Inner Detector `TrackingGeometry` in shown in

[8] E.g. the number of pixel barrel and endcap layers differs for the ATLAS-Rome-Initial and ATLAS-CSC layouts. The `PixelLayerBuilder` adopts through parsing of the full description to the actual number of layers, and the `PixelVolumeBuilder` encloses the layers dynamically.

[9] For the use in FATRAS, the TRT straws are grouped and ordered on layers similarly to the pixel or SCT detector. As this operation requires the parsing of about 300 000 elements this is omitted for the standard reconstruction job setup.

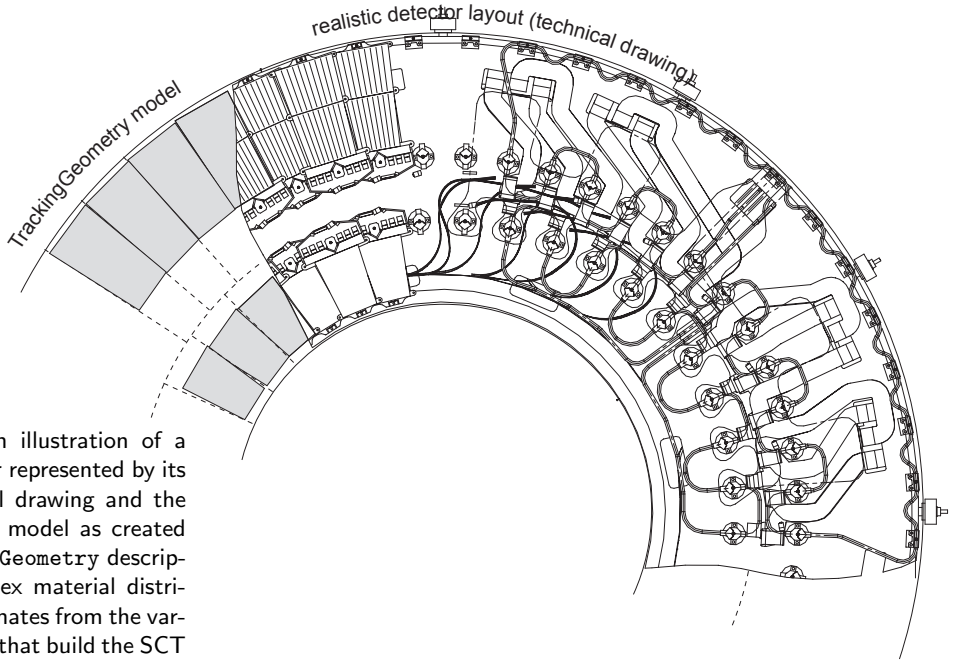


Figure 10: An illustration of a SCT endcap layer represented by its detailed technical drawing and the simplified binned model as created for the `TrackingGeometry` description. The complex material distributions that originates from the various components that build the SCT endcap disk is described by a material map, see Sec. 5.4.1.

Fig. 11. The InDetTrackingGeometry package also contains dedicated `TrackingGeometry` builders for the combined testbeam and commissioning setups. Since these builders extend the same intefarce classes as the geometry builders for the full geometry setup, they can be automatically taken instead at runtime, when job configuration triggers these run configuration.

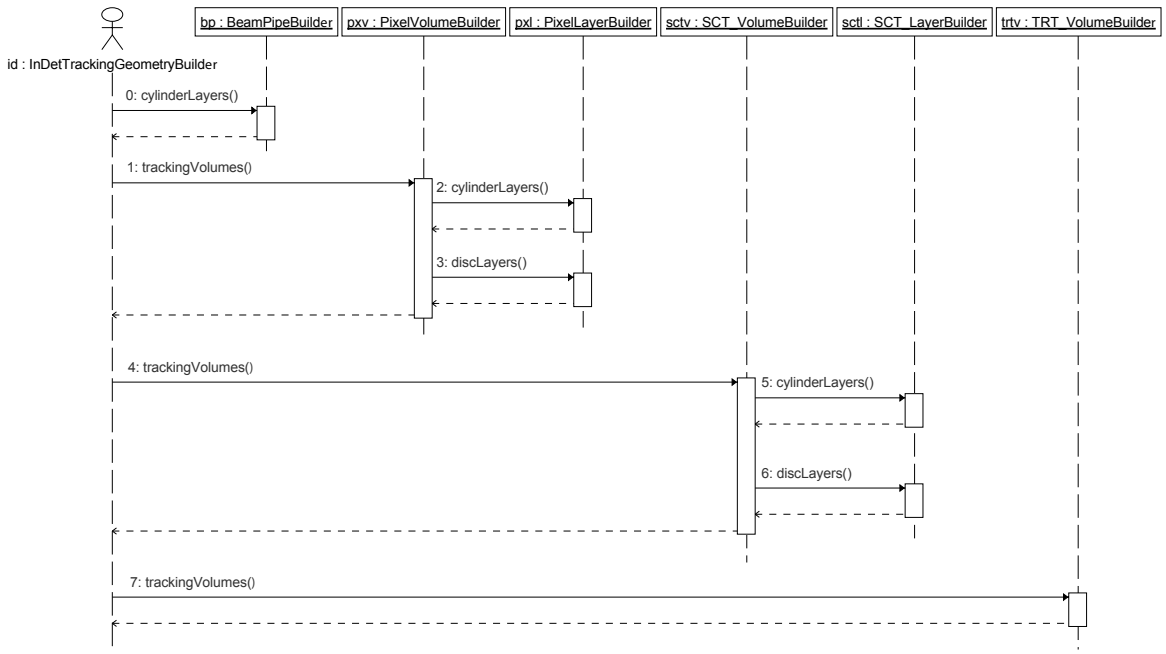


Figure 11: UML Sequence diagram for the building process of the Inner Detector `TrackingGeometry`. Volume builders for the Pixel, SCT and TRT detectors are used, in the standard configuration pixel and SCT volume builders themselves use implementations of `LayerBuilder` classes.

5.1.1 Overlap Description in the Inner Detector `TrackingGeometry`

In the pixel and SCT detector, neighboring modules in both directions (but mainly in the azimuthal direction) overlap to maximise the coverage of the detecting surface. An example of the module overlap for the SCT barrel in the ϕ direction can be seen in Fig. 7. The resulting increase of material seen by a track crossing through such an overlap region is already taken care of with the `ReferenceMaterial` mechanism, described in Sec. 4. For a qualitative *holes-on-track* search and for the use of the `TrackingGeometry` in the fast track simulation this overlap information is a very important feature to estimate the number of pixel and SCT hits per track correctly. In the `TrackingGeometry` a special overlap description schema has been created to handle these overlap situations in a fast but precise way. Dedicated helper classes that extend a common `OverlapDescriptor` base class have been implemented for the various sub-detector technologies, which hold information about the neighboring modules and provide them in an optimised way to the extrapolation package. The `OverlapDescriptor` can be registered to the `Layer` object and retrieved from it when needed in the navigation process. The overlap description has been extended to be able to handle any neighboring modules, e.g. the backside module of a SCT module is handled as an overlap module as well as the next straw in azimuthal direction on a idealised TRT cylinder layer[10].

5.2 The Calorimeter `TrackingGeometry`

The Calorimeter `TrackingGeometry` is built by the `CaloGeometryBuilder` located in the CaloTrackingGeometry CVS package. It is capable of building various versions of the calorimeter description:

- A static configuration very close to the actual GeoModel description can be built, including volume builders for both, the *Liquid Argon* (LAr) and *Tile* calorimeter. The `CaloGeometryBuilder` steers the building process and delegates the creation of the `TrackingVolume` objects to the `LArVolumeBuilder` and `TileVolumeBuilder`, respectively, that are located in the sub-detector repositories. Dedicated converter classes, located in the TrkDetDescrGeoModelCnv translate the logical top volumes of the GeoModel description into according `TrackingVolume` objects, and `Layer` instances are inserted to quantify the material. Continuous (i.e. `TrackingVolume` based) and point-like (i.e. `Layer` based) material update mechanisms are both enabled by the static geometry description. The positions of the inserted layers are retrieved from the `CaloSurfaceBuilder AlgTool` that is also used in the `TrackToCalo` reconstruction algorithm; this guarantees an optimal description of the calorimeter material when being used for extrapolations between sensitive sampling layers.
- A very simplified description of the calorimeter volume can be chosen alternatively, that is linked to a `IDynamicLayerCreator`. `Layer` objects are hereby created on demand from an external source.

The `CaloTrackingGeometryBuilder` adapts automatically to a given inner volume that is wrapped to keep the connectivity within the `TrackingGeometry`, Fig. 12 shows an illustration of the combined Inner Detector and Calorimeter `TrackingGeometry` in the full static configuration, the `Layer` objects are omitted in this illustration.

5.3 The Muon `TrackingGeometry`

The `TrackingGeometry` for the Muon System is build by four `AlgTool` implementations that are located in the MuonTrackingGeometry package:

- the `MuonTrackingGeometryBuilder` ist a concrete implentation of the `IGeometryBuilder` interface building the combined ATLAS geometry (combining the Muon `TrackingGeometry` with ID/Calorimeter `TrackingGeometry`, or, for Muon stand-alone setup, with a dummy central volume).

[10] The TRT barrel detector is build in a 32 folded geometry that is for speed reasons in the TRT `TrackingGeometry` simplified as cylindrical layers. This simplification introduce a small error in the $r\phi$ coordinate of the straw centers when being registered in the binned arrays. This leads to a rare wrong prediction of the following track intersection that is cancelled by the `TRT_OverlapDescriptor`.

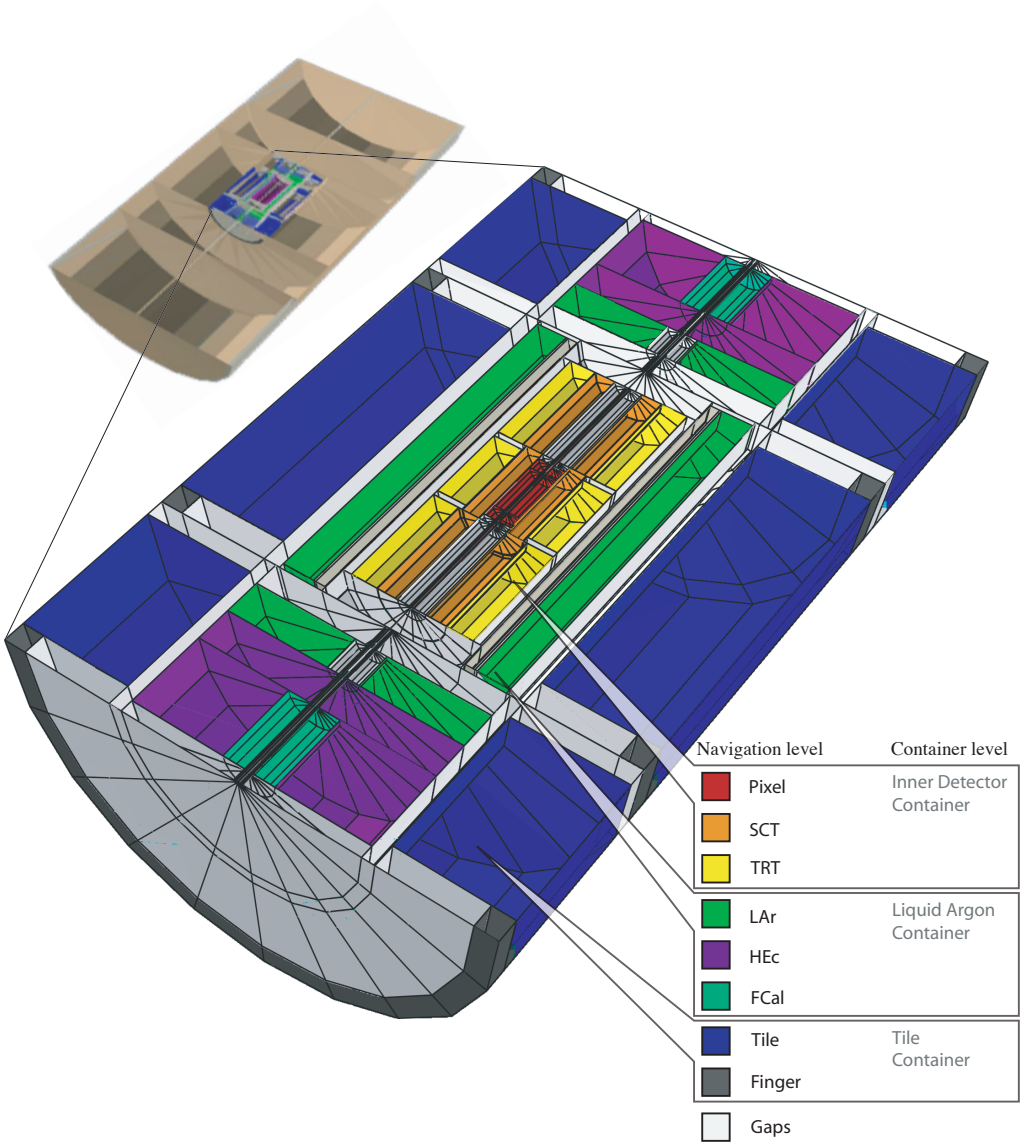


Figure 12: The joined `TrackingGeometry` for the Inner Detector and the Calorimeter in the static configuration. The `TrackingVolume` objects of the Inner Detector and the Calorimeter are embedded in the global ATLAS `TrackingGeometry`: the illustration shows also the enclosing static envelope `TrackingVolume` structure of the Muon System `TrackingGeometry`. The visualization is done using the `TrackingVolumeDisplayer AlgTool` based on the ROOT graphics extension. The layers are omitted in this illustration.

- a dedicated `MuonStationBuilder` for building of the active geometry (muon stations); also provides identification of active layers.
- a `MuonStationTypeBuilder` class for processing of station prototypes, and
- a `MuonInertMaterialBuilder` implementation for building of the passive material of the reconstruction geometry.

The strategy deployed throughout the Muon `TrackingGeometry` is as follows:

- all material objects are represented by `DetachedTrackingVolume` objects (and are therefore, in principle, alignable, although this feature is superfluous for inert material)
- the active material is concentrated on `Layer` objects with point-like update

- the active layers are constructed in a way that allows a unique (geometrical) association of a GeoModel `DetectorElement` with a layer, therefore opening the possibility to use layer surface representation as an alternative tracking surface
- the passive material (outside stations) is described by dense volumes; this description relies on use of `STEP_Propagator` for extrapolation through these volumes.

All Muon Spectrometer objects are confined in a system of (fully connected) static volumes (*muon envelope*) which form (with the central detectors) the ATLAS combined tracking geometry. The detached volumes can span over several static volumes within the muon envelope, i.e. a given `DetachedTrackingVolume` can be simultaneously confined in several container volumes. Both enveloping `TrackingVolume` and confined `DetachedTrackingVolume` objects are defined at navigation level. Figure 13 shows parts of the Muon System `TrackingGeometry` build in stand-alone setup.

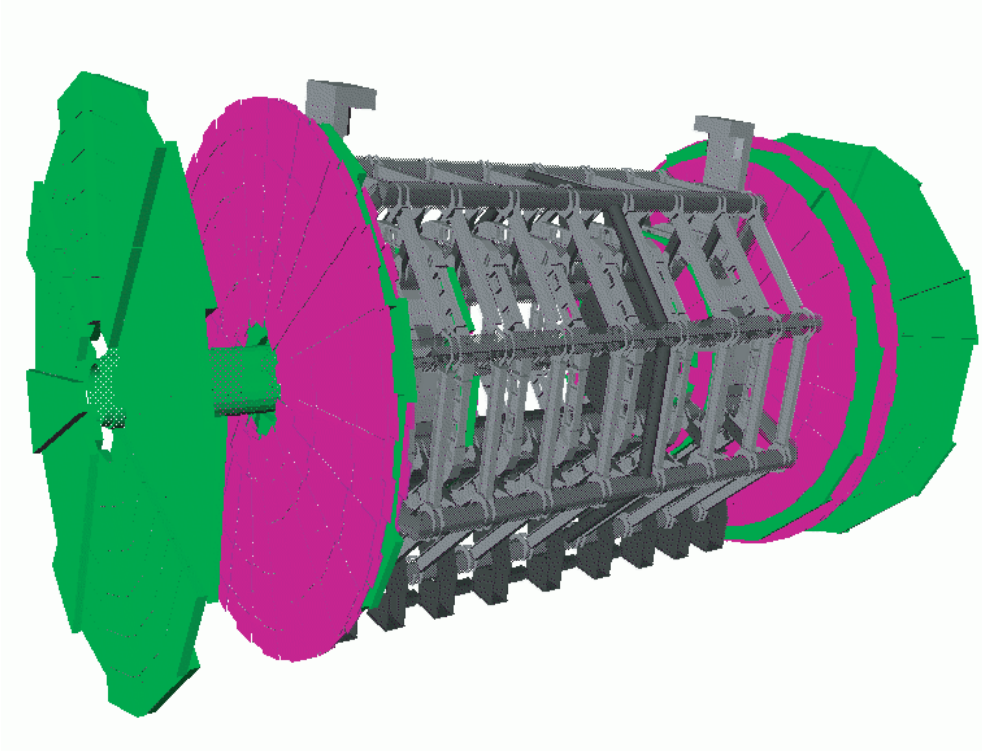


Figure 13: Parts of the Muon System `TrackingGeometry` visualised with the ATLAS HepVis event display, the Barrel stations are omitted in this illustration. The most complex version of the Muon System `TrackingGeometry` is generated by parsing and translating the GeoModel detector description.

5.4 Combined `TrackingGeometry` Building and Material Association

The building of the `TrackingGeometry` in the ATLAS case is organised in an inside-out structure: starting from the ID, the calorimeter volumes are build and finally the Muon System is wrapped around, where the inner enclosed volumes are integrated into the outer `TrackingVolume` objects. This guarantees that the navigation schema of the static `TrackingVolume` classes is carried out to the outermost volumes.

The steering of this consecutive geometry building is done by a general ATLAS `GeometryBuilder AlgTool`, which is interfaced to the job steering by python configuration classes. As the wrapping process of the inner volumes adapts automatically to the size of the inserted volume respectively is able to fill gaps with dedicated fill volumes that carry out the navigation stream, sub-detectors can be optionally left out in the construction procedure: i.e. all sub-detector `TrackingGeometry` parts can be constructed in a stand alone mode or single sub-detector parts can be left out.

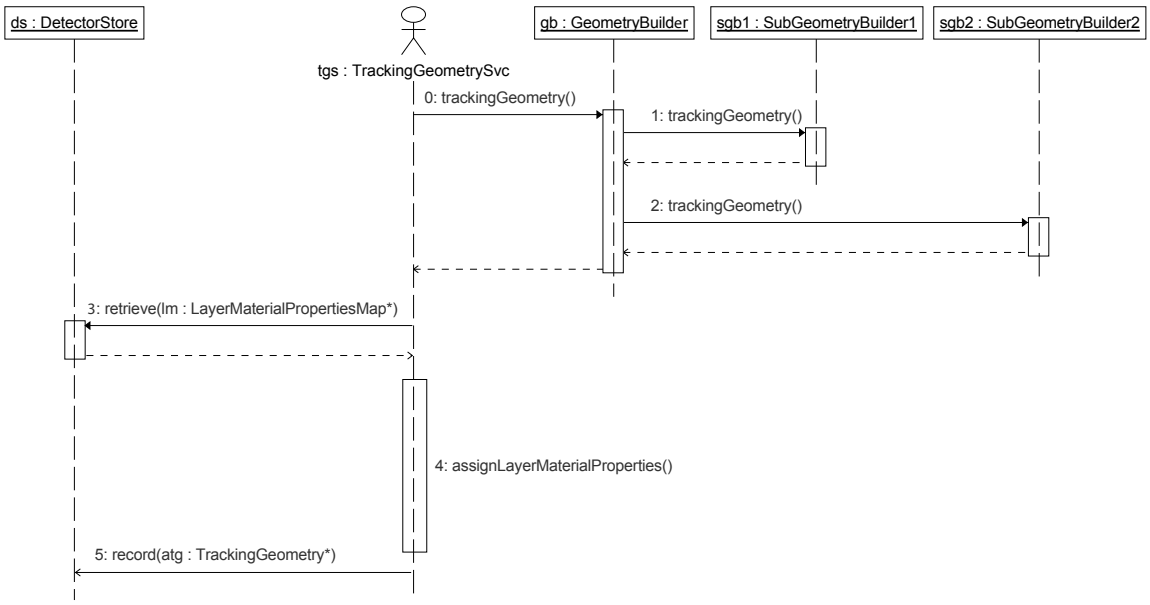


Figure 14: UML Sequence diagram for the building process of the `TrackingGeometry`. Two different implementations of the `IGeometryBuilder` interface are illustrated in the picture, steered by the general GeometryBuilder AlgTool that is controlled by the TrackingGeometrySvc.

An Athena `Service` class, the `TrackingGeometrySvc` located in the TrkDetDescrSvc package, executes and steers the building process of the ATLAS `TrackingGeometry`. Depending on the job specifications, the sub-detector geometry builders are configured and the main `GeometryBuilder` is called to start the construction of the `TrackingGeometry`. The geometry building process is realised as a callback function, i.e. the geometry version tag (see Sec. 1) is determined with the data header of the first event, which triggers a callback function to the `TrackingGeometrySvc`. This is particularly important for correct retrieval of the material description file, which is described in more detail in the following section. Additionally the callback mechanism enables an update of the reconstruction geometry for alignment, and even the running on multiple files with different layouts in one single reconstruction job.

5.4.1 Automatic Material Association and Calibration

Material maps: The material association for layers is done — for the static layer setup in the ID and calorimeter — by mapping the material described in the full simulation geometry onto the simplified `TrackingGeometry`. A direct translation of the detector material from the geometry database would be in principle possible, but would still require some underlying tracking engine to associate the detailed detector parts to layers or layer cells: the simplification of the complex geometry to few basic geometrical shapes (described by mathematical surfaces) requires that the real positions of detector material is projected onto the surface. In the `TrackingGeometry` this is done by using a straight line propagation from the extrapolation package, while using a constraint that the track origin is the nominal interaction point. This generalisation can be done, since the projection distance is very small in comparison to the geometrical structure or the track bending. The mapping process requires as an input a detailed map of the simulation geometry, which is produced by a dedicated Geant4 `UserAction`[11]. The spatial information (position, step length) and the material properties (radiation

[11] The Geant4 toolkit allows the user to interact with the simulation process through the dedicated `UserAction` mechanism. Geant4 also provides a non-interacting particle type, the so called geantino that is used for the mapping process in this scope.

length X_0, density ρ, atomic number A and mass index Z) of the Geant4 simulation steps — when tracking a non-interacting particle — are translated into a persistent class and written to a ROOT file. Figure 15 shows as an example the hit position map of the recorded Geant4 simulation in comparison with the simplified `TrackingGeometry` model.

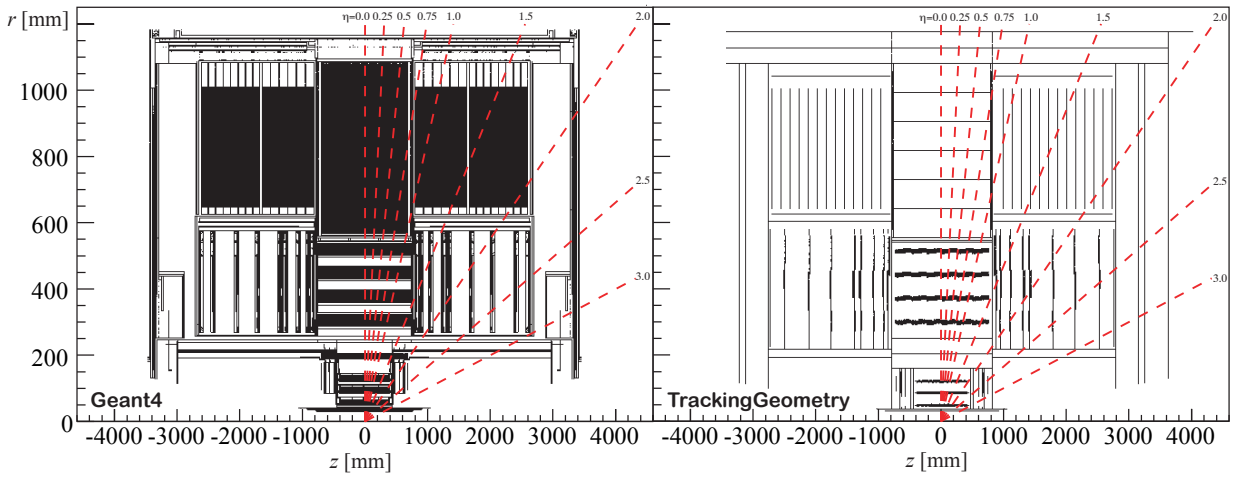


Figure 15: The material hit map of the Inner Detector in the rz plane created with the ATLAS Geant4 simulation and its simplified representation by the `TrackingGeometry` after the material mapping process. `Layer` objects and the boundaries of the `TrackingVolume` objects can be identified in the `TrackingGeometry`. Additionally, the layers with incorporated `ReferenceMaterial` (pixel and SCT Detectors) representations can be clearly identified through their diverse structure.

Creation and Retrieval of Material Files from COOL: a dedicated `Algorithm` that is located in the TrkDetDescrAlgs, the so-called `MaterialStepAssociation`, reads the produced file from Geant4 and maps the material steps onto the `TrackingGeometry` layers or layer bins (if a binned material description is has been chosen). This is done by using the internal navigation of the `TrackingGeometry` together with the new track extrapolation package. The areas around the mathematical infinitesimal thin layers (layers extend the `Surface` class) are automatically determined using a pointing direction to the nominal interaction point and the material cell is then projected onto the intersection point with the `Layer` object. An illustration of this schema including the pointing direction constraint can be seen in Fig. 16. The material steps are corrected for their incident angle to the associated layer and, given the high statistics of the Geant4 sample, an average description of the material associated to every particular `Layer` can be obtained.

The output of this `Algorithm` is translated into `LayerMaterialProperties` objects as described in Sec. 4.1.1 and written into a ROOT file. For each given layout, this process is repeated and the resulting ROOT file, which is typically of a size of about 250 kByte on disk, is registered with the according layout tag in the ATLAS condition database (COOL)[15]. When starting a reconstruction job, the appropriate data set according to the chosen detector layout is in reverse retrieved from COOL and the material is loaded onto the `TrackingGeometry`.

Material Calibration: During the start up of the ATLAS experiment, different techniques will be required to calibrate the material in both, the simulation and reconstruction geometry. The TrkDetDescr realm has been designed to provide a great variety of calibration methods for the reconstruction material. All descriptive material classes have scaling methods implemented to tune the material with given calibration data. Together with the powerful extrapolation package and the internal navigation of the reconstruction geometry, material calibration within reconstruction algorithms can be deployed.

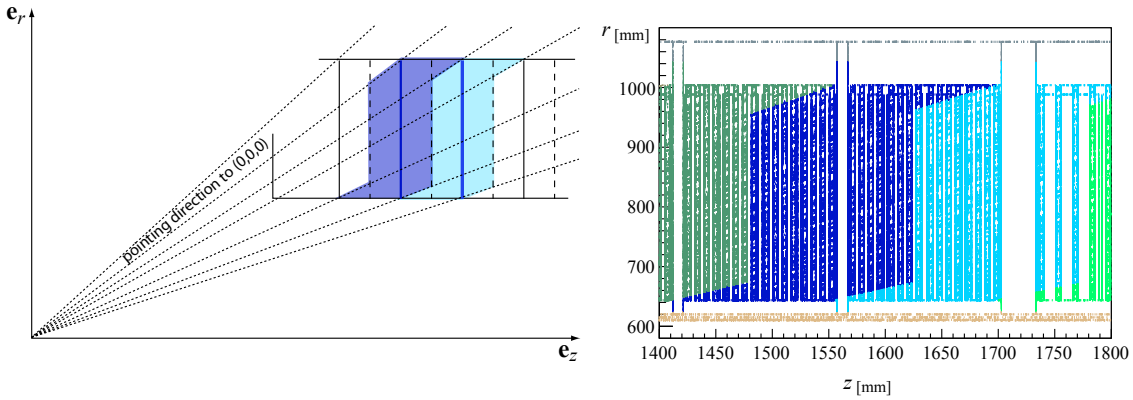


Figure 16: The Geant4 step association with pointing direction using the nominal interaction point as a pointing constraint is shown in the left illustration. The second plot shows the actual Geant4 hit map in the rz plane with the associations to different layers identified by different colors for a part of the positive TRT endcap.

6 Validation of the ATLAS `TrackingGeometry`

Any complex software framework should be designed keeping an automatic or at least semi-automatic validation procedure in mind. This is in particular important for basic component packages that create manifold dependencies to components of higher complexity or diversity. The geometry description in track reconstruction is surely one of these basic components since it impacts directly the functionality and — more important — the quality and performance of the track reconstruction. The validation process can be, in general, divided into these two different realms:

- **quality:** the main task of the `TrackingGeometry` is providing an adequate material description for the track extrapolation or track fitting operations. Thus, the natural evaluation of the quality is the performance of the overall track reconstruction, in particular the pull distributions of the track parameters representations after track fitting. However, the final tracking quality depends on multiple parameters and the factorisation of different contributions is almost impossible. The `TrackingGeometry` provides therefore a complete framework to validate the material description in comparison to the full Geant4 description.
- **functionality:** the validation of the building process and the correct setup of the constructed `TrackingGeometry` for different layouts, checks for holes in the connectivity of volumes and between layers covers the functionality of the `TrackingGeometry`.

6.1 Comparison between Simulation and Reconstruction Geometry

The intrinsic navigation of the ATLAS `TrackingGeometry` allows a very detailed validation of the material budget in comparison with the one represented by the simulation geometry. This process starts already during the mapping of the Geant4 information onto the `TrackingGeometry` (described in Sec. 5.4.1): the simulation material is recorded for further processing through a dedicated `MaterialMapper AlgTool` that identifies the `TrackingVolume` and/or `Layer` the Geant4 step is associated to and fills a dedicated output container.

The same `MaterialMapper AlgTool` can be used as a plugin to the new ATLAS extrapolation engine: in detail, it becomes part of a special flavored `AlgTool` that implements the `IMaterialEffectsUpdator` interface and — instead of updating the track representation according to the amount of traversed material — records the material through the `MaterialMapper`[12]. A simple validation algorithm, the so-called `TrackingGeometryValidation`, located in the TrkDetDescrAlgs performs the mapping of

[12] The power of the component pattern architecture of the ATLAS Tracking software can be seen with this example remarkably well

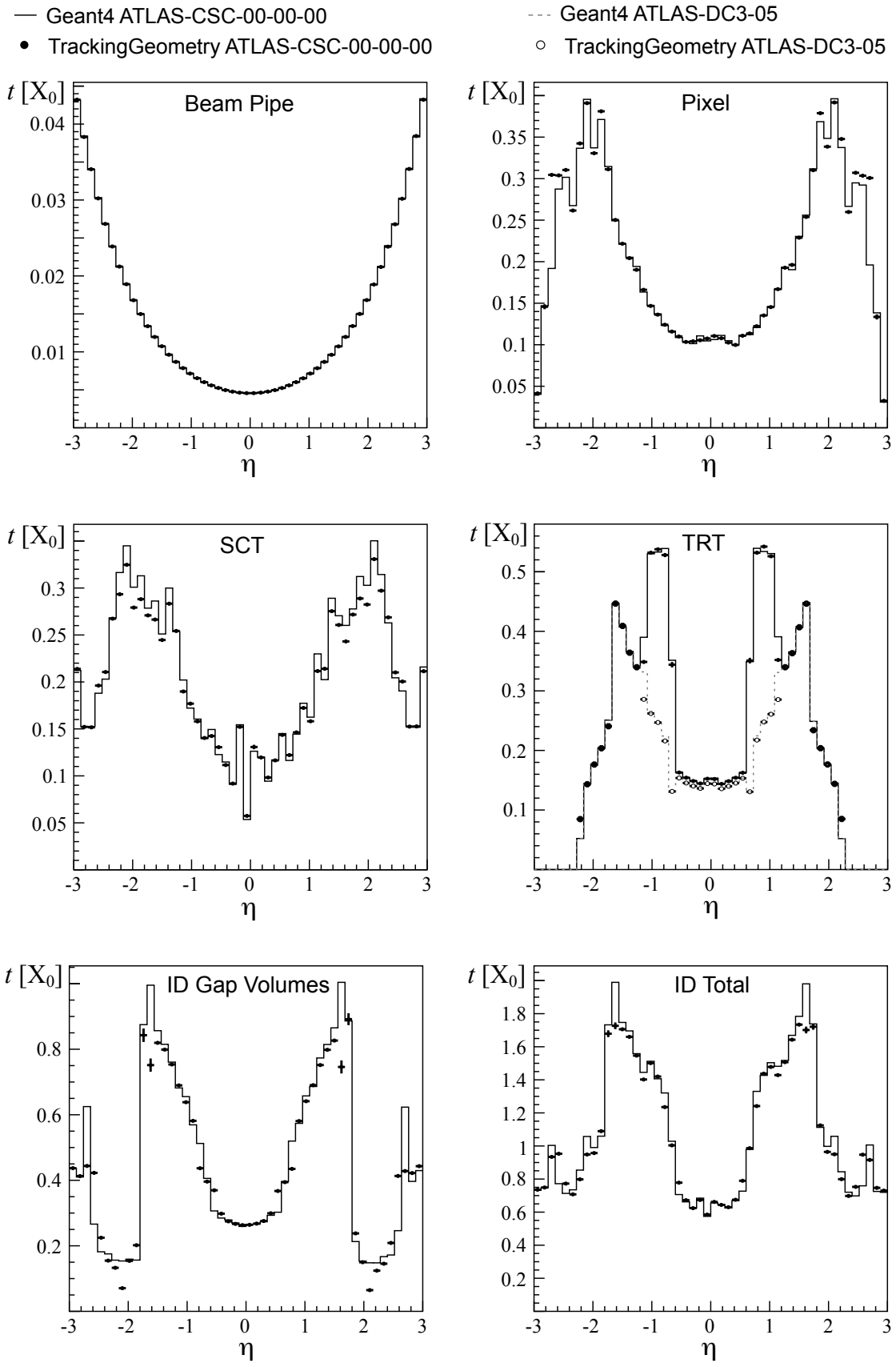


Figure 17: The material distribution in terms of radiation length thickness for the volumes of the Inner Detector `TrackingGeometry` in comparison with the material used in full simulation (Geant4). The material of the `TrackingGeometry` layers is shown. The TRT comparison also incorporates the difference between the ATLAS-CSC-00-00-00 and ATLAS-DC3-05 layout, respectively, and shows the automatic adaption of the reconstruction material description to the simulation geometry.

the reconstruction material after the associated material file is read in from COOL. The identical structure of the output ROOT trees eases the comparison between the Geant4 and reconstruction geometry; dedicated ROOT based comparison applications that are adopted to the output structure of the `MaterialMapper` are located in the TrkDetDescrExample package. This automatic comparison is only possible for a fully static `TrackingGeometry` description. Hence, it can not be applied in this mode for the prototype `TrackingGeometry` of the Muon System. Figure 17 and Fig. 19 show comparisons of the material budget in terms of radiation length for the Inner Detector and Calorimeter in the ATLAS-CSC-00-00-00 layout. Fig. 18 shows the same comparison for the `ReferenceMaterial` of the Pixel B layer.

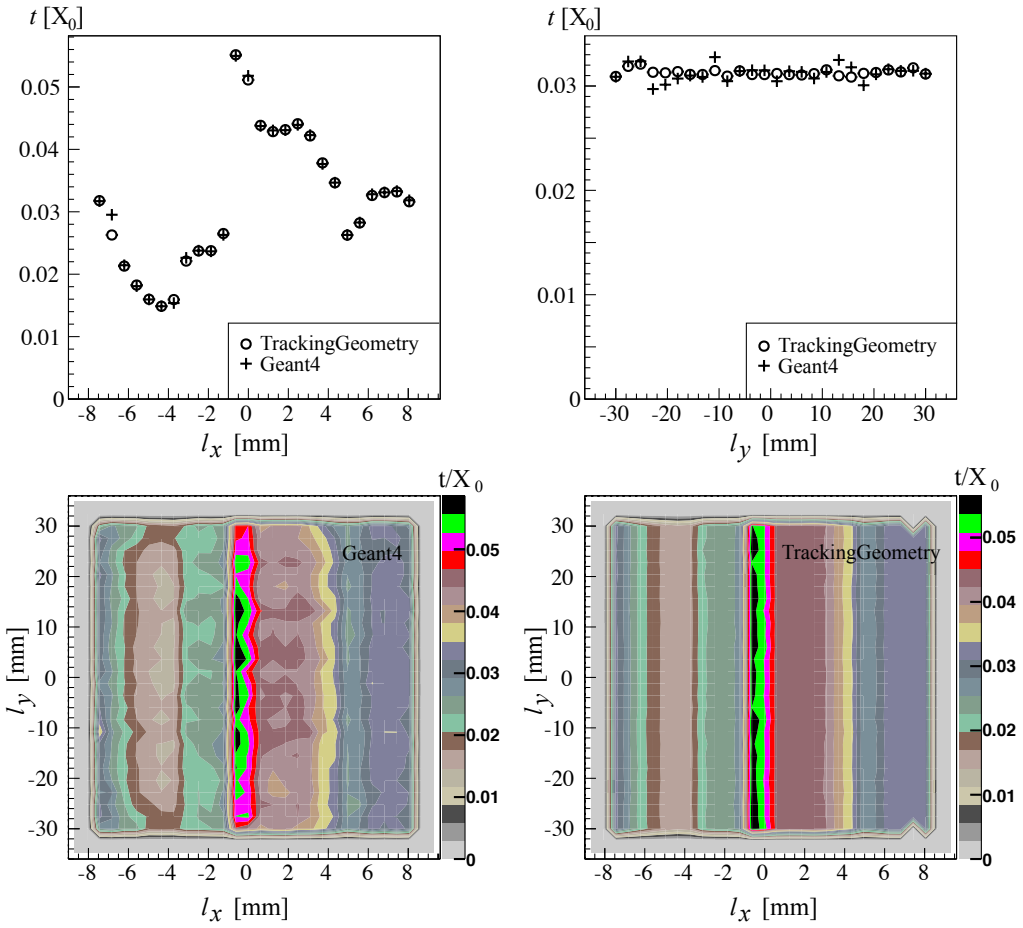


Figure 18: The material distribution in terms of radiation length thickness of a sample Pixel B layer module in the comparison between full simulation (Geant4) and reconstruction (`TrackingGeometry`). Refined structures can be in the `TrackingGeometry` separately described by the so-called `ReferenceMaterial` mechanism. The first plots shows the `ReferenceMaterial` description in a direct comparison of the material thickness in units of radiation length along the local x and local y coordinate of the surface, respectively. The latter two plots show two-dimensional scatter plots of the material distribution within the senstive area.

6.2 Automatic Testing

The testing of the material association in comparison with the simulation material still needs human intervention by checking the validity of the produced output. The geometry building process, however, can be done in an automatic way. A dedicated algorithm `GeometryBuilderTest`, located in the TrkDetDescrAlgs repository, can be executed for this purpose.

Since the `TrackingGeometry` is built, in general, by parsing the full ATLAS detector description, it

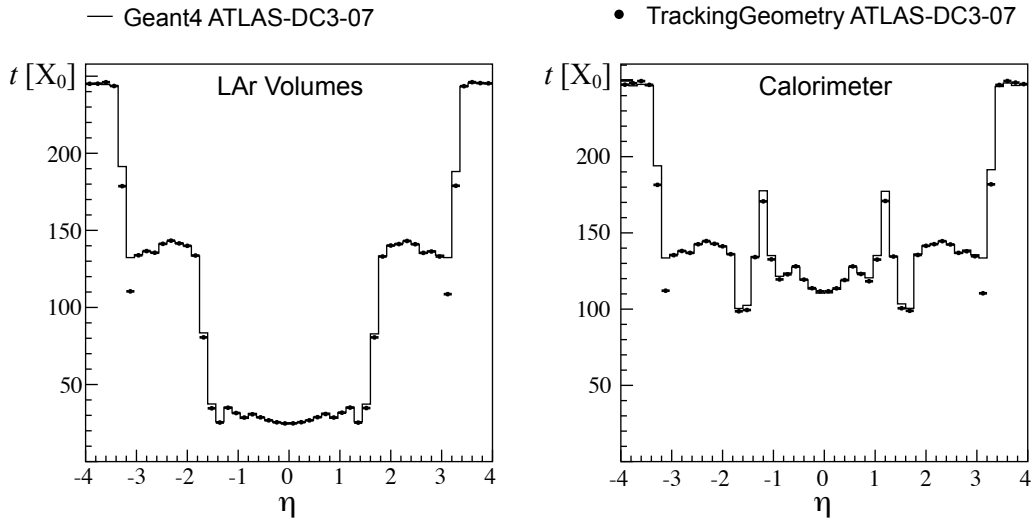


Figure 19: Comparison of the material budget in terms of radiation lengths for the Liquid Argon volumes (including forward calorimeter, FCAL, and the hadronic endcap calorimeters, HEC) and the entire Calorimeter between simulation (Geant4) and reconstruction (`TrackingGeometry`) geometry.

is important to monitor the output of the parsing process instead of directly comparing the numbers from detector description database. The `GeometryBuilderTest` algorithm evokes the building process and parses the `TrackingVolume` tree. For each `TrackingVolume` the main information about positioning, bounds, confinement and navigation level is written into a HTML file. Simultaneously, a Javascript based file incorporating the hierarchy tree structure is created and interlinked with the various files describing the `TrackingVolume` objects of the parsed `TrackingGeometry`. Any standard HTML Browser can be then used as a `TrackingGeometry` **Tree Browser**. The concept of generating a purely standalone and easy to use browser has been chosen to guarantee that the `TrackingGeometry` nodes can be parsed with local software installation kits or even after local code manipulations by users during an eventual development phase. Figure 20 shows a screen snapshot of the `TrackingGeometry` Tree Browser.

6.3 Visualization of the ATLAS `TrackingGeometry`

Two different visualisation methods for `TrackingGeometry` have been enhanced, mainly for debugging possibilities. A dedicated controller for the ATLAS 3D event display, HepVis [16], that is based on the Open Inventor technology [17] has been created. HepVis serves as a fully functional and powerful event display. It allows to show objects of the `TrackingGeometry` in the context of actual event data, e.g. the track intersection with a given `Surface` object can be illustrated, Fig. 13 is an example for an illustration done with the HepVis package. Additionally `Track` objects can be displayed in such a way together with the `TrackingGeometry` objects.

The TrkDetDescrTools also contain a dedicated `TrackingVolumeDisplayer` `AlgTool` capable of writing a ROOT output file that can be interpreted by the ROOT OpenGL viewer. Figure 12 is an example picture produced by the `TrackingVolumeDisplayer`. The motivation for yet another visualisation technique in parallel to HepVis has been driven by very similar needs that lead to the development of the `TrackingGeometry` Tree Browser: the `TrackingVolumeDisplayer` does not need any other precondition to be operated than an existing `TrackingGeometry` instance in the detector store.

7 Extensions to the TrkDetDescr Realm

Two extensions of the TrkDetDescr have been introduced recently to handle specific cases of alignment and calibration issues. They exist only in prototype versions and are not yet part of a production

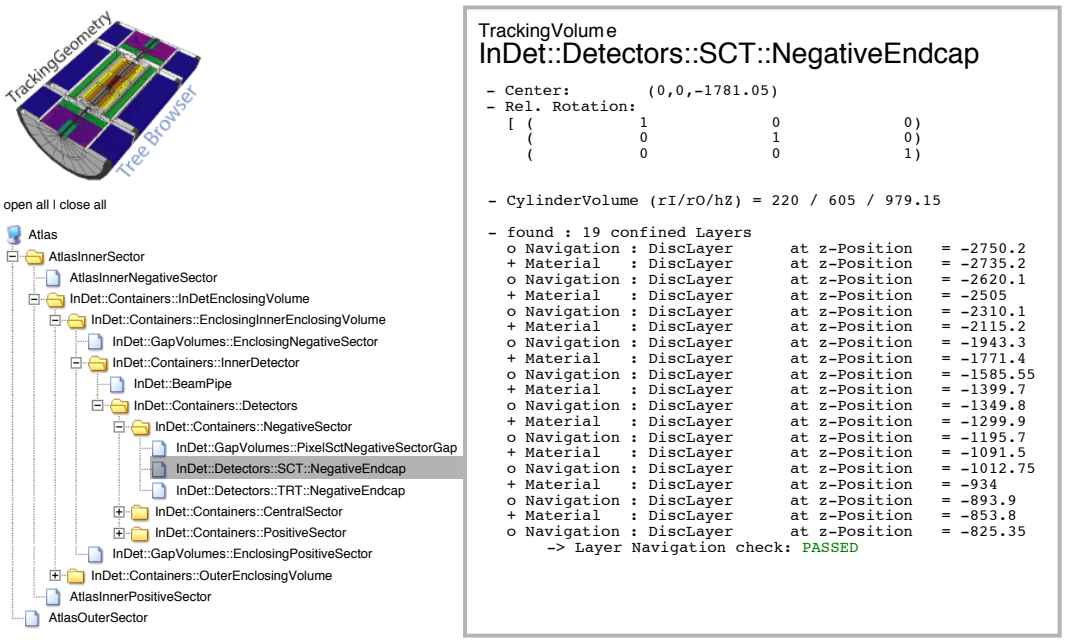


Figure 20: The `TrackingGeometry` Tree Browser for a sample Inner Detector stand-alone `TrackingGeometry` objects. Both `TrackingVolume` objects on **navigation** and **container** level can be parsed, informations about the `TrackingVolume` and confined objects are displayed in the right frame.

release.[13] The presented extensions to the `Surface` class illustrate how a coherent software design that builds on clear interfaces definitions and a polymorphic data structure eases the integration of new tasks into an already existing frame. In both cases this could be realised with minimal intervention to existing code by gaining substantial additional functionality. For convenience and completeness these extensions are described in the following.

7.1 Distorted `Surface` Objects

The `Surface` objects in the ATLAS TrkDetDescr realm are realised as ideal geometrical objects. In reality, the detector suffers from distortions to the ideal geometry due to various effects: gravitational sagging, mounting on frames and construction uncertainties. In track reconstruction, some of these effects can be dealt with in a parametric description, but in particular for the large structures of the ATLAS Muon System geometrical distortions may be taken into account directly through a more complex geometrical modeling. This lead to the development of a distorted surface schema, which has been fully integrated into the existing geometry structure.

The distorted surfaces are located in the TrkDistortedSurfaces CVS package that provides the base class for all distorted `Surface` objects. A concrete implementation of a `DistortedPlaneSurface` would then extend the `PlaneSurface` class and the `DistortedSurface` base class. The main client of the `DistortedSurface` objects is the extrapolation package that switches to a different strategy once a distorted surface is provided: a first intersection with the nominal or parent surface is performed and a more realistic — but still geometrically ideal — surface is provided depending on the local position of the intersection. The second extrapolation to the dynamically provided `Surface` object is then performed to optimise the track prediction.

[13] Neither the TrkAlignableSurfaces nor the TrkDistortedSurfaces are part of the ATLAS offline release 12.0.6, which is the base release for this document. The described extensions have been introduced in the ATLAS development release 12.3.0.

Line Sagging The layout of the ATLAS detector consists of two technologies (TRT in the Inner Detector, MDT in the Muon System) that describe drift tube measurements and hence need the concept of a `StraightLineSurface` for measurement expressions. `StraightLineSurface` instances, however, describe idealised mathematical objects, but in reality the tube wire described by this line is deformed due to sagging effects caused by gravity. Within the Inner Detector, this effect is very small and it is corrected for by a parametric approach with satisfactory accuracy. Thus the idealised straight lines can be used for track fitting in this context. For the Muon System, since the MDT tubes are by an order of magnitude longer, the sagging of the wire can not be described on calibration level very accurately. A prototype `SaggedLineSurface` has therefore been introduced that extends the `StraightLineSurface` with a sagging description. It allows to create an appropriate `StraightLineSurface` dynamically depending on the local z direction of the closest approach to nominal (parent) line. Figure 21 shows the UML class diagram for the `SaggedLineSurface`, and additionally for the common interfaces of distorted surface objects.

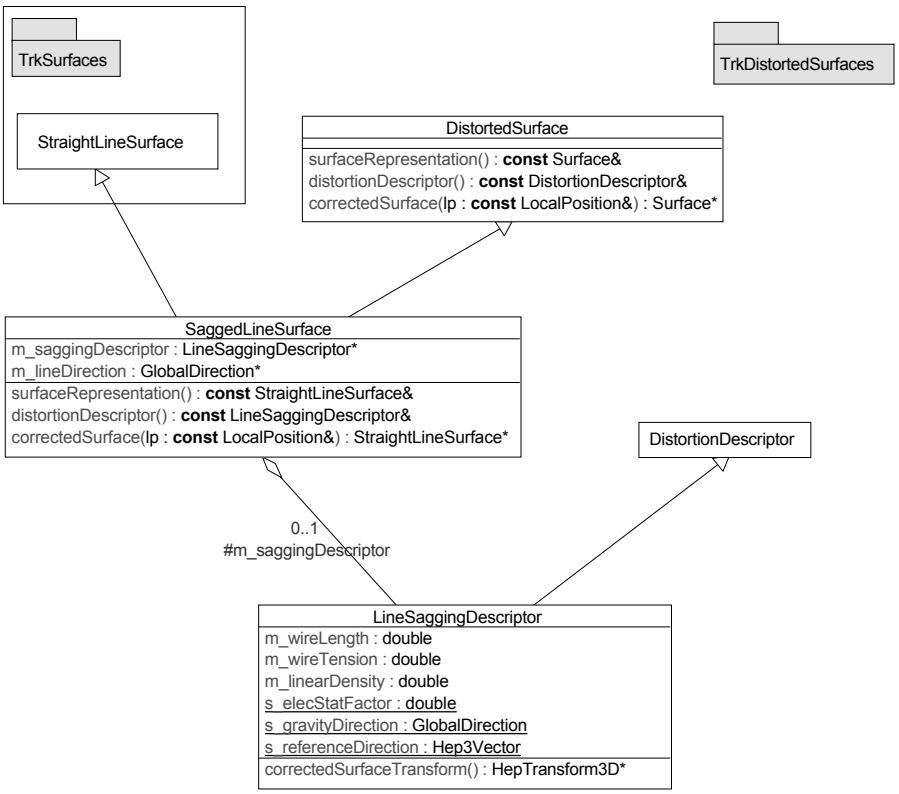


Figure 21: Simplified UML class diagram of a `SaggedLineSurface`. This class extends the `StraightLineSurface` and can therefore be naturally used within the extrapolation package. A concrete implementation of a `LineSaggingDescriptor` concentrates the mathematical description of a wire sag and provides a corrected rotation and position with respect to the local longitudinal direction on the line. Constructors and Destructors are omitted in this diagram.

7.2 Alignable `Surface` Objects

The alignment of the ATLAS detector will be a challenging task that requires various different robust, local and global alignment techniques. Recently, local alignment algorithms have been expanded using a Kalman filter formalism to estimate the geometrical correction parameters for the positioning of the sensitive detector modules [18]. In ATLAS, this technique has been first implemented for the SCT and pixel detector. Since the alignment Kalman filtering is an iterative process that requires the correction of module parameters repeatedly, a special flavor — the `AlignableSurface` — of tracking surfaces had to be introduced. Concrete implementations exist currently only as the `AlignablePlaneSurface`,

that extends both the `PlaneSurface` from the TrkSurfaces package and the `AlignableSurface` base class. This double inheritance schema enhances the use of the alignable surfaces with the extrapolation package, such that the dedicated Kalman fitter can automatically use this instance for the prediction steps after updating the positioning information. The inclusion of the `AlignableSurface` into the standard ATLAS Kalman fitter is realized by a `IAlignalbeSurfaceProvider` interface. If the Kalman fitter is configured to run in alignment mode, the concrete implementation of this interface simply exchanges the nominal measurement surface with an alignable surface for each prediction step.

8 Conclusion

A new geometry description for track reconstruction has been introduced to the ATLAS offline software. It serves the challenging needs of modern track reconstruction algorithms in terms of speed, flexibility and accuracy of the material description. It can account for material and geometrical distortions and has been designed in consideration of future calibration and alignment needs when starting real data reconstruction. The intrinsic navigation of the geometry — together with the coherently developed extrapolation package — builds a fundamental part of the new offline ATLAS track reconstruction and common Event Data Model. Several large scale test of the new ATLAS tracking chain have been successfully performed using taken data from the ATLAS combined testbeam in 2004, cosmics data in 2005/2006 and large scale Monte Carlo production.

A Appendix

A.1 Typesetting

The following type setting conventions are followed throughout this document: Software packages within the ATLAS offline software repository [19] are written in Sans-sarif face, C++ or python class names are written in `Courier` face. Namespace definitions as used in the software repository are omitted in this document for readability. An exhaustive list of software packages, their location within the ATLAS software repository and the used namespaces can be found in Tab. 4.

Table 4: Software packages described or referred to within this document.

Container Package	Leaf Package	Namespace
Tracking/TrkDetDescr	TrkDetDescrUtils	`Trk`
Tracking/TrkDetDescr	TrkDetDescrAlgs	`Trk`
Tracking/TrkDetDescr	TrkDetDescrTools	`Trk`
Tracking/TrkDetDescr	TrkDetDescrSvc	`Trk`
Tracking/TrkDetDescr	TrkDetDescrExamples	`Trk`
Tracking/TrkDetDescr	TrkDetDescrGeoModelCnv	`Trk`
Tracking/TrkDetDescr	TrkSurfaces	`Trk`
Tracking/TrkDetDescr	TrkVolumes	`Trk`
Tracking/TrkDetDescr	TrkGeometry	`Trk`
Tracking/TrkEvent	TrkEventPrimitives	`Trk`
InnerDetector/InDetDetDescr	InDetTrackingGeometry	`InDet`
Calorimeter	CaloTrackingGeometry	`Calo`
LArCalorimeter	LArTrackingGeometry	`LAr`
TileCalorimeter	TileTrackingGeometry	`Tile`
MuonSystem	MuonTrackingGeometry	`Muon`

A.2 Nomenclature

A.2.1 Three-dimensional Frames

Three different types of three-dimensional frames are used within this document or are referred to by the TrkDetDescr container:

- a coordinate system corresponding to the description of the magnetic field and the detector geometry, referred to as *global frame*.
- a cartesian frame moving along the track, the so-called *curvilinear frame*; the tangential momentum vector of the track builds the z-axis of this frame, x-axis and y-axis are then constructed with an additional global constraint.
- three-dimensional cartesian frames, different from the global frame, mainly attached to surfaces and volumes.

The standard cartesian set E of unit vectors describing the Tracking frame are indicated as

$$E = (\mathbf{e}_x\,, \mathbf{e}_y\,, \mathbf{e}_z\,). \tag{1}$$

The standard base of the cartesian curvilinear frame is, for convenience, indicated as

$$U = (\mathbf{e}_u\,, \mathbf{e}_v\,, \mathbf{e}_t\,). \tag{2}$$

Finally, the standard base H of a three-dimensional cartesian frame different or eventually different from the Tracking frame is noticed as

$$H = (\mathbf{h}_x\,, \mathbf{h}_y\,, \mathbf{h}_z\,). \tag{3}$$

In conjunction with a `Surface` object, this frame is sometimes referred to as *helper frame*, since it builds the carrier of the two-dimensional intrinsic surface frame (also called *local* frame).

A three-dimensional vector in either of the two frames is indicated by bold. It's expression with respect to the different frames is done using the standard base sets:

$$\mathbf{a} = a^e_x\mathbf{e}_x\, + a^e_y\mathbf{e}_y\, + a^e_z\mathbf{e}_z\, = a^e_u\mathbf{e}_u\, + a^e_v\mathbf{e}_v\, + a^e_t\mathbf{e}_t\, = a^l_x\mathbf{l}_x\, + a^l_y\mathbf{l}_y\, + a^l_z\mathbf{l}_z\,. \tag{4}$$

A.3 Utility Classes in the TrkDetDescr Realm

Several utility classes are used throughout the TrkDetDescr package and are grouped together in a TrkDetDescrUtils package.

A.3.1 Binning Schema

The hierarchy schema in the `TrackingGeometry` requires extended containers that provide a binning structure. The templated `BinnedArray` class imposes such a functionality, containing a standard STL vector container and a `BinUtiltiy` class, that translates local and global positions into the actual bin of the underlying STL vector. The `BinUtility` is extended by one-dimensional and two-dimensional concrete implementations deploying, equidistant, bi-equidistant and arbitrary binning. Evidently the complexity of the binning type influences the timing performance during global search actions in the `TrackingGeometry` navigation. The equidistant binning requires a simple integer division, the bi-equidistant two integer divisons and the arbitrary bin search a loop-like comparison. The binning schema is also used in the material description for the layer based material integration.

A.3.2 Reference Counting

The concept of sharing surfaces in the fully connective `TrackingGeometry` requires - together with the constraint of minimal memory consumption - the sharing of objects between nodes of relative higher hierarchy level. To still guarantee a safe memory cleanup, a reference counting class, the `SharedObject` class, is used for various objects in the `TrackingGeometry`.

A.3.3 Geometry Statics

The TrkDetDescrUtils package also provides a header file definition containing static objects declarations used within the geometry description: the nominal origin, an identity rotation and transformation, as well as the tracking frame axes are available in a singleton pattern design.

A.4 The main ATHENA Framework Components

A.4.1 `Service`, `Algorithm` and `AlgTool` Classes

The ATHENA software framework provides a pattern for the sequence steering of the program flow. Three main component base classes — a `Service` an `Algorithm` and an `AlgTool` class — are defined to be extended to concrete implementations of different purposes:

- The `Service` class is designed to provide dedicated functionality during the entire program execution, e.g. the magnetic field access is realised as an ATHENA `Service`. `Service` instances are handled by a central `ExtSvc` manager, that regulates its initialisation and finalisation.
- The `Algorithm` class is dedicated for actions be taken exactly one time at every event, e.g. most of the data preparation algorithms are realised as `Algorithm` classe. These have to be registered to a central `ApplicationMgr` in the job configuration that steers initialisation, finalisation and the execution of the `Algorithm` at every event.
- Unlike the `Algorithm` class the `AlgTool` provides the possibility to be called several times per event, mainly through an `Algorithm` that either owns the associated `AlgTool` or retrieves it from the central `ToolSvc`, where all public Tools are registered. This pattern allows `AlgTool` instances to be shared between different applications, such as e.g. the same `extrapolator AlgTool` instance is used at several places within the program flow.

All three framework components are made use of for the creation and validation of the ATLAS `TrackingGeometry`.

A.4.2 The Component Software Structure

As part of the ATLAS ATHENA framework, the new tracking geometry description software is realised in a component pattern design. Data classes and objects that require linking at compiliation time are situated in `installed` libraries, whereas `AlgTool Algorithm` and `Service` classes are grouped in `component` libraries, which enable dynamic loading of libraries at runtime. All interface definitions for component classes are concentrated in a TrkDetDescrInterfaces package,such tat dependencies between the various software components are kept at a minimum.

References

[1] V. Boisvert et al, *Final Report of the ATLAS Reconstruction Task Force*, ATLAS Note, ATL-SOFT-2003-010, 2003.

[2] Athena homepage,
http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture.

[3] Barrand G. et al., *GAUDI - A software architecture and framework for building LHCb data processing applications*, Proc. of CHEP 2000, 2000.

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

[5] ATLAS Quality Assurance Group, *Atlas C++ Coding Standard Specifications*, ATLAS Note, ATL-SOFT-2002-001, 2002.

[6] ATLAS Collaboration, *ATLAS Detector and Physics Technical Design Report*, CERN/LHCC/97-16 & 17, 1997.

[7] S. Agostinelli et al., *Geant4 — a simulation toolkit*, Nucl. Inst. & Meth. in Phys. Res., **A 506**, 2003.

[8] J. Boudreau and V. Tsulaia, *The GeoModel Toolkit for Detector Description*, Proc. of CHEP2004, 2004.

[9] CLHEP homepage, *http://cern.ch/clhep*

[10] POOL homepage, *http://cern.ch/lcg/Pool*

[11] ROOT homepage, *http://root.cern.ch*

[12] A. Salzburger, *The new Fast ATLAS Track Simulation (FATRAS)*, Proc. of CHEP2006, 2006.

[13] E. Lund, L. Bugge, A. Strandlie and I. Gavrilenko, *Simulatneous Track and Error Propagation with Material Effects*, ATLAS Note in preparation

[14] A. Salzburger, *The ATLAS Extrapolation package*, ATLAS Communication to be published, ATLAS-COM-SOFT-2007-008, 2007.

[15] COOL homepage, *lcgapp.cern.ch/project/CondDB/*

[16] Alverson, G. et al., *Status of the HEPVis class library*,FERMILAB-CONF-98-363 , FERMILAB, 1998.

[17] Open Inventor homepage, *http://oss.sgi.com/projects/inventor/*

[18] R. Frühwirth, T. Todorov and M. Winkler, *Estimation of detector alignment parameters using the Kalman fitter with annealing*, Journal of Physics G: Nuclear and Particle Physics, 29, 2003.

[19] ATLAS offline CVS repository, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*

Language ought to be
the joint creation
of poets and
manual workers.

Geore Orwell

Chapter 6

The Event Data Model

The development of a new common event data model (EDM) has marked — together with the introduction of the main geometry classes as described in Chap. 5 — the first part in the restructuring of the ATLAS track reconstruction software. In ATLAS, several attempts have been taken in the past to establish a commonly recognised EDM for all tracking devices, being flexible enough to serve the needs of the different algorithmic solutions of track reconstruction on the one hand, and simple enough to be directly used by physicists in the event analysis on the other hand. The author states with little proud, that this could finally be achieved with the new EDM that will be described in the following sections.

6.1 Introduction

Given the long lifetime of the ATLAS experiment, the EDM does not only have to fulfill the recent requirements from algorithmic reconstruction code, but should be flexible enough for future updates and adaption evoked by the development of new algorithms on the one hand, or by necessary changes evoked by first feedback from early data taking on the other hand.

Nothing worse than a future analysis that can not be performed due to the limitations of the EDM. Such a data model would rule itself out immediately. In a picture where the EDM is the language spoken between the algorithms, a linguist would clearly argue that if a language hinders you from expressing yourself, another language would easily be chosen.

A first, almost complete version of the ATLAS tracking EDM has been presented in an extensive document [6.1]. Recently, some adaption has been performed to optimise and conclude the functionality of the EDM. This update should give the reader enough understanding for the following sections of this document, for a complete picture of the ATLAS tracking EDM is the reader, however, referred to the original document. Recently [6.2], a second document has been published that describes the specific extensions of the EDM for the Inner Detector. It also reflects one of the main concepts of the deployed data model design: particularities of specific detector technologies are only present in extended child classes (such as the ID specific implementations of the tracking EDM classes). This is, to avoid heavy interfaces on the one hand, and to decouple unrelated detector elements in the software design, on the other hand.

6.2 The ATLAS Tracking Event Data Model

ATL-SOFT-PUB-2007-003, 14 p.

Contributions of the Author

The author's contribution to the tracking EDM has been on several levels: large contribution on the overall design concepts (e.g the three column structure of base classes for parameterisation, measurement and particle representation) that can not be listed easily, such as the specific realisations of the entire track parameterisation concept including the charged/neutral template schema or the solution for the identification key to flag measured parameters have been done by the author in person. However, it should be mentioned that the overall outcome of the EDM design is a product of good piece of teamwork and would not have been possible without constant discussion and exchange of ideas.

Bibliography

[6.1] P. F. Åkesson, T. Atkinson, M. J. Costa, M. Elsing, S. Fleischmann, A. N. Gaponenko, W. Liebig, E. Moyse, A. Salzburger, and M. Siebel. Atlas tracking event data model. *Public ATLAS Note*, ATL-SOFT-PUB-2006-004, 2006.

[6.2] P. F. Åkesson, M. J. Costa, D. Dobos, M. Elsing, S. Fleischmann, A. Gaponenko, K. Gnanvo, P. T. Keener, W. Liebig, E. Moyse, A. Salzburger, M. Siebel, and A. Wildauer. ATLAS Inner Detector Event Data Model. *Public ATLAS Note*, ATL-SOFT-PUB-2007-006, 2007.

Updates of the ATLAS Tracking Event Data Model (Release 13)

T. Cornelissen, M. Elsing, A. Wildauer
CERN
N. van Eldik, E. Moyse
University of Massachusetts, USA
W. Liebig
NIKHEF, Amsterdam, The Netherlands
N. Piacquadio
Albert Ludwigs Universität Freiburg, Germany
K. Prokofiev
University of Sheffield, UK
A. Salzburger*
Leopold Franzens Universität Innsbruck, Austria & CERN

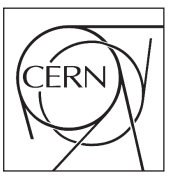
December 6, 2007

ATL-SOFT-PUB-2007-003
13 December 2007

Abstract

In a previous document [1] we have presented the ATLAS tracking *Event Data Model* (EDM) that has been developed during the recent restructuring of the ATLAS offline track reconstruction. The tracking EDM has become a cornerstone of the new modular track reconstruction algorithms of both tracking devices of the ATLAS detector, the Inner Detector and the Muon System. Recently, some components have undergone yet another design evolution targeted at completing missing modules and at establishing anticipated functionality for the startup of the ATLAS experiment. One particular aspect of the EDM is that is does not only have to fulfill the requirements of today's algorithmic modules, but has to provide the flexibility for future developments. This document is based on the ATLAS software release 13.0.40.

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*



*corresponding author: Andreas.Salzburger@cern.ch

1 Introduction

A common event data model (EDM) is inevitable for the component software structure given by the ATHENA framework [2]. It allows to define interfaces for the single tasks of the overall reconstruction process. In other words, it defines the language that is *spoken* by the reconstruction algorithms. In a previous paper [1] we have presented the concepts, design and implementation of the ATLAS tracking EDM to very detail. The tracking EDM has become a cornerstone for the main ATLAS track reconstruction algorithms in both the offline and third level trigger (*Event Filter*) application.

The tracking EDM has been deployed at full production level for almost three major release circles and has undergone several extensions and modifications, adapting either to new algorithmic developments or simply as an outcome of constant evaluation of the complied functionality. Recently, yet another substantial design evolution has taken place which was mainly motivated to complete missing modules and establish full functionality for first data taking with the ATLAS detector. A new, neutral track parameterisation has been included to enhance constrained vertex fitting that includes both charged and neutral particle representations. Additionally, the common `Track` class has been extended to store material descriptions on reference surfaces that have been used in the track fit. Pseudo-measurements have been introduced to stabilise poorly constrained fits, and the segment representations have been coherently modified to enhance the inclusion of the pseudo-measurement objects.

A major design revolution has taken place for the representation of the tracking information for physics analyses, the `TrackParticle`. This has been motivated to achieve full support of `TrackParticle` objects by common tracking tools[1] and to optimise the class shape for a common use of track representations from both tracking devices, the *Inner Detector* (ID) and the *Muon Spectrometer* (MS).

This document describes the extensions and modifications that have been applied to the tracking EDM since release 12.0.0 and presents — if necessary for the understanding of the presented context — a short review of the various modules that are integrated in the tracking EDM. This document is based on ATLAS offline release 13.0.40, a consecutive production release on basis of the ATLAS release 13.0.0. The authors are fully aware that this document only presents the corresponding status of the EDM at one particular point in time. The EDM is the language that is used for the communication between the individual parts of the full reconstruction chain — and consequently, like every language, it is matter of modification and evolution. Future changes to the tracking EDM may thus be presented in consecutive documents.

1.1 The `Track` Class

The ATLAS Tracking EDM is concentrated around a flexible `Track` class that is realised as a container of `TrackStateOnSurface` (TSOS) objects. The single `TrackStateOnSurface` objects are able to hold polymorphic tracking information: track expression with respect to a given surface in form of track parameters objects, hit information by an extended class of a common `MeasurementBase` class, traversed material or integrated material effects, a `FitQuality` object and an identification type. Table 1 lists the possible base classes that can be held by the `TrackStateOnSurface` class. The fitted track representation to the nominal beam line or vertex position, the so-called `Perigee`, is e.g. represented by one `TrackStateOnSurface` object, just like every single hit, a track segment, or even a fitted scattering angle. The flexible container structure of the `Track` object guarantees hereby that one single trajectory representation can be used for different fitter types, tracking devices or reconstruction algorithms. With release 13.0.0, the way how to store material interactions on a `Track` object has been changed (see Sec. 6), which led to an extension of the `TrackStateOnSurface` class; Figure 1 shows an illustrated track as a collection of `TrackStateOnSurface` objects.

Track fitting requires, in general, two main data structures: the representation of measurements to be integrated in the track fit, and the representation of the (fitted) trajectory, in the following referred to as *track parameterisation*. In ATLAS, track fitting is carried out on calibrated hits or hit collections that extend a common base class, the `MeasurementBase`. The track parameterisation is done using classes that extend a `ParametersBase` interface, restricted to a charged parameterisation for child objects that can be stored in the `Track` class. This is because neutral particles are not matter of tracking in the sense of track finding and fitting, since they do not cause hit signatures in the tracking

[1] E.g. the extrapolation engine or the vertexing tools.

Table 1: The different (base) classes that can be contained by the `TrackStateOnSurface` class. All of the given classes are contained and returned by pointers, thus they can be left out optionally.

Leaf Package	Description
`FitQualityOnSurface`	χ^2 and number of degrees of freedom
`TrackParameters`	the track parameterisation (or intersection) w.r.t. a surface
`MeasurementBase`	the measurement expression
`ScatteringAngleOnTrack`	a fitted scattering on track
`MaterialEffectsOnTrack`	applied material effects description

devices. The found tracks are input to higher level reconstruction modules such as vertex fitting or topological event fitters that include neutral particles in particular for mass-constrained fits. Common tracking tools such as the extrapolation engine or the various vertex fitters need to operate on both neutral and charged track representations. The formerly existing model that was limited to charged track parameterisations has therefore been extended to cope with charged and neutral trajectories in a type safe way. The newly integrated schema is presented in further detail in Sec. 2, the adaption of the vertexing data model to this new base class is described in Sec. 5. The full object tree of measurement representations that has been recently adapted with pseudo-measurements and modified track segment representations is presented in Sec. 3.

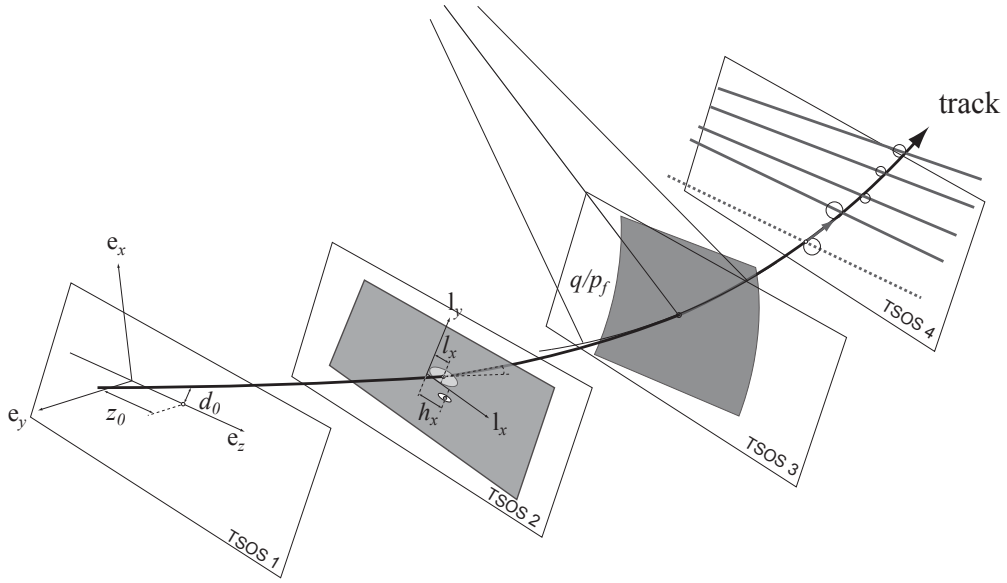


Figure 1: An illustrated `Track` as a container of four different `TrackStateOnSurface` (TSOS) objects. TSOS 1 contains the track representation to the nominal beam line, the so-called `Perigee` representation. TSOS 2 holds both a measurement through a hit information and the fitted track parameters on the measurement surface, while TSOS 3 represents an applied material interaction — illustrated through a curvature change of the track. Finally, TSOS 4 integrates several fitted measurements through a single segment representation.

1.1.1 Track Slimming

One important aspect of the tracking EDM is the size which the written data takes when being stored on disks. This is far less trivial than it first sounds: on the one hand, small *persistent* representations of the tracking EDM are necessary to comply with the computing budget of the experiment, but on the other hand, as much information as possible should be accessible for the physicist to allow optimal event analysis techniques. In the context of the tracking EDM, a major step towards achieving a good balance between disk size and usability was to identify all information that can be recreated when the track is read back from disk storage. In principle, this includes all fitted parameters and estimated

covariances, but excludes obviously the hit collection. A simple refit such a *slimmed* track after it has ben read from the persistent storage would recreate the full track information as achieved in the original event reconstruction. The flexible TSOS container design of the `Track` class was hereby a key feature, since it allows to create a track collection of stripped hits and a `Perigee` representation[2] that is then written to disk. The track collection size could be significantly reduced (depending on the track collection, the reduction factor varies between 6 to 10).

Representation for Physics Analysis Few analyses based on data taken with the ATLAS detector will directly incorporate the `Track` objects. The `Track` itself is, in general, not more than a trajectory representation of the particle when passing through the detector, while the — for the event analysis — most important representation of the particle as a four momentum vector at the production vertex is not given by the `Track`; neither is particle identification[3] nor the vertex association performed at the stage of track reconstruction. In the ATLAS EDM, the `Track` information is represented as a `TrackParticle` object for further use in a particle-oriented event analysis. Vertex fitting with or without constraints can be performed on `TrackParticle` objects, but needs the extrapolation engine to express the trajectory with respect to the (iteratively fitted) vertex position. To enhance common tracking tools to work together with the `TrackParticle` object (which combines a broader bundle of aspects to be dealt with in event reconstruction), without breaking the philosophy of keeping the tracking modules independent from specific reconstruction algorithms, a new `TrackParticleBase` class has been introduced that concentrates the tracking-relevant information and builds the new interface for tracking tools. These tools are designed to operated also on event reconstruction and analysis level; a detailed description of the new `TrackParticleBase` class can be found in Sec. 4.

2 Trajectory Parameterisation: The `ParametersBase` class

The parameterisation of a particle trajectory with respect to a given surface is inevitable for track reconstruction. It can be done in many different ways, for a charged trajectory in magnetic field a minimal set of five parameters has to be chosen; it can be reduced by one parameter for a trajectory representation in a no-field environment or a neutral particle that follows a straight line. This is, since the charge q and the momentum magnitude p are superfluous for the purely geometrical description of a line. For constrained vertex fitting that includes both charged and neutral particle traces, however, the momentum (hypothesis) is necessary — see Sec. 5.

The trajectory parameterisations for both neutral and charged particles are thus realised in the ATLAS tracking EDM as a set of five parameters

$$\mathbf{x} = (l_1, l_2, \phi, \theta, c/p)^T, \tag{1}$$

when l_1 and l_2 denote the local coordinate expression on the given surface (and thus depend on the surface type), ϕ and θ are the azimuthal and polar angle, respectively, and c is defined as

$$c = \begin{cases} q & \text{if } q \neq 0, \\ \\ 1 & \text{if } q = 0. \end{cases} \tag{2}$$

For every surface type that is defined in the ATLAS reconstruction geometry [4], a dedicated parameterisation exists, realised by a specific class to ensure an unambiguous identification of the given measurement frame. In track fitting — since the trajectory itself can not be measured, but only a localisation at discrete points in the detector can be done — a set of *measurement mapping* functions h_j is needed to map the track parameterisation on a measurement surface to the measured coordinates and thus to establish a *predicted* measurement[4]. This yields for the single predicted measurement

[2] This is for the simple convenience of the user that is not forced to refit the track collection if the focus is only drawn onto the impact parameterisation.

[3] Only a `ParticleHypothesis` exists for the steering of material effects integration.

[4] Since the two most common track fitting techniques, the least squares method and the Kalman filter are both linear estimators, these measurement functions are even required to be linear, or at least approximated by a linear function.

components

$$m_j^{pred} = h_j(\mathbf{x}), \tag{3}$$

or, when using matrix notation and assuming linear measurement mapping functions at the detection device i

$$\mathbf{m}_i^{pred} = \mathbf{H}_i \mathbf{x}_i. \tag{4}$$

The predicted measurement is necessary when building the hit residuals $\mathbf{r}_i$, where the measurement position $\mathbf{m}_i$ on the ith device is compared with the predicted or fitted trajectory intersection on the measurement surface, explicitly

$$\mathbf{r}_i = \mathbf{m}_i - \mathbf{m}_i^{pred} = \mathbf{m}_i - \mathbf{H}_i \mathbf{x}_i. \tag{5}$$

Since in ATLAS the track parameterisation on a given surface has been chosen to be expressed in local surface coordinates, the measurement functions are simply given by projection matrices, yielding

$$\mathbf{H}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{6}$$

for a one-dimensional measurement in the first local coordinate, and

$$\mathbf{H}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

for a two-dimensional position measurement, respectively. Most actual measurements taken with the technologies of the ATLAS sub-detector are given in these two forms. However, the ATLAS tracking EDM is not restricted to measurements of this type. Instead, every possible combination of sub-sets measured out of the full parameterisation can be chosen, which is exploited by the new `PseudoMeasurementOnTrack` class and the `Segment` classes in particular, see Sec. 3. This leads to 31 different projection matrices $\mathbf{H}_i$ and requires in addition an internal identification schema for the parameter vector[5]. The `LocalParameters` class that extends the `HepVector` class of the CLHEP [3] maths library establishes a mechanism that provides the user with an internal identification schema — realised in a bitwise packed key i — of the contained parameters and the appropriate projection matrix. An example of the simple usage of the `LocalParameters` class can be found in the Appendix, Sec. A.2. In this sense, the chosen track parameterisation can be regarded as being motivated by the measurement setup of the detector. Figure 2 shows a track expressed w.r.t. two different reference surfaces in the ATLAS tracking EDM.

Since the parameterisations of a trajectory state with respect to a given surfaces differ for charged and neutral particles only slightly in the interpretation of one parameter, it would be a natural attempt to represent them with the same EDM object. Equation (2), however, shows the potential danger of such a choice: a particle with charge $q = 1$ would then be expressed through the same signature as a neutral particle represenation of same momentum. One main design principle of the ATLAS tracking EDM is *type safety*, i.e. any misinterpretation of a given data object has to be avoided[6]. It can be argued that a simple check of the charge parameter q could distinguish the two identical cases, but this relies on every user to perform this check before the parameter vector can be interpreted. A type safe model has to be favored preferably without code duplication. In ATLAS, this is realised through a C++ template method that allows to define the actual class type through the template parameter. A small helper class, the so-called `ChargeDefinition` that is extended by a `Neutral` and a `Charged` class is hereby used to define the appropriate template argument of the given track representation. A charged trajectory — when being expressed with respect to a planar surface — is then defined as a class extending an `AtaPlaneT<Charged>` class, while a neutral trajectory representation is enhanced by an `AtaPlaneT<Neutral>`. Since the common implementation is shared by the two classes, but type safety is preserved through the template mechansim, both main requirements — type safety and avoiding code duplication — have been met. The `ChargeDefinition` classes define for convenience an implicit cast operator to a `double`, returning zero for the `Neutral` and ± 1 for the `Charged` class, respectively. Since the charge is not stored otherwise in the parameter classes, it is impossible to construct an inconsistent object.

[5] There are, in total, $32 = 2^5$ possible subsets of parameters, but the empty set is excluded, since there exist no invalid measurement representations in the ATLAS tracking EDM.

[6] Clearly no one wants to have a curved photon track in magnetic field, simply for the fact that the used propagation tool misinterpreted the $1/p$ parameters of a neutral representation as a q/p information.

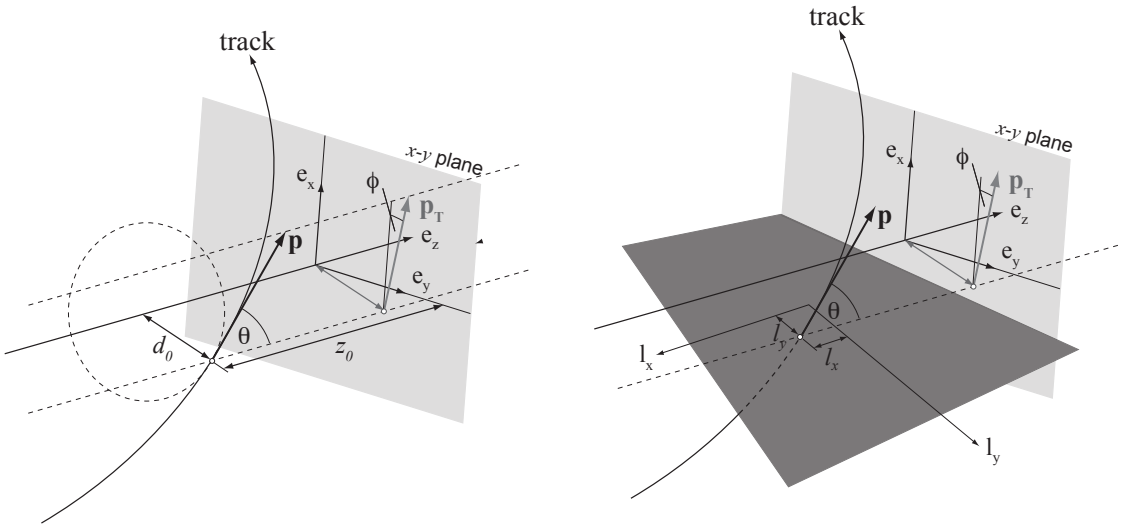


Figure 2: A track parameterised with respect to two different surfaces: the expression to the nominal z axis yields the `Perigee` representation of the track to the left, while the expression of an intersection with a planar surface (right) is described by the `AtaPlane` object. The parameterisations differ only in the first two local coordinates that are defined by the surface type and are optimised with respect to the given detector layout. The momentum expression through the azimuthal angle ϕ, the polar angle θ and the (charged) inverse momentum is identical for both cases.

Hidden Template Method The authors are aware that template solutions are in general not amongst the most popular techniques within the client community and track representations belong clearly to the most widely spread classes of the ATLAS tracking EDM. The template resolving has therefore be *hidden* from the user through inserting actual class types for the track parameterisations on the various surfaces for charged and neutral particles that extend the class templates to non-virtual objects[7]. Figure 3 shows an UML class diagram that illustrates the charged and neutral track parameterisation with respect to a planar surface.

The `ParametersBase` base class is restricted to the attributes that are identical for both a neutral and a charged trajectory parameterisation and can be used for applications that only work on the global parameters of a trajectory expression, i.e. a position, a momentum and the charge. The template mechanism, on the other hand, forces the client to resolve the template argument and consequently an object has to be identified to be either of `Neutral` or `Charged` flavor, before the parameters vector can be retrieved[8].

3 Measurement representation: The `MeasurementBase` Class

Measurement representations exist in manifold ways in the ATLAS tracking EDM: in most of the cases, measurements are directly integrated as fully calibrated representations clusters or drift radii. These objects are realised as classes that extend the `RIO_OnTrack` class, and represent either one-dimensional or two-dimension measurements; the calibration applied on the input objets from the clusterisation process (in ATLAS terms `PrepRawData` objects) is hereby based on the already collected track information. In the MS, a second additional calibration step is applied on `RIO_OnTrack` objects in the preparation phase for track fitting (*pre-tracking*), that is based on the local pattern recognition output for the various detector chambers.

As described in [1] an even more flexible way of representing single and combined measurements with a extended `MeasurementBase` object has been implemented in ATLAS. These types include pre-grouped (and fitted) measurements as `Segment` realisations and a dedicated competing measurement collection

[7] The technically interested reader may find that the class templates mark virtual class descriptions and can thus not be instantiated in the program flow.

[8] In C++ terms this is done using the `dynamic_cast` operator.

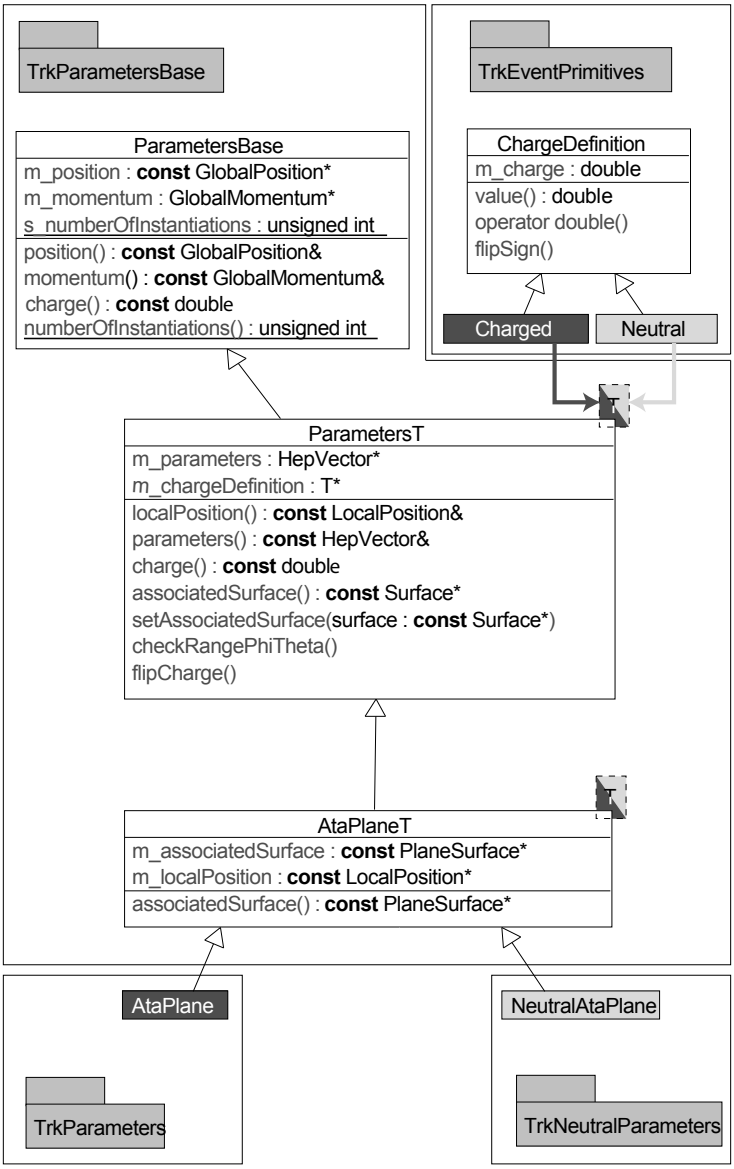


Figure 3: The charged and neutral track parameterisation as realised in the ATLAS tracking EDM. A `ChargeDefinition` base class and its extended `Charged` and `Neutral` classes are located in the TrkEventPrimitives package. The two child classes are used to define the type of the templated trajectory representations — in the illustrated example with respect to a `PlaneSurface` of the ATLAS reconstruction geometry.
The `flipCharge()` and the `flipSign()` methods are hereby necessary just for technical aspects, but are not accessible for the client.

(`CompetintRIOsOnTrack`) on one detection surface that enhance fuzzy hit assignment techniques. Both classes can be resolved into their constituents and thus can the hits be individually included in the track fit. On the other hand, the `Segment` class represents already several measurements at different discrete positions in the detector and holds, in general, more information than a simple localisation of the track trajectory on a reference surface. This is in many cases at least one directional parameter of the momentum vector, or even a momentum estimate itself. The `Segment` classes can therefore be used for fast track matching and play a particularly important role in the MS pattern recognition.

`RIO_OnTrack` and `Segment` objects are mainly used in track fitting or local pattern recognition. For pattern recognition, a global representation is often used for trajectory seeding on the one hand, or for global pattern techniques (such as histogramming methods or Hough transformations) on the other hand. The `SpacePoint` class object establishes global hit representations of measurements. A `SpacePoint` can hereby represent a single or a collection of measurements if the spatial information provided by one measurement together with the measurement surface is not sufficient to constrain a global position.

All these different measurement representations extend one single base class, the `MeasurementBase` interface. The `MeasurementBase` concentrates the minimal information needed for track fitting, i.e. a one- to five-dimensional measurement, the according measurement errors and the corresponding

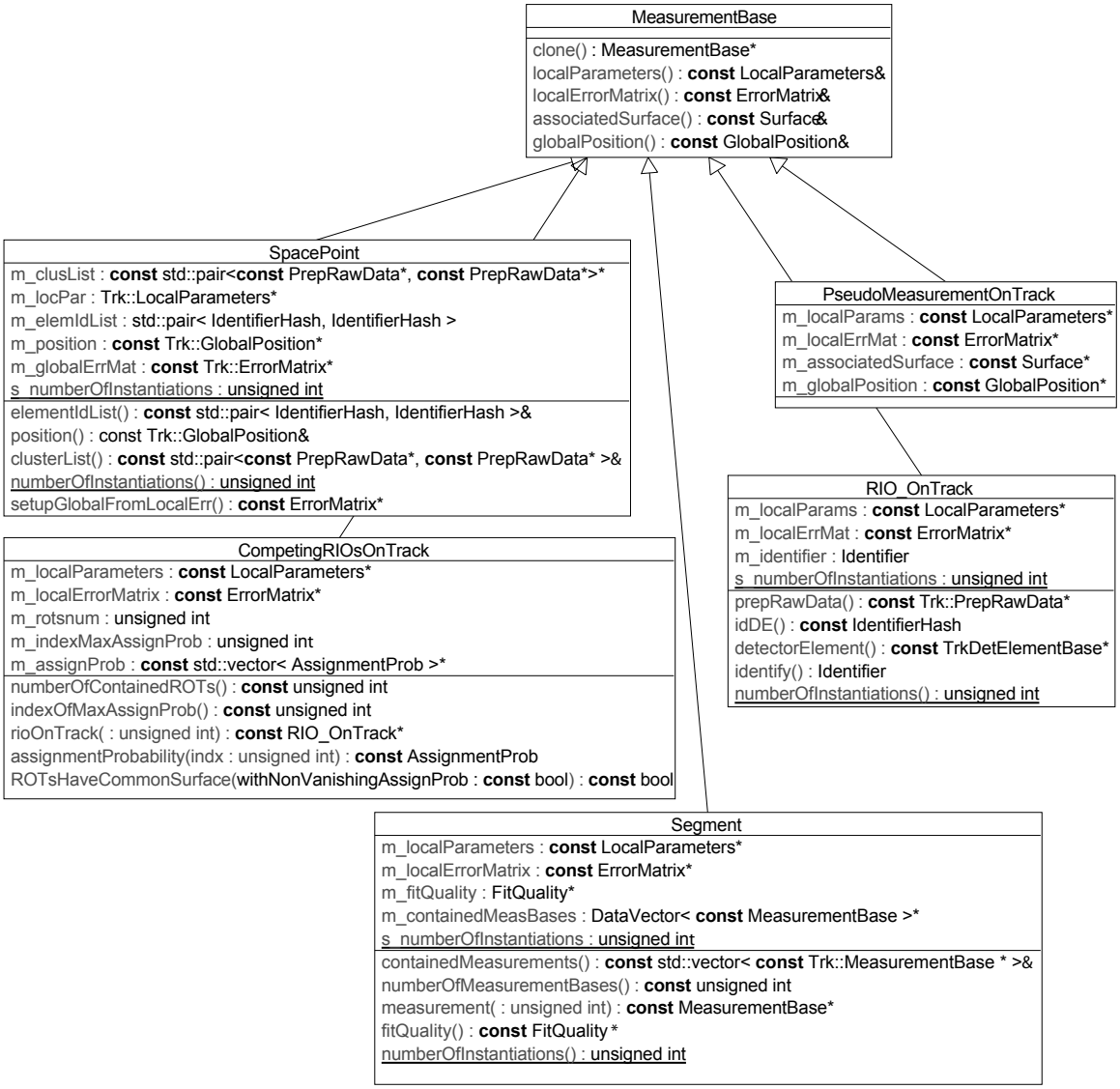


Figure 4: Simplified UML diagram, showing the classes that extend the `MeasurementBase` class and can be used for the representation of one to many measurements in the ATLAS tracking EDM. Only the base classes of the inherited families are shown, concrete detector-specific implementations of most of the classes can be found in the corresponding sub-detector repositories.

reference frame that is provided by a `Surface` object of the ATLAS reconstruction geometry. In fact, the `ITrackFitter` interface is defined to operate only on a set of `MeasurementBase` objects and thus common track fitter implementations are completely shielded from actual measurement representation. A polymorphic track fit, combining several different types is often performed and eases successive hit inclusion, track segment matching or simply saves CPU time when a prior fit has already compacted the information of several single hits into one representation.

Pseudo-Measurement Representations In ATLAS, there exist several tracking devices that can not determine the full track parameterisation without information from other devices: e.g. the *Transition Radiation Tracker* (TRT) in the Inner Detector or the *Monitored Drift Tube* (MDT) chambers in the Muon Spectrometer. Although a full track parameterisation can not be obtained, it is necessary that isolated measurement sets from these devices can serve as an input for track fitting. For the ID this in particular important to enhance track segment search from the detector boundary towards the

interaction point. In the MS, on the other hand, the track finding starts often with a segment search on chamber level. To constrain the track fit it is necessary to include an artificial measurement with a crude estimate of the missing coordinate. This estimate can be at the level of *within the chamber* or *in the barrel*, i.e. something that can be included without biasing the fit result but can be postulated on the other hand by the pure existence of the measurement. A generic `PseudoMesurementOnTrack` has therefore be created for such a purpose. It extends the `MeasurementBase` base class, and implements the minimal interface necessary for fitting. The UML class diagram can be seen as part of Fig. 4

Track segment adaption to the pseudo-measurements The inclusion of fake measurements evoked a necessary adaption of the `Segment` classes. Segments are often used for local pattern recognition in the track fit of hit collections from single detector components that are prone to lead to unconstrained fits. It is thus necessary that `Segment` objects can contain the new `PseudoMeasurementOnTrack` objects which led to a shift from `PrepRawData` to `MeasurementBase` as the top level class contained by hit collections in a single track segment. Since segments are the most complex objects in the `MeasurementBase` tree and depend on the algorithm that has formed them, a new authorship schema has been included to achieve transparency about the object origin in higher level reconstruction algorithms. A list of currently registered `Segment` authors can be found in Sec. A.3.

4 The Analysis Object: the `TrackParticleBase`

In the ATLAS computing model, there exist two main levels of event data processing, that are reflected by the written data collections: the *Event Summary Data* (ESD) and the *Analysis Object Data* (AOD). ESD based algorithms and modules are in total part of the event reconstruction, and clearly first stage analyses to be done with the ATLAS detector, mainly targeted at evaluating the detector performance and calibrating or aligning the detector components will need access the event data to full detail level. However, recording and transferring all data taken with the ATLAS detector to remote sites at ESD level would exceed the computing capacities of the experiment and — in particular — of the attached institutes. Therefore a compression of the ESD to AOD is done, that should still provide sufficient information for modern analyses techniques, while reducing the disk storage significantly. The track representation in the AOD containers is the so-called `TrackParticle`. Since the `TrackParticle` combines information from tracking devices, calorimetry and particle identification algorithms, it can not be subject of the tracking EDM. Until recently this resulted in the fact that the `TrackParticle` class was not known to common tools used in track reconstruction. Reconstruction tools therefore acted mainly as wrappers for common tracking tools, forwarding the only available information — the measured track representation at the `Perigee` definition close to the interaction point — to common tracking tools such as vertex fitters and the extrapolation engine. In particular for tracks being found with stand-alone reconstruction algorithms in the Muon Spectrometer, this situation was far from optimal.

Recently, the `TrackParticle` class has undergone a major design evolution that established a new `TrackParticleBase` class in the `Tracking` repository. The introduced `TrackParticleBase` concentrates the tracking relevant data, while remaining independent from higher level reconstruction algorithms. The restrictive definition of the `TrackParticle` to be defined only through a `MeasuredPerigee` object has vanished in the base class and has been replaced by a vector of `ParametersBase` objects. Yet another reason why an intervention has become inevitable was the integration of the neutral track parameterisation that, consequently, has to be followed in the data hierarchy down to the analysis representation. However, keeping in mind that all inherited child classes are aimed to describe objects to be used in physics analyses and thus a Lorentz-vector representation should be existent, one of the parameters provided has to be used to define the four-momentum vector. This parameter has to be identified explicitly at the construction of the `TrackParticleBase` object. The widely used `TrackParticle` now inherits from the `TrackParticleBase` class and again restricts the *defining parameter* to be the `MeasuredPerigee` representation (or, e.g. the `NeutralMeasuredPerigee` for a neutral particle representation). This is, since the analysis of the underlying event clearly deals with the particle parameters closest to production vertex. Figure 5 shows a simplified UML class diagram for the newly created `TrackParticleBase`, Fig. 6 illustrates how the muon `TrackParticle` profits from the more flexible way of holding parameters at several stages within the detector.

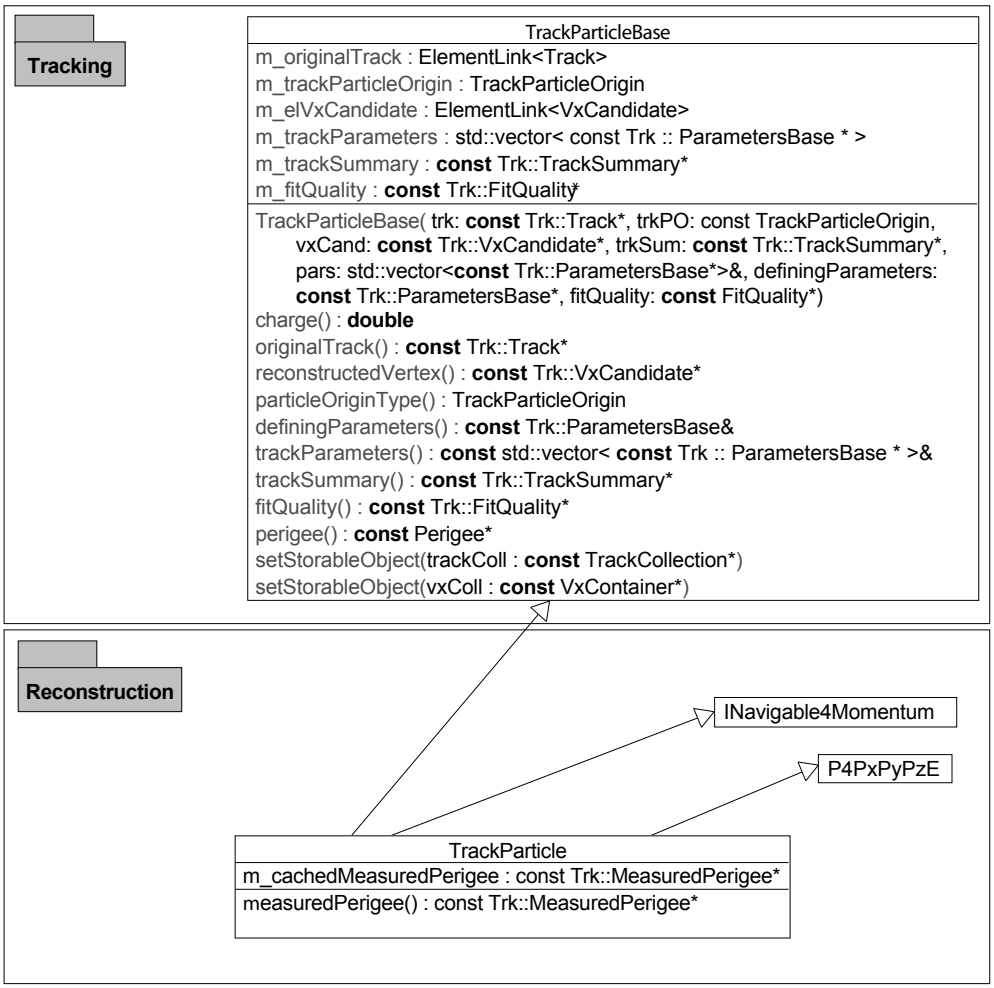


Figure 5: Simplified UML diagram, showing the new `TrackParticleBase` and the extended class `TrackParticle` that marks the analysis representation of tracking. The constructor of the `TrackParticleBase` shows the new philosophy that allows multiple representations of the underlying track within the detector, while keeping one `ParametersBase` object specifically outstanding to identify the track state where the four-momentum is defined.

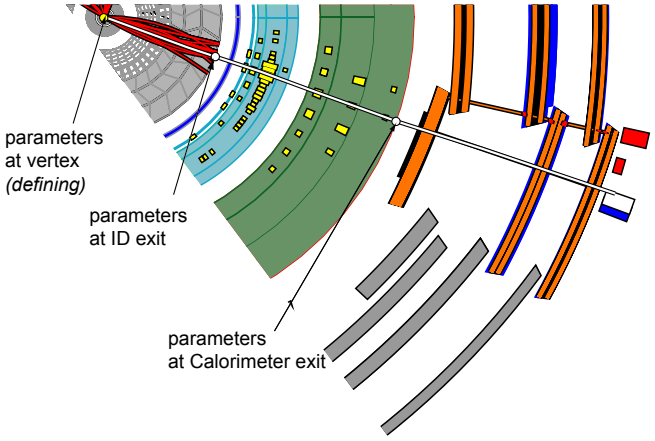


Figure 6: The new `ParticleBase` object illustrated in an example based on the ATLANTIS [5] event display. The `Track` is hereby represented with one single `TrackParticleBase` object at three different stages in the detector: as a `MeasuredPerigee` expression close to the interaction point (defining parameters), through `TrackParameters` at the exit of the Inner Detector and the Calorimeter, respectively.

5 Event Data Model for Vertex Reconstruction

The part of the ATHENA event data model that is related to the vertex reconstruction has changed significantly between releases 12 and the latest production release on top of the 13.0.0 base release. In general, this changes are related to the introduction of two new EDM concepts: the use of the `ITrackLink` and the use of the `Trk::ParametersBase` class — which is described in more detial in Sec. 2. The new parameterisation base class replaces the formerly used `Trk::TrackParameters` class, that is limited to charged trajectories. This marks a first step of preparation towards the vertex reconstruction EDM for fully constraint and kinematic fitting algorithms[9]. Starting from ATHENA rel. 13.0.20, the dependencies on the `Trk::TrackParameters` are dropped in all classes of the vertexing EDM; the use of `Trk::ParametersBase` is introduced instead. This change has been motivated by the fact that the `Trk::ParametersBase` is the new common base class for both neutral and charged track parameters. The future inclusion of neutral trajectories in the vertex fits thus becomes possible[10].

5.1 The `Trk::ITrackLink` class

The `ITrackLink` class is a new part of the Tracking EDM, implemented in the dedicated TrkTrackLink package. The class itself is purely abstract and so far has only two concrete implementations: the `Trk::LinkToTrack` class in the `TrkTrack` package and the `Trk::LinkToTrackParticleBase` in the `TrkParticleBase` package. The latter two classes also inherit from the `ElementLink <TrackCollection>` and `ElementLink <TrackParticleBaseCollection>` respectively.

This schema of inheritance allows one to use a pointer to the `Trk::ITrackLink` to manipulate both a `Trk::Track` and `Trk::TrackParticleBase` and thus use the same EDM components on both AOD and ESD levels.

An example of the use of a pointer to the `Trk::ITrackLink` as a private member, can be found in the new version of the `Trk::VxTrackAtVertex` class. Here the use of the `ITrackLink` is justified by the fact that it gives the access to initial parameters of tracks used to fit a vertex, disregarding, whether the fit was done on the AOD or the ESD level. This allows to reduce significantly the storage size of reconstructed vertices: the initial perigee parameters of each trajectory does not need be stored anymore.

The disadvantage of the use of the `ITrackLink` is that the user needs to perform a `dynamic_cast` operation each time the knowledge of the exact type of the `ElementLink` is required. The parameters of the trajectory, however, can be accessed directly via the a dedicated access method.

6 Material Effect Description on Track

The correct treatment of effects caused by the interaction of the particle with detector material is inevitable for track reconstruction. A dedicated reconstruction geometry [4] is therefore built and provides a simplified version of the ATLAS detector setup to track reconstruction algorithms. The integration of material effects happens, in general, during the track fit itself and is thus not necessarily matter of the event data model. However, the tracking EDM has been recently expanded to store the applied corrections due to material interactions directly on the track class. This can be either a fitted parameter from global track fitting, realised through a `ScatteringAngleOnTrack` object, or a more profound description of the traversed material in terms of radiation length X_0 and applied energy loss corrections. Latter is made possible through a `MaterialEffectsOnTrack` object. Both descriptions can be stored as primary classes on a `TrackStateOnSurface` instance, see Sec. 1.1.

Allowing to store material effects together with the track object has been motivated by the following considerations:

- recent developments, e.g. in electron track fitting, can provide information about the applied material effects for further processing, such as track kink finding, or calorimeter cluster matching;

[9] In a second step, which is currently under preparation, a new extended track parameterisation will be introduced that adds the mass parameter to the existing track parameterisation as described in Sec. 2

[10] The obvious disadvantage of this change is the need to `dynamic_cast` the parameters returned by a particular class in order to understand whether this is a charged or a neutral trajectory.

- stored material effects on track will allow refitting of tracks from different sources that use different material descriptions, in particular for global refitting, where several competing material descriptions exist. This enhances a dedicated validation and comparison of the different material integration algorithms;

Both `MaterialEffectsOnTrack` and `ScatteringAngleOnTrack` have been extended to host a surface which defines the spatial information along the track, where the actual update had been integrated.

Material Integration from custom Sources The `MaterialEffectsOnTrack` class has also become the data object for custom material integration into the track extrapolation engine [6]. In track extrapolation, the description of the material is, in general, taken from a simplified detector description. The extrapolation engine, however, allows users to optionally provide custom material effects to the extrapolation process. The integrated `MaterEffectsOnTrack` objects can hereby originate from other parametric descriptions, material maps, or even from detector measurements such as the energy loss application based on calorimeter measurements for combined track reconstruction.

7 Conclusion

Four years after the *Final Report of the Reconstruction Task Force* (RTF) [7] invoked a redesign of the ATLAS reconstruction software (based on a common EDM and a component software structure), and less than one year before data taking with the deployed ATLAS detector, the tracking EDM is close to completion. The tracking EDM classes provide the necessary functionality for all current aspects of track reconstruction and have been optimised in both functionality and memory usage. The tracking EDM has been recently adapted for new developments and expanded to support common tacking tools in the user event analysis.

It has proven its stability and functionality during test beam data taking and the commissioning runs using cosmic rays, while being flexible enough to integrate new developments and corrections without breaking existing service. The ultimate aim is to provide a powerful EDM for the long lifetime of the experiment that allows backward compatibility for reading data during the entire period of data taking.

A Appendix

A.1 CVS Repository Structure

The described EDM classes in this document can be entirely found in the Tracking/TrkEvent container package of the ATLAS CVS software repository [8]. The referred surface classes are located in the Tracking/TrkDetDescr/TrkSurfaces package, and the event primitives, e.g. the position vectors, algebraic vectors and matrices are either part of the CLHEP math library or bundled in the TrkEvent/TrkEventPrimitives utility package.

A.2 The `LocalParameters` Class

While track parameterisations always provide the full five parameter set to describe the track (see Sec. 2), measurements can be expressed as any subset of one to five parameters of the full parameterisation (Sec. 3). In track fitting, where usually residuals on measurement surface are build that are the differences between the predicted measurement that is given through the track parametersiation and the track extrapolation and the actually taken measurement, it is necessary to collaps the full parameters vector or the track parameters to the ones represented by the measurement. This is done by so-called measurement mapping functions. Measured parameters can hereby exceed the simple localisation of a track on a surface, since track segments can also be included into a track fit as a measurement (and usually include already directional information).

In the tracking EDM, the full parameter vector is expressed through a five-dimensional `HepVector` object from the CLHEP maths library. The `HepVector` class is not restricted to any dimension and

in a naive approach the measurement representation could simply use the same class to represent the (mostly lower-dimensional) measured parameters. However, for the client algorithm it is not obvious which of the five parameters — or what subset of the five parameters — is stored in the `HepVector` object. To save both disk space and the consumption of physical memory it is however desired to store only the obtained parameters of the measurement in the appropriate data class. Clearly this can be handled by convention, but in a generic model, where track measurements can be used without knowing the underlying detector technology, this is far from trivial. The solution to this problem has been the introduction of the `LocalParameters` class that extends the `HepVector` by an identification schema. The measurements are hereby expressed by a key which is the `integer` representation of a binary number that puts 0 for non-measured and 1 for measured.

A pixel measurement that localises both coordinates on a silicon module is then represented as a binary number 11000. For technical aspects that are based on the fact the the overwhelming majority of measurements represent purely local coordinates, the digits of the binary number are flipped, and the identification key is calculated as the transformed `interger` representation, yielding in the given example:

```
int(0b00011) = 3
```

Given the identification through the integer key it is possible to check the measured parameters contained by the `LocalParameters` class. Some examples of the usage of the `LocalParameters` class will be shown in the following.

```
// retrieve the local parameters from the measurement
const Trk::LocalParameters& measPars = measurement->localParameters();

// check whether it measured the local Y coordinate
bool measLocY = measPars.contains(Trk::locY);
```

The argument given through the method signature of the `contains(...)` method is the the same `enumeration` type as used for accessing the parameters from the track parameterisation, the so-called `ParamDefs`. The `ParamDefs` are also used to define the parameter(s) to be put into the `LocalParameters` class at construction level.

```
// created a defined parameters - it is a segment that defines x/phi
Trk::DefinedParameter measuredX(0.452, Trk::locX);
Trk::DefinedParameter measuredPhi(1.452, Trk::phi);

// create a vector of defined parameters
std::vector<Trk::DefinedParameter> defParameters;
defParameters.push_back(measuredX);
defParameters.push_back(measuredPhi);

// finally create the LocalParameters
Trk::LocalParameters segParameters(defParameters);
```

For the most common usecases there exist dedicated constructors for the convenience of the clients. The `LocalParameters` class also provides the projection matrices that correspond to the measurement mapping functions, which are in the ATLAS tracking EDM simply realised as projections matrices. A static member of type `ProjectionMatricesSet` that holds the 31 possible matrices is therefore registered to the `LocalParameters`. A residual calculation as given in Eq. 5 can then be calculated as

```
// get the track parameters vector - always 5-dimensional
const HepVector& parameters = trackParameters->parameters();

// retrieve the local parameters from the measurement - n-dimensional ( n <= 5 )
const Trk::LocalParameters& measPars = measurement->localParameters();
```

```
// build the n-dimensional residuum vector
HepVector residuum = measPars - measPars.reductionMatrix()*parameters;
```

It is worth noticing that the last line hereby is only possible since the `LocalParameters` object extends the `HepVector` and thus all operators are defined accordingly. The transposed measurement mapping matrices $\mathbf{H}_i^T$ are also accessible for the convenience of the user.

A.3 Segment Authors

The adapted `Segment` class now incorporates an authorship schema for later identification of the algorithm used for the segment creation. The authorship is hereby registered by an `enumeration` object; the current list of possible segment authors can be found below.

```
enum Author {
      AuthorUnknown = 0,
      MooMdtSegmentMakerTool = 1,
      MooCscSegmentMakerTool = 2,
      Muonboy = 3,
      DCMathSegmentMaker = 4,
      MDT_DHoughSegmentMakerTool = 5,
      CSC_DHoughSegmentMakerTool = 6,
      Csc2dSegmentMaker = 7,
      Csc4dSegmentMaker = 8,
      TRT_SegmentMaker = 9,
      NumberOfAuthors = 10
};
```

References

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

[2] Athena homepage, *http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/ architecture.*

[3] CLHEP homepage, *http://cern.ch/clhep*

[4] A. Salzburger, M. Wolter, S. Todorova, *The ATLAS Reconstruction Geometry Description*, ATLAS Communication to be published, ATLAS-COM-SOFT-2007-009, 2007.

[5] ATLANTIS homepage, *http://cern.ch/atlantis*

[6] A. Salzburger, *The ATLAS Extrapolation package*,ATLAS Communication to be published, ATLAS-COM-SOFT-2007-008, 2007.

[7] V. Boisvert et al, *Final Report of the ATLAS Reconstruction Task Force*, ATLAS Note, ATL-SOFT-2003-010, 2003.

[8] ATLAS software CVS reporsitory, online CVSView, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*

When in doubt,
predict that the present trend
will continue.

Merkin's Maxim

Chapter 7

Track Extrapolation

7.1 Introduction

The role of the track extrapolation in track fitting has been briefly discussed in Chap. 4 of this document. Condensed in Eq. (4.6) it describes the system evolution of track states when following the trajectory through the detector. However, this application does not only appear in track fitting; the transport of track parameters is essential for many additional stages of the event reconstruction: prior to the track fitting, propagation methods are often used in pattern recognition define roads for hit searching; in vertex fitting, the track representation has to be evaluated iteratively w.r.t. the current vertex estimation. Furthermore, track-cluster matching in particle identification algorithms require the extrapolation of the estimated track to the calorimeter entrance. The usage of the ATLAS extrapolation engine even extends the given examples. It has been deployed in coherence with the the new reconstruction geometry model, see Chap. 5, that allows a predictive navigation and serves therefore the hole search algorithm as well as the heart of a new fast track simulation engine, which will be presented in Chap. 9.

The presented track extrapolation engine has become a standard component of several reconstruction and analysis modules of the ATLAS offline software. The importance of the track extrapolation package for the ATLAS reconstruction software can be probably best shown when giving actual call numbers to the extrapolation or propagation engines, a table that summarises the calls for only 10 hard scattering events can be found in the following section.

Many more considerations than the pure mathematical solution of the underlying problem had to be taken into account for the realisation of the track extrapolation engine in context of the ATLAS reconstruction. Accuracy, reliability and timing are just few mainly technical challenges that affect the implementation strategy, while simple human aspects are often forgotten that should also be considered for any large scale software project: since the extrapolation engine serves many users on different levels of interactions (there is the usage in core reconstruction application on the one hand, or even user analysis code on the other hand), it is necessary to provide an easy-to-use user interface that does not require dedicated expert knowledge. The use of the common event data model, presented in Chap. 6, is hereby essential, but also the configuration has to be accessible and understandable for the single user.

The following section presents the ATLAS track extrapolation engine that has been developed during the recent restructuring of the ATLAS track reconstruction software.

7.2 The ATLAS Track Extrapolation Package

ATL-SOFT-PUB-2007-005, 43 p.

The ATLAS Track Extrapolation Package

A. Salzburger*
Leopold Franzens Universität Innsbruck, Austria & CERN

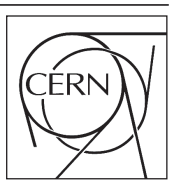
December 11, 2007

Abstract

The extrapolation of track parameters and their associated covariances to destination surfaces of different types is a very frequent process in the event reconstruction of high energy physics experiments. This is amongst other reasons due to the fact that most track and vertex fitting techniques are based on the first and second momentum of the underlying probability density distribution. The correct stochastic or deterministic treatment of interactions with the traversed detector material is hereby crucial for high quality track reconstruction throughout the entire momentum range of final state particles that are produced in high energy physics collision experiments. This document presents the main concepts, the algorithms and the implementation of the newly developed, powerful ATLAS track extrapolation engine. It also emphasises on validation procedures, timing measurements and the integration into the ATLAS offline reconstruction software.

ATL-SOFT-PUB-2007-005
13 December 2007

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*



*E-mail: Andreas.Salzburger@cern.ch

1 Introduction

The transport of track parameters (i.e. the representation of a track with respect to a given surface) and their associated covariances is a very frequent process in track reconstruction. Most progressive fitting techniques such as the Kalman filter formalism [1] rely on the prediction of the gathered track information on the successive measurement surface (a simplified illustration of a track extrapolation in a typical Kalman filter step can be seen in Fig. 1). In global fitting techniques, on the other hand, the prediction of the track depending on the initial parameters (i.e. the fitted parameters) enters the global χ^2 function to be minimized. For both, global and sequential track fitting algorithms, the correct treatment of effects caused by the interaction of the particle with traversed detector material is essential; in the least squares fit the uncertainties due to material interactions regulate the contribution of the fitted scattering angle to the global χ^2 function. In most sequential fitting techniques the uncertainties of the momentum direction and magnitude caused by interactions with the detector material are directly applied as deterministic energy loss and additional contributions to the covariance matrix during the extrapolation process.

Track extrapolation is furthermore necessary in vertex fitting, where the expression of the track with respect to the estimated vertex position has to be evaluated iteratively towards convergence. Moreover, in pattern recognition the seeded prediction of a track may be used for trajectory building and hit finding. Finally, the track parameters representation on a destination surface is needed for combined reconstruction to enhance the matching of tracking information and calorimeter clusters on the one hand, and the combination of tracks and track segments from different tracking devices on the other hand.

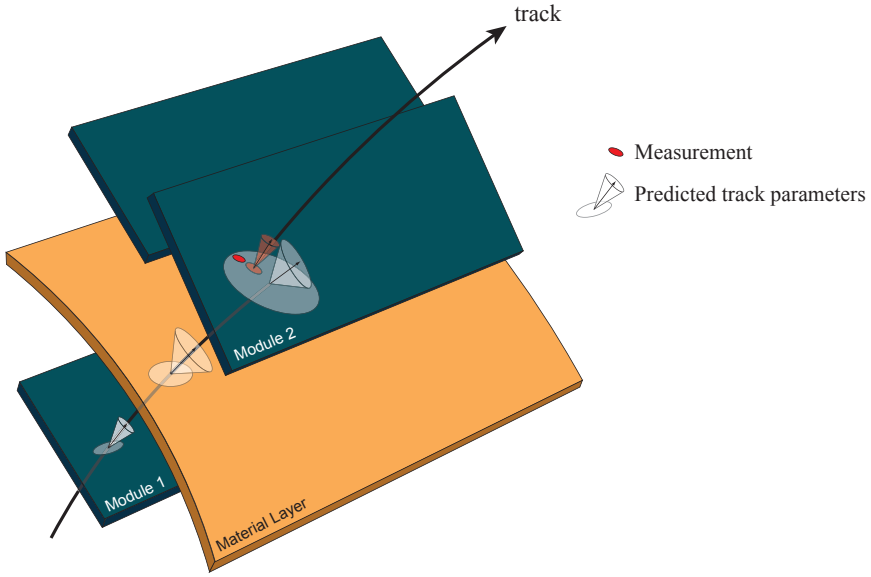


Figure 1: Simplified illustration of a typical extrapolation process within a Kalman filter step. The track representation on the detector module 1 is propagated onto the next measurement surface, which results in the track prediction on module 2. The traversing of the material layer between the two modules causes an increase of the track direction uncertainties and thus — by correlation — an increased uncertainty of the predicted track parameters. In the Kalman filter formalism, the weighted mean between prediction and associated measurement build the updated measurement which builds the start point for the next filter step; this leads to the illustrated non-continuous track model.

The ATLAS experiment puts stringent requirements to the track reconstruction software: for the track parameter propagation, in particular, the highly inhomogeneous magnetic field setup — a solenoidal field with a central magnitude of 2 Tesla in the *Inner Detector* (ID) and a toroidal magnetic field with a peak strength higher than 4 Tesla in the *Muon Spectrometer* (MS) — and the complex material distribution of the sub-detectors must be dealt with correctly to achieve a satisfactory tracking resolution. Additionally, the expected high track occupancy in the Inner Detector and the resulting high number of track candidates impose combinatorial and timing constraints for track finding and fitting

algorithms. Since the CPU time consumption of the track reconstruction chain has to be minimised to fit with the computing budget of the experiment, the algorithmic solution of a powerful but fast track extrapolation engine is a challenging task.

During the redesign of the ATLAS offline reconstruction software that has been invoked by the *Final Report of the Reconstruction Task Force* (RTF) [2], a new extrapolation package has been designed and developed. First big scale tests of the new extrapolation engine have been performed during event reconstruction of the ATLAS *Combined Test Beam* 2004 (CTB2004), the first commissioning runs using cosmic rays since 2005 and the reconstruction of Monte Carlo simulated data. The extrapolation package is fully integrated in the object oriented C++ based ATLAS software framework ATHENA [3] and has been developed respecting ATLAS coding standards [4]. A description of the main components of the ATHENA framework, the `Service`, the `Algorithm` and the `AlgTool` interfaces, can be found in [5]. This document is based on the ATLAS software release 13.0.10, while extensions that have been applied after this release are indicated within the context.

1.1 Design Principles and Document Structure

The transportation of a track representation to a destination surface can be divided into three conceptionally different tasks that have been independently realised as single components of the ATLAS extrapolation package. For convenience, they will also build the guideline of this document:

- the **propagation** process describes the mathematical transport of the track parameters and associated covariances to the target surface. The propagation module is defined through a dedicated `IPropagator` interface, that is implemented through various different propagation `AlgTool` classes. A more detailed description of the propagation process can be found in Sec. 2.

- the **navigation** relates the trajectory to the various entities of the reconstruction geometry. This is necessary to find the appropriate material description, to ensure the access to the (configured) magnetic field map, or even to find the detector volume that contains the destination surface. A dedicated `INavigator` interface defines these navigation methods, see Sec. 3.

- the **integration of material effects** according to the traversed detector material that is provided by the navigation marks the third column of the extrapolation package. It is enhanced through various different `AlgTool` classes and further described in Sec. 4.

The complete process including all three tasks will in the following be referred to as *track extrapolation*. It is concentrated in a single `AlgTool` implementation, the `Extrapolator`; a schematic illustration of the three column structure that is incorporated by the extrapolation engine — represented through the various interface definitions — can be seen in Fig. 2. A further description of the user interface and the successive steering of propagation, navigation and material effects integration will be described in Sec. 5. Section 6 will give performance numbers in both timing and accuracy for typical track reconstruction applications. The Appendix explains used conventions and typesetting formats, and gives an exhaustive list of formulas used within the various track propagation techniques.

1.2 Component Pattern and Data Factory Design

The TrkExtrapolation repository follows a strict component pattern design, i.e. abstract interface classes are concentrated in dedicated interface packages, while concrete implementations of algorithmic code and data classes are separated in different component and installed libraries, respectively. This allows dynamic loading of libraries at job execution level, and is thus the key to the high flexibility. The single components of the track extrapolation package are, in general, realised as factories, i.e. the performed operations lead to newly created objects. In C++ terms this is done by returning pointers to objects that are dynamically created using the `new` operator, while the return of a pointer with value 0 indicates that the operation could not be performed. The ATLAS tracking data model, briefly presented in the following section, does not foresee invalid objects, such as e.g. a representation of a failed extrapolation process.

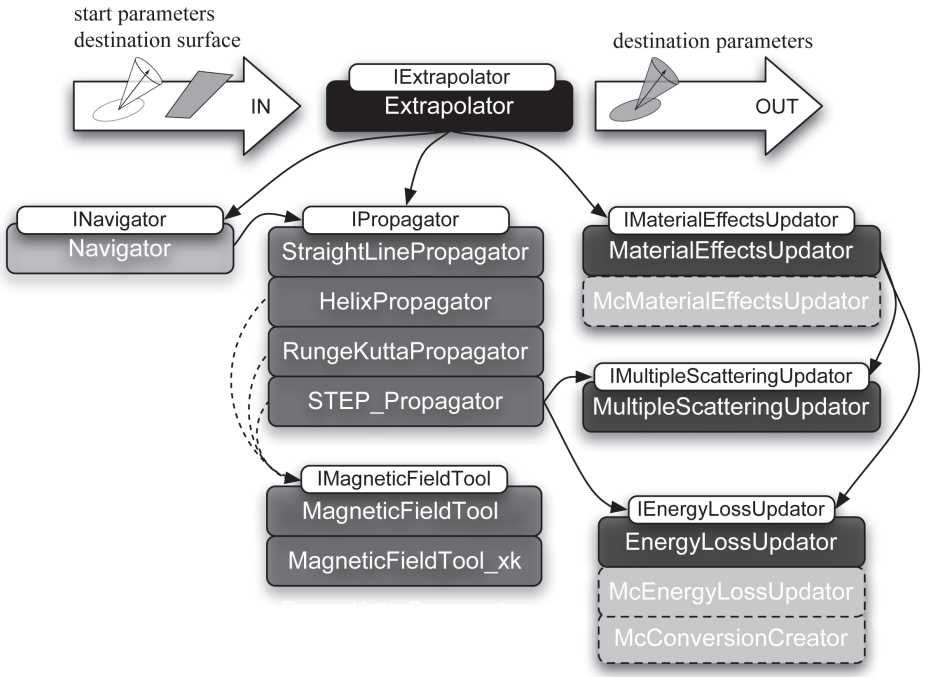


Figure 2: The ATLAS extrapolation engine illustrated as a task diagram. The various modules of propagation, navigation and material effects integration are identified through dedicated interfaces. The main concrete implementations are also shown in this diagram, brighter colored boxes framed by a dashed line represent hereby the exchanged modules for the use of the extrapolation engine in the fast track simulation application.

1.3 Tracking Event Data Model and `TrackingGeometry`

The extrapolation package is based on the common tracking event data model (EDM) and the ATLAS reconstruction geometry. In particular, the main EDM classes extending the `TrackParameters` base class and the geometry classes `Surface`, `TrackingGeometry`, `TrackingVolume` and `Layer` are essential ingredients of the track extrapolation process. The full description of these classes would go far beyond the scope of this document the interested reader is, however, encouraged to read further in [6] and [7], respectively, for an exhaustive description of these modules.

Track Parameterisation A track can be parameterised with respect to a surface in many different ways. If a particle propagates through magnetic field, however, a minimal set of five parameters is necessary to provide a complete and unambiguous parameterisation with respect to any given detector surface. In the following, such a representation of the trajectory will be referred to as `TrackParameters`. In ATLAS, dedicated `TrackParameters` objects exist for every defined surface type and provide the following track parameterisation[1]

$$\mathbf{x} = (l_1, l_2, \phi, \theta, q/p)^T \tag{1}$$

when l_1 and l_2 denote the local coordinates on the given surface (and thus depend on the surface type), ϕ and θ are the azimuth and polar angle, respectively, and q/p is the inverse momentum multiplied by the charge q. For the widely used perigee representation that is often used to express the track with respect to the nominal beam axis the local parameters l_1 and l_2 correspond to the transverse and longitudinal impact parameter d_0 and z_0, respectively. Figure 3 shows an illustration of the perigee representation using ATLAS conventions.

[1] Many experiments in the past used a — for many analysis aspects — more handy parameterisation of a helix with a curvature-optimised momentum representation q/p_T and the polar direction represented as $\cot\theta$. Since for ATLAS a general representation had to be chosen that is valid throughout the detector and thus within different magnetic field setups, a helical representation that is bound to one magnetic frame is not optimal, as the transverse momentum does not remain a constant of motion.

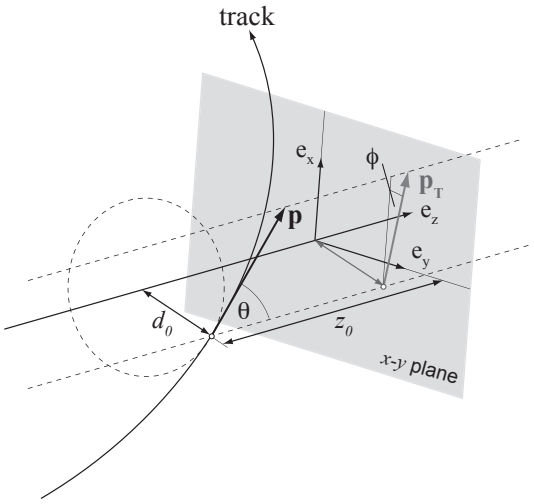


Figure 3: The perigee representation expressed in the ATLAS track parameterisation. The local expression of the point of closest approach is given by the signed transverse impact parameter d_0 and the longitudinal impact parameter z_0. The momentum direction is expressed in global coordinates using the azimuthal angle ϕ that is defined in the projected $x-y$ plane and the polar angle θ, which is measured with respect to the global z axis.

Neutral Parameters Recently, the ATLAS tracking EDM has been extended to deploy a dedicated schema for neutral particle representations [8]. The fifth parameter of the representation as given in Eq. (1) is hereby modified to represent $1/q$, omitting the charge definition. Charged and neutral trajectory representations are realised through the same templated class objects to avoid code duplication, while keeping the type diversity to prevent misinterpretations to happen during the reconstruction flow. The extrapolation package and propagation tools have been adapted to cope with both charged and neutral types, but the ATLAS `Track` class remains restricted to charged trajectories[2]. Neutral parameters are only transported along a straight line to the provided target surface. Material effects are not taken into account and thus the navigation process is not necessary in this context. This documents concentrates therefore on the extrapolation process of charged track representations and will only briefly mention the particularities for neutral parameterisation in the various different modules.

2 Propagation

The mathematical propagation of track parameters to a destination surface is — when omitting energy loss and multiple scattering effects — determined by the starting parameters and the traversed magnetic field. A homogenous magnetic field setup (no field or constant field value and direction) allows to use an underlying parametric track model for the propagation. Many propagation processes can then be solved purely analytically to find the intersection of the track with the destination surface and even for the transported covariances. However, the highly inhomogeneous magnetic field of the ATLAS detector setup requires tracking of particles by numerical methods. Figure 4 shows the magnetic field of the ATLAS detector in an $r-z$ projection for both, the Inner Detector in detail, and the Muon Spectrometer.

The variety of the different propagation techniques is enhanced by different implementations of a common abstract `AlgTool` interface, the `IPropagator`. The interface for propagator `AlgTool` classes is kept very simple; it reflects the pure principle of the task: an input `TrackParameters` object, a destination surface, magnetic field properties and a boolean for the surface bound handling is passed through the method signature, while on the other hand the propagated parameters are returned as the method value. Returning a pointer to a new object puts the responsibility of memory cleanup onto the client algorithm, but complies fully with the factory pattern design described in Sec. 1.2.

The following main interface methods are defined for the `IPropagator` interface:

- The `propagate()` method shall be used in cases when the track parameters to be transported are likely to carry a covariance matrix and the client algorithm relies on the transported error description as well. If the input parameters do not have associated errors, only the parameters are transported to the destination surface.
- To save CPU time, the `propagateParameters()` that only performs the transport of the pa-

[2]This is because neutral particles are not subject of tracking in the classical terms of track finding and track fitting.

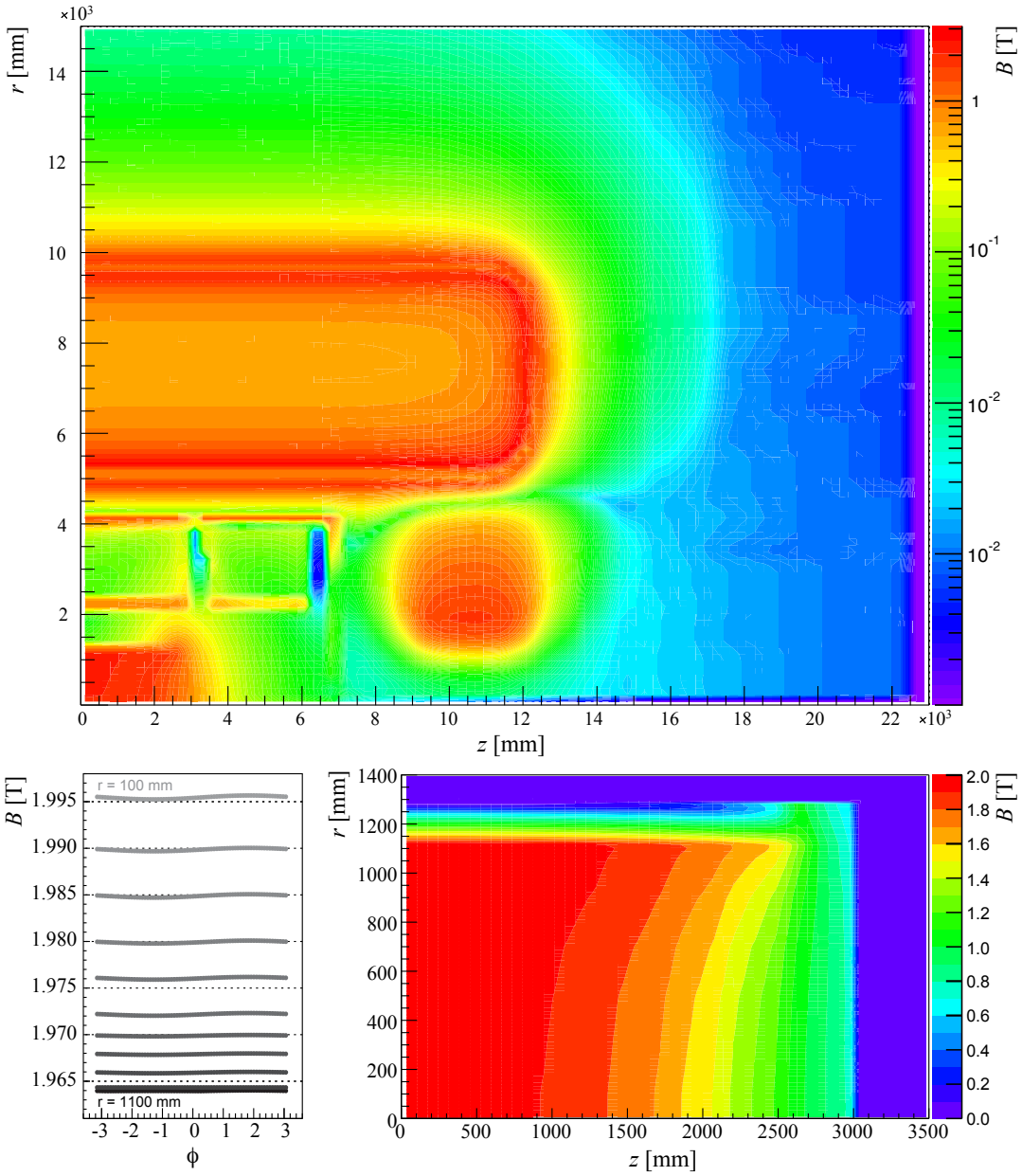


Figure 4: The realistic magnetic field in the $r-z$ plane for the entire ATLAS detector. The upper plot shows the magnetic field strength in the $r-z$ plane at an azimuthal angle of $\phi = \pi/8$ which lies within one Muon System toroid structure. The plots at the bottom focusses on the magnetic field of the Inner Detector as described by the ATLAS-CSC-01-02-00 layout. The first plot at the bottom shows the ϕ-dependency of the magnetic field at different radii in steps of 100 millimeter at $z = 0$: the homogeneity of the field in the ID is broken in radial and azimuthal direction even in the very central part of the solenoid. The second plot shows the magnitude of the magnetic field shown within a quarter of the Inner Detector.

rameters and omits the transport of the associated covariances can be chosen. This is optimised for situations where the transported error represented at the destination surface is not needed.

- The `globalPositions()` method is designed to fill a list with 3D points along the track in intervals of a given step length and confined within a given volume. It is mainly performed during the road building process of the pattern recognition stage.

- The `validationAction()` enables to call event- or track-based validation directives from outside (e.g. such as parameter resetting or the filling of validation information into appropriate output

containers). This interface method is not particular to the `IPropagator` interface. In fact, most of the used `AlgTool` classes in the TrkExtrapolation realm implement a method of this type and purpose.

Four different propagators have been implemented and are located in different packages of the ATLAS software repository, the `StraightLinePropagator` and `HelixPropagator` incorporate a straight line and respectively helical track model, whereas the `RungeKuttaPropagator` and `STEP_Propagator` are based on the numerical Runge-Kutta-Nystrøm [9] integration technique to evaluate the field integral.

2.1 Intersection with Surfaces and Direction Instructions

The intersection of a track with a surface can be expressed in both global and local coordinates. In terms of track parameters propagation also the direction of the track at the destination surface is needed for a complete track parameterisation: the two local coordinates of the surface intersection and the momentum of the track at the destination surface build the five parameters of the propagated `TrackParameters` expression. The local intersection point — together with the surface constraint — establish the global representation of the trajectory intersection with the target surface; a detailed discussion of the `TrackParameters` object and the used local frame definitions can be found in [6].

The propagation can be performed with a direction instruction, steered by an `enumeration` object; propagations `alongMomentum`, `oppositeMomentum` and `anyDirection` can be performed. The direction does not only determine the intersection solution to be chosen — in case that multiple solutions exist — but also handles the material effects integration during the extrapolation process. For planar surfaces, i.e. the `PlaneSurface` and `DiscSurface` in the ATLAS reconstruction geometry model, there is, in general, one unique solution[3] for the intersection. The *intersection* with a straight line (in the ATLAS reconstruction geometry described by either the `StraightLineSurface` or the `PerigeeSurface`) is defined as the transverse closest approach to the line. The local expression of this point of closest approach is done by a signed transverse impact parameter and the longitudinal impact parameter with respect to the line frame. For surfaces of type `CylinderSurface` typically one or two intersections exist for a given start parameter set. If the propagation direction excludes one of the solutions, obviously the other solution is taken, if both intersections are compatible with the provided directive (or the directive has been chosen to be `anyDirection`), the closer intersection is taken by convention.

2.1.1 The Measurement Frame

The transport of the covariance matrix to a given surface requires the definition of the so-called measurement frame, i.e. the coordinate system attached to the destination surface in which the error on the local parameters is properly defined. For planar surfaces this definition is trivial: the measurement frame is hereby identical with the local surface frame, a cartesian frame for the `PlaneSurface` and a polar frame for the `DiscSurface`. For cylindrical surfaces, the measurement frame is defined — by convention — as the tangential plane to the intersection point with the cylinder. Since in ATLAS no measurement is given on a cylindrical surface (these are mostly needed for the navigation through the volumes of the reconstruction geometry), this choice saves unnecessary conversions from cartesian to cylindrical coordinates (and *vice versa*).

For track expressions with respect to a line (`Perigee` or `AtaStraightLine` definitions) the measurement frame can only be constructed once the point of closest approach is found: it is characterised by the transverse *drift* direction $\mathbf{d}_D$ as the first local axis and the line direction $\mathbf{d}_L$ as the second, perpendicular axis. Figure 5 shows an illustration of the measurement frame for a closest approach to a straight line.

2.2 Analytical Track Models: `StraightLinePropagator` and `HelixPropagator`

Propagations along a straight line can be solved analytically for all surface types. This transport type is applied for particles of zero charge or in absence of a magnetic field. Furthermore, they may

[3] In the ATLAS experiment, the geometrical dimensions of the destination surfaces are in general small compared to the curvature of the track due to the bending of the magnetic field. Hence, multiple intersections with planar surfaces are very rare.

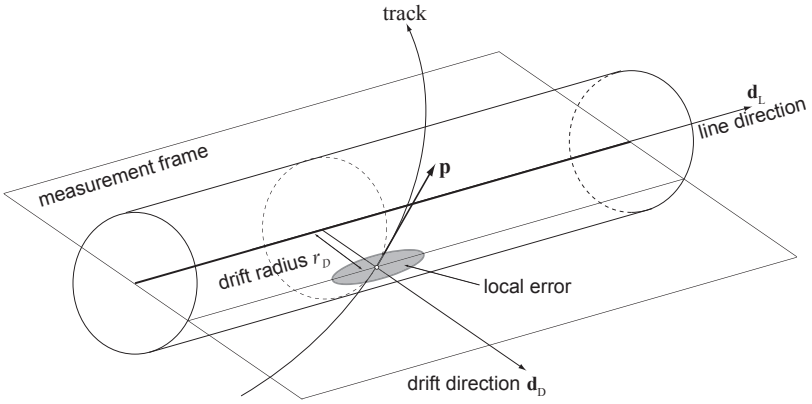


Figure 5: The measurement frame for the closest approach of a track to a nominal line surface. The error on the local coordinates is given in a two-dimensional cartesian system, defined through the drift direction $\mathbf{d}_D$ and the line direction $\mathbf{d}_L$, respectively.

be used for crude estimates, in particular in the non-bending plane. The `StraightLinePropagator`, located in the TrkExSlPropagator package, provides this functionality. It has been extensively used during the reconstruction of events taken during the ATLAS CTB2004 and data taken from the first commissioning runs using cosmic rays, where several periods of data taking without magnetic field have taken place.

The `HelixPropagator` is suitable for propagations of parameters in a homogeneous magnetic field, a very rare or even non existing situation in the ATLAS reconstruction. However, for the CTB2004 Monte Carlo simulation has been done assuming a constant magnetic field and for the new *Fast ATLAS Track Simulation* (FATRAS) [10], the constant magnetic field together with the `HelixPropagator` may be a sufficient simplification for various applications. When using a helical track model the analytical solution for track intersections with many surfaces can only be performed if the surfaces are *aligned* with the guiding center of the helix. Assuming perfect alignment of the ATLAS detector, many surfaces fulfill this requirement. For arbitrarily oriented surfaces, a numerical method to solve the equation has to be used, which is solved through a standard Newton-Rhapson approach. Due to its rare usage in the reconstruction this method has been, however, poorly validated and clearly lacks both accuracy and stability for production usage.

The formulas used for the calculations of the surface intersections for both the `StraightLinePropagator` and the `HelixPropagator` can be found in the Appendix, Sec. A.3 and Sec. A.4.

2.2.1 Analytical Error Propagation along the curvilinear Frame

Since both, the straight line and the helical propagation incorporate an underlying analytical track model, the transport of the covariance matrix can be performed analytically as well. This is done using the so-called *curvilinear* frame that can be defined at every point of the track and has been widely used in the past in high energy physics experiments [11]. The intersection solution with a target surface is, in general, a non-linear function of the starting parameters and it is thus not obvious how to propagate the associated error. A solution to this is to linearise the problem by taking the first order of a Taylor series expansion for the error propagation. The derivatives of the transported parameters with respect to the start parameters build hereby the Jacobian matrix that yield the transformation of the covariance matrix. Within the curvilinear frame, the transport Jacobian matrix $\mathbf{J}_{\mu,\nu}$ becomes an analytical expression with the single parameter s, which denotes the propagation length along the line or helix. Given a global point $\mathbf{m}$ and the momentum $\mathbf{p}$ (defined at $\mathbf{m}$), the right handed curvilinear frame $\mathbf{u}_i = (\mathbf{e}_u\,, \mathbf{e}_v\,, \mathbf{e}_t\,)$ is defined by

$$\mathbf{e}_t = \frac{\mathbf{p}}{|\mathbf{p}|}$$

$$\mathbf{e}_u = \frac{\mathbf{e}_z \times \mathbf{e}_t}{|\mathbf{e}_z \times \mathbf{e}_t|}$$
$$\mathbf{e}_v = \mathbf{e}_t \times \mathbf{e}_u \,, \tag{2}$$

where $\mathbf{e}_z$ denotes the global z axis[4]. Figure 6 shows the curvilinear frame construction for a measurement along a straight line.

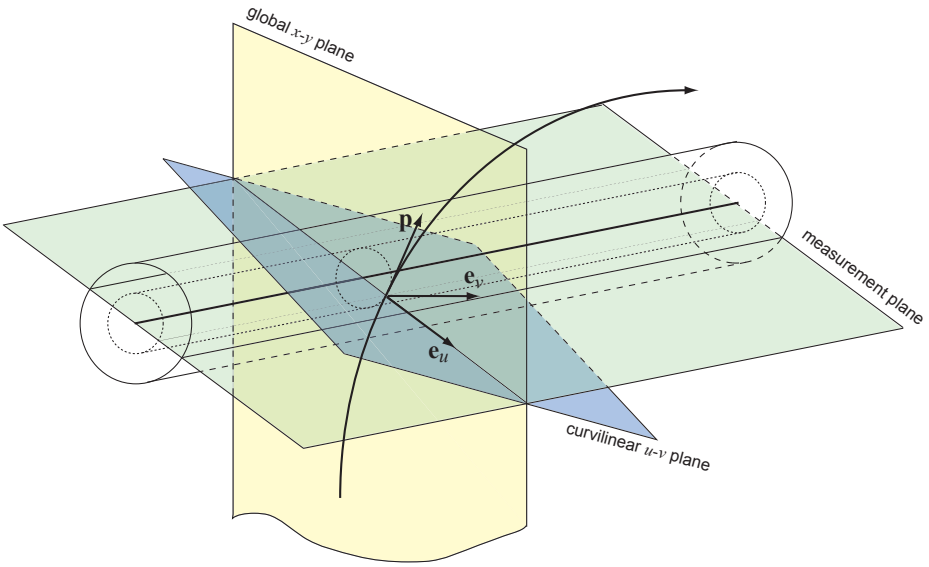


Figure 6: The curvilinear frame $U = (\mathbf{e}_u \,, \mathbf{e}_v \,, \mathbf{e}_t \,)$ for a measurement along a straight line. The normalised momentum vector $\mathbf{e}_t = \frac{\mathbf{p}}{|\mathbf{p}|}$ builds the z direction of the curvilinear frame. Its vector product with the nominal global z axis creates the second base of the frame, which lies in the global $x-y$ plane. Finally, the third base vector is chosen in such a way that a right handed coordinate system is well defined.

Let $\mathbf{C}_{li}$ denote the covariance matrix of the initial parameters in the local coordinates of the starting frame, and $\mathbf{J}_{li,ci}$ the Jacobian matrix describing the parameter transformation from the initial local frame li to the initial curvilinear frame ci. The initial covariance matrix expressed in the curvilinear frame $\mathbf{C}_{ci}$ of the starting parameters can then be written as

$$\mathbf{C}_{ci} = \mathbf{J}^{ci,li} \cdot \mathbf{C}_{li} \cdot \mathbf{J}_{li,ci}, \tag{3}$$

with $\mathbf{J}^{ci,li} = \mathbf{J}^T_{li,ci}$ being the transposed Jacobian of the *local-to-curvilinear* transformation.
The transport Jacobian matrix $\mathbf{J}_{ci,cf}$ between the curvilinear frame at start point i and destination point f (i.e. track intersection with the target surface) becomes only dependent on the rotations of the initial and final curvilinear frames with respect to the global frame and the propagation length s. The calculation of this Jacobian such as the coefficients for local to curvilinear frame transformations (and *vice versa*) can be found in the Appendix, Sec. A.5, of this document. The transported covariance matrix at the destination surface after back transformation to the local frame using an appropriate Jacobian $\mathbf{J}_{cf,lf}$ is then given by

$$\mathbf{C}_{lf} = \mathbf{J}^{lf,cf} \cdot (\mathbf{J}^{cf,ci} \cdot \mathbf{C}_{ci} \cdot \mathbf{J}_{ci,cf}) \cdot \mathbf{J}_{cf,lf}, \tag{4}$$

with $\mathbf{J}^{cf,ci} \cdot \mathbf{C}_{ci} \cdot \mathbf{J}_{ci,cf}$ describing the transport along the track and within the curvilinear frame.

Neutral Track Representations The adaption of the `StraightLinePropagator` to neutral parameters was only challenging from a technical point of view, where the same mathematical steps have to be performed on different input classes that share only a high level common base class. This is

[4]In principle, any unique global reference axis can be chosen instead of $\mathbf{e}_z$. The choice of the global z axis in this scope is not arbitrary though, but reflects the detector geometry. It guarantees numerical stability since track directions along $\mathbf{e}_z$ are not expected.

solved by templating the private propagation methods to the different surfaces by the actual charged or neutral parameters type. The `HelixPropagator` simply uses the `StraightLinePropagator` for any parameters transport of neutral parameters or in case of a no-field environment. The transport of the neutral parameters classes is fully supported with ATLAS release 13.1.0.

2.3 Numerical propagation: the `RungeKuttaPropagator` and `STEP_Propagator`

Most of the track parameter propagations to be performed in the ATLAS event reconstruction are within a highly inhomogeneous magnetic field where a global track model can not be used for solving the transport equations. Hence, a fast numerical solution for calculating the intersection of the trajectory with the destination surface is needed. In ATLAS this is realised by two implementations of the `IPropagator` interface, the `RungeKuttaPropagator` and `STEP_Propagator`. Both rely on a fourth order Runge-Kutta-Nystrøm formalism with an integrated adaptive step estimation. The main difference between the two realisations is that the `STEP_Propagator` includes energy loss in the equation of motion and applies corrections to the covariance matrices continuously during the parameter transport along the track. It is designed for the description of a particle that traverses a dense block of material, while the `RungeKuttaPropagator` complies with the classical model of point-like material update on detector layers that is carried out by dedicated `AlgTool` classes in the ATLAS track extrapolation engine. Both `IPropagator` implementations perform the propagation in global coordinates and use common Jacobian matrices for the transformations between the global frame and local surface-attached coordinate systems[5]. The equation of motion of a charged particle with momentum p and mass m through a magnetic field $\mathbf{B}(\mathbf{r})$ can be expressed in many different ways that mostly differ through the parameterisation and choice of the free parameter. For collider experiments a helix-based parameterisation along the arc length s is a good choice since it is not restricting nor favoring any specific particle direction[6]. The equation of motion of a particle with charge q, defined by the Lorentz force, can then — when omitting multiple scattering and energy loss effects — be written as the second order differential equation

$$\frac{d^2\mathbf{r}}{ds^2} = \frac{q}{p}\left[\frac{d\mathbf{r}}{ds} \times \mathbf{B}(\mathbf{r})\right], \tag{5}$$

and, when including an energy loss function $g(p, \mathbf{r})$, as

$$\frac{d^2\mathbf{r}}{ds^2} = \frac{q}{p}\left[\frac{d\mathbf{r}}{ds} \times \mathbf{B}(\mathbf{r})\right] + g(p, \mathbf{r})\frac{d\mathbf{r}}{ds}. \tag{6}$$

Equation (5) and Eq. (6) are the fundamental transport equation used by the `RungeKuttaPropagator` and `STEP_Propagator`, respectively. The calculations are in both cases performed using the Runge-Kutta-Nystrøm method, which is well suited to solve second order differential equations. The basic principle of the Runge-Kutta method can be found in may textbooks [9], an exhaustive review of the used Runge-Kutta-Nystrøm method and the description of error matrix transport (carried out by the Bugge-Myrheim method) for both propagators of the ATLAS track reconstruction is in addition presented in [12].

Both the `RungeKuttaPropagator` and the `STEP_Propagator` stop the numerical iteration when the distance to the surface drops below a certain cut value. For the last step starting at the position $\mathbf{r}_{f-1}$, a simple Taylor expansion to second order is used:

$$\mathbf{r}_{final} = \mathbf{r}_{f-1} + h\frac{d\mathbf{r}}{ds}\,|_{\mathbf{r}_{f-1}} + \frac{1}{2}h^2\frac{d^2\mathbf{r}}{ds^2}|_{\mathbf{r}_{f-1}}, \tag{7}$$

with h denoting the distance to the destination surface at the approach point $f-1$.

Straight and Helical Track Model The `RungeKuttaPropagator` and `STEP_Propagator` are clearly the most flexible propagation techniques in the ATLAS track reconstruction software. It is inert to the Runge-Kutta formalism that in case of a homogenous magnetic field setup, the propagation is carried

[5] The common data classes are located in the shared TrkExUtils package.

[6] In fix target experiments, however, a different choice representing the main particle direction may be taken.

out in one single step, complying a helical model for constant field or a straight line propagation when no magnetic field is present, respectively. This eases the support of neutral track parameters that is completely integrated with ATLAS software release 13.1.0.

3 Navigation

The navigation builds the binding link between propagation and material effects integration in the realisation of the ATLAS track extrapolation engine. During a single extrapolation process, the navigation gathers the information about the traversed material from the reconstruction geometry, the so-called `TrackingGeometry`. The internal connectivity of the `TrackingGeometry` is hereby used to save CPU time expensive global search operations.

Based on the morphology of the ATLAS reconstruction geometry, three different navigation streams can be followed throughout the extrapolation process: navigation within a full static and connective setup, navigation within a dynamic setup and navigation within a detached or unordered setup. The concepts and realisation of the different navigation strategies will be described in the following sections.

Initialisation of the Navigation The first step of the navigation procedure is to determine the starting and destination volume for the track extrapolation. This is done following a three-step procedure that optimises the time spent in the search of the start and end configuration:

- **recall:** in many fitting applications it is necessary to extrapolate from one measurement surface to the successive one along a track. The destination configuration of the previous step is hereby often identical to the start configuration of the next extrapolation process. The `Extrapolator AlgTool` *remembers* therefore the end configuration of the extrapolation process in dedicated private member variables and compares as a first step the starting parameters with the solution of the previous extrapolation call.

- **association:** the ATLAS `TrackingGeometry` provides associations between the objects in the different levels of the geometry hierarchy, i.e. `Surface` objects often point to embedding `Layer` objects that in turn may point to enclosing volumes. This hierarchy model can not be established for all detector structures, but is realised e.g. for a big part of the Inner Detector `TrackingGeometry`.

- **global search:** the global search is the most CPU time intensive navigation step, since it involves the stepping down in the object hierarchy tree of the `TrackingGeometry`. However, it is a backup system for the recall and association attempts and is the only possibility to find the associated start and destination configuration for arbitrary surfaces that are not *a priori* known to the `TrackingGeometry`.

3.1 Navigation between Volumes

The underlying `TrackingGeometry`, characterised at first means by a fully connective set of so called `TrackingVolume` objects, is the key to the navigation process. The `TrackingVolume` class, which is in principle a container of confining boundary surfaces, combines the access to the magnetic field with the information about the traversed material (represented as `Layer` objects or through a dense volume description). The boundary surfaces that build the confinement of the volume extend the common ATLAS `Surface` class and can therefore be coherently used with the propagation `AlgTool`. Since they incorporate information about the attached volumes, the intersection with the boundary surface leads automatically to the next `TrackingVolume` that is crossed by the trajectory. This process is repeated until the destination volume is reached.

The `Navigator AlgTool`, a concrete implementation of the abstract `INavigator` base class, is called by the `Extrapolator` to perform the search for the next `TrackingVolume`. It therefore uses the provided propagator to intersect the appropriate boundary surface of the current `TrackingVolume` and find information about the attached volumes. Each `TrackingVolume` is for this purpose capable of providing an ordered list of boundary surfaces to be intersected, starting from the *best guess* that

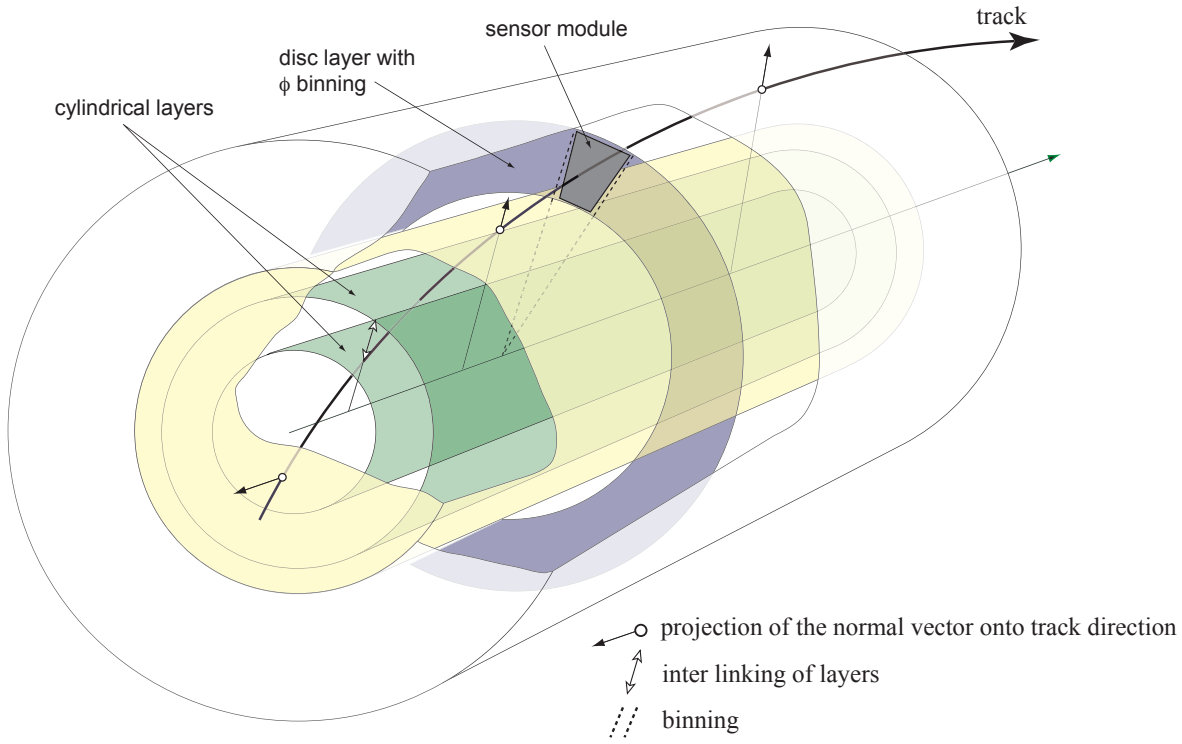


Figure 7: Illustration of the main navigation actions in the static geometry setup: volumes that are via the boundary surface mechanism attached to the currently traversed volume are found by the projection of the track with the according normal vector, layers within a given `TrackingVolume` are inter-linked by pointers and symmetric structures such as sub-surfaces on a `Layer` structure are found by binning methods.

originates from a straight line approximation of the starting parameters in the non bending plane. The provided list of boundary surfaces to be be intersected is processed until the right exit point of the surface is found, for performance reasons only the track parameters and never the associated covariance matrix is transported onto the boundary surface. Performance statistics of the navigation are discussed in further detail in Sec. 6.1.1. Figure 7 summarises the different navigation steps for a full static setup in a sample illustration.

3.2 Navigation between Layers within a `TrackingVolume`

A `TrackingVolume` can contain a static, dynamic or unordered set of `Layer` objects that are suitable for a point-like material update description. Inside a `TrackingVolume`, the navigation has to find the contained layers that are intersected by the trajectory to account for their material budget and apply material effects corrections at the correct point along the trajectory. The intersection with the layer is also needed to be able to use refined local material maps on a single layer.

3.2.1 Static Layer Setup

A static layer setup is realised when the entire description of the material within a `TrackingVolume` is available at the initialisation of the reconstruction geometry and can be ordered uniquely with a given reference direction. The entire default ATLAS ID and calorimeter geometry is described by a static layer setup. `Layer` objects in a static setup are inter-linked by C++ pointers and the navigation from one to the other can be simply done by a `nextLayer()` call, which only requires the track direction in the method signature.

3.2.2 Dynamic Layer Setup

In many configurations the material seen by a track depends very strongly on the track trajectory or a static setup is simply not possible due to the complex geometry structure. In such cases or if the material description is gathered from an outside source (e.g. a material map or material database), dynamically created layers along the trajectory can be taken. A dedicated `IDynamicLayerCreator` interface has been created for this purpose. The `IDynamicLayerCreator` mechanism is also used for global fitting techniques, where the material layers along an entire track have to be known and enter as a whole collection into the global χ^2 function. While the `IDynamicLayerCreator` is defined to provide a material description and still leaves the calculation of the resulting effects on the parameters to the client algorithm, the extrapolation package has been recently expanded with an additional, even more user-open dynamic schema that is designed to provide material effects for the track directly in terms of an energy loss to be applied and a parameter that determines the scattering correction. This second way, realised through the `IMaterialEffectsOnTrackProvider` interface, is mainly targeted at allowing custom integrations of material effects, such as energy measurements provided by the calorimeter.

3.2.3 Unordered layer setup

The `TrackingVolume` also enhances more complex layer structures that can not be easily ordered along reference directions. In this case the navigation has to follow a *trial and error* principle. Hence, the numbers of unordered layers contained by one `TrackingVolume` have to be kept low.

3.2.4 Sub-surfaces on Layers

`Layer` objects can contain ordered sub-sets of surfaces that represent active detector elements. The search for detector elements completes the full predictive navigation provided by the TrkExtrapolation engine and is useful for track predictions and holes-on-track search: the `Surface` class points back to an associated sensitive detector element and since the track trajectory is very precisely known after the track fit a list of geometrically intersected modules can be provided by following the layer and sub-surface navigation (see also Fig. 7). The comparison of the list of surfaces (and thus detector elements) stored on the track with the ones that are geometrically intersected by the given trajectory can be used as a hole definition for tracking. Additionally, the full access to intersected sensitive modules led to the development of the new FATRAS simulation. In both applications, the holes search and the fast track simulation, the local parameters of the trajectory intersection with the layer surface are hereby used for a fast look-up in the binned arrays that hold the sensitive detector surfaces.

3.3 Detached Volumes

The ATLAS reconstruction geometry also describes entities that can not be conveniently integrated into the model of a fully connective `TrackingVolume` set. This is mainly the case for complex structures such as the toroid magnet system in the Muon Spectrometer. A `DetachedTrackingVolume` class which fulfills many requirements of a standard `TrackingVolume`, but is not connected to neighboring volumes via the boundary surface schema, has therefore been created such that they can be contained by a higher level `TrackingVolume`. As a direct consequence, detached volumes require a more CPU time expensive search in the navigation stream. The entry point into a `DetachedTrackingVolume` has to be found first by propagations to the associated boundary surfaces. If a `TrackingVolume` contains more than one detached sub-volume, the sequence of traversing one after the other has to be evaluated first. The `DetachedTrackingVolume` itself contains a complete static setup, such that the fast navigation schema that relies on pre-ordering and boundary surface sharing can be used internally, once a detached object has been entered by the navigation stream.

3.4 Navigation Breaks

The navigation can break in very rare cases mainly due to the reason that provided start and destination information is not compatible, i.e. the trajectory defined by the starting parameters does not

intersect the destination surface at all or at least not in the volume that contains the destination surface. The following sources of navigation breaks have been identified and are recorded in case the `AlgTool` is configured to run in a dedicated validation mode:

- **distance increase**: the geometrical distance to the destination surface is checked progressively during the full navigation chain and a navigation break is triggered if this distance increases rapidly at one step, while small fluctuations are allowed. This usually happens if — given a set of starting parameters — it is geometrically impossible to intersect the provided destination surface.

- **oscillation**: the navigation is caught in an oscillation loop, which mostly indicates a geometry setup problem or a curling particle trajectory due to low momentum.

- **loop**: the navigation is caught in a straight loop, i.e. the next assigned volume is identical with the current volume; this indicates usually a geometry setup problem or a wrong propagation solution (i.e. wrong sided cylinder intersection in all recorded cases).

- **no next volume**: the navigation hits the end of the described `TrackingGeometry`; in general, this is also due to incompatible input parameters.

Once a navigation break is triggered, the extrapolation is reset and carried out again without navigation and hence without material effects integration. This behavior can be switched off for validation reasons, such that navigation breaks can be triggered and investigated, see Sec. 6.1.1.

4 Material Effects Integration

A particle that traverses material is subject to both energy loss and directional scattering. While for all stable particles (that are subject of track reconstruction in high energy physics experiments) multiple coulomb scattering is the by far dominant contribution to the directional deflection and thus no distinction between the different particles has to be done, energy loss has to be treated differently for electrons than for all other particles that are in the following also referred to as *heavy particles*. This is, because electrons lose a substantial part of their energy due to *bremsstrahlung*, while for heavy particles ionisation loss remains the dominant effect. As a consequence, a particle definition has to be present in the extrapolation realm to distinguish between the different energy loss mechanism. This is steered by a dedicated `enumeration` type: the so-called `ParticleHypothesis`[7].

Since multiple scattering is a stochastic process with a zero mean deflection it is, in general, not regarded during the transport of the track parameters, but applied as Gaussian or multi-Gaussian process noise addition to the track covariances. Section 4.1.3 will show in the following that the assumption of a Gaussian-distributed multiple scattering noise is valid to a large extent. It will also cover the integration of multiple scattering into the track extrapolation software structure.

Energy loss effects, on the other hand, introduce changes to both the trajectory of the particle and the track uncertainties: the loss of momentum clearly changes the particle trajectory in presence of magnetic field, but since energy loss is a stochastic process it can not be corrected for in a purely deterministic way. In practice, energy loss is taken into account by a deterministic mean (or most probable) value and a relatively small variance to this value. Again, the variance is applied as Gaussian process noise addition to comply with the linear least squares methods predominately used in track fitting. Nevertheless, the Gaussian model is far from optimal for electron energy loss description: Section 4.1.4 will concentrate on both the energy loss treatment for heavy particles and electrons and will discuss the validity range of the given applications.

Since both processes, multiple scattering and energy loss, rely evidently on a correct description of the traversed detector material, providing this material distribution within the reconstruction geometry is an *a priori* requirement, fulfilled by the ATLAS `TrackingGeometry`. The navigation, as described in

[7] The name *hypothesis* is chosen deliberately in this context. In most track fitting applications, the particle identification is not done yet, since it relies on additional information from calorimetry, combined reconstruction or specific interaction signatures with the detector material.

Sec. 3 is responsible for finding and forwarding the appropriate material information to the extrapolation and material integration algorithms. A generic updater `AlgTool`, the `MaterialEffectsUpdator` performs the actual correction of the `TrackParameters` objects.

Point-like and continuous Material Effects Integration The ATLAS extrapolation package enhances two different methods for the integration of material effects during tracking the particle through the reconstruction geometry. A classical point-like material update mechanism is the main material interaction philosophy, carried out through a dedicated `AlgTool`, the `MaterialEffectsUpdator`. It requires a purely `Layer`-based (and thus `Surface`-based) description of the ATLAS detector through a simplified reconstruction geometry. The transport of the track parameters and their associated covariances is hereby fully decoupled from the actual correction of the track representation due to interaction with the detector material. An example for the `Surface`-based modeling of the pixel barrel detector can be seen in Fig. 8.

A second, modern model of continuous material effects integration that embeds the correction as additional terms into the equation of motion can be chosen alternatively, see Sec. 2.3. The continuos material effects integration relies on a `Volume`-based description of the detector through the `TrackingGeometry`. This requires, in general, a more detailed description close to the realistic detector setup or is limited to large structures that can be modeled as one homogeneous block of material[8]. In ATLAS, the continuous integration of material effects is mainly planned to be used for the track extrapolation through the calorimeter and for traversing the large inert material structures in the Muon Spectrometer.

4.1 The `MaterialEffectsUpdator AlgTool`

Although both, multiple scattering and energy loss depend on the particle momentum p, they can be treated as two independent (stochastic) processes, since — in general — the energy loss Δ is small compared to the momentum p of the particle. It can therefore be assumed that p is constant during the multiple scattering process, which disentangles the energy loss from the multiple scattering integration. In the realisation of the ATLAS extrapolation package, this relation is respected by two dedicated `AlgTool` implementations that perform the tasks of energy loss respectively multiple scattering calculations separately. The `Extrapolator`, however, only refers to a single `AlgTool`, the `MaterialEffectsUpdator` that uses the other `AlgTool` instances.

4.1.1 Pre-, post- and full update

The material effects integration in the ATLAS extrapolation engine enhances the concept of *pre-*, *post-* and *full* update directives, to account for an optimal spatial position of the material distribution. This is necessary since the reconstruction geometry is a simplified version of the real detector geometry and consists of only several idealised layers that carry a projected material distribution. For the ATLAS silicon detector, for example, the layers of the reconstruction geometry correspond to the actually build barrel cylinders and endcap discs. The largest contribution to the material budget per layer in the silicon detector is the sensor material itself and, furthermore, most of the additional support and cooling structures are located on the back-side of the detector sensor. Deflections and energy loss caused by traversing these modules contribute in a Kalman forward filter consequently only on the successive measurement module — and are applied therefore as *post-update* with respect to the measurement update on the given surface; however, they have to be taken into account before the measurement update in the smoothing or backward filtering process (*pre-update*). The *Transition Radiation Tracker* (TRT) in the ID, on the other hand, can be regarded to have an almost continuous material distribution and is in the reconstruction geometry represented by condensed layers. This description is anyhow only valid in a global picture and the *pre/post* update mechanism is superfluous: a full update is applied when crossing a layer of such a type or any other layer that is not associated to sensitive (tracking) detector elements. Figure 8 shows the necessary simplifications for the pixel

[8] It is of little sense to describe detector parts with a discrete material distribution such as the silicon detector through such a model. The material taken into account would hereby be only valid for traversing the entire volume and is therefore not suitable for track reconstruction, where correct knowledge of the spatial information of the material is essential for a satisfactory description of both multiple scattering and energy loss effects.

barrel reconstruction geometry and the consequences for the material integration in an extrapolation process.

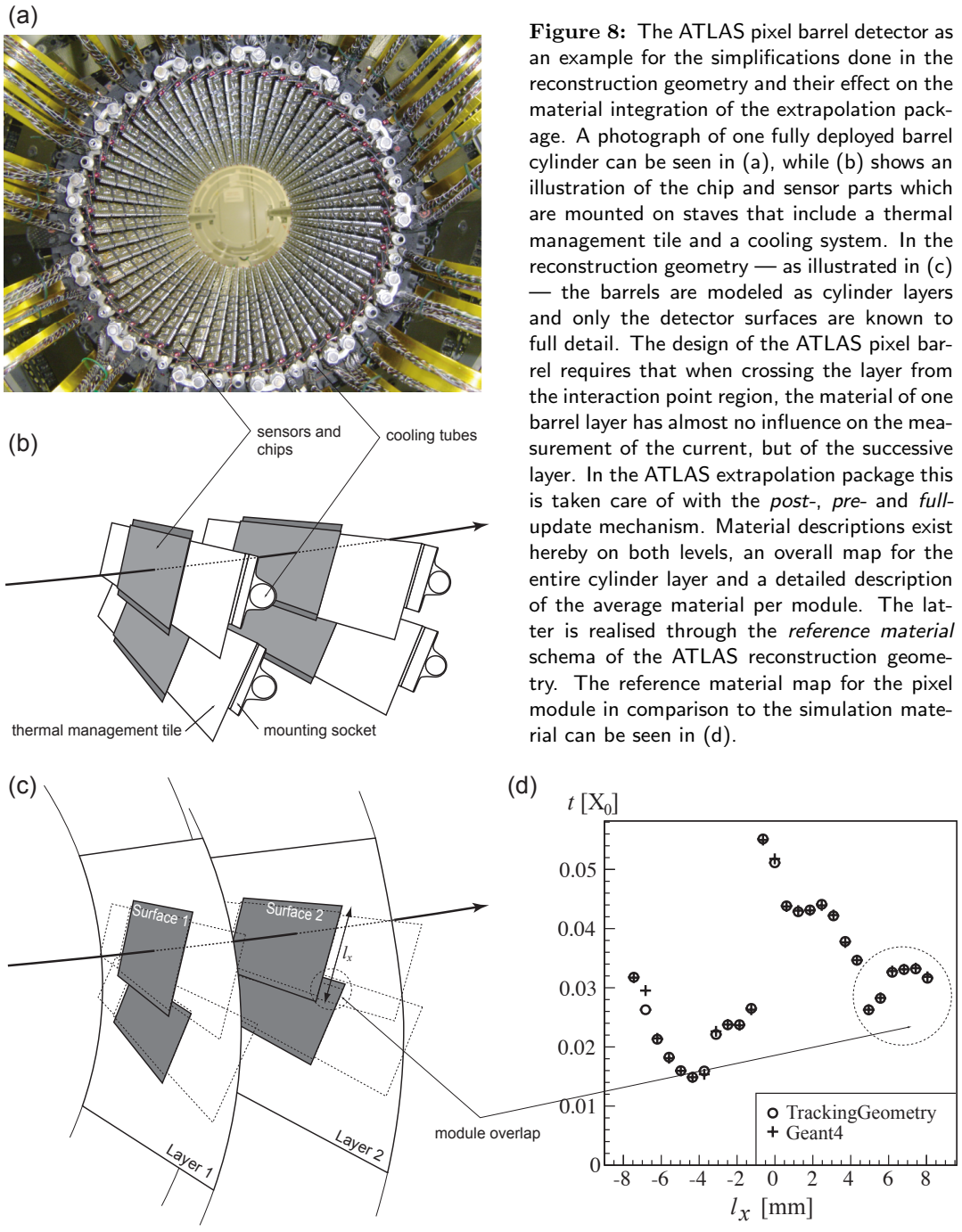


Figure 8: The ATLAS pixel barrel detector as an example for the simplifications done in the reconstruction geometry and their effect on the material integration of the extrapolation package. A photograph of one fully deployed barrel cylinder can be seen in (a), while (b) shows an illustration of the chip and sensor parts which are mounted on staves that include a thermal management tile and a cooling system. In the reconstruction geometry — as illustrated in (c) — the barrels are modeled as cylinder layers and only the detector surfaces are known to full detail. The design of the ATLAS pixel barrel requires that when crossing the layer from the interaction point region, the material of one barrel layer has almost no influence on the measurement of the current, but of the successive layer. In the ATLAS extrapolation package this is taken care of with the *post-*, *pre-* and *full-*update mechanism. Material descriptions exist hereby on both levels, an overall map for the entire cylinder layer and a detailed description of the average material per module. The latter is realised through the *reference material* schema of the ATLAS reconstruction geometry. The reference material map for the pixel module in comparison to the simulation material can be seen in (d).

4.1.2 Correction for the Incident Angle and for Track Bending

The structure of the ATLAS reconstruction geometry and in particular the projective mapping of the simulation material onto the simplified frame of cylinder and disc layers requires two additional corrections to the material distribution that depend on the trajectory properties of the particle and can therefore be only accounted for while performing the actual extrapolation:

- **Correction for the incident angle:** the amount of material seen by a particle when traversing a layer-like structure depends on the relative incident angle of the track with the layer. In the ATLAS reconstruction geometry, particular emphasis has been put on creating a description

that is not only valid for particles originating from the nominal interaction point (or center of the coordinate system). That is why — when creating the material maps from the simulation geometry — the material projection of the material seen by a particle coming from the nominal interaction region is recalculated to the fixed frame of cylinders and discs. During the track extrapolation process, the angle of the trajectory with the crossed layer is calculated and the relative correction to the track length in the material applied: $\mathbf{n}$ and $\mathbf{t}$ denote in the following the normal vector of the layer and the momentum direction of the trajectory at the intersection point, respectively — both vectors at unit length. The actual crossed path length s through any arbitrarily positioned and oriented layer of thickness d can then be generically expressed as

$$s = \frac{d}{\mathbf{n} \cdot \mathbf{t}}. \tag{8}$$

- **Bending correction:** in a dense volume (e.g. the TRT or calorimeter volumes) that is described through condensed layers in the `TrackingGeometry` a second effect has to be accounted for. Since the mapping of the material distribution is done following a straight line projection, the material budget seen by strongly bent tracks is in general underestimated. This effect can be simply cancelled by introducing a correction factor that is dependent on the transverse particle momentum p_T. ΔR denotes in the following the traversed radial distance in the detector, and B_z the longitudinal component of the magnetic field. The bending radius r of a track can then be written as $r = \frac{p_T}{0.3 \cdot B_z}$. The angle α then describes the opening angle of a transverse track segment and can be expressed as $\alpha = 2 \arcsin(\frac{\Delta R}{2 \cdot r})$. Performing some simple manipulations, the correction factor c_b introduced due to bending can be expressed as[9]

$$c_b = \frac{s}{\Delta R} = \frac{p_T}{0.15 \cdot B_z} \cdot \arcsin \frac{0.15 \cdot B_z \Delta R}{p_T} \approx 1 + \frac{(\Delta R)^2 \cdot B_T^2}{266.7 \cdot p_T^2}, \tag{9}$$

when the magnetic field is given in Tesla and the momentum is expressed in GeV. Equation (9) shows the dependency of correction the factor to be $1 + 1/p_T^2$, which approaches rapidly towards 1 for transverse momenta bigger than 1 GeV while remaining an about 5% effect at a transverse momentum of 500 MeV, which marks the transverse momentum threshold of many ATLAS reconstruction applications. Figure 9 illustrates the relevant track information that is necessary to calculate the bending correction and shows the correction factor c_b as a function of the transverse momentum p_T.

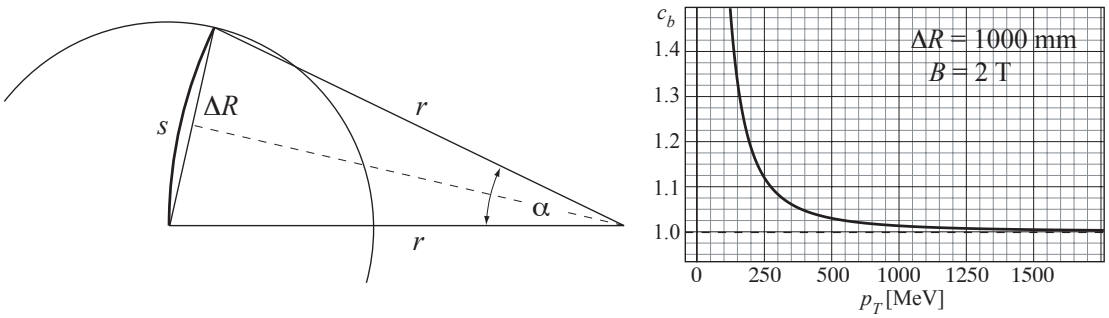


Figure 9: The correction to the material seen by a track crossing condensed layers due to track bending in the magnetic field in an schematic illustration to the left and to the right the dependency of the correction factor on the transverse momentum for a concrete example of traversing a sample distance of 1 m which corresponds roughly to the Inner Detector dimension.

Both the incident angle and the bending correction are matter of the reconstruction geometry, but have to be applied within the material integration during the track extrapolation process. The extrapolation

[9] It is assumed in this formalism that the track enters a volume along a radial direction, which is certainly not true for strongly bent tracks that originate from the nominal interaction point. The resulting correction to ΔR is, however, already taken into account through the incident angle correction when traversing a layer.

engine, however, has been designed to stick to a generic detector model such that no information about the underlying detector specifications can be accessed, while still keeping the maximum performance level. This requirement has been met by defining the corrections as actions known to the `Layer` and `TrackingVolume` that are able to return a `pathCorrection()` or `bendingCorrection()`, respectively, depending on the actual input parameters of the track.

4.1.3 Multiple Scattering: the `MultipleScatteringUpdator AlgTool`

A particle that traverses detector material undergoes successive small angle deflections, caused by multiple (Coulomb) scattering. Given the central limit theorem it can be assumed that the sum of these small variations is Gaussian distributed and symmetrically centered around zero. However, large angle single scattering processes disturb the purely Gaussian probability density function (PDF) and add large non-Gaussian tails. As a rule of thumb, the assumption of the Gaussian character of the underlying PDF is valid to about 98%, being limited to the core region of the distribution.

The integration of multiple scattering effects is handled by a dedicated `AlgTool`, the so-called `MultipleScatteringUpdator`. The calculation is done using the well known *Highland formula* [13], which is an empirical adoption of Molière's solution of the transport equation starting from the classical Rutherford cross section of a single scattering process [14].

Highland expanded the original expression given by Molière for the root mean square σ_{ms}^{proj} of the projected scattering angle θ^{proj} with an empirical logarithmic correction term to adopt for the slightly underestimated screening of the nucleus Coulomb potential in materials with lower Z. Furthermore, he transformed — for convenience — the result into a function of the pathlength t in terms of the radiation length X_0 which leads to the well-known expression[10]

$$\sigma_{ms}^{proj} = \frac{13.6\ \mathrm{MeV}}{\beta cp} Z\sqrt{t/\mathrm{X}_0}\,[1 + 0.038 \ln(t/\mathrm{X}_0)], \tag{10}$$

when Z and p describe the charge and momentum of the incident particle, respectively.

Projection Correction and lateral Displacement In the ATLAS track parameterisation the momentum direction is expressed through the globally defined polar angle θ and the azimuthal angle ϕ, see Eq. 1. Since θ already represents a projected angle with respect to the z axis, σ_{ms}^{proj} can be directly applied to the corresponding covariance term, while for the azimuthal angle a correction term of $\frac{1}{\sin\theta}$ has to be applied to the root mean square to account for the out of plane projection.

Another aspect of multiple scattering is that, in general, there exists a correlation between the actual deflection in space θ^{space} and the local coordinates after the scattering process. The local displacement due to scattering is hereby depending on the two projected scattering angles and the thickness of the traversed material. In a `Layer`-based description of the reconstruction geometry, the layer thickness is, however, only a model parameter and has little to do with the actual thickness traversed during the multiple scattering process (in the following referred to as *scattering thickness*). In addition, the `Layer`-based description intrinsically assumes that the material free regions in the according detector volumes are big in comparison to a typical scattering thickness, and the local error on the successive measurement surface is therefore mainly dominated by the directional uncertainties in ϕ and θ. The displacement on the scattering surface caused by the multiple scattering process is therefore omitted in this application. For the continuous integration of material effects, on the other hand, the actual path length s corresponds to the scattering thickness and it is included in the treatment of multiple scattering [12].

Molière's theory of multiple scattering is — when being applied in the small angle assumption — not restricted to a specific particle type nor spin. It is based on the assumption that the deflection of the scattered particle does not change the magnitude of the particle's momentum, or, in other words, it is a pure *elastic* single scattering theory. Rossi and Greisen [15], however, showed that for electrons that traverse a significant amount of material this assumption is not valid anymore since the electron momentum changes substantially due to radiation loss, which is described in more detail in Sec. 4.1.4. This leads to a modification of the momentum dependency from $1/p^2$ to $1/(p_i p_f)$, when

[10] The multiple scattering process itself has little to do with the radiation length X_0 other that both show the same dependency on the atomic number Z and the molecular weight A of the material.

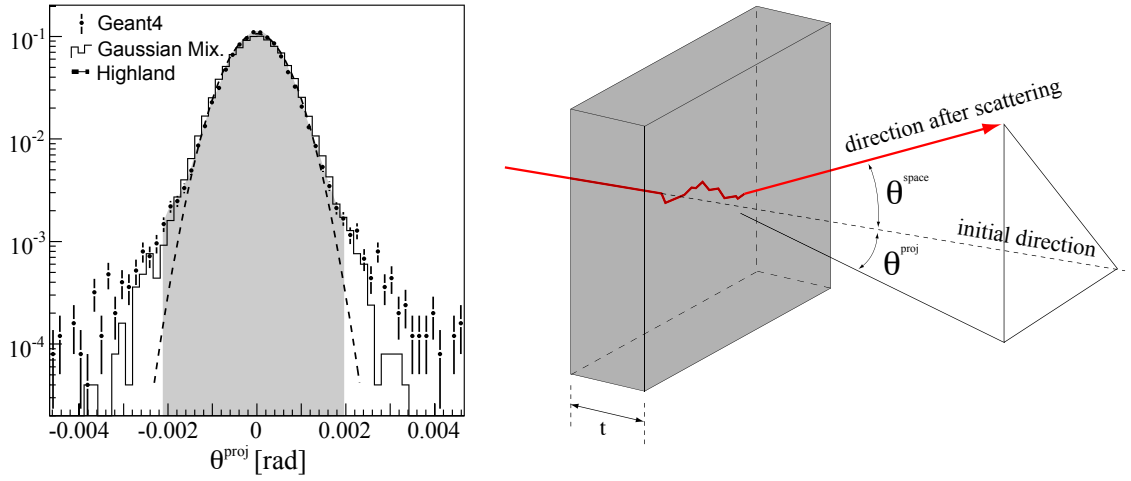


Figure 10: The projected scattering angle distribution θ^{proj} (left) of a muon with 5 GeV that traverses a silicon detector with a thickness t that corresponds to 1% of radiation length X_0 . The distribution (illustrated at standard measure) has been created by Monte Carlo simulation of 25000 muon events using the Geant4 simulation toolkit. It shows the Gaussian distribution originating from the Highland formula (dashed) and the a Monte Carlo Gaussian Mixture model (solid). The shaded area represents a 98% core fraction of the Geant4 distribution. The illustration (right) shows the definitions of the space angle θ^{space} and one projected angle θ^{proj} for an example multiple scattering process.

p_i and p_f denote the initial and final momentum of the multiple scattering process, respectively, and a constant loss assumption is applied[11]. Electron track reconstruction is, on the other hand, mainly determined by the highly non-Gaussian character of the energy loss distribution due to bremsstrahlung that shadows the multiple scattering effects. Identified electron tracks are therefore best fitted with dedicated track fitting techniques. The momentum dependency of the multiple scattering process plays hereby a negligible role.

The integration of such a modified multiple scattering theory for electrons would require the knowledge of the momentum transfer and is not compatible with disentanglement of multiple scattering and energy loss effects. A simplified version of the Rossi-Greisen results, using the $1/p^2$ dependency but sticking to the parameters obtained for electrons can be chosen though for electron multiple scattering in the ATLAS extrapolation package.

Gaussian Mixture Model The Gaussian PDF of stochastic multiple scattering is disturbed by single large angle scattering processes that add significant tails to the distribution of the projected scattering angle. For track reconstruction this effect is negligible since it is small in comparison to loss of accuracy introduced by the necessary simplification of the detector geometry[12]. The ATLAS extrapolation engine that has been initially developed for track reconstruction is also used as the trajectory creation module in the FATRAS simulation and incorporates therefore an additional model for multiple scattering, capable of reproducing parts of the tail distribution. A mixture of two Gaussians with zero mean value, one for the core contribution, one for the tail approximation is used to describe the projected scattering angle distribution

$$f(\theta_{ms}) = (1-\epsilon)\cdot g_0(\theta_{ms};\sigma_{core}) + \epsilon \cdot g_0(\theta_{ms};\sigma_{tail}), \tag{11}$$

where

$$g_0(x;\sigma) = g(x;\mu=0,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{x^2}{2\sigma^2}). \tag{12}$$

[11] More sophisticated models can also be considered where the energy loss function enters directly to the multiple scattering calculation and an integration of the energy loss distribution has to be performed during the scattering process.

[12] It is of second priority to account for a few percent tail effect in the description of the scattering process, when the material distribution is locally not known to such an accuracy.

The model parameters ϵ, σ_{core} and σ_{tail} are hereby depending on the traversed material thickness in terms of the radiation length X_0 and are taken from [16]. Recently [17] even more precise models of the multiple scattering descriptions have been developed that expand the double Gaussian mixture model with a non-Gaussian tail descriptiond. The current ATLAS extrapolation engine has stuck to the Gaussian mixture since it shows satisfactory agreement with the full Geant4 simulation while being lightweight and convenient. A future inclusion of more sophisticated models will however be easily possible by yet another implementation of the `IMaterialEffectsUpdator` interface[13].

Figure 10 shows both the Highland formula application and the Gaussian mixture model as implemented in the `MaterialEffectsUpdator` in comparison with data from a Monte Carlo simulation using the well known and validated Geant4 [18] simulation toolkit. It also illustrates the main definitions used in the calculation of the projected scattering angle.

4.1.4 Energy Loss: the `EnergyLossUpdator AlgTool`

Energy loss of particles traversing detector material occurs due to electromagnetic effects - mainly ionisation (in the order of α^2), bremsstrahlung (order of α^3), direct pair production (order of α^4) and photonuclear interactions; α denotes the fine-structure constant with $\alpha \simeq 1/137$. The PDF $\rho(\Delta)$ (often referred to as *straggling function*) of the energy loss Δ is highly non-Gaussian, but for the use in most track fitting applications an approximation to a Gaussian distribution has to be done.

For heavy particles with masses above 100 MeV peripheric collisions with the detector material — also called *ionisation loss* — dominate the overall energy loss process. The probability for hard collisions with the nuclei of the detector material is suppressed by the factor $\frac{1}{m^2}$ and can therefore be neglected for heavy particles. The electron mass, however, is about 200 times smaller than the mass of the next heavier stable particle and hence interactions with the strong electromagnetic field of the nuclei that cause bremsstrahlung have to be considered. Above a certain energy threshold, bremsstrahlung starts to dominate the energy loss distribution for electrons.

The ATLAS `EnergyLossUpdator AlgTool` performs the energy loss calculation during the track extrapolation process, which depends on the provided `ParticleHypothesis`, the material properties and the kinematic parameters of the particle. The applied corrections are described in the following paragraphs.

Energy Loss of heavy Particles The energy loss Δ of heavy particles in the energy range of final state particles originating from high energy collision experiments is dominated almost entirely by ionisation loss. Although this is a stochastic process that follows a PDF $\rho(\Delta)$, it is justified to treat it as a deterministic mean or averaged energy loss and a relatively small variance. This is, because Δ is usually small in comparison to the particle momentum.

The mean energy loss of a heavy particle per unit length x due to ionisation loss is described by the well known Bethe-Bloch formula [19]

$$\frac{dE}{dx} = \alpha^2 2\pi N_a \lambda_e^2 \frac{Z m_e}{A\beta^2} \left[\ln \frac{2m_e\beta^2\gamma^2 E'_m}{I^2(Z)} - 2\beta^2 + 1/4 \frac{E'^2_m}{E^2} - \delta \right], \tag{13}$$

where

N_a	=	$6.023 \cdot 10^{23}$, Avogadro's number
Z, A		atomic number and weight of the traversed medium
m, m_e		rest masses of the particle and the electron
β	=	p/E, where p is the particle momentum
γ	=	E/m
λ_e	=	$3.8616 \cdot 10^{-11}$ cm is the Compton wavelength of the electron
$I(Z)$		the mean ionisation potential of the medium,
E'_m		the maximum energy transferable to the electrons of the medium with $E'_m = 2m_e \frac{p^2}{m_e^2 + m^2 + 2m_e\sqrt{p^2+m^2}}$
δ		density correction.

[13] It is worth mentioning that this is one of the biggest benefits of the component software model that has been deployed in the ATLAS track reconstruction.

A parameterisation for the density correction δ and calculated values for various materials can be found in [20].

The Bethe-Bloch formula shows that the mean value of the energy loss depends on the ratio of the particle mass to the particle momentum. The ionisation loss plays therefore often an important role in particle identification, whereas for track reconstruction in the momentum range of interest it is of minor importance. Figure 11 illustrates the defined validity range of the pure ionisation loss assumption given by the Bethe-Bloch formula and shows a comparison of the mean energy loss calculation done within the `EnergyLossUpdator` to values found in literature for various different materials [21].

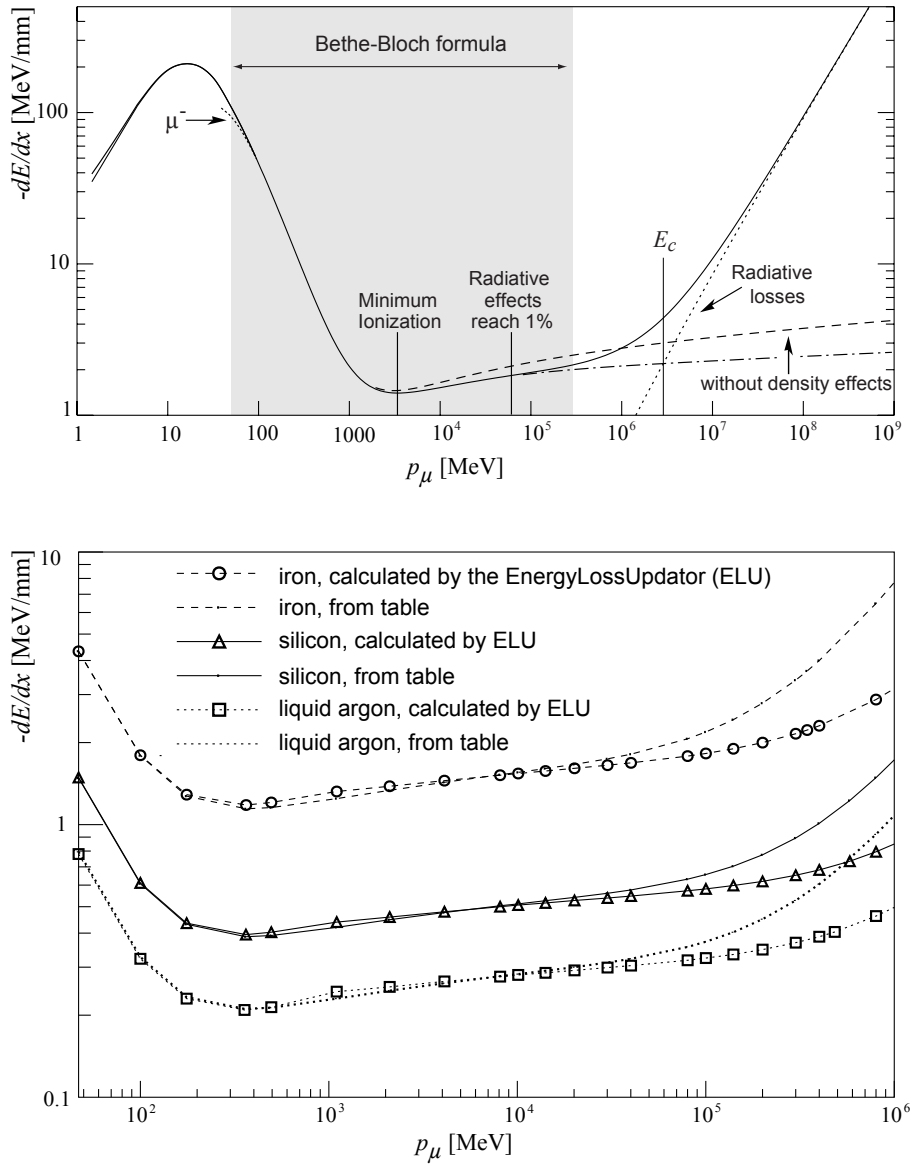


Figure 11: The upper plot shows the mean energy loss per unit length of a muon in copper and is based on [22]. The grey area shows the momentum range where the Bethe-Bloch formula is valid to satisfactory accuracy. The second plot shows a comparison of the mean energy loss of muons in silicon, liquid argon and iron with data taken from [21] and calculated using the `EnergyLossUpdator`. Good agreement can be observed up a particle momentum of about 150 GeV, where radiative effects start to dominate the energy loss distribution. Radiative Effects are not yet implemented in the standard ATLAS `EnergyLossUpdator`.

Landau Distribution and most probable Energy Loss The Bethe-Bloch formula only describes the expected mean energy loss of charged particles due to ionisation. In reality, ionisation loss is a

stochastic process that leads to asymmetric fluctuations around a most probable value (MPV). Landau [23] described this distribution[14] and showed that the most probable energy loss ${}_L\Delta_p$ can be expressed as

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\beta^2\gamma^2}{I} + \ln \frac{\xi}{I} + 0.2 - \beta^2 - \delta \right], \tag{14} $$

with $\xi = ZN_a \frac{k}{\beta^2} t$, when t denotes the thicknesss of the traversed material. It can be shown that in the asymptotic behavior of $\gamma >> 1$ (i.e. $\beta \approx 1$) the density correction δ can be modeled as

$$ \delta \approx 4.447 - \ln \gamma^2, \tag{15} $$

which simplifies Eq. (14) to

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\gamma^2}{I} + \ln \frac{\xi}{I} - 0.8 + 4.447 \right]. \tag{16} $$

Equation (16) is used in the standard `EnergyLossUpdator AlgTool` for the calculation of the most probable energy loss of heavy particles. Figure 12 illustrates the energy loss distribution for a heavy particle in a silicon layer and includes possible Gaussian approximations that are enhanced by the ATLAS `EnergyLossUpdator`.

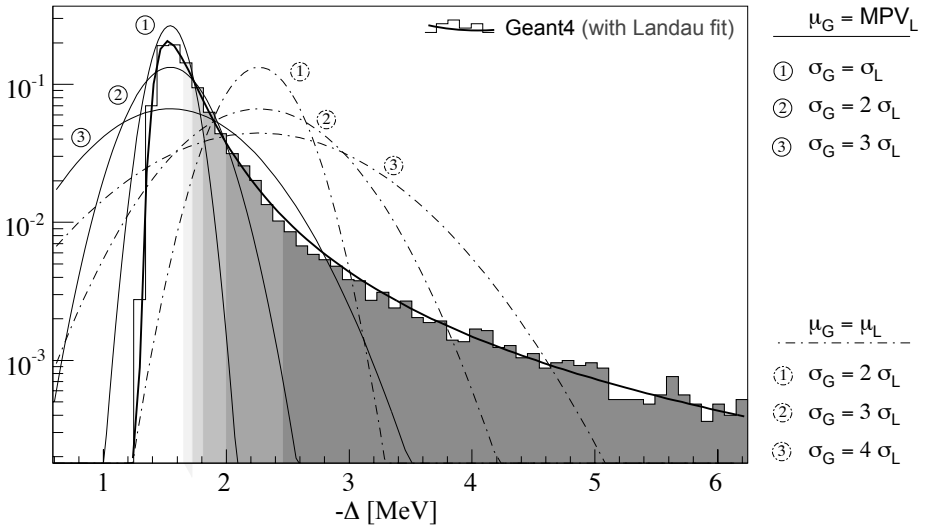


Figure 12: The energy loss distribution at standard measure for single muons with 5 GeV traversing 4.68 mm (i.e. 5% X_0) of Silicon. The distribution has been created with the Geant4 toolkit and fitted with a landau distribution. Different possible Gaussian approximations around the theoretical mean (μ_L, dashed curves) or respectively most probable value (MPV$_L$, solid curves) that can be chosen for modeling the landau distribution are shown. The shaded areas illustrate additional 10% of entries starting from 50% (non-filled area).

Energy Loss of Electrons Electrons lose a substantial part of their energy due to bremsstrahlung, but inonisation loss still remains a contribution to the total effect. The ATLAS extrapolation package is capable of correcting for both, mean ionisation loss and bremsstrahlung even if the ionisation loss can be neglected in many cases. The description of the ionisation loss is slightly modified in the form factors in comparison with the heavy particle case. Energy loss by bremsstrahlung is well described by the theory of Bethe and Heitler [24]. Following a standard notation where z denotes the ratio of the final energy E_f to the initial energy E_i and denoting the amount of material traversed by the particle in terms of radiation length X_0 as t, the PDF of z is given by

$$ \rho(z) = \frac{[-\ln z]^{c-1}}{\Gamma(c)}, \tag{17} $$

[14] Nowadays only known as *Landau distribution*.

where $c = t/\ln 2$ and z is evidently restricted to $z \in (0, 1)$.
The average mean (radiative) energy loss per unit length is then given as[15]

$$(dE/dx)_{rad} = -E_i/X_0 \tag{18}$$

From Eq. (18) one can learn that the expectation value for z is $< z >= e^{-t}$ and the variance can be approximated by $\mathbf{var} < z >= e^{-t \ln 3/\ln 2} - e^{-2t}$, which propagates a noise addition of $\sigma^2_{q/p}$ to the covariance matrix of the ATLAS track parameter q/p as

$$\sigma^2_{q/p} = \frac{1}{< z >^2 p^2} \cdot \mathbf{var} < z >, \tag{19}$$

when this kind of update is applied.
The standard track parameterisation used in the ATLAS tracking EDM is defined such that the uncertainties of the track are implicitly assumed to be Gaussian distributed, and it can be shown that the application of the average energy loss described by the Bethe-Heitler formula (including the Gaussian noise addition to the track uncertainties) introduces a strong bias towards too low momentum reconstruction [25]. Figure 13 shows a comparison of the energy loss distributions of a heavy particle (μ) to an electron when traversing the same detector layer.

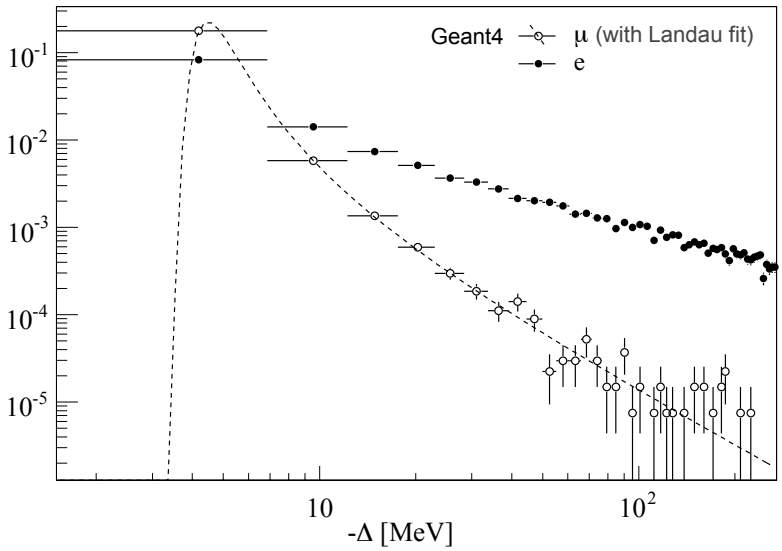


Figure 13: Comparison of muon energy loss to electron energy loss in a silicon layer of 10% X_0 thickness. The particles have been generated using Geant4 and where propagated with a initial momentum of 2 GeV . The muon energy loss distribution follows hereby the Landau distribution, while the electron energy loss distribution is disturbed by the long tail due to radiation loss. This results in a theoretical mean value up to 10 times bigger than for pure ionisation loss.

4.2 Summary of the Material Effects Integration

The ATLAS extrapolation engine enhances different material update mechanisms that have been described within this section. Some of the described options are dedicated to the fast track simulation FATRAS. Table 1 gives — for the convenience of the reader — a summary of the implemented techniques and indicates the configuration flags to be chosen for the various applications. The property flags refer to the `MultipleScatteringUpdator` and `EnergyLossUpdator`, respectively.
The width of the Gaussian approximations to the energy loss functions can be adjusted by specifying one additional property of `EnergyLossUpdator`.

[15]Note that Eq. (18) led to the definition of X_0 through ${X_0}^{-1} \approx 4\alpha r_e^2 Z(Z+1)N \cdot E_i(\ln 183 Z^{-1/3} + 1/18)$, when $N = \rho N_a/A$ is the number of atoms per unit area and r_e the classical radius of the electron.

Table 1: Possible material effects mechanisms that are accessible through the ATLAS extrapolation package, the description is given for multiple scattering (MS) and energy loss effects.

Mechanism	Equation	Particle	Property flag
MS, single Gaussian	Eq. (10)	all	default
MS, double Gaussian	Eq. (11)	all	`GaussianMixtureModel=true`
mean ionisation loss	Eq. (13)	all	default
most probable ionisation loss	Eq. (16)	all	`bool` in method signature
mean radiation loss	Eq. (18)	$e^{+/-}$	`UseBetheBlochForElectrons=false`

5 Extrapolation

Propagation, navigation and material effects integration are steered by one `AlgTool`, the `Extrapolator`. It establishes the simple user front-end and exists in a two-folded way: as a *fully configured* `AlgTool` whose entire setup is done at job configuration via python setup, or as a *strategy pattern* interface where the executing propagation `AlgTool` is part of the method signature.

The following four different extrapolation methods can be performed either in the fully configured or component pattern setup:

- `extrapolate(...)` this is the main extrapolation interface; a starting track representation, a destination surface, a particle hypothesis that determines the type of material interactions, a propagation direction and a boundary directive for the destination surface can be provided. The `extrapolate(...)` interface exists in a two-folded signature: starting from a single track representation, given by the `TrackParameters` class, or from an entire `Track` object. In the latter case, the closest position on the given trajectory is chosen as a starting point of the extrapolation process. This guarantees an optimal accuracy depending on the given track information[16].
- `extrapolateDirectly(...)` this is the direct forward call to the propagation `AlgTool`, it allows to do a simple propagation without navigation and material effects through the `IExtrapolator` interface.
- `extrapolateStepwise(...)` the method signature is identical to the main `extrapolate(...)` method, but the interface method differs by the return type. A step-wise navigation to the destination surface is performed, stepping down to sub-surface level on intersected layers as described in further detail in Sec. 3.2.4. The `extrapolateStepwise(...)` returns the track representations on all crossed active detector surfaces in a `std::vector`. The last element of the vector is the track extrapolation at the destination surface. This method builds the core of the *a posteriori* holes-on-track search.
- `extrapolateBlindly(...)` this method is identical with the step-wise extrapolation, but does not require a provided destination surface. It stops with the last boundary surface of the known `TrackingGeometry` and is used for trajectory creation within FATRAS.

Recently, the extrapolation package has been extended to support neutral track parameterisations and the new track particle base class. This is aimed at extending the current use of the extrapolation package (that is mainly restricted to track reconstruction) and providing the powerful extrapolation methods to hight level event reconstruction algorithms and physics analysis applications.

5.1 Internal Extrapolation Flow

Encapsulated from the user interface, every extrapolation process is performed as a sequence of actions that are realised as `private` method of the `Extrapolator AlgTool`. The first step is the initialisation

[16] In the ATLAS computing model, the track representation loses the hit information when being transformed from the *Event Summary Data* (ESD) the the *Analysis Object Data* (AOD). Consequently, many operations performed on ESD level lead to higher accuracy compared to the similar operation on AOD data.

of the the navigation, performed by the `initializeNavigation()` method: the starting volume and associated layer to the starting parameters are determined as well as the according destination volume. From this point on, the entire extrapolation flow is driven automatically through the volumes that are crossed by the trajectory: the extrapolation is carried out to the boundary of the current volume, until the destination volume is reached. The mechanism of the `BoundarySurface` objects together with the `Navigator AlgTool` are described in Sec. 3, Fig. 14 shows an UML sequence diagram of a sample extrapolation process.

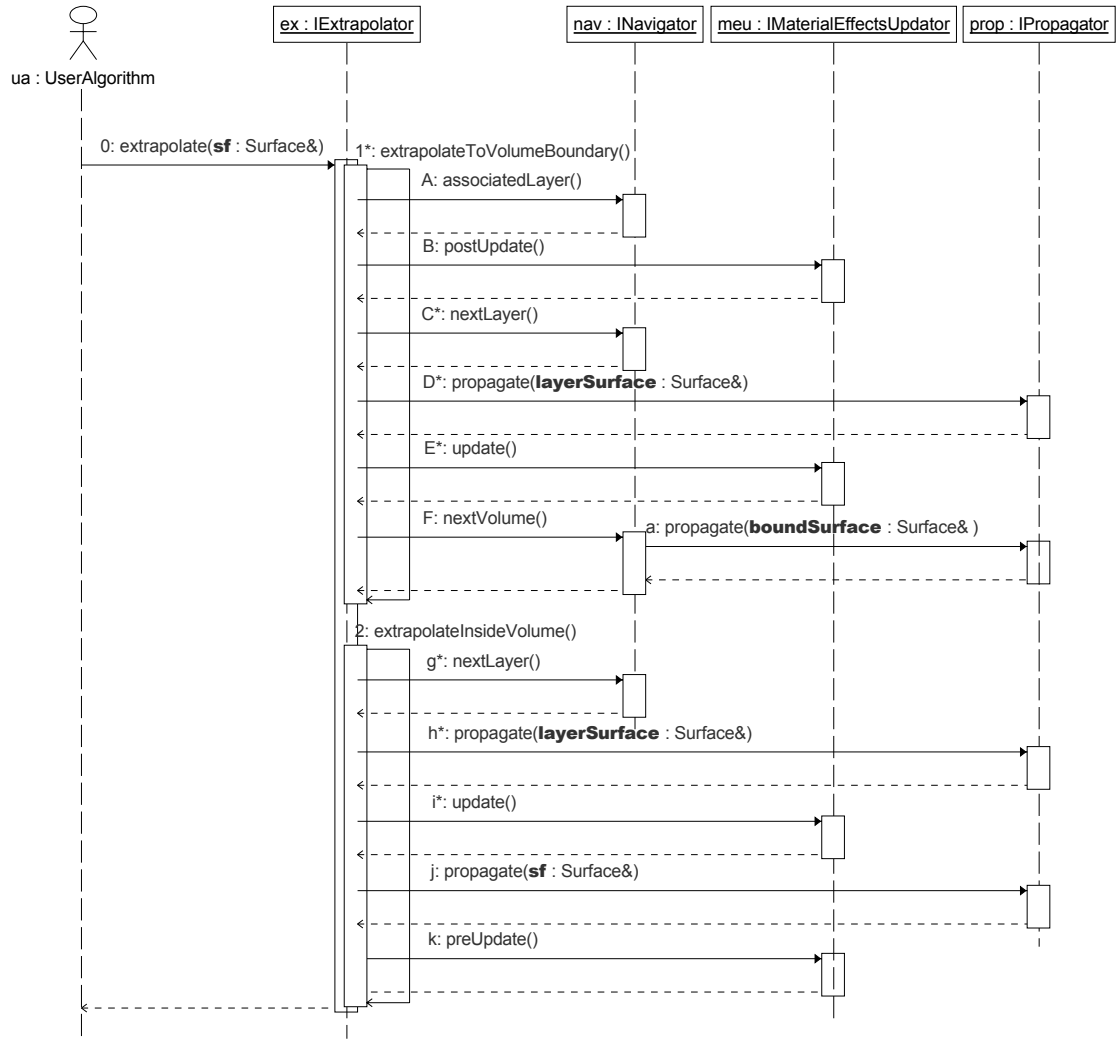


Figure 14: Simplified UML sequence diagram of an extrapolation process; for convenience, the method signatures are mostly omitted and the process is only illustrated for a fully static setup. The only method signatures indicated in the diagram are the different surface types that occur: the destination surface **sf**, various surfaces that represent material layers **layerSurface** and several boundary surfaces labelled as **boundSurface**. Since all surface types extend the same `surface` base class they can be used with the underlying propagation `AlgTool`. The initial `extrapolate()` call to the destination surface **sf** invokes repetitive private `extrapolateToBoundary()` calls until the destination volume is found and the `extrapolateInsideVolume()` succeeds. Inside the volume a sequential extrapolation from one `Layer` class to the next is done.

Volume driven Extrapolation Methods The ATLAS `TrackingGeometry` is put together from volumes of different internal structure; each type triggers a different navigation strategy in the extrapolation process. The most commonly used setup is a fully static description of the detector using layers

that are pre-ordered with a given reference direction. The track parameters are hereby propagated from layer to layer and updated according to the material description of the intersected layer. An adjustable and volume dependent maximum number of failed attempts to intersect the successive layer triggers the search for the next volume through intersecting the appropriate boundary surface of the volume. Other parts of the detector may refer to an external material description (e.g. a database, parameterisation) or, when regarding the traverse of the calorimeter, the integration of an actual energy loss measurement could be taken instead of a parametric description through the reconstruction geometry. A dynamic approach has been developed for these cases that allows to create layers for a point-like material integration once the path through a volume is roughly known. It is realised by a dedicated `AlgTool`, the `IDynamicLayerCreator` that can be registered to a volume of the reconstruction geometry and implemented in various different ways. Currently, a wrapper for the material description used in the `Muonboy` reconstruction algorithm [26] that is restricted to the MS exists in a prototype setup.

Additionally, volumes in the reconstruction geometry can be modeled as dense material which allows the usage of the `STEP_Propagator`, see Sec. 2.3. Moreover, detached structures exist in the reconstruction geometry that need specific navigation. All these cases are determined by the structure of the current volume to be crossed.

6 Performance Tests

Track extrapolation processes are very frequently performed by many different modules of the event reconstruction. These client algorithms are immediately affected by the performance and reliability of the track extrapolation engine. Thus, dedicated focus has been drawn on (partly automated) validation and error triggering mechanisms during the design phase of the TrkExtrapolation realm. One major part of the overall track extrapolation quality is the correct description of the underlying detector material by the reconstruction geometry, which is not in particular subject of the extrapolation process itself and will not be covered in this section. A detailed consideration and reflection of this topic can be found in [7].

Both the reliability and the accuracy (or quality) of the track parameter extrapolation have to be investigated. Whereas the first part is somewhat trivial since it simply includes consistency checks and the monitoring of a high number of extrapolation processes, the second task appears to be less trivial as it first occurs. The quality of the extrapolation solution is determined by the correct handling of material interactions and the precision of the mathematical transport of both parameters and covariances through the magnetic field. Latter can be done by comparing the calculated transport Jacobian matrices with numerically evaluated ones, which will be shown in Sec. 6.2. The numerical estimation of the transport Jacobian relies on the correct propagation of the track representation, which can not easily be validated, since the true solution of the intersection is not known. One is thus restricted to compare different modules and propagation techniques amongst each other and to perform self-consistency checks with forward-backward extrapolations.

6.1 Stand-alone Extrapolation Tests

The resulting track parameters resolution is evidently the ultimate validation of the the track extrapolation. However, it accumulates all different aspects of track reconstruction and thus complicates the monitoring of single effects. It is sometimes useful to factorise the various parts of the track reconstruction into smaller modules that can be investigated in a controlled environment. The ATLAS extrapolation package offers dedicated algorithms to monitor special aspects of the extrapolation process. Most of the `Algorithm` classes designed to perform sample extrapolations are located in the TrkExAlgs package of the Tracking repository.

6.1.1 Self-consistency and Navigation Performance

A stable and reliable navigation process through the `TrackingGeometry` is necessary to assure the correct material integration during the transport of the track parameters. While in the track reconstruction the `Extrapolator` is configured to switch to a direct extrapolation without navigation

in case that a navigation break has been triggered (see Sec. 3.4), a dedicated `Algorithm` exists for validation purpose that stops the navigation process in such a case and records the specifications of the input parameters and the break condition. This algorithm also allows the monitoring of the boundary surface mechanism, since it records the necessary switches to the second or third boundary hypothesis when exiting a `TrackingVolume` of the reconstruction geometry. It can be shown that the navigation is reliable at highest level, starting from compatible input parameters (i.e. after testing that a geometrical intersection of the trajectory with the given destination surface exists) only one in about 40000 test extrapolations fails due to a navigation break[17]. The extrapolation tests have been carried out incorporating full material effects integration in a momentum range between 500 MeV and 150 GeV . For each randomly generated destination surface within the Inner Detector volume, the transport of the track parameters has been performed before the parameters on the destination surface have been back extrapolated to the starting position. Energy loss is applied in the forward direction, while the deposed energy is again added to the $\frac{q}{p}$ parameter during backward extrapolation. By this, the self-consistency of the material effects integration can be checked accordingly. Figure 15 shows absolute or relative errors for track parameters after the extrapolation to the destination surface and back extrapolation to the initial starting frame. One other aspect can be monitored during theses tests: if the `Navigator AlgTool` is configured to run in validation mode, it records the performance of the entry, respectively exit surface estimation of the various volumes. It can hereby be shown that within typical extrapolation processes, the straight line approximation that is used to estimate the hit volume surface is to more than 99.8 % correct.

6.2 Validation of the Covariance Propagation

The correct transport of the track parameters errors is essential for a good tracking resolution, in particular when the extrapolation is used with progressive fitting techniques that rely on the transported covariances[18]. The validation of the covariance transport through the propagator implementations is therefore inevitable. A dedicated validation algorithm has been established that is capable of comparing the transport Jacobian matrix as calculated by the `IPropagator` implementations with numerically evaluated ones. The numerical derivatives are hereby calculated by using the method of Ridders [27] that is based on a symmetric derivative

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{20}$$

and aims to calculate $f'(x)$ in the limit $h \to 0$. Evidently, Eq. (20) can not be evaluated for this limit, some approaching procedure to the limit value has therefore to be taken. The basic idea of Ridders method is to decrease the parameter h successively and to evaluate the derivative for each case. The limit value is then calculated by a polynomial extrapolation towards $h = 0$. Figure 16 shows the relative residuals of the Jacobian matrix entries

$$r_{jk} = \left(\frac{\mathbf{J}_{if}^{calc} - \mathbf{J}_{if}^{num}}{\mathbf{J}_{if}^{num}} \right)_{jk} \tag{21}$$

between the numerically evaluated and the provided Jacobian matrices, but is probably not very intuitive for representing the quality of the covariance transport, since it does not reflect the different relevance of the various coefficients. Naturally, some components of the covariance matrix play a more important role than others in the track fit, since some track parameters pairs are stronger correlated then others (this is mainly determined by the detector and magnetic field setup). The pure comparison of all Jacobian coefficients, however, does not distinguish between *relevant* and *non-relevant* components. A second, more striking way of showing the quality of the covariance transport

[17] In a typical reconstruction job this number is evidently larger, since the compatibility test can not be *a priori* done. During the track reconstruction the assignment of a hit to a given trajectory is, in general, based on a cruder assumption that is given through the pattern recognition.

[18] In least squares fitters, a crude prediction of the covariance matrix is, in general, sufficient, since it is mainly used to evaluate the compatibility of the hit with the given track hypothesis but does not enter the fit *per se*. However, the transport Jacobian matrix that relates the fitted parameters with the track parameters on the various measurement surfaces is indeed needed for the least squares fit.

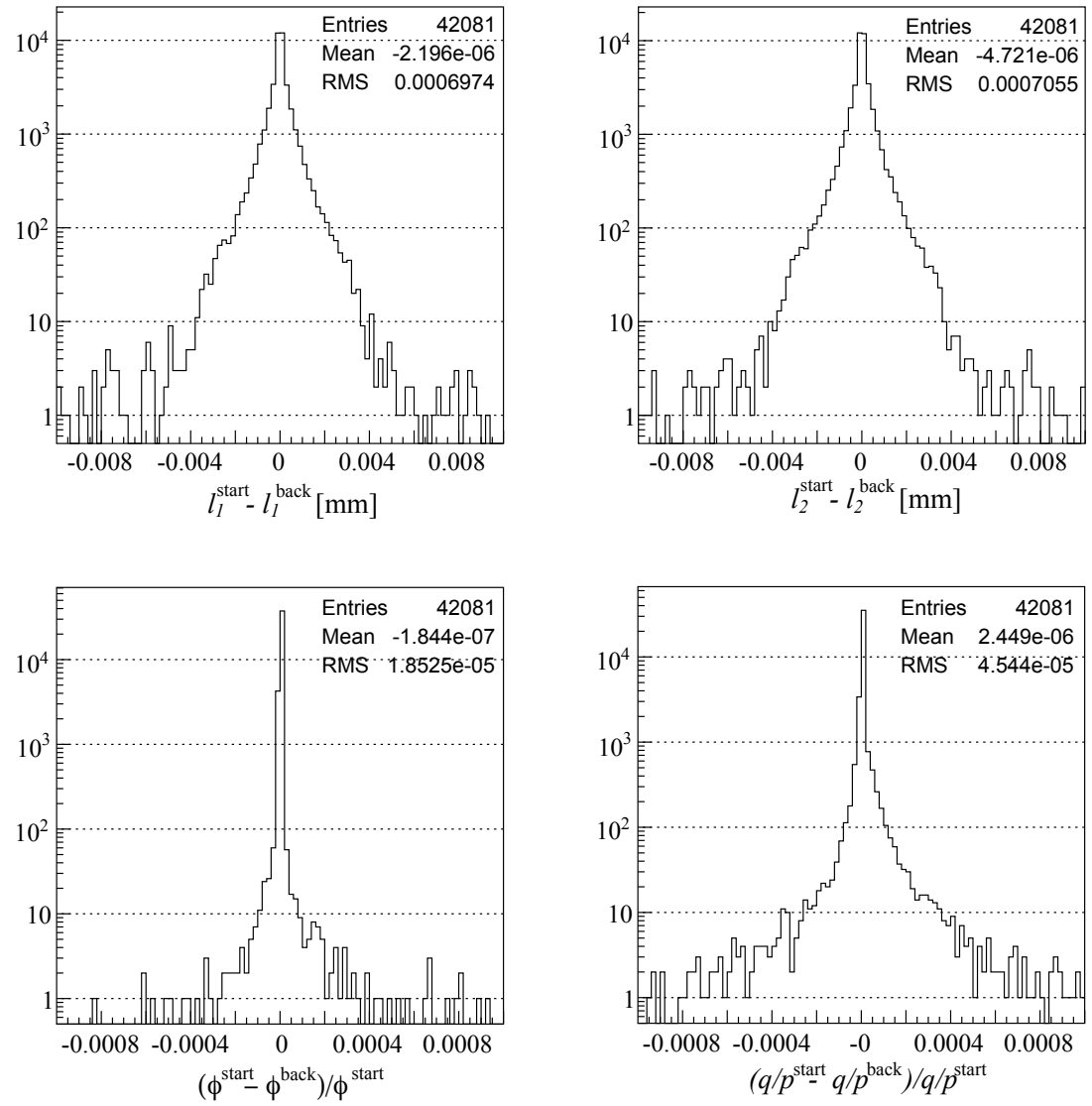


Figure 15: The absolute respectively relative error of about for sample extrapolations within the Inner Detector including full navigation and material effects integration. About 40000 extrapolations for the interaction region (smeared start parameters) to randomly distributed planar surfaces in the Inner Detector and back to the start surface are performed. The two upper plots show the absolute difference of the local coordinates of the start parameters with the back propagated ones. The latter plots show the relative error on the initial ϕ respectively $\frac{q}{p}$ values, the start parameters have been generated equally distributed over the entire azimuthal range and in the momentum interval between 500 MeV and 150 GeV .

is to check the parameter pull distributions

$$f(x_i) = \frac{(x^{rec} - x^{true})_i}{\sigma_i} \tag{22}$$

for the five parameters after the track fit. Unfortunately — when using full detector simulation — the pull distributions are mainly dominated by the quality of the material description used in the track fit and the error estimation based on clusterisation. The pure parameter transport is thus, in general, shadowed by these more significant effects. The fast track simulation FATRAS, however, allows to create tracks with fully controlled input in both material description and hit errors. Figure 17 shows the pull distributions of the track parameters after refitting of 50000 single tracks simulated without material effects and purely gaussian cluster errors in the ID barrel ($|\eta| < 2.5$) using FATRAS .

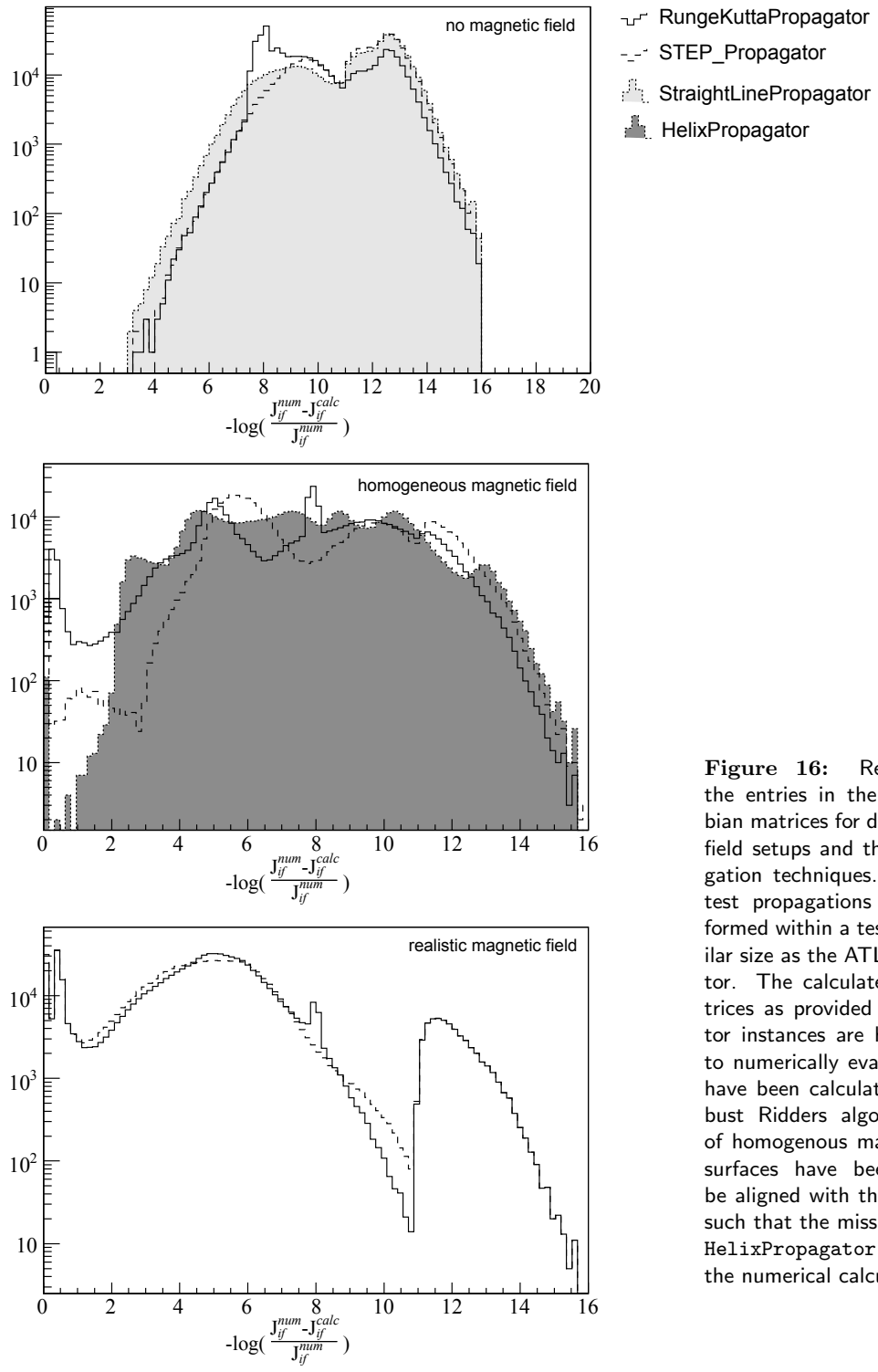


Figure 16: Relative errors of the entries in the transport Jacobian matrices for different magnetic field setups and the various propagation techniques. Fifty thousand test propagations have been performed within a test volume of similar size as the ATLAS Inner Detector. The calculated Jacobian matrices as provided by the propagator instances are hereby compared to numerically evaluated ones that have been calculated using the robust Ridders algorithm. In case of homogenous magnetic field, the surfaces have been restricted to be aligned with the magnetic field, such that the missing tuning of the `HelixPropagator` does not affect the numerical calculation.

6.3 Timing Performance

Since the extrapolation of track parameters is a very frequent process in the event reconstruction, it is of particular interest to be keep the CPU time contribution at minimal cost. The highest relative frequency of extrapolation calls in the event reconstruction is within the track fit, where every hit causes — depending on the fitting technique — usually at least two extrapolation steps. The optimisation of the timing performance of the track extrapolation package can therefore be best evaluated in the minimisation of the contribution to the track fitting applications.

In the standard ID event reconstruction of 10 $t\bar{t}$ events using the *New Tracking* (NEWT) [29] reconstruction chain, the `Extrapolator` is called about 80 000 times, predominately in the track fit;

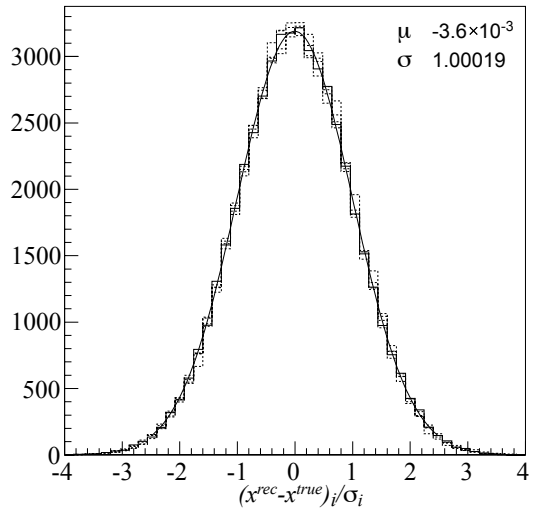


Figure 17: Perigee parameter pull distributions of 50000 tracks that have been simulated and refitted without material interactions and assuming Gaussian distributed cluster errors. All five track parameters are shown, as an example the pull distribution of transverse impact parameter d_0 is fitted with a Gaussian curve. The `RungeKuttaPropagator` has been used for these test.

the `Propagator AlgTool` is called additionally ten times more often in the pattern recognition. Table 2 presents an overview of the number of method calls and relative timing contributions of the extrapolation package per event.

The contribution of the extrapolation process to the track fit is obviously dependent on the used implementation of the ATLAS `ITrackFitter` interface. For the standard ATLAS Kalman filter, the extrapolation accounts to about two thirds of the overall time spent in track fitting during the entire event reconstruction. This number, however, includes the large number of track fits in the ambiguity solving process where many track candidates describe fake tracks and hence the navigation between the given measurement surfaces is not always straight-forward. It can be shown that for the refit of already found and resolved tracks the contribution of the extrapolation drops below 40 %. When using the global χ^2 implementation of the standard ATLAS track fitter interface, the relative contribution to the track fit is in general higher, since for each measurement the extrapolation is at least done three times to calculate the track derivatives numerically stable on the measurement surface. Recently, the propagator interface has been expanded to provide the used transport Jacobian to the client algorithms and thus the number of performed extrapolations in the least squares fit could be reduced to a single one per track. Additionally, this led to an increased stability of the track fit.

Table 2: Relative timing contribution to the event reconstruction measured for 10 $t\bar{t}$ events in the Inner Detector New Tracking reconstruction for the extrapolation components in the pattern recognition and track fitting process. The number marked with (★) are contained in the caller contribution.

Method	**Calls**	**Caller**	**Time/Evt [%]**
`RungeKuttaPropagator::propagate()`	789 238	pattern recognition	8.39
`Extrapolator::extrapolate()`	53 714	track/vertex fitting	2.77
`Extrapolator::extrapolateStepwise()`	26 549	*holes-on-track* search	1.56
`RungeKuttaPropagator::propagate()`	118 189	`Extrapolator`	1.63★

Internal timing statistics Two main contributions to the total time spent during a extrapolation process exist: the transport of the track parameters (and in particular their errors), such as the navigation that is responsible for finding the next volume to traverse. The modification of track parameters due to material effects is a negligible factor in the overall timing of the extrapolation engine. In the current realisation of the ATLAS `Extrapolator AlgTool` both steps are performed with the underlying propagation engine. The relative time contribution of the navigation (including the transport of the navigation parameters to the boundary surface) to the time spent while traversing a volume is about 25 %, but since the fast straight line test performed by the boundary surface mechanism fails only in less than 0.1 % this number can be clearly reduced in a future evolution of the extrapolation engine. Figure 18 shows a caller-diagram of the main extrapolation method in a

static volume setup.

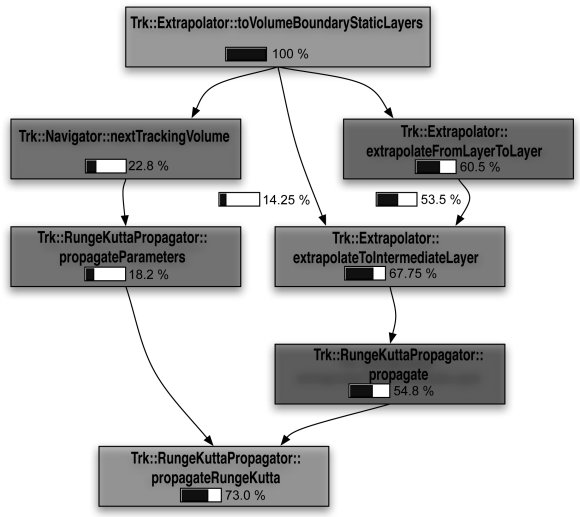


Figure 18: Simplified caller diagram for 500 sample extrapolations within the Inner Detector volume. The graph shows the contribution of navigation and actual extrapolation to the total time spend in the extrapolation to a volume boundary. In the current realisation of the ATLAS extrapolation package, the navigation is also performed using the underlying propagation engine. Encouraging results of stand alone navigation tests — see Sec. 6.1.1 — indicate however that a satisfactory performance can be also achieved by the simple straight line assumption of the boundary surface mechanism. It can be seen that when using the default `RungeKuttaPropagator` about 75% of the extrapolation time is spent in the propagation of track parameters and their errors, while the main contribution to the Runge-Kutta integration is the multiple lookup of the magnetic field value.

7 Conclusion and Outlook

During the last three years a powerful and well performing new extrapolation package has been developed in full coherence to the new ATLAS reconstruction geometry and integrated in the new modular software structure of the ATLAS track reconstruction. It serves as a central part of many applications in the event reconstruction including pattern recognition, track and vertex fitting and combined reconstruction. Several different propagation techniques have been integrated and the modular design facilitates future adaption and extension.

The new extrapolation engine extends the pure functionality needed for track and vertex fitting with a predictive navigation that can be used — together with the underlying tracking geometry — for holes-on-track search or for the fast track simulation FATRAS.

Large scale event reconstruction using the new extrapolation package has been carried out using Monto Carlo simulated data and taken data from the CTB 2004 and the commissioning runs using cosmic rays [28].

7.1 Outlook

An improvement in terms of a lower CPU time consumption of the extrapolation process can be achieved by simplifying the navigation process to directly use a straight line approximation for most of the cases, while including a fallback solution to full propagation if the straight line case fails to determine the correct exit surface, as shown in Sec. 6.3. An alternative approach could be to evaluate the usage of the recently integrated less complex `IIntersector` implementations, that exist as prototype versions in the ATLAS software release 13.1.0.

Recently, the propagation interface has been expanded to provide the transport Jacobian matrices to the client algorithms. This is in particular important for global track fitting techniques, but has not been fully propagated through the extrapolation interface. A future adaption foresees the access to the full transport information (including the noise terms due to material effects integration) through a modified `IExtrapolator` interface.

The ATLAS extrapolation engine has become a widely used tool for many different applications and is currently being used with various different geometry setups and navigation philosophies. This resulted in a rather extensive interface that concentrates these various different tasks. Current work is ongoing to split these different tasks that can be associated to with the different geometry setups of static,

dynamic and detached volumes structures in the extrapolation process into separate `AlgTool` classes that comply with the same interface definition.

A Appendix

A.1 Conventions and Typesetting

The following type setting conventions are followed throughout this document: Software packages within the ATLAS offline software repository [30] are written in Sans-sarif face, C++ or python class names are written in `Courier` face. Namespace definitions as used in the software repository are omitted in this document for readability. An exhaustive list of software packages and their location within the ATLAS software repository can be found in Tab. 3.

Table 3: Software leaf packages to be found in the Tracking/TrkExtrapolation CVS repository that are described or referred to within this document.

Leaf package	**Brief description**
TrkExAlgs	test and validation algorithms
TrkExExample	test steering package
TrkExSIPropagator	propagation `AlgTool` with a straight track model
TrkExHelixPropagator	propagation `AlgTool` with a helical track model
TrkExSTEP_Propagator	propagation `AlgTool` with material interactions
TrkExRungeKuttaPropagator	propagation `AlgTool` for inhomogeneous magnetic field
TrkExInterfaces	concentration of interfaces
TrkExDynamicLayerCreator	`AlgTool` to provide layer information to global fitters
TrkExTools	extrapolation `AlgTool`
TrkExUtils	shared utility classes

Typesetting of mathematical Formulas and geometrical Conventions Three different types of three-dimensional frames are used within this document or are referred to by the TrkDetDescr container:

- a coordinate system corresponding to the description of the magnetic field and the detector geometry, referred to as *global frame*.
- a cartesian frame moving along the track, the so-called *curvilinear frame*; the tangential momentum vector of the track builds the z-axis of this frame, x-axis and y-axis are then constructed with an additional global constraint.
- three-dimensional cartesian frames, different from the global frame, mainly attached to surfaces and volumes.

The standard cartesian set $\mathbf{e}_i$ of unit vectors describing the tracking frame are indicated as

$$E = (\mathbf{e}_x\,, \mathbf{e}_y\,, \mathbf{e}_z\,). \tag{23}$$

The standard base of the cartesian curvilinear frame is, for convenience, indicated as

$$U = (\mathbf{e}_u\,, \mathbf{e}_v\,, \mathbf{e}_t\,). \tag{24}$$

Finally, the standard base $\mathbf{h}_i$ of a three-dimensional cartesian frame different different from the tracking frame is noticed as

$$H = (\mathbf{h}_x\,, \mathbf{h}_y\,, \mathbf{h}_z\,). \tag{25}$$

In conjunction with a `Surface` object, this frame is sometimes referred to as *helper frame*, since it builds the carrier of the two-dimensional intrinsic surface frame (also called *local* frame).

A three-dimensional vector in either of the two frames is indicated by bold. Its expression with respect to the different frames is done using the standard base sets:

$$\mathbf{a} = a_x^e \mathbf{e}_x + a_y^e \mathbf{e}_y + a_z^e \mathbf{e}_z = a_u^e \mathbf{e}_u + a_v^e \mathbf{e}_v + a_t^e \mathbf{e}_t = a_x^h \mathbf{h}_x + a_y^h \mathbf{h}_y + a_z^h \mathbf{h}_z \,. \tag{26}$$

A.2 Extrapolation to Distorted Surfaces

In reality, many surfaces in the ATLAS detector that are modeled in event simulation and reconstruction as ideal geometrical objects will be deformed or distorted with respect to their mathematical description. Detector straws or wires in drift tube detectors (such as the TRT in the ID or the *Monitored Drift Tubes* (MDT) in the MS) will be deformed due to gravitational wire sagging; planar surface may be bent due to gravitation for extensive objects or simply due to mounting tension. Some of these effects will be dealt with at the stage of hit calibration when a parametric or projective approach is of satisfactory accuracy, but in particular for large objects, such as the long MDT tube wires, the geometrical distortion has to be taken into account during the extrapolation process. The ATLAS `TrackingGeometry` provides hereby a dedicated `DistortedSurface` base class; concrete surface types extend this base class and the appropriate surface class from the TrkSurfaces repository. The extrapolation process to a distorted surface is then realised by a two step propagation to the final destination surface. Since the `DistortedSurface` types extend the common surface class it can be naturally used with the propagator to find an intersection with the nominal or non-distorted surface. In the following, a corrected surface in position and rotation of same type is created dynamically depending on the intersection solution of the first propagation. The final propagation is then performed to the corrected and more realistic surface position. Figure 19 shows an illustration of an example propagation to a wire surface with gravitational sagging correction (`SaggedLineSurface`).

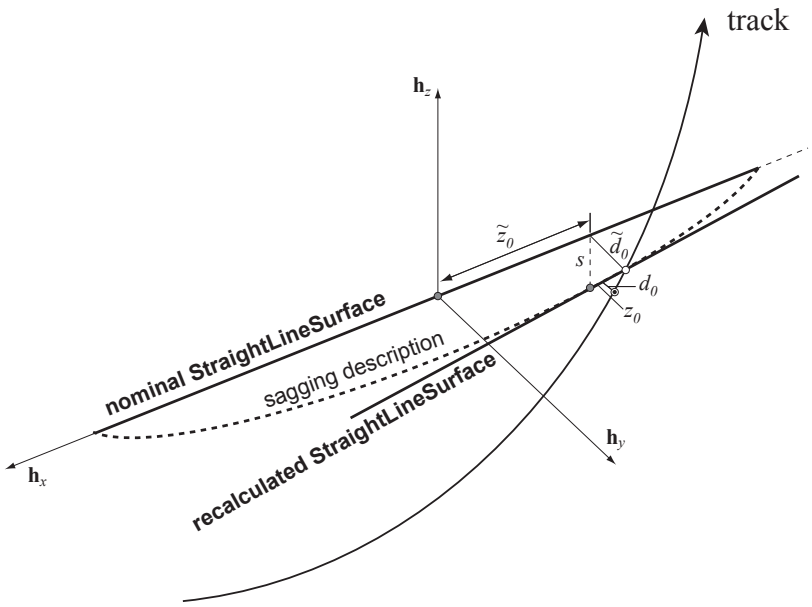


Figure 19: Illustration of a propagation to a wire surface with gravitational sagging correction: a first propagation to the nominal `StraightLineSurface` is performed which yields to the local coordinates $\tilde{d}_0$ and $\tilde{z}_0$. Depending on the longitudinal position on the line and the line orientation, the corrected `StraightLineSurface` is created and the final propagation performed.

A.3 Intersection formulas for straight line propagation

The following sections summarise the formulas used for the calculation of trajectory intersections with destination surfaces of different types in the `StraightLinePropagator`.

A.3.1 Line-Line Approach

The problem of finding the closest approach between two lines in three-dimensional space — one line describing the particle trajectory, the other one a straw measurement or the z axis for a perigee representation, respectively — can be easily solved when both lines are given in parametric form

$$\mathbf{l}_a(s) = \mathbf{m}_a + s \cdot \mathbf{d}_a,$$
$$\mathbf{l}_b(t) = \mathbf{m}_b + t \cdot \mathbf{d}_b. \tag{27}$$

The vector between any two points on the two lines can then be expressed by

$$\mathbf{k}(s,t) = \mathbf{l}_b(t) - \mathbf{l}_a(s) = \mathbf{m}_{ab} + t \cdot \mathbf{d}_b - s \cdot \mathbf{d}_a, \tag{28}$$

where $\mathbf{m}_{ab} = \mathbf{m}_b - \mathbf{m}_a$. The vector $\mathbf{k}(s_0, t_0)$ denotes the vector between the two closest points $\mathbf{l}_{a,0} = \mathbf{l}_a(s_0)$ and $\mathbf{l}_{b,0} = \mathbf{l}_b(t_0)$ and is perpendicular to both, $\mathbf{d}_a$ and $\mathbf{d}_b$.
This defines a system of two linear equations:

$$\mathbf{k}(s_0,t_0) \cdot \mathbf{d}_a = \mathbf{m}_{ab} \cdot \mathbf{d}_a + t_0 \cdot \mathbf{d}_a \cdot \mathbf{d}_b - s_0 = 0 \tag{29}$$

$$\mathbf{k}(s_0,t_0) \cdot \mathbf{d}_b = \mathbf{m}_{ab} \cdot \mathbf{d}_b + t_0 - s_0 \cdot \mathbf{d}_a \cdot \mathbf{d}_b = 0 \tag{30}$$

Solving (29) and (30) for s_0 and t_0 yields

$$s_0 = \frac{(\mathbf{m}_{ab} \cdot \mathbf{d}_a) - (\mathbf{m}_{ab} \cdot \mathbf{d}_b)(\mathbf{d}_a \cdot \mathbf{d}_b)}{1 - (\mathbf{d}_a \cdot \mathbf{d}_b)^2}, \tag{31}$$

and, respectively,

$$t_0 = -\frac{(\mathbf{m}_{ab} \cdot \mathbf{d}_b) - (\mathbf{m}_{ab} \cdot \mathbf{d}_a)(\mathbf{d}_a \cdot \mathbf{d}_b)}{1 - (\mathbf{d}_a \cdot \mathbf{d}_b)^2}. \tag{32}$$

A.3.2 Line-Plane Intersection

An arbitrary plane is defined as the set of points $\mathbf{x}$ that comply with

$$\mathbf{n}(\mathbf{x} - \mathbf{p}) = 0, \tag{33}$$

where $\mathbf{n} = (n_x, n_y, n_z)$ describes the normal vector of the plane and $\mathbf{p}$ a freely chosen reference point that lies within the plane. Given a line with $\mathbf{l}(u) = \mathbf{m} + u \cdot \mathbf{d}$, the solution for the intersection can be found by inserting this into Eq. (33) and solving for u. This results in

$$u = \frac{\mathbf{n}(\mathbf{p} - \mathbf{m})}{\mathbf{n} \cdot \mathbf{d}}. \tag{34}$$

A.3.3 Line-Cylinder Intersection

The calculation of the intersection of a line with an arbitrarily oriented cylinder is shown here after transforming the line into the cartesian frame of the cylinder, i.e. the coordinate system where the symmetry axis of the cylinder builds the z-axis; In this frame the cylinder is centered around the origin and can be projected to a circle in the $x - y$ plane.

Let $\mathbf{p}_1 = (p_{1x}, p_{1y}, p_{1z})$ and $\mathbf{p}_2 = (p_{2x}, p_{2y}, p_{2z})$ be two points of the line in the cylinder frame. The solution can then be found in the projective $x-y$ plane where the lines can be described as $y = kx+d$, with

$$k = \frac{p_{2y} - p_{1y}}{p_{2x} - p_{1x}}, \tag{35}$$

and respectively

$$d = \frac{p_{2x}p_{1y} - p_{1x}p_{2y}}{p_{2x} - p_{1x}}. \tag{36}$$

It intersects the corresponding circle $x^2 + y^2 = R^2$, which can then be found by a simple quadratic equation and reinsertion into the line equation.

A.4 Intersection Formulas for helical Propagation

All calculations performed to find the intersection of the helix with a given `Surface` object are performed in the so-called *helix frame* in which the helix progresses along the z axis and has an initial starting position of $\mathbf{h}_0 = (R, 0, 0)$. The helix parameterisation in this frame is done in the azimuthal angle ϕ and can be expressed as

$$\mathbf{h}(\phi) = \begin{pmatrix} R \cdot \cos\phi \\ R \cdot \sin\phi \\ \hat{o} \cdot R \cdot \phi \cot\theta \end{pmatrix}, \tag{37}$$

when $\hat{o} = \pm 1$ defining the orientation of the helix. The solution for the intersections with different surface types are found by determining the parameter ϕ_i at the intersection point. The momentum vector at the intersection is then expressed as

$$\mathbf{p}(\phi_i) = p \cdot \frac{\mathbf{h}'(\phi_i)}{||\mathbf{h}'(\phi_i)||}, \tag{38}$$

when $\mathbf{h}'(\phi) = \frac{\partial \mathbf{h}(\phi)}{\partial \phi}$.
In general, more than one solution exists for the intersection of a helix with different surfaces. In the ATLAS track reconstruction, however, the helix radius given by the momentum range of particles that are subject of track reconstruction is big in comparison to the detector component to be intersected, thus the solution is in most cases unambiguous.

A.4.1 Helix-Plane Intersection

The intersection of a helix with an plane that is expressed in the helix frame can be found by inserting the helix parameterisation given in Eq. (37) into to the normal vector form of a plane given by Eq. (33). After some manipulations, the intersection solution is given by the root of the function

$$\begin{aligned} f(\phi) &= n_x \cos\phi + n_y \sin\phi + \hat{o} \cdot n_z \cdot \phi \cot\theta - \tfrac{\delta}{R} \\ &= a_1 \cos\phi + a_2 \sin\phi + a_3 \cdot \phi + a_4 \qquad = 0 \end{aligned}, \tag{39}$$

when $\delta = \mathbf{n} \cdot \mathbf{p}$. Equation (39) is characterised by a linear part in ϕ overlaid by an oscillating part with the coefficients a_1 and a_2. Two special cases exist for which the intersection can be calculated analytically: for surfaces parallel to the $x - y$ plane of the helix frame $n_x = n_y = 0$ and the solution is given by the linear equation in ϕ. On the other hand, if the the normal vector of the plane is perpendicular to the z axis of the helix frame, the problem can be solved in the $x - y$ coordinate plane and reduces to the same case as described in Sec. A.3.3. In the general case, however, the solution can not be found analytically it has to be calculated numerically using an iterative approach. In the implementation of the ATLAS `HelixPropagator` a standard Newton-Rhapson method is used for finding the root of Eq. (39). First a starting point for the iterative procedure is found by solving the linear part, which yields

$$\phi_0 = -\frac{a_4}{a_3}. \tag{40}$$

The solution of the azimuthal angle ϕ_i at the intersection point is then found by iterating

$$\phi_i = \phi_{i-1} - \frac{f(\phi_{i-1})}{f'(\phi_{i-1})} \tag{41}$$

when $f'(\phi) = \frac{\partial f(\phi)}{\partial \phi}$. The iteration is stopped once the function value $f(\phi)$ drops below a certain numerical cut value.

A.4.2 Helix-Cylinder Intersection

The intersection of a helix with a cylinder is trivial in the case when the symmetry axis of the cylinder is parallel to the helix guiding center (z axis in the helix frame). It reduces to finding the intersection of two circles in the projected $x-y$ plane and can be solved accordingly. This is the most common case for the `HelixPropagator` since the solenoidal magnetic field in the ATLAS Inner Detector points along the global z axis (and one could assume it to be constant in an defined inner volume) and most cylinders that represent layers or volume boundaries are aligned in the same way.

For completeness, the `HelixPropagator` does also include the calculation with arbitrarily oriented cylinders, which is similar to the intersection of a helix with a plane solved in an iterative Newton-Rhapson approach. In the helix frame, the set of point $\mathbf{x}$ that define a cylinder surface of radius r are given by satisfying the equation

$$|[\mathbf{x}-\mathbf{c}]\times\mathbf{l}_z|^2 = r^2, \tag{42}$$

when $\mathbf{c}$ denotes the center position and $\mathbf{l}_z$ the symmetry axis of the cylinder. Inserting the helix parameterisation $\mathbf{h}(\phi)$ given in Eq. (37) and defining $\mathbf{d}=\mathbf{c}\times\mathbf{l}_z$ follows

$$[\mathbf{h}(\phi)\times\mathbf{l}_z]^2 - 2\cdot(\gamma(\phi)\times\mathbf{l}_z)\cdot\mathbf{d} + \mathbf{d}^2 - r^2 = 0. \tag{43}$$

Solving Eq. (43) in components yields

$$\begin{aligned} f(\phi) \quad &= a_0 + a_1\cdot\phi + a_2\cdot\phi^2 \\ &+a_3\cos\phi + a_4\sin\phi + a_5\sin(2\phi) \\ &+a_6\phi\cos\phi + a_7\phi\sin\phi \\ &+a_8\cos^2\phi + a_9\sin^2\phi \qquad\qquad = 0 \end{aligned} \tag{44}$$

The solution for this equation is again performed in an iterative way using the solution of the quadratic equation in the coefficients a_0, a_1 and a_2 as seed for the iterative solution of the entire expression.

A.4.3 Helix-Line Approach

The calculation of the closest approach of a helix to a line that is parallel to the z axis is trivial, since it is positioned on the plane that is defined through the two parallel lines. For calculating the closest approach of a helix to an arbitrary straight line it is convenient to transform the line into the helix frame representation. The line parameterisation as given in Eq. (27) and the helix expression — Eq. (37) — are used in this frame, respectively. The two points of closest approach $\mathbf{l}_c = \mathbf{l}(t_c)$ on the line, and respectively $\mathbf{h}_c = \mathbf{h}(\phi_c)$ on the helix are then represented by a parameter set (t_c, ϕ_c).

The vector joining these two points $\mathbf{j} = (\mathbf{h}_c - \mathbf{l}_c)$ is orthogonal to both, the line direction and the track direction at $\mathbf{h}_c$ which is expressed through

$$\begin{aligned} \mathbf{j}\cdot\mathbf{d} &= 0 \\ \mathbf{j}\cdot\tfrac{\partial\mathbf{h}_c}{\partial\phi} &= 0 \end{aligned} \tag{45}$$

Solving the first equation for t_c and inserting the solution into latter, yields, after doing some manipulation,

$$\begin{aligned} f(\phi) \quad &= a_0 + a_1\phi \\ &+a_2\cdot sin(\phi) + a_3\cdot cos(\phi) + a_4\cdot sin(2\phi) + a_5\cdot cos(2\phi) \\ &+a_6\cdot\phi\cdot sin(\phi) + a_7\cdot\phi\cdot cos(\phi) \qquad\qquad = 0 \end{aligned} \tag{46}$$

The solution is then carried out in the same way as for finding the intersection of a line with a cylinder.

A.5 Analytic Transport of the Covariance Matrix using the curvilinear Frame

The analytic transport of the covariance matrix along the the curvilinear frame is shown for a helical track model and follows the path laid out in [11], but is adapted for the ATLAS track parameterisation. Since the straight line propagation is just one specific type of a helix propagation it can be evidently applied to this case as well. The initial problem of transporting a local covariance matrix $\mathbf{C}_{li}$ from an initial surface to a target surface is hereby split into the transformations carried out between the local

and curvilinear frame on the starting surface, the transport of the covariance along the curvilinear track frame and the final transformation between the curvilinear frame at the destination surface and its attached measurement frame, see Eq. (4). The curvilinear frame is hereby defined as given in Eq. (2).

A helical trajectory of a particle with charge q and momentum magnitude p in a constant magnetic field $\mathbf{B}$ can be parameterised as the global position $\mathbf{m}(\psi)$ and direction $\mathbf{t}(\psi)$ in a three dimensional space

$$\mathbf{m}(\psi) = \mathbf{m}_0 + \frac{\gamma}{Q}(\psi - \sin\psi)\mathbf{h} + \frac{\sin\psi}{Q}\mathbf{t}_0 + \frac{\alpha}{Q}(1-\cos\psi)\mathbf{n}_0$$

$$\mathbf{t}(\psi) = \frac{\partial \mathbf{m}(\psi)}{\partial s} = \frac{\partial \mathbf{m}(\psi)}{\partial \psi} \cdot \frac{\partial \psi}{\partial s} = \gamma(1-\cos\psi)\mathbf{h} + \cos\psi \cdot \mathbf{t}_0 + \alpha \sin\psi \cdot \mathbf{n}_0 \tag{47}$$

when $\psi = Q \cdot s$ denotes the momentum scaled run parameter s, $Q = -|\mathbf{B}|\frac{q}{p}$, and the initial vectors at $s = 0$ are written as $\mathbf{m}_0$ and $\mathbf{t}_0$, respectively. The vector $\mathbf{h}$ describes the direction of the magnetic field, $\mathbf{n} = \frac{\mathbf{h}\times\mathbf{t}}{\alpha}$ the normalised *trihedron* vector, α evidently the length of $\mathbf{h} \times \mathbf{t}$ and γ the projection of $\mathbf{h}$ onto $\mathbf{t}$.

We now investigate the influence of small variations on the start parameters $\mathbf{m}_0$, $\mathbf{t}_0$ and $\frac{1}{p_0}$ on the transported parameters $\mathbf{m}$, $\mathbf{t}$. The variations are applied through a displacement by $d\mathbf{m}_0$ that is restricted to the curvilinear $u - v$ plane, a deflection $d\mathbf{t}_0$ to the direction and a modification of the curvature given by $\delta(q/p_0)$. The variations result not only in modified target parameters but also in a variation of propagation length s to comply with the orthogonality relation

$$d\mathbf{m} \cdot \mathbf{t} = 0. \tag{48}$$

The total differentials $d\mathbf{m}$ and $d\mathbf{t}$ that incorporate the resulting variations on the the target surface with respect to modified starting parameters are build as

$$d\mathbf{m} = \frac{\partial \mathbf{m}}{\partial \mathbf{m}_0} d\mathbf{m}_0{}^{(a)} + \frac{\partial \mathbf{m}}{\partial \mathbf{t}_0} d\mathbf{t}_0{}^{(b)} + \frac{\partial \mathbf{m}}{\partial (1/p_0)} \delta(1/p_0)^{(c)} + \frac{\partial \mathbf{m}}{\partial s}\delta s^{(d)} \tag{49}$$

and, respectively,

$$d\mathbf{t} = \frac{\partial \mathbf{t}}{\partial \mathbf{t}_0} d\mathbf{t}_0{}^{(e)} + \frac{\partial \mathbf{t}}{\partial (1/p_0)} \delta(1/p_0)^{(f)} + \frac{\partial \mathbf{t}}{\partial s}\delta s^{(g)}. \tag{50}$$

Figure 20 shows the fundamental relations between the applied variations and their effects on the target surface.

Curvilinear to Curvilinear Transformations In the following, the transport of the covariance matrix between to curvilinear frames is shown to illustrate the principle of the further calculations.

We can, by inserting the partial derivatives of Eq. (47) identify the components of Eq. (49) and Eq. (50) as

$$\frac{\partial \mathbf{m}}{\partial \mathbf{m}_0} d\mathbf{m}_0 = d\mathbf{m}_0 \tag{51a}$$

$$\frac{\partial \mathbf{m}}{\partial \mathbf{t}_0} d\mathbf{t}_0 = \frac{\psi - \sin\psi}{Q}\mathbf{h}(\mathbf{h} \cdot d\mathbf{t}_0) + \frac{\sin\psi}{Q} d\mathbf{t}_0 + \frac{1-\cos\psi}{Q}(\mathbf{h} \times d\mathbf{t}_0) \tag{51b}$$

$$\frac{\partial \mathbf{m}}{\partial (1/p_0)} \delta(1/p_0) = p[s \cdot \mathbf{t} + \mathbf{m}_0 - \mathbf{m}] \cdot \delta(1/p_0) \tag{51c}$$

$$\frac{\partial \mathbf{m}}{\partial s}\delta s = \mathbf{t} \cdot \delta s \tag{51d}$$

$$\frac{\partial \mathbf{t}}{\partial \mathbf{t}_0} d\mathbf{t}_0 = (1-\cos\psi)\mathbf{h}(\mathbf{h} \cdot d\mathbf{t}_0) + \cos\psi + \sin\psi(\mathbf{h} \times d\mathbf{t}_0) \tag{51e}$$

$$\frac{\partial \mathbf{t}}{\partial (1/p_0)} \delta(1/p_0) = \alpha \cdot Q \cdot s \cdot p \cdot \mathbf{n} \cdot \delta(1/p_0) \tag{51f}$$

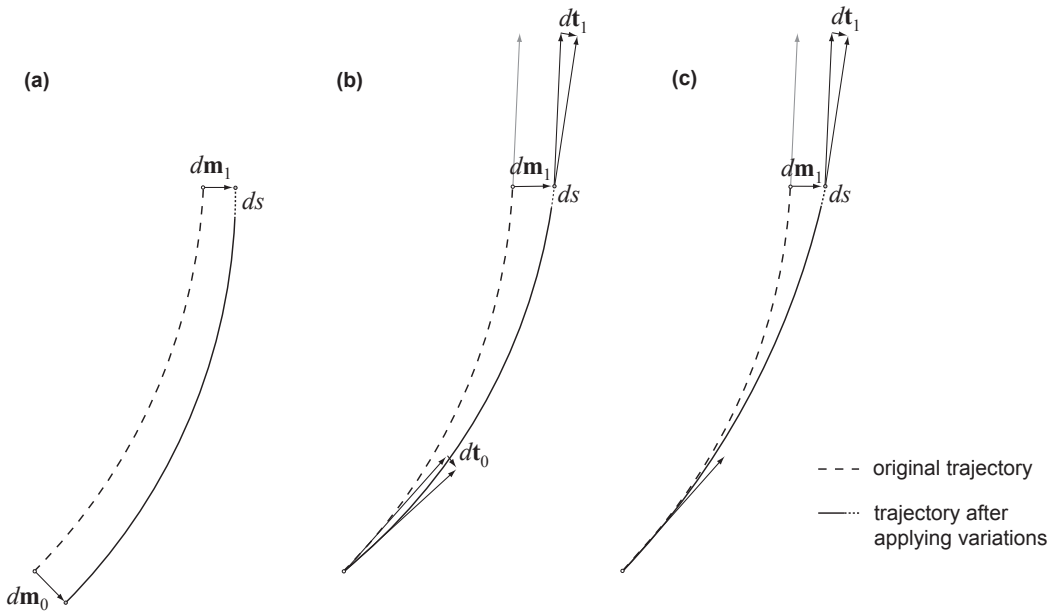


Figure 20: The initial variations on the track state and the directly resulting variations at the target position of the extrapolation process. The illustration decouples the possible start parameter variations: **(a)** illustrates the effect of an initial variation $d\mathbf{m}_0$ at the starting position, while **(b)** shows the effect of a slightly modified momentum direction; **(c)** displays the changed trajectory for a modified curvature while keeping the same initial direction.

$$\frac{\partial \mathbf{t}}{\partial s}\delta s = \alpha \cdot Q \cdot \mathbf{n} \cdot \delta s \tag{51g}$$

We used that $\gamma = \mathbf{h}\cdot\mathbf{t} = \mathbf{h}\cdot\mathbf{t}_0$ remains constant along s and so does α. Let us in addition mention that Eq. (51f) follows directly from Eq. (51c) and that Eq. (51g) represents the centripetal acceleration caused by the Lorentz force.

We search for transport jacobian matrix for the errors on the initial curvilinear parameters $\mathbf{c}_i = (u_i, v_i, \phi_i\,, \theta_i\,, 1/p_i)^T$ and the final curvilinear parameters $\mathbf{c}_f = (u_f, v_f, \phi_f\,, \theta, 1/p_f)^T$. In other words, we look for the coefficients C_{if} that transform a variation of the initial parameter parameter set c_i to the final parameters set $\mathbf{c}_f$. As an example we will specify these components for the variations on δu_f:

$$\delta u_f = \frac{\partial u_f}{\partial u_i}\delta u_i + \frac{\partial u_f}{\partial v_i}\delta v_i + \frac{\partial u_f}{\partial \phi_i}\delta \phi_i + \frac{\partial u_f}{\partial \theta_i}\delta \theta_i + \frac{\partial u_f}{\partial (1/p_0)}\delta(1/p_0). \tag{52}$$

The track parameters q/p as defined through the ATALS tracking EDM is hereby replaced by the $1/p$ since the variations on the signed momentum representation is restricted to change the momentum magnitude only, while leaving the charge parameter untouched. In any curvilinear frame, the total differentials $d\mathbf{m}$ and $d\mathbf{t}$ can be expressed with respect to the curvilinear coordinates and are then given by[19]

$$d\mathbf{m} = \mathbf{e}_u\,\delta u + \mathbf{e}_v\,\delta v \tag{53}$$

$$d\mathbf{t} = \frac{\partial \mathbf{t}}{\partial \phi}\delta\phi + \frac{\partial \mathbf{t}}{\partial \theta}\delta\theta = \sin\theta \mathbf{e}_u\,\delta\phi - \mathbf{e}_v\,\delta\theta \tag{54}$$

Equation (54) can be shown when differentiating $\mathbf{t}$ with respect to ϕ and θ and using the fact that the curvilinear frame is constructed by using the global z axis.

Without losing generality, we will in the following use Eqs. (47) to (51) assuming that the initial frame is given at $s = 0$, i.e. $\mathbf{m}_i = \mathbf{m}_0$ and denote $\psi_{if} = \psi_f - \psi_i \equiv \psi$ and $s_{if} \equiv s$ for convenience.

[19] Note that the orthogonality relation as required by Eq. (48) is hereby met by restricting the total differential to a variation in $\mathbf{e}_u$ and $\mathbf{e}_v$, respectively.

After inserting Eqs. (53) and (54) into Eq. (49) and using the partial derivatives found in Eq. (51) we gain an expression that correlates the variations on the local coordinates in the final curvilinear frame with the variations of the start parameters

$$\begin{aligned}
\mathbf{e}_u^f \cdot \delta u_f + \mathbf{e}_v^f \cdot \delta v_f &= \mathbf{e}_u^i \cdot \delta u_i \\
&+ \mathbf{e}_v^i \cdot \delta v_i \\
&+ \tfrac{\sin\theta_i}{Q}\left[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)\mathbf{h} + \sin\psi\mathbf{e}_u^i + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\right]\cdot\delta\phi_i \\
&- \tfrac{1}{Q}\left[\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)\mathbf{h} + + \sin\psi\mathbf{e}_v^i + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\right]\cdot\delta\theta_i \\
&+ p\cdot[s\cdot\mathbf{t}_f + \mathbf{m}_i - \mathbf{m}_f]\cdot\delta(1/p_i) \\
&+ \mathbf{t}\cdot\delta s.
\end{aligned} \tag{55}$$

This equation is multiplied by $\mathbf{e}_u^f$ to isolate the terms given in Eq. (52), since it cancels the contributions in the orthogonal component $\mathbf{t}_f \equiv \mathbf{e}_t^f$. This yields

$$\begin{aligned}
\delta u_f &= \mathbf{e}_u^i \cdot \mathbf{e}_u^f\,\delta u_i \\
&+ \mathbf{e}_v^i \cdot \mathbf{e}_u^f\,\delta v_i \\
&+ \tfrac{\sin\theta_i}{Q}\left[(\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_u^f) + \sin\psi(\mathbf{e}_u^i\cdot\mathbf{e}_u^f) + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\right]\cdot\delta\phi_i \\
&- \tfrac{1}{Q}\left[\psi - \sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_u^f) + + \sin\psi(\mathbf{e}_v^i\cdot\mathbf{e}_u^f) + (1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_u^f\right]\cdot\delta\theta_i \\
&+ p\cdot\left[(\mathbf{m}_i\cdot\mathbf{e}_u^f) - (\mathbf{m}_f\cdot\mathbf{e}_u^f)\right]\cdot\delta(1/p_i).
\end{aligned} \tag{56}$$

By comparing the coefficients of Eq. (56) with those defined in Eq. (52), the first row of the Jacobian matrix is found and since Eq. (55) is on the left hand side symmetrical in $\mathbf{e}_u$ and $\mathbf{e}_v$, a similar equation for δv_f can be obtained. For evaluating the covariance entries that regulate the transport of the directional uncertainties we take Eq. (50) and insert again both, the partial derivatives as given in Eq. (51) and the expression of the total differential in the curvilinear frame coordinates. This yields the expression

$$\begin{aligned}
\sin\theta_f\,\mathbf{e}_u^f\,\delta\phi_f - \mathbf{e}_v^f\,\delta\theta_f &= \\
&+ \sin\theta_i\left[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)\mathbf{h} + \cos\psi\mathbf{e}_u^i + \sin\psi(\mathbf{h}\times\mathbf{e}_u^i)\right]\delta\phi_i \\
&- \left[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)\mathbf{h} + \cos\psi\mathbf{e}_v^i + \sin\psi(\mathbf{h}\times\mathbf{e}_v^i)\right]\delta\theta_i \\
&+ \alpha\cdot Q\cdot p\cdot s\delta(1/p_i) \\
&+ \alpha\cdot Q\cdot\mathbf{n}\cdot\delta s
\end{aligned} \tag{57}$$

We first try to eliminate the explicit variation on the propagation length δs from Eq. (57) before we can proceed identically as for the local coordinates. Taking Eq. (49) and multiply $d\mathbf{m}_f$ with the perpendicular direction $\mathbf{t}_f$ gives an expression for δs as

$$\delta s = -d\mathbf{m}_i\cdot\mathbf{t}_f - \left[\frac{\partial\mathbf{m}_f}{\partial\mathbf{t}_i}d\mathbf{t}_i\right]\cdot\mathbf{t}_f - \left[\frac{\partial\mathbf{m}_f}{\partial(1/p_i)}\cdot\mathbf{t}_f\right]\cdot\delta(1/p_i), \tag{58}$$

which we insert in Eq. (57). When multiplying Eq. (58) again with $\mathbf{e}_u^f$ and $\mathbf{e}_v^f$ and and isolating the components, the third and fourth row — describing the propagation of the directional uncertainties — of the transport Jacobian is determined. The missing terms of the Jacobian matrix can be found without additional calculations; they are

$$\frac{\partial(1/p_f)}{\partial(1/p_i)} = 1 \tag{59}$$

and, respectively,

$$\frac{\partial(1/p_f)}{\partial(u_i, v_i, \phi_i, \theta_i)} = 0 \tag{60}$$

and correspond to momentum conservation and an interaction-free transport.

The fully deployed Jacobian matrix describing the transport from one curvilinear frame to another can be found in Fig. 21.

$$\mathbf{J}_{ci,cf} =$$

$$\left(\begin{array}{c|c|c|c|c}
\mathbf{e}_u^f \cdot \mathbf{e}_u^i & \mathbf{e}_u^f \cdot \mathbf{e}_v^i & \begin{array}{c}-\frac{\sin\theta_i}{Q}[(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_u^f) \\ +\sin\psi(\mathbf{e}_u^i\cdot\mathbf{e}_u^f) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{e}_u^f)]\end{array} & \begin{array}{c}\frac{1}{Q}[(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_u^f) \\ +\sin\psi(\mathbf{e}_v^i\cdot\mathbf{e}_u^f) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_u^f)]\end{array} & p\cdot(\mathbf{m}_i-\mathbf{m}_f)\mathbf{e}_u^f \\ \hline
\mathbf{e}_v^f \cdot \mathbf{e}_u^i & \mathbf{e}_v^f \cdot \mathbf{e}_v^i & \begin{array}{c}-\frac{\sin\theta_i}{Q}[(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_v^f) \\ +\sin\psi(\mathbf{e}_u^i\cdot\mathbf{e}_v^f) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{e}_v^f)]\end{array} & \begin{array}{c}\frac{1}{Q}[(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_v^f) \\ +\sin\psi(\mathbf{e}_v^i\cdot\mathbf{e}_v^f) \\ +(1-\cos\psi(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_v^f)]\end{array} & p\cdot(\mathbf{m}_i-\mathbf{m}_f)\mathbf{e}_v^f \\ \hline
\frac{\alpha\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}_u^f)(\mathbf{e}_u^i\cdot\mathbf{t}_f) & \frac{\alpha\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}_u^f)(\mathbf{e}_v^i\cdot\mathbf{t}_f) & \begin{array}{c}\frac{\sin\theta_i}{\sin\theta_f}[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_u^f) \\ +\cos\psi(\mathbf{e}_u^i\cdot\mathbf{e}_u^f)+\sin\psi(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{e}_u^f \\ -\alpha(\mathbf{n}\cdot\mathbf{e}_u^f)\{(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}_u^i) \\ +\sin\psi(\mathbf{e}_u^i\cdot\mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{t}_f\}]\end{array} & \begin{array}{c}-\frac{1}{\sin\theta_f}[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_u^f) \\ +\cos\psi(\mathbf{e}_v^i\cdot\mathbf{e}_u^f)+\sin\psi(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_u^f \\ -\alpha(\mathbf{n}\cdot\mathbf{e}_u^f)\{(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}_v^i) \\ +\sin\psi(\mathbf{e}_v^i\cdot\mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{t}_f\}]\end{array} & \begin{array}{c}\frac{\alpha\cdot p\cdot Q}{\sin\theta_f}(\mathbf{n}\cdot\mathbf{e}_u^f) \\ \cdot[\mathbf{t}_f(\mathbf{m}_i-\mathbf{m}_f)]\end{array} \\ \hline
-\alpha\cdot Q(\mathbf{n}\cdot\mathbf{e}_v^f)(\mathbf{e}_u^i\cdot\mathbf{t}_f) & -\alpha\cdot Q(\mathbf{n}\cdot\mathbf{e}_v^f)(\mathbf{e}_v^i\cdot\mathbf{t}_f) & \begin{array}{c}-\sin\theta_i[(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_u^i)(\mathbf{h}\cdot\mathbf{e}_v^f) \\ +\cos\psi(\mathbf{e}_u^i\cdot\mathbf{e}_v^f)+\sin\psi(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{e}_v^f \\ -\alpha(\mathbf{n}\cdot\mathbf{e}_v^f)\{(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}_u^i) \\ +\sin\psi(\mathbf{e}_u^i\cdot\mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}_u^i)\mathbf{t}_f\}]\end{array} & \begin{array}{c}(1-\cos\psi)(\mathbf{h}\cdot\mathbf{e}_v^i)(\mathbf{h}\cdot\mathbf{e}_v^f) \\ +\cos\psi(\mathbf{e}_v^i\cdot\mathbf{e}_v^f)+\sin\psi(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{e}_v^f \\ -\alpha(\mathbf{n}\cdot\mathbf{e}_v^f)\{(\psi-\sin\psi)(\mathbf{h}\cdot\mathbf{t}_f)(\mathbf{h}\cdot\mathbf{e}_v^i) \\ +\sin\psi(\mathbf{e}_v^i\cdot\mathbf{t}_f) \\ +(1-\cos\psi)(\mathbf{h}\times\mathbf{e}_v^i)\mathbf{t}_f\}\end{array} & \begin{array}{c}-\alpha\cdot p\cdot Q(\mathbf{n}\cdot\mathbf{e}_v^f) \\ \cdot[\mathbf{t}_f(\mathbf{m}_i-\mathbf{m}_f)]\end{array} \\ \hline
0 & 0 & 0 & 0 & 1
\end{array}\right)$$

Figure 21: The full deployed Jacobian matrix for the transportation of errors on the track parameters between two curvilinear frames on a track for a helical track model. For a straight line track model, the Jacobian matrix reduces quite substantially.

Curvilinear to Local Transformations and vice versa For a full transport of the covariance matrix between to arbitrary track representations it is also necessary to determine the Jacobian matrices that transform the local parameter expressions $\mathbf{x} = (l_1, l_2, \phi, \theta, q/p)^T$ to the curvilinear parameters $\mathbf{c} = (u, v, \phi, \theta, 1/p)$ and the reverse way. In ATLAS, all track representations can be expressed on a planar surface, and including an additional transformation, even in a two-dimensional cartesian coordinate system, which thus remains the only case to be investigated in this scope (this is guaranteed through the *measurement frame* mechanism, see [6]).

We use the definition of a lateral variation in the curvilinear frame — Eq. (53) — and express it through a variation in the local coordinates and along the momentum

$$d\mathbf{m} = \mathbf{e}_u\, \delta u + \mathbf{e}_v\, \delta v = \mathbf{e}_1 \delta l_1 + \mathbf{e}_2 \delta l_2 + \mathbf{t} \cdot \delta s, \tag{61}$$

and redo the same for the variation on the momentum

$$d\mathbf{t} = \sin\theta \mathbf{e}_u\, \delta\phi - \mathbf{e}_v\, \delta\theta = \sin\theta \mathbf{e}_u\, \delta\phi - \mathbf{e}_v\, \delta\theta + \frac{\partial \mathbf{t}}{\partial s}\delta s. \tag{62}$$

It is worth mentioning that although the directional expression is identical in both local and curvilinear frame, the angular parameters ϕ, θ have to be first treated independently since we want to investigate both sides individually.

Multiplying Eq. (61) with the orthogonal vector $\mathbf{t}$ (which eliminates components in $\mathbf{e}_u$ and $\mathbf{e}_v$) and solving for δs establishes an expression of the necessary[20] variation in the propagation length caused by a local variation on the starting surface

$$\delta s = -(\mathbf{t} \cdot \mathbf{e}_1)\delta l_1 - (\mathbf{t} \cdot \mathbf{e}_2)\delta l_2, \tag{63}$$

while the multiplication with $\mathbf{e}_u$ and $\mathbf{e}_v$, respectively, yields the expression of the variations in δu and δv

$$\begin{aligned} \delta u &= (\mathbf{e}_u \cdot \mathbf{e}_1)\delta l_1 + (\mathbf{e}_u \cdot \mathbf{e}_2)\delta l_2 \\ \delta v &= (\mathbf{e}_v \cdot \mathbf{e}_1)\delta l_1 + (\mathbf{e}_v \cdot \mathbf{e}_2)\delta l_2. \end{aligned} \tag{64}$$

The coefficients of the Jacobian matrix that relate the local surface parameters with the curvilinear parameters u and v can be directly gained from Eq. (64). In a second step, Eq. (62), is multiplied by $\mathbf{e}_u$ and $\mathbf{e}_v$ and Eq. (63) is inserted to eliminate δs. This yields the additional components that relate the angular variables with the local surface parameters.

The reverse transformation from the (transported) curvilinear frame to the local frame of the target surface can be found in a similar way, starting from Eq. (61). We clearly want to evaluate the variations hereby in the target frame, i.e. we demand that $d\mathbf{m}$ is orthogonal to $\mathbf{e}_3$ in this case, which builds the normal vector of the target surface. Under this assumption, the multiplication of Eq. (61) with $\mathbf{e}_3$ yields

$$\delta s = \frac{\mathbf{e}_u \cdot \mathbf{e}_3}{\mathbf{t} \cdot \mathbf{e}_3}\delta u + \frac{\mathbf{e}_v \cdot \mathbf{e}_3}{\mathbf{t} \cdot \mathbf{e}_3}\delta v. \tag{65}$$

Introducing Eq. (65) to Eq. (61) now enhances the same procedure to find the components for the inverse Jacobian matrix. The fully deployed matrices can be found in the code documentation of the TrkExUtils CVS package [30].

References

[1] R. Frühwirth et al., *Application of Kalman Filtering to Track and Vertex Fitting*, Nucl. Inst. Meth., **A 262**, 1987.

[2] V. Boisvert et al, *Final Report of the ATLAS Reconstruction Task Force (RTF)*, ATLAS Note, ATL-SOFT-2003-01, 2003.

[3] Athena homepage, *http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/.*

[20] To comply with the orthogonality $d\mathbf{m} \cdot \mathbf{t} = 0$.

[4] ATLAS Quality Assurance Group, *Atlas C++ Coding Standard Specifications*, ATLAS Note, ATL-SOFT-2002-001, 2002.

[5] Atlas Collaboration, *ATLAS Computing Technical Design Report*, ATLAS TDR, CERN-LHCC-2005-022, 2005.

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

[7] A. Salzburger, S. Todorova and M. Wolter, *The Atlas Tracking Geometry Description*, ATLAS communication to be published, ATL-COM-SOFT-2007-009, 2007.

[8] T. Cornelissen et al., *Updates of the ATLAS Tracking Event Data Model*, ATLAS communication to be published, ATL-COM-SOFT-2007-008, 2007.

[9] J.R. Dormand and P.J. Price, *A family of embedded Runge-Kutta formulae*, J. Comp. Appl. Math.,**6**, 1980.

[10] A. Salzburger, *The new Fast ATLAS Track Simulation (FATRAS)*, Proceeding of CHEP 2006, 2006.

[11] A. Strandlie and W. Wittek, *Derivation of Jacobians for the propagation of covariance matrices of track parameters in homogeneous magnetic fields*, Nucl. Inst. & Meth. in Phys., **A 566**, 2006.

[12] E. Lund, L. Bugge, A. Strandlie and I. Gavrilenko, *Simulatneous Track and Error Propagation with Material Effects*, ATLAS Note in preparation

[13] V.L. Highland, *Some Practical Remarks on Multiple Scattering*, Nucl. Inst. & Meth. **129**, 1975, and *Erratum*, Nucl. Inst. & Meth., **161**, 1979.

[14] G. Moliére, *Therorie der Streuung schneller geladener Teilchen I. Einzelstreuung am abgeschirmten Coulomb-Feld*, Z. Naturforschung **2a**, 1947, and *Theorie der Streuung schneller geladener Teilchen II. Mehrfach- und Vielfachstreuung*, Z. Naturforschung **3a**, 1948.

[15] B. Rossi and K. Greisen, *Cosmic-Ray Theory*, Rev. Mod. Phys.,**13**, 1941.

[16] R. Frühwirth, M. Regler, *On the quantitative modeling of core and tails of multiple scattering by Gaussian mixtures*, Nucl. Inst. & Meth. **A 456**, 2001.

[17] R. Frühwirth, M. Liendl, *Mixture models of multiple scattering: computation and simulation*, Comp. Phys. Comm. textbf141, 2001.

[18] Agostinelli et al.,*GEANT4: A Simulation toolkit*, Nucl. Inst. & Meth., **A 506**, 2003.

[19] H. Bethe, *Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie*, Annalen der Physik, **397**, 1930.

[20] Lohmann W., Kopp R. Voss, R., *Energy loss of muons in the energy range 1-10 000 GeV*, CERN-85-03 Yellow Reoprt, 1985.

[21] D.E. Groom, N.V. Mokhov and S.I. Striganov, *Muon stopping power and range tables 10 MeV-100 TeV*, Atomic Data and Nuclear Tables, **78**, 2001.

[22] Particle Data Group, *Review of Particle Physics*, Phys. Rev., **D 66**, 2002.

[23] L.D. Landau, *On the energy loss of fast particles by ionisation*, J. Phys. USSR, **8**, 1944.

[24] H. Bethe and W. Heitler, *On the stopping of fast particles and the creation of positive electrons*, Proc. Roy. Soc.**A 146**, 1934.

[25] T.M. Atkinson, *Electron Reconstruction with the ATLAS Inner Detector*, PhD thesis, University of Melbourne, 2006.

[26] Muonboy homepage, *http://cern.ch/atlas-samusog/muonboy/Muonboy.htm.*

[27] C.J.F. Ridders, *Advances in Engineering Software*, **Vol. 4**, 1982.

[28] T.G. Cornelissen, *Track Fitting in the ATLAS Experiment*, CERN-THESIS-2006-07, 2007.

[29] A. Salzburger (Editor) et. al., *Concepts, Design and Implementation of the ATLAS New Track Reconstruction (NEWT)*, ATLAS Communication to be published, ATLAS-COM-SOFT-2007-002, 2007.

[30] ATLAS software CVS repository, online CVSView, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*

Long is the road
from conception
to completion.

Molière

Chapter 8

The New Modular Track Reconstruction

The preparation of track reconstruction software has been an ongoing field of activity in ATLAS for more than 15 years, starting with first track reconstruction packages that have been written in FORTRAN and which were integrated in the former ATLAS reconstruction framework ATRECON. Several competing packages existed at that time for the two ATLAS tracking devices, the *Inner Detector* (ID) and the *Muon Spectrometer* (MS). These monolithic package have been mostly single-author driven and each of which incorporated an individual event data model, a separate geometry description and specific algorithmic steering. When ATLAS moved to the new C++ based object-oriented software framework ATHENA [8.1], which is an enhanced version of the GAUDI [8.2] project that has been developed by the LHCb collaboration, these well performing and optimised applications have indeed been ported to C++, while the main block structure remained unchanged. In 2003, a dedicated reconstruction task force has realised the potential danger of this software model [8.3], in particular in conjunction with the necessary adaption of the ATLAS track reconstruction applications to a more realistic detector setup (including misalignment, distortions and an excessive use of detector conditions data). The inclusion of these components is inevitable for the readiness of the reconstruction applications for first data taking. These extensions would have been necessary to be integrated separately in the several reconstruction packages, since a common interface level has not been compliant with their software model.

8.1 Introduction

During the last four years, a new track reconstruction — also referred to as *New Tracking* (NEWT) — has been deployed that is based on the common tracking EDM and a coherent interface model, which enhances a pure component software structure. A lot of well performing algorithmic code from the former reconstruction programs has been integrated into NEWT, but also new development has been facilitated. NEWT has been established as the default ATLAS ID reconstruction and many components of NEWT are also used in track reconstruction algorithms of the Muon Spectrometer, and, furthermore, in combined reconstruction and physics analyses. In this sense, NEWT is not yet another track reconstruction program for the ATLAS experiment, but rather a collective term for the new philosophy deployed in the current track reconstruction algorithms, that are in ATLAS — for the first time — based on a common model for both tracking devices. NEWT runs in similar sequences that access mostly common tools in the reconstruction of test beam and commissioning data

using cosmic rays, as well as in the offline reconstruction and the seeded third level trigger, the so-called *Event Filter* (EF).

Section 8.2 will give an exhaustive description of the components and modules that build the new reconstruction chain. It will also focus on its coherent integration into the global ATLAS reconstruction framework. Section 8.3 will then give an overview on the track reconstruction performance based on a study of simulated single track events.

8.2 The ATLAS New Track Reconstruction (NEWT)

ATL-SOFT-PUB-2007-002, 25 p.

Contributions of the Author

The establishment of a full reconstruction chain on such level needs the contribution and interaction of several authors; many of the working fields overlap hereby and a clear discrimination of single unique contributions is difficult to draw. The author's contribution to the establishment of NEWT includes besides the already described components of the event data model, the reconstruction geometry and the extrapolation engine the general design of the algorithmic flow, specific contributions to the integration of the deployed track fitters, the hole search for ambiguity solving and to the validation framework.

Concepts, Design and Implementation of the ATLAS New Tracking (NEWT)

T. Cornelissen, M. Elsing, I. Gavrilenko
CERN
S. Fleischmann
University of Bonn, Germany
W. Liebig
NIKHEF, Amsterdam, The Netherlands
E. Moyse
University of Massachusetts, USA
A. Salzburger (Editor)*
Leopold Franzens Universität Innsbruck, Austria & CERN

for the ATLAS Inner Detector Software Group

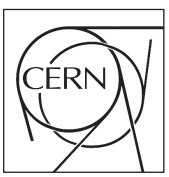
December 17, 2007

ATL-SOFT-PUB-2007-007
17 December 2007

Abstract

The track reconstruction of modern high energy physics experiments is a very complex task that puts stringent requirements onto the software realisation. The ATLAS track reconstruction software has been in the past dominated by a collection of individual packages, each of which incorporating a different intrinsic event data model, different data flow sequences and calibration data. Invoked by the *Final Report of the Reconstruction Task Force*, the ATLAS track reconstruction has undergone a major design revolution to ensure maintainability during the long lifetime of the ATLAS experiment and the flexibility needed for the startup phase. The entire software chain has been reorganised in modular components and a common *Event Data Model* has been deployed during the last four years. A complete new track reconstruction that concentrates on common tools aimed to be used by both ATLAS tracking devices, the Inner Detector and the Muon System, has been established. It has been already used during many large scale tests with data from Monte Carlo simulation and from detector commissioning projects such as the combined test beam 2004 and cosmic ray events. This document concentrates on the technical and conceptual details of the newly developed track reconstruction, also referred to as *New Tracking*.

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*



*corresponding author: Andreas.Salzburger@cern.ch

1 Introduction

The event reconstruction of modern high energy physics experiments is a very complex task but indispensable for any analysis of the underlying physics process. At the LHC with its design luminosity of 10^{34} $\mathrm{cm}^{-2}\mathrm{s}^{-1}$ and multiple overlaying proton-proton collisions the resulting high track density puts stringent challenges to the track reconstruction software. The ATLAS detector consists of two independent tracking devices: the *Inner Detector* (ID) close to the interaction region — realised as the pixel detector at innermost radii, *Semiconductor Tracker* (SCT) using silicon strips, and a *Transition Radiation Tracker* (TRT) — and the *Muon System* (MS), that combines *Monitored Drift Tubes* (MDT), *Cathode Strip Chambers* (CSC), *Thin Gap Chambers* (TGC), and *Resistive Plate Chambers* (RPC). While the Inner Detector reconstruction has to deal with the high track density that imposes a large number of combinatorial track candidates, the Muon System track reconstruction is mainly limited by the huge amount of inert material, the cavern background and the highly inhomogeneous magnetic field.

In the past[1], the ATLAS track reconstruction software consisted for both, ID and MS, of several competing reconstruction programs, each of which incorporating its own data model, different reconstruction geometries, varying concepts in material integration and separate philosophies in algorithmic sequence and steering[2]. Performance comparisons between the various programs on different levels of the reconstruction chain have been almost impossible. New developments in terms of calibration, conditions or alignment data that are needed for a more realistic description of the ATLAS detector had to be integrated separately into the existing software packages, since the use of common tools was not incorporated into the underlying software design.

The long lifetime of the ATLAS experiment, however, requires a reconstruction software that is flexible, extendable, easy to maintain or change, whilst keeping the performance at the highest and the CPU time consumption at the lowest possible level. The *ATLAS Reconstruction Task Force* (RTF) recognised in 2003 [1] the potential danger of sticking with the monolithic, and sometimes even single-author software structure, and developed a guideline for the transition of the ATLAS reconstruction software to a modern, modular pattern that is based on common interfaces and a shared *Event Data Model* (EDM). During the last four years this shared EDM has been developed and established [2], which builds a cornerstone of the ATLAS *New Tracking* (NEWT). Given the modular software structure, *New Tracking* is not a name for yet another reconstruction program, but rather a collective term for the philosophy that has been deployed throughout the new track reconstruction software. A lot of the existing and well performing code that has been part of the past reconstruction programs has been integrated as components into the new modular design. Currently, NEWT spans over about 250 software packages, located in the common Tracking repository as well as in the associated sub-detector repositories, and concentrates the work of many developers. NEWT has been developed in full coherence with the ATLAS software framework ATHENA and respects ATLAS coding standards [3]. This document is based on the ATLAS software release 12.0.6. Developments that have been integrated after this release are clearly marked within the context.

1.1 Document Structure

This document is organised as follows: Section 2 gives an introduction to the ATHENA framework, concentrating on the concepts and modules used in the realisation of New Tracking, Sec. 3 focuses on the high level structure of the ATLAS New Tracking in both algorithm sequence and data flow; additionally, an overview of the most common Tracking interfaces and tools is given. Section 4 describes the track reconstruction strategies and their implementation in the current ID New Tracking. Section 5 describes the usage of the common tools in the Muon System and in the context of muon combined reconstruction. Finally, Sec. 6 will give a conclusion, but will also present an outlook of the ongoing evolution of the New Tracking realm. The appendix explains typesetting and gives a brief overview on the used software packages such as an index of used abbreviations within this document for the orientation of the reader.

[1] Starting from the design phase of the ATLAS experiment.

[2] Commonality has not even been met in the used programming language, as for historical reasons these track reconstruction programs have been written in `C`, `C++`, `Fortran` or even a mixture of all three.

2 The embedding Software Framework ATHENA

The main concept of NEWT is to factorise the complex process of track reconstruction into well defined modules that represent a single task or a grouped operation. A common interface structure for these modules has been defined and the EDM acts hereby as the language between the different components. This allows single parts of the entire reconstruction process to be modified or exchanged without disrupting the untouched parts of the software chain.

To understand the structure of the ATLAS New Tracking, it is necessary to be familiar with the main concepts of the underlying software framework, not necessarily on a detailed technical, but on an abstract level. The following section should give the reader enough knowledge to understand the concepts of algorithmic sequences and the overall data flow, while attempting to minimise the purely technical aspects of the ATLAS software structure. The reader is, however, encouraged to read further in [4], Sec. 3.3, where a complete overview of the ATHENA framework is given.

2.1 The ATHENA Framework and the Component Architecture

The ATLAS ATHENA software framework is an enhanced version of the original C++ based software framework GAUDI [5] that was initially developed by the LHCb collaboration. Nowadays, the two projects have been merged to a joint ATLAS-LHCb project, sometimes referred to as GAUDI-ATHENA.

2.1.1 The ATHENA Component Pattern Architecture

The component model is a very common software architecture of large-scale software projects. Commonly used interfaces are defined and allow various alternative realisations — i.e. in software terms, different implementations — of the same task. The component library structure allows that these alternative modules are loaded as shared libraries at job configuration level. This leads to a flexible software structure and reduces in addition dependencies between the various libraries used in the final executable, which in turn increases the stability and decreases the complexity of the build process of the ATLAS software[3]. Further information about the ATLAS software model can be found in [6].

The ATHENA framework provides a set of defined interfaces at different levels in the program sequence that also build the base pattern of the ATLAS New Tracking:

- The `Service` class is designed to provide dedicated functionality during the entire program execution, e.g. the central data storages (transient event store or the detector store) are realised as ATHENA `Service` objects. `Service` instances are handled by a central `ExtSvc` manager, that regulates initialisation and finalisation.
- The `Algorithm` class is dedicated for actions to be taken exactly one time for every processed event, e.g. most of the data preparation algorithms are realised as `Algorithm` classes. These have to be registered to a central `ApplicationMgr` in the job configuration that steers initialisation, finalisation and the execution of the `Algorithm` at every event.
- Unlike the `Algorithm` class provides the `AlgTool` are more flexible solution for repeated operations, mainly called through an `Algorithm` that either owns the associated `AlgTool` (i.e. consequently a *private* `AlgTool`) or retrieves it from the central `ToolSvc`, where all *public* tools are registered. This pattern allows `AlgTool` instances to be shared between different applications, such as e.g. the same `Extrapolator AlgTool` instance is used at several places within the reconstruction process.

In general, the `Algorithm` is responsible for retrieving input data collections from and writing the output data to the transient event store, which is realised through the ATHENA `Service StoreGateSvc`. The actual operations are usually not performed by the `Algorithm`, but delegated to `AlgTool` units, that can be shared between different tasks. Per event, the various `Algorithm` instances are called one

[3] In the ATLAS computing model, test builds of the entire software suite take place on a daily basis to allow a coherent and parallel code development of the multiple authors. Additionally, it increases the frequency of the development cycles and allows large scale evolutions of the software in a gradual way.

after the other by the `ApplicationMgr` through which the reconstruction sequence is defined. There is no additional dependency between the different `Algorithm` classes rather than they rely on the input data to be existing and retrievable through the `StoreGateSvc`. Following [7], this data centered architecture will be further on referred to as *blackboard architecture style*, where the `StoreGateSvc` acts as the blackboard to which the clients write to or read from (in ATHENA, this is represented by the templated `StoreGateSvc::retrieve()`, respectively `StoreGateSvc::record()` interface). The `ApplicationMgr` plays the role of a *controller* (teacher) in this model and organises the reading and writing to and from the blackboard. The result of the blackboard design can be seen as a *pseudo data flow*: in an abstract picture the data objects are handed over from one `Algorithm` to the next one in the reconstruction sequence, while in reality the data exchange always progresses via the blackboard.

Figure 1 shows a schematic UML sequence diagram for a sample usage of the three main framework components in a simple processing example.

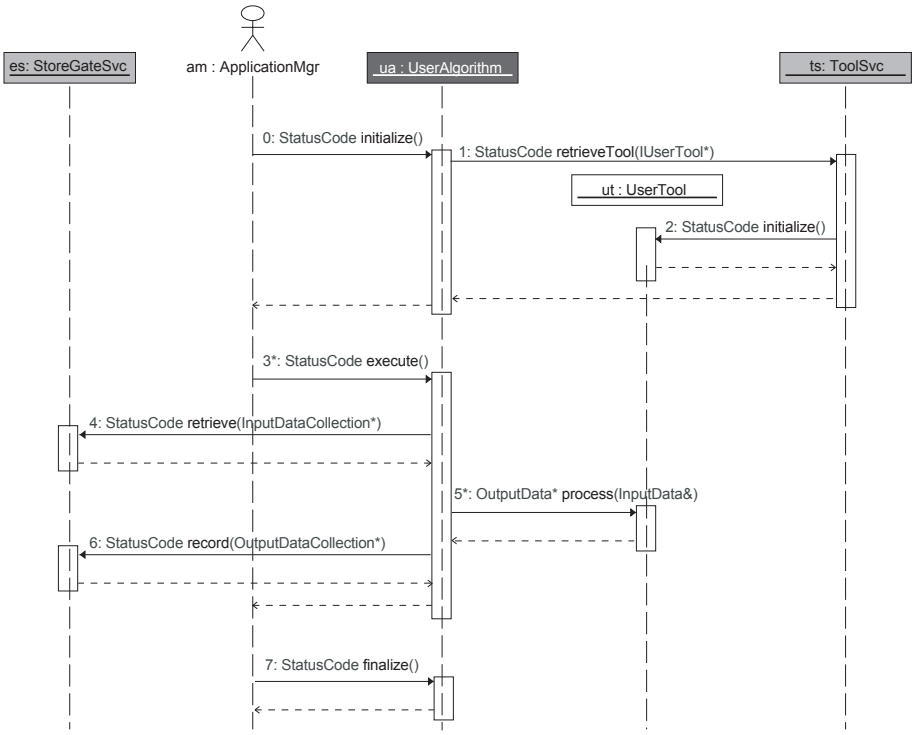


Figure 1: A simplified UML sequence diagram for the data flow and sequence steering in the ATHENA framework. Every `Algorithm` is registered to the `ApplicationMgr` that calls the `initialize()` interface method at job startup, the `execute()` for every event and the `finalize()` method in the job termination phase. The `initialize()` call to the `Algorithm` may trigger the retrieval of an `AlgTool` from the `ToolSvc`. The main framework methods are templated and return a `StatusCode` object that indicates the success state of the performed operation.

2.1.2 Data Factory Pattern and Encapsulation of algorithmic Code

A main concept on the ATLAS computing model — and a natural consequence of the blackboard design — is a clear and strict separation between data classes and algorithmic code, that is mainly carried out by `AlgTool` classes. Operations are performed on input data classes, and newly created output data is hereby produced. In general, modifications to the input data are not allowed, but the modification of an object leads rather to the creation of a new one. In C++ terms, these new objects are dynamically created using the `new` operator. This concept is intrinsically imposed by the blackboard framework design that is concentrated around a central data store service: allowing modifications of an object would change the object immediately at every appearance in the reconstruction chain and since there is no relation between the different `Algorithm` modules, a direct manipulation of EDM objects could affect any decoupled `Algorithm` that uses the same input sample without being

triggered. This is the reason why the tracking EDM classes in general do not have modification or set methods which are commonly used in many other application libraries[4]. This model will in the following be referred to as *data factory pattern*.

3 Concepts and Common Components of the New Tracking

In a complex software project of large scale that serves both users and developers on different levels of interaction, it is inevitable that common structures and patterns are identified and named, such that an intuitive guideline through the — for the single user — almost unmanageable matrix of components is present[5]. Section 3.1 tries in the following to define these common structures in an abstract way, while Sec. 3.2 concentrates on the actual used common components in the context of the ATLAS New Tracking reconstruction.

3.1 Reconstruction Sequences, Modules, Tasks and Operations

In the following, the components of the track reconstruction software will be classified as sequences, modules, tasks and operations:

- A *sequence* specifies a complete process of several complex reconstruction steps leading to high level output objects; track finding with two strategies may be performed through two different sequences, established as different `Algorithm` chains.

- A *module* denotes, in general, the complete chain of retrieving input data from the transient event store, processing the data, and the successive recording of the output data. Most data preparation processes are realised as modules; another example would be track refitting, where a complete track collection is retrieved from the `StoreGateSvc` and a new collection of tracks is recorded to it after refitting with a track fitter. Modules are in general realised as `Algorithm` classes with their associated access to the transient event store.

- A *task* describes on the other hand a well-defined execution without direct interaction with the algorithm sequence. The refit of a track, e.g. is a task in this scope and a module consists usually of one to several tasks. Tasks are often performed by `AlgTool` classes which are directly called by an `Algorithm`; such `AlgTool` instances will be in the following called *at first level*.

- The lowest component to be mentioned in this context is an *operation*, usually performed by an `AlgTool` object of lower level. Operations are often shared between tasks (and a task is in general built of several operations) and performed several times, but do — contrary to a task — not lead to a direct output data object. The extrapolation of a track representation to surfaces, which is needed in track fitting as well as in pattern recognition, vertex fitting and combined reconstruction is an example for such an operation.

Lower level algorithmic code or utilities are not forced into the `AlgTool` schema since this creates a quite substantial overhead to a simple class definition: small mathematical utility classes, functors for object sorting and access, but also little manipulator `friend` classes are therefore concentrated in dedicated utility packages that do not embody the component library pattern, but are of an installed library type[6].

[4] For low level objects that have not been written to the transient event store or those that are scheduled for immediate further processing it is, however, sometimes useful when small modifications are allowed without necessarily creating a new object and destroying the old one. In this case, dedicated helper classes that are in a C++ `friend` relation to the data objects can be called that perform the manipulation, while respecting the self-consistency of the object.

[5] The ATLAS offline release has already exceeded the number of 1000 component packages.

[6] The major difference between *component* libraries and *installed* libraries is their role in the linking process of the software build. While component libraries can be loaded dynamically at run time as shared libraries, installed libraries have to be linked against. In more illustrative words: when one deals with goods it is necessary to know their size and shapes, while it is not obligatory to about the dealer who delivers them.

3.2 Common Interfaces and Tracking `AlgTool` Classes

The framework `Algorithm` class builds the natural interface for a module, while operations and tasks impose a higher granularity and have to be defined by an interface structure that allows multiple actual implementations. In the ATLAS New Tracking, there exist two different types of interface definitions for `AlgTool` objects: *common interfaces* that define operations and tasks performed on base class level of the tracking EDM and *specific interfaces* that describe operations that are particular to a sub-detector. Common interface definitions are concentrated in the Tracking repository, while specific interfaces are spread out over the sub-detector repositories. Common interfaces do not necessarily imply that the implementations can be kept in an generic way or an abstract level. Moreover, some operations and tasks will be best performing if a very specific and for the sub-detector technology optimised way can be found. In this case, the actual `AlgTool` classes are located in the sub-detector repositories and the method interface is, if possible, overridden such that the base class EDM object is hidden by the concrete sub-detector implementation. Table 1 gives an overview of the locations of common interfaces defined in the Tracking repository.

Table 1: Locations and brief description of common interfaces concentrated in the Tracking repository. For future releases it is scheduled to move the track fitting interfaces into a dedicated TrkFitterInterfaces package to sustain a coherent naming schema.

Interface package	Description	Example
TrkExInterfaces	propagation, extrapolation, material effects	`IPropagator`
TrkDetDescrInterfaces	building of the `TrackingGeometry`	`IGeometryBuilder`
TrkMagFieldInterfaces	magnetic field access, parameterisations	`IMagneticFieldTool`
TrkFitterUtils	fitting interfaces and extensions	`ITrackFitter`
TrkToolInterfaces	truth processing, updator, ...	`IUpdator`
TrkValInterfaces	validation tools	`IResidualPullCalculator`
TrkVertexFitterInterfaces	vertex seed finding, fitting	`IVertexFitter`

The list of specific interfaces that define tasks and operations in the sub-detectors is wide-spread, every single `AlgTool` is represented through a dedicated interface even if only one single implementation is currently present[7]. This allows code development beyond the level of pure structured programming, but it also necessary for the component pattern design. Modularity is hereby achieved through the fact that all interactions between modules are kept purely on interface level.

3.3 The ATLAS Tracking Event Data Model and Reconstruction Geometry

A common EDM is inevitable for a modular software architecture. It allows the definition of abstract interfaces by identifying similar tasks to be performed through the method signature and the return type of the method, respectively. Different concrete implementations of the same abstract interface class may be created for e.g. different sub-detectors, realising a similar operation with different but optimised implementations.

During the last four years a common ATLAS tracking EDM has been deployed, concentrated around a very flexible and extensible `Track` class. The collection of `Track` objects is capable of holding the entire tracking information of the event. The polymorphic inheritance structure of the ATLAS tracking EDM enables the definition of tasks that can be performed on base class level, or — if detailed information about the actual data type is required — on objects of concrete type. Even a brief description of the EDM classes that are used in NEWT would go far beyond the scope of this document, but is essential for the understanding of the implementation of the New Tracking chain. The reader is therefore encouraged to find all detailed information about the EDM in [2]. The main tracking EDM classes, such as the track and measurement representations will in the following be used without further description of their class structure.

[7] The definition of tasks and operations through interfaces allows to evaluate, modify and eventually exchange the single modules of the track reconstruction during the long term operation of the ATLAS detector.

Together with the new EDM a common reconstruction geometry has been developed coherently to guarantee a consistent geometrical description for the use in track reconstruction. A core `Surface` class description has been introduced that builds the interface of geometrical information (given by the central ATLAS detector description) with tracking relevant data, such as hits on detecting surfaces. In other words, the `Surface` builds the binding link between the geometry and the linear algebra applications of the EDM. The purely mathematical description given by the `Surface` classes is extended to `Layer` and `TrackingVolume` object that give access to the material and magnetic field information. The full reconstruction geometry will be in the following referred to as `TrackingGeometry` and is described in detail in [8].

Common Tools Many repeating tasks and operations during track reconstruction can be performed without the need of any dedicated information about the underlying detector technology. This is in particular given for purely mathematical operations or higher level tasks that operate on base class level of the underlying tracking EDM. Track and vertex fitting are, in general, independent from the way the track information has been gathered as long as the EDM provides enough — and mathematical consistent — information about the track to perform the fit. In the ATLAS New Tracking realm, these tasks have been identified and concentrated in common `AlgTool` implementations, ordered in an intuitive package structure in the Tracking repository. The main tools and concepts are described in the following sub-sections. The building of the reconstruction geometry, which is also realised in the common interface structure of the New Tracking reconstruction is omitted in this consideration, since it is done only once at startup of the reconstruction job.

3.3.1 The Extrapolation Engine

The transport of track parameters (i.e. the representation of a track with respect to given detector surfaces) is a very frequent process in track reconstruction. It can be performed with different complexity, following diverse track models and propagation techniques. In many tasks, special consideration has to be paid on the correct integration of the effects originating from interactions of the particle with the traversed detector material, while in others this is of minor importance. The ATLAS extrapolation package provides a very flexible set of `AlgTool` implementations, including propagators for the purely mathematical transport of the track parameters, classes for material effects integration and magnetic field access. The extrapolation package is located in the TrkExtrapolation CVS repository and is based on the newly developed reconstruction geometry package. Further details about the extrapolation package can be found in [9].

3.3.2 Magnetic Field Access

The access to the magnetic field information in ATHENA is done by a dedicated `Service`, the `MagFieldAthenaSvc`. In the pattern finding stage of track reconstruction it is usually sufficient to use a less granular parameterisation of the magnetic field, while in the final track fit, the best description of the magnetic field is used to achieve the optimal track resolution. In NEWT, a dedicated `AlgTool` interface layer between the standard ATHENA magnetic field access service and the client code has been inserted to allow specific simplifications, modifications of the given magnetic field map or the direct access to the ATHENA `MagFieldSvc` through one single interface. For the extrapolation package the access to the magnetic field tool is given through a special `MagneticFieldProperties` class. The `MagneticFieldProperties` class is a base class of the `TrackingVolume` class, which allows the definition of different field configurations for different parts of the detector. This is of special interest for many pattern recognition programs where simplified parameterisations provide, in general, a satisfactory accuracy for the pattern search. The additional flexibility of modifications and distortions will be in particular important for the scaling and adjustment of the magnetic field during the start-up phase of the ATLAS experiment and did proof well during data taking of the ATLAS combined test beam in 2004.

3.3.3 The `ITrackFitter` Interface and common Track Fitters

A track fit is the estimation of the best set of parameters describing the track with respect to a given reference surface when a collection of hits is given to define the track trajectory. This fitting procedure can be performed in various different ways, while the input data to the fit is properly defined: fitting can be done on a set of non-calibrated hits (i.e., in the ATLAS EDM, a collection of `PrepRawData`) objects, on a set of calibrated measurements of various types (in ATLAS realised as extensions of a `MeasurementBase` base class) or as a refit of a given `Track` or `TrackSegment` object. Only few additional parameters, such as a particle type hypothesis for the material effects integration or a possible outlier logic steering are needed to specify the fitting procedure. In NEWT, a general `ITrackFitter` interface has been imposed that defines the main functionality of any used track fitter; currently six different fitting techniques are implemented under this interface and can be chosen at job configuration level:

- `KalmanFitter` (KF): the `KalmanFitter` is a straight-forward implementation of the Kalman filter technique that has been adopted for the track fitting in high energy experiments [10]. It combines forward filtering, backward smoothing and an outlier rejection; it uses the extrapolation engine with its underlying reconstruction geometry for filter step predictions. For the ATLAS silicon detector, the KF has a dedicated extension for he fitting of electron tracks, that lose stochastically a significant part of their energy due to bremsstrahlung effects. In this case, the purely Gaussian process noise assumption — for energy loss based on ionisation loss — that is intrinsic to the best estimator mechanism of the Kalman approach is far from being optimal. A special dynamic noise adjustment schema (DNA) [11] has been developed that still uses a Gaussian error assumption, but adapts the value of the applied variance with respect to the amount of traversed material.

  The gain matrix driven update of a track prediction with a given measurement is the main concept of the Kalman filter technique. In NEWT, this specific operation is defined by an `IUpdator` interface and accessed by the `KalmanFilter`. The different `IUpdator` realisations that exist in the ATLAS New Tracking realm are described in Sec. 3.3.4.

- `DeterministicAnnealingFilter` (DAF): the deterministic annealing technique [12] combines the standard Kalman filter formalism with a probabilistic description of the measurement assignment to a track; per detecting surfaces several measurements can be fitted at once, each weighted by the current assignment probability given through the track fit. An annealing schema using a defined Boltzman function is performed by *cooling* the system *temperature* to a threshold level while iterating the track fit. This schema allows the DAF to perform local pattern recognition in combination with a track fit in detector regions with high hit multiplicities. The DAF is implemented using the `KalmanFitter` underneath and imposing the additional annealing schema. The DAF extends the tracking EDM with a probabilistic hit description, i.e. multiple hits are grouped together in one single measurement class. Details about the actual implementation of the DAF concept in the ATLAS New Tracking realm can be found in [13].

- `GaussianSumFilter` (GSF): the GSF is a special multi-Gaussian extension of the standard Kalman fitter [14], aimed at the reconstruction of electron tracks. In the GSF approach, the highly non-gaussian probability density function of the electron energy loss is modeled by a mixture of several Gaussians. To integrate this model into the track parameterisation, the track parameters for the GSF are realised as a multi-component bundle and processed simultaneously. Evidently, the GSF needs an extended EDM for handling the multi-component approach. Component reduction is imposed at several steps to avoid an exponential growth of the number of components describing the track during the full track fit.

- `AlignmentKalmanFitter` (AKF): the alignment of the ATLAS detector will be a challenging task that is of particular interest at the startup phase of the experiment. Track based alignment procedures play hereby an important role and will be carried out trough the entire lifetime of the experiment. Recently, an extended version of the Kalman filter has been developed [15] that integrates the update of the detector surface orientation and position into the intrinsic measurement update of a Kalman filter step. This technique is realised in ATLAS through a special `AlignmentKalmanFitter` that is based upon the standard `KalmanFitter` implementation

and an extended version of the ATLAS reconstruction geometry that introduces custom alignable `Surface` objects[8].

- `GlobalChi2Fitter`: the track fit through the minimisation of a global χ^2 value is a very common and robust fitting technique that is described in various publications (such as [16]). Given purely Gaussian process noise, the minimisation of the χ^2 value that is built from the hit residuals at every measurement surface marks the best set of estimators of the track trajectory. Material effects enter the χ^2 function as additional fitting parameters, weighted by their expected variance due to their stochastic behavior. The minimisation of the global χ^2 value is then, in general, performed by solving a set of linear equations through a matrix inversion. Thus, the number of global parameters have to be kept low to minimise the CPU time consumption. In NEWT, this technique is implemented through the `GlobalChi2Fitter`, which has been of extensive use while reconstructing data from the combined test beam 2004 and during the reconstruction of tracks originating from cosmic rays, see Sec. 4.4. The `GlobalChi2Fitter` is interfaced with the common reconstruction geometry via a dedicated dynamic layer schema.

- `DistributedKalmanFilter` (DKF): this is a modified version of the the Kalman filter formalism that estimates the track parameters only for the perigee representation. This leads to a significant speed-up of the algorithm, since the for the original Kalman approach necessary propagations of the state vector to the measurement surfaces are reduced to a pure integration of material effects. The DKF also deploys a χ^2 based outlier rejection and an internal node schema for representing barrel or endcap measurements. It has been designed mainly for the use in the ID LVL2 Trigger and Event Filter and is a substantial part of the ID LVL2 application IDSCAN [17].

3.3.4 The `IUpdator` Interface

The update of the predicted track parameters with a measurement is a commonality of most progressive track fitting algorithms. Different models based on a covariance or weight matrix formalism can hereby be used. The mathematical background for the measurement update is well defined, however, the realisations can differ through approximations. In NEWT a dedicated `IUpdator` interface is provided and four different `AlgTool` implementations exist: three of which incorporate a covariance matrix based formalism and differ mainly through the used underlying math library and numerical stability. The fourth `IUpdator` implementation deploys a weight matrix approach and is superior in timing performance when being used with the DAF since it saves unnecessary matrix inversion when weighting the individual measurements.

The `IUpdator` can also be used to perform a reverse measurement update, as long as the full information on the measurement surface (including the combined covariance matrix) is given. This allows to calculate unbiased hit residuals without performing an additional track fit, which is a useful feature for the validation and alignment algorithms.

3.3.5 Calibration on Track

One concept of the ATLAS New Tracking is the (re-)calibration of the measurement based on the track direction and sensor intersection point. This can be used e.g. for the definition of the drift radius sign, error scaling, future calibration or conditions data depending on a given module intersection such as dead or noisy channel information. The imposed calibration model includes the adaption of cluster errors as well as the integration of chamber or module distortions to account for a realistic geometry description. Every sub-detector technology in ATLAS will have different strategies for calibration, optimised for performance of the individual part. To integrate the *on track* calibration into the general track fit, it is necessary to provide a schema that is at highest level independent from the used concepts and encapsulated from the sub-detector realms. This is established through a mixture of a common interface and various different implementations in the sub-detector repositories.

[8] The `Surface` objects of the ATLAS reconstruction geometry are fully integrated into the conditions data schema of ATLAS and retrieve alignment data at geometry construction or triggered through a callback in case that the geometry setup has changed during a single reconstruction job. The `AlignableSurface` that extends the common `Surface` base class introduces, however, a user-open alignment update possibility which is needed for this iterative alignment approach.

The IRIO_OnTrackCreator defines this transition from non-calibrated (PrepRawData) to calibrated (RIO_OnTrack) EDM objects and finds a common implementation in the Tracking realm: the so-called RIO_OnTrackCreator, holds a collection of pointers to further IRIO_OnTrackCreatorTool objects, each of which representing a different sub-detector.

3.3.6 Summary, Scoring and Helper Tools

The ATLAS tracking EDM provides various models of classifications for tracks: the most widely used quality information of fitted tracks is the χ^2 value together with the number of degrees of freedom n_{dof} of the track fit. This information is directly accessible through the Track object, but does not contain a lot of information about the track morphology. The χ^2/n_{dof} is a good parameter for a fast separation of good and bad tracks or for the application of quality cuts on a given track sample. However, it does not help in classifying the tracks any further which is necessary at various stages in the track reconstruction (i.e. the identification of fake tracks, the solving of hit ambiguities or evaluation of track extensions). Detailed information about the track characteristics in the sub-detector has to be accessible, which in the ATLAS EDM is realised through a TrackSummary object. The New Tracking uses a dedicated track scoring approach: first the TrackSummary is created by the TrackSummaryTool, which parses the Track object and records the hit statistics and characteristics for the various sub-detectors into a pre-defined enumeration schema. Helper tools in both, ID and MS, guarantee hereby the access to the relevant information that is specific to the detector technology. The TrackSummary is not assigned to a Track object as a private member, since many of the hit characteristics, such as shared hits, are a property of the processed track collection and not of a single track. Dedicated scoring functions that give bonus or penalty points for different patterns calculate a final track score that is then used for the track classification. Since the ITrackScoringTool allows different implementations of scoring functions that may also incorporate a different scoring system, the track score is not stored on track to prevent comparisons of track scores from different sources and with different meanings.

3.3.7 Truth Association and Validation

The validation of the entire track reconstruction chain is a necessary but complex task, since many different processes contribute to the final track reconstruction result. In the ATLAS New Tracking realm, dedicated emphasis has been put on having automated validation procedures for different stages of the track reconstruction process. The common EDM allows to concentrate the validation algorithms into a separate structure, while supporting the input of several different track reconstruction sequences as long as they comply with the tracking EDM.

Truth Association For many validation studies using Monte Carlo simulated data the association of reconstructed input data with the Monte Carlo truth is essential. The ATLAS tracking EDM follows a strict association pattern for the truth binding, i.e. no direct link between the event data object and the corresponding Monte Carlo truth object exists, but the relation between the two is purely done by an associative container. Several sets of Algorithm and AlgTool classes exist that parse given EDM input containers and establish the association to the truth objects. The truth association is mainly done on three different levels:

- **Hit Truth Association:** the clusters and drift circles are in ATLAS represented by PrepRawData objects. The first stage in the truth association of the track reconstruction is thus to find the relation of the PrepRawData objects to the simulated hits and through back navigation in the Monte Carlo record to the generated particle. Given the track density in the ATLAS detector it is possible that during the clusterisation process one PrepRawData object is constructed from several simulated hits that are caused by different generated particles. To account for this ambiguity, the PrepRawData truth collection is implemented as an associative container that allows this multiple relationship, even with given assignment probabilities (in C++ terms this is done using an STL multimap object).
- **Track Truth Association:** the task of associating the reconstructed track objects with the generated particles is defined defined by the IDetailedTrackTruthBuilder interface. A track can hereby correspond to one or many generated particles; since in full detector simulation many

hard interaction processes change the identifier of the particle, while for tracking the given chain of truth particles is in the optimal case still reconstructed as a whole, a one-to-many relationship has been established as default. The classification whether a track corresponds to a given truth trajectory is done using the `PrepRawData` truth collection created in the hit truth association process.

- `TrackParticle` **truth association:** the `TrackParticle` class is not in particular a component of NEWT, but builds the interface of track reconstruction to many analysis applications. Additionally it marks the representation of the track in the AOD containers[9]. Since in the ATLAS data model the `Track` is not contained anymore in the AOD, it is convenient to establish a dedicated truth association container for the `TrackParticle` objects, by simply forwarding the track truth information.

The Monte Carlo truth association is highly detector specific, therefore most of the `Algorithm` implementations are located in the sub-detector repositories. However, the common steering and the interfaces to be used by the different technologies are concentrated in the ATLAS Tracking realm.

Validation The validation of NEWT concentrates on two different topics: performance and reliability. Performance validation is in general done at various stages and is in many cases deeply woven into the sub-detector concepts. However, on a general track level it can be to some extent performed within the common tracking framework. A dedicated package, the TrkValidation concentrates therefore several `Algorithm` and `AlgTool` objects capable of filling histograms from tracking EDM objects from base class level, Tab. 2 presents an overview and a brief description of the components that can be found in the TrkValidaiton container.

Table 2: Container packages and brief description of their content in the TrkValidation package, packages dedicated for the validation of the vertexing performance are omitted since they are not particular to NEWT.

Container	Description
TrkValAlgs	steering algorithm for track based validation, including track parameter residuals and pulls, hit residuals and pulls (biased and unbiased), an `Algorithm` for track difference calculations, material validation
TrkValInteraces	interface definitions for `AlgTool` classes used within this context
TrkValTools	several `AlgTool` classes that are mainly used by modules from the TrkValAlgs package: a general residual and pull calculator, a hit position helper, etc.

Many of the performance actions rely on the Monte Carlo truth information and thus require the truth association to be done before. Hit residuals and pull distributions for hits on surfaces can however be calculated without truth association, since the *true* hit position is given by the measurement. The TrkValidation follows hereby the described structure of a common interface and dedicated implementations of `AlgTool` classes in the sub-detector realms to guarantee access to the underlying specific hit information. If the complete track information is provided, i.e. both the measurement and the track representation are given with covariance matrices, the unbiased residual can be calculated by applying an reverse update step to the track. Additional validation and statistics packages can be found in the sub-detector repositories such as e.g. the InDetRecStatistics package that focusses on a detailed validation of the pattern recognition efficiency and track reconstruction resolution in comparison with Monto Carlo truth information for the Inner Detector.

The reliability of the software is of similar importance as the tracking performance. In the ATLAS computing model, automatic test runs on standard input samples are performed for every new build of the software project to monitor sudden changes on the performance level. Additionally, the ATHENA framework provides several `Service` implementations to monitor memory consumption and leaks[10],

[9] In the ATLAS computing model, the event data exists on the persistent side in two levels, the Event Summary Data (ESD) and the highly compressed Analysis Object Data (AOD) that should be suitable for almost any analyses.

[10] The liability of memory leaks is one of the drawbacks of the data factory software design, since the latter requires a precise tracking of the object ownership.

timing performance and status code return values. The ATLAS New Tracking adds in addition an instance counting schema for the EDM object, performed by the `EventDataMonitor` (also located in the TrkValidation package) that can only be executed in a special debug mode.

4 The ATLAS New Tracking in the Inner Detector

In modern track reconstruction strategies there is no clear border between the classical modules *pattern finding* and *track fitting*. This is — on the one hand — due to the fact that many pattern finding strategies (contrary to a classical histogram based approach) nowadays incorporate a two stage pattern: a global pattern search, as well as a local pattern recognition where track fitting is already part of. On the other hand, many track fitters such as the combinatorial Kalman filter or the deterministic annealing filter incorporate an intrinsic pattern recognition during the fitting process. Thus, the full chain of pattern recognition and track fitting will be in the following described as a single unit.

The ID New Tracking currently covers two sequences, the main *inside-out* track reconstruction and a consecutive *outside-in* tracking. The primary pattern search concepts for both sequences have been to a large extent adopted from the already existing ATLAS ID reconstruction program `xKalman` [18], but integrated and accomplished by additional components in the common NEWT approach. A third sequence, the *second stage pattern* recognition for the finding of V0 vertices, kink objects due to bremsstrahlung and their associated tracks has been also deployed using the common tracking tools and EDM, but is not particular to the New Tracking approach. Section 4.1 and Sec. 4.2 will in the following describe the main concepts and tools used for establishing a full Inner Detector reconstruction chain under the New Tracking, while Sec. 4.3 highlights the special adoption of the ATLAS ID New Tracking for the third level trigger stage, the ATLAS *Event Filter* (EF). To distinguish the EF realisation from the standard ID reconstruction, latter will be in the following also referred to as *offline* reconstruction.

4.1 Inside-out Track Reconstruction

The primary ID pattern recognition follows an inside-out strategy for track finding. It is realised as a sequence of modules — each represented through a dedicated `Algorithm`— and described in more detail in the following sub-sections. Figure 2 shows an extended UML sequence diagram for the inside-out tracking. In a classical picture, many modules of the sequence can be divided into global pattern recognition and consecutive local pattern recognition that only works on the reduced output sample of the global search results.

4.1.1 `SpacePoint` Formation

The first step in the inside-out track reconstruction is the creation of three-dimensional representations of the silicon detector measurements, which are then called `SpacePoint` objects. For measurements with the pixel detector this is a very simple task, since the pixel modules provide a two-dimensional local measurement that is — using the constraint of the `Surface` representing the detector element – transformed into a `SpacePoint` by a simple *local-to-global* transformation. Single SCT clusters can not be transformed directly into a three-dimensional representation, since the precise measurement on an SCT module can only be given orthogonally to the silicon strip direction. The transformation to a three-dimensional point is therefore not constraint and a direct representation as a three-dimensional point can not be defined. However, the SCT detector is built with a sandwich module structure, i.e. two silicon modules are glued together back to back, but rotated by a stereo angle with respect to another, this relation — and together with a beam spot constraint — this can be used to construct the three-dimensional `SpacePoint`. The position of the beam spot is hereby automatically retrieved from the conditions data or can be set by hand (e.g. for cosmic ray reconstruction). In contrast to the pixel detector, where each cluster directly leads to a `SpacePoint` object, the SCT `SpacePoint` formation features an intrinsic noise suppression at the very first pattern stage, since it requires two different modules with separate readout for the creation of one single `SpacePoint` object.

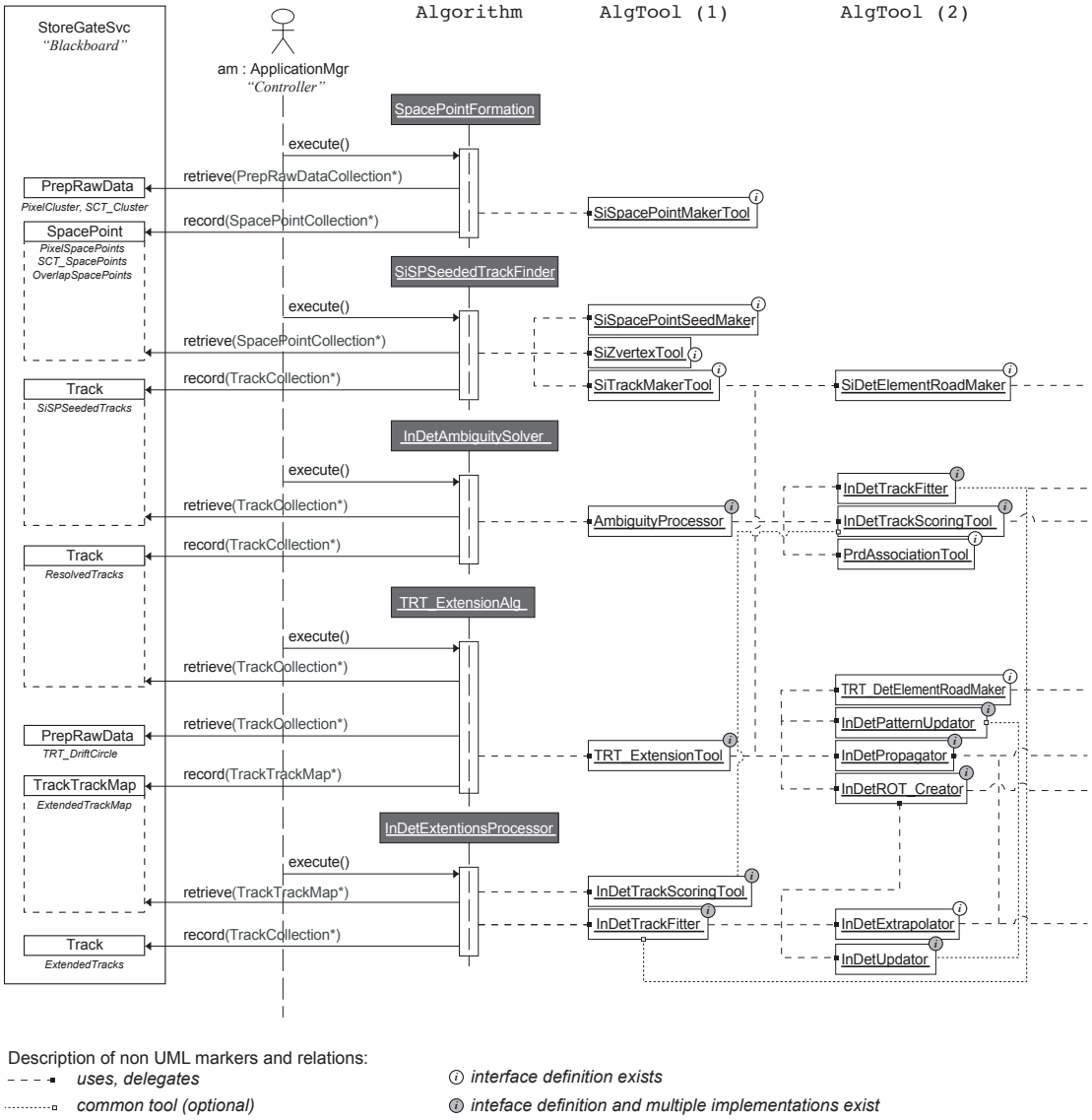


Figure 2: The main sequence of modules and some of the `AlgTool` classes used in the ATLAS New Tracking for the Inner Detector illustrated as an extended UML sequence diagram. Following a blackboard architecture structure, the transient event store (StoreGateSvc) acts as the *blackboard* for reading and writing of event data, the `ApplicationMgr` as the *controller* of the system. Only two levels of embedded `AlgTool` classes are shown, where dashed lines indicate a *uses* relationship, while the dotted line indicates possible sharing of `AlgTool` instances.

4.1.2 `SpacePoint` seeded Track Finding

The `SpacePoint` collections filled in the previous step are further processed for seeding the track candidate search in the Inner Detector. The `SiSPSeededTrackFinder Algorithm` represents this second module in the inside-out sequence. It can be divided into two different tasks, the track seed finding and the track candidate creation, based on the seeds found in the first step.

The seed search marks the global part of the pattern recognition and can be done in the following ways, using the `SiSpacePointSeedMaker`:

- **Seed search with z vertex constraint:** `SpacePoint` pairs from only the pixel detector are found in a first step and z vertices are built from these pairs using a dedicated `SiZVertexMaker`

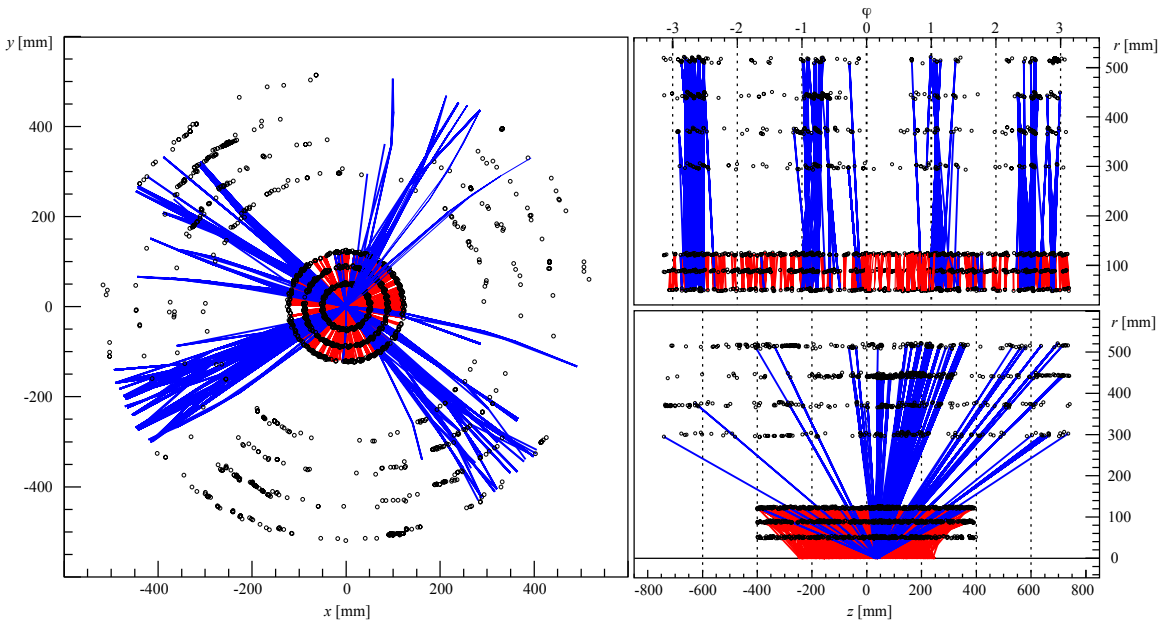


Figure 3: `SpacePoint` seeds consisting of two (short seeds) respectively three (long seeds) objects in the ATLAS Inner Detector barrel for a $t\bar{t}$ event, found with the z-vertex constraint seed search: the seeds consisting of two `SpacePoint` objects are used to determine z-coordinates of the predicted vertex positions. Only vertices within a defined range around the interaction point are used to constrain further seeds with three or more `SpacePoint` objects. For convenience, only seeds that are entirely in the barrel region are drawn.

`AlgTool`. The vertices are filled in histograms, keeping the seeds compatible with a given momentum and transverse impact range. A fast primary vertex search is performed and the primary vertex is used to further constrain the seeds with three or more space points. The tolerance region for predicted vertices from constructed seeds can hereby be chosen as a cut parameter. Figure 3 shows the seeds for vertex finding and track candidate search for an example $t\bar{t}$ event in the pixel and SCT barrel.

- **Unconstrained seed search:** The seed search can also be performed without the given z vertex constraint, which leads to a significantly higher number of initial track seeds (and in the following track candidates). The unconstrained seed search is evidently more time consuming, but more efficient to find tracks in events with loosely constraint primary vertices, such as $H \rightarrow \gamma\gamma$ decays or non-physical single track events with superimposed pile-up signatures. Figure 4 shows the z vertex distribution for an example $t\bar{t}$ event and Fig. 5 shows the resulting `SpacePoint` seeds found without z vertex constraint.

The z vertex scan is the standard `SpacePoint` seeded track search strategy in the ATLAS release 12.0.6, while for further production releases the unconstrained seeding is foreseen to be default in the ID NEWT track reconstruction.

Once the `SpacePoint` seeds are found, the road building process is started: the seeds provide already enough directional information to build roads of detector elements for the further search of associated hits to one track candidate. This marks the beginning of the local part of the silicon pattern recognition. At this stage, the `SpacePoint` objects are dissolved into the cluster objects of which they have been originally build from. This is, because the track candidate creation involves track fitting, which is in general performed on either `PrepRawData` or `RIO_OnTrack` level[11]. The cluster collections that contain also the clusters that have not been used to create `SpacePoint` objects are retrieved from the transient event store and those that are located on detector elements that build a road are used for the track candidate. A Kalman fitter-smoother formalism is used to simultaneously follow the trajectory

[11] The `SpacePoint` class, however, has been recently integrated into the `MeasurementBase` schema and could also be used for track fitting on this level. Since the creation of the `SpacePoint` objects include a projective error treatment, the fit on `RIO_OnTrack` level is more precise.

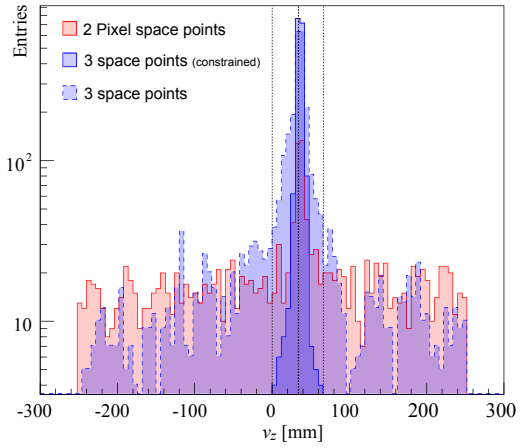


Figure 4: The distribution of the z vertex coordinate v_z of the projected vertices that originate from the different `SpacePointSeed` objects in the same event as shown in Fig. 3. The narrow distribution shows the v_z distribution for the seeds made of two pixel `SpacePoint` instances, where the other two show the same distributions for the seeds build from three `SpacePoint` objects for both, the z vertex constraint (solid) and the non-constraint (dashed) configuration. Additionally the region around the primary vertex found through the fast z vertex scan on the pixel seeds is shown.

and include successive hits in the track candidate fit. This approach is intrinsic to the Kalman filter formalism: it progressively updates the track information (including the covariances) and thus predicts precisely the track representation on the next measurement surface. Since, in general, a silicon detector element has more than one hit per event, the prediction leads to the most likely extension of the trajectory, while detecting outliers immediately via their large contribution to the χ^2 of the track.

`SpacePointSeed` objects do not necessarily lead to a track candidate, in contrary, only in about 10 percent of the cases the seed is successfully extended to a track candidate, stored in the common `Track` EDM format. The track finding from seeds, realised through the `SiSPSeededTrackMaker AlgTool` provides also the possibility to find more than one track candidate from a given seed, but this is a very rare case in the ATLAS ID event reconstruction.

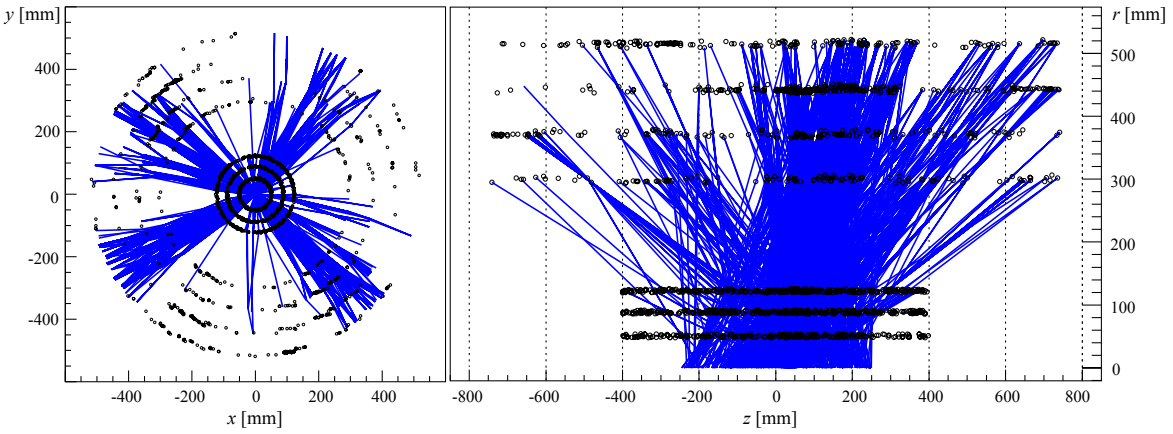


Figure 5: The same event as shown in Fig. 3 using the second silicon seed strategy without vertex constraint. The `SpacePoint` seed build of three objects are spread over a large z-range that leads to an increase of the track candidates for further processing.

4.1.3 Ambiguity Solving

The seeded track finding results in a very high number of track candidates, that have to be resolved before the extension into the outer TRT can be done. Many of these track candidates share hits, are incomplete or describe fake tracks, i.e. tracks where the majority of associated measurements do not originate from one single particle. The tracks have to be therefore ranked in their likelihood to describe the real trajectories of particles from the underlying physics event. A first step here is to refit

the track using the refined reconstruction geometry that has a detailed material description. However, the track fit only results in a global parameter, the χ^2 divided by the degrees of freedom, which is in most cases not appropriate to decide whether a track was a good or fake track. For the classification of tracks, a so-called *track scoring* strategy has been developed [19], that describes in addition to the fit quality morphologic parameters of the track: different characteristics of a track are hereby represented by a beneficial or penalty track score, which all together form an overall track score. In general, each hit associated with the track leads to a better score value to favor fully reconstructed tracks rather than small track segments. The measurements of different sub-detectors are, in general, weighted with different scores, preferring the precision measurements (e.g. pixel clusters) and downgrading measurements from less precise detector parts. Table 3 gives a qualitative overview of the different benefits and penalties of tracks found in the `SpacePoint` seeded track search, and Fig. 6 illustrates some of the track characteristics to be resolved in the SCT barrel detector.

Table 3: Track characteristics that lead to benefits or penalties in the ATLAS silicon detector track score.

Track characteristics	**Detector**	**Effect on the track score**
B layer hole	pixel	strong penalty
Layer hole	pixel	penalty
Overlap hit	pixel, SCT	strong benefit
Sensor hole	SCT	weak penalty
Layer hole (module)	SCT	strong penalty

Hits that are shared between tracks are — after the track scoring has happened — mainly assigned to the track with higher score, while the remaining track is being refitted without the formerly shared hit[12]. The refitted track is again scored and enters the remaining list of tracks to be evaluated. In an iterative procedure, the tracks with highest score are bundled and tracks that fall beyond a certain quality cut are neglected for further processing.

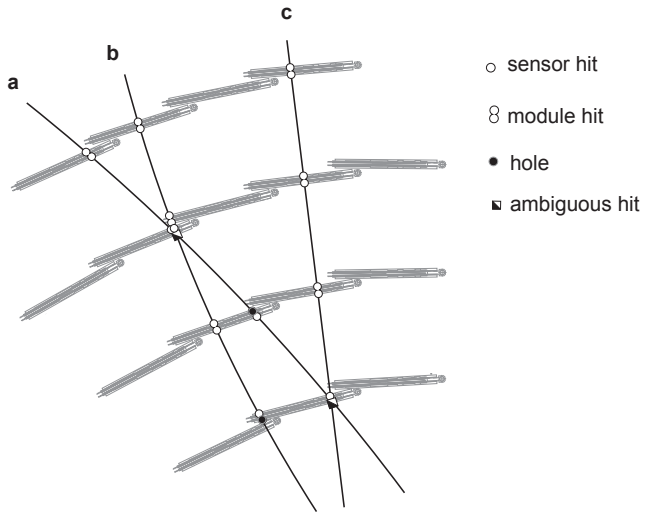


Figure 6: Simplified model of the ambiguity solving process, illustrated in the SCT Barrel. Tracks **a**, **b**, and **c** have been found through the seeded track finding, but share several hits. The χ^2/n_{dof} may not be appropriate to distinguish a true from a fake track, therefore dedicated track scoring that is optimised for each sub-detector is used. In the shown example, e.g. a *module hit* representing measurements on both sides of the SCT silicon detector are scored relatively higher than two single hits without associated backside module. Hits in a overlap region as for track **b** are in particular high scored, while holes on track, i.e. an expected hit that has not been found, lead to a penalty in the track score.

4.1.4 TRT Track Extension

The track (segment) extension from the silicon detector into the TRT consists of two modules, the `TRT_ExtensionAlg` and the `InDetExtensionProcessor`. The `TRT_ExtensionAlg Algorithm` steers the extension finding on a single track by track basis; the tracks found through the silicon seeds and

[12] In a recently introduced strategy, that has not yet been part of the ATLAS 12.0.6 release, hit sharing between tracks is allowed when the track fulfills dedicated quality criteria. This is to account for the fact that in the pixel system the readout creates an artificial ambiguity between hits that are joined together to one readout element (ganged pixels).

resolved by the ambiguity processing are used as an input to find compatible sets of TRT measurements that are further processed as candidate extension. Again, the `TRT_ExtensionAlg` simply delegates this task to a dedicated `AlgTool`, represented through the `ITRT_TrackExtensionTool` interface. The silicon-only track must hereby not be modified, the association of the TRT hits are therefore purely extensions and is not done by a combination. Figure 7 shows the extensions of the silicon seeded tracks into the TRT detector for a sample $t\bar{t}$ event.

Two concrete implementations of the track extension tool exist:

- `TRT_ExtensionTool_xk`: the `TRT_ExtensionTool_xk` is the standard implementation used in the Inner Detector New Tracking. It follows a classical approach starting with road finding through track extrapolation, and — using the hit coordinates expressed in $r-\phi$ in the barrel, and $r-z$ in the endcap region, respectively — performs a line fit to estimate whether the hit is compatible with the silicon track seed or not.

- `TRT_ExtensionTool_DAF`: this extension tool is designed using the deterministic annealing filter (see Sec. 3.3.3) that is optimised for very high hit densities. The DAF strategy also starts with the road building (and uses hereby the same `AlgTool` as the first extension strategy), but follows a different philosophy for the hit finding and hit assignment; TRT measurements within the road are grouped together on the same readout element (or on planes perpendicular to the extrapolated track, depending of the initial configuration) and represented as one input object to the track fit. The different hits within the group are weighted by their likeliness to represent the true hit, while the weights correspond mainly to the distance of the hit from the trajectory prediction (i.e. the residual).

The `TRT_ExtensionAlg` produces a map of the seeded silicon tracks and the found extensions, if no extension is found through the concrete version of the `ITRT_TrackExtensionTool` the second map entry is simply left empty. This map is written to the transient event store and successively picked up by the second `Algorithm` in the TRT extension module, the `InDetExtensionProcessor`. This `Algorithm` is responsible for evaluating the extended track with respect to the pure silicon track. The comparisons of the two tracks is based on a combined track refit and then done using the track scoring mechanism, comparing the track score of the original track with the one after refitting. Unlike the track extension `Algorithm` which is not allowed to change the initial silicon track, the refit in the `InDetExtensionProcessor` can modify the silicon hits by flagging them as outlier measurements. In case that the track score of the silicon track is higher than the extended version, the silicon track is kept and the TRT hits are put as outlier measurements onto this track. The `InDetExtensionProcessor` can be configured to work optionally with the DAF specific extension algorithm. In this configuration the DAF is specified as the track fitter used for the evaluation of the track extension, applying the annealing schema on the multiple measurements associated to the track in the `TRT_ExtensionTool_DAF`.

4.2 Outside-in Track Reconstruction

The inside-out sequence of the ID New Tracking relies on a track seed found in the silicon detector. In the track reconstruction process, some of these initial track seeds may not be found or do even not exist: ambiguous hits can shadow the track seed in the silicon and prevent the score of the silicon seeded track to survive the ambiguity processor on the one hand, and on the other hand, tracks coming from secondary decay vertices further inside the Inner Detector volume (e.g. K_s decays) or from photon conversions may not have any or only insufficient silicon hits to comply with the inside-out sequence. In a third pattern, substantial energy loss — mostly of electrons — at outer radii of the silicon `SpacePoint` seeded track and not known to the road building may guide the `TRT_TrackExtensionTool` into a *wrong* direction, such that no corresponding TRT hits are found. The ID New Tracking realisation will establish therefore a second sequence in track reconstruction, following an outside-in approach. The sequence will be realised as two different modules, starting with a dedicated segment finding algorithm and a successive back tracking of the segments into the silicon detector. In release 12.0.6 only the first part has been included and is discussed below.

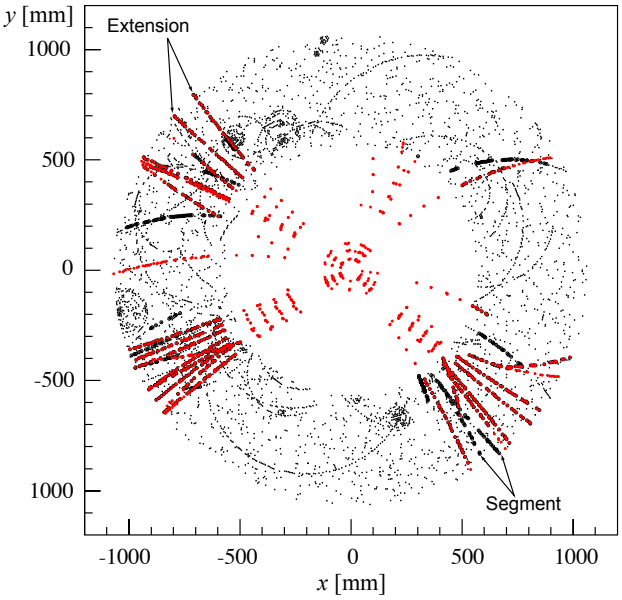


Figure 7: The same $t\bar{t}$ event as shown in Fig. 3 for the two possible TRT hit associations (only the barrel measurements): the brighter colored hits show the extensions that originate from following the `SpacePoint` seeded tracks into the TRT, the silicon `SpacePoint` objects are also shown in the same color. The black circles mark hits that have been associated to TRT segments, which builds the start point of the back tracking application. The particular power of the back tracking approach is to find the additional track segments, that have not been found through the inside-out sequence, simply for the fact that no appropriate silicon seed did exist for the further extension process. This is mainly due to strong energy loss of the particle, or due to the fact that the track segments originate from photon conversions or other decay vertices inside the Inner Detector volume.

4.2.1 TRT Segment Finding

The currently existing TRT segment finder implementation in the ID New Tracking realm is based on the outside-in track reconstruction strategy taken from the legacy `xKalman` program. It follows a two step procedure, starting with a global pattern search and a subsequent local pattern recognition with intrinsic track segment building. Since the TRT drift tube measurements do not provide any information about the coordinate along the straw direction, `SpacePoint` objects can not be built and the global pattern recognition has to be done in projective planes. Evidently, the most adequate projection planes for the TRT geometry have been chosen: the $r-\phi$ plane in the TRT barrel region and the $r-z$ plane in the TRT endcap part, where the single straws fan out on disc structures. Assuming that the tracks originate roughly from the primary interaction region, track segments from tracks with transverse momentum greater than 500 MeV appear as almost straight lines in the $r-\phi$ and rigorous straight lines in the $z-\phi$ projection. There exist many techniques to find straight line patterns. A very common one in high energy physics event reconstruction, the Hough transform [20], is used to find the hit pattern: it is based on the simple fact that by transforming the projection plane $r-\phi$ (or $z-\phi$, respectively) into the parameter space of the straight line — in this specific case identified as the initial azimuthal angle ϕ_0 and the inverse momentum parameter c_T (respectively c_z) — the points associated with one line are transformed into one single cell, since they satisfy the same line parameterisation. The global track segment search thus can be reduced to the local maximum finding in a two-dimensional histogram. To reduce the number of overlaying track segments, this histograming process is done for several η slices of the TRT detector. The missing hit information along the drift tubes in the TRT, however, results in the fact that hits have to be in general considered in several different slices. This relation has to be tracked and resolved by a simple maximisation of the the straw hits per found track candidate. Figure 8 shows a two-dimensional histogram for an example η slice in the Hough space.

Local Pattern Recognition and Event Sample Cleaning The histogram method provides a set of track segment candidates that are further processed in a second step of the TRT segment finding. Whereas the global hough transform uses the straw center position for the finding of compatible sets of hits, the drift time information is also used in the local pattern recognition process: using a Kalman filter-smoothing formalism the track segments are build and the final collection of `TrackSegment` objects are written to the transient event store.

In many cases, the `TrackSegment` finding will pick up segments that have been already successfully associated to tracks found in the silicon detector by the extension `Algorithm`. To save CPU time the segment finding is planned to work on a cleaned out hit sample. For the event cleaning, an association

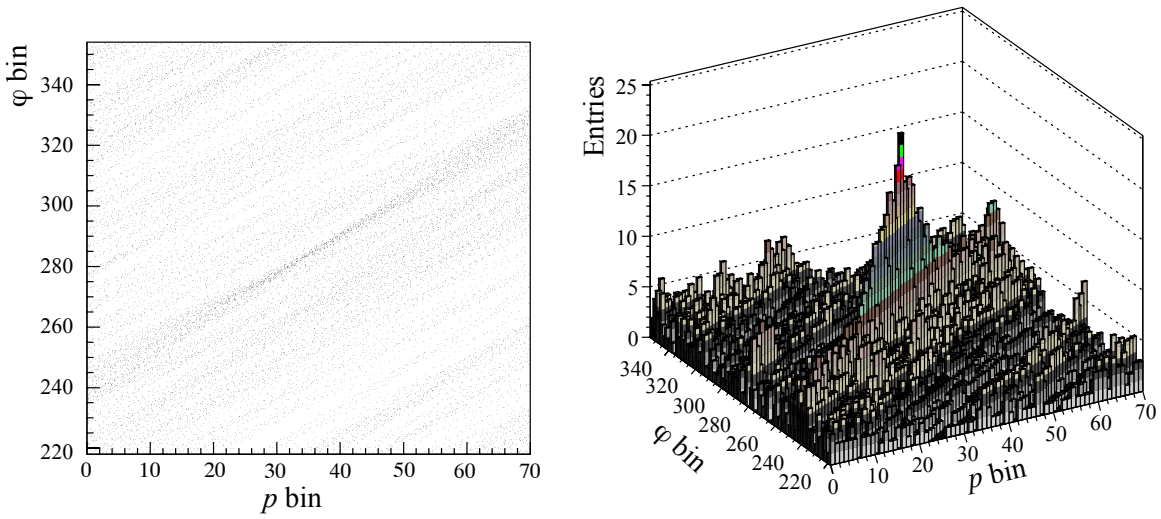


Figure 8: Straw representations in a part of the parameter space (ϕ bins and momentum bins) after applying the Hough transform. The Illustration shows a subset of the full parameter space that is divided in the standard configuration into 500 bins in the azimuthal direction, while trying 70 momentum hypotheses. The left side shows the Hough space in a scatter plot, while the right histogram shows the same subset in a three-dimensional version, where the most probably (ϕ_0, c_T) hypothesis can be clearly identified; the plot is restricted to a single η slice.

tool — the `PRD_AssociationTool` — that is also used in the ambiguity processing phase can be taken to find the hit-track relation.

The second step of the outside-in approach, the backtracking of the TRT `TrackSegment` objects into the silicon Detector is planned to be integrated with release 13.0.0 and will be presented in a separate document [21].

4.3 The Event Filter Realisation

One special aim that has been followed during the development of the ATLAS ID New Tracking software was to be able to provide the same track reconstruction chain to the ATLAS High Level Trigger realm. This is achieved through the modular NEWT design which allows to replace time-critical components and full-featured offline modules by trigger-specific implementations.

The ATLAS trigger system consists of a three step triggering system, with one pure hardware-based LVL1 trigger, followed by the software LVL2 and the Event Filter (EF) [22]. The trigger strategy starts from a very fast calorimeter and Muon Spectrometer based region of interest (ROI) estimation. The ROIs found in the LVL1 might be further refined in the LVL2 trigger, where the calorimeter clustering is repeated with higher granularity and first tracking stand-alone algorithms are run in the ID and the Muon System. The Event Filter is the last step in the ATLAS Trigger chain. It is a pure software trigger, working on the output of the LVL2 trigger objects. These objects already incorporate a hypothesis of the event morphology and steer the EF to run in one of several pre-defined slices, among which are the *electron*, *muon*, *b-jet*, *b-physics*, *tau* and *gamma* slices. Each of these slices contains a very similar `Algorithm` sequence as the ID inside-out tracking and are followed, depending on the given slice by dedicated event reconstruction algorithms including vertex finding, b-tagging or electron processing. In the EF realisation of NEWT dedicated `Algorithm` classes steer the underlying `AlgTool` objects with ROI seeded input collections. The used `AlgTool` classes are directly taken from the offline reconstruction chain but operated in a ROI seeded mode, where the trigger slice defines the width of the ROI. Given the time constraint of the EF, which is of about 1 second of total process time per event, the NEWT `AlgTool` classes are configured using simplifications such as the default z vertex scan in the `SpacePoint` seed finding or the use of the fast vertex fitter in the post processing time. For each trigger slice, a dedicated output collection is written to the transient event store, while overlapping can take place in both ROI regimes and multiple appearance in the different pre-defined

slices. Figure 9 shows an example simulated $t\bar{t}$ event reconstructed with the offline Inner Detector New Tracking, and the ROI seeded event filter reconstruction, illustrated with the ATLAS event display ATLANTIS [23].

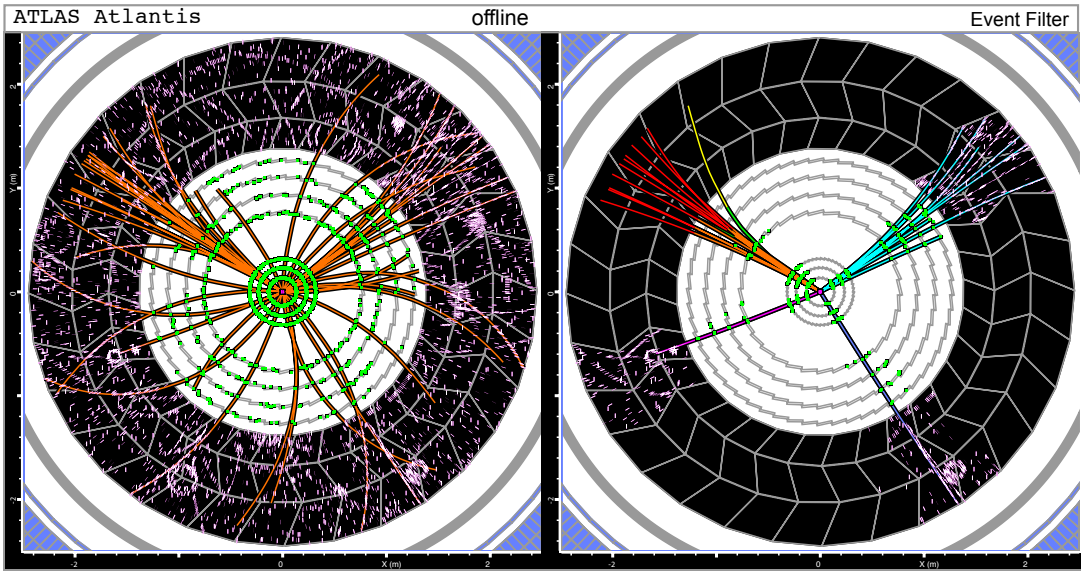


Figure 9: An simulated $t\bar{t}$ event reconstructed with the offline and Event Filter versions of the ATLAS New Tracking. The Event filter has been executed here in the electron slice which determines a dedicated region of interest size in which the full NEWT inside-out sequence is performed. The four regions of interest found by the prior LVL1 and LVL2 trigger and associated to an electron hypothesis can be clearly distinguished.

4.4 NEWT in the Combined Test Beam and Commissioning Setup

A complete independent New Tracking sequence, the CTBTracking [24], has been initially developed for the combined test beam run in 2004 (CTB2004). The CTB2004 was the first time that the ATLAS offline reconstruction software had to reconstruct real data of a complete slice of the ATLAS detector. The CTBTracking was the first complete reconstruction chain that was realised through the modular NEWT design and served as a prototype for the new software model. It incorporates a combinatorial track finding based on silicon space point samples that is valid for a small number of track candidates and is based on the `GlobalChi2Fitter`, see Sec. 3.3.3. Intermediate fits of the track candidates are used to reduce the number of track candidates and a χ^2 based ambiguity solving mechanism disentangles track candidates with ambiguous hit patterns. The CTBTracking has been extensively used for the track reconstruction and alignment studies of the CTB2004 and has been extended to be applicable for the ongoing ATLAS commissioning runs with cosmic ray tracks. Figure 10 shows a cosmic muon track reconstructed with the CTBTracking.

5 NEWT for Combined Reconstruction

Initially deployed in the ATLAS Inner Detector, common components and modules of the New Tracking have already spread widely into software applications for the Muon System and combined reconstruction. The common ATLAS tracking EDM has also become the input event data format of all track reconstruction algorithms of the Muon System and the ATLAS `TrackingGeometry` has been expanded to a full description of the ATLAS detector. Although no complete reconstruction chain based on the New Tracking concepts exists so far in the Muon System or combined reconstruction, the following section will cover some of the main tools used in combined reconstruction.

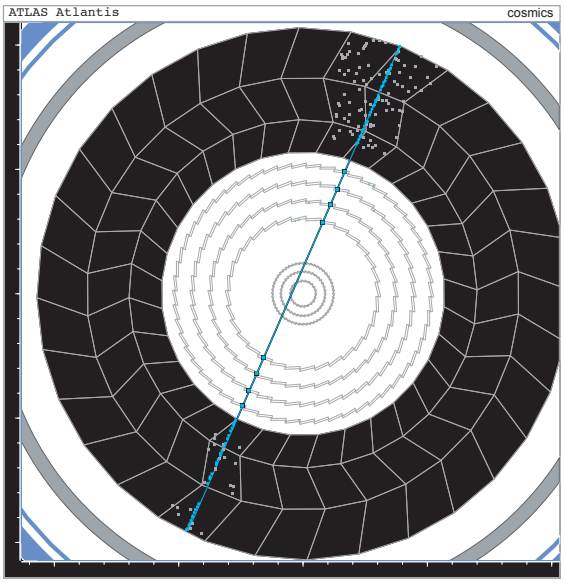


Figure 10: A real cosmic ray event found by the CTBTracking package and shown with the ATLAS event display ATLANTIS. The CTBTracking has been initially developed for the combined test beam run and was the first reconstruction sequence entirely build from New Tracking components using the common event data model. The CTBTracking has been also adopted to serve the needs for reconstructing cosmic ray tracks.

5.1 Calorimeter Impact and Vertex Expression

Based on the new ATLAS extrapolation engine, the `TrackToCalo` application was the first in a list of following combined muon reconstruction tools[13] that have been based on the tracking EDM and common tracking tools. The `TrackToCalo AlgTool` is realised as a wrapper of the `Extrapolator AlgTool` and prepares the input data for the track-to-cluster comparison: a specified track collection is retrieved from the transient event store, for each of the tracks that are contained by the given collection an extrapolation is performed to the different sampling layers of the calorimeter and the cells with energy deposit are collected in a given isolation cone. The different coordinate definitions between the Inner Detector and the calorimeter have to be hereby taken into account, which is performed in the Reconstruction repository and completely shielded from the common Tracking tools. `TrackToCalo` is realised as a central `Algorithm` that performs the comparison per default for every track. However, many applications will be only interested in a particular subset of the track collection. For this purpose, the single track operation is encapsulated in a dedicated `AlgTool`, the so-called `ExtrapolTrackToCaloTool` which can be used independently by any client.

A very similar `AlgTool`, the `TrackToVertex`, performs the extrapolation of a track to a given vertex position. It simply wraps the extrapolation engine preparing a user-friendly interface.

5.2 Combined Muon Refitting

The combined fit of a track that originates from a muon with the use of both independent tracking devices is an outstanding goal of the new ATLAS track reconstruction. Clearly, two main strategies can be identified to combine tracks from the Inner Detector with those found in the Muon System: a single combination of the two tracks that are represented with respect to the same reference surface (e.g. both track segments are in a perigee representation) based on a χ^2 minimisation, or the complete dissolving of the track into the hit objects and performing a combined track fit. While the first robust method is widely used in ATLAS and achieves a satisfactory performance, the latter technique has not been fully established yet using NEWT in the ATLAS offline reconstruction. A combined refit is not in particular expected to gain on the track resolution substantially when being compared to resolution given by the respectively better single sub-detector[14], but should help to recover tails in the combined track resolution, since the combined fit allows to redefine outlier measurements or to re-weight hit contributions to the final track representation. The complete track refit will in addition play an important role in the global alignment of the ATLAS detector. Various inevitable requirements for the establishment of combined refitting have been met already: the common `Track` class as a well

[13] Most of these tools are located in the RecoTools CVS package of the ATLAS offline repository.

[14] It can be shown that the ID performance in terms of momentum resolution is superior to the MS for low momenta, while for high momenta the outer Muon Spectrometer outplays the Inner Detector due to the higher magnetic field and larger dimension.

defined output of track reconstruction programs (including the entire common hit and measurement descriptions), the creation and deployment of a global reconstruction geometry and the realisation of tracking tools that work on complete technology independent base class level. First attempts of combined refitting have been done [25] and show promising results and the validity of the applied software design.

6 Conclusion and Outlook

Four years after the public release of the *Final Report of the Reconstruction Task Force*, a new modular track reconstruction software has been established that includes a common EDM, an underlying reconstruction geometry and is based on well-defined interfaces in a component pattern design. A first complete model of the inside-out track reconstruction has been deployed in the Inner Detector and is competitive in terms of performance and CPU time consumption to prior ATLAS reconstruction programs, while providing an open and interactive model for future modifications and adoptions. The second sequence based on common tracking tools, the outside-in tracking, is close to completion. NEWT builds in addition the backbone of the ATLAS Inner Detector Event Filter by providing the seeded pattern recognition and fitting tools used in the Event Filter trigger slices.

Moreover, all track reconstruction algorithms now provide the EDM objects as the output data, allowing common validation and analysis applications to work independently on the reconstruction chain used for track finding and fitting.

6.1 Outlook

The full migration of the previous track reconstruction programs into the New Tracking schema is an ongoing effort of the ATLAS offline software project. NEWT was designed to allow both, the easy introduction of newly developed concepts as modules to the common tracking effort and the re-integration of the existing well-performing algorithms from past reconstruction packages. Latter is deliberately foreseen without wiping the identity (in both concepts and performance) of these well-tested algorithms. Recently, main components such as the intersection algorithms from the Inner Detector stand-alone `iPatRec` [26] application have been integrated into the extrapolation engine of New Tracking, and is followed by an ongoing effort to employ internally the common tracking EDM. Finally, the track fitting `AlgTool` from `iPatRec` will be integrated as another implementation of the common `ITrackFitter` interface to provide full flexibility to the user of the ATLAS offline reconstruction.

A second big field of further development is the realisation of combined reconstruction and recovery and detection strategies of bremsstrahlung radiation using the ATLAS New Tracking tools. Clearly many parts of NEWT that have been described in this document can become parts of new `Algorithm` chains that concentrate on higher stage pattern recognition, particle identification and event topology classifications.

Acknowledgments

The authors would thank the various contributers to both fields the conceptual design as well as the software implementation of the ATLAS New Tracking.

A Appendix

A.1 Typesetting and Conventions

The following type setting conventions are followed throughout this document: software packages within the ATLAS offline software repository [27] are written in Sans-Serif face, C++ or python class names are written in `Courier` face. Namespace definitions as used in the software repository are omitted in this document for readability. Table 4 gives an exhaustive list of the active container packages in the Tracking repository, adding short descriptions of their content.

The software illustrations comply, in general, with Unified Modeling Language (UML) [28] standards, extensions to the UML standard are identified explicitly in the figure capture.

Table 4: Active container packages and brief description of their content in the Tracking CVS repository

Container	description
TrkAlgorithms	Tracking specific `Algorithm` classes to be executed once per event, i.e. truth association and conversion to persistent objects
TrkCBNT	legacy package to fill the Combined Ntuple (CBNT)
TrkDetDescr	container package for the reconstruction geometry, concentrating the installed classes and geometry builders
TrkDoc	documentation package
TrkEvent	container package for all tracking EDM base classes, concrete implementations may be found in the InnerDetector or MuonSpectrometer, respectively
TrkEventCnv	converter packages for persitification process
TrkExtrapolation	container package for the extrapolation engine
TrkFitter	track fitters implementing the `ITrackFitter` interface
TrkMagneticField	magnetic field access tools, magnetic field parameterisations
TrkTools	common tracking tools other than fitting, extrapolation, vertexing
TrkUtilityPackages	mathematical utilities, such as a Hough transform
TrkValidation	common validation package define on EDM base class level
TrkVertexFitter	vertex fitting interface and implementations

A.2 Index of Abbreviation and Acronyms

AKF *Alignment Kalman Filter* is a dedicated extension of the Kalman Filter that incorporates geometrical updates to the reconstruction geometry

AOD *Analysis Object Data* is the compressed event data dedicated for physics analysis

ATLAS *A Toroidal LHC ApparatuS*

CSC *Cathode Strip Chambers* are a technology used in the Muon System

CVS *Concurrent Versions System* is the used code archive and versioning software in ATLAS

CTB2004 *Combined Test Beam* was a combined test of a full sectorial ATLAS slice in 2004

DAF *Deterministic Annealing Filter* is an extension of the Kalman filter with an additional annealing schema

DKF *Distributed Kalman Filter* is a fast version of the Kalman filter designed for the Trigger and Event Filter

DNA *Dynamic Noise Adjustment* is a special extension of the Kalman filter in ATLAS

EDM *Event Data Model* - nomen est omen

EF *Event Filter* is the third level, software based trigger in ATLAS

ESD *Event Summary Data* is the uncompressed event information

GSF *Gaussian Sum Filter* is a special multi-Gaussian extension of the standard Kalman Filter

ID *Inner Detector* is the inner tracking device of the ATLAS detector

LVL1 *First Level Trigger* in ATLAS, purely hardware based

LVL2 *Second Level Trigger* in ATLAS

MS *Muon System* is the outer tracking device for muons of the ATLAS detector

MDT *Monitored Drift Tubes* are used in the ATLAS Muon System

NEWT *New Tracking*

RCP *Resistive Plate Chambers* are another technology in the ATLAS Muon System

ROI *Region of Interest* a defined region for further processing in the Trigger chain

RTF *Reconstruction Task Force* - a task force held in 2003 focussing on the evolution of the ATLAS event reconstruction

SCT *Semi Conductor Tracker* is the middle part of the ATLAS Inner Detector

STL *Standard Template Library* is a widely used container library for C++

TGC *Thin Gap Chambers* are used in the ATLAS Muon System

TRT *Transition Radiation Tracker* is the outermost part of the ATLAS Inner Detector

UML *Unified Modeling Language* is a standard in software modeling and illustration

References

[1] V. Boisvert et al, *Final Report of the ATLAS Reconstruction Task Force*, ATLAS Note, ATL-SOFT-2003-010, 2003.

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

[3] ATLAS Quality Assurance Group, *Atlas C++ Coding Standard Specifications*, ATLAS Note, ATL-SOFT-2002-001, 2002.

[4] Atlas Collaboration, *ATLAS Computing Technical Design Report*, ATLAS TDR, CERN-LHCC-2005-022, 2005.

[5] G. Barrand et al., *GAUDI - A software architecture and framework for building LHCb data processing applications*, Proc. of CHEP 2000, 2000.

[6] E. Obreshkov , et. al, *Organization and management of ATLAS software releases*, ATLAS Public Note, ATL-SOFT-PUB-2006-008, 2006.

[7] Lixin Tao, Xiang Fu and Kai Qian, *Software Architecture Design - Methodology and Styles*, Stipes Publishing L.L.C., ISBN 1-58874-621-6, 2006.

[8] A. Salzburger, M. Wolter and S. Todorova, *The ATLAS Tracking Geometry Description*, ATLAS Public Note, ATL-SOFT-PUB-2007-004, 2007.

[9] A. Salzburger, *The ATLAS Extrapolation package*, ATLAS Public Note, ATL-SOFT-PUB-2007-005, 2007.

[10] R. Frühwirth et al., *Application of Kalman Filtering to Track and Vertex Fitting*, Nucl. Inst. Meth., A 262, 1987.

[11] V. Kartvelishvili, *Electron bremsstrahlung recovery in ATLAS*, Proceedings of the 10th Topical Seminar on Innovative Particle and Radiation Detectors (IPRD06), Siena, 2006.

[12] R. Frühwirth, A. Strandlie, T. Todorov and M. Winkler, *Recent results on adaptive track and multitrack fitting*, Nucl. Inst. Meth., Volume 502, 2003.

[13] S. Fleischmann, *Track Reconstruction in the ATLAS Experiment : The Deterministic Annealing Filter* , CERN-THESIS-2007-011, 2007.

[14] R. Frühwirth, A. Strandlie, *Track finding and fitting with the Gaussian-sum Filter*, Proc. of CHEP 1998, 1998.

[15] R. Frühwirth, T. Todorov and M. Winkler, *Estimation of detector alignment parameters using the Kalman fitter with annealing*, Journal of Physics G: Nuclear and Particle Physics, 29, 2003.

[16] L. Buggem J. Myrheim, *Tracking and Track Fitting*, Nucl. Inst. Meth., 179, 1981.

[17] N. Konstantinidis et al., *Fast Tracking for the ATLAS LVL2 Trigger*, Proceedings of CHEP2004, Interlaken, 2004.

[18] I. Gavrilenko, *Description of Global Pattern Recognition Program (XKALMAN)*, ATLAS Internal Note, ATL-INDET-97-165, 1997.

[19] D. Wicke, *A New Algorithm For Solving Track Ambiguities*, DELPHI 98-163, PROG 236 TRACK 92, 1998.

[20] R. Duda and P. Hart, *Use of the Hough Transformation to Detect Lines and Curves in Pictures*, Comm. ACM, Vol. 15, 1972.

[21] T. Koffas, *private communication.*

[22] The ATLAS Collaboration, *ATLAS High Level Trigger, Data Acquisition & Controls - Technical Design Report*, CERN-LHCC-2003-021, 2003.

[23] ATLANTIS homepage, *http://cern.ch/atlantis*

[24] T. Cornelissen, *CTBTracking: track reconstruction for the testbeam and cosmics*, ATLAS Internal Note, ATL-INDET-INT-2006-001, 2006.

[25] T. Cornelissen, *Track Fitting in the ATLAS Experiment*, PhD Thesis, ISBN 90-6464-051-3, 2006.

[26] R. Clifft, A. Poppleton, *IPATREC: inner detector pattern-recognition and track-fitting*, ATLAS Internal Note, Soft-94-009, 1994.

[27] Atlas offline software repository, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*

[28] M. Fowler, K. Scott, *UML Distilled*, Addison Wesley Longman, ISBN 0-201-32563-2, 1997.

8.3 Single Track Performance of the Inner Detector New Reconstruction

ATL-INDET-PUB-2008-002, 20 p.

Acknowledgements

The presented validation study was entirely carried out by the author; in the spirit of acknowledging the result as an outcome of many well working components the main contributers to the standard NEWT reconstruction are granted authorship to the presented note.

Bibliography

[8.1] The ALTAS Collaboration. *ATLAS Computing Technical Design Report*. Technical Design Report ATLAS. CERN, Geneva, 2005.

[8.2] P. Mato et al. GAUDI-Architecture design document. Technical Report LHCb-98-064, CERN, Geneva, 1998.

[8.3] V. Boisvert et al. Final report of the atlas reconstruction task force. *Public ATLAS Note*, ATL-SOFT-2003-010, 2003.

Single Track Performance of the Inner Detector New Track Reconstruction (NEWT)

T. Cornelissen, M. Elsing, I. Gavrilenko
CERN
W. Liebig
NIKHEF, Amsterdam, The Netherlands
A. Salzburger*
Leopold Franzens Universität Innsbruck, Austria & CERN

for the ATLAS Inner Detector Software Group

March 24, 2008

ATL-INDET-PUB-2008-002
24 March 2008

Abstract

In a previous series of documents we have presented the new ATLAS track reconstruction chain (NEWT) and several of the involved components. It has become the default reconstruction application for the Inner Detector. However, a large scale validation of the reconstruction performance in both efficiency and track resolutions has not been given yet. This documents presents the results of a systematic single track validation of the new track reconstruction and puts it in comparison with results obtained with different reconstruction applications.

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*

*Corresponding author: Andreas.Salzburger@cern.ch

1 Introduction

In a previous document [1] we have presented a newly established track reconstruction application (NEWT) that recently became the default reconstruction strategy for the ATLAS *Inner Detector* (ID). NEWT is characterised by a component model architecture and is based on a common event data model (EDM), which is together with other key parts of the new track reconstruction program described in another series of documents to very detail (see [2] to [5]). However, a global performance overview of NEWT has been missing in this context. Recently, a large scale validation of the ATLAS ID performance using single track events has been carried out for the existing reconstruction programs in preparation for an exhaustive document [6] that describes the setup and performance of the entire ATLAS detector just before data taking starts in 2008. The large scale validation has been initially performed with the ATLAS offline release 13.0.30, and revealed several problems that have been — at least partly — solved in the meantime. This document aims to fulfill a two-folded purpose: it marks the first large scale validation of the NEWT reconstruction chain, but should also show points of weakness or potential problems to be revised towards first data taking. It presents various performance figures of NEWT and puts them also in comparison to the second, powerful track reconstruction iPatRec [7] that has been a reference for ATLAS ID track reconstruction for many years.

Many modules of NEWT have currently spread out into realms of combined reconstruction and track reconstruction in the *Muon Spectrometer* (MS). Being immature in comparison to the well established NEWT chain in the ID, this document will restrict itself to the inner tracking devices and leave similar considerations for the MS to the future. The presented performance figures will also be restricted to single particle events and will cover only the first sequence — the inside-out track search — of NEWT. Vertex reconstruction, particle identification and heavy quark tagging will not be covered, since they are not particular to one reconstruction program[1].

1.1 Document Structure and Typesetting

The structure of this document reflects the task of track reconstruction, which can be — in a classical picture — divided into pattern recognition and track fitting. Section 2 focusses on the reconstruction efficiencies that reflect to a large extent the quality of the pattern recognition modules. Section 3 covers in the following the final track parameter resolutions that usually incorporate several more aspects: the reconstruction material model, the quality of the measurement description and outlier handling. It should be remarked that this strict division of pattern recognition and track fitting is usually not valid when dealing with full physics events. For single track events, however, such a factorised investigation is indeed useful and can be done without compromising either of the two parts.

Blocks written in `Courier` face refer to actual software implementations.

1.2 Used Event Samples

Single particle events with large statistics and constant transverse momentum p_{T} (50 000 single particle events per p_{T} value) have been produced for the presented validation of NEWT. The transverse momentum values spread in discrete steps from 1 GeV to 2 TeV; very large transverse momenta are used for charge misinterpretation studies. The production vertex for the single particles has been smeared according to standard ATLAS primary vertex smearing, i.e. Gaussian smearing with $\sigma_{xy} = 15$ μm in the transverse plane and $\sigma_z = 56$ mm in the longitudinal plane. The single tracks have been produced according to flat distributions in a pseudorapidity range of $|\eta| < 3.0$ and uniformly spread over the entire azimuthal angle of 2 π.

Event simulation and digitisation have been done with ATLAS software release 13.0.30; the reconstruction has been using release 13.0.30 for iPatRec and 13.2.0 for NEWT[2]. The default track fitter for this release has been the `KalmanFitter` implementation. The event digitisation has been restricted to simulate electronic noise hits only on detector modules that had recorded hits from primary or

[1] Thanks to the nowadays fully established common EDM, these modules can be performed regardless of the used track reconstruction program.

[2] After 13.1.0, the TRT error description has been changed to a more realistic drift time dependent model. The current tuning parameters of iPatRec have not been adapted accordingly and thus the former 13.0.30 release has been used in this comparison.

secondary tracks from the full detector simulation; for single track events, this simplification is totally valid[3] and speeds up the digitisation process significantly. The used detector layout was CSC-02-00-00.

The authors would also like to emphasise that the presented results are in consideration of the most optimistic scenario that includes perfect alignment and exact knowledge of the inert detector material and calibration. In this sense, the given performance figures should be regarded as the *potential* performance of the ID track reconstruction, while it is clearly recognised that these performance numbers will not be met with initial data.

1.2.1 Statistical Uncertainties and Method Description

The quoted results are produced with 50000 events per given momentum range and in the pseudorapidity of $|\eta| < 3$, with the implicit restriction of the ID acceptance region up to a pseudorapidity value of $|\eta| < 2.5$. Efficiencies, fake rates and resolutions are either given for the full η range and rely thus on a statistic of 40 000 events, or they are quoted for either 10 or 20 equidistant $|\eta|$ bins. The single bins contain consequently about 4000 or 2000 generated single particle events, respectively.

Resolutions are given as the root mean square (RMS) of the residual distributions. The boundaries for the according distributions are estimated, such that 99.7 % (i.e. an equivalent of 3 σ of a Gaussain distribution) of all entries are included in the RMS calculation. It should be remarked that this definition includes most of the significant tails that are present in e.g. the tack parameter residuals of pions and that a Gaussian fit to the core of the distribution would lead to improved resolution figures. The method chosen should, however, guarantee compatibility with the resolution calculation as presented in [6], but should also allow for the non-Gaussian character of some distributions to be respected while guaranteeing one consistent definition throughout the document. Quality cuts are applied on the reconstructed tracks, and described in more detail in Sec. 2.1. Hit pulls and residuals are given as the unbiased quantities on the associated reference frames (i.e. the primary vertex or measurement surfaces). This means that if a measurement has been present in the very same examined track frame, it has not been used for the residual or pull calculation.

The relative uncertainty for the RMS of track parameter residuals is thus about 0.5 % for the total sample and either 1.6 % or 2.2 % for each $|\eta|$ bin. For hit residuals, the statistical error is even smaller since the event statistic is multiplied by the according hit multiplicities per track. Errors on efficiencies and fakes are calculated using the method given in [8]. In general, if errors are not shown or visible in figures, they are within the regime of the marker size and thus omitted for convenience.

2 Reconstruction Efficiencies

The definition of any reconstruction efficiency is controversial itself; it relies on the definition of what is regarded as to be *successfully* reconstructed on the one hand, but also on the definition of the signal on the other hand. In some sense, this is remains a decision of taste, and — when analysing future taken data with the detector, this question is anyway impossible to answer. Keeping latter in mind, the authors tried to impose quality cuts as they may be used in future analyses of ATLAS data rather than relying on stringent truth matching criteria. The reconstructed tracks are as such not matched by an identifier to the generated particle, only a hit matching probability to one particle is required; even if this particle is a secondary particle (e.g. coming from an interaction at the beampipe or an innermost layer), it would be regarded as a successfully reconstructed particle as long as the quality cuts are fulfilled. It should be mentioned that it is not always possible in this context to separate detector effects from the pure software performance. The authors will, however, try to identify the contributing effects to the overall track reconstruction efficiencies and resolutions.

2.1 Cut and Efficiency Definitions

The applied cuts on generator level are mainly targeted at providing a clean signal sample that is characterised by prompt tracks within the geometrical acceptance region of the detector. Ensuring

[3] There is little chance that a bunch of noise hits can be misidentified as a track, in particular when applying track quality cuts that have been used in this study, see Sec. 2.1.

only prompt tracks is respected on two levels: a generator identifier number, the so-called *barcode* is used to restrict the signal to primary particles is used together with a restriction in the transverse and longitudinal displacement to the primary vertex (expressed though the perigee parameters d_0, z_0 and θ, which are in more detail described in Sec. 3.1). The latter restriction is superfluous for single track events, since it is inert to the barcode requirement. As it is part of a common set of generator cuts that has been applied, it is however mentioned here for completeness. On reconstruction level, on the other side, the impact parameter restriction is indeed needed to identify prompt tracks.
The following cuts on generator level have been applied for tracks:

- *tracker acceptance region:* $|\eta^{gen}| < 2.5$
- *prompt tracks:* barcode < 100000
- *primary vertex:* $|d_0^{gen,PV}| < 2$ mm, and $|z_0^{gen,PV}| \cdot \sin\theta^{gen} < 10$ mm

They define the set of tracks which should have been recorded by the detector and reconstructed by the tracking algorithms to later fulfill the some quality cuts. These quality cuts are mainly motivated by the interest to use the found tracks in context of meaningful physics analyses:

- *tracker acceptance region:* $|\eta^{rec}| < 2.5$
- *primary vertex:* $|d_0^{rec,PV}| < 2$ mm, and $|z_0^{rec,PV}| \cdot \sin\theta^{rec} < 10$ mm
- *hit quality cut silicon:* a number of minimum 7 hits in the silicon detector is required;

Both generator and reconstruction cuts rely on the impact parameters d_0 and z_0 to be expressed with respect to the generated primary vertex (PV). In the data taking scenario, quality cuts on the impact parameters will be applied, when the point of closest approach is expressed w.r.t. the reconstructed primary vertex; in single track events this is clearly not applicable, since vertex reconstruction can not be performed. The choice of the generated vertex in the presented study helps in addition to cancel the influence of vertex reconstruction algorithms on efficiency calculations and track parameter resolutions.
When momentum resolutions are quoted, an additional constraint of a successful extension of the silicon track candidate into the TRT is required. This is because the TRT contributes significantly to the momentum resolution. A successful track match is defined if at least 80 % of the hits contributing to the track fit (i.e. outliers are omitted) correspond to one particle trajectory[4]. The reconstruction efficiency is henceforth defined as

$$\epsilon^{rec} = \frac{N^{rec}_{match}}{N^{gen}}, \tag{1}$$

where N^{rec}_{match} denotes the number of tracks that comply with the reconstruction cuts and are matched to a generated prompt particle, and N^{gen} is the number of tracks that are within generator cuts. Contrary to successfully matched tracks, *poor* and *fake* tracks are defined as the number either poorly matched tracks N^{rec}_{poor} or unmatched tracks $N^{rec}_{unmachted}$ that have still passed the reconstruction quality cuts. The matching criteria are then defined as:

- *poorly matched*: more than 50 %, but less then 80 % of hits match with one generated particle trajectory;
- *unmatched*: less than half of the hits originate from one particle.

Consequently, the *fake rate* ζ and *poor rate* ξ are defined as

$$\zeta = \frac{N^{rec}_{unmatched}}{N^{rec}} \text{ and } \xi = \frac{N^{rec}_{poor}}{N^{rec}} \tag{2}$$

[4]The definition of such a matching criteria to the truth trajectory is not always trivial. This is due to the fact that during the full simulation strong interaction processes of the particle with detector material (such as the emission of a hard brem photon) change the particle identification. The track matching has therefore to be done with an entire truth trajectory collection that follows the original generated particle during these changes.

where N^{rec} is the number of all tracks that have been successfully reconstructed within the quality cuts. Clearly, by broadening the reconstruction quality cuts, higher efficiencies can be achieved. However, the risk of contaminating the track output sample with fake or poorly measured tracks increases.

Reconstruction efficiencies and fake rates are strongly dependent on the particle type, since they are highly sensitive to the interaction process between the particle and the detector material. Minimum ionising particles yield the highest reconstruction efficiencies (in the given momentum range of final state particles this is close to 100 % over the entire acceptance region of the ID, see Sec. 2.1.1). Hadronic particles[5] are degraded by nuclear interactions with the detector material and show lower reconstruction efficiencies. Finally, electrons suffer strongly from radiation loss and lead to the lowest reconstruction probability for a large part of the investigated momentum range, in particular when reconstruction quality cuts are applied.

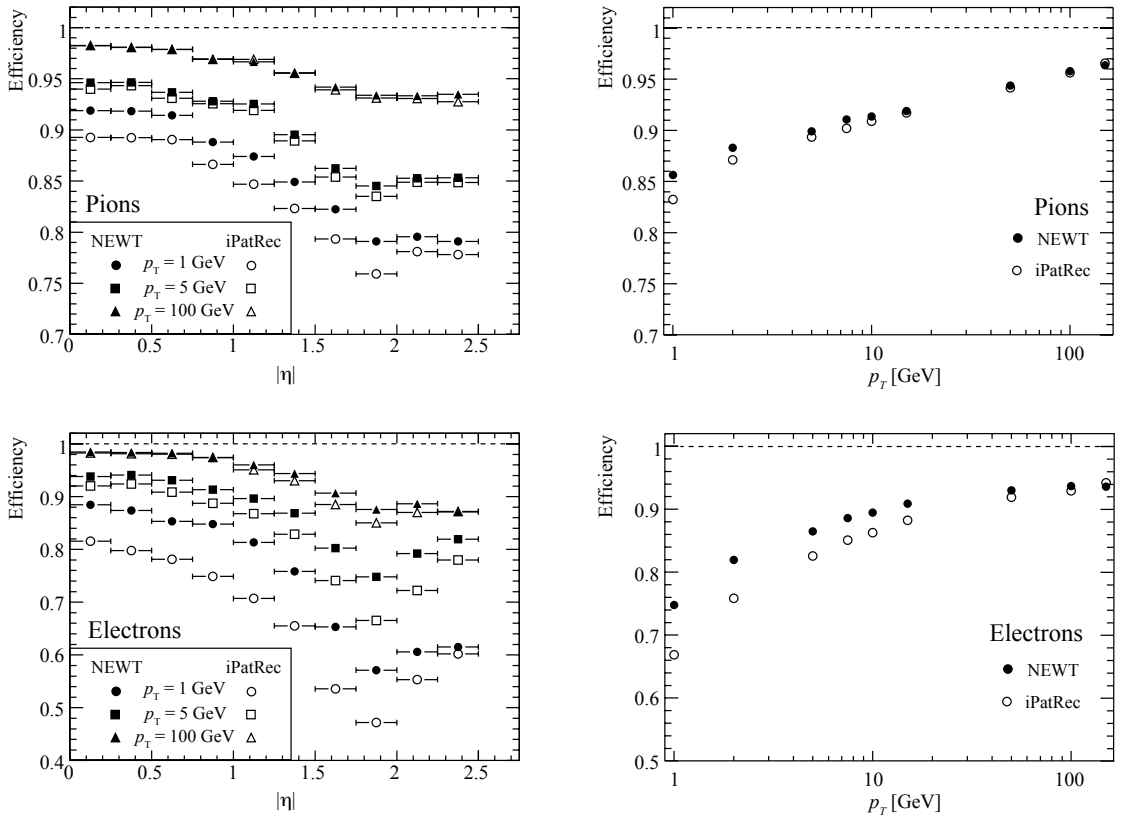


Figure 1: Reconstruction efficiencies and for prompt pions and electrons in single track events for different particle momenta. The closed markers represent the NEWT performance, while the open ones show in comparison the obtained results for iPatRec. The plots to left shows the reconstruction efficiencies in dependence of the pseudorapidity for certain transverse momenta; right: total reconstruction efficiency and fake rate depending on the p_T value is shown.

The reconstruction efficiencies for single π^+ and e^- tracks are shown for NEWT and iPatRec in Fig. 1. It can be seen that under application of the given reconstruction and matching cuts NEWT yields slightly higher reconstruction efficiencies than iPatRec for the major part of the momentum range. Only for particles with p_{T} higher than 100 GeV iPatRec tends to be similarly efficient (pions) or even slightly superior (electrons). One other aspect is revealed in this picture: while electron tracks at low momenta have a relatively low reconstruction efficiency compared to single pion tracks, the situation is flipped for high momentum tracks. The reason for this is that bremsstrahlung effects are

[5]In the ATLAS track reconstruction, only three types of particles are considered: muons, electrons and pions. No distinction between different hadrons is done, the particle identification is carried out at a later stage in the event reconstruction process.

most pronounced at lower momenta, while they lose importance the stiffer the track gets. In addition, when hadrons undergo nuclear interactions which can effectively lead to a loss of the original particle. Hence, there is a maximum limit for pions to be reconstructed in the ID.

Fake and Poor Rates No significant fake rates could be observed for the three different particle types and the two reconstruction strategies. However, tracks with poor hit matching are present in small but measurable quantities in pion and electron samples and they appear more likely at higher initial particle energies, see Fig. 2. This may be due to the fact that induced secondary particles are mostly collinear to the initial particle direction and their hits are thus more likely to be (falsely) associated to the initial track.

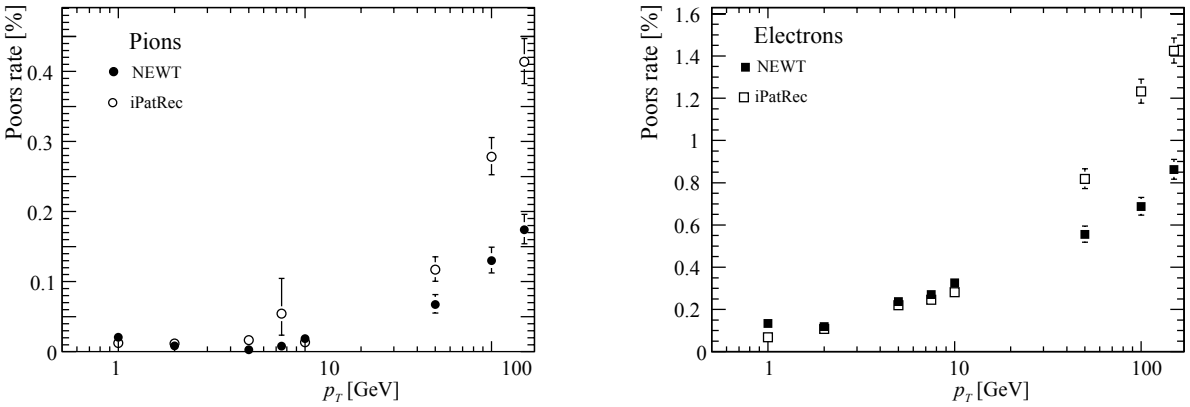


Figure 2: Rates for poorly matched tracks that have passed the reconstruction quality cuts for single pions and electrons comparing NEWT and iPatRec.

Both iPatRec and NEWT show a very small rate of poorly associated tracks that stays well below 1 % for the momentum range up to 100 GeV. A slightly lower poorly associated track rate for NEWT can be observed.

2.1.1 Single Component Performance

In the previous section, the *global* performance statistics of NEWT has been quoted; these numbers represent the results from the main sequence of the NEWT ID tracking, the inside-out track search. The NEWT inside-out sequence is build from two main parts: the silicon track finding and the extension of tracks into the outer *Transition Radiation Tracker* (TRT). The modular design of NEWT allows to split the performance validation into individual software components. Reconstruction efficiencies and fakes or poor rates can thus also be given independently for the track candidates found in the silicon detector, the resolved track collection after the first ambiguity solving and final track after extension into the outer tracker. In this situation, the advantage of the component module design of NEWT is revealed: since the EDM objects that are passed between the several steps are well defined objects, common validation algorithms can work on any of the given steps: the independent validation modules can be plugged in at several places into the algorithm flow, or even be called on an on-demand basis. A further description of this mechanism can be found in the original reference [1], where the NEWT philosophy is explained to very detail.

Figure 3 shows the reconstruction efficiencies for single muon tracks of high and low transverse momenta for NEWT in the two main stages of the inside-out reconstruction chain: the candidate track collection built from seeds in the silicon detector and the track collection after passing the ambiguity solver and extension into the TRT. The latter can — under the given matching conditions — increase the efficiency since they may overrule a misidentified silicon cluster (this can be either through an increase in the matching probability or simply by removing the fake cluster from the track fit in the outlier logics). The efficiency drop at a pseudorapidity of about 1.4 could be identified as a significantly increased number of failed track fits in comparison to other $|\eta|$ bins, however, the candidate track is provided in these cases by the pattern recognition process.

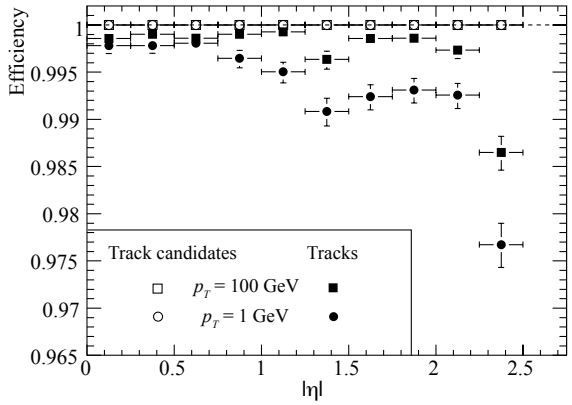


Figure 3: Reconstruction efficiencies for single muon tracks with low and high transverse momenta as they are processed by the two main parts of the NEWT inside-out reconstruction sequence. The filled symbols show the global efficiencies after the entire reconstruction process, while the open symbols represent the efficiency for the candidate track collection at the given transverse momenta. The efficiency drop at $|\eta| \approx 1.4$ indicates a problem in the track fitting, a more detailed investigation has identified a significantly increased number of failed track fits in this region.

It could be shown in a recent study that the exchange of the standard `KalmanFitter` with the `GlobalChi2Fitter` [9] successfully recuperates the loss of efficiency in the given pseudorapidity ranges. Figure 4 shows a comparison of reconstruction efficiencies for low and high transverse momentum muon tracks with both fitting techniques as the standard fitter used in the NEWT reconstruction chain. This study has been carried out with a development release that is close to the expected performance of the ATLAS 14.0.0 release. It can be seen that also the efficiency of single muon tracks when using the default `KalmanFilter` implementation has increased due to a modified track scoring approach in the ambiguity solving process.

There are few more striking arguments for the component software model of NEWT than this simple example: since both fitter types implement the same `ITrackFitter` interface and can be loaded independently at run-time, the change from one to yet another fitter type is simply a one-line change in the job configuration. It reveals how the component model helps to react rapidly when a problem in one single part is found.

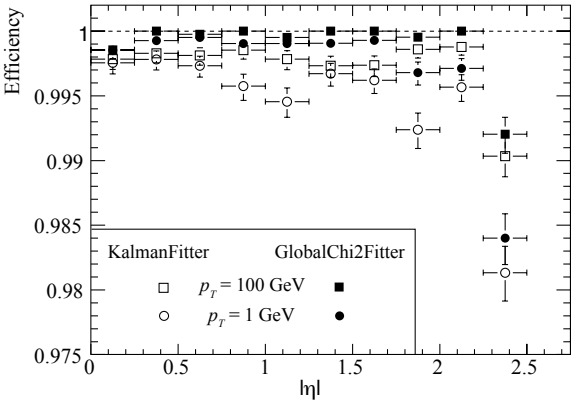


Figure 4: Reconstruction efficiencies for single muon tracks with two different fitter types being used in NEWT. The current `KalmanFitter` implementation in release 13.2.0 shows an increased number of failed track fits, while the `GlobalChi2Fitter` recuperates many of these tracks and yields almost 100 % reconstruction efficiency over the entire acceptance range. An updated version of the `KalmanFitter` is currently worked on to be integrated in the next ATLAS software release.

TRT Extension In the track extension from the silicon detectors to the TRT, the hit statistics of the entire track can be changed; since the track fit is performed on a new, expanded hit collection outlier definitions can be changed and thus even reconstruction efficiencies may be modified slightly, see Fig. 3. On the other hand, the TRT extension is not capable of rejecting a track that has survived the ambiguity solving process[6]. A highly efficient TRT extension is required, since the longer lever arm and the additional precise measurements further along the track contribute significantly to the momentum resolution. Figure 5 shows the extension efficiency for NEWT depending on different particle types at a transverse momentum $p_{\mathrm{T}} = 100$ GeV, such as the effect of the TRT extension on the momentum resolution.

[6]In general, a failed track fit acts as an intrinsic track rejection, but in this specific case of the precise TRT extension it indicates usually rather a fitting problem than a classical track rejection.

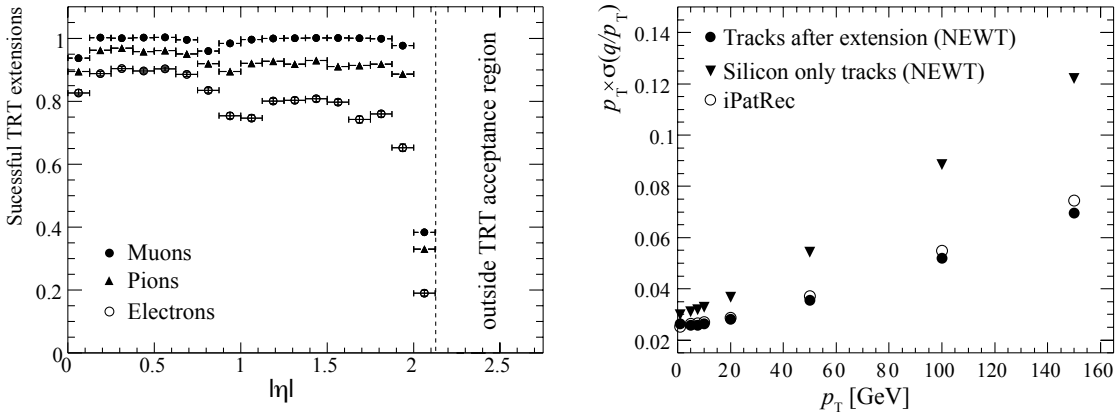


Figure 5: Extension efficiency into the TRT for single particle types of $p_{\mathrm{T}} = 100$ GeV; a successful TRT extension is hereby defined by finding at least 10 TRT hits; the end of the TRT acceptance region at a pseudo-rapidity slightly higher than 2 can be seen. The plot to the right compares the momentum resolution for tracks fitted in the silicon detector parts (triangles) with tracks after the TRT extension process (closed circles) without explicitly requiring a successful TRT extension. The result of the iPatRec default track collection is displayed as open circles.

There are three regions where the extension probability into the TRT is slightly degraded, all of which have a geometrical reason: at the centermost bin there is a small gap between straws on the positive and negative side of the TRT barrel, leaving the possibility that a particle can pass through without interaction with any (or with too few) sensitive detector parts; at $|\eta| \approx 0.75$ there is a lower number of hits on track due to the transition from the TRT barrel straws to the endcap discs; this region also marks the highest material distribution in the Inner Detector. This effect is also reflected in a change of momentum resolution in this particular area. Finally, at large pseudorapidity values the geometrical acceptance region of the TRT is exceeded.

3 Track Parameter Resolutions

The track parameter resolutions are the ultimate quantities to describe the reconstruction quality: they accumulate many effects that can not always be easily disentangled. Material effects integration (and thus the quality of the reconstruction material description) contributes alongside to effects from the pattern recognition (through wrong or missed hit assignments), but also calibration data and clusterisation algorithms.

3.1 Used Track Parameterisation

The track parameterisation in the ATLAS EDM [2] is slightly modified w.r.t. the helix-based track expression that has been widely used in many former high energy physics experiments. This is, because the tracking EDM should serve the track reconstruction algorithms of both ATLAS tracking devices, the Inner Detector and the outer Muon System. Since only one global reference frame can be chosen while the magnetic field setup of the ID and MS are somewhat perpendicular (a solenoidal field *aligned* with the beam line in the ID and a toroidal field in the MS), a helical parameterisation bound to the solenoid field would leave the Muon Spectrometer with an almost meaningless choice of parameters. For this reason, a parameterisation has been chosen that is closely bound to the constants of motion in both tracking devices. A perigee representation in the Inner Detector, that is used to express the closest approach to the nominal interaction point, is described by

$$\tau_i = (d_0, z_0, \phi, \theta, q/p). \tag{3}$$

The following comparison, however, is entirely based on ID tracks and for the convenience of backward comparison with formerly published performance figures [10], a purely helical based track parameter-

isation is chosen, which yields

$$\tau_i = (d_0, z_0, \phi, \cot\theta, q/p_{\mathrm{T}}), \tag{4}$$

keeping in mind that within NEWT the common tools act on the EDM parameters as given in Eq. (3). The track parameters are often divided into the strongly correlated transverse parameters $(d_0, \phi, q/p_{\mathrm{T}})$ and their longitudinal counterparts $(z_0, \cot\theta)$.

It should also be mentioned that for investigations of the longitudinal impact parameter a necessary transformation from the global frame to the helical frame at the point of closest approach has to be applied; taking account of this, the longitudinal component of the impact parameter in three dimensions is given as $z_0 \times \sin\theta$ and will be used as such in the following. iPatRec uses an internal event data model, the output is finally converted into the tracking EDM.

3.2 Parameter Resolutions

The following parameter resolutions are obtained for muon and pion tracks, respectively, and refer to the RMS of the residual distribution, accounting for 99.7 % of all tracks that have passed the tracking quality cuts. No further hit truth matching is applied in this context. While muons yield the best track parameter resolutions as they leave the cleanest trace in the detector, pion resolutions are of particular interest, since an overwhelming fraction of tracks in a typical event will be regarded as hadrons. Pions suffer mainly from one additional effect: they undergo nuclear interactions with the detector material, which may either effectively shorten the track length, or adds an additional deflection to the coulomb multiple scattering. The following section will summarise the results obtained for single muon and pion tracks, an exhaustive table for the different pseudorapidity bins can be found in the Appendix, Sec. A.2.

Impact Parameter Resolutions A precise impact parameter estimation is crucial for vertex reconstruction and quark flavor tagging. Figure 6 shows the transverse and longitudinal impact parameter resolutions for single muon tracks of certain momenta reconstructed with NEWT and iPatRec.

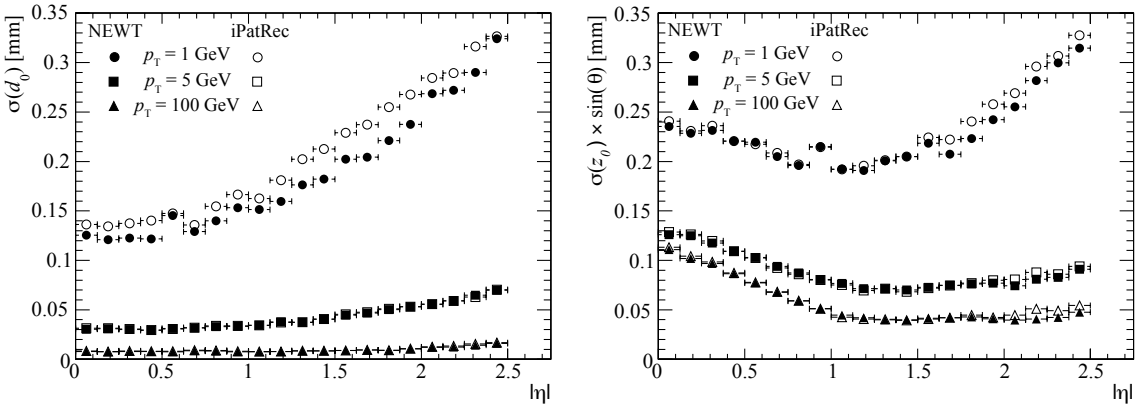


Figure 6: Transverse and longitudinal impact parameter resolutions for single muons tracks with transverse momenta $p_T = 1, 5, 100$ GeV reconstructed with NEWT and iPatRec.

In general, NEWT yields a slightly better impact parameter resolution than iPatRec, in particular for lower momenta. This indicates a better description of the detector material, that is in NEWT handled through a dedicated reconstruction geometry, the so-called *TrackingGeometry* [4]. For high momenta, the transverse impact parameter resolution approaches rapidly an asymptotic limit which is given through the intrinsic detector resolution (see, for more details, Sec. 3.3). In the high momentum limit, the results obtained with NEWT and iPatRec are almost identical, since they share the same common cluster descriptions. It can be demonstrated that the main discrepancy between the resolutions obtained with NEWT and iPatRec results from the different tail contributions that are present in the residual distributions. A brief comparison for the d_0 distribution at low momenta can be found in the appendix, Sec. A.1.

For single muon tracks the resolution of the transverse impact parameters d_0 is significantly less than 10 μm for the high momentum limit in the central region, and increase only slightly at higher pseudorapidity values. For single pions, the situation is worse: hadronic interactions with the detector material add significant tails to the impact parameter resolutions and in particular the longitudinal impact parameter component is heavily disturbed. This effect gets more pronounced at higher pseudorapidity values, since it is enhanced by the back propagation to the nominal vertex position. For large $|\eta|$ values, the longitudinal propagation distance grows accordingly fast, while the transverse projection stays roughly the same[7].

Figure 7 shows a comparison between the impact parameter resolutions obtained for single muon and pion tracks with NEWT.

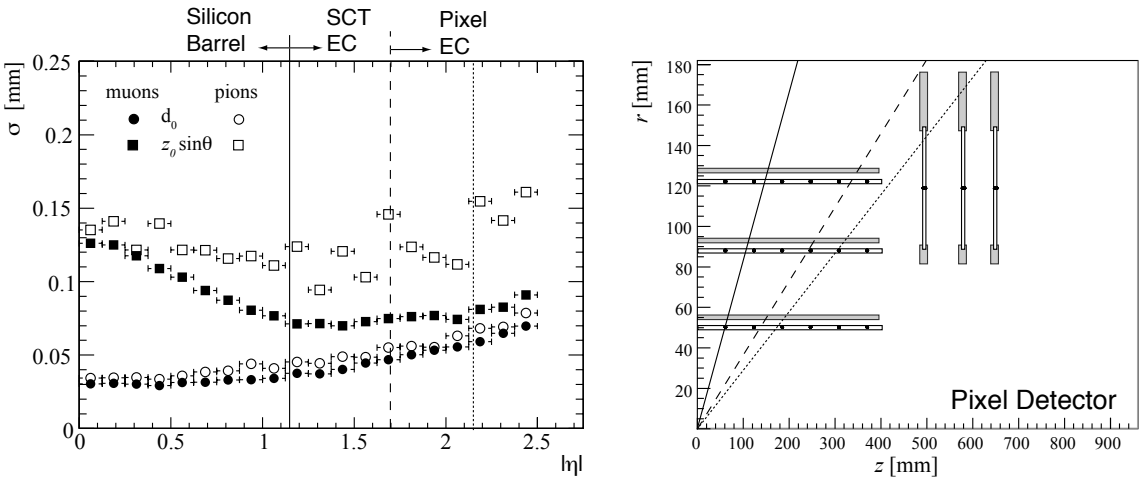


Figure 7: Comparison of the impact parameter resolutions for single muon and pion tracks with $p_{\mathrm{T}} = 5$ GeV. Hadronic interactions disturb the clean minimal ionising signature for pions and lead to an increased impact parameter resolution; this effect is most prominently pronounced for the longitudinal impact parameter resolution for high-η tracks. The strong spread and scattering structure at higher η is somewhat coincident with the endcap structures of the silicon detector. For the convenience of the reader, a simplified $r - z$ view of the pixel detector is added to the right, indicating the same $|\eta|$ boundaries as given in the left illustration.

Angular Resolutions The angular track resolutions play an important role in vertex reconstruction and via constraint fitting and their essential component to the momentum determination to any mass measurement. In a solenoidal field setup, where the trajectory of a charged particle follows closely a helix, there is a strong correlation amongst both the transverse and longitudinal track parameters, respectively. The characteristics of the angular resolutions follow thus very closely the results obtained for the impact parameter resolutions. Figure 8 shows the corresponding angular track parameter resolutions for low and high momentum single muon tracks. Again, when comparing results obtained with NEWT and iPatRec, the new tracking chain yields a higher resolution for low momentum tracks due to the improved material description of the ID.

Momentum Resolutions Obtaining an accurate momentum estimation is of similar importance as the impact parameter resolution. A correct momentum estimation is crucial for every kinematic analysis of the underlying event. In particular for electrons that are subject to significant radiation loss, this is a non-trivial task that requires dedicated fitting algorithms; electron fitting, however, is usually performed in a later stage of the event reconstruction and the following results are thus obtained for muon tracks to illustrate the potential performance of the ID tracker, see Fig. 9. At momentum resolution at high momenta is dominated by geometrical effects of the TRT detector that are reflected in both reconstruction programs: the end of the acceptance region, the barrel-endcap

[7]Since the transverse and longitudinal track parameters can be close to the center of the solenoid regarded as uncorrelated quantities, such a division between longitudinal and transverse components is indeed valid.

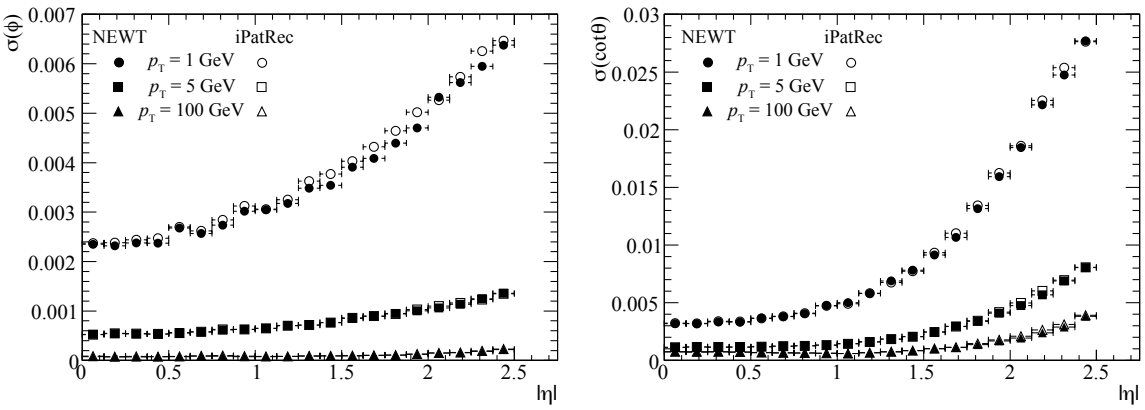


Figure 8: Angular track parameter resolutions for single muons tracks with transverse momenta $p_T = 1, 5, 100$ GeV reconstructed with NEWT and iPatRec.

crack, and the gap between the barrel straws at positive and negative side of the TRT detector can be identified similarly to the previously discussed TRT extension efficiency.

A new measurement handling in the TRT that was recently introduced in combination with a flexible decision about the drift sign has led to a gain in the momentum resolution for NEWT in comparison to iPatRec results.

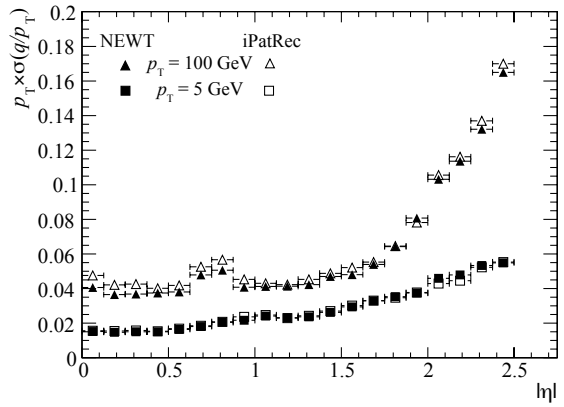


Figure 9: Relative momentum resolution for single muon tracks of different momenta in the Inner Detector. The results obtained with NEWT and iPatRec are shown in comparison and indicate a better momentum resolution of NEWT in the high momentum limit.

In contrast to the impact parameter and angular resolutions, which improve with higher momenta due to the vanishing multiple scattering effects, the momentum resolution degrades with increasing momenta, since the track become stiffer and the influence of the intrinsic cluster error becomes more pronounced. For very large momenta, there is even a possibility that the curvature sign of the track can not be clearly determined in the track fit. This can lead to an effective misidentification of the particle charge and will be in more detail discussed in Sec. 3.4. In the following section, Sec. 3.3, it will be shown that the impact parameter resolutions are dominated by the measurements close to the vertex, while for the momentum resolutions all measurements are equally important.

The momentum resolution can be improved when constraining the track in addition to the assigned hits to the beam spot (or nominal interaction point). In fully deployed physics analyses, this constraint for prompt tracks will be with respect to the reconstructed primary vertex. For the simulated single track events, a simple constraint to the smeared generated vertex has been used to demonstrate this effect. First tests with a modified refit algorithm have shown an improvement of the momentum resolution from 1 % at 1 GeV increasing to about 5 % at 100 GeV. When applying optimised track seeding in case of a beam constraint, this numbers may even be slightly increased.

3.3 The $A \oplus B$ Model

The resolution of a track parameter τ can be expressed as a function of the transverse momentum p_{T}. This is due to the fact, that — in general — multiple scattering is the dominant process noise in track fitting at low momenta, while it vanishes at high momenta and the parameter resolution is dominated by the intrinsic detector resolution. In a simplified model, the track parameter resolution can be expressed in the form

$$\sigma_\tau(p_{\mathrm{T}}) = A_\tau \oplus B_\tau / p_{\mathrm{T}}. \tag{5}$$

This expression is approximate, working well at high p_{T} (where the resolution is dominated by the intrinsic detector resolution) and at low p_{T} (where the resolution is dominated by multiple-scattering), respectively. It originates from a simplified two layer detector model, and is often referred to as $A \oplus B$ *model*. It is thus not applied as a fit to the parameter resolutions of different momenta, but rather estimated at the high and low end of the momentum spectrum. Equation (5) can also be expressed through the asymptotic resolution at *infinite* momentum $\sigma_\tau(\infty)$, which yields

$$\sigma_\tau(p_{\mathrm{T}}) = \sigma_\tau(\infty)(1 \oplus p_{\mathrm{T}}^c / p_{\mathrm{T}}), \tag{6}$$

when p_{T}^c marks the critical momentum where the contribution of the intrinsic measurement error and the multiple scattering is equal. Figure 10 shows the critical momentum for the impact parameters $(d_0, z_0 \times \sin\theta)$ and the momentum estimate q/p_{T} when reconstructing muon tracks with NEWT. The according collection of tables for NEWT can be found in the Appendix, Sec. A.2, of this document, where also the angular parameter resolutions are included.

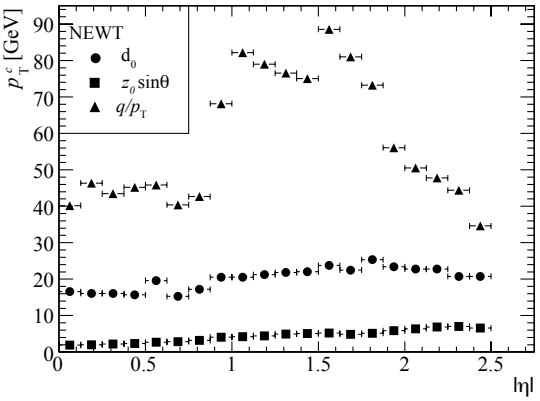


Figure 10: The critical momentum p_{T}^τ at which the intrinsic measurement error contributes equally to the track parameter resolution than the multiple scattering effects for the impact parameters and the momentum parameter for NEWT. It is remarkable to see how the transverse momentum resolution reassembles the shape of the material distribution in the Inner Detector.

3.4 Charge Misidentification

In Sec. 2.1.1 — Fig. 5 — it can be seen that the relative momentum resolution degrades with increasing transverse momentum. This can be straight-forwardly understood when regarding the momentum measurement as a *sagitta* measurement at several discrete localisations[8]. Since at very high transverse momenta the tracks appear as stiff straight lines in the tracking device, there is a possibility (evoked by the intrinsic localisation uncertainty of the detection devices) that the curvature of particle estimated in the track fit carries the wrong sign, i.e. the charge of the particle has been misidentified. Figure 11 shows the obtained inverse transvere momentum estimation for muon and electron tracks at $p_{\mathrm{T}} =$ 2 TeV. The charge misidentification probability for muons is lower than for electrons for the major part of the shown momentum range. In the very high momentum limit, however, both values seem to approach an asymptotic limit and there is almost no difference between muon and electron tracks. This observations are at first sight quite surprising: one would expect that the loss of energy due to bremsstrahlung pulls the momentum distribution towards lower momenta and thus towards a higher probability to estimate the correct sign of the track curvature. A more detailed investigation, however, reveals the nature of this unexpected behavior; electron tracks suffer from an additional effect that contributes to the charge misidentification rate: radiated bremsstrahlung photons are at

[8] In the high momentum limit, this leads to an almost linear dependency of the relative momentum resolution with the particle momentum, while for low momenta multiple scattering effects contribute also significantly.

very high energies collinear to the track direction. If they convert into an electron-positron pair also the conversion products continue in the original track direction. These particles can cause fake hits that are falsely assigned to the track and thus disturb the momentum resolution. When restricting the track collection to tracks that have a very stringent requirement of 99 % truth matching on hit level, the charge misidentification for electrons indeed falls below the rate for muons with the *help* of bremsstrahlung effect. Figure 12 shows the integrated charge misidentification for muons and pions at two different hit truth matching values, the lower bound of 80 % is hereby almost identical to no truth matching at all, since the reconstruction quality cuts already clean the track collections from the main part of a potential contamination through poor or fake tracks.

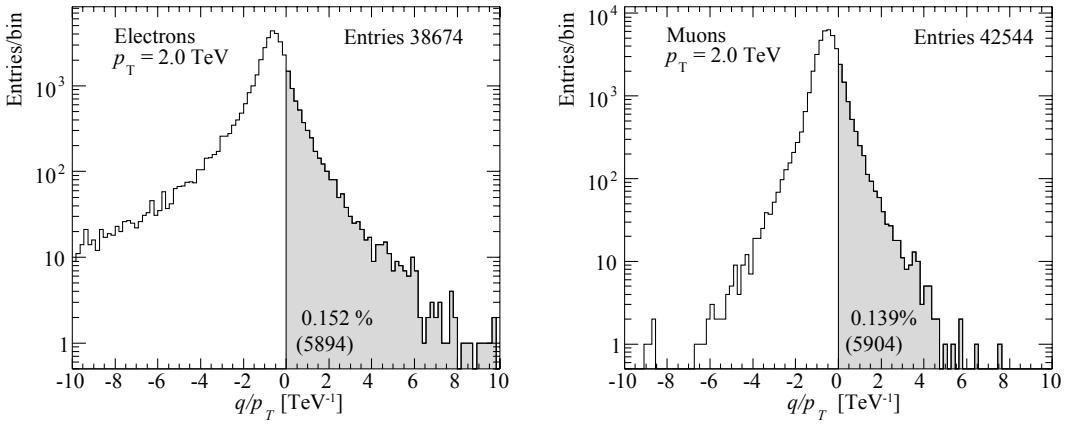


Figure 11: Fraction of misidentified (shaded area) particle charge for high p_{T} electrons and muons. For a large momentum range, the charge misidentification rate for muons is lower than for electrons. In the limit case of transverse momenta ≈ 2 TeV, electrons charge misidentification becomes slightly lower than for muons, since the energy loss contribution pulls the electrons towards lower momenta, which then feel a stronger bending power; this leads to the significantly bigger tail contributions in the electron q/p_{T} distribution. A hit truth matching requirement of 80 % has been applied for the track selection.

The charge misidentification probability is not only dependent on the transverse momentum, but also on the pseudorapidity. The dominant effect is that the TRT detector only spans over a limited pseudo-rapidity range and thus particles at $|\eta| > 2.0$ have significantly shorter track lengths, which in turn results in a worse momentum resolution — and thus to a higher charge misidentification probability. Figure 12 shows the charge misidentification probability for muons and electrons starting from 100 GeV to 2 TeV, and presents in addition the η dependency for both particle types at a fixed transverse momentum of $p_{\mathrm{T}} = 2$ TeV.

4 Error Description

The correct handling of the track reconstruction error is of great importance in conjunction with an accurate track parameter estimation. This is inevitable for almost any successive event reconstruction — beginning with vertex reconstruction and b-tagging that rely on correct covariance matrices. It is also necessary for quality and consistency checks in the reconstruction process. A stringent test in this respect is to investigate the parameter pull distributions

$$\mathrm{pull}(\tau) = \frac{\tau^{rec} - \tau^{true}}{\sigma_\tau}, \tag{7}$$

that should — if σ_τ is correctly estimated — lead to unbiased Gaussian distributions with a width of ≈ 1. Figure 13 shows the RMS of the pull distributions for the five track parameters for different transverse momenta and integrated over the entire pseudorapidity range.

Both, iPatRec and NEWT, incorporate a good error description for the impact parameters and the directional parameters of the perigee representation. The q/p_T distribution, however, shows a significant deviation from an RMS value of 1, an effect that is similarly pronounced for iPatRec and

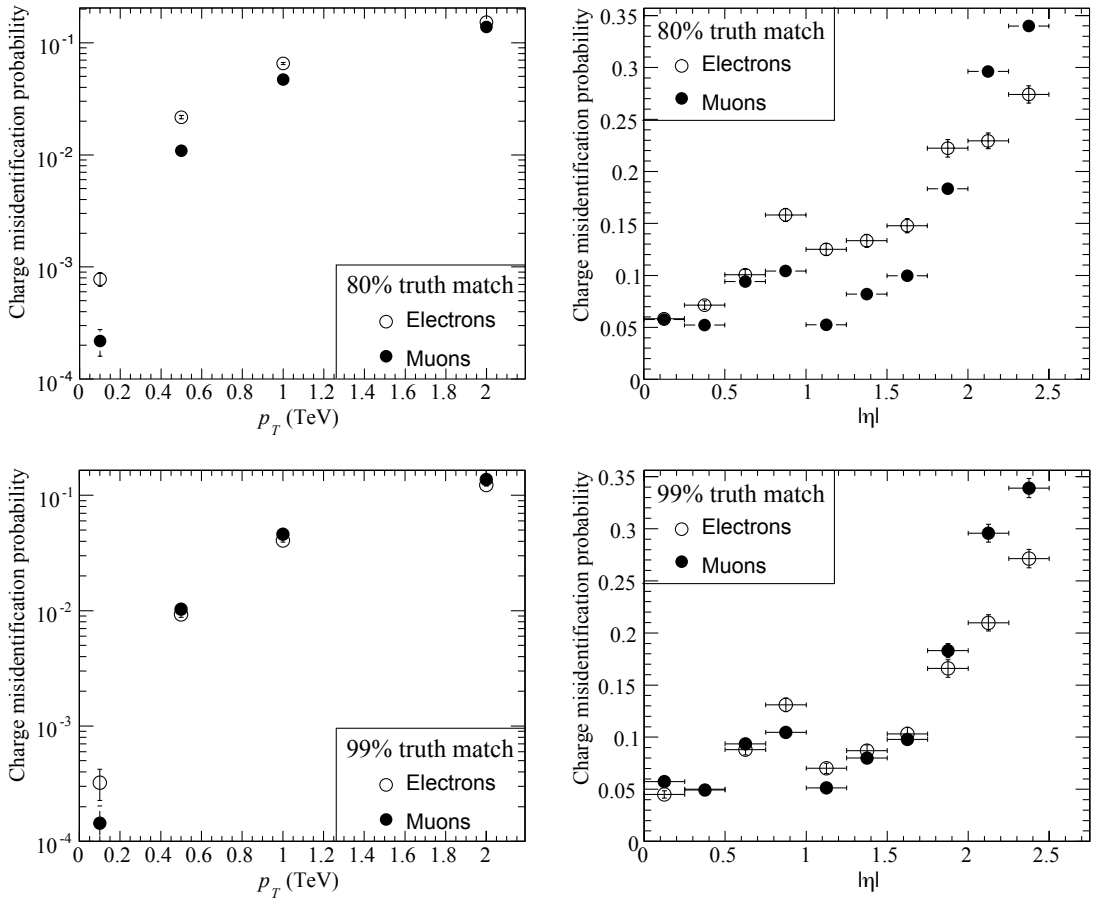


Figure 12: Fraction of misidentified particle charge for muons and electrons depending on their transverse momenta (left). Right: dependency of the charge misidentification fraction on the pseudo-rapidity for a fixed transverse momentum of $p_{\mathrm{T}} = 2$ TeV. The upper plots show the misidentification probability for tracks with a minimum hit truth matching of 80 %. The bottom plots, on the other hand, show the rates for a very stringent hit truth matching of 99 %. In this case, electron tracks are less likely to be falsely tagged, since bremsstrahlung effects help to determine the correct curvature sign. For less *clean* tracks this effect is shadowed by fake hit contributions to the overall momentum resolution.

NEWT, but appears less pronounced in Silicon only tracks. It leads to the assumption that it is either related to cluster error assignments in the TRT, or maybe caused by different propagation precision in simulation and reconstruction. Differences in the precision of the underlying propagation engines have been observed (in particular at larger extrapolation distances), but no conclusive evidence favoring one of the used transport engines has been found yet [12].

Pull Distributions on Measurement Level The correct description of the intrinsic measurement error contributes alongside to the accuracy of the applied material effects integration to the overall quality of the error description. A proper description of the intrinsic measurement error is less trivial than it first sounds: for Silicon detectors there is, in general, a strong dependency of the cluster sizes on the relative incident angle of the track with the detection devices; on the other hand, additional aspects such as the Lorentz-force dependent drift corrections or characteristics of the clustering and readout systems have to be understood for a correct estimation of the expected cluster errors. In NEWT, an entire layer of post-measurement calibration has been included to allow for an independent scaling of the measurement errors. This will be in particular necessary for first data taking, when misaligned detector elements will fake huge measurement uncertainties and disregarding this effect would stop any successful track fit from happening. Since the ID alignment approach is entirely built on a track-based alignment procedure, this situation would be unacceptable. A progressive approach starting

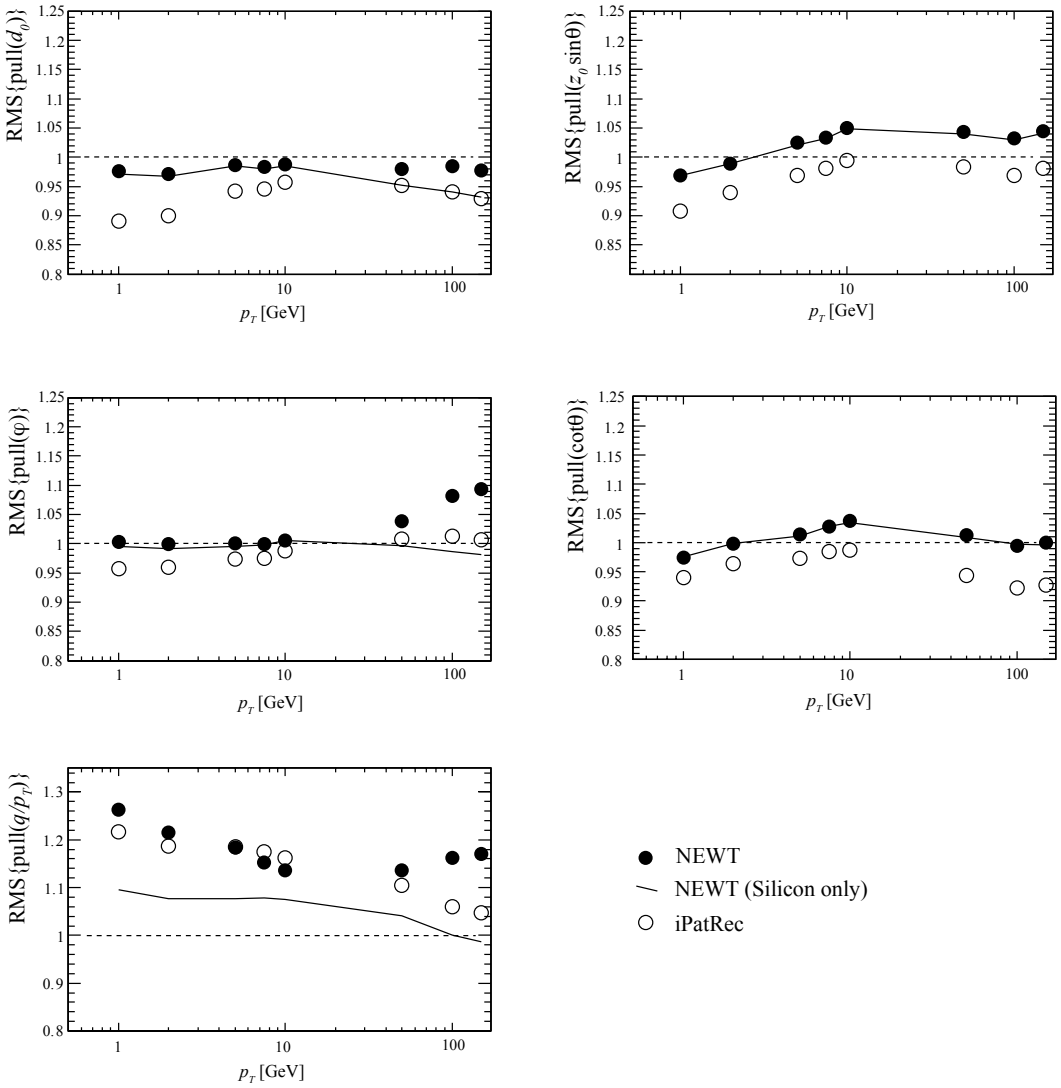


Figure 13: RMS of the pull distributions for the five track parameters at the point of closest approach to the beam axis obtained for NEWT, NEWT (Silicon only) and iPatRec. The distributions are shown for single muon tracks of different transverse momenta. A significant deviation from the anticipated value of $\approx$ 1 is observed for the momentum error estimation. This effect is way less pronounced for the Silicon only tracks, and indicates a non-optimal error handling in the TRT, or a connection to longer propagation distances.

from highly inflated measurement errors towards more realistic cluster description is therefore needed in parallel to the iteratively carried out alignment of the ID tracker. In the present tests, however, the scaling of the cluster errors has been set to 1.

Figure 14 shows the unbiased pull distributions of Silicon clusters and TRT measurements for single muon tracks at 1 GeV and 100 GeV for the overall acceptance range of the ID. The RMS of the distributions are highly compatible with 1, which indicates a good overall error description (and a high quality reconstruction material description). It can be seen that the cluster shapes and drift time distributions are in the high momentum limit deviated from Gaussian distributions, while this effect is smeared out at lower momenta, when the material effects dominate the measurement errors.

While the overall error description seems to be reasonably good, some anomalies are visible when investigating the cluster errors as a function of the pseudorapidity. A strong η-dependence of the pull distributions on hit level could be observed, which indicates a wrong error estimation in several $|\eta|$ bins. These findings have been reported to the according communities [11] and an updated model is expected to be delivered with ATLAS release 14.0.0. Figure 15 shows the $|\eta|$-dependence of the Silicon cluster

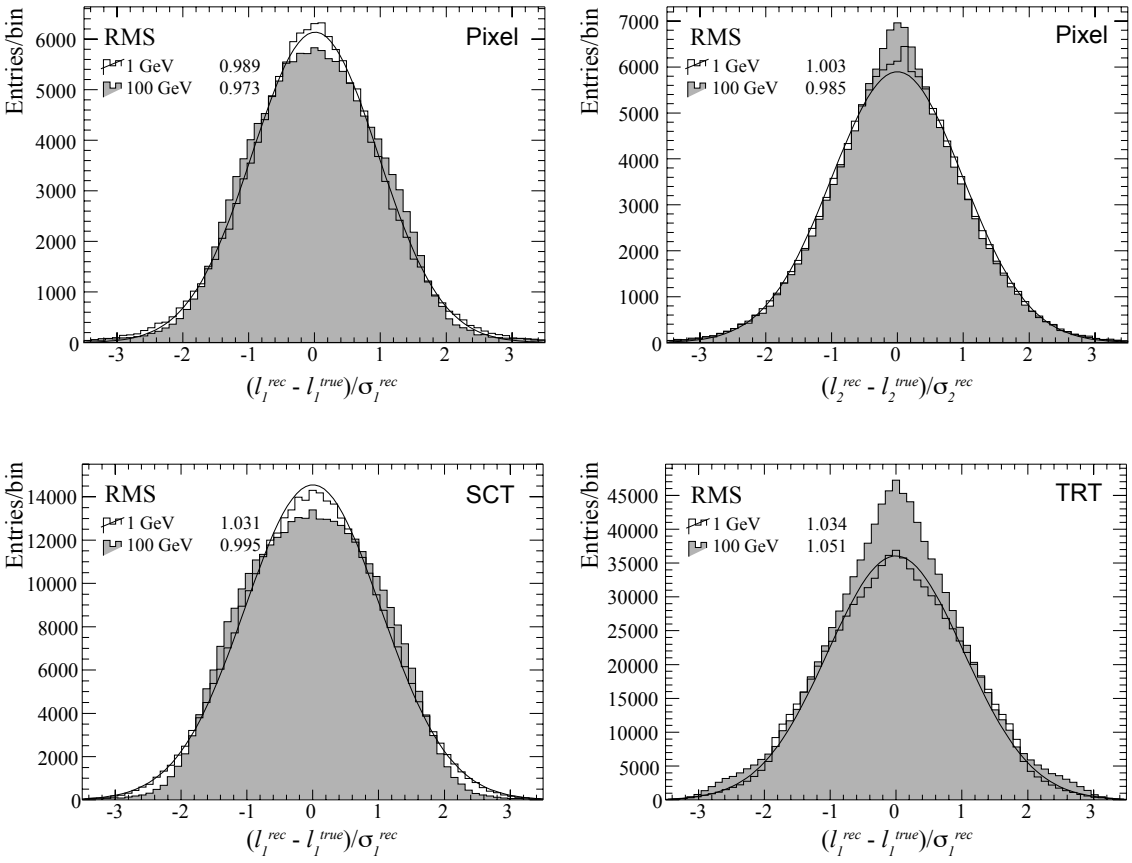


Figure 14: Unbiased pull distributions for Silicon cluster and TRT measurements for single muon tracks with low and high transverse momenta. The upper two plots show the pull distributions of the pixel detector in both local coordinates. For the SCT and TRT, only the actually measured cluster coordinates are shown. While the cluster shapes at high momenta are non-Gaussian, they tend to Gaussian distributions at lower momenta, since multiple scattering contribution gain importance.

errors for high momentum muons with $p_{\mathrm{T}} = 100$ GeV and their associated reconstruction biases.

5 Conclusion and Outlook

The performance of reconstruction software is a moving target and thus difficult to conclude. We have presented the first large scale single track validation of the new track reconstruction NEWT in the full awareness that the presented picture is not more than a snapshot in time. However, it could be shown that NEWT has become a highly performing reconstruction strategy for the ATLAS Inner Detector. It is comparable in speed and track reconstruction quality with iPatRec and benefits from the component software model and the established event data model. In particular the use of the new reconstruction geometry, that has an automated procedure to adapt itself to changing detector descriptions results in a high quality track reconstruction over the entire momentum range of final state particles that are subject of track reconstruction.

In the process of this performance validation, several problems have been revealed and many of those have already been successfully addressed, which e.g. already resulted in a strong improvement of the transverse momentum resolution and a better error description during the latest major release cycle of the ATLAS offline software. Other findings such as the non-optimal description of cluster errors in the Silicon detector or the poor momentum error description are currently investigated or have been solved in order to achieve full functionality and performance for first data taking in 2008.

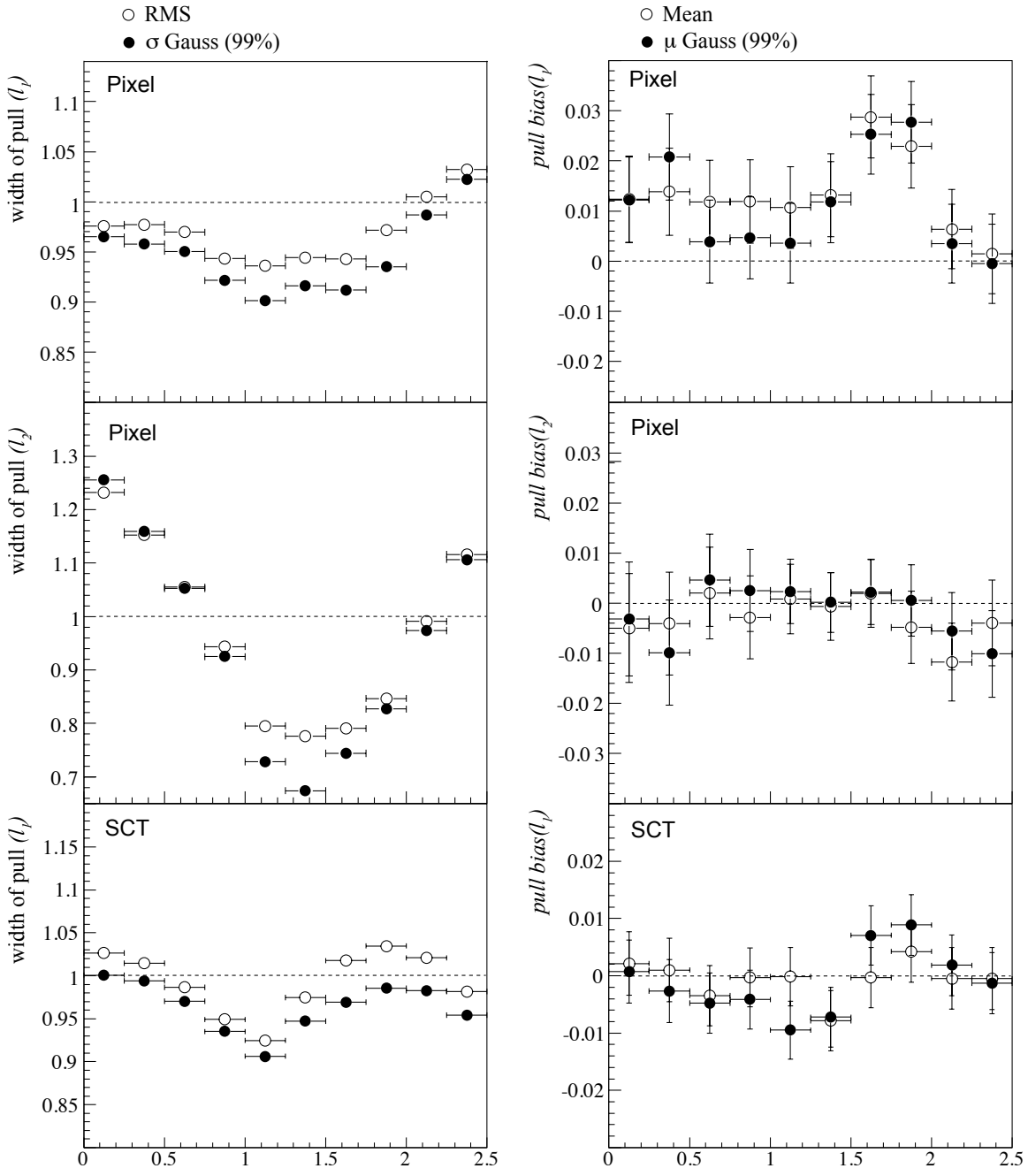


Figure 15: Silicon cluster hit pulls widths and biases for single muon tracks of high transverse momentum $p_{\mathrm{T}} = 100$ GeV. The width of the pull distributions is given as the RMS over the full distribution or the σ of a Gaussian distribution fitted a central part of 99 % of all entries. For pixel modules, pull widths and biases are given for both local coordinates l_1 and l_2, while for SCT measurements only the measured coordinate transverse to the strip is illustrated. A clear dependence of the hit pull distributions could be observed with release 13.2.0. An updated description is planned to be integrated into the next major ATLAS offline release 14.0.0.

A Appendix

A.1 Tail Comparison

In Sec. 3.2, the resolutions for the transverse impact parameter d_0 has been given as the RMS over 99.7 % of all reconstructed tracks. In particular at lower momenta, a higher resolution for NEWT could be observed, dominated by a better handling of the tail contribution. Figure 16 illustrates this fact in two ways: it shows the impact parameter resolution $\sigma(d_0)$ for different transverse momenta, when the resolution is obtained with a core fit including a $2-\sigma$ equivalent of all reconstructed tracks. NEWT and iPatRec are much more compatible within this definition. The plot to the right shows the fraction of entries above the Gaussian curve defined by the fitted $\sigma(d_0)$ for $p_T = 1$ GeV.

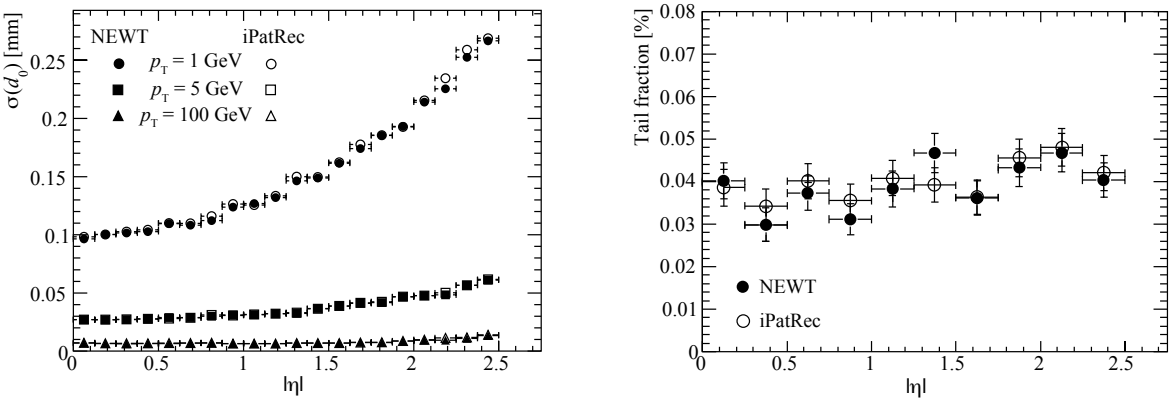


Figure 16: Left: Transverse and longitudinal impact parameter resolutions for single muons tracks with transverse momenta $p_T = 1, 5, 100$ GeV reconstructed with NEWT and iPatRec obtained from a core fit including an equivalent of a 2-σ fraction of all entries ($\approx$ 95.4 %). Right: fraction of entries above the Gaussian curve resulting from the 2-σ fit for $p_T = 1$ GeV.

A.2 Resolution Tables

The following tables, Tab. 1 and Tab. 2, show the track parameter resolutions parameterised with the $A \oplus B$ model for single muon and pion tracks, respectively. The results are given in 20 equidistant $|\eta|$ bins with a statistic of about 2000 entries per bin. The big discrepancy between the muons and pions at very high momenta is due to tail distributions that contribute strongly in the RMS calculation; these tails are mainly caused by effectively shorter tracks or misidentified secondary tracks, both caused by nuclear interactions.

References

[1] A. Salzburger (Editor) et al, *The new ATLAS Track Reconstruction (NEWT)*, ATLAS Public Note, ATL-SOFT-PUB-2007-007, 2007.

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

[3] T. Cornelissen et al., *Updates of the ATLAS Tracking Event Data Model*, ATLAS Public Note, ATL-SOFT-PUB-2007-003, 2007.

[4] A. Salzburger, S. Todorova and M. Wolter, *The ATLAS Tracking Geometry Description*, ATLAS Public Note, ATL-SOFT-PUB-2007-004, 2007.

[5] A. Salzburger, *The ATLAS Extrapolation package*, ATLAS Public Note, ATL-SOFT-PUB-2007-005, 2007.

[6] The ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, ATL-COM-PHYS-2007-102, to be published in *Journal of Instrumentation*.

[7] R. Clifft, A. Poppleton, *IPATREC: inner detector pattern-recognition and track-fitting*, ATLAS Internal Note, Soft-94-009, 1994.

[8] T. Ullrich, Z. Xu, *Treatment of Errors in Efficiency Calculations*, arXiv:physics/0701199v1.

[9] T. Cornelissen, *Track Fitting in the ATLAS Experiment*, PhD Thesis, CERN-THESIS-2006-072, 2006.

[10] Benekos, N. et al. *ATLAS Inner Detector Performance with the Rome-Initial Layout*, ATLAS Public Note, ATL-INDET-PUB-2007-008, 2007.

[11] T. Lari, *private communication.*

[12] N. v. Eldik and A. Poppleton, *private communication.*

Table 1: Parameter resolutions in 20 equidistant $|\eta|$ bins for single muons given in the $A \oplus B$ parameterisations.

		$d_0[\mu\mathrm{m}]$		$z_0 \times \sin\theta[\mu\mathrm{m}]$		ϕ		$\cot\theta$		$q/p_\mathrm{T}[\mathrm{TeV}^-1]$	
$\lvert\eta_<\rvert$	$\lvert\eta_>\rvert$	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c
0	0.125	7.57	16.5	111	1.88	7.4e-05	31.8	0.000741	4.18	0.377	40.2
0.125	0.25	7.5	16.1	102	2.01	6.84e-05	33.9	0.000714	4.36	0.332	46.4
0.25	0.375	7.61	16.1	96.9	2.17	6.93e-05	34.3	0.000732	4.46	0.339	43.4
0.375	0.5	7.74	15.7	86.4	2.35	7.17e-05	33.1	0.000703	4.62	0.341	45.2
0.5	0.625	7.4	19.6	77.1	2.67	6.99e-05	38.3	0.000668	5.38	0.346	45.9
0.625	0.75	8.45	15.3	67.7	2.86	8.28e-05	31	0.000646	5.82	0.445	40.4
0.75	0.875	8.13	17.2	58.9	3.18	8.55e-05	32	0.000641	6.25	0.466	42.6
0.875	1	7.47	20.5	51.2	4.08	7.32e-05	41.2	0.00062	7.62	0.337	68.1
1	1.125	7.36	20.6	44.6	4.19	7.31e-05	41.7	0.000603	8.25	0.319	82.2
1.125	1.25	7.51	21.2	42	4.43	7.76e-05	40.9	0.000657	8.78	0.325	79
1.25	1.385	8.07	21.8	39.8	4.95	7.92e-05	44	0.000739	9.26	0.336	76.5
1.385	1.5	8.27	22	39.8	5.06	8.58e-05	41.3	0.000848	9.15	0.376	75.1
1.5	1.625	8.51	23.7	41.4	5.18	8.6e-05	45.5	0.00101	9	0.36	88.6
1.625	1.75	9.07	22.5	41.8	4.86	9.45e-05	43.2	0.00112	9.43	0.42	81
1.75	1.885	8.72	25.3	42.6	5.14	9.99e-05	44	0.00138	9.5	0.519	73.2
1.885	2	10.1	23.4	41	5.83	0.000117	40.1	0.00169	9.38	0.705	56.1
2	2.12	11.8	22.8	39.5	6.38	0.000139	38.2	0.00194	9.45	0.921	50.5
2.12	2.25	11.9	22.7	40.5	6.88	0.000146	38.6	0.00238	9.27	1.02	47.8
2.25	2.385	14	20.7	42.2	7.03	0.000172	34.5	0.00288	8.53	1.21	44.4
2.385	2.5	15.6	20.8	47.5	6.55	0.000209	30.5	0.00381	7.2	1.56	34.6

Table 2: Parameter resolutions in 20 equidistant $|\eta|$ bins for single pions given in the $A \oplus B$ parameterisations.

		$d_0[\mu\mathrm{m}]$		$z_0 \times \sin\theta[\mu\mathrm{m}]$		ϕ		$\cot\theta$		$q/p_\mathrm{T}[\mathrm{TeV}^-1]$	
$\lvert\eta_<\rvert$	$\lvert\eta_>\rvert$	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c	σ_∞	p_T^c
0	0.125	10.6	11.8	127	1.89	0.000108	22.6	0.000975	3.73	0.518	31.1
0.125	0.25	11	12.9	125	1.87	0.000107	23.8	0.00102	3.4	0.527	29.5
0.25	0.375	9.25	14.7	115	1.93	9.59e-05	26.3	0.000951	3.52	0.535	30
0.375	0.5	11.9	12.5	118	1.91	0.000114	24.5	0.000998	4.22	0.596	27.3
0.5	0.625	10.8	14.7	113	2.17	0.000102	29.5	0.000916	4.89	0.564	32.4
0.625	0.75	12.2	13.1	109	2.17	0.000117	25.4	0.00109	4.98	0.668	27.9
0.75	0.875	11.8	13.5	107	2.27	0.000117	26.7	0.000967	5.33	0.606	34.8
0.875	1	12.5	14	88.2	3.2	0.000121	27.3	0.000902	6.55	0.659	38.4
1	1.125	12.2	15.2	74.2	3.66	0.000118	29.9	0.000961	6.25	0.634	42.8
1.125	1.25	12	16.4	92.2	3.16	0.000116	32.4	0.00108	6.28	0.613	48.7
1.25	1.385	14.3	14.2	92.7	2.87	0.00013	30.2	0.00113	6.44	0.676	46.1
1.385	1.5	17.4	12	106	2.44	0.000143	29.7	0.00133	6.62	0.724	46.5
1.5	1.625	16.2	13.5	102	2.43	0.000139	31.3	0.00139	6.98	0.706	50.9
1.625	1.75	16.6	13.8	120	2.22	0.000144	31.5	0.00166	6.6	0.745	50.8
1.75	1.885	15.5	16.2	137	2.12	0.000155	34	0.00177	7.84	0.86	50.3
1.885	2	18.9	14.2	132	2.14	0.000163	33	0.00206	7.81	0.936	45.7
2	2.125	19.7	13.6	114	2.98	0.000177	30.4	0.00232	7.78	1.12	43.6
2.125	2.25	20.2	14.9	136	2.84	0.000179	34.3	0.0027	7.68	1.13	46.3
2.25	2.385	20.8	14.9	147	2.44	0.000189	34.3	0.0033	7.17	1.24	44.7
2.385	2.5	21	15.9	142	2.77	0.000228	30	0.00387	7.04	1.58	35.8

fatras (French):
m clutter; junk;
(choses sans valeurs, inutiles)
un ~ d'idées: a load of muddled ideas

dictionary.cambridge.org

Chapter 9

Fast Track Simulation

9.1 Introduction

Monte Carlo simulation of the expected detector response is one main technique in high energy physics. It is extensively used for background and cross section estimations or model comparisons. The very detailed simulation of the detector response is — not very surprisingly — a highly CPU time expensive operation and the required event samples needed to minimise the statistical uncertainties can not be produced in such a way. Different fast simulation approaches are needed to reach the required simulated event numbers.

In addition, the deployed component software model, as described in Sec. 8.1, immediately brought a rapid increase of algorithmic modules in the ATLAS New Tracking, which required a consistent validation framework. The most prominent example is probably the development of several different fitting algorithms, each of which deriving of a common base and interchangeable just by job configuration. The validation of these new modules has usually taken place in several steps: a purely mathematical validation of self-consistency, large scale tests on Monte Carlo simulated data and in the end also the usage of the reconstruction chain in test beam and commissioning runs with the ATLAS detector. The first aspect requires a completely controlled input environment — somehow fortunately — led to the development of a new fast track simulation for the ATLAS experiment.

In Chap. 5, the new ATLAS reconstruction geometry has been presented to very detail, putting special emphasis on the inert navigation schema between the main components of the geometry model. Together with the track extrapolation engine, described in Chap. 7, a predictive navigation has been established that allows to predict the trajectory through the detector and identify the sensitive tracking devices that are intersected along the path. This feature has been not only been integrated in the hole search algorithm, which is part of the ambiguity solving process in the new offline track reconstruction (see Sec. 8.2), but is also the core of a new fast track simulation application, that has been recently established in ATLAS. Since the trajectory of the particle (including intersections with sensitive detector material) could be gathered through the track extrapolation engine, only few parts have been missing to turn this into a new fast track simulation:

- a decent **clusterisation** model for the simulation of clusters and drift radii;
- Monte Carlo based material effects integration that include multiple scattering, energy loss through both ionisation and radiative losses;

- a dedicated module for the creation of photon conversions;
- accomplished by several additional components for the algorithmic steering and truth handling.

This new *Fast ATLAS Track Simulation* (FATRAS) has become a powerful alternative to the two existing simulation strategies, the full detector simulation and the fast simulation ATLFAST, which is based on parameter smearing. FATRAS has been extensively used in the validation of many components deployed in the new track reconstruction, but also for high occupancy studies and first simulation tests for a future update of the ATLAS Inner Detector.

There are several advantages of the FATRAS approach when comparing it to the widely used smearing approach that is implemented in ATLFAST: the Monte Carlo simulation of the underlying physics processes guarantee a consistent track creation that can also be picked up for a successive calorimeter simulation. In addition, FATRAS can be restricted to a handful of controllable tuning parameters, rather than complex smearing functions that need constant updates that account for changes in the detector description or conditions data. FATRAS follows such changes directly through the usage of the reconstruction geometry and the clusterisation model. Finally, replacing the full simulation with a fast simulation engine that produces input data for the standard reconstruction algorithms prevents the user to write different client code for both approaches.

9.2 The Fast ATLAST Track Simulation (FATRAS)

ATL-SOFT-PUB-2008-001, 49 p.

Acknowledgements

The FATRAS project was started as an initiative of the author and established at first means a validation framework for newly established track reconstruction algorithms. The increasing functionality and usage of FATRAS, and in particular the extension with more complex physics processes involved several more contributors. The author would like to thank the involved collaborators, not only for their enthusiasm and great contributions to this project, but also for a good deal of friendship.

Sebastian Fleischmann should be granted credits for his effort in sorting and organising the EDM collections, for his great contribution to the hit post processing, the inclusion of the offline truth model, and the reconstruction integration. Tatjana Lenz contributed strongly to the simplified decay model and even more to the inclusion of a correct bremstrahlung and conversion simulation. Carsten Magass deployed the parameteric model for hadronic interactions with the detector material and Keith Richards supplied willingly constant validation plots. Finally, Jörg Mechnich included the generic geometry model for the SLHC extension, and imported the particle decay modules from Geant4.

Erratum

The given reference [11] in the presented note has been falsely assigned as ATL-SOFT-PUB-2007-002, while the correct number is ATL-SOFT-PUB-2007-007.

The Fast ATLAS Track Simulation (FATRAS)

K. Edmonds, S. Fleischmann, C. Magass
Universität Bonn, Germany
T. Lenz
Bergische Universität Wuppertal, Germany
J. Mechnich
Universität Freiburg, Germany
A. Salzburger*
Leopold Franzens Universität Innsbruck, Austria & CERN

March 5, 2008

ATL-SOFT-PUB-2008-001
11 March 2008

Abstract

Various systematic physics and detector performance studies with the ATLAS detector require very large simulated event samples. Since the full detector simulation is a highly CPU time consuming operation, fast simulation techniques are widely used in such applications. Furthermore, the simulation of background events does, in general, not require the very detailed detector simulation and fast simulation techniques satisfy the needed accuracy.

In ATLAS, the fast simulation program ATLFAST has been extensively used for such purposes. It is, however, based on the smearing of the initial particle properties and is not capable of producing hits along the track. Tracking relevant studies that include both hit information and pattern recognition effects can not be performed when using ATLFAST. An alternative simulation program, the new *Fast ATLAS Track Simulation* (FATRAS) has been recently deployed, capable of producing full track information, including hits on track. Initially developed as a validation tool for the ATLAS offline track reconstruction, it has become a powerful engine for various use cases. In general, the CPU time determining factor of the full simulation is the tracking of the particle through the very complex detector geometry, while the event reconstruction including pattern recognition and track fitting is relatively fast. In FATRAS, the simplified reconstruction geometry is used as a simulation geometry model, which leads to a significant speed up of the simulation process. FATRAS uses furthermore mainly common offline track reconstruction code and the reconstruction event data model. It is fully embedded in the ATLAS C++ based software framework ATHENA.

ATLAS NOTE
The ATLAS Experiment, *http://www.atlas.ch*

*Corresponding author: Andreas.Salzburger@cern.ch

1 Introduction

The Monte Carlo simulation of physics events together with the according detector response is an essential technique in high energy physics. In the preparation phase of an experiment this becomes the only source to predict the sensitivity of the detector setup with respect to various event channels, while during data taking it builds the framework to test theoretical models against the real detector response. Furthermore, since most of the readout and reconstruction software is developed in parallel to the detector installation and deployment, simulated data is — besides taken data from test beam setups and commissioning runs using cosmic rays — the only input available for testing and validating the performance of the reconstruction software.

The event simulation process can be divided into two sequential parts: the primary physics event generation is usually carried out by common high energy physics libraries such as PYTHIA [1] or HERWIG [2], while the second step, the simulation of the detector response, is obviously particular to the experimental setup. Latter includes the particularities of the detector geometry and the integrated detection technologies. These components are in the most sophisticated detector simulation — in the following referred to as *full simulation* — realised through a very detailed geometry model and an accurate description of the particle interaction with the sensitive detector material, followed by a realistic clusterisation model. In fast simulation techniques based on parameter smearing, on the other hand, both components are respected intrinsically through the smearing functions that are obtained from full simulation results. These two track simulation techniques have been existing and extensively used in ATLAS: the detailed full detector simulation that is based on the well known Geant4 simulation toolkit [3], and a fast track simulation (as a part of the ATLFAST [4] program) that works on the basis of four momentum vector smearing. In the full detector simulation particles are tracked through a very complex geometry model, and interactions with the sensitive and non-sensitive detector material are simulated. Hits are generated in the sensitive parts and further processed in a digitisation[1] module that prepares the simulated hit information for the reconstruction algorithms. 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 falls underneath a certain energy threshold. The reconstruction software, a so-called *online* application for the event triggering and the *offline* part for the final event reconstruction and analysis, is then executed subsequently to the detector simulation and digitisation. It yields the final track resolutions and reconstruction efficiencies. For convenience, the compound of full detector simulation, digitisation and offline reconstruction will be in the following also referred to as *offline chain*.

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 represent the track as it is expected to be reconstructed by the offline track reconstruction; this is only valid in a purely statistical manner. The smearing functions are for this purpose 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 has also to accumulate all aspects of the entire simulation and reconstruction chain (including the detector layout, the material budget, the digitisation and clustering, as well as the tracking performance); they have to be, in principle, updated if any of the involved components changes substantially. Many studies have been in the past performed using the ATLFAST simulation, however, in particular for tracking performance studies it is not suitable since no hit information is available.

Recently, a new *Fast ATLAS Track Simulation* (FATRAS) has been established that realises a full Monte Carlo simulation approach, but makes use of the simplified reconstruction geometry model rather than the simulation geometry. FATRAS is based on common reconstruction tools and uses the common tracking event data model (EDM) [5] natively[2]. FATRAS is capable of tracking the particle through the entire reconstruction geometry (in the following referred to as `TrackingGeometry` [6]), using the ATLAS extrapolation engine [7] for the transport of the track parameters and the inert

[1] 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.

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

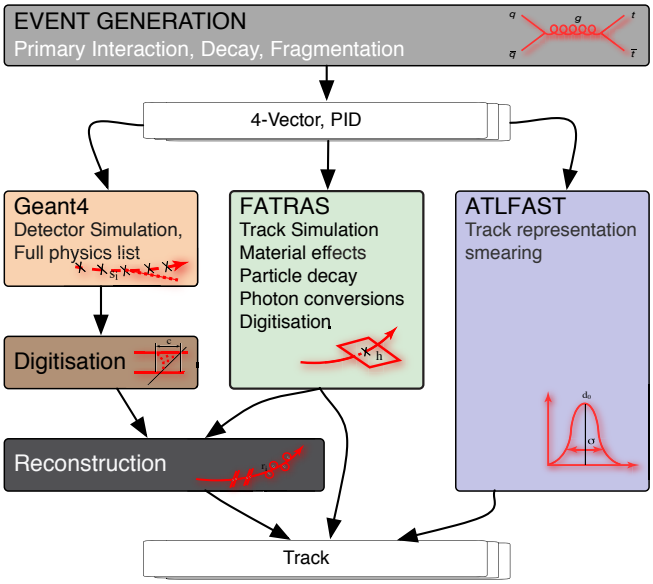


Figure 1: The track simulation techniques used in ATLAS: the event generation provides 4-vector momentum representations together with a particle identification, which builds the input to the different simulation strategies. The very precise Geant4 simulation is based on tracking the particle through the detector and simulating its interaction with the detector material. The produced hits are then fed into a digitisation module for further processing in the event reconstruction. FATRAS follows a similar approach, while using the simplified reconstruction geometry and parameterised models for the interaction of the particle with the detector material. A fast digitisation can be applied, such that the hits can either serve as input to the standard reconstruction or directly be used through the output track object. The ATLFAST simulation is based on the smearing of the generated input, aimed at representing the particle in the stage after the track reconstruction.

navigation schema of the `TrackingGeometry` 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 development of the track reconstruction components [8], it became a powerful tool for broader purpose including a (limited) usage in the simulation of physics events. FATRAS is able to produce hits along a track and enhances track fitting, vertex reconstruction and even the input creation for the standard offline reconstruction chain. It allows large scale tracking, vertex finding and heavy quark tagging studies while guaranteeing low execution times. Quark tagging, however, requires also the inclusion of calorimetric measurements for the jet building process. FATRAS is aimed to be executed in a final configuration together with a dedicated fast shower simulation; a very brief outlook and discussion of the current status of such a combined simulation can be found in Sec. 6.1. Figure 1 presents an overview on the track simulation techniques used in ATLAS, and Fig. 2 shows the same $t\bar{t}$ production event simulated with the full simulation and FATRAS. The tracks shown are those found by the identical offline track reconstruction; the visualisation is done with the ATLAS event display ATLANTIS [9]. FATRAS is currently limited to the ATLAS *Inner Detector* (ID), mainly because the `TrackingGeometry` description of the second tracking device, the *Muon Spectrometer*, is still in a prototype version. A future extension of FATRAS to include also the MS is one challenging part for the further development of the FATRAS project.

This document is based on ATLAS software release 13.2.0 and several attached packages, that are installed on a common group area; since this release marks the first production release that contains the close-to-final FATRAS setup, not every module of FATRAS has been tuned and calibrated to full extent. This work is expected to be integrated for the next major production release 14.0.0. A brief discussion of missing components and planed tuning and calibration work is given in Sec. 6. For the convenience of the reader, an overview of the used packages is given in the appendix A.6.

1.1 Document Structure

This document is structured as follows: Section 2 will give a high level overview of the concepts and modules that build FATRAS, concentrating on the integration of the new fast track simulation into the software framework of the ATLAS experiment and in particular the newly developed track reconstruction chain. Section 3 describes in the following the single modules of FATRAS in more detail, while already giving performance comparisons with the full simulation on the level of material

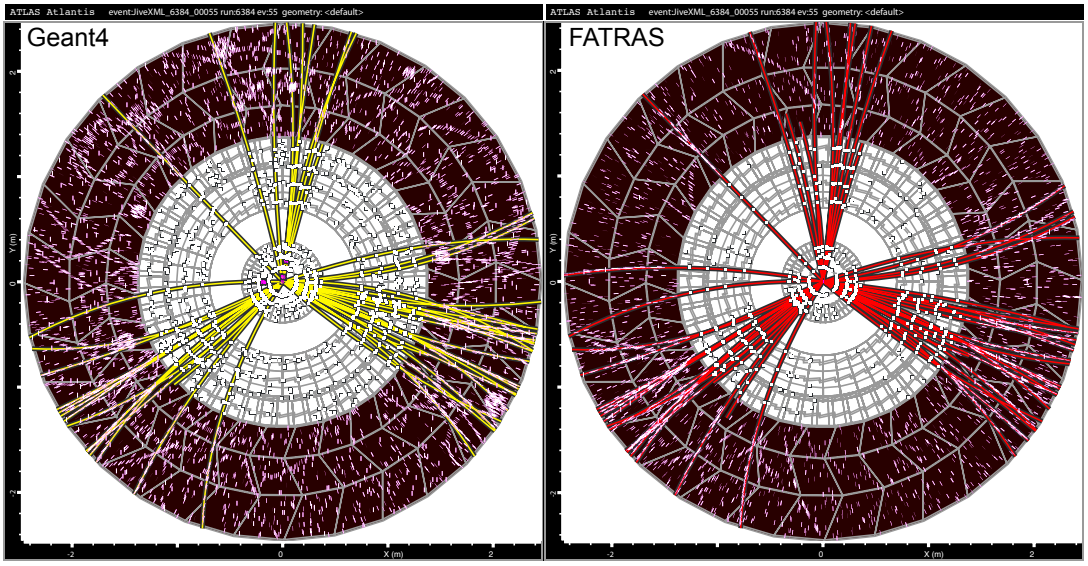


Figure 2: An identical hard proton-proton scattering event leading to the production of a $t\bar{t}$ pair simulated with the full detector simulation and the FATRAS fast simulation in the ATLAS Inner Detector. In both cases, the standard ATLAS offline track reconstruction is performed, the found tracks are also illustrated.

effects integration and hit creation. Section 4 focusses on the comparison of FATRAS with the full and ATLFAST simulation mainly on single track level and includes a comparison of CPU time for the different simulation strategies, while Sec. 5 shows the usage of the new fast track simulation in some dedicated applications. Section 6 will conclude this document, but will also give an outlook on planned future expansions or modifications to the FATRAS project. The Appendix covers conventions and typesetting used within this document and gives a small introduction to the FATRAS usage. It also summarises the involved software packages and available tuning parameters for the convenience of the reader.

2 Concepts and Modules of FATRAS

FATRAS is a full Monte Carlo based track simulation that makes use of the reconstruction geometry and tools from the offline track reconstruction. It re-uses to a large extent modules and resources of the offline track reconstruction, while only few dedicated components replace standard offline algorithms and tools. The main benefits of the chosen approach — besides the pure performance aspects — can be summarised as follows:

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

The following section will give a brief overview of the high level modules used in FATRAS, while a detailed description of the single components can be found in Sec. 3.

2.1 The ATHENA Framework and the New Tracking Approach

The new fast track simulation is fully embedded in the ATLAS C++ software framework ATHENA [10]. ATHENA is realised as a data centered software framework that follows a strict component

pattern design and provides interfaces for modules at different levels of the program flow. The use of common interfaces and modules is hereby essential for the FATRAS design, since it allows the exchange of single offline reconstruction tools with modified Monte Carlo based versions that comply with the same interface definition. Many modules that are used in the FATRAS application are directly taken from the new ATLAS track reconstruction, the so-called *New Tracking* (NEWT) [11]. In particular, the tracking event data model (EDM), the reconstruction geometry description (which is used as the simulation geometry in FATRAS), and the extrapolation engine are essential parts of the FATRAS simulation. A description of these modules would go far beyond the scope of this document; the reader is, however, encouraged to find detailed documentation in [5], [6] and [7], respectively.

2.2 Module Sequence and Data Flow

The default FATRAS simulation sequence consists of six different modules, each of which realised as an ATHENA `Algorithm` class. Figure 3 illustrates the `Algorithm` sequence executed in FATRAS by a simplified UML activity diagram.

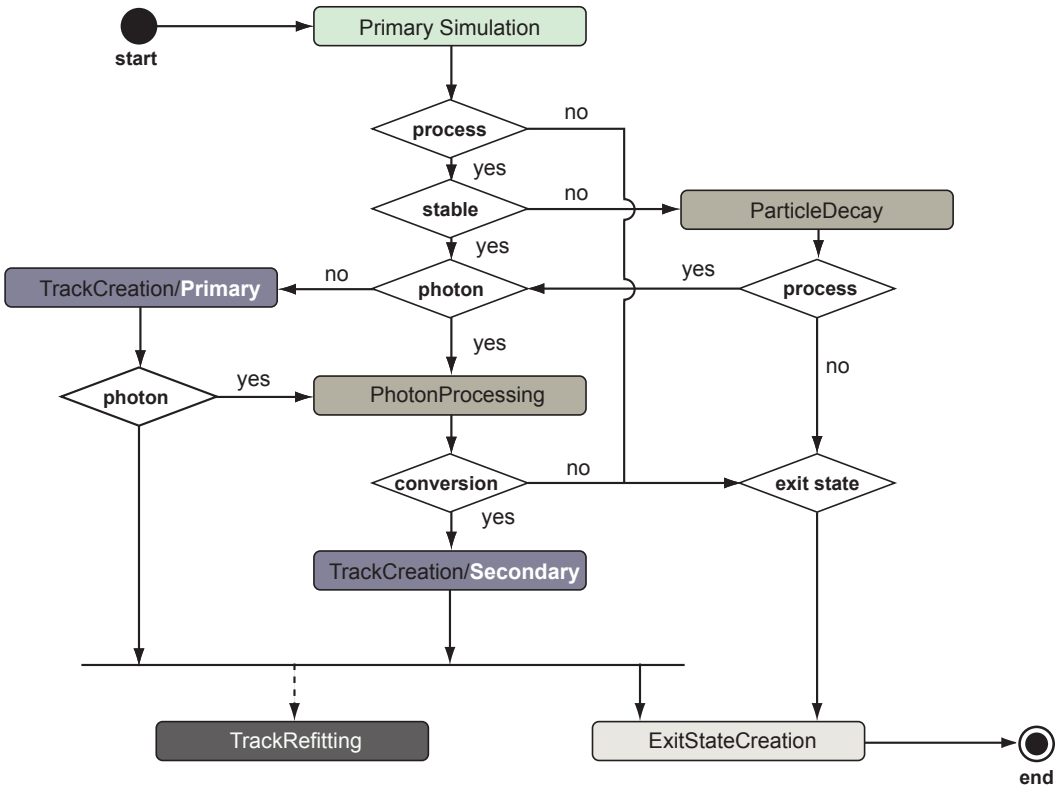


Figure 3: Simplified UML Activity diagram showing the six different modules that build the FATRAS simulation. This illustrated configuration includes only one iteration of photon processing, while this has been recently expanded with a second photon processing, particle decay and a third track creation module to further track lower energetic particles that are created in decay or conversion cascades. The component software model that has been followed throughout the FATRAS design eases this modified configuration by simple adaptions of the job configuration.

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 and in particular the usage 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, such as e.g. the insertion of an additional iteration on conversion or decay products. Latter is necessary to correctly handle secondary particles that are induced by hadronic shower interactions.

The following list presents a brief overview of the different FATRAS modules; a detailed description

can be found in Sec. 3.1 to Sec. 3.7 of this document:

- **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 without the — in this respect — unnecessary overhead of event generation through a dedicated generator module. 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. The input collection is hereby sorted by *prompt* tracks, decaying particles, photons and non-interacting particles that are immediately scheduled to be transported to the tracker exit.

- **Particle Decay:** particles that are not flagged as stable by the generator are in FATRAS filtered into a dedicated container by the primary simulation `Algorithm`. The existence of a non-empty container of decaying particles triggers the execution of the `ParticleDecay` module. The stable decay products are added to the collection of particles that are scheduled 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 most actions are outsourced to `AlgTool` entities, while the `Algorithm` class usually just delegates; the actual decay is performed by a dedicated `ParticleDecayer`, that exists in two flavors: a simplified version that only supports a limited number of decay channels, and a sophisticated wrapper of the particle decay modules that part of the Geant4 simulation.

- **Track Creation:** the track creation `Algorithm` is the core of FATRAS; the track creation can occur several times in the FATRAS simulation sequence to allow for an iterative treatment of secondary particles induced by hadronic shower reactions or photon conversions.

- **Photon Processing:** photons from the Monte Carlo generator (i.e. final state radiation) as well as hard photons originating from the transport of electrons through the detector are further handled by the dedicated `PhotonProcessing Algorithm`. They are extrapolated through the detector while — depending on the traversed material — the conversion probability is calculated and pair production is performed. Tracks originating from photon conversions are created by the secondary track creation and may again lead to hard photon emission. The photons are then, depending on the chosen configuration, either integrated into yet another iteration of the electron-bremsstrahlung cascade, or directly scheduled for a transport into the calorimeter.

- **Track Refitting:** the refit of the simulated track is essential for FATRAS to gain comparable track resolutions with tracks found in the standard offline reconstruction — when FATRAS is performed in the *refit* mode. The initial track parameters used for simulating the track are hereby smeared before refitting to remove the bias that is given by seeding the track fit with the true initial track parameters. Omitting this smearing step would lead to artificially narrow cluster residuals on the first detection layer[3]. Any track fitter that implements the `ITrackFitter` interface from the New Tracking realm can be used to perform the track fit. Tracks that originate from photon conversions (and are thus produced in the second track creation step) do not necessarily have to be refitted. The main focus is hereby drawn on the appropriate description of these particles at the calorimeter entrance for the shower simulation in the electromagnetic calorimeter. Hits that have been created by particles that originate from photon conversions are indeed added to the event hit collection, such that they are correctly included when FATRAS is used to feed the offline pattern recognition.

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

[3]The smearing has to be kept in the linear regime of the underlying track model and is very similar to finding an appropriate track seed for the fit in the the standard offline reconstruction.

- **Post Processing:** the hit post-processing module is somehow independent from the main algorithmic sequence in FATRAS, but necessary to prepare 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.

FATRAS incorporates the data centered blackboard architecture design of the ATHENA framework: the different `Algorithm` classes do not have any relation other than that they all read and write to the same transient event store (TES). In many cases the existence of a collection in TES indicates whether an algorithm has to run or not.

2.3 The FATRAS Event Data Model

The FATRAS event data model is to a large extent identical with the tracking EDM of the ATLAS offline reconstruction. Non-calibrated and calibrated measurements are expressed through the according offline EDM classes; the use of the extrapolation engine for the trajectory creation and the common track fitting tools ensures that the created track objects are identical 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 [12] can directly work on FATRAS output. Only the top object collections in FATRAS are integrated in a special EDM schema, based on a generic `ParticleState` base class. This is necessary for the intrinsic truth steering in the simulation process.

EDM Extensions The polymorphic structure of the ATLAS tracking EDM allows to create generic cluster objects and use them together with the FATRAS simulation. This is in particular interesting when using FATRAS in design studies for future upgrade scenarios of the ATLAS tracker, see Sec. 5.2. A generic silicon cluster class that inherits from the common ATLAS `PrepRawData` base class and the associated calibrated version that fulfills 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 enhances 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[4]. The full extension of FATRAS to satisfy the needs of tracker upgrade studies requires also an updated detector model. This is facilitated by the design of the `TrackingGeometry` that provides generic geometry classes that are independent from any given detector technology. In FATRAS, a flexible generic geometry builder has been deployed that enhances the creation of different geometrical configurations and associated clusterisation models.

Truth Association The association of the FATRAS tracks (both, from the refit mode and the reconstruction mode) with according truth objects from the event generation and hit simulation is necessary for many validation studies. The ATLAS tracking EDM, which builds the base EDM for FATRAS, does not allow direct links between reconstruction objects and generated truth information. Thus, associative relationships have to be established to bind the simulated track to the truth information. In FATRAS — since the truth information is *a priori* known — this is enhanced by the `ParticleState` class which is used for the main FATRAS containers that are written to the transient event store. `TruthAssociation` objects can be registered to the `ParticleState` and thus a direct relation between the simulated EDM objects and the generated particle is established. When refitting tracks in FATRAS, an associative container (realised as an STL map) is created and recorded in the transient event store to keep the truth information available on this level.

There exist several dedicated truth association `Algorithm` classes that have been omitted in Sec. 2.2, where the general `Algorithm` sequence of FATRAS has been presented. This is, because these components are not part of the FATRAS simulation itself, but serve pure validation purpose. A more detailed description of these modules and their functionality can be found in Sec. 4 of this document.

[4]It is worth mentioning that this is a striking argument for the component software pattern and the polymorphic data model design that has been realised through the new track reconstruction software.

Recently, the full Monte Carlo truth tree that uses the standard `HepMC` [13] class descriptions has been deployed and concluded the effort to turn FATRAS into a look-alike pendent to the full simulation chain. The full `HepMC` truth tree has not only been necessary for the correct integration of FATRAS with a successive calorimeter simulation, but also enhanced standard validation algorithms that are widely used on offline data to cope with FATRAS simulated events.

2.4 Modes and Reconstruction Feeding

One major achievement of the FATRAS simulation is that it can serve as an input for the offline track reconstruction. This is possible, since FATRAS uses the offline tracking EDM and produces `PrepRawData` objects in the track creation process, which mark the input objects of the offline pattern recognition. A dedicated `PostProcessing` module exists that strips the created `PrepRawData` objects from the track and fills them into the standard hit container that build the input to the `SpacePointFormation`, the first module in the ATLAS New Tracking. Hereby, the noise level can be adjusted freely, which turns FATRAS into a useful application for occupancy studies and pattern recognition validation with different noise levels, see Sec. 5.1. While for a fast parameterisation of the Inner Detector targeted at providing fast but accurate input for a more detailed calorimeter shower simulation the pure track simulation is sufficient, any tracking based study has to at least use the refitted FATRAS tracks and — if pattern recognition effects are of interest for the according analysis — finally the reconstructed tracks. These different modes will be in the following referred to as *simulation*, *refit* and *reconstruction mode*, respectively[5].

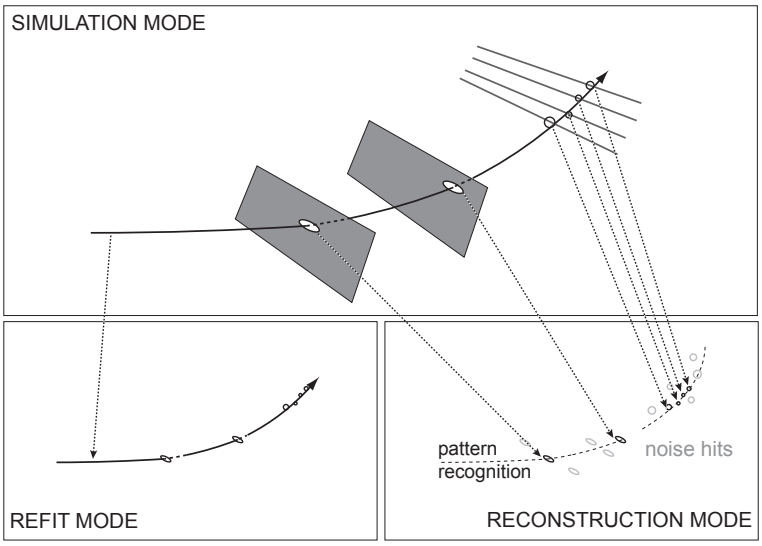


Figure 4: Simplified illustration of the three different FATRAS modes: the *simulation* mode creates only the truth tracks that are then further processed in the *refit* and *reconstruction* mode, respectively. For the *refit* mode, the simulated track is as a whole passed to a track fitting module, 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; latter includes pattern recognition and track fitting.

3 The FATRAS Simulation Modules

In Sec. 2 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. The following sections — Sec. 3.1 to Sec. 3.7 — will describe the single modules in more detail and will bring each of them in context with the full detector simulation.

3.1 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:

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

- The `SingleTrackSimulation` has been mainly designed for the validation of both FATRAS and the offline reconstruction software. It offers the possibility to simulate user defined single track events with different particle types and kinematic input variables. In general, random based input parameters are generated, but the `SingleTrackSimulation` can also be executed in a scan mode and a fully configured one-event basis for debugging single tracks that show unexpected or faulty behavior in the offline reconstruction or the track fitting modules.
- The `GenEventSimulation Algorithm` interfaces the output from any given Monte Carlo generator with the fast track simulation and thus enhances 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. Table 1 summarises the distinguished particle types and their further processing module in FATRAS.

Table 1: Particle processing in FATRAS; currently only a limited number of particle decays are supported that are mainly related to tracking relevant studies. All particles are finally processed in the `ExitStateCreation` that prepares the Monte Carlo input data for a final representation at the Inner Detector exit surface for further processing in a successive calorimeter simulation.

Particle (e.g.)	**Comments**	**FATRAS Container**	**Processing**
$\mu^{\pm}$,e$^{\pm}$,p$^{\pm}$	stable final state	TrackStates	track creation
γ	final state, or bremsstrahlung	PhotonStates	photon processing
$\pi^0, \pi^{\pm}, K_{L,S}, K^0$	non-stable	DecayingStates	particle decay
n	stable, non-interaction	ExitingStates	exit state creation

3.2 Particle Decay

In general, most of the relevant particles decays which are contributing to the final state signature of an ATLAS event happen inside the beam pipe volume and are determined by the event generator. However, a non-negligible amount of mesons will produce a decay vertex inside the ID and can therefore be a source for lepton tracks and jet fakes. 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 electromagnetic calorimeter.

Taking account of this, FATRAS provides two algorithms with dedicated `AlgTool`s to take care of particle decays after primary simulation (only at the time of writing). 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 $\xi \in [0, 1)$, such that — when using a simple transformation method —

$$\lambda = c \cdot (\beta\gamma) \cdot (-\log \xi), \tag{1}$$

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 retrieve 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 handed over to the exit state creation.

Further processing of the remaining particles is performed by 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. All tracks are created by an `AlgTool` implementing the `ITrackCreator` interface. It is used to extrapolate the trajectory to the decay vertex and to create a `Trk::Track` from simulated detector measurements.

3.2.1 ParticleDecay

The `ParticleDecay Algorithm` is to be found inside the `FatrasAlgs` package. Only a limited number of particles and decay channels are supported, focussing on situations that are important for tracking studies (i.e. π_0, $K^{\pm}$ and $K^0_{S/L}$). The `ParticleDecayCreator AlgTool` residing inside the `FatrasTools` package is handling the decays of the already mentioned particles (see Sec. A.2 for details).

3.2.2 G4ParticleDecay

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

3.3 Track Creation

The track creation in FATRAS in done in two steps: the first one marks the trajectory building and is carried out by the extrapolation engine together with the underlying reconstruction geometry. The ATLAS `TrackingGeometry` is characterised by a fully connective navigation model using neighboring volumes that are attached at shared surfaces. The confining surfaces of the `TrackingVolume` class (the main components of the `TrackingGeometry`) extend hereby the common surface description that is the foundation of the ATLAS event data model and can therefore be natively used with the extrapolation engine. In such a way, the trajectory can be followed through the detector, since every boundary surface — when being intersected — leads directly to the next detector volume traversed by the particle. 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 with a layer to the associated detector element and consequently, the trajectory of hits can be built by intersecting one layer after the other — always guided from one volume to the next volume by the internal navigation tree. Material effects, such as ionisation loss, radiation loss or multiple coulomb scattering are applied during the trajectory building. This is enhanced by simply exchanging the stochastic material effects treatment as used in the track reconstruction with Monte Carlo based modules[6]. A detailed description of the material effects integration can be found in the following section, Sec. 3.3.1.

The second part of the track creation is the conversion of the given trajectory into a meaningful track object. This involves cluster creation on the one hand and applying efficiency tuning on the other hand. The FATRAS clusterisation model is further described in Sec. 3.3.2.

3.3.1 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 on several complexity levels which quickly lead to a high calculation complexity[7]. As a general rule, a cumulative simulation is always more accurate than a fully parametric one, while the single step intervals in which the simulation is performed regulate both, the accuracy and the the CPU time. For a fast track simulation it is thus of particular interest to find a good compromise between a reasonable 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 applied corrections to the trajectory are done as a single action. In the reconstruction geometry all of these detector components are described as layer objects with according material descriptions, this update mechanism is thus often called *layer-based* or *point-like*.

[6] The exchange of single modules in the program flow is facilitated by the component software model of the new offline track reconstruction.

[7] 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 5 shows a distribution of typical path lengths expressed in terms of radiation length X_0 for the ATLAS Inner Detector and shows a comparison of the overall material distribution for the simulation geometry based on Geant4 and the `TrackingGeometry` that is used in FATRAS.

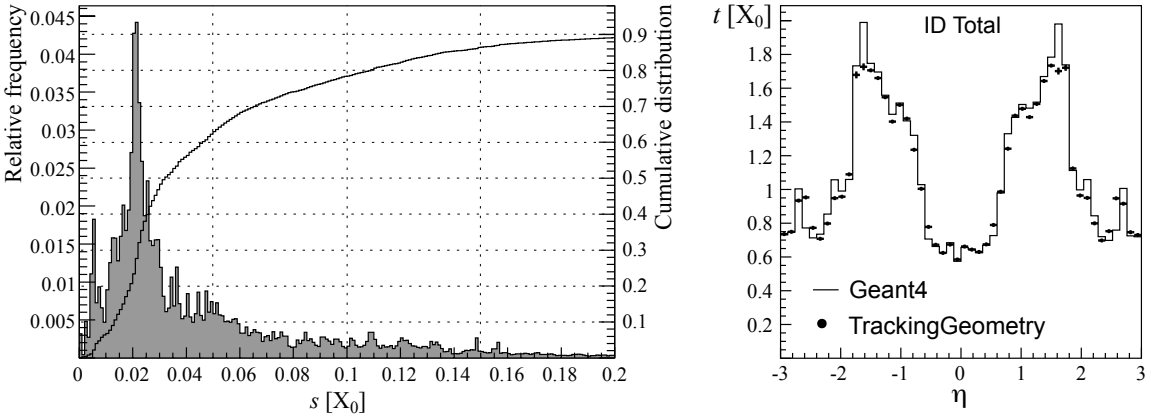


Figure 5: Left: Relative frequency and cumulative distribution of the single step lengths when traversing the Inner Detector reconstruction geometry. About 80 % of all steps through the detector material are described by less than 10 % of radiation length — a value that reaches almost 100 % for the barrel section. Steps that account for more than 10 % of the radiation length are mostly due a large incident angle of the particle in respect to the crossed layer or due to support structures in the Inner Detector. Right: Comparison of the material budget described by the Geant4 simulation geometry to the `TrackingGeometry` description in terms of total path length in units of radiation length.

Simulation of Multiple Scattering When a particle traverses detector material it undergoes multiple small angle deflections caused by the Coulomb force of the nuclei of the detector material. Following the central limit theorem, the distribution of the sum of these small deflections — the multiple scattering process — can be approximated with 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 the two methods provided by the `MultipleScatteringUpdator` that is part of the extrapolation package: the first possibility is a purely Gaussian approximation expressed by the Highland [14] formula

$$\sigma_{ms}^{proj} = \frac{13.6\text{MeV}}{\beta cp}\sqrt{t/\text{X}_0}\,[1 + 0.038\ln(t/\text{X}_0)], \tag{2}$$

where t denotes the traversed path length and X_0 the radiation length of the material. For many applications — and in particular for the usage of a fast track simulation to validate the reconstruction software — it is of particular interest to simulate tail effects. FATRAS offers therefore 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 as proposed in [15]. The root mean square of the projected scattering angle is hereby modeled by a Gaussian core distribution with a similar σ_{core} as given by the Highland formula and an additional broader Gaussian distribution to approximate the tail contribution. The PDF can be then written as

$$f(\theta_{ms}) = (1-\epsilon)\cdot g_0(\theta_{ms}; 0, \sigma_{core}) + \epsilon \cdot g_0(\theta_{ms}; 0, \sigma_{tail}), \tag{3}$$

with $g_0(x;\mu,\sigma)$ denoting a Gaussian random number distribution of width σ and mean μ. The model parameters ϵ, σ_{core} and σ_{tail} are hereby taken from the given reference. Recently [16] more elaborated models replacing the Gaussian tail description by a non-Gaussian distribution have been developed. The modular design of the ATLAS extrapolation engine hereby facilitates a future inclusion of such refined models by simply exchanging the according simulation component.

Figure 6 shows a comparison of both the Highland approach and the Gaussian mixture model for various particles and materials with the well known Geant4 simulation.

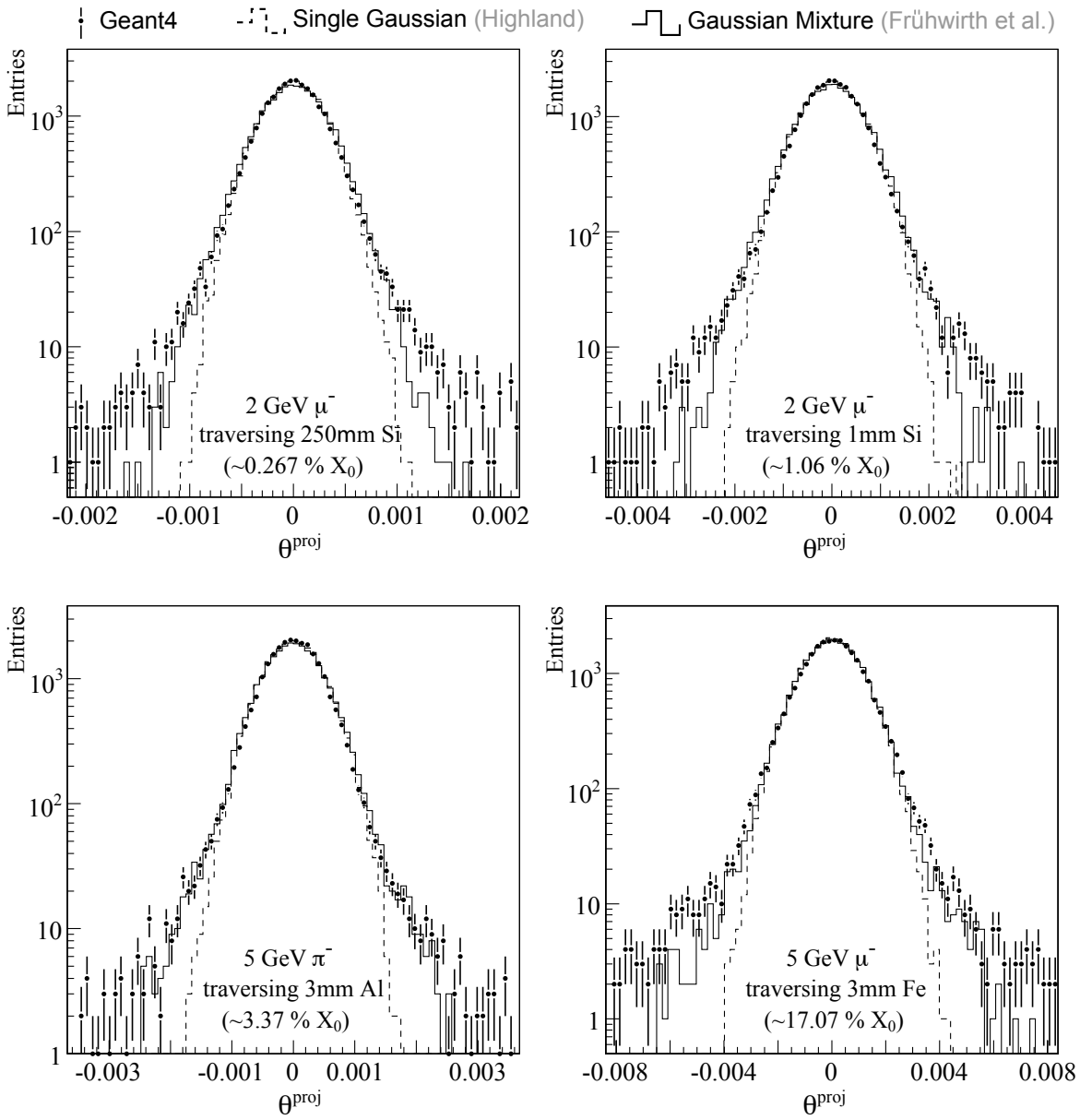


Figure 6: Comparison of the distribution of the projected scattering angle θ^{proj} for the different models available in FATRAS to the Geant4 simulation that builds the base of the ATLAS full detector simulation. Both the single Gaussian distribution following the Highland formula and the Gaussian mixture model are shown for different particle momenta, materials and layer thickness.

Both, the Highland formula and the Gaussian mixture model only provide an estimation of the projected scattering angle, the deflection of the particle due to multiple scattering requires, however, a three-dimensional modeling. The full deflection is in FATRAS simulated by taking a random number α from a uniformly distributed range $\alpha \in [0, 2\pi)$ as a deflection α in the polar direction.

Energy Loss Simulation The energy loss of particles that traverse detector material is mainly dominated by electromagnetic processes. For particles heavier than an electron, energy loss due to ionisation is the by far dominating process. It is well described by the theory of Landau [17], and defines an asymmetric probability density $\rho(\Delta)$ function around a narrow peak through an integral equation

$$\rho(\Delta) = \frac{1}{2i\pi} \int_{c-i\infty}^{c+i\infty} \exp(s \log s + \Delta E \cdot s) ds \tag{4}$$

It can be shown that this leads to a most probable energy loss

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\beta^2\gamma^2}{I} + \ln \frac{\xi}{I} + 0.2 - \beta^2 - \delta \right], \tag{5} $$

with $\xi = ZN_a \frac{k}{\beta^2} t$, when t denotes the thickness of the traversed material. Additionally, when investigating the asymptotic behavior for $\gamma \gg 1$ (i.e. $\beta \approx 1$), which is well met by all particles of interest in this context, the density correction δ can be modeled as

$$ \delta \approx 4.447 - \ln \gamma^2. \tag{6} $$

This simplifies Eq. (5) to

$$ {}_L\Delta_p = \xi \left[\ln \frac{2mc^2\gamma^2}{I} + \ln \frac{\xi}{I} - 0.8 + 4.447 \right]. \tag{7} $$

No theoretical model exists so far to determine the width of the Landau distribution analytically. In general, it is derived from cumulative sampling which would be far beyond the scope of a fast track simulation. In FATRAS, the energy loss of heavy particles is sampled as a Monte Carlo based Landau distribution using the most probable value ${}_L\Delta_p$ as given in Eq. (7) and a parameterised width of the distribution that has been obtained using the Geant4 simulation toolkit. Good agreement between the energy loss distribution between FATRAS compared to Geant4 could be achieved. Figure 7 shows a comparison of the energy loss distributions as given by Geant4 and FATRAS for heavy particles in different materials and varying momenta.

When traversing 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 addition that originates from a long tail towards high energy loss. The according theory for the bremsstrahlung loss has been developed by Bethe and Heitler [18] and is only briefly described in the following. May z in the following denote the ratio between the final energy E_f and the initial energy E_i and t describe the thickness of the traversed material in terms of radiation length X_0 . The PDF for the fraction factor $z \in (0, 1)$ can then be written as

$$ \rho(z) = \frac{[-\ln z]^{c-1}}{\Gamma(c)}, \tag{8} $$

where c denotes $c = t/\ln 2$. The factor z is sampled in FATRAS, which is carried out 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. (8), while the `GSFPDF AlgTool`, that also implements the `IEnergyLossUpdator` interface models the Bethe-Heitler distribution as a sum of single Gaussian distributions, each of which weighted through a given probability. It has been developed for validating the *Gaussian Sum Filter* (GSF) [19].

Photon Emission Significant loss of energy due to bremsstrahlung causes another aspect to be tackled: 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 leptonic 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 still lead to different cluster signatures in the calorimeter. The dedicated handling of these photons, covering both effects — the conversion and the transport to the calorimeter volume — is in detail described in Sec. 3.4 of this document.

The FATRAS bremsstrahlung model calculates the emission of hard photons and their respective angle to the initial electron. The photon energy corresponds hereby to the energy loss fraction according to the Bethe-Heitler theory as given in Eq. (8). It can be assumed that the angle of emitted photon w.r.t. the parental electron direction is proportional to m_e/E_e; electrons that are matter of track reconstruction in the ATLAS detector have typical momentum values that are significantly higher than

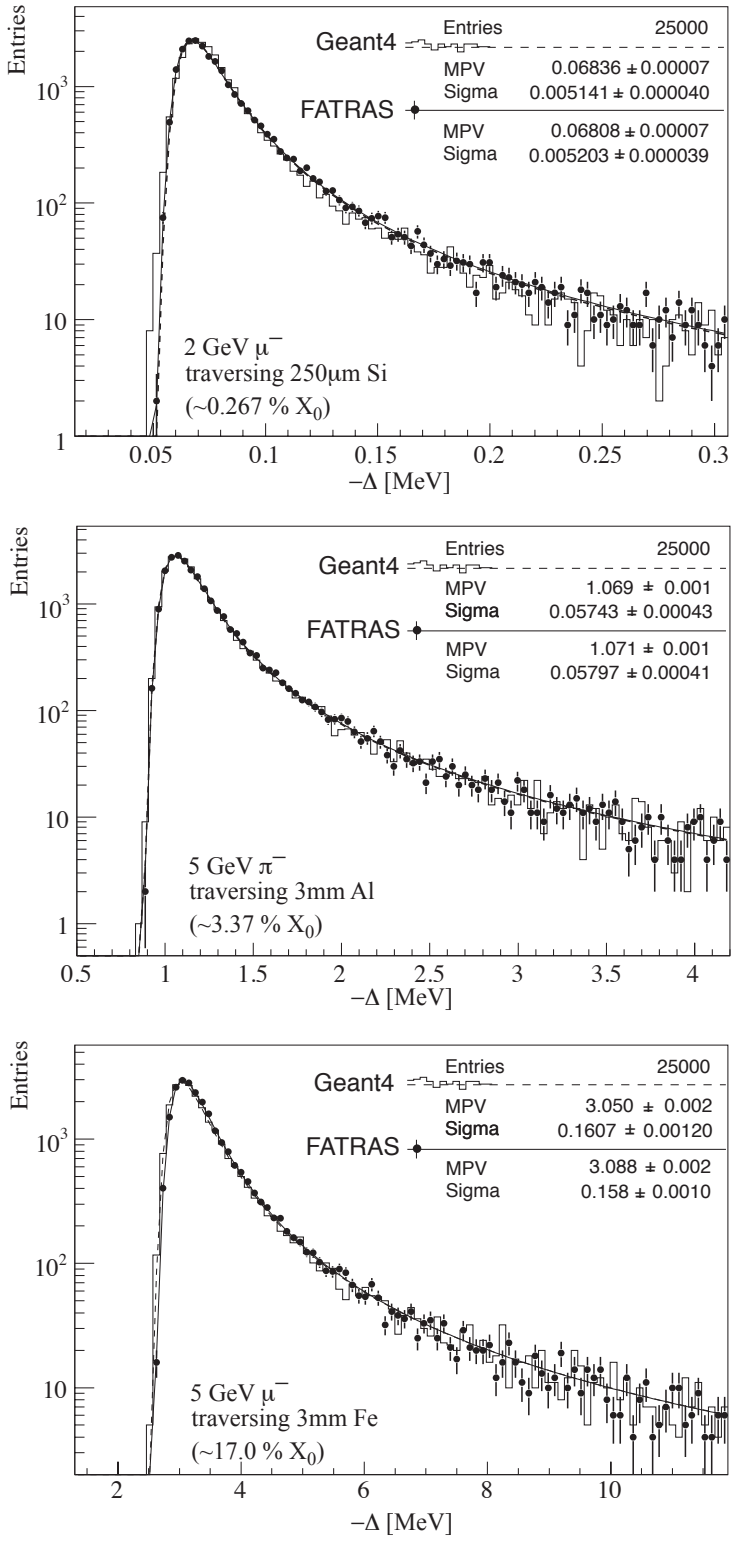


Figure 7: Energy loss distributions of heavy particles for different material layers and momentum ranges, showing the full Geant4 simulation in comparison to the FATRAS energy loss implementation. FATRAS uses the well-known Landau formula for the determination of the most probable energy loss value and a parameterised width of the distribution that has been determined from the Geant4 simulation.

the electron mass, thus, the emitted photons are almost collinear to the original electron trajectory. Figure 8 shows a comparison of emitted high-energetic photons between Geant4 and FATRAS for electrons with a transverse momentum of $p_T = 15$ GeV within the ID tracking acceptance region $|\eta| < 2.5$. The full detector simulation produces usually a higher number of emitted photons, mainly due to the fact that in FATRAS only a limited number of photon-conversion iterations are carried out, while in Geant4 a full cascade simulation is performed; this leads to an underestimation of brem photons in the low momentum spectrum. In a recent study it could be shown that Geant4

and FATRAS behave very similarly in case that both simulation engines are restricted to only one iteration in the photon-conversion cascade. A comparison of FATRAS with a modified Geant4 output is briefly discussed in the Appendix, Sec. A.3.

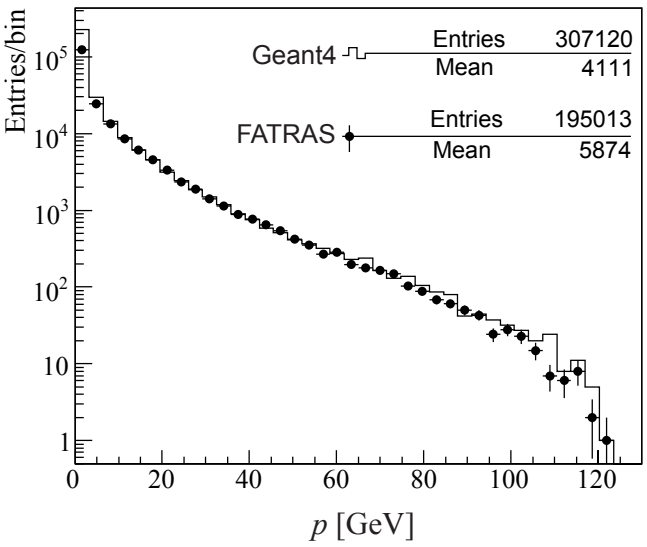


Figure 8: The momentum distribution of hard photons that are emitted from simulated electrons using Geant4 and FATRAS. An identical input sample of 50000 single electron tracks with transverse momenta of $p_T = 15$ GeV are used and restricted to $|\eta| < 2.5$. When restricting the electron energy to 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.

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 model for this is planned to be integrated after 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 ($\pi \rightarrow \mu\nu$) and the scattering process ($\pi \rightarrow \pi$) 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 module. According to the philosophy of a fast and light-weight simulation, a parameterised approach is pursued using properties and fits from the full simulation. For this purpose, a large event sample containing one charged pion (π^+ and π^-) with flat energies between 15 GeV and 50 GeV, flat η and ϕ distributions has been investigated. The hadronic shower model is carried out in a simplified way and is parameterised from data that has been simulated with Geant4. It includes several fit parameters and restricting assumptions and is thus described in more detail in this context.

First of all, the probability of a hadronic interaction cascade has to be calculated which is allowed only for charged pions, kaons and protons as stable, charged particles with energy above 1 GeV (adjustable). Since the hadronic interaction length λ is currently not accessible through the reconstruction geometry, the radiation length X_0 and the average atomic number $\bar{Z}$ are used in the following approximation, introducing an additional scaling factor s_{had}

$$\lambda = 0.37 \cdot s_{\mathrm{had}} \cdot \bar{Z} \cdot \mathrm{X}_0 \,, \tag{9}$$

so that the probability p of an interaction is given by

$$p = \exp\{-p_c \cdot d/\lambda\}, \tag{10}$$

with d being the thickness of the layer and the path correction p_c for inclined passage of particles. The scaling factor is adjusted by comparing with the full simulation.

The multiplicity N of the simulated shower is extracted from a fit to the multiplicity distribution observed in the full simulation, see Fig. 10; the following fit function has been used

$$g(x) = \exp\left\{-\frac{1}{2}\left(\frac{x-p_1}{p_2} + \exp\left\{-\frac{x-p_1}{p_2}\right\}\right)\right\}., \tag{11}$$

and the obtained fit parameters p_1 and p_2 are given in the appendix, Sec. A.5.
The multiplicity is restricted to $3 \leq N \leq \min(N_{\max}, 34)$, with

$$N_{\max} = 0.25 \cdot \frac{E/\text{MeV}}{1000} + 18, \tag{12}$$

whereas E denotes the energy of the incoming particle.

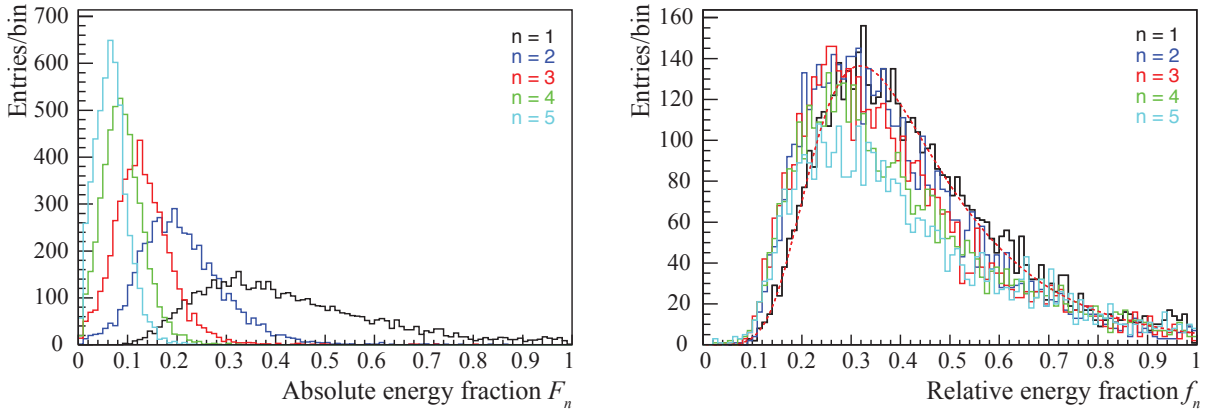


Figure 9: Absolute (w.r.t. the incoming particle) and relative (w.r.t. the rest energy) energy fractions for the five most-energetic particles in the hadronic cascade obtained with Geant4. The right plot includes the generic fit function used for calculating the relative energy fraction.

The energy of the shower particles is determined iteratively. Figure 9 (left) shows the absolute energy fraction $F_n = E_n/E$ of a particle in the shower. The relative energy fractions f_n with

$$f_n = \frac{E_n}{E_{\text{rest}}} \approx f \quad \text{with} \quad E_{\text{rest}} = E - \sum_{i=1}^{n-1} E_i \tag{13}$$

are shown in the right-hand plot of Fig. 9. Since the relative energy fraction f is approximately independent of n (thus the same for all shower particles), this function is fitted with Eq. (11) for various multiplicity bins and used as input for the shower simulation. The energy of a particle n $(1 < n < N)$ is calculated iteratively from the remaining energy E_{rest} with

$$E_n = f \cdot E_{\text{rest}}, \tag{14}$$

whereas f denotes a random number following the distribution Eq. (11) with the parameters that can be found in the Appendix, Sec. A.5. The following constraints are applied:

- first particle: $E_1 > 2 \cdot E/N$,
- energy ordering: $E_{n+1} < E_n$,
- kinematic constraint: $F_n < F_{\max}$ with $F_{\max} = -0.033 \cdot N + 1.16$.

The last cut restricts the phase space and removes unphysically high values of the absolute fraction (high F_n at large multiplicities). The energy calculation using f ends if

- $E_n < 6 \cdot E_{\min}$

- or $E_{\text{rest}} - E_n < 6 \cdot E_{\text{min}}$
- or $E_{\text{rest}} - E_n < 0.1 \cdot E$

with $E_{\text{min}} = 200$ MeV (adjustable). In this case, the energy for the subsequent shower particles is calculated via

$$E_n = E_{\text{min}} + \frac{1}{8}(E_{\text{rest}} - E_{\text{min}}) + g_0(0, 200 \text{ MeV}) \quad \text{if} \quad E_n > E_{\text{min}}. \tag{15}$$

$g_0(\mu, \sigma)$ denotes a random number from a gaussian distribution with mean μ and width σ. The last particle $n = N$ always gets the remaining energy, $E_N = E_{\text{rest}}$. The creation of shower particles ends if $E_{\text{rest}} < E_{\text{min}}$. Note, that due to this restriction the final number of particles created in the shower may be lower than the originally anticipated number of particles in the shower N (see Fig. 10).

The angle θ_n of the outgoing particle with

$$\cos\theta_n = \frac{\mathbf{p}_\pi}{|\mathbf{p}_\pi|} \cdot \frac{\mathbf{p}_n}{|\mathbf{p}_n|} \tag{16}$$

is assumed to be proportional to the inverse momentum of the particle, $\theta_n \propto 1/|\mathbf{p}_n|$. This gives a reasonable description, but is limited due to the restrictions on the energies in the shower simulation (here: $\theta_{\text{max}} \sim 2$ corresponding to $E_{\text{min}} = 200$ MeV). As shown in Fig. 11 (h) it is not possible to generate a sufficient number of particles at extremely large emission angles with this approach.

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. Because of this, only these particles are created with fractions 33% (π^0), 25% (π^+), 25% (π^+), 10% neutrons and 7% protons. These fractions are adjusted to the shower content generated by Geant4 and can be modified as tuning parameters.

Figure 10 illustrates the particle multiplicity in hadronic showers from nuclear interactions of the primary traversing particle with detector material. It includes the distribution obtained with Geant4, the generic fit function used for the parameterised level and the according distribution produced with FATRAS. Events with very low particle multiplicities that appear in the Geant4 simulation are suppressed in FATRAS due to the chosen momentum cuts that require a minimal particle energy for the application of a hadronic interaction.

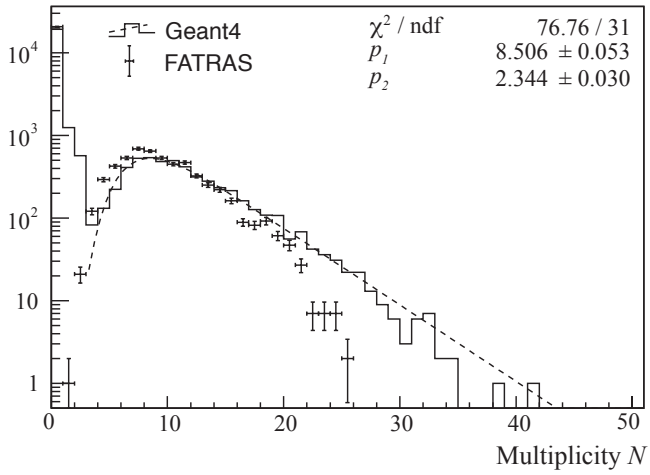


Figure 10: Particle multiplicity N in hadronic showers generated with Geant4 and FATRAS. The fit function which is basis for the FATRAS hadronic shower model is also shown in the histogram

Figures 11 (a) to (f) show comparisons of absolute and relative energy fractions for the four most energetic shower particles between the Geant4 simulation and FATRAS. It also includes the overall direction distribution of the simulated shower particles as their relative output direction to the original hadron.

In general, a good agreement in the distributions between the parameterised shower in FATRAS and the full simulation can be observed.

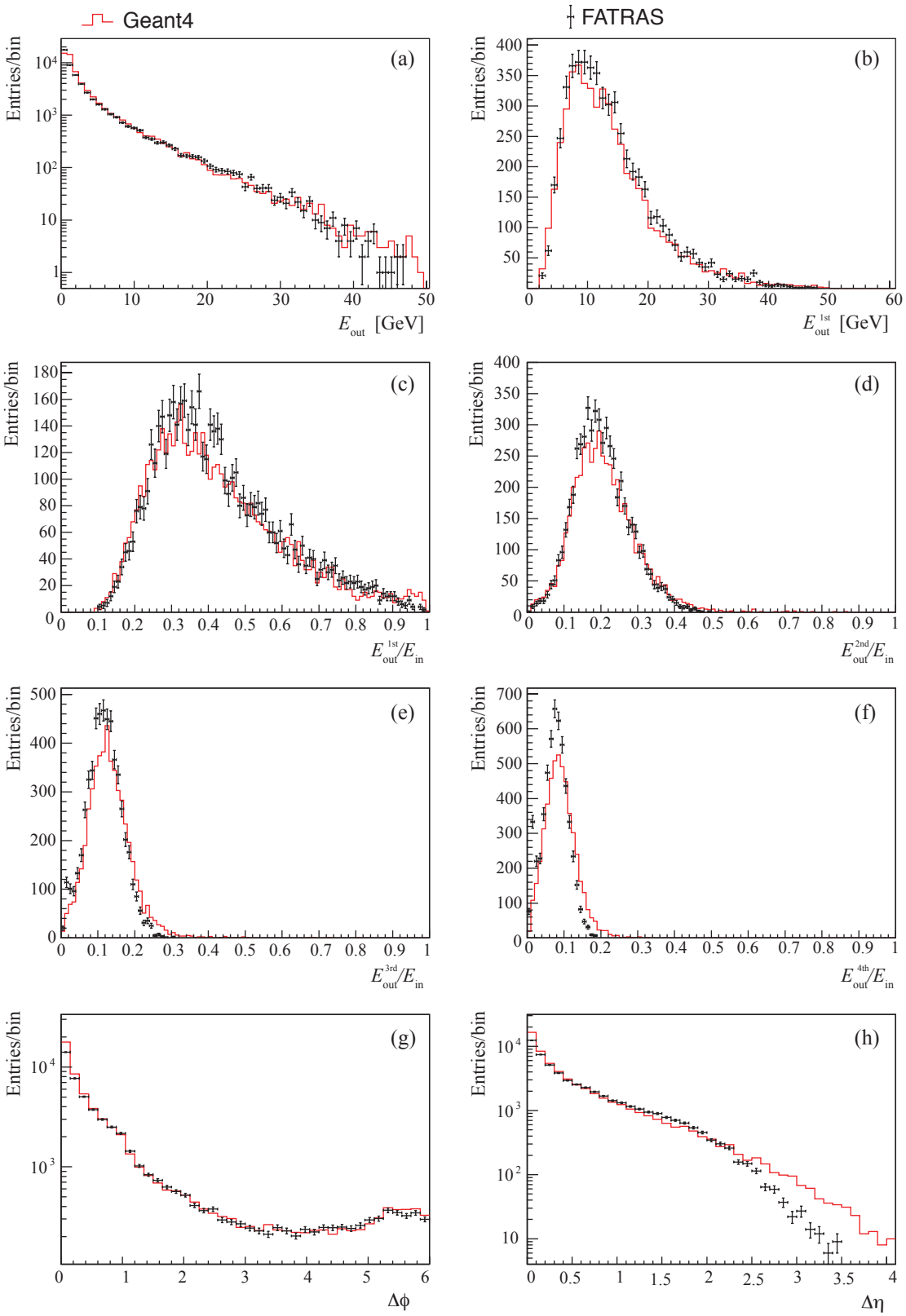


Figure 11: (a) and (b) - comparison of the absolute energy fractions of all and the most energetic shower particle in hadronic showers caused by nuclear interactions simulated with Geant4 and FATRAS, respectively. (c) to (f) show the relative energy fractions of the four most energetic shower particles, while (g) and (h) show the angular directions of the outgoing shower particles w.r.t. the incoming particle direction simulated with Geant4 and FATRAS.

3.3.2 Track Creation from a given Trajectory

The extrapolation engine is not capable of producing 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. A dedicated `AlgTool` — the `TrackCreator` — performs this task; it calls the extrapolation engine and dissolves the returned trajectory into 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 an appropriate clusterisation or hit 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 for the sub-detectors can hereby be applied by introducing efficiency parameters for the sub-detectors.

Clusterisation and Simplified Smearing A correct clusterisation description is essential for obtaining meaningful track parameter resolutions when being compared to tracks from the full simulation and offline reconstruction chain. It can be shown, that the impact parameter resolutions are dominated by the measurements on the innermost detection layers[8], while the momentum resolution depends on the entire hit collection of the track. The clusterisation model for 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 parameterisation is the most significant one.

In the ATLAS offline reconstruction, the pixel clusterisation uses an analog model that determines the cluster position with the help of the time over threshold information that is available for every readout (or simulated) pixel signal [20]. 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, only energy loss is applied to the particle when traversing detector material. A geometrical clusterisation approach is used instead to recreate similar cluster sizes and shapes as given through the realistic full detector simulation and digitisation. The geometrical clusterisation is an analog clustering process that uses the path length of the track in the single intersected pixels to calculate the cluster position from the center positions of the associated pixel cells. Given n geometrically intersected pixels with s_i the according path length in pixel i, the cluster position $\mathbf{p}$ is then calculated as

$$\mathbf{p} = \frac{1}{\sum_{i=0}^{n} \Theta(s_i - s_{cut}) \cdot s_i} \sum_{i=0}^{n} \Theta(s_i - s_{cut}) \cdot s_i \cdot \mathbf{p}_i, \tag{17}$$

where the $\mathbf{p}_i$ denote the individual center positions of the intersected pixels and the Heaviside function Θ is inserted to demand a minimal path length of the track within the single silicon pixel which is equivalent to a minimal charge deposition in a single pixel; s_{cut} becomes hereby the single model parameter of the pixel clusterisation. Since FATRAS does not simulate the charge deposition in the Silicon pixels, a correction of cluster shifts accounting for the Lorentz force effects on drift electrons has to be applied. This fake correction is then cancelled by the official ATLAS clusterisation tool and may be one source for the slight discrepancies between cluster sizes in the full offline chain and FATRAS. Figure 12 shows an illustration of the geometrical clusterisation approach.

The clusterisation 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 the intersected strips to build the final cluster position. This yields to one-, two- and three-strip clusters without using any charge deposition information. For the *Transition Radiation Tracker* (TRT), the hit creation in FATRAS is simply done by Gaussian smearing of the geometrical closest approach to the wire, 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 for different drift time bins, 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.

[8] This can be easily shown by assuming a simplified 2 layer detector model, see [21].

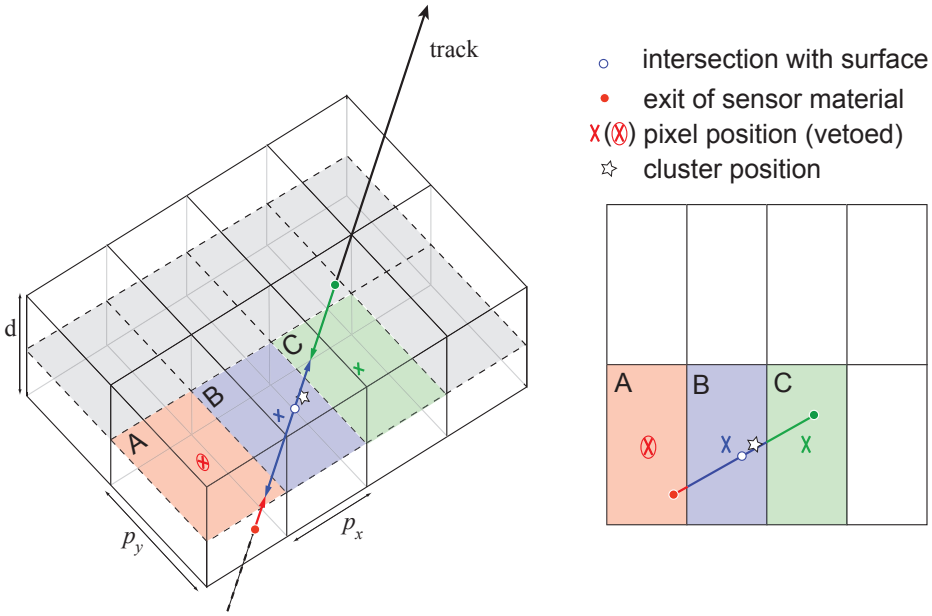


Figure 12: The geometrical clusterisation model for the pixel detector that is used in the FATRAS simulation. Analog clusterisation is hereby performed by weighting the readout center positions of intersected pixels with the track distance inside the pixel. Single silicon pixels that are traversed by the track, but do not host a sufficiently long path length for the pixel to detect a signal are vetoed for the clusterisation process (pixel A).

Figure 13 shows a comparison of obtained cluster sizes and resulting hit residuals on the tracking devices between offline reconstruction and the FATRAS clusterisation model for all three tracking technologies in the ID, the double Gaussian smearing of the TRT is, however, omitted in this figure.

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.

3.4 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 — as described in Sec. 3.3.1 — are collected in one `ParticleState` container and further processed by a dedicated algorithm, the `PhotonProcessing`. Each photon is hereby 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 conversions of the photon into an electron-positron pair depending on the amount material when a layer is crossed[9]. This process can be iteratively repeated to account for a more detailed cascade simulation.

Figure 14 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 can be seen through 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.

[9]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.

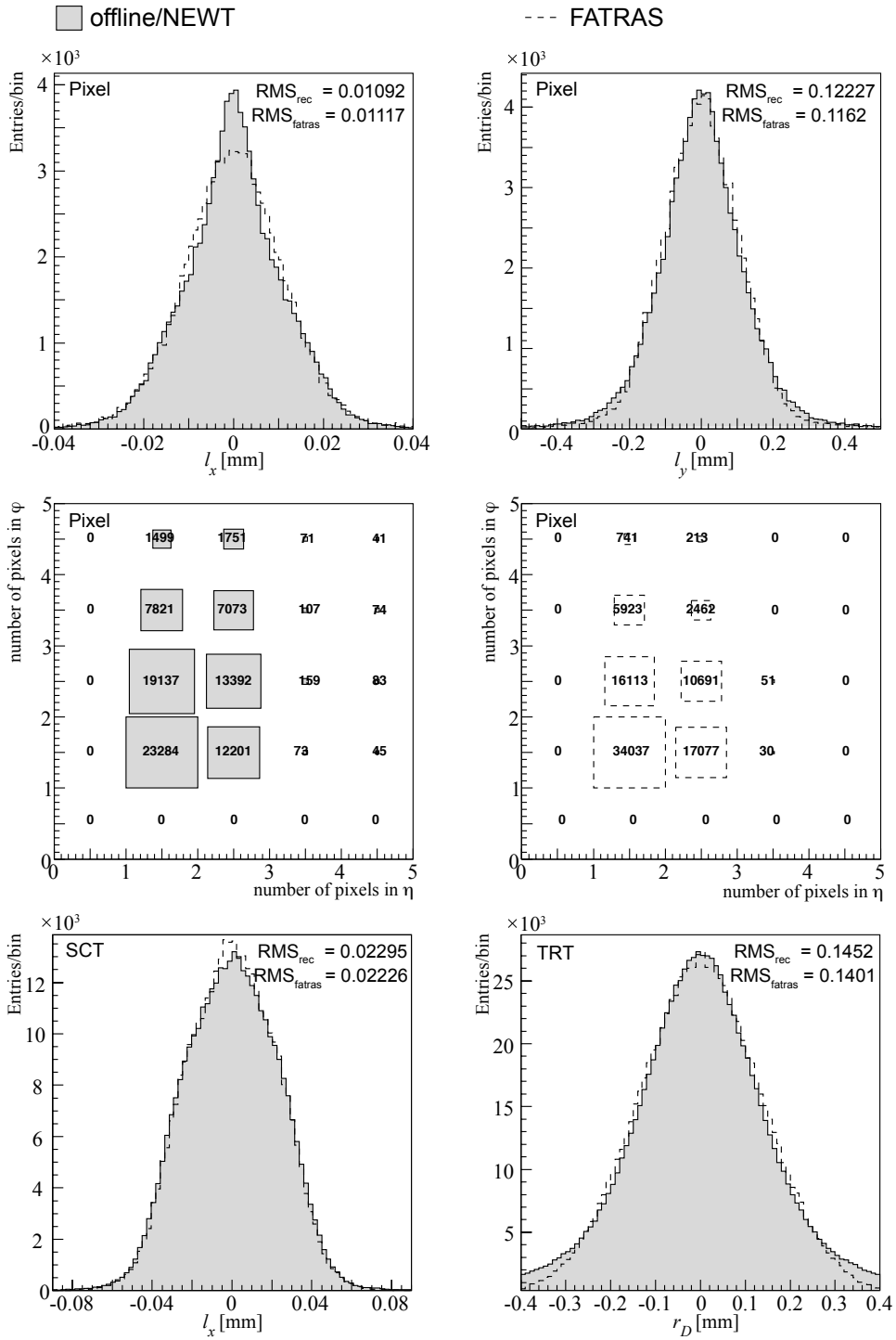


Figure 13: A comparison of cluster sizes and resulting residuals on the tracking devices between offline reconstruction and FATRAS. Most emphasis has been put on a correct description of the pixel clusterisation since it has maximum impact on the track parameter resolutions. The slightly narrower pixel residuals from the offline reconstruction may be caused by broader cluster sizes that could not fully be reproduced with the current geometrical clusterisation module in FATRAS.

The conversion probability has been parameterised and fitted to data taken from [22], where the photon conversion probability dependent on the traversed material thickness is given for aluminum and photons with energies between 0.1 and 100 GeV . The given parameterisation fits fairly well to the numbers obtained from fast simulation. However, additional scaling parameters have been inserted to allow custom tuning of FATRAS with respect to Geant4. These include also the energy and angular

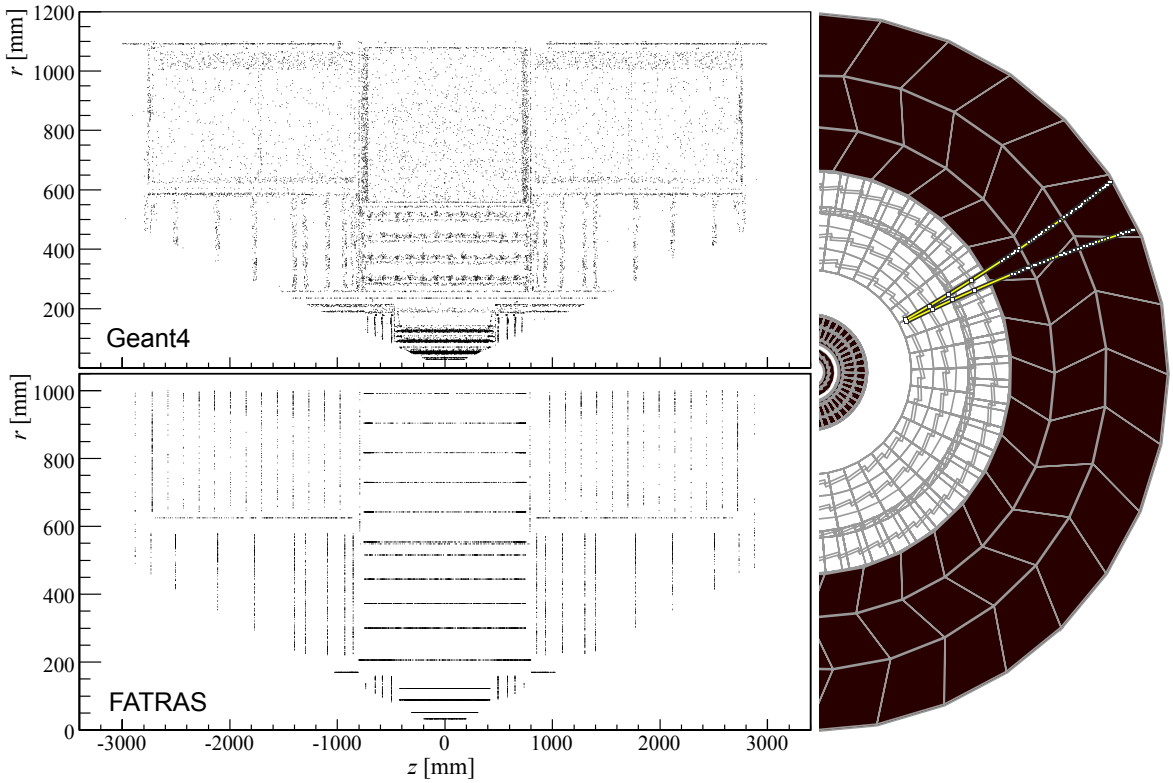


Figure 14: Photon conversions in the ATLAS Inner Detector, simulated with Geant4 and FATRAS; the simplified simulation geometry of FATRAS can be seen; it is limited to several discrete layers, while the Geant4 simulation geometry is more detailed. The picture to the right shows a photon conversion before the first SCT barrel layer simulated with FATRAS and shown with the ATLAS event display ATLANTIS.

distribution of the produced particles with FATRAS calculated using the same model [23] that is implemented in the Geant4 simulation toolkit. Since the Geant4 model provides a very accurate description of the material, while the `TrackingGeometry` represents only a strong simplification of the actual detector geometry, some tuning has to be applied. Based on large scale single photon events simulated with both, FATRAS and Geant4 the photon probability and energy spectrum of the pair products have been compared and it could be shown that only the latter requires a constant scale factor to yield a satisfactory agreement between FATRAS and the full simulation. The scaling procedure has been performed by using photon conversions that have been caused by the beam pipe material, since the beam pipe is implemented in both simulation geometries to the same detail. A detailed list of the tuning parameters can be found in the Appendix, Sec. A.5, of this document.

Figure 15 shows a comparison of simulated electron energies for particles originating from photon conversions in the ATLAS ID and gives a comparison of the mean electron energy depending on the transverse photon momentum.

When comparing a full physics event, FATRAS underestimates the number of brem photons and photon conversions for the simple reason that only a limited number of iterations of the entire cascade are carried out. For the simplest possible FATRAS configuration that only implements one iteration, this means that an electron which comes from a pair production process can indeed again 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 Sec. 3.6. An already mentioned further investigation of this subject revealed the necessity to integrate a second iteration of the photon-conversion cascade (→ Appendix, Sec. A.3), which is planned to be fully integrated in the ATLAS release 14.0.0.

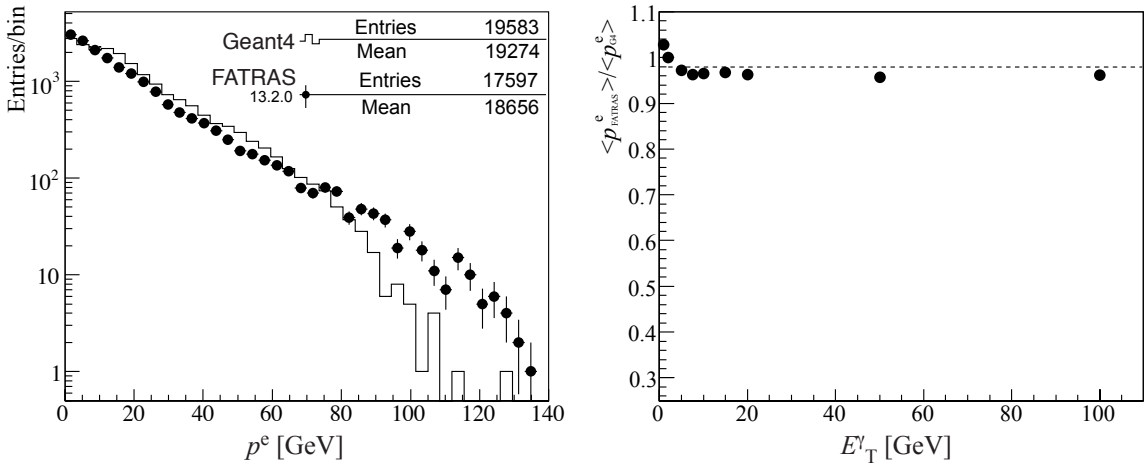


Figure 15: Comparison (Geant4/FATRAS) of the electron energy distribution originating from photon conversions in the ATLAS ID for photons with initial transverse energy of $E_T^{\gamma} = 15$ GeV (left). The right plot shows the ratio of the mean child electron momentum $< p^e_{FATRAS} > / < p^e_{G4}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

3.5 Track Refitting

The track refitting module has to be only performed 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 on the first measurement layer. The special `TrackRefitting Algorithm` that is contained in the FatrasAlgs package allows therefore to smear the initial perigee representation. The smearing, however, has to be kept in a regime that is compatible with the linearised measurement model that is inert to most fitting techniques; the smearing parameters for the applied Gaussian smearing of the initial perigee representation is accessible as tuning input from the job configuration.

3.6 Exit State Creation

The creation of particle representations at the exit surface of the tracking detector volume marks the last step in the FATRAS simulation sequence. It has little to do with the track simulation itself, but prepares FATRAS for future integration with both the full or fast calorimeter simulation, such as a future extension of FATRAS for the Muon Spectrometer.

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. A small outlook on a potential integration of FATRAS with a fast shower application can be found in Sec. 6.1.

3.7 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 some pattern recognition effects such as holes on track or missed extensions from one sub-detector to another[10], see Sec. 4.1.1. When using FATRAS to prepare input for the standard

[10]The standard ATLAS reconstruction incorporates an inside-out procedure for the pattern recognition that starts at the innermost silicon layers and extends a successful silicon track segment to the TRT straw detector at larger radii.

offline reconstruction, another aspect has to be considered: 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. Since the pattern recognition is fully performed, a special post processing in terms of hit correction is not necessary. The different topologies that originate from effects in the pattern recognition process are automatically present in the reconstruction mode; a comparison of reconstruction efficiencies for different particles can be found in Sec. 4.2. What remains is the hit container and noise creation handling: 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. Noise hits can hereby be added independently to every sub-detector data collection and the noise levels can be adjusted freely by the user. The chosen defaults of noise levels in the three sub-detectors can be found in the Appendix, Sec. A.5. This turns FATRAS into a powerful engine for occupancy studies, see Sec. 5.1; the low execution time enables a *scan* through different noise level settings.

4 Validation and Performance

The dedicated aim of a fast simulation is to build a fast alternative to the full detector simulation (and — in the case of the ATLFAST simulation or the FATRAS refit mode — also of the reconstruction), which marks an important guideline for the validation. Thus, the main focus has to be the validation of FATRAS against results obtained with the full offline chain and not to optimise the FATRAS performance itself. As a consequence, some tuning parameters have to be introduced that allow to turn the outcome of the fast track simulation more towards the according results obtained with the full offline chain[11], sometimes this turns to the simple but non-trivial task of *how to make things worse*. Some requirements such as the correct description of material effects or a proper clusterisation model have been already presented in Sec. 3 of this document. The following sections will concentrate on hit statistics comparison and track parameter resolutions for FATRAS compare them with full simulation results, and — if applicable — with according results of the fast ATLAS simulation program ATLFAST.

Validation `Algorithm` classes Several dedicated `Algorithm` classes exist in the FATRAS repository that prepare convenient user output such as track resolution and correlation data in form of ROOT [24] tuples. Since in FATRAS the simulation trajectory is known[12] as an own `Track` object, convenient truth association can be performed in terms of associative containers that relate the simulated track to the refitted (or reconstructed, respectively) and a similar structure for the hit association. In the refit mode this can be done without any matching procedure, since the simulated and refitted track are available at once in the `TrackRefitting` module. For the reconstruction mode, a certain `TrackAssociation Algorithm` has to be performed for matching the simulated and reconstructed track collections. The matching procedure is hereby carried out in (η, ϕ) bins and is based on the lowest χ^2 of the candidate track pairs. The full integration of the MC truth chain in FATRAS that has the same identical structure as in the full offline chain allows also to include all standard validation modules that are used in the offline reconstruction.

4.1 Single Track Validation

A comparison of pure track resolutions is best performed on single track events, since pattern recognition effects or shared clusters arising from high track densities can be neglected in this situation.

Validation Samples and Quality Cuts For the following comparison, dedicated validation samples have been produced for single particle events with constant transverse momentum. 50 000 single track

[11] FATRAS leads, in general, to slightly better performance in track parameter resolutions and hit statistics. This is, since only a limited number of disturbances are simulated in the fast track simulation; the omitted effects are typically of second or third order and contribute usually mainly to the tail distribution of the track parameter resolutions.

[12] When using the Geant4 detector simulation, a truth trajectory has to be build first and matched with a reconstructed track object. For a direct comparison, this also involves the translation of the kinematic input object to the a track representation.

events have been produced for electrons, pions, muons and photons in the transverse momentum range $p_T \in [1\ \text{GeV}, 1\ \text{TeV}]$, spreading over a pseudo-rapidity of $|\eta| < 3.0$. The production vertex of the single particles is smeared according to standard ATLAS vertex smearing, i.e. Gaussian smearing with $\sigma_{xy} = 15\ \mu\text{m}$ in the transverse plane and $\sigma_z = 56\ \text{mm}$ in the longitudinal direction.
Tracks that have been used for the comparison passed the following generator cuts:

- *prompt particles* : generation barcode < 100000
- *tracking acceptance* : $|\eta| < 2.5$

Successfully reconstructed tracks have in addition passed the following reconstruction quality cuts:

- *prompt particles (I)* : $d_0^{rec,PV} < 2\ \text{mm}$, when d_0 is expressed to the primary vertex (PV)
- *prompt particles (II)* : $z_0^{rec,PV} < 10\ \text{mm}$, when z_0 is expressed to the PV
- *Si hits* : number of hits in the silicon detector greater than 6
- *TRT extension* (if $|\eta| < 2.0$) : number of TRT hits greater than 0

For ATLFAST, the hit cuts can not be applied, since no hit information for the track is available.

4.1.1 Hit Creation

A correct number of simulated hits per track is not in particular necessary for all tracking related studies with the ATLAS detector. The impact parameter resolution, on the one hand, is e.g. almost entirely dominated by the innermost layer measurements and missing hits further along the track are of small influence for the spatial resolution. The momentum resolution, on the other hand, improves substantially with the track length, since a longer lever arm helps to correctly determine the track curvature. In the offline reconstruction, the pattern recognition is responsible for finding the hits that are likely to belong to one single track, while the track fit is performed to test (and either verify or falsify) the track hypothesis and finally to determine the track resolution. For convenience, the track is mostly expressed with respect to the production vertex of the particle. Since FATRAS is used to feed the standard offline reconstruction algorithms, many pattern recognition effects (such as missing hits or segments that have not been found by the appropriate sub-algorithms) are directly included. Studies that are sensitive to pattern recognition effects should therefore be based on the reconstruction mode of FATRAS. In the refit mode, however, the pattern recognition is bypassed. An overview on the single CPU time contributions of the various FATRAS modules can be found in Sec. 4.3. To still keep compatibility with offline results, FATRAS offers the possibility to force the introduction of hit finding inefficiencies by losing single hits or full segments. The almost geometrical efficiency of FATRAS can thus be downscaled to meet the figures obtained by the offline reconstruction. The excluded hits can, however, be still included in the event hit collection, to guarantee a correct treatment in the pattern recognition process. Figure 16 shows a comparison of the hit numbers per track for single muon tracks with a momentum of 5 GeV in the ID for both, the calibrated and non-calibrated case in the refit mode. It can be shown that the hit finding efficiency for single muon tracks is practically at a level of 100 % in case that track segments in all sub-detectors are found. The hit numbers for the reconstruction mode are therefore omitted in this illustration, since they overlay with the ones obtained in the refit mode.

4.1.2 Single Muon Samples

Muons create the cleanest signal trace through the detector, since they cause minimal interaction of all charged particles with the detector material. This is, because muons do not undergo nuclear or hadronic interactions with the nuclei of the detector material and neither are they subject to significant energy loss through radiation. The reconstruction, i.e. the finding and successful fitting of the particle trace, of single muon tracks shows almost full efficiency over the geometrical acceptance region of the ATLAS Inner Detector. Single muon events are therefore often taken as reference tracks for the detector performance validation and have also marked the first step in the FATRAS

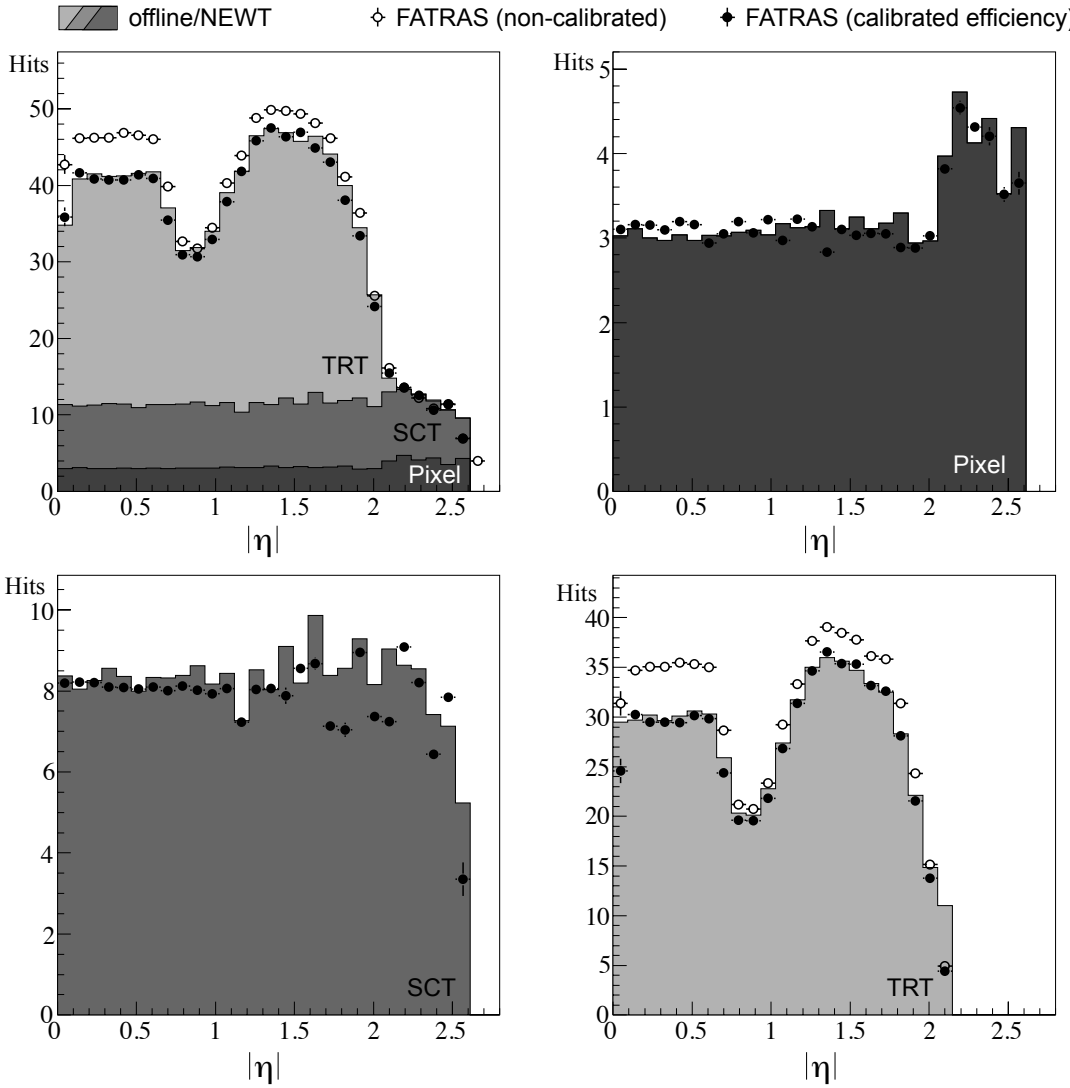


Figure 16: Comparison between the number of hits created on a FATRAS track to those from equivalent tracks simulated with the full ATLAS Geant4 detector simulation and reconstructed with New Tracking (NEWT). 5000 single muon tracks with a momentum of 5 GeV have been used for this study. In FATRAS, the nearly geometrically limited efficiency that leads in general to higher hit numbers per track can be adjusted in different eta bins.

validation against offline simulated and reconstructed events. Good agreement between tracks from offline reconstruction events and FATRAS could be met over a large momentum range. Figure 17 and Fig. 18 show comparisons of track parameter resolutions in the low and high momentum regimes for singe muons and ATLFAST, the offline chain and FATRAS, respectively. FATRAS is in comparison to the ATLFAST simulation superior in describing the tail distributions due to scattering effects at the low momentum spectrum and also in the correct cluster description at the high momentum limit. The asymmetric tail in the reconstruction of low p_T muons is due to the usage of single-charged tracks (μ^-) together with an overestimation of the ionisation loss at low energies in the reconstruction modules. This effect vanishes for FATRAS since the identical parameterisation of ionisation loss is used in both, simulation and reconstruction modules.

The $A \oplus B$ Model A general feature of the track parameter resolutions is that they degrade with lower momenta, since the contribution of multiple scattering is inverse proportional to the particle momentum, see Eq. (2). In a simplified model, the dependency of the track parameter resolutions on

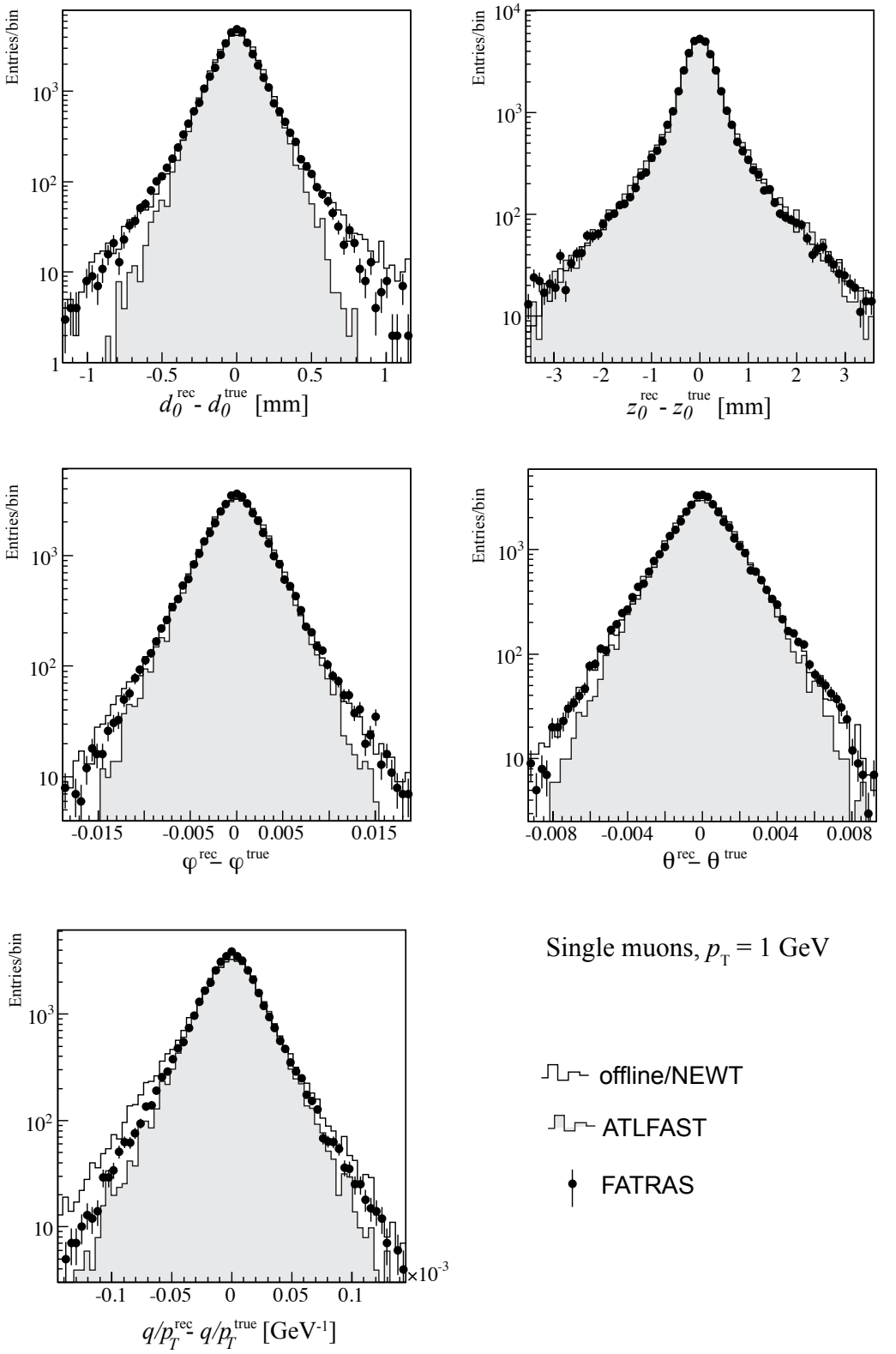


Figure 17: Track parameter resolutions for single muon tracks in the low momentum limit over the entire acceptance range of $|\eta| < 2.5$ of the ID. Results obtained from offline reconstruction are shown and compared to FATRAS tracks from the refit mode (dotted line) and results from the ATLFAST smearing (shaded area). FATRAS performs better than ATLFAST in the correct description of the tail distributions.

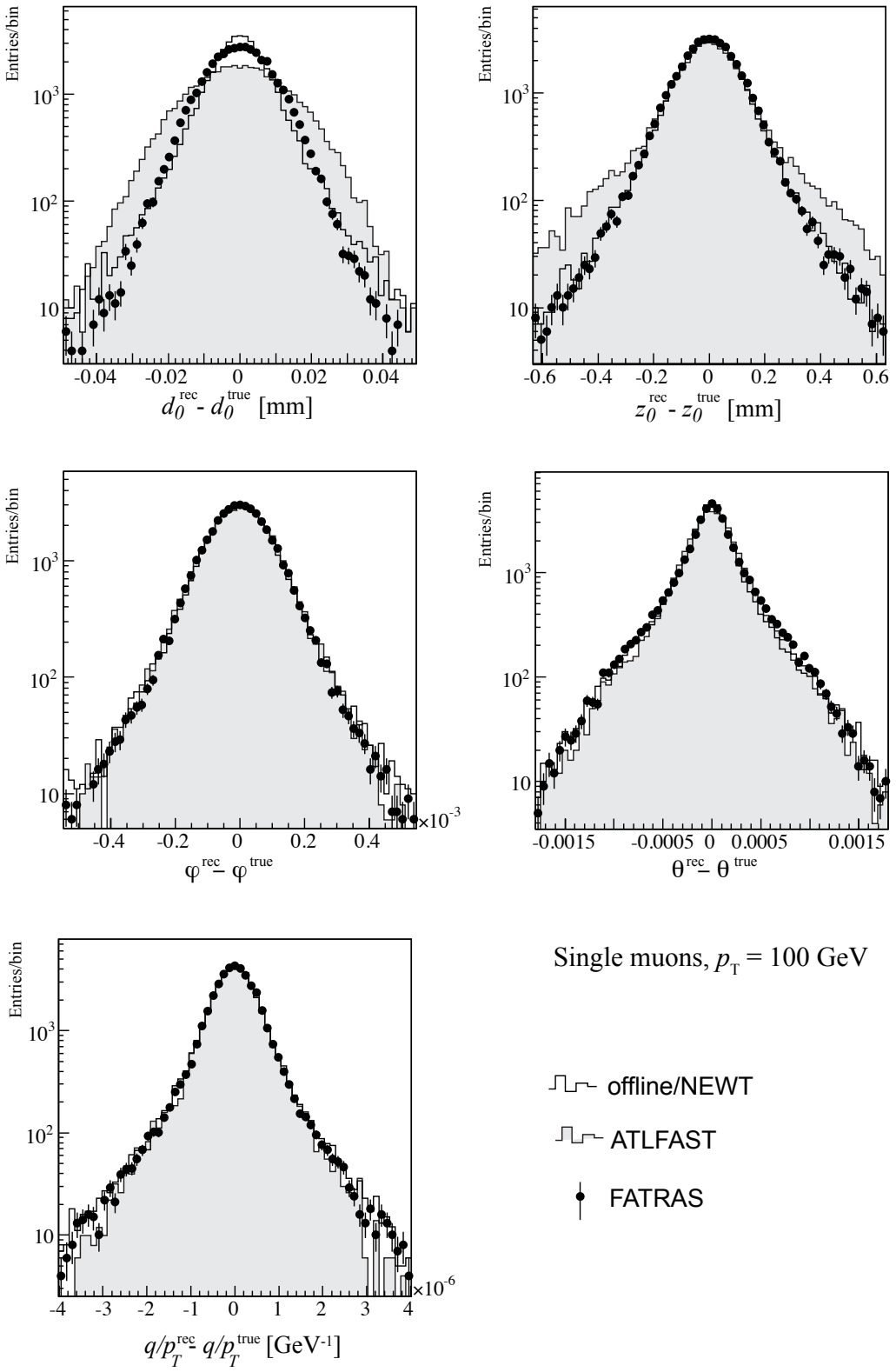


Figure 18: Track parameter resolutions for single muon tracks in the high momentum limit over the entire acceptance range of $|\eta| < 2.5$ of the ID. Results obtained from offline reconstruction are shown and compared to FATRAS tracks from the refit mode (dotted line) and results from the ATLFAST smearing (shaded area).

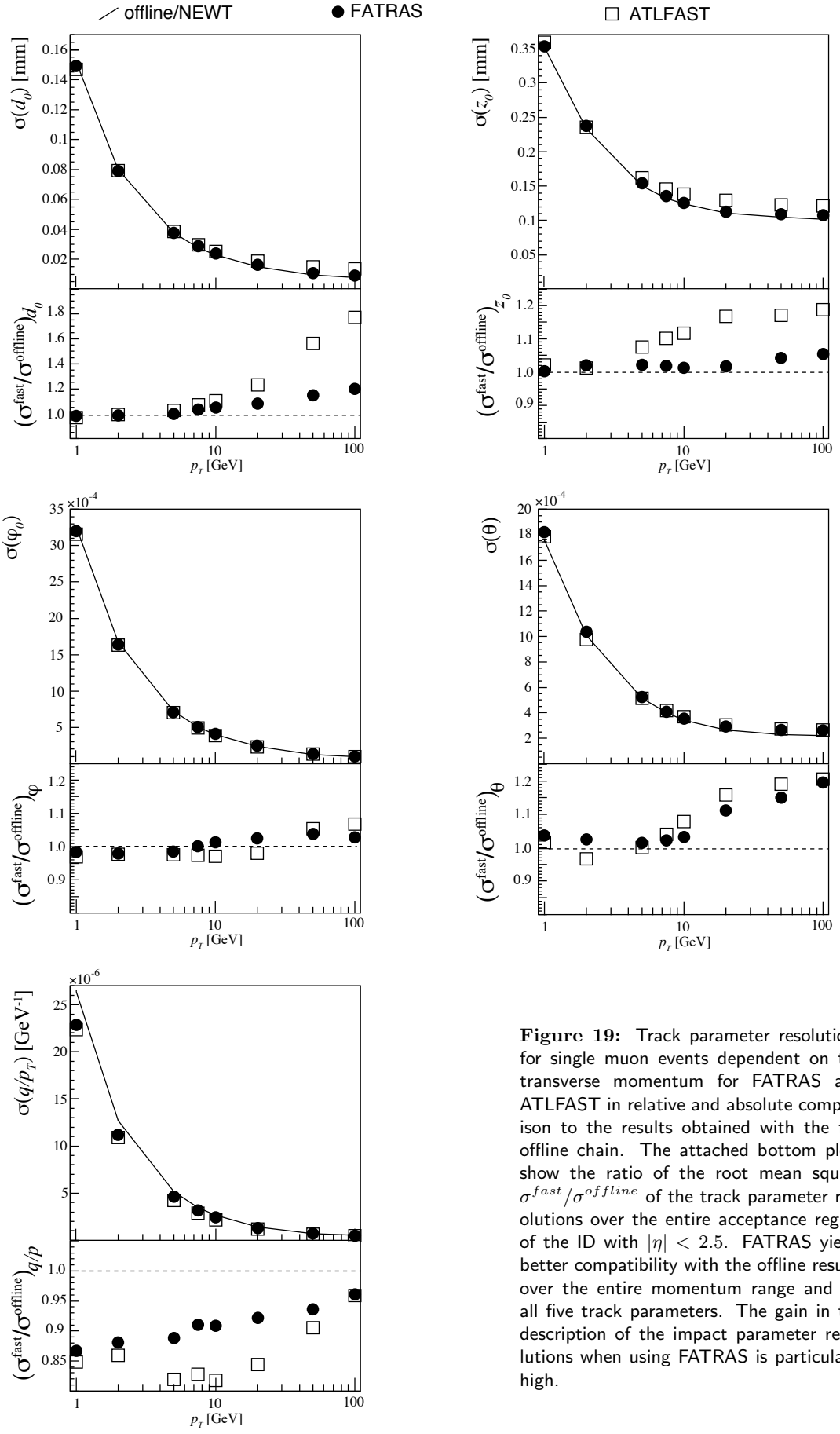


Figure 19: Track parameter resolutions for single muon events dependent on the transverse momentum for FATRAS and ATLFAST in relative and absolute comparison to the results obtained with the full offline chain. The attached bottom plots show the ratio of the root mean square $\sigma^{fast}/\sigma^{offline}$ of the track parameter resolutions over the entire acceptance region of the ID with $|\eta| < 2.5$. FATRAS yields better compatibility with the offline results over the entire momentum range and for all five track parameters. The gain in the description of the impact parameter resolutions when using FATRAS is particularly high.

the momentum can be approximated through the quadratic sum

$$\sigma_\tau = A_\tau \oplus \frac{B_\tau}{p}, \tag{18}$$

where A_τ marks the intrinsic detector resolution and B_τ includes the multiple scattering contribution for the given track parameter τ, see [21]. This approximate model — in the following referred to as $A \oplus B$ *model* — has been expanded to more sophisticated version that includes a Taylor expansion of the underlying track model [25]. For the further comparison, however, the $A \oplus B$ model will be taken, since it is intuitive and accurate enough for a simple comparison. The $A \oplus B$ model is not restricted to muon tracks, but it can be best demonstrated for minimal interacting particles, since the resulting distributions are likely to be unbiased and follow a Gaussian PDF[13]. Figure 19 shows an absolute and the relative comparison of the track parameter resolutions in dependence of the transverse momentum for ATLFAST and FATRAS with results from the full offline chain. Table 2 shows a comparison of the single parameters obtained for the $A \oplus B$ model fitted to the root mean square (RMS) of the track parameter resolutions for the entire acceptance region of the ID.

Table 2: Track parameter residuals parameterised according to the $A \oplus B$ model, comparing the offline reconstruction with FATRAS and ATLFAST. The given values mark the root mean square of the distributions that include tracks over the entire acceptance range of the ATLAS ID. The track parameter resolutions of muon tracks are very close to Gaussian curves. The RMS is therefore a good measure when overlaying the different resolutions in the various $|\eta|$ bins.

Track Parameter	Function Parameter	**offline/NEWT**	**FATRAS**	**ATLFAST**
d_0	$A_{d_0}[\mu\mathrm{m}]$	8.17	9.76	14.46
	$B_{d_0}[\mu\mathrm{m\,GeV}]$	176.4	176.5	167.4
z_0	$A_{z_0}[\mu\mathrm{m}]$	110.3	114.5	128.5
	$B_{z_0}[\mu\mathrm{m\,GeV}]$	358.0	355.2	349.2
ϕ_0	A_ϕ	$9.3 \cdot 10^{-5}$	$9.6 \cdot 10^{-5}$	$10.1 \cdot 10^{-5}$
	$B_\phi[\mathrm{GeV}]$	$3.58 \cdot 10^{-3}$	$3.56 \cdot 10^{-3}$	$3.46 \cdot 10^{-3}$
θ	A_θ	$2.36 \cdot 10^{-4}$	$2.75 \cdot 10^{-5}$	$2.82 \cdot 10^{-4}$
	$B_\theta[\mathrm{GeV}]$	$1.88 \cdot 10^{-3}$	$1.91 \cdot 10^{-3}$	$1.87 \cdot 10^{-3}$
q/p_T	$A_{q/p_T}[\mathrm{GeV}^{-1}]$	$4.63 \cdot 10^{-4}$	$4.55 \cdot 10^{-4}$	$4.62 \cdot 10^{-4}$
	B_{q/p_T}	0.026	0.023	0.021

Both ATLFAST and FATRAS show a reasonable agreement with the full reconstruction chain, although some additional tuning may have to be applied in order to fully comply with the offline results. ATLFAST shows the same features as demonstrated in Fig. 18, the impact parameter resolutions in the asymptotic high momentum region can not be reproduced. For FATRAS, in particular the good agreement of the multiple scattering term indicates that the reconstruction geometry material description is a good representation of the simulation geometry.

Dependency on the Pseudorapidity The track parameter resolutions do not only depend on the momentum, but also on the pseudorapidity. This is caused by several reasons:

- the distribution of inert detector material is far from being constant (see Fig. 5) in the full η acceptance region of the ID; in general, the material budget increases at higher pseudo-rapidity ranges, since more support structures and cable/electronics material are situated outside the central region. In addition, a geometrical effect plays another role: the incident angle of the track with respect to the intersected detector module varies with $|\eta|$ and thus results in different effective thickness of detector elements and support structures.
- when investigating particles at constant transverse momenta (which makes sense for the momentum resolution, since the transverse momentum is related to the bending power), higher $|\eta|$ values correspond to higher momenta, and have thus a lower multiple scattering contribution;

[13] For electrons, the loss of energy due to bremsstrahlung disturbs the Gaussian character of the track parameter resolutions.

- finally, the changing incident angle results in different cluster sizes; that is why the intrinsic resolution of the track parameters varies for different pseudo-rapidity values even at the high momentum limit, where the multiple scattering contribution can be neglected;

For FATRAS the track parameter resolutions versus pseudorapidity builds a stringent test for the material distribution (described by the `TrackingGeometry`) and the clusterisation module. The η-dependency of the track parameter resolutions is reproduced (without adding any dedicated tuning) when these two modules are described correctly, a feature which will be proven in the following context. In ATLFAST, on the other hand, the η dependency has to be put in *by hand*: the smearing functions are obtained from offline in certain $|\eta|$ bins and an interpolation is being performed to cover the full acceptance region. Figure 20 shows a comparison of the impact parameter resolutions for the offline chain, FATRAS, and ATLFAS when using single muon tracks of different constant transverse momenta.

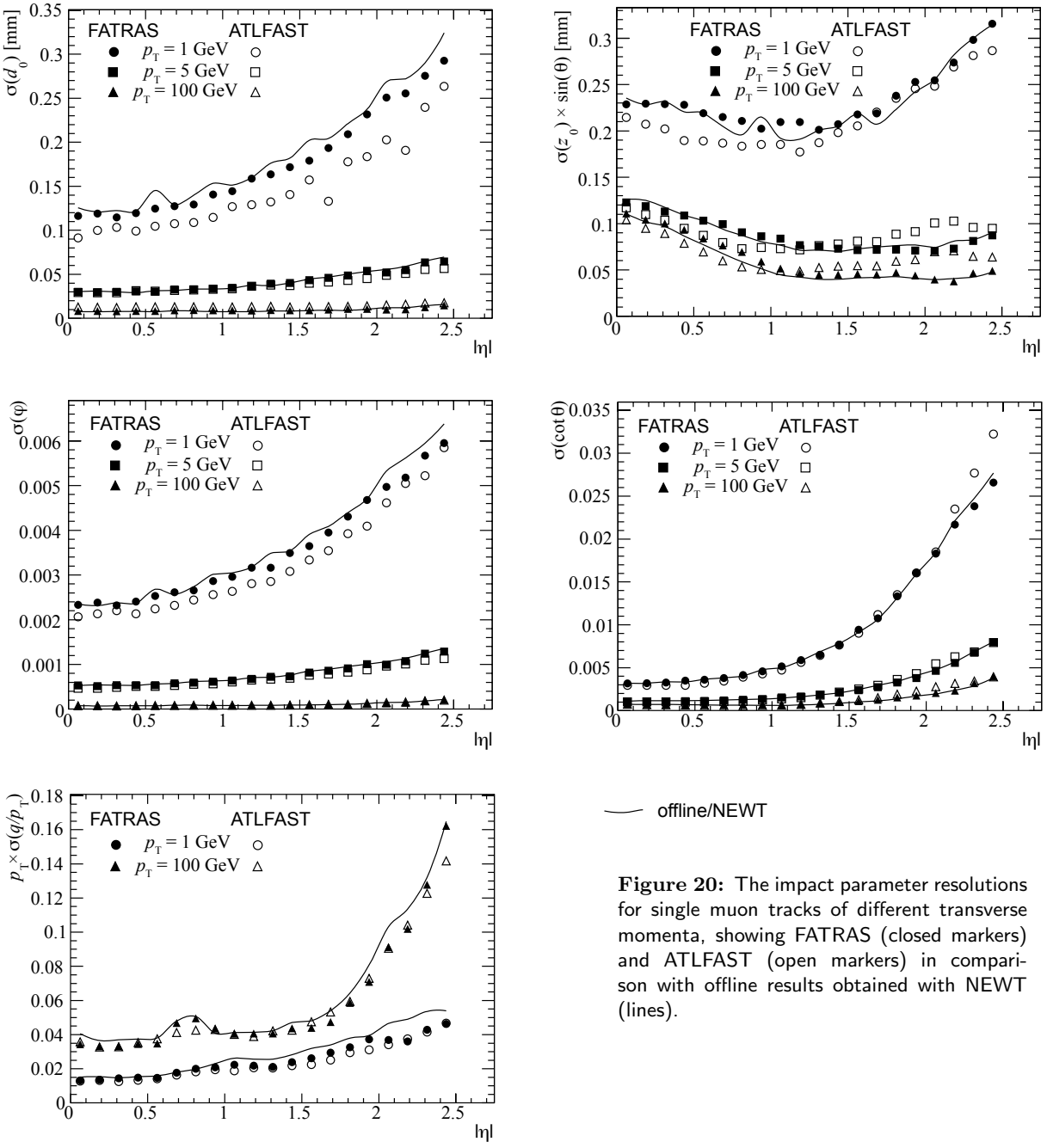


Figure 20: The impact parameter resolutions for single muon tracks of different transverse momenta, showing FATRAS (closed markers) and ATLFAST (open markers) in comparison with offline results obtained with NEWT (lines).

FATRAS shows a remarkable good agreement over the entire pseudorapidity range. The remaining differences are due to the simplified cluster creation and material effects model, but can be minimised with appropriate parameter tuning. ATLFAST follows the general trend of the offline results, but remains however at poorer level of compatibility.

Parameter Correlations One additional aspect of track reconstruction is that the track parameters are strongly correlated, i.e. an incorrect or uncertain estimation of one parameter usually corresponds to associated reconstruction errors of one or few other parameters. The correlations are hereby mainly given by the detector geometry and the magnetic field setup. In a cylindrical detector that is confined in a solenoidal magnetic field (such as the ID) the track trajectory is to a good extent described by a helical track model, which leads to four sets of strongly correlated track parameter pairs: the three transverse track parameter correlations (d_0, ϕ_0), $(d_0, q/p_T)$, $(\phi_0, q/p_T)$, and the correlation between the longitudinal parameters z_0 and θ. A correct description of the track parameter correlations is of particular interest for successive vertex fitting and b-tagging applications, since they rely on the covariance matrix of the track expression. Figure 21 shows a comparison of the four non-trivial track parameter correlations for single muon tracks with $p_T = 10$ GeV between offline, FATRAS and ATLFAST for the entire ATLAS ID. The cross-shaped correlation plot in the longitudinal sector is due to the overlap of barrel and endcap tracks that have different correlation parameters.

Since the correlations between the track parameters are mainly determined by the detector setup and magnetic field setup, it is not surprising that very similar results between FATRAS and the offline reconstruction can be achieved over the entire momentum range of reconstructed particles, see Fig. 22. The weakness of ATLFAST in this respect can probably be explained by several contributing aspects: on the one hand, the correlated smearing of the track parameters in the according ATLFAST module seems to be erroneous, in particular cases where correlation factors bigger than 1 could be identified. On the other hand, the parameterisations of the correlation factors are usually more difficult than for the according track residual parameterisations, since they usually do not follow a similarly trivial function as given through the $A \oplus B$ model[14].

The correlation factor $\rho(\tau_i, \tau_j)$ that relates the reconstruction uncertainty of the track parameter τ_i with the τ_j can be written as

$$\rho(\tau_i, \tau_j) = \frac{\mathrm{cov}(\tau_i, \tau_j)}{\sigma_i \sigma_j}, \tag{19}$$

when $\sigma_i^2 = \mathrm{cov}(\tau_i, \tau_i)$. A comparison of the obtained correlation factors for the single muon tracks over the momentum range of $p_T \in [1\ \mathrm{GeV}, 100\ \mathrm{GeV}]$ is presented in Fig. 22.

4.1.3 Single Electrons Samples

Electron reconstruction is usually more difficult than the reconstruction of muon or pion tracks. This is, on the one hand, mainly due to the contribution of bremsstrahlung to the energy loss of electrons, and on the other hand due to the fact that many electrons in the detector originate from photon conversions and have thus production vertices at large radii. While for the first aspect more elaborated fitting techniques can be applied, latter requires also different pattern recognition strategies[15]. The radiation loss results in a highly asymmetric energy loss distribution and the Gaussian approximation that is inert to most track fitting techniques becomes invalid to a large extend. The resulting *transverse* track parameter resolutions show therefore asymmetric tails and dedicated fitting techniques are often used to account for the non-Gaussian process noise[16]. Figure 23 shows a comparison of the transverse track parameter resolutions for single electron tracks obtained with offline, FATRAS and the ATLFAST simulation for low and high momentum single electron tracks. Both fast simulation techniques achieve asymmetric distributions for the transverse track parameters, but in particular for the momentum measurement FATRAS performs better than ATLFAST.

[14] Further investigations of the falsely calculated correlation parameters have already been started in the ATLFAST community [26].

[15] In ATLAS, a second outside-in reconstruction chain is dedicated to this purpose

[16] Any misinterpretation of the particle momentum — as long as it is within a similar regime for the multiple scattering inclusion — can only affect the track parameters that are directly affected by the momentum measurement. In the ATLAS ID, since the cylindrical geometry is aligned with a solenoidal magnetic field, the affected parameters are the so called *transverse* the parameters (d_0, ϕ) and the transverse component of the momentum p.

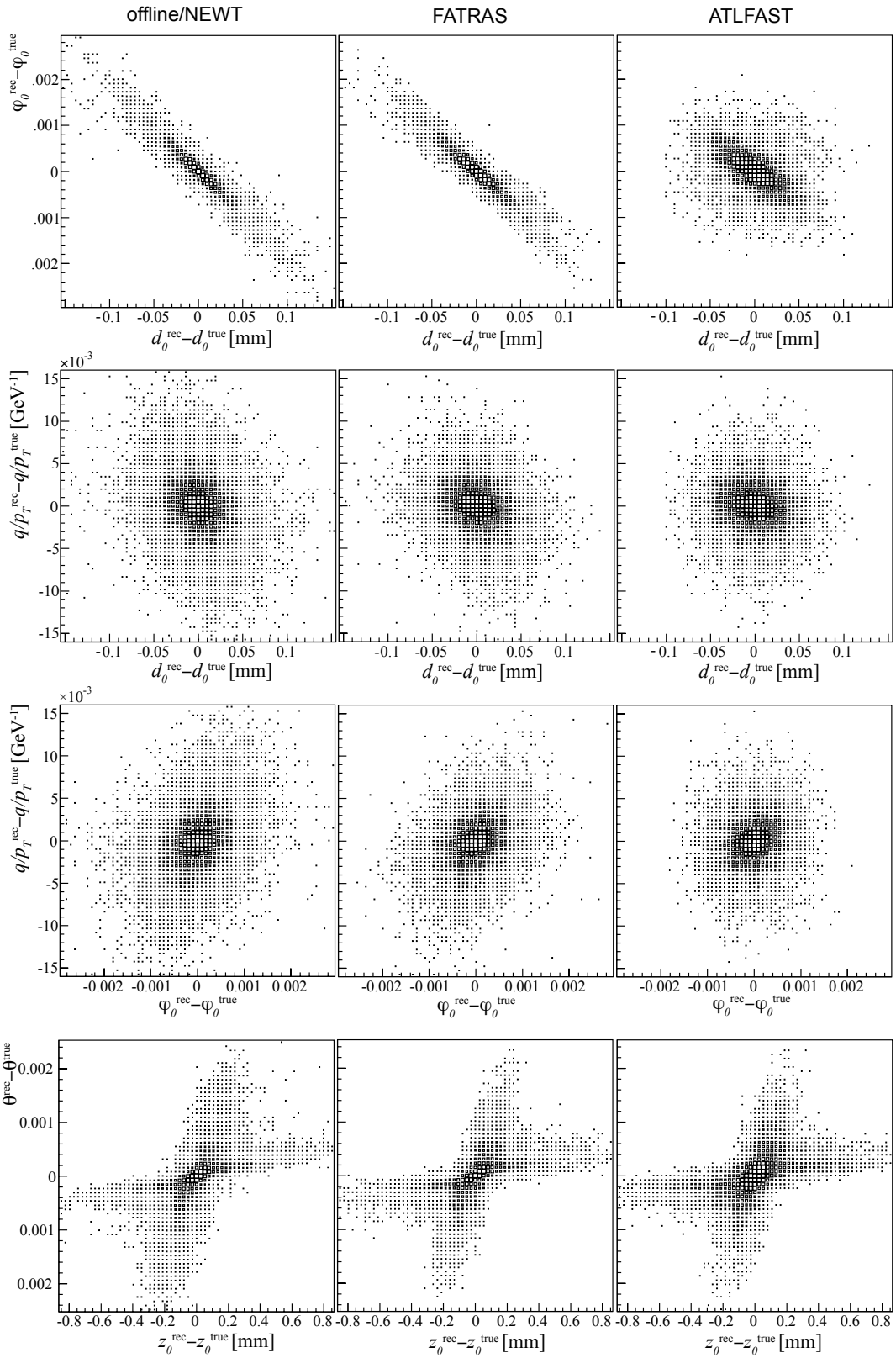


Figure 21: The four non-trivial track parameter correlations for single muon tracks of transverse momentum 10 GeV in comparison between offline, FATRAS and ATLFAST. The first column shows the correlation of the reconstruction errors for the offline reconstruction chain, the second for FATRAS in refit mode. The third column shows the according results obtained with ATLFAST that indicate a problem in the correlated smearing of the transverse track parameters.

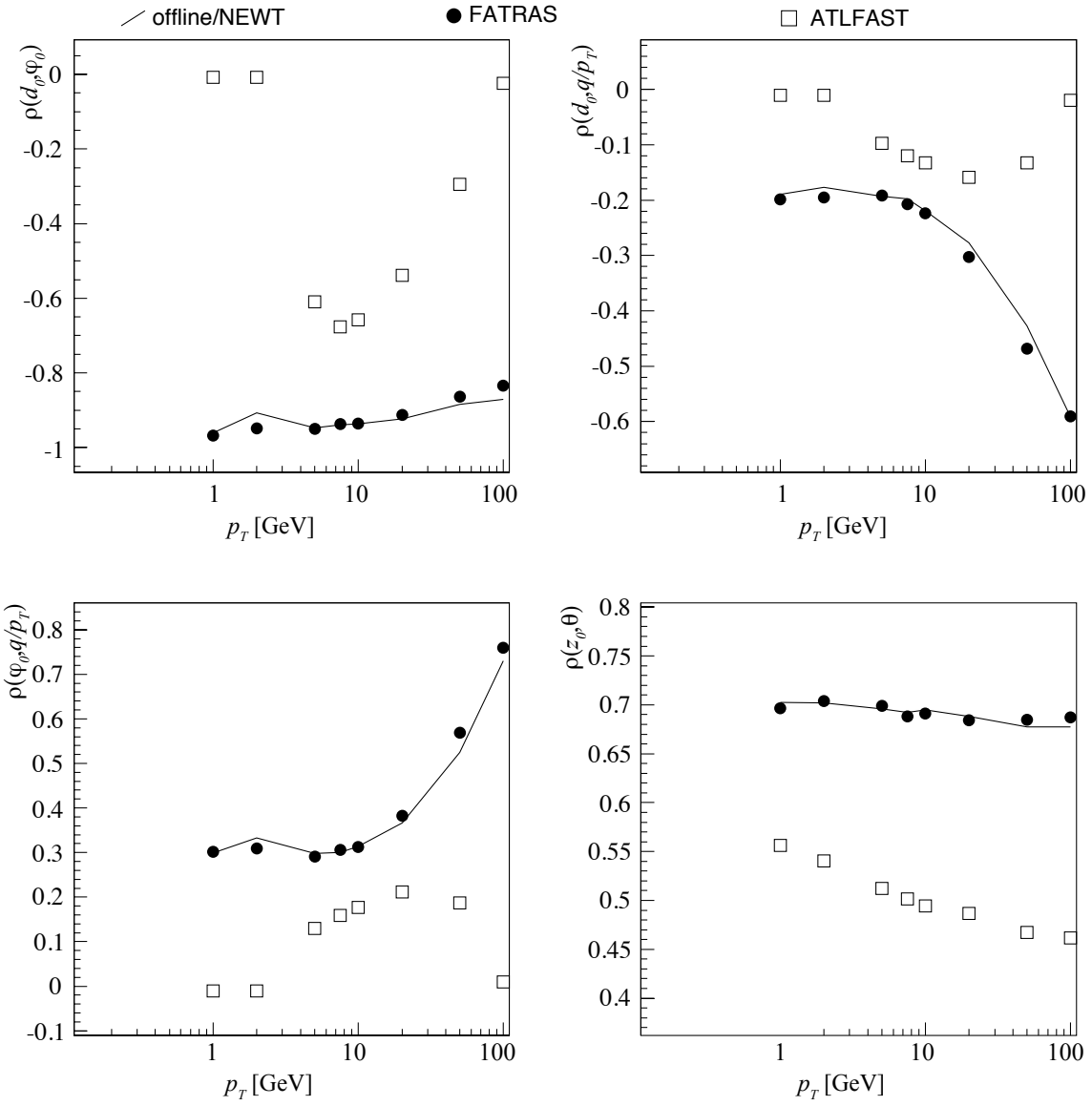


Figure 22: Correlation parameters in dependency of the transverse momentum p_T obtained for the four non-trivial track parameter pairs from offline reconstruction, ATLFAST and the FATRAS simulation.

A striking example for for the additional potential of FATRAS — in particular when comparing it with the ATLFAST track simulation is the application of dedicated electron fitting. The track fit of electron trajectories is somewhat difficult, since due to bremsstrahlung the Gaussian assumption of the included transport error caused by material interactions is not valid. Several dedicated fitting techniques exist in ATLAS that allow to include these effects in the track fit, amongst them a Gaussian Sum Filter (GSF) and a newly developed extension of the Kalman Filter that integrates a dynamic noise adjustment schema (DNA). Both fitting techniques can be applied to tracks that originate from the FATRAS simulation and show a remarkable similarity in comparison to results obtained with the full offline chain, see Fig. 24. FATRAS can thus not only be used for a very refined event analysis, but is also very powerful in helping to develop dedicated techniques for electron fitting[17].

[17] In particular for the GSF FATRAS has extensively used to validate the various component models, but also for achieving stability on large event samples.

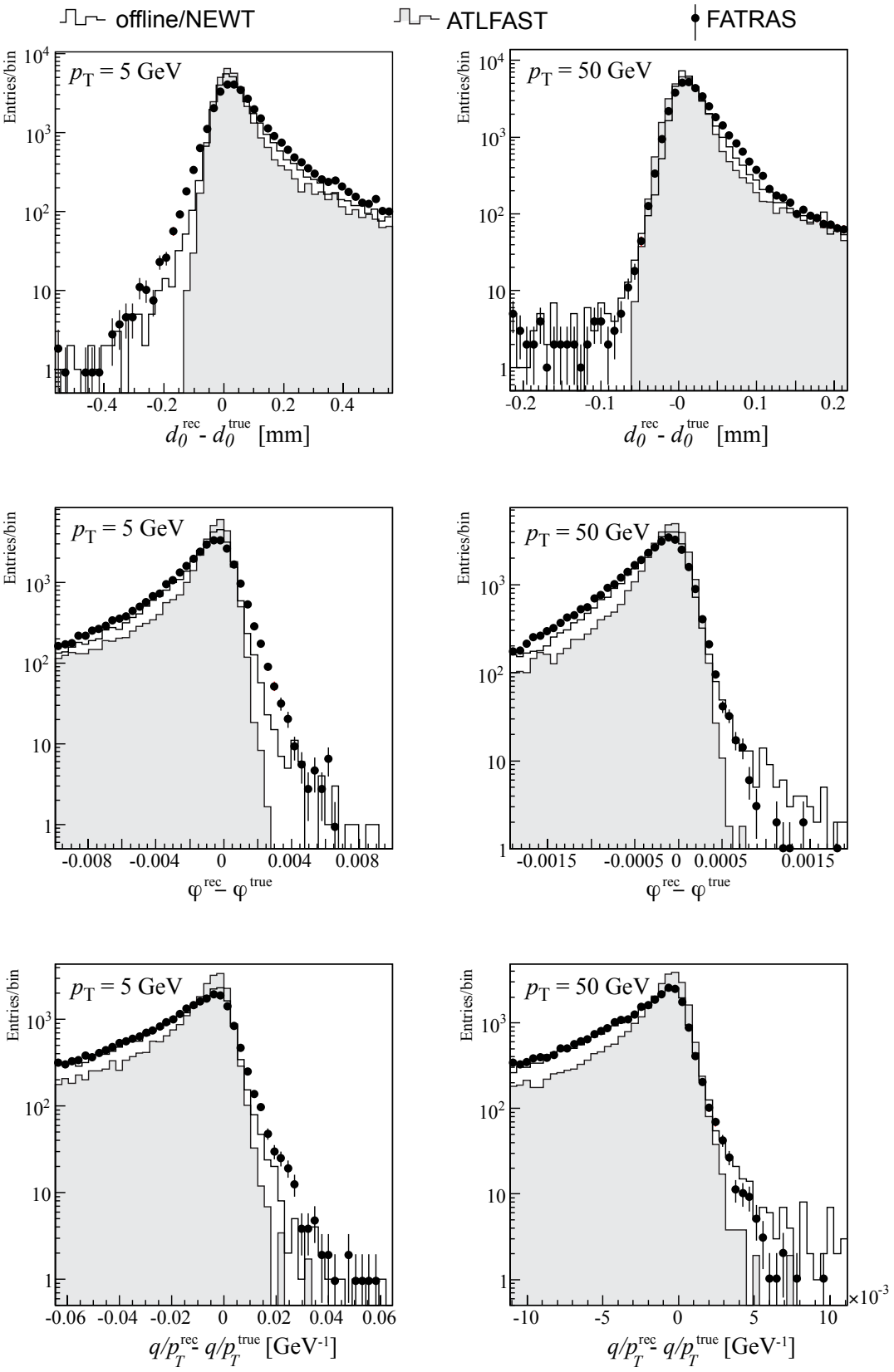


Figure 23: Transverse track parameter resolutions for the low and high (transverse) momentum electron tracks in the ATLAS Inner Detector for the full offline chain, ATLFAST and FATRAS.

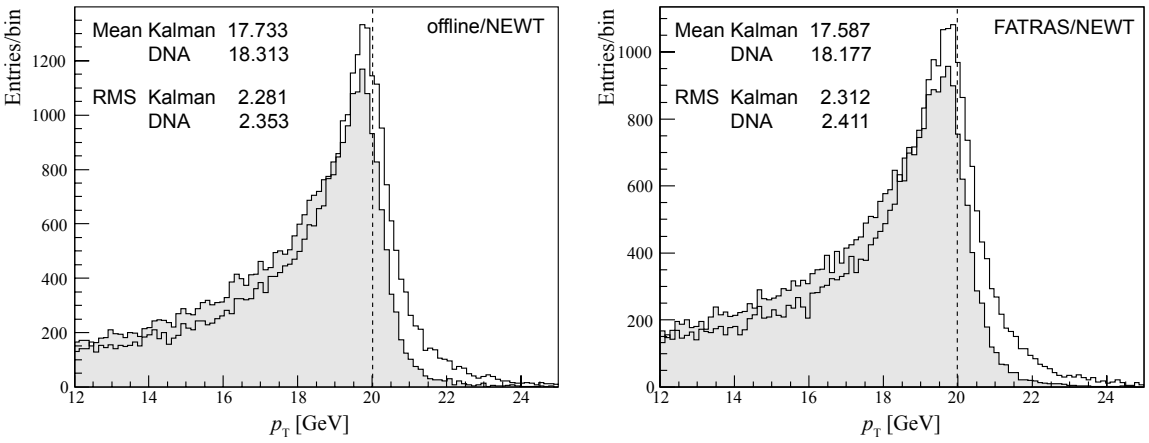


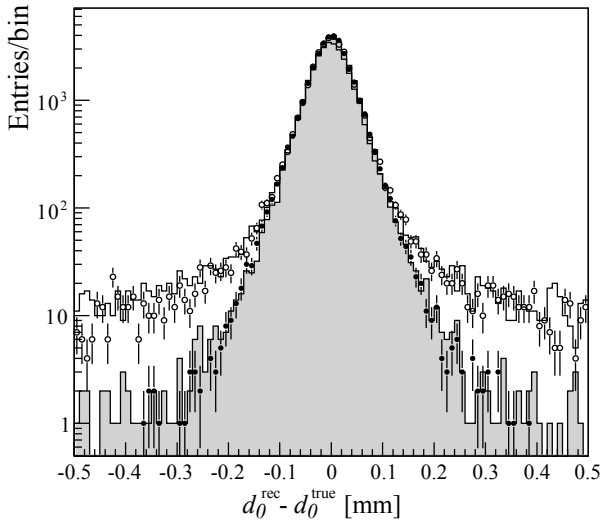
Figure 24: Single electron tracks with transverse momentum of $p_T = 20$ GeV simulated with Geant4 and FATRASand reconstructed with NEWT. In both cases, the initial track collection that has been fitted with a standard Kalman filter has been refitted with an extend Kalman fitter version, that includes dedicated dynamic noise adjustment schema for regulating large energy loss contributions (and resulting directional changes) due to bremsstrahlung effect.

4.1.4 Single Pion Samples

Pions are — similar to muons — mainly influenced by multiple scattering and ionisation loss, but also hadronic or nuclear interactions can occur with the detector material. The correct description of hadronic interactions (in particular of pions) with the detector material is an immense field of interest for the LHC simulation engines [27]. This is mainly because hadronic showers are amongst the most prominent final states to be expected with from proton-proton interactions and in particular for the simulation of the hadronic calorimetry devices the understanding of the shower developments and energy distributions is necessary. In terms of track reconstruction, a naive approach can be taken: if the pion does not undergo substantial hadronic interactions, the particle can be regarded as a minimum ionising particle (MIP) and leads to very similar track resolution as a muon. On the other hand, most of the nuclear interactions disturb the original pion in such a way that it is very unlikely to be found by the pattern recognition and thus, the particle is *lost* for the further event analysis. Latter is reflected in the overall reconstruction efficiency and will be briefly discussed in Sec. 4.2. One other aspect of pion tracks is that they often have production vertices far away from the nominal interaction point; for the ATLFAST simulation this required a special binning of the smearing functions for different vertex radii. Since FATRAS starts the track simulation at the production vertex, effects on the track resolutions originating from fewer precision hits are automatically included.

The additional hadronic interactions with the nuclei of the detector material also disturb the track parameter resolutions and result in bigger uncertainties of the impact parameter resolutions. While the core distribution remains to be similar to the ones obtained with muon tracks, the tail distribution contributes significantly; this is either through effectively shortened track lengths in case of the hadronic interactions stopping the original pion track within the detector, os by secondary shower particles that are misidentified as prompt tracks. Figure 25 shows the impact of hadronic interactions on the estimation of the transverse impact parameter d_0 in comparison for single muon and pions tracks. The agreement of FATRAS and the full offline results is hereby remarkable, although the absolute number of secondary tracks that are found by the reconstruction of FATRAS simulated events is slightly overestimated. Again, an appropriate tuning of the hadronic interaction probability is still missing in FATRAS[18]; a final integration of the nuclear interaction length to the reconstruction geometry description will hopefully account for many of these effects.

[18] In the current example, the probability of a nuclear interaction based on Eq. (9) has been scaled by an additional factor of 2.



	FATRAS	offline/NEWT
Muons		
Entries	33814	33707
RMS	0.0446 mm	0.0469 mm
Pions		
Entries	37146	34539
RMS	0.070 mm	0.076 mm

Figure 25: The transverse impact parameter resolution d_0 for single muon and pion tracks of transverse momentum 5 GeV. The results of FATRAS and the full offline chain are shown without applying any quality or matching cuts to include secondary tracks caused by hadronic shower interactions.

4.2 Track Reconstruction Efficiencies

Both aspects of track reconstruction — the pattern recognition and the track resolutions — contribute to the final event typology. A dedicated aim of FATRAS is to reproduce similar reconstruction efficiencies than obtained with the full offline chain. For electrons and muons this goal is almost automatically achieved by the fact that FATRAS integrates the standard offline pattern recognition modules. A comparison of the reconstruction efficiencies for muons and pions is between offline and FATRAS can be seen in Fig. 26.

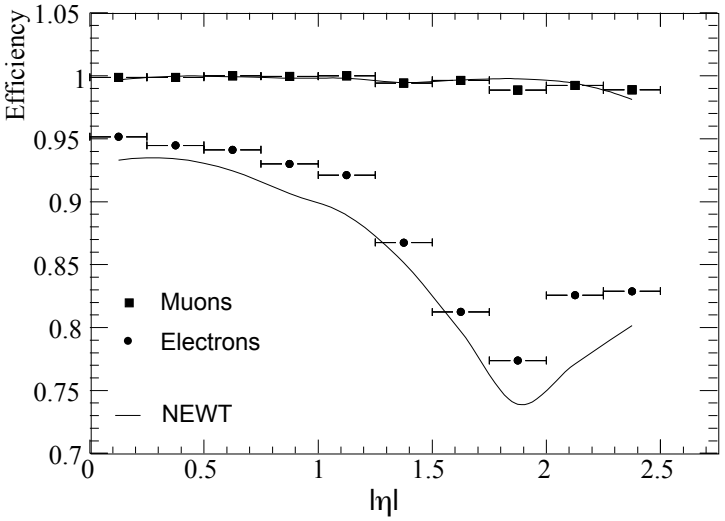


Figure 26: Comparison of reconstruction efficiencies for single electron and muon tracks with transverse momentum of $p_T = 5$ GeV that have been simulated with Geant4 and FATRAS. In both cases, the identical reconstruction chain — the ID New Tracking — has been performed.

For pions (and other hadrons), however, the situation is more difficult since the integration of hadronic interactions is currently limited in FATRAS. This is mainly due to the fact that the nuclear interaction length is not available from the `TrackingGeometry` that has been designed for reconstruction purpose. Currently, the nuclear interaction length λ is approximated by the use of the radiation length, the average atomic number Z, and a global scale factor, see Eq. (9). Figure 27 shows the reconstruction efficiencies of pion tracks, when applying two different global scale numbers in comparison with results from the offline reconstruction. While the general features and characteristics of the η dependence can be reproduced, the radiation length remains only an approximate description of the nuclear interaction length and a global scaling parameter that is valid over the entire pseudorapidity range can not be defined. To conclude the full integration of the hadronic interactions in FATRAS an

updated reconstruction geometry model is inevitable. This adaption is planned to be integrated after the major ATLAS release 14.0.0.

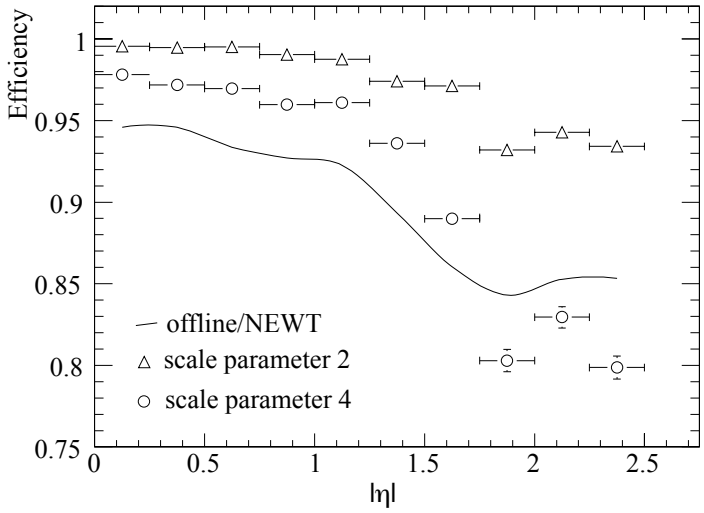


Figure 27: Comparison of reconstruction efficiencies for pion tracks with transverse momentum of $p_T = 5$ GeV that have been simulated with Geant4 and FATRAS. In both cases, the identical reconstruction chain — the ID New Tracking — has been performed.

4.3 CPU Time Consumption

Large simulated event samples are needed for many analyses with the ATLAS detector; the CPU time consumption used for the simulation and reconstruction of these events has become a stringent constraint, not only in a financial manner. Providing a fast alternative to the time consuming full detector simulation was one of the motivations of the FATRAS project. The challenging task of every fast simulation strategy is to find a good balance between the level of accuracy that can be achieved and the computing complexity that goes along with it. It is of little surprise that the ATLFAST simulation, that only performs a smearing of the kinematic variables is the by order of magnitudes fastest track simulation that is currently deployed, but has, on the other hand, the drawback that many aspects of the event reconstruction are not considered at all.

The event reconstruction, which usually only deals with a subset of the simulated information in a mainly probabilistic manner is usually by orders of magnitudes faster than the event simulation. To gain maximum compatibility between FATRAS and the offline chain, it was a necessary assessment that in FATRAS the full reconstruction part is performed. Since the simulation part also relies mainly on offline reconstruction algorithms, a similar execution time as for the event reconstruction can be achieved. In fact, since the pattern recognition imposes a high number of combinatorial track candidates that have to be processed, the pure simulation module of FATRAS is for full physics samples even faster than the event reconstruction.

The following timing tests have been obtained for repeated simulation of 100 inclusive $t\bar{t}$ events using ATLFAST, FATRAS and the full simulation. The event simulation has been hereby entirely limited to the Inner Detector and results are given in normalised CPU seconds (KSI2K) [28].

Table 3: Total simulation (and digitisation) time per inclusive $t\bar{t}$ event for the full Inner Detector, FATRAS and the ATLFAST track maker modules given in normalised CPU seconds.

Simulation Type	**Average Total Time per Event [KSI2K]**
Geant4/offline	$\approx 146\ (sim) + 4.3\ (digit) \pm 4.5$
FATRAS	$\approx 2.78\ (total) \pm 0.33$
ATLFAST	$\approx 0.0226\ (total)$

Table 3 summarises the normalised CPU time per event for the event simulation and digitisation. For ATLFAST, since simulation, digitisation and reconstruction can not be separated, the total time for the ATLFAST tracking is given; the contribution of event detector simulation and digitisation for

the full simulation chain are given individually. The FATRAS timing contribution does not contain the track refitting, a more conclusive comparison between the processing time of FATRAS for refit or reconstruction mode is given in the following section, Sec. 4.3.1. It is demonstrated that the total simulation time spent in the FATRAS modules is lower than the digitisation step of the full offline chain. It is not very surprising, that the ATLFAST simulation has by far the lowest execution time, since only few random number samplings and trivial mathematical operations are performed to smear the generated particle parameters. For FATRAS, a factor of 50 can be easily gained without compromising the event characteristics. The overall computing time of FATRAS is dominated by the transport time of low energetic particles through the detector setup; In the default configuration, the following restrictions are applied:

- particles with $p_T < 250$ MeV are not used for track simulation
- particles with $E < 250$ MeV are not processed in the particle decay
- the track simulation is stopped, when the particle falls under an energy threshold of 50 MeV
- photons with an energy of less than 150 MeV are not further processed in the conversion module

it can be shown that by raising these threshold numbers, an event processing rate per second that is 100 times higher than the full simulation can be achieved. This optimisation, however, should be carried out in careful cooperation with the physics communities to achieve an optimal balance between the physics performance of the simulation module and the computing complexity.

4.3.1 Component Time Consumption

In the full simulation chain, there is a clear separation between the the actual detector simulation and the digitisation module. In fact, most of the time these tasks are even performed in different jobs. In FATRAS, the digitisation is performed during the simulation process, hence it is not possible to give timing figures in such a classical frame. However, the time contribution of the single FATRAS modules — described in Sec. 2 — can be measured independently and are presented with additional comments in Tab. 4.

Table 4: The execution time for the single FATRAS simulation modules and their contribution to the overall event processing time. For the FATRAS refit mode, the additional time that has to be spent in the refitting module is listed below. The time spent in the NEWT reconstruction algorithms is given for data simulated with FATRAS.

Module	**NSI2K**	Av. time [ms]	Min time [ms]	Max time [ms]
`FatrasSecondaryPhotonProcessing`	-	3 (± 1.8)	0	19
`FatrasSecondaryParticleDecay`	-	4 (± 2.5)	0	19
`FatrasCopyMcCollection`	-	6 (± 2.7)	1	18
`FatrasTertiaryTrackCreation`	0.04	19 (± 19)	0	138
`FatrasGenEventSimulation`	0.08	35 (± 19)	8	106
`FatrasPhotonProcessing`	0.11	48 (± 23)	8	164
`FatrasParticleDecay`	0.18	79 (± 58)	1	403
`FatrasHitPostProcessing`	0.39	171 (± 20)	109	285
`FatrasSecondaryTrackCreation`	0.64	282 (± 127)	36	866
`FatrasPrimaryTrackCreation`	1.36	604 (± 295)	109	2150
`FatrasTrackRefitting`	1.52	697 (± 344)	64	3202
NEWT: inside-out sequence	1.50	663 (± 135)	-	-
NEWT: outside-in sequence	0.24	108 (± 38)	-	-
NEWT: 2^{nd} stage pattern	0.56	250 (± 162)	-	-

Table 4 includes the event reconstruction for the reconstruction of FATRAS simulated events in the ID. These numbers are divided into the two main sequences, the inside-out and the outside-in track

finding, respectively, and separately for the second stage pattern that includes vertex reconstruction, conversion finding and the particle candidate creation.

The large spread between minimum and maximum time spent in the track creation and refit modules is given by different event morphologies: when many low momentum tracks are being processed (such as being caused by hadronic shower interactions or multiple decay products), a huge increase in the CPU time needed for this event is observed. This is, because the track extrapolation gets immediately slower due to a rapid increase of needed steps in the numerical integration of the magnetic field. This effect can, of course, be cancelled or regulated with appropriate minimum momentum cuts. In the reconstruction modules, most of these low p_T tracks are rejected by the pattern recognition modules that have intrinsic cuts for a minimum particle momentum[19].

5 FATRAS Applications

Besides playing an important role in the validation of the track reconstruction software, FATRAS has so far also been used for applications that could not have been carried out easily with the the full simulation chain. One of which has been a large scale study of the pattern recognition stability in the TRT detector, when increasing the noise (or occupancy) level far beyond 50 %. A small review of the original study [29] is given in the next section, Sec. 5.1.

FATRAS is also used for first upgrade studies of the ATLAS detector that are — amongst other reasons — not possible with the full detector simulation, since no pattern recognition exists for the modified detector layouts. In FATRAS, however, the pattern recognition can be by-passed and first track parameter resolutions or track and hit multiplicity studies could be achieved. A more detailed description of the SLHC studies that have been performed with FATRAS will be presented in the following, Sec. 5.2.

5.1 Occupancy Study in the ATLAS TRT Detector

FATRAS was used to study the influence of the noise occupancy in the ATLAS TRT on the quality of pattern recognition and track fits. In the the ATLAS offline track reconstruction NEWT, tracks from the silicon layers are extended into the TRT by a dedicated reconstruction module. TRT measurements are collected around a cone seeded by the silicon track and local pattern recognition algorithms are used to find the correct measurement to track assignment. Additionally, the left/right ambiguity in each TRT measurement has to be solved[20]. Currently two algorithms exist for this purpose, a Kalman filter based approach and the *Deterministic Annealing Filter* (DAF). A more detailed description of both concepts can be found in [11].

The noise occupancy in the TRT can easily be set to arbitrary values in FATRAS. Noise measurements mask measurements coming from tracks in the noise simulation, such that the track measurements are not visible anymore for the pattern recognition. Therefore the pattern recognition has to cope with additional noise measurements as well as wrong measurements on the particle‘s trajectory at higher noise levels. The noise occupancy in the Pixel and SCT detector was kept at the nominal value during this study, because additional noise measurements in those detectors just increase the number of fake seeds, but do not deteriorate the TRT extended track fit significantly up to reasonable noise levels[21].

The truth matching from the FATRAS validation packages was used to calculate the residuals between simulated track parameters and reconstructed track parameters. A Gaussian fit was performed to the resulting distributions for each noise level and its width is quoted as the resolution of the track parameter. 5000 single muon events with a momentum of 5 GeV were simulated in the TRT barrel range $|\eta| < 0.5$. Figure 28 shows a small example of the mentioned study, that is laid out in more detail in [29]; the left plot shows the uncertainty of the transverse momentum estimation in including

[19] The NEWT reconstruction has, however, a dedicated mode for searching low momentum tracks. This has been designed for early data and minimum bias events. Currently, the combination of this reconstruction mode with FATRAS has not been tested.

[20] This is, because the TRT is a drift time detector, only a circle around the central drift tube is given as the *measurement*.

[21] The reconstruction breaks down due to the enormous number of seeds from combinatorics before the overall performance of the track fit degrades

the TRT barrel measurements in absence of any noise contribution, while the right plot shows the degradation of the momentum resolution with increasing noise level and compares hereby the standard Kalman fitter to the DAF extension.

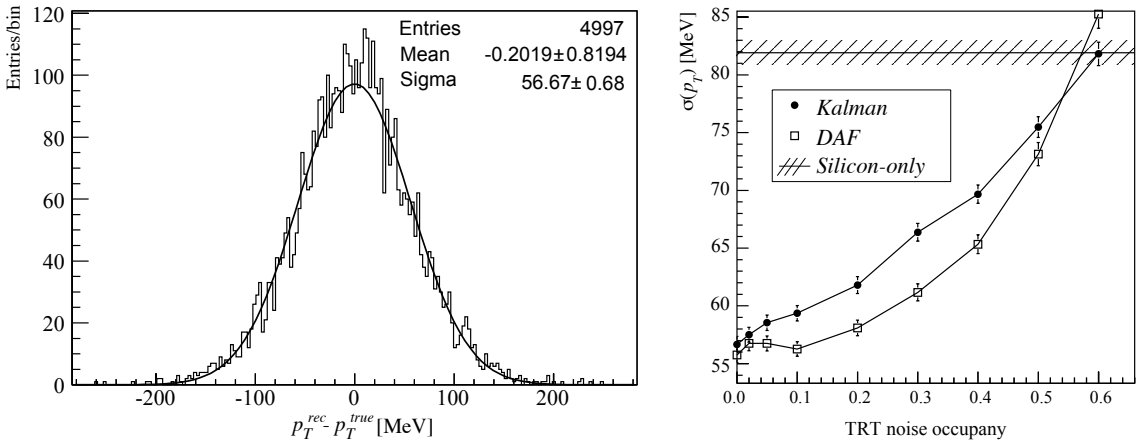


Figure 28: An example application of the FATRAS simulation in a TRT noise level study, comparing the standard Kalman filter to the Deterministic Annealing Filter; the transverse momentum resolution of 5 GeV single muon tracks is shown in the left plot, while the degradation of the momentum resolution with increasing noise level is presented to the right.

FATRAS allows to do a very detailed comparison between simulated and reconstructed tracks. In the noise study this feature was used to investigate how many noise measurements are wrongly assigned to a track. It could even be studied how often the solution of the left/right ambiguity was correct. Such a detailed study is not possible using the full simulation, since the truth information is not sufficiently available there.

5.2 Detector Upgrade Simulations

The preparation of an experiment that is of a comparable size as the ATLAS experiment spans over a long period in time and includes many studies based on Monte Carlo simulation to optimise the detector setup and to estimate the detector performance. Already during the startup phase of the ATLAS experiment, first studies are carried out to simulate different detector setups foreseen to be integrated in the first upgrade phase that is scheduled to take place after about ten years of operation. The detector upgrade of the ATLAS detector will be evoked and accomplished by an according upgrade of the LHC machine — to become the *Super Large Hadron Collider* (SLHC) — with an increased peak luminosity of $10^{35} cm^{-2} s^{-1}$. The increased collision rates, a direct consequence of the higher luminosity, require higher granulated detector devices, in particular for the TRT detector it will become almost impossible to resolve the high track density (although, Sec. 5.1 has shown, that with the use of new pattern recognition techniques, a significantly higher track multiplicity than in the ATLAS startup setup can be handled.). Several layout proposals exist in the meanwhile [30] to exchange the existing ATLAS ID detector with a silicon-only detector structure. For a final decision, however, it is important to test these proposed layouts in both feasibility and performance.

The description of such detector setups is a first requirement for the establishment of a detector simulation. In FATRAS, since the offline reconstruction geometry is used as the simulation geometry, this can realised through a dedicated `TrackingGeometry` description based on generic input parameters. FATRAS provides for this purpose generic detector builders that allow a flexible choice of cylinder and disc detector layouts with both pixel and strip detector technologies. Figure 29 shows a picture of the 3D `TrackingGeometry` for one proposed SLHC layout, illustrated with the ATLAS event display VP1. Custom clusterisation algorithms — e.g. needed for newly established developments such as the 3D-pixel detector — can be included and since the ATLAS offline EDM has been extended with appropriate generic hit and cluster classes, track simulation and refit can be performed very similarly to the ATLAS offline setup.

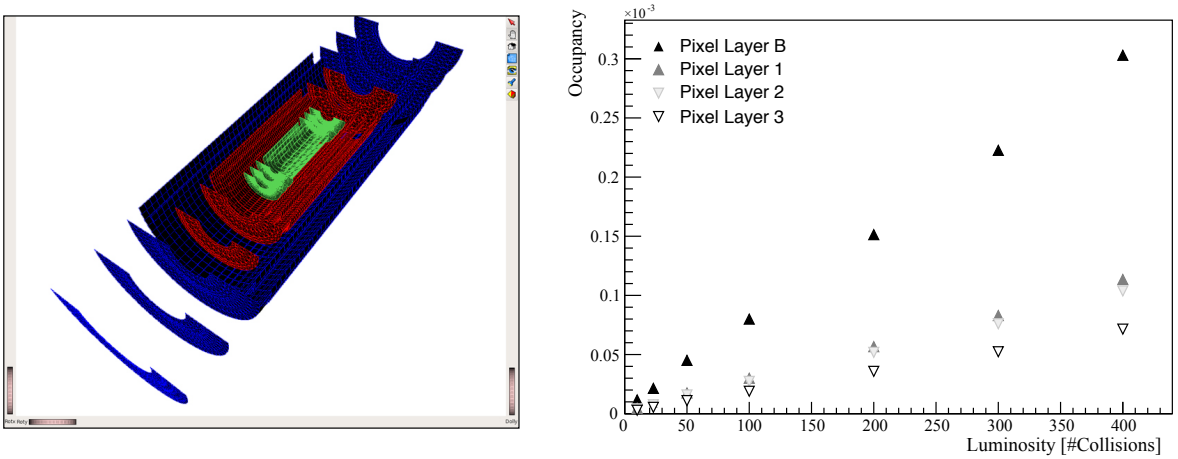


Figure 29: Left: A picture of one proposed SLHC geometry for the ATLAS ID produced with the ATLAS event display VP1. The upgrade scenario for the ID foresees the complete exchange of the ID tracker with a purely silicon based pixel vertex and strip detector in 2015. A framework for a flexible change between different layouts has been implemented in FATRAS to ease the comparison of various upgrade scenarios. Right: Example for a hit occupancy study on the pixel layers of the four pixel barrel cylinders depending on the number of pile-up events.

6 Conclusion and Outlook

We have presented the concepts, implementation and first performance figures of the new ATLAS fast track simulation application FATRAS. FATRAS is a full Monte Carlo simulation that is based on the reconstruction geometry and offline reconstruction tools and establishes a third simulation technique for ATLAS alongside to the full detector simulation and the parametric smearing approach. It agrees to a large extend with results obtained by full simulation and offline reconstruction, while using only a fraction of CPU time. The FATRAS simulation could be established to yield event processing rates of at the level of 1 Hz; together with the consecutive track reconstruction, a total event processing time of about 2.5 seconds for the ATLAS ID could be achieved.

Good agreement with the full offline chain could be met for track parameter resolutions, hit and track multiplicities but also for reconstruction efficiencies of muon and electron tracks. Latter is achieved by simply using FATRAS as an input for the standard offline track reconstruction software. FATRAS combines the possibility to use very detailed event analysis techniques with fast execution times and is thus a perfect tool for the development of a new tracking based analysis before applying it on full simulated or taken data.

6.1 Muon System and Calorimeter Fast Shower Simulation

FATRAS is in the current state limited to the the Inner Detector (and its SLHC upgrade version), mainly for the fact that a purely predictive navigation in the detector geometry that is needed for the trajectory creation has only been exiting for ID. Recently [31], first attempts have been carried out to use the more complex `TrackingGeometry` of the Muon System for a similar trajectory creation approach that is also based on the extrapolation engine. These promising results open the window to a future full fast track simulation in the ATLAS detector, since many of the existing FATRAS modules, such as the material effects integration and the clusterisation can be to a large extent re-used for the Muons System[22].

FATRAS establishes a fast alternative to the full simulation, but the time spent in tracking detectors contributes only partially to the overall CPU time consumption of the detector simulation. The simulation of the calorimetry response is still the largest fraction of all sub-detectors. This is, because

[22] In the most optimistic scenario, the integration of the Muon System requires only the creation of a dedicated track creation tool that extends the FATRAS `ITrackCreator` interface.

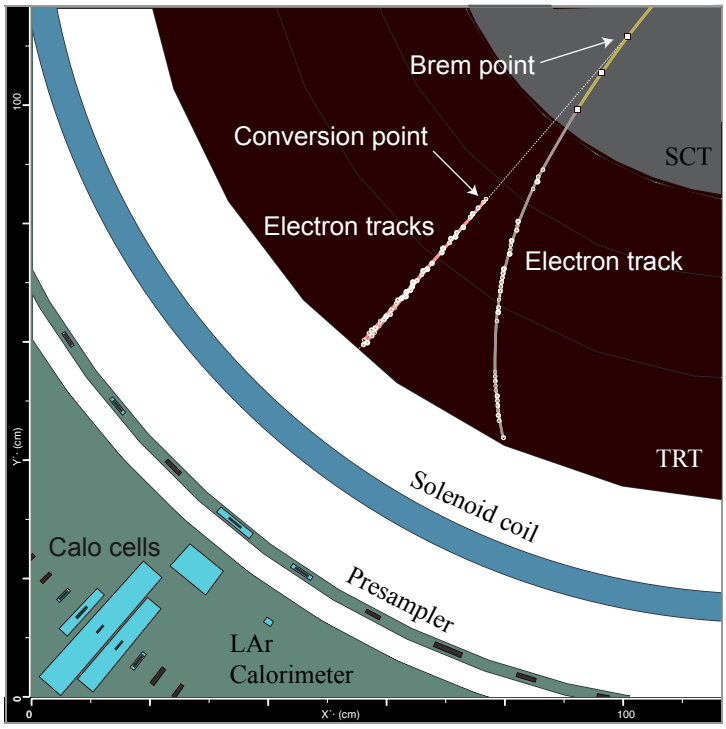


Figure 30: A detail of a simulated event shown with ATLANTIS using FATRAS for the Inner Detector simulation and the `FastCaloSim` for the simulation of the calorimetric response. The picture shows an electron track that undergoes an emission of a high energetic photon that converts into an almost collinear electron-positron pair in the TRT detector. The main part of the initial electron energy is hereby carried away by the brem photon (and then the conversion products), which leads to the cluster measurement in the electromagnetic calorimeter.

almost every particle causes shower reactions in the calorimeter and the number of particles to be tracked is multiplied by orders of magnitudes. In addition, the material distribution in the calorimeter is far denser than in tracking devices where a discrete detector setup of alternating detection layers and air- or gas-filled volumes is predominately chosen. Besides the constant optimisation of the Geant4 based simulation, recent work [32] has focussed on creating parameterised shower models that allow fast shower simulations on basis of the kinematic properties and the type of the particle. FATRAS has been expanded with the full Monte Carlo truth tree and by a dedicated module — see Sec. 3.6 — to prepare the track simulation output for a successive use in fast shower simulations. Figure 30 shows an ATLANTIS event display detail of an event that was simulated after interfaces the ID FATRAS simulation with the fast shower parameterisation[23].

The inclusion of the fast calorimeter simulation enables standard jet reconstruction and b-tagging algorithms to be performed on the combined output. A next step is thus to validate this extended usage in future event analyses. In a recent effort, a complete truth tree with an identical structure to the one produced by the full detector simulation has been established in FATRAS. This allows standard truth matching algorithms to work on FATRAS output and encapsulates the user analyses finally from the underlying used simulation technique.

A fully operational and validated FATRAS setup that includes first user feedback and a detailed parameter tuning is expected for the ATLAS offline release 14.0.0.

A Appendix

A.1 Typesetting and Nomenclature

The following type setting conventions are followed throughout this document: Software packages within the ATLAS offline software repository [33] are written in Sans-serif face, C++ or python class names are written in `Courier` face. Namespace definitions as used in the software repository are omitted in this document for readability. A exhaustive list of software packages and their location

[23] FastCaloSim

within the ATLAS software repository can be found in Sec. A.6.

A.2 Particle Decay Table

As described in Sec. 3.2, FATRAS offers two different ways to include particle decay processes in the track simulation: a simplified internal decay-model that supports only a few dedicated decay channels and the (default) full decay module that wraps the appropriate decay engines of Geant4. Latter is to a full extent described in [23]; for completeness, the supported decay modes and branching ratios of the simplified version is given in Tab. 5.

Table 5: Supported decay channels and branching ratios of the simplified FATRAS decay module.

Particle	Branching Ratio	Decay Channel
π_0	0.98798	$\pi_0 \to \gamma + \gamma$
	0.01198	$\pi_0 \to e^- + e^+ + \gamma$
$K^\pm$	0.63440	$K^\pm \to \mu^\pm \nu_\mu$
	0.20920	$K^\pm \to \pi^\pm \pi_0$
	0.05590	$K^\pm \to \pi^\pm \pi^\mp \pi^\pm$
	0.04980	$K^\pm \to \pi_0 e^\pm \nu_e$
	0.03320	$K^\pm \to \pi_0 \mu^\pm \nu_\mu$
	0.01757	$K^\pm \to \pi^\pm \pi_0 \pi_0$
K_S^0	0.69200	$K_S^0 \to \pi_0 \pi_0$
	0.30690	$K_S^0 \to \pi^+ \pi^-$
K_L^0	0.40530	$K_L^0 \to \pi^\pm e^\mp \nu_e$
	0.27020	$K_L^0 \to \pi^\pm \mu^\mp \nu_\mu$
	0.19560	$K_L^0 \to \pi_0 \pi_0 \pi_0$
	0.12560	$K_L^0 \to \pi^\pm \pi^\mp \pi_0$

A.3 Additional Iteration on Conversion and Brem Photon Handling

In the initial configuration, FATRAS has been limited to one iteration of photon transport through the detector and its associated conversion creation. This lead to an underestimation of low energetic photons and electron/positron tracks. Although it is not very likely that these tracks are found by the main reconstruction sequence, they are still of interest since they create patterns that are found by the second reconstruction sequence, the so-called backtracking, or simply add traces that may disturb the pattern recognition process. It could be shown that the inclusion of one additional iteration in FATRAS improves the compatibility with full simulated results significantly, while adding only a tiny addition timing fraction to the overall simulation time. Figure 31 shows a momentum comparison of radiated brem photons in FATRAS with two different Geant4 configurations: the default Geant4 setup (blank histogram) and the artificially cut configuration that stops just as FATRAS after one iteration. The remarkable good agreement of FATRAS with the also restricted Geant4 results proves the quality of the bremsstrahlung model in FATRAS alongside with the accurate material description of the reconstruction geometry.

A.4 Little User Guide and Steering

The FATRAS steering is in full coherence with the overall ATHENA job configuration schema that has been in the last year adapted to a modern job configuration framework based on python [34]. Single components of the actual reconstruction flow are entirely configured through auto-generated python configuration classes, while parameters and overall steering flags are concentrated in so-called *JobProperty* classes. The FATRAS `JobProperty` classes can be found in the FatrasExample/python directory and are listed — including some brief description — in Tab. 6.

The single property flags can be modified by the user to change the (default) behavior of the FATRAS simulation process; they include overall simulation steering (such as e.g. the run mode, the detector

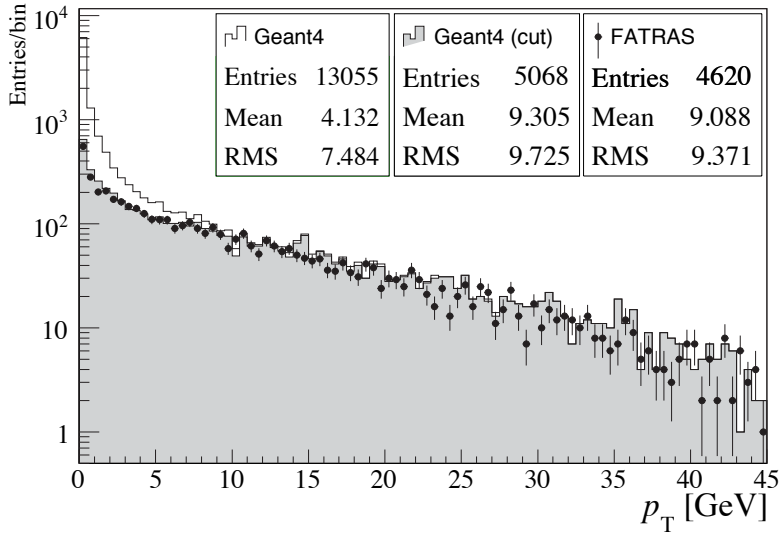


Figure 31: Photon momenta of radiated brem photons simulated with FATRAS and Geant4. The blank histogram shows the standard Geant4 setup, while the shaded area shows the Geant4 results when being restricted to one single photon/conversion iteration (i.e. the identical setup to FATRAS in this comparison).

Table 6: `JobProperty` python classes for the FATRAS job configuration.

JobProperty	**Description**
`FatrasJobProperties`	general FATRAS steering: modes, geometry, auditors
`FatrasKeys`	names of FATRAS collections in the transient event store
`FatrasSingleTrackSimulation`	dedicated steering for single track simulation
`FatrasPhysicsList`	switches for physics processes
`FatrasClusterCreation`	tuning parameters for hit creation and clustering
`FatrasTuning`	remaining tuning parameters: e.g. minimum energy cuts, probability scaling for physics processes
`FatrasValidation`	definition of validation modules
`FatrasOutputLevels`	screen output steering for different FATRAS modules

setup, the physics processes to be included), but also contain tuning parameters that can be used to adjust the single FATRAS modules. A more detailed description of all the contained tuning possibilities are further described in the following, Sec. A.5. Once the `JobProperty` classes are defined, one single python script (`Fatras_jobOptions.py`) is parsed and the algorithmic sequence and the necessary tool and service classes are created according to the chosen parameters and configurations. For convenience, these tasks are sub-divided into different python setup classes: `FatrasGeometry`, `FatrasExtrapolation`, `FatrasFitting`, `FatrasTools`, `FatrasAlgs`, `FatrasReconstruction`, and `FatrasValidation`.

Stand-Alone Jobs The easiest example how to execute FATRAS is in a stand-alone job that only performs the FATRAS simulation and the default offline NEWT reconstruction. A user-friendly run configuration script can for this purpose found in the FatrasExample/share as `jobOptions.py`. It configures the default FATRAS properties and executes the `Fatras_jobOptions.py` script.

FATRAS Integration FATRAS aims to provide a fast alternative to the full offline chain that includes event simulation, digitisation and reconstruction. It is thus of great interest to provide a coherent integration into the standard ATLAS reconstruction job configurations. Taking account of this, the `Fatras_jobOptions.py` flags also include a full reconstruction and post processing setup for the Inner Detector and can be used in general ATLAS reconstruction setups replacing the standard ID

reconstruction configuration (`InDetRec_jobOptions.py`)[24].

VP1 Plugin A recent development established a dedicated FATRAS plugin for the rapidly growing ATLAS 3D event display VP1. It includes a single particle generation that is coupled to the FATRAS simulation sequence and allows thus a precise single track debugging together with a convenient graphical user interface. Figure 32 shows a screenshot of the VP1 user interface including the FATRAS steering panel to the left, and the actual 3D display canvas to the right: it shows a π^+ particle with a momentum of 5 GeV that causes a hadronic shower with the detector material.

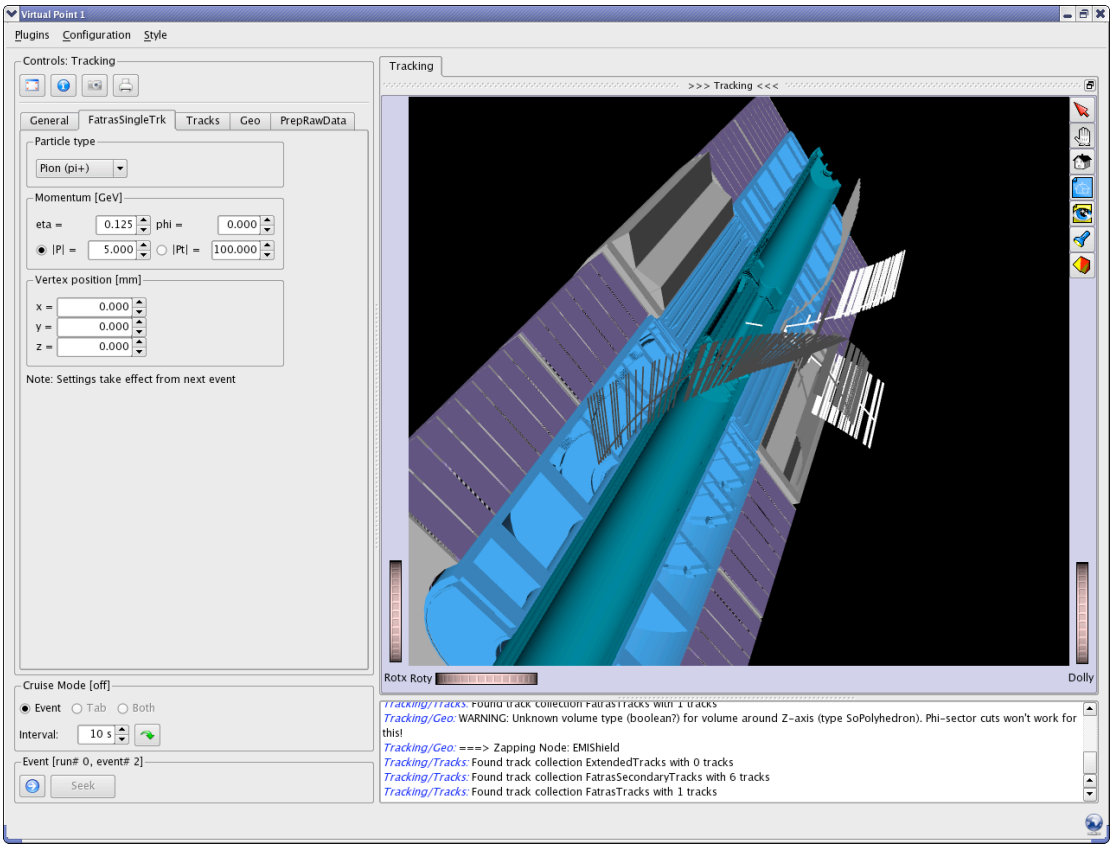


Figure 32: The VP1 FATRAS plugin for single particle event simulation. The current event shows a π^+ particle that causes a hadronic shower after interaction with the detector material.

A.5 Fit And Tuning Parameters

FATRAS offers the possibility to change its default behavior through the modification of a set of different tuning and cut parameters. Most of these parameters can be directly set through the `JobProperty` classes as described in the previous section. In the following, a complete list of the tuning parameters are given and their effect on the standard reconstruction will be described:

- the `FatrasClusterCreation` property class holds boolean flags for changing the clusterisation strategies. Simplified hit smearing can be chosen for each sub-detector, but also more complex clustering, such as the geometrical cluster creation in the silicon detectors (default) and the drift circle error description for the TRT from the offline conditions data (default). It also contains the scaling parameters for the high threshold hit simulation based on the transition radiation function used in the TRT.

[24] As a single requirement, the input collection has to be restricted to generator output, since both simulation and digitisation would have already filled event data collections that are otherwise filled by FATRAS.

- the majority of the tuning parameters can be found in the `FatrasTuning` property class. It includes the several scaling parameters for the implemented physics processes, the definitions of volumes where these processes are carried out. In addition, energy and momentum threshold cuts can be set and the noise levels in the FATRAS postprocessing modules can be adjusted. The given scaling parameters and their current default values are shown in Tab. 7.

There are certain cuts that seem to be contradictive, such as the creation of decay and conversion product at 250/150 MeV without a successive track simulation (p_T cut of 250 MeV). This is due to the fact, that when created particles would still be filled into the truth event collection. This behavior can, however, be simply switched off by adjusting the thresholds accordingly.

Table 7: `FatrasTuning` parameters for changing the default FATRAS simulation setup.

Parameter	Default Value
`MinimumParticlePtPrimaryTrackCreation`	250 MeV
`MinimumParticlePtSecondaryTrackCreation`	250 MeV
`MinimumParticlePtTertiaryTrackCreation`	250 MeV
`MinimumBremPhotonMomentum`	150 MeV
`BremCreationVolumeRadius`	1200 mm
`BremCreationVolumeHalfZ`	4000 mm
`BetheHeitlerScalor`	1 MeV
`ConversionMinChildEnergy`	150 MeV
`ConversionChildEnergyScalor`	2.
`ConversionProbabilityScalor`	0.98
`ConversionVolumeRadius`	1200 mm
`ConversionVolumeHalfZ`	4000 mm
`PixNoiseLevel`	10^{-6}
`SctNoiseLevel`	10^{-5}
`TrtNoiseLevel`	0.02

There are some additional model parameters that are mainly needed in the simulation of hadronic interaction, the photon conversions or in the clusterisation modules. All of which are adjustable but no public interface for the user is given since they are strongly coupled with the integrated fit or model functions; their behavior with completely random input variables can not be predicted.

The Component Software Model in FATRAS One important way of changing the FATRAS simulation has been missed out in this context: since the entire simulation chain is written in a component model software style with well-defined interfaces, several modules that include the core simulation processes in FATRAS can be exchanged with other plugins that follow completely different approaches or paradigms. The main interfaces used for FATRAS are contained in the TrkDetDescrInterfaces CVS location for the geometry description, the TrkExInterfaces concerning extrapolation and material integration and the FATRAS internal interface locations, see the following Sec. A.6. This choice of design should ease future adaptions and corrections of FATRAS, either in context of changed physics objectives or simply as an outcome of a validation at larger scale.

A.6 Package Overview

Many components of FATRAS are part of the general reconstruction software of ATLAS, in particular of the Tracking repository, while some of the hit classes are directly taken from the InnerDetector container. The central part of FATRAS that includes the `Algorithm` sequence and the steering of the simulation, on the one hand, and the dedicated Monte Carlo based components for the extrapolation engine such as the track creation engine, on the other hand, is concentrated in a dedicated container package of the ATLAS CVS repository. A complete overview of the contained packages including a brief description of their content is given in Tab. 8 for the convenience of the user.

Table 8: The `JobProperty` classes available in FATRAS for job configuration.

Package	Content
FatrasAlgs	main `Algorithm` classes that build the FATRAS sequence
FatrasDetDescrExample	example package for generic geometry (SLHC)
FatrasDetDescrInterfaces	an interface package for generic geometry building
FatrasDetDescrSvc	FATRAS geometry service steering
FatrasDetDescrTools	`AlgTool` classes for generic geometry creation
FatrasDetDescrUtils	data classes for generic geometry
FatrasEvent	extended FATRAS event data model
FatrasEventAthenaPool	persistency converter classes
FatrasExample	example package for FATRAS steering, tuning and execution
FatrasG4Algs	wrapper `Algorithm` for Geant4 based particle decay
FatrasG4Tools	wrapper `AlgTool` for Geant4 based particle decay
FatrasInterfaces	all specific interface definitions
FatrasTools	dedicated FATRAS `AlgTool` classes
FatrasTruth	truth data model
FatrasValidation	validation `Algorithm`classes

References

[1] T. Sjörstrand, S. Mrenna and P. Skands, *PYTHIA 6.4 - Physics and Manual*, hep-ph/06013175

[2] G. Corcella et al., *HERWIG 6.5*, hep-ph/0210213

[3] Agostinelli et al., *GEANT4: A Simulation toolkit*, Nucl. Inst. & Meth., **A 506**, 2003.

[4] Atlfast homepage, *www.hep.ucl.ac.uk/atlas/atlfast*

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

[6] A. Salzburger, M. Wolter and S. Todorova, *The ATLAS Tracking Geometry Description*, ATLAS Note, ATL-SOFT-PUB-2007-004, 2007.

[7] A. Salzburger, *The ATLAS Extrapolation package*, ATLAS Note, ATL-SOFT-PUB-2007-005, 2007.

[8] A. Salzburger, *The new Fast ATLAS Track Simulation (FATRAS)*, Proc. of CHEP 2006, 2006.

[9] Atlantis homepage, *www.cern.ch/atlantis/*

[10] The ATLAS Collaboration, *ATLAS Computing Technical Design Report*, ATLAS TDR, CERN-LHCC-2005-022, 2005.

[11] A. Salzburger (Editor) et al, *Concepts, Design and Implementation of the ATLAS New Tracking (NEWT)*, ATLAS Note, ATL-SOFT-PUB-2007-002, 2007.

[12] VP1 homepage, *http://atlas-vp1.web.cern.ch/atlas-vp1/*

[13] HepMC homepage, *https://savannah.cern.ch/projects/hepmc/*

[14] V.L. Highland, *Some Practical Remarks on Multiple Scattering*, Nucl. Inst. & Meth. **129**, 1975, and *Erratum*, Nucl. Inst. & Meth., **161**, 1979.

[15] R. Frühwirth and M. Regler, *On the quantitative modelling of core and tails of multiple scattering by Gaussian mixtures*, Nucl. Inst. & Meth. **A 456**, 2001.

[16] R. Frühwirth and M. Liendl, *Mixture models of multiple scattering: computation and simulation*, Comp. Phys. Comm. **141**, 2001.

[17] L.D. Landau, *On the energy loss of fast particles by ionisation*, J. Phys. USSR, **8**, 1944.

[18] H. Bethe and W. Heitler, *On the stopping of fast particles and the creation of positive electrons*, Proc. Roy. Soc. **A 146**, 1934.

[19] R. Frühwirth and A. Strandlie, *Track finding and fitting with the Gaussian-sum Filter*, Proc. of CHEP 1998, 1998.

[20] T. Lari, *Study of silicon pixel sensors for the ATLAS detector*, CERN-THESIS-2001-028, 2001.

[21] E. .J Buis, R. J. Dankers, S. Haywood and A. Reichold, *Parameterisation of the Inner Detector Performance*, ATLAS Note, INDET-97-195, 1997.

[22] Yung Tsai, Rev.Mod.Particle Physics Vol. 74, No.4, October 1974

[23] Geant4 homepage, *http://www.cern.ch/geant4*

[24] ROOT homepage, *http://root.cern.ch*

[25] A. Salzburger, *A Parametrization for Fast Simulation of Muon Tracks in the ATLAS Inner Detector and Muon System*, CERN-THESIS-2004-051, 2003.

[26] A. Nairz, *private communication*

[27] Physics validation project homepage, *http://lcgapp.cern.ch/project/simu/validation/*

[28] KSI2K definition, *http://computingcourier.web.cern.ch/ComputingCourier/NL_2004Nov/Batch-Time-Units.doc*

[29] S. Fleischmann, *Track Reconstruction in the ATLAS Experiment : The Deterministic Annealing Filter*, CERN-THESIS-2007-01, 2006.

[30] ATLAS Upgrade homepage, *http://atlas.web.cern.ch/Atlas/GROUPS/UPGRADES/*

[31] S. Todorova, *private communications*

[32] FastCaloSim twiki page, *https://twiki.cern.ch/twiki/bin/view/Atlas/FastCaloSim*

[33] Atlas offline software repository, *http://atlas-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/*

[34] P. Calafiura et al., *Physics-Level Job Configuration*, Proc. of CHEP 2006, 2006.

A conclusion
is the place where
you got tired
of thinking.

Arthur Bloch

Chapter 10

Conclusion

In this thesis, a broad aspect of a newly established track reconstruction chain together with a new fast track simulation engine for the ATLAS experiment have been presented. The work includes detail description of many key modules that have been deployed to the ATLAS software release; all of them are indispensable for track reconstruction and are part of the new default tracking strategy. Completely new concepts, such as e.g. the connective geometry model for navigation or the inclusion of a new propagation technique through dense material, have been integrated in a modern, modular software structure that allows for necessary adaptions and extensions to complete the needed functionality and gain highest performance for future analyses and fundamental searches with the ATLAS detector.

The performance of the involved modules, but also the entire new ATLAS track reconstruction has reached (and partly exceeds) the level of former reconstruction chains, that have marked the reference for track reconstruction in high energy physics experiments for many years.

One of the strongest arguments, however, is the flexibility of the deployed component model, which is necessary for the maintenance of the reconstruction software during the long lifetime of the experiment. With the start of data taking coming up in just a few months, the deployed solutions will undergo the most stringent tests that may very likely force the revision of several individual parts of the track reconstruction. This aspect has been considered with dedicated care. Many successful tests of the new ATLAS track reconstruction strategy have been performed on Monte Carlo simulated events or taken data from the combined test beam that took place in 2004. Finally, ongoing commissioning runs using cosmic rays add confidence about the readiness for first physics data. A high level of stability could be reached, while the CPU time consumption needed for track reconstruction could be kept at a reasonable small cost. Most importantly, however, the needed performance for future physics analyses could be achieved, serving the collaboration with high quality data from track reconstruction.

A major part of the presented work covers a new fast track simulation program for the ATLAS detector. Monte Carlo simulation is an essential tool for high energy physics, and many studies with the ATLAS detector will need large event samples, in particular for the background and cross section estimations. A complete usage of the full detector simulation will not be possible, since it is very CPU time intensive and can not be done exclusively under the constraints of time and the computing budget of the experiment. In current figures, the very detailed full detector simulation will be used for a maximum of 25 % of all needed MC simulated data and fast simulation techniques are required to fill the missing gap and achieve the full event statistics. Fast track simulation has been for many years limited to the

pure parametric smearing of the input variables and neither refined tracking studies nor realistic vertexing or heavy quark tagging could be performed on the according output data. This thesis, however, presents a new fast track simulation strategy that has been recently established and which produces full hit information and thus supports the entire track reconstruction chain. The relevant physics processes, such as the fundamental interaction of the particles with the detector material or particle decays have been integrated and a realistic cluster creation is contained. It incorporates a full MC simulation, but mainly uses the fast modules from the offline track reconstruction chain that is covered in the first part of this thesis. By doing so, the new fast track simulation outmatches the full detector simulation in terms of execution speed by two orders of magnitudes, while — at the same time — remains in a sub to low percentage discrepancy with results from full simulation. A full integration of this fast track simulation with a fast calorimeter simulation has been done and together they build the backbone of a new, third simulation strategy alongside the full detector simulation and the parametric smearing approach.

A major effort has been done to establish and complete the described components of the new track reconstruction and simulation chain just before ATLAS starts taking data later this year. The full success or failing of the deployed structure will only be revealed then. However, the author is — also backed up by results from test beam runs and detector commissioning using cosmic rays — confident that the path chosen has indeed been a good one.

In the spirit of this thesis' introduction, may it lead to one of these tops that will allow us to look through the gates of the known world. Only then it will fulfill its purpose: to be a tool for exploring and hopefully answering some of the questions about the fundamental laws of nature that cause everything to be.

Appendix A

Appendix

A.1 Impact parameter resolutions

In this section it will be briefly discussed why the resolution of the impact parameters d_0 and z_0 mainly depend on the measurements on the first two layers. The following trivial assumptions illustrate also the often used expression of the dependency of the impact parameter resolutions on the momentum

$$\sigma_{d_0} = A \oplus \frac{B}{p} \tag{A.1}$$

We assume in the following a simple detector setup with two cylindrical layers at radii r_1 and r_2. In addition, the layers will be confined in a homogenous magnetic field pointing along the symmetry axis of the detector. The track parameterisation incorporates a helical track model, yielding

$$\mathbf{x} = (d_0, z_0, \phi_0, \cot\theta, q/p_{\mathrm{T}})^T \tag{A.2}$$

for the perigee representation. When assuming a perfectly homogeneous magnetic field, the transverse parameters $\mathbf{x}_T = (d_0, \phi_0, q/p_{\mathrm{T}})^T$ can be dealt independently from the longitudinal ones $\mathbf{x}_L = (z_0, \cot\theta)$. The track extrapolation to the detection layer i is then given by the intersection $\mathbf{m}_i^{pred} = (t_i^{pred}, z_i^{pred})^T$ of the helix with ith cylinder

$$t_i^{pred} = d_0 + r_i\phi_0 + \frac{1}{2\rho}r_i^2, \tag{A.3}$$

in the transverse, and

$$z_i^{pred} = z_0 + r_i \cot\theta \tag{A.4}$$

in the longitudinal plane, respectively. In Eq. (A.3) we have used a parabolic approximation of the circle, which is valid for the high momentum approximation.

We will restrict the continuing considerations to the longitudinal parameter set, simply for the fact that it reflects a purely linear problem and is thus very intuitive. However, a similar path can be laid out for the transverse coordinates with using a linearised transport equation. When including multiple scattering effects, Eq. (A.4) has to be modified to

$$z_i^{pred} = z_0 + r_i \cot\theta + \sum_{j=1}^{i-1} \Delta\cot\theta_j \cdot \Delta r_{ij}, \tag{A.5}$$

where θ_j denotes the deflection due to scattering at the jth layer and $\Delta r_{ij} = (r_i - r_j)$. In a global track fit that is restricted to the longitudinal coordinates, we can now — following

Eq. (4.14) — denote the χ^2 contributions on the two layers as

$$\chi_1^2 = \frac{[z_1 - (z_0 + r_1 \cot\theta)]^2}{\sigma_1^2} + \frac{(\Delta \cot\theta_1)^2}{\sigma_{MS,1}^2} \tag{A.6}$$

and, respectively,

$$\chi_2^2 = \frac{[z_2 - (z_0 + r_2 \cot\theta) + \Delta r_{12} \Delta \cot\theta_1]^2}{\sigma_2^2} + \frac{(\Delta \cot\theta_2)^2}{\sigma_{MS,2}^2}. \tag{A.7}$$

In Chap. 7 it has been shown that the variance of the multiple scattering distribution is inverse dependent on the momentum of the particle, denoting

$$\sigma_{MS}^2 = \frac{k}{p^2}. \tag{A.8}$$

It is probably more intuitive to denote Eqs. (A.6) and (A.7) in matrix notation, which yields when inserting Eq. (A.8)

$$\chi^2 = \chi_1^2 + \chi_2^2 = \mathbf{x_0}^T \mathbf{C}^{-1} \mathbf{x_0}, \tag{A.9}$$

when $\mathbf{x}_0 = (z_0, \cot\theta, \Delta\cot\theta_1, \Delta\cot\theta_2)^T$ is the vector of parameters to be fitted[1]. The inverse covariance matrix $\mathbf{C}^{-1}$ can hereby we written as

$$\begin{aligned} \mathbf{C}^{-1} &= \begin{pmatrix} 1 & 0 \\ r_1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{k_1^2}{p^2} \end{pmatrix} \begin{pmatrix} 1 & r_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ &+ \begin{pmatrix} 1 & 0 \\ r_2 & 0 \\ r_{12} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma_{z,2}^2} & 0 \\ 0 & \frac{k_{z,2}^2}{p^2} \end{pmatrix} \begin{pmatrix} 1 & r_2 & r_{12} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned} \tag{A.10}$$

and, after performing the matrix multiplication,

$$\mathbf{C}^{-1} = \begin{pmatrix} \frac{1}{\sigma_z^2} + \frac{1}{\sigma_z^2} & \frac{1}{\sigma_z^2} r_1 + \frac{1}{\sigma_z^2} r_2 & \frac{1}{\sigma_z^2} r_{12} & 0 \\ \frac{1}{\sigma_z^2} r_1 + \frac{1}{\sigma_z^2} r_2 & \frac{1}{\sigma_z^2} r_1^2 + \frac{1}{\sigma_z^2} r_2^2 & \frac{1}{\sigma_z^2} r_2 r_{12} & 0 \\ \frac{1}{\sigma_z^2} r_{12} & \frac{1}{\sigma_z^2} r_2 r_{12} & \frac{1}{\sigma_z^2} r_{12}^2 + \frac{k^2}{p^2} & 0 \\ 0 & 0 & 0 & \frac{k^2}{p^2} \end{pmatrix}. \tag{A.11}$$

The single covariance matrix entries can now be found by inverting $\mathbf{C}_1$, yielding

$$\mathbf{cov}(z_0) = (C_z)_{11} = \frac{r_1^2 \sigma_2^2 + r_2^2 \sigma_1^2}{(r_2 - r_1)^2} + \frac{k_1^2 r_1^2}{p^2}, \tag{A.12}$$

described by the orthogonal sum as

$$\sigma_{z_0} = A_{z_0} \oplus \frac{B_{z_0}}{p} = \frac{r_1 \sigma_2 \oplus r_2 \sigma_1}{r_{12}} \oplus \frac{k_1 r_1}{p}. \tag{A.13}$$

[1] In Sec. 4.3.1 it has been motivated how the deflection angles are fitted parameters in the least squares method.

Curriculum Vitae

Personal Data

Name: Andreas Salzburger
Born: 22^{nd} of November 1977, Wörgl, Austria
Parents: HOL Hubert Salzburger, Anna Salzburger
Brothers: Dr. Walter Salzburger, Mag. Thomas Salzburger
Nationality: Austrian

Education

1984-1988	grammar school	Volksschule Kramsach
1988-1996	high school	Bundesrealgymnasium Wörgl
Mai 1996	A-level degree, first honors	
1997-2003	study of physics	Univ. of Innsbruck
June 2003	Master of science degree, first honors	Univ. of Innsbruck
July 2003 - 2005	PhD student in the Austrian Doctoral Student program at CERN	CERN, Univ. of Innsbruck
Jan. 2006 - July 2008	Project Associate to the ATLAS experiment at CERN	

Oct. 1996 to Sept. 1997 social service, Landessonderschule Kramsach

Scholarships

summer 2001	official CERN summerstudent
summer 2002	CERN NA48 summerstudent

Awards

November 2003	Award of the Austrian Ministry of Science in appreciation of outstanding achievement during the studies (*Würdigungspreis des Ministeriums*)

Publications

Conference Proceedings

Salzburger, A. et al., *The new ATLAS Fast Track Simulation engine (FATRAS)*, Proceedings of CHEP2006, Mumbai 2006.

Lavrijsen, W. et al., *Physics-Level Job Configuration*, Proceedings of CHEP2006, Mumbai 2006.

Salzburger, A., *Track extrapolation with intrinsic navigation in the new ATLAS tracking scheme* 9^{th} ICATPP Conference on Astroparticle, Particle, Space Physics, Detectors and Medical Physics Applications, published in: World Scientific, 2005.

Salzburger, A., *The Track Extrapolation package in the new ATLAS Tracking Realm*, Proceedings of CHEP2004, Interlaken 2004.

Collaboration Notes

Edmonds, K. et al., *The Fast ATLAS Track Simulation (FATRAS)* Public ATLAS note, ATL-SOFT-PUB-2008-001, 2008.

Cornelissen, T. et al., *Single Track Performance of the Inner Detector New Track Reconstruction (NEWT)*, ATLAS communication, ATL-COM-INDET-2008-004, 2008.

Lopez Mateos, D., Hughes, E. W. and A. Salzburger, *A Parameterization of the Energy Loss of Muons in the ATLAS Tracking Geometry*, Public ATLAS note, ATL-COM-MUON-2008-001, 2008.

Salzburger, A. (ed.) et al.,*Concepts, Design and Implementation of the ATLAS New Tracking (NEWT)*, Public ATLAS note, ATL-SOFT-PUB-2007-007, 2007.

Akesson, F. et al., *ATLAS Inner Detector Event Data Model*, Public ATLAS note, ATL-SOFT-PUB-2007-006, 2007.

Salzburger, A.,*The ATLAS Track Extrapolation Package*, Public ATLAS note, ATL-SOFT-PUB-2007-005, 2007.

Salzburger, A., Todorova, S. and M. Wolter, *The ATLAS Tracking Geometry Description*, Public ATLAS note, ATL-SOFT-PUB-2007-004, 2007.

Cornelissen, T. et al., *Updates of the ATLAS Tracking Event Data Model (Release 13)*, Public ATLAS note, ATL-SOFT-PUB-2007-003, 2007.

Akesson, F. et al., *The ATLAS Tracking Event Data Model*, Public ATLAS note, ATL-SOFT-PUB-2006-004, 2006.

Salzburger, A., Kuhn, D. (dir), *A Parametrization for Fast Simulation of Muon Tracks in the ATLAS Inner Detector and Muon System* , CERN thesis , CERN-THESIS-2004-051, Innsbruck 2003.