Diplomarbeit[1] von

Jan Fiete Grosse-Oetringhaus

Institut für Kernphysik
Universität Münster

Determination of the Physics Performance of the ALICE Central Barrel using a Distributed GRID Computing Environment

— 2005 —

[1] Revised Version: Plots 4.17 and 4.18 which are external input from [Som05b] have been updated, together with their related section 4.7.

For Mareike

Contents

1. Introduction

The questions "What are we made of?" and "Where do we come from?" have occupied the human mind since the very beginning. Scientists have tracked down the constituents of matter to molecules, atoms, nucleons and, today, to quarks and leptons. Theories evolved that describe the fundamental forces and interactions as well as the creation of the universe. Following the generally accepted Big Bang theory, everything started in an area of very high energy density and has since then been expanding and cooling down rapidly. Theories supported by experiments describe the evolution from 10^{-5} s after the Big Bang quite well. At this point hadronization took place: quarks and gluons were confined together to hadrons – including protons and neutrons, the matter we are made of. This process is usually associated with a phase transition from a state called quark-gluon plasma to the hadronic phase – similar to a transition from a liquid phase to a solid phase, where molecules bind together forming crystals. This was studied at SPS[1] and RHIC[2] and traces of a new phase have been seen. However, these are faint and could also be explained differently with not taking the formation of the quark-gluon plasma into account. Today tremendous efforts of world-wide collaborations are combined to make experiments of a new scale to answer the still open questions. At the European Organization for Nuclear Research (CERN) in Geneva the Large Hadron Collider with a circumference of 27 km is being built. It will speed up protons to an energy of 14 TeV per pair and Pb ions to 5.5 TeV per nucleon pair in their center of mass system. Five experiments will record the outcome of collisions and a gigantic computing infrastructure will be needed to store and analyze the data. One of the experiments, ALICE[3], is designated to analyze hot dense matter and find convincing evidence of the quark-gluon plasma.

In this diploma thesis, the physics performance of the ALICE central barrel will be parameterized. This enables performance studies, which will show that ALICE will be able to measure states of the quarkonia family and therefore may reveal the quark-gluon plasma. Furthermore, necessary tools will be developed to be able to analyze the outcome of the experiments. These tools will be part of the mentioned computing infrastructure – called Grid.

The following chapter will give an introduction to the theoretical basics of high energy physics. The Large Hadron Collider and the ALICE experiment in particular will be

[1] SPS – Super Proton Synchrotron, CERN (Geneva).
[2] RHIC – Relativistic Heavy Ion Collider, BNL (Brookhaven).
[3] ALICE – A Large Ion Collider Experiment.

outlined in chapter 3. In the fourth chapter, the physics performance of the ALICE central barrel will be determined. This is the basis for performance studies which will also be shown. Chapter 5 will then cover a general introduction into distributed analysis and Grid environments. In chapter 6, performed developments which support distributed analysis and Grid computing will be explained.

2. Theoretical Basics

2.1 History

In 1909, Rutherford performed his famous alpha-scattering experiments which led him, in 1911, to the postulate that the atom contains a nucleus, although according to its Greek name *"άτομος" (atomos)*, which means indivisible, it should not have a substructure. Shortly after, Rutherford realized that hydrogen is the lightest element and has an indivisible nucleus. He concluded that this charged nucleus is an elementary particle and named it proton. The electrons, discovered by Thomson already in 1897, were identified as part of the atom. In 1932, Chadwick found a particle similar to the proton but without charge and called it neutron. In the same year, Heisenberg postulated that nuclei are composed of protons and neutrons. The first anti-particle, the positron, was discovered and proved Dirac's hypothesis that all charged particles have a counterpart with the opposite charge. Theoretical models were developed which explain why protons and neutrons bind together to nuclei. The invention of the Geiger counter[1] and the synchrotron principle for particle acceleration then boosted particle physics. Numerous elementary particles (the so-called *particle zoo*) were discovered.

The elementary particles seemed indivisible for a while, but in 1964 constituents, called quarks, were postulated by Gell-Mann who developed schemes to classify elementary particles, analogous to the periodic system of elements. Quarks were then confirmed in deep inelastic scattering experiments. Today a theory, called *Standard Model*, of the elementary particles has been established which explains most of the observed phenomena.

2.2 The Standard Model

The Standard Model describes the composition of matter and its fundamental forces. Constituents of matter are a set of point-like particles which have a spin of $\frac{1}{2}\hbar$: quarks and leptons that appear in three families containing two members each (see Table 2.1). Four fundamental forces are known in nature: gravitation, the strong and weak nuclear forces and the electromagnetic force. These are mediated by the exchange of

[1] H. Geiger, W. Müller, 1928

Family	Quarks			Leptons		
	Name	Charge	Mass	Name	Charge	Mass
1	u	$2/3\ e$	$1.5 - 4$ MeV/c^2	e^-	$-e$	0.511 MeV/c^2
	d	$-1/3\ e$	$4 - 8$ MeV/c^2	ν_e	0	< 3 eV/c^2
2	c	$2/3\ e$	$1.15 - 1.35$ GeV/c^2	μ^-	$-e$	105 MeV/c^2
	s	$-1/3\ e$	$80 - 130$ MeV/c^2	ν_μ	0	< 0.19 MeV/c^2
3	t	$2/3\ e$	174.3 ± 5.1 GeV/c^2	τ^-	$-e$	1.78 GeV/c^2
	b	$-1/3\ e$	$4.1 - 4.4$ GeV/c^2	ν_τ	0	< 18.2 MeV/c^2

Table 2.1: Quarks and leptons in the Standard Model [Eid04]
The Standard Model divides the constituents of matter into quarks and leptons. Both appear in three families with two members each.

particles, called *gauge bosons*[2] (see Table 2.2). The Standard Model describes all forces mentioned above except gravitation.

Leptons appear as free particles which interact weakly and, if charged, electromagnetically. Quarks appear in three colors and are subject to the strong, weak and electromagnetic forces. The interaction among quarks is governed by the strong force and described by Quantum Chromodynamics (QCD), a theory similar to Quantum Electrodynamics (QED) which describes the interaction between charged particles. In QCD color charges take the place of electrical charges in QED. Furthermore, in contrast to QED which has one gauge boson, the photon, QCD has 8 gauge bosons, the gluons. Numerical approaches to QCD are complicated because the coupling constant of the strong force is of the order of unity – which prevents using perturbation theory. However, at very short distances, or large energy scales, the coupling constant decreases and thus quarks appear to be only weakly coupled.

Matter is divided into leptons and hadrons, out of which the latter are subdivided into baryons and mesons. Examples of leptons are the electron e and the muon μ. Baryons are formed out of three quarks (qqq or $\overline{qqq}$ for an antibaryon) and are fermions[3], where q is denoting a quark and $\overline{q}$ an antiquark. Mesons are formed by two quarks ($q\overline{q}$) and are therefore bosons. Examples of baryons are the proton p (quark content: uud) and the neutron n (udd); mesons are for example the charged pions π^+ ($u\overline{d}$) and π^- ($\overline{u}d$).

[2] *Bosons* are particles which have integer spin and are subject to the Bose-Einstein statistic.
[3] *Fermions* are particles which have half-integer spin, they are subject to the Fermi-Dirac statistic.

Force	Strength	Gauge Bosons	Applies on
Strong nuclear force	1	Gluons	Quarks
Electromagnetic force	$\sim 10^{-2}$	Photons	All charged particles
Weak nuclear force	$\sim 10^{-7}$	$W^{\pm}$, Z^0	Quarks, leptons
Gravitation	$\sim 10^{-39}$	Gravitons	All massive particles

Table 2.2: Fundamental forces [Per00]

All forces except gravitation are described by the Standard Model. The strength is given in relation to the strength of the strong nuclear force. The gravitons, the gauge bosons of the gravitation, are postulated, but have not been found yet.

Quarks do not appear as free particles; they always bind together to form hadrons. This constraint is due to the potential of the strong force and is called *confinement*. The potential between two quarks can be approximated by [Sat90]

$$V(r) = -\alpha/r + \sigma r. \tag{2.1}$$

The first term is coulomb-like, the second shows the confinement.

There may be, however, a state of matter where the confinement theory does not hold. It is called the *quark-gluon plasma*. When particles pass from today's hadronic phase to this new phase they deconfine and exist not as hadrons, but as freely moving quarks and gluons. The phase transition that occurs in the sun is similar: Atoms in a gas phase split up to their nuclei and their electrons, and form a plasma. However, nuclei and electrons are in principle able to move freely in a gas phase, unlike quarks, which cannot move freely before reaching the deconfined state.

The quark-gluon plasma, its creation and signatures will be topic of the next section.

2.3 The Quark-Gluon Plasma

QCD calculations predict for high temperatures T, or high baryochemical potential[4] μ_B, a phase transition to a state where quarks and gluons are not bound as hadrons, but move freely (see Figure 2.1). Interactions are predicted at short distances which are governed by weak coupling – due to the logarithmically falling coupling constant at short distances. Long-range interactions are prevented by a process called color screening caused by the high density of quarks and gluons. This will be outlined later in detail.

[4]The baryochemical potential μ_B of a system is a measure for the energy which is needed to add a baryon to it.

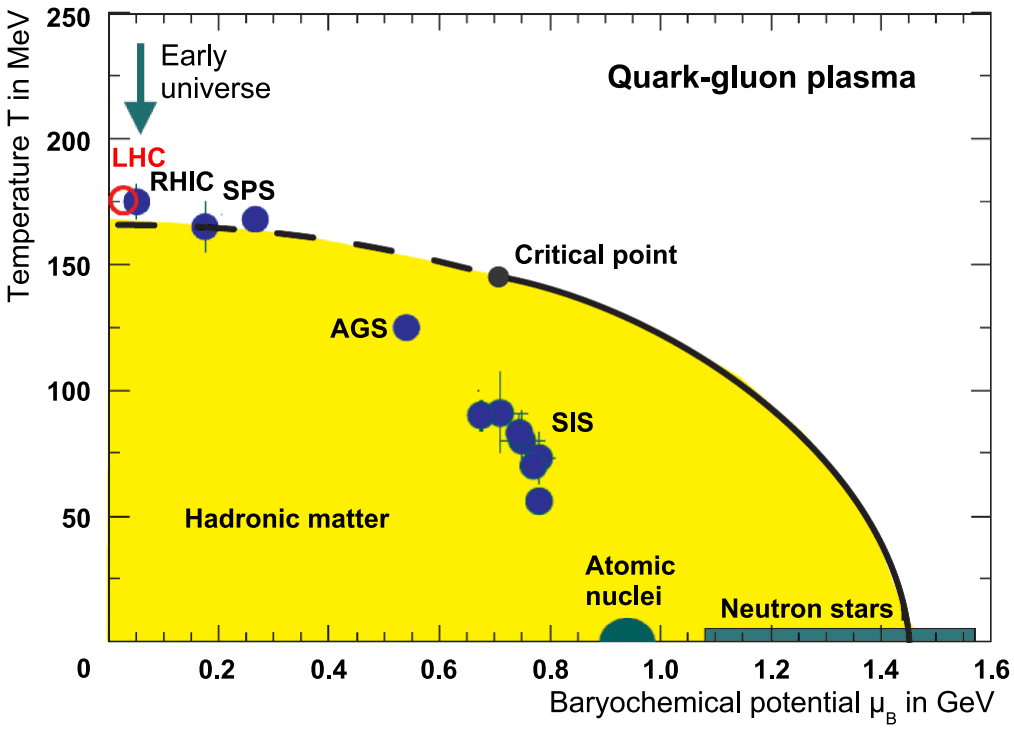

Figure 2.1: The QCD phase diagram

The diagram shows the transition from hadronic matter to the quark-gluon plasma as a function of the temperature T and the baryochemical potential μ_B. The energies which former experiments could and the LHC and RHIC can reach are shown in the diagram. Beyond the critical point (dashed line) no first order phase transition is possible anymore. A first order phase transition is defined as a phase transition where the first derivative of the free energy, $\partial F/\partial T$, is discontinuous while all the lower derivatives in T are continuous [Won94]. Beyond the critical point no discontinuity occurs.

The new phase, not subject to confinement and without broken chiral symmetry[5], is called the *quark-gluon plasma*.

Lattice QCD calculations show the transition temperature T_C to the new phase at 150 ± 10 MeV and vanishing μ_B [Har96], which is in the range of today's particle accelerators. The universe exceeded 200 MeV about 10 μs after the Big Bang. If in a high energy physics experiment a quark-gluon plasma is formed, it will happen very early after the collision; when particles reach the detectors the system will have passed through the phase already. It is therefore the major task to find signatures of a phase

[5]The Lagrangian of QCD implies chiral symmetry. Thus the baryon number should be conserved for right-handed and left-handed quarks separately. However, it is broken explicitly due to the finite quark masses; therefore only the overall baryon number is conserved.

which has existed long before the measurement. This is similar to the 3 K background radiation which is studied as a remainder of the early evolution of the universe. Several signatures are expected to be seen: however, like the new phase, these are not well defined. Furthermore, there is no single, unique signal which proves the existence of this new phase. Many of the signatures can also be explained differently, not taking the quark-gluon plasma into account. Nevertheless, the measurement of a set of signatures would be strong evidence for the formation of the quark-gluon plasma [Won94].

Strongly interacting matter with sufficiently high temperatures and densities for forming the quark-gluon plasma can be produced in ultra-relativistic heavy-ion collisions. In contrast to proton-proton collisions these imply a much higher energy density.

2.3.1 Color Screening and Quark Deconfinement

The process called screening is well known from electrodynamics. The force between two charges is influenced by other present charges; in particular by charges between them. The force between two charges decreases as more charges are present. An example is the lamb shift in the spectrum of hydrogen, caused by the screening of electrons and positrons from vacuum polarization. In QCD, color charges take the place of electrical charges and in dense, strongly interacting matter, quarks and gluons – which carry a color charge – can cause the so-called *color screening*.

A simple approach proposed in [Sat90], showing the influence of color screening on the binding potential of quarks, will be given as an example. The non-relativistic potential, given in Eq. (2.1), is justified by the fact that the quark masses are much larger than the relevant kinetic energies. The first term denotes a Coulomb-like behavior, the second the confinement. Taking screening into account the potential changes to

$$V'(r,\omega) = -\frac{\alpha}{r}\, e^{-\omega r} + \sigma r \left[\frac{1 - e^{-\omega r}}{\omega r}\right] \tag{2.2}$$

with the screening mass ω, which is equal to the inverse screening length – the length where two charges do not see each other anymore. For short distances the Coulomb term dominates

$$\lim_{r\to 0} rV'(r,\omega) = -\alpha, \tag{2.3}$$

but for large distances the exponential damping of the binding potential can be seen

$$\lim_{r\to\infty} \frac{1}{r} \ln\left[\frac{\sigma}{\omega} - V'(r,\omega)\right] = -\omega. \tag{2.4}$$

The evaluation of the Hamiltonian results in bound states of the system with energies $E_i(r_i,\omega)$ for $\omega < \omega_c$. Above ω_c no bound states are possible. For example, all $c\bar{c}$-bound

states are dissolved at $\omega \geq 0.5$ GeV/c^2. For each state a dissociation energy dependent on ω can be calculated, which is the energy needed to dissociate the $q\overline{q}$-pair. The dissociation of bound states implies that quarks move freely, they are *deconfined*. In reality, bound states are suppressed before reaching ω_c because its calculation assumed a static model. However, thermal motion of the system causes an earlier dissociation.

In addition to the outlined screening between quarks, gluons can also participate in the screening due to their color charge. This is different from QED where the gauge boson, the photon, does not carry a charge. The gluon-gluon screening causes a temperature-dependent change in the Coulomb potential and also influences ω.

In the experiment, the dependence of ω on the temperature T or the baryochemical potential μ_B should be determined. It is also very important to study the behavior at the phase transition ($T = T_C$) and this will probably answer the question if deconfinement is exclusively an effect of color screening.

2.3.2 Signatures of the Quark-Gluon Plasma

The signatures of the quark-gluon plasma will be outlined briefly; for a detailed description see for example [Har96].

Quarkonia Suppression or Enhancement

The ratio of quarkonia states (J/Ψ taken as example here) per number of nucleon-nucleon collisions, measured at different centralities, is constant in the case that no quark-gluon plasma is present. Reference data for this case can be obtained from collisions of protons or light ions.

In the quark-gluon plasma, J/Ψs cannot form because the corresponding quark pairs, created in the collision, cannot merge due to color screening and deconfinement. When a pair leaves the phase or the system freezes out the quarks are far apart. Thus it is more likely that they bind with other quarks, which are available at much higher densities. Therefore, the number of J/Ψ should be suppressed and the previously mentioned ratio should decrease towards more central collisions. SPS and RHIC data supports this prediction. The suppression effect is explained in more detail in the following section.

However, new approaches modelling statistical hadronization in heavy-ion collisions [And03] predict an enhancement of the J/Ψ signal at LHC energies caused by coalescence of c and $\overline{c}$ quarks in the quark-gluon plasma. This effect superimposes the

suppression and leads to an increase of the J/Ψs per number of nucleon-nucleon collisions towards more central collisions. Of course the ratio is still below the case when no quark-gluon plasma is present.

Strangeness Enhancement

Strange particles have to be produced in pairs due to strangeness conservation. The production of an associated kaon pair has a threshold energy of about 987 MeV. Kaons are the lightest strange particles and thus this value is also the threshold for the production of any strange particle pair in hadronic matter. In the quark-gluon plasma $s\overline{s}$-pairs can be produced directly, which has a much lower threshold energy of about 300 MeV. Consequently, strange particles will be produced in larger numbers.

Jet Quenching

The energy loss of quarks or gluons traversing the quark-gluon plasma is predicted to be higher than in hadronic matter. Partons traversing the plasma also loose energy and are deflected. The co-planarity with the beam axis of two jets, originating in hard parton-parton scattering, may thus be destroyed. This is called *Jet Quenching*.

Signatures of Restoration of Chiral Symmetry

The transition restoring chiral symmetry may be preceded by a drop of vector meson masses. As a consequence the widths and positions of the ρ, ω and ϕ peaks are changed. This is of special interest because the modifications are predicted to be small, except in the immediate vicinity of the phase transition.

More exotic models predict formations of domains called *disoriented chiral condensate* (DCC) when chiral symmetry is restored. This may lead to a significant change in the pion charge ratio N_{π^0}/N_{π}, which is usually $\frac{1}{3}$ [Har96].

Kinematic Probes

The behavior of the energy density ϵ, pressure p and entropy density s of the phase as a function of the temperature T and the baryochemical potential μ_B can be studied. A rise in the effective numbers of freedom would lead to changes of the variables and is characteristic of a phase transition. The variables T, s, ϵ are usually identified with

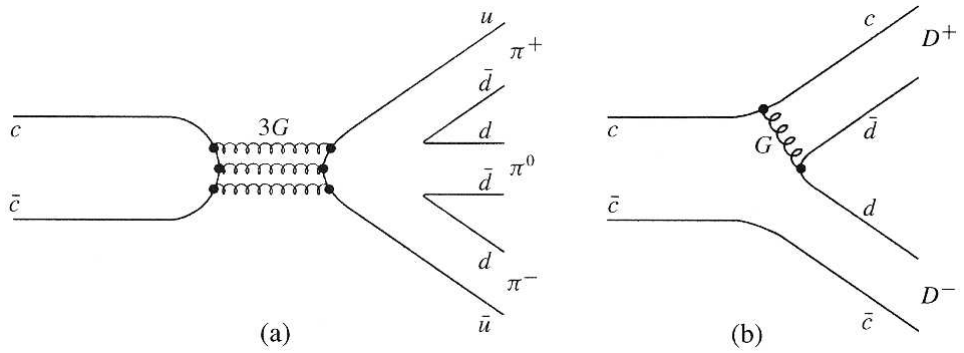

Figure 2.2: Quark line diagrams for charmonium decay [Per00]

Shown are decays of charmonium: (a) $c\overline{c} \rightarrow 3\pi$, which implies a change in quark flavor and is suppressed due to unconnected quark lines (OZI-rule); (b) $c\overline{c} \rightarrow D\overline{D}$, which is favored, but forbidden by energy conservation for the lowest charmonium states J/Ψ and Ψ' and therefore causes their narrow states.

the average transverse momentum, the hadron rapidity distribution or the transverse energy, respectively, which are measurable.

However, most variables show a distinct behavior only at a first order phase transition, which will most likely not be measurable at LHC.

Electromagnetic Probes

Photons and lepton pairs are probes of the earliest and hottest phase in the evolution of the quark-gluon plasma, because they do not undergo strong final state interactions. Their spectrum is superimposed by relatively large backgrounds from hadronic processes. Direct photons in the transverse momentum range $2 - 5$ GeV/c can be used as a probe for a very hot plasma that is formed initially. Unfortunately the photon spectra of a hadron gas and the quark-gluon plasma near the critical temperature T_C are similar. Lepton pairs can also be used to measure the thermalization time of the quark-gluon plasma.

2.4 The Quarkonia Family

ALICE will study the behavior of quarkonia rates under certain conditions as evidence for deconfinement. The quarkonia family and its properties will be covered by this section; for a detailed description see also [Per00].

In 1974, sharp resonances in e^+e^- annihilation at high energies were discovered, the fine structure of which seemed similar to the structure of positronium. The first spotted resonance – later named J/Ψ – was simultaneously found at the Stanford Linear Accelerator Center (SLAC) and the Brookhaven National Laboratory (BNL) by groups headed by B. Richter and S. Ting, respectively. They were awarded the Nobel Prize in Physics in 1976 because of this discovery.

The J/Ψ could not be explained with u, d or s quarks due to the extreme narrowness of its resonance. Thus it was identified as a bound state between a, previously postulated, charm (c) quark and its antiquark ($c\overline{c}$) and named *charmonium*. Another narrow state was identified as the excited state Ψ'. The narrowness can be explained by the necessity of change in quark flavor when the particle decays, which implies unconnected quark lines and is therefore suppressed (OZI rule). More excited states were found; however, these are broad states because their higher energy allows them to decay to $D\overline{D}$, a process without unconnected quark lines. Quark line diagrams of both cases are shown in Figure 2.2.

Similar resonances were seen in the mass region $9.5 - 10.5$ GeV/c^2 and identified as bound states between a bottom[6] (b) quark and its antiquark ($b\overline{b}$), consequently named *bottomonium*. The narrow states Υ, Υ' and Υ'' were identified and, analogous to charmonium, broader states at higher energies exist. The properties of these narrow quarkonia states can be seen in Table 4.1 in the section: *Quarkonia Measurements* (page 26).

In heavy-ion collisions, b and c quarks are not available in the colliding nucleons and their production in any thermal system is unlikely[7] because of their high mass. Quark-antiquark pairs are therefore produced at a very early stage of the collision by hard, pre-thermal interactions [Sat90]. In confined hadronic matter, these pairs form bound states, for example J/Ψ or Υ. In the deconfined quark-gluon plasma they just drift apart. At freeze-out the partners are separated and combine with lighter quarks which are available at higher densities. This process results in open charm or open beauty states whose rates therefore increase. Unfortunately, this change cannot be measured due to their initial very high rates, but the suppression of the quarkonia states should be measureable. Important is the fact that fast quark-antiquark pairs can leave the quark-gluon plasma before they are separated too far and still form bound states. This leads to an energy-dependency of the suppression, which can be analyzed experimentally as a dependency on the transverse momentum p_t. In longitudinal direction the quarks move with the slices of the quark-gluon plasma and thus a measurement of the dependency

[6]The bottom quark is also referred to as *beauty* quark.

[7]The production rate of a quark with the mass m_q in a thermal system with the temperature T is governed by the Boltzmann factor $\exp(-m_q/T)$.

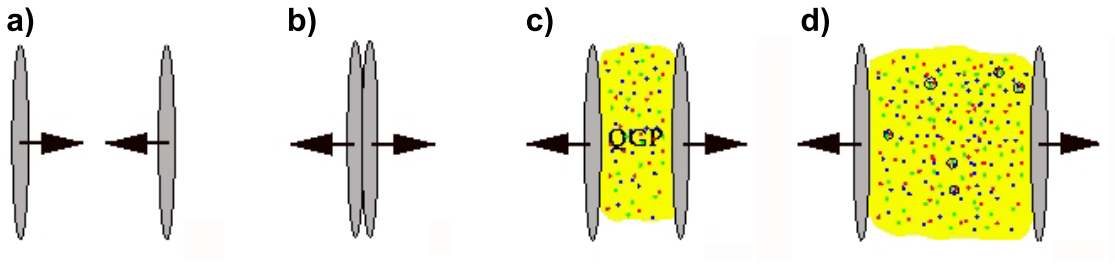

Figure 2.3: The formation and expansion of the quark-gluon plasma

on the longitudinal momentum is not that easy (see Figure 2.3 that shows the forming of the quark-gluon plasma). A measured p_t dependency also allows to calculate the plasma size and lifetime.

An alternative explanation of quarkonia suppression, that does not involve a new state of matter, is *absorption*. The high density of matter leads to interactions between, for example, J/Ψ and constituents of the hadronic matter such as baryons and mesons. This eventually results in reactions of the form $c\overline{c} + q\overline{q} \rightarrow c\overline{q} + \overline{c}q$, which reduces the number of quarkonia states. As mentioned above, the observation of several signatures of the quark-gluon plasma in a heavy-ion collision would provide strong hints if a new state of matter was found or other explanations apply.

The collider providing such heavy-ion collisions and the experiments measuring their outcome will be explained in the following chapter.

3. The Large Hadron Collider

Introduction

The Large Hadron Collider (LHC) is a particle accelerator of a new scale currently being built at the European Organization for Nuclear Research (CERN) in Geneva. Two beam pipes with a circumference of 27 km are designed for accelerating particles in opposite directions. The LHC is capable of speeding up protons to a center of mass energy[1] of 14 TeV per proton-pair and lead ions (Pb) to 5.5 TeV per nucleon pair. Five experiments at four different positions will measure the outcome of the collisions. The LHC exceeds energies at RHIC by a factor of 30 which opens a new physics domain and, concluding from historical situations with similar energy improvements, will most likely lead to new discoveries. LHC's superconducting magnets, running at a current of 12,000 A and cooled to 1.9 K, produce a magnetic field of 8.36 T. Bunches of protons collide every 25 ns, producing 10^9 events per second which corresponds to $10^{11} - 10^{12}$ tracks per second. This makes high demands on detector components and the computing infrastructure. Per year, the LHC will provide about seven months of protons and one month of heavy-ions.

The experiments examine various physics topics: ATLAS[2] and CMS[3] will search for the Higgs particle, which generates mass, and particles predicted by supersymmetric extensions of the Standard Model. LHCb[4] will study CP-symmetry violating processes in heavy b quarks. TOTEM[5] will measure the total cross section, elastic scattering and diffractive processes of LHC collisions. Finally, ALICE[6] is dedicated to analyze strongly interacting matter, explore the phase transition to the quark-gluon plasma, its phase diagram and its physics. The role of chiral symmetry in the generation of mass will also be studied. The ALICE experiment will be the topic of the next section.

[1] The energy in a collision is measured by the energy of the two colliding particles in their center of mass system. In case of colliding protons denoted by $\sqrt{s}$; in case of ions the energy is given per nucleon pair and denoted by $\sqrt{s_{NN}}$. See also section B of the appendix.

[2] ATLAS – A Toroidal LHC Aparatus.

[3] CMS – Compact Muon Solenoid.

[4] LHCb – The Large Hadron Collider beauty experiment.

[5] TOTEM – Total and Elastic Measurement.

[6] ALICE – A Large Ion Collider Experiment.

All LHC experiments can contribute to the field of astrophysics by providing a deeper understanding of cosmic-ray interactions at high energies, especially around the knee[7].

3.1 The ALICE Experiment

ALICE – A Large Ion Collider Experiment – is a general-purpose experiment which will be able to identify hadrons, leptons and photons in a broad range of transverse momenta and in an environment of large charged-particle multiplicities of up to 8000 per unit of rapidity[8] at mid-rapidity (dN_{ch}/dy). This will allow reconstruction of short-lived particles like hyperons and D and B mesons including their decay vertices.

ALICE will take data from proton-proton and nucleus-nucleus collisions. Although the emphasis is on data from Pb-Pb collisions, lighter ions will be studied to analyze the energy-density dependency of the measurements. For this purpose proton-nucleus collisions will provide reference data, just like colliding protons. Although ALICE mainly will study strongly interacting matter in ultrarelativistic heavy-ion collisions, a dedicated programme will concentrate on proton-proton physics.

A schematic view of the detector can be seen in Figure 3.1. The detector system at central rapidity – called *central barrel* – is able to identify hadrons, electrons and photons from very low transverse momenta, around 100 MeV/c, to large momenta of 100 GeV/c. It consists of the Inner Tracking System (ITS) featuring six layers of high-resolution silicon detectors, the Time Projection Chamber (TPC) as the main tracking system of the experiment, the Transition Radiation Detector (TRD) which provides electron identification, and the Time Of Flight (TOF) detector for particle identification. These detectors have full azimuthal[9] and central rapidity ($|\eta| < 0.9$) coverage. The design also includes two small-area detectors: an array of ring-imaging Cherenkov detectors for identification of high-momentum particles (High-Momentum Particle Identification Detector – HMPID) and an electromagnetic calorimeter made of high density crystals (Photon Spectrometer – PHOS). The central barrel is covered by a magnetic field of a maximum 0.5 T.

Tracking is performed by the ITS, TPC, TRD and TOF. Particle identification is based on energy loss measurements of the tracking detectors, transition radiation measure-

[7]The flux of cosmic rays as a function of the cosmic ray energy shows an exponential behavior. The slope changes between $10^{15} - 10^{16}$ eV, which, due to the shape in a double-logarithmic scale, is called *knee*.

[8]Kinematic variables in high energy physics are described in section B of the appendix.

[9]For the definition of the ALICE coordinate system see section A of the appendix.

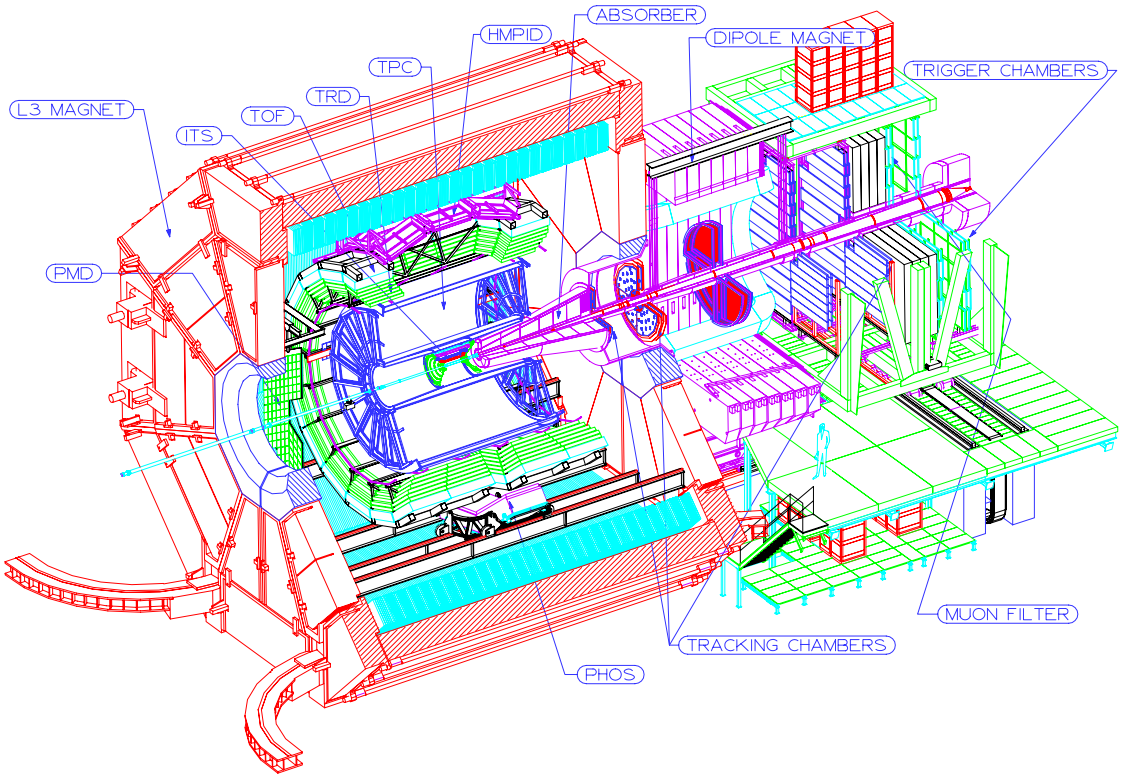

Figure 3.1: Schematic view of the ALICE detector

ments of the TRD, time of flight information from the TOF, Cherenkov radiation measured in the HMPID, and on photons detected in the PHOS.

At large rapidities the detector system consists of a muon spectrometer shielded by an absorber, and multiplicity detectors – the Forward Multiplicity Detector (FMD) and the Photon Multiplicity Detector (PMD). Trigger signals are provided by scintillators and quartz counters (T0 and V0). The impact parameter is measured by the Zero-Degree Calorimeter (ZDC). It consists of two sets of neutron and hadron calorimeters which are located about 100 m away from the interaction point in beam direction.

The main detectors of the ALICE central barrel will be explained in the following sections. For a detailed description of the ALICE design see also [ALI04a].

ALICE – being built by a collaboration of more than 1,000 members from 29 countries – will be completed and start data taking in 2007.

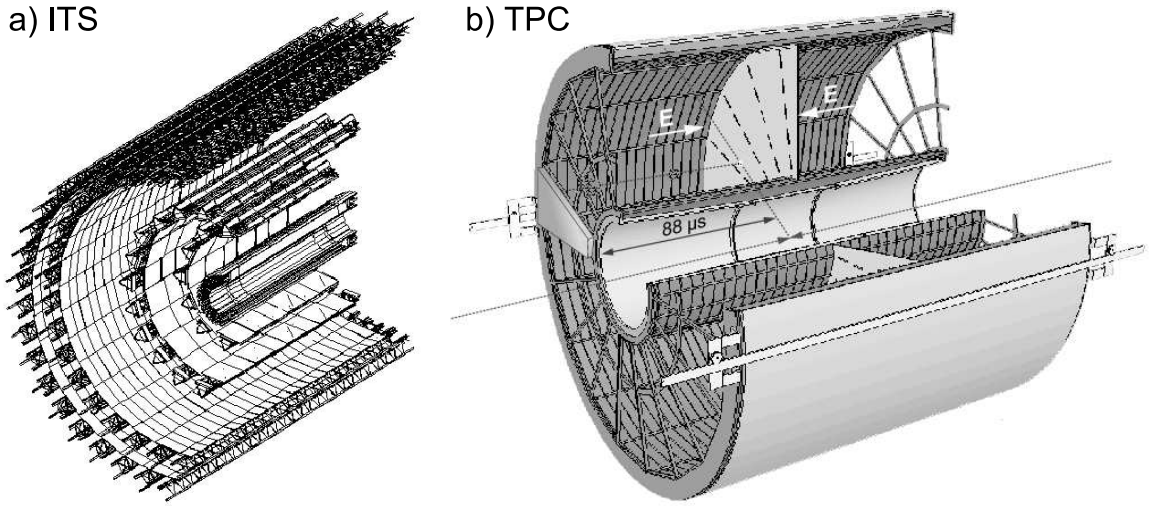

Figure 3.2: Schematic View of the ITS and the TPC

3.1.1 The Inner Tracking System

The Inner Tracking System (ITS) is located at a radius span from 4 cm, where particle densities of up to 80 particles per cm^2 are expected, to about 50 cm (see Figure 3.2a). It consists of 6 layers: the first two are pixel detectors, layer 3 and 4 are silicon drift detectors, and the outermost two are double-sided silicon micro-strip detectors. All layers cover central rapidity ($|\eta| < 0.9$), but the first has an extended coverage ($|\eta| < 1.98$) to support the measurement of the charged-particle multiplicity. The outer four layers have analog readout capabilities for particle identification by energy loss [ALI99, ALI04a].

The tasks of the ITS are to localize the primary vertex with a resolution better than 100 μm and to reconstruct secondary vertices of hyperons, D and B mesons. It provides tracking and particle identification of particles with low momenta (< 100 MeV/c) which do not reach the TPC. Furthermore the ITS information improves the resolution of the overall tracking and allows to track particles which traverse dead zones of the TPC.

3.1.2 The Time Projection Chamber

The Time Projection Chamber (TPC) is located at a radius span from 0.85 m to 2.5 m with a length of 5 m and is thus the biggest detector in ALICE (see Figure 3.2b). The large cylindrical field cage with a volume of 88 m^3 is covered by a highly uniform electrostatic field. The field is created between a central high-voltage electrode and the

two end-caps of the cage. Multi-wire proportional chambers mounted on the end-caps measure electron clusters which are created by the traversing particles [ALI00, ALI04a].

The TPC is the main tracking detector and together with the other detectors of the central barrel enables precise momentum measurements of charged particles. In this process particle identification and vertex determination is performed which requires the matching of track candidates with hits in the ITS. The TPC is capable of measuring up to 20,000 charged primary and secondary particles per event, which are expected at a multiplicity of $dN_{ch}/dy = 8000$. It covers central rapidity for particles with full radial length[10]; if only $\frac{1}{3}$ radial track length is required the coverage is $|\eta| < 1.5$.

3.1.3 The Transition Radiation Detector

The Transition Radiation Detector (TRD) is located in a radius span between 2.9 m and 3.7 m from the interaction point. It is divided into 18 supermodules in the azimuthal direction which consist of 5 stacks in z direction. Each stack has 6 layers, resulting in a total of 540 modules. Each module consists of a radiator, to create transition radiation, a drift chamber and read-out electronics [ALI01]. The drift chamber is divided into a drift and an amplification region. The structure of a module is depicted in Figure 3.3.

Apart from supporting the tracking, the TRD is the centerpiece for electron identification and trigger for momenta above $2 - 3$ GeV/c. This is of special importance for studies involving electrons[11], performed in this thesis. Therefore its working principle will be outlined in detail.

The detector is based on the measurement of *transition radiation* (TR). Transition radiation is created by charged particles which propagate through boundaries between media with different dielectric constants. The probability for the creation of TR is linearly dependent on the particle's Lorentz factor[12] γ. Ultra-relativistic particles create TR photons with similar wavelengths than x-rays. Therefore, the radiation aids in distinguishing between particles with the same momenta, but different γ. In the TRD, it is used to separate electrons from pions. However, the overall probability of the creation of a TR photon when a particle traverses a medium boundary is low. Therefore the radiator consists of many layers, which results in a high probability for the creation of a TR photon, when an electron traverses it. The number of layers cannot be arbitrarily increased due to saturation and interference effects.

[10] A track without *full radial length* leaves the TPC by one of the end-caps and therefore creates less electron clusters which leads to a less precise tracking.

[11] Arguments for electrons in the following chapters can be extended to cover positrons, as well. Nevertheless, for clarity, positrons are also named in single specific sections.

[12] $\gamma = \frac{1}{\sqrt{1-\beta^2}}$, $\beta = \frac{u}{c}$, u speed of the particle.

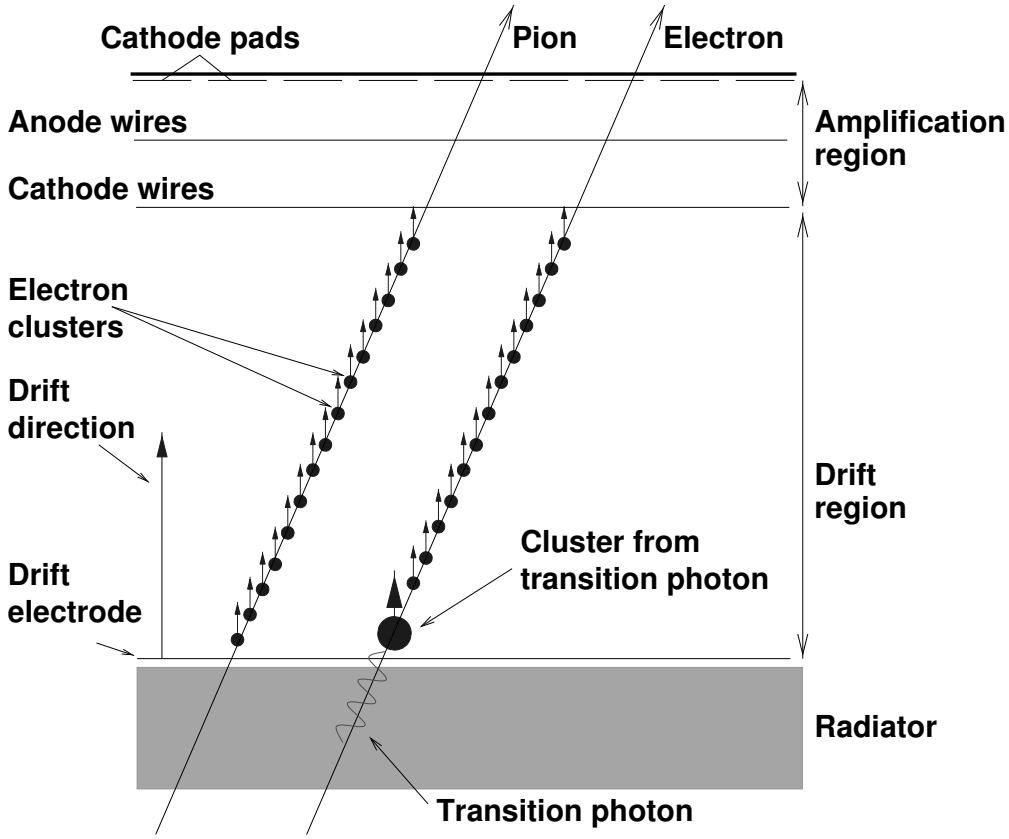

Figure 3.3: The structure of a TRD module

The module consists of a radiator, to create transition radiation, the drift chamber, divided into a drift and an amplification region, and the read-out section. Exemplarily, tracks of an electron and a pion are shown. The transition photon, created by the electron while traversing the radiator, is absorbed at the beginning of the drift region and creates a large cluster. This is specific to electrons and used for their identification.

A charged particle which traverses a TRD module first passes the radiator. Transition radiation is created dependent on the particle's Lorentz factor γ. Then the particle, if applicable together with a transition photon, enters the drift chamber. Both ionize the gas in the chamber and produce electron clusters. The photon is absorbed shortly after it entered the drift chamber, contrarily to the particle which constantly produces electron clusters on its way through the chamber and thus creates a track of clusters. The clusters drift towards the amplification region, where they are accelerated and collide with gas atoms, which produces further electrons. On the way to the anode wires, the electrons thus form avalanches. These create induced charges in the cathode pads which are then read out. Not only the amount of charge is detected, also the time when it arrived. Figure 3.4 shows average drift spectra for electrons and pions. In the

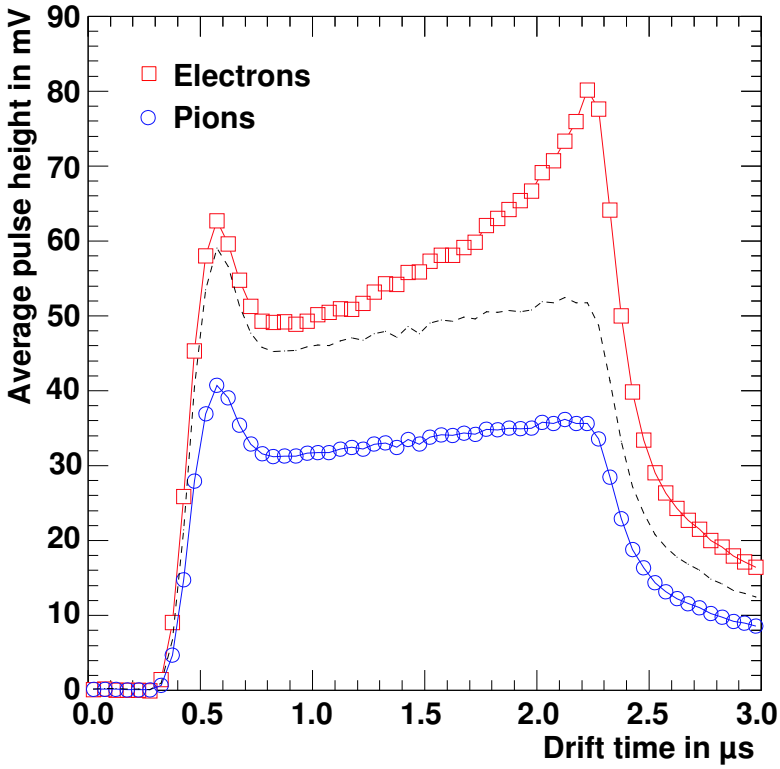

Figure 3.4: Averaged drift spectra of electrons and pions

Averaged drift spectra of electrons (red) and pions (blue) are shown. Both show a peak at the beginning of the drift time. It is explained by the fact that charges which are created before and after the anode wires have the same drift time. The overall energy loss of electrons is larger than that of pions which can be calculated by the *Bethe-Bloch formula* (see [Per00]). Furthermore the drift spectrum, which would be expected for electrons without transition radiation, is depicted (black, dashed line).

spectrum of the electrons, a peak caused by transition radiation at the end of the drift time can be clearly seen. The cluster from the transition radiation photon is created at the beginning, thus it has to drift the longest time and is measured after all other clusters, at the end of the drift time.

The read-out pads are divided in rows and columns. When data is taken, every pad is read out during the total drift time in bins of 100 ns. Thus the data from the TRD is characterized by the four coordinates: module, row, column and time bin.

3.1.4 The Time Of Flight Detector

The Time Of Flight detector (TOF) is located at a radius of 3.8 m with a coverage and module structure analogous to the TRD. It consists of a high resolution array of so-called *Multigap Resistive-Plate Chambers*. These feature a high, uniform field over their whole sensitive volume. Ionization therefore immediately causes an avalanche which can be measured. The fact that no drift time occurs results in a very high time resolution [ALI02, ALI04a].

The TOF provides particle identification by measuring the time of flight of the particles. With a time resolution of 150 ps, particles from the momentum range of $0.2-2.5$ GeV/c, which covers the majority of produced charge particles, can be identified. Together with the ITS and TPC, it will identify large samples of pions, kaons, and protons. It does not exceed an occupancy of $10-15\%$, even at highest multiplicities of $dN_{ch}/dy = 8000$.

3.1.5 Computing Requirements

The number of collisions and their products far exceed previous experiments. Enormous demands are made on the computing infrastructure, including online (during the physics run and data taking) and offline (after data taking) analyses.

In the ALICE Computing Technical Design Report [ALI05b] this is described as follows: a Pb-Pb collision rate of $4 \cdot 10^3$ Hz is expected. Out of these, data can only be stored at a rate of 100 Hz. Consequently a detailed online trigger system is crucial to save the "correct" events for physics analysis. Recording 100 events per second results in a recording bandwidth of 1,250 MB/s, which corresponds to about two CDs per second. One month of heavy-ion collisions and seven months of proton collisions lead to a total of 5.5 PB[13] per year that need to be stored. This amount, which is equal to about 8 million CDs, includes results from Monte Carlo simulations which are necessary to understand the measurements of the experiment.

Alignment, calibration and analysis of the data, including necessary simulations, are expected to need about 37 MSI2K[14] of computing power as the goal is to analyze the data well before the next physics run starts. This corresponds to about 37,000 of today's typical computers. Still, a reconstruction time of 4 months is expected for the heavy-ion data.

[13] 1 Petabyte (PB) = 1,024 Terabyte (TB) = 1,048,576 Gigabyte (GB).

[14] 1 MSI2K = 10^6 SPECint2000 – a benchmark specification for the integer processing power of CPUs [Spe99]. 1 MSI2K approximately corresponds to 1,000 of today's typical computer.

It was realized by the high energy physics community that both the amount of computing power and storage cannot be provided by one single institute. In the so-called *Monarc* model [MON99] centers called *Tiers* provide computing resources; the Tiers are arranged in a hierarchical way where Tier-0 is CERN, major and smaller computing centers are Tier-1s and Tier-2s, respectively, university computing centers are Tier-3s and, finally, the users' workstations are Tier-4s. The software infrastructure supporting this model is based on the concept Grid which is to provide efficient and seamless access to world-wide distributed computing and storage resources. In a final stage analysis tasks are to be sent to the Grid at single entry points and propagate transparently[15] to suitable resources. The Grid and the benefits for the high energy physics community will be outlined in detail in chapter 5 of this thesis.

[15] A program or system is called *transparent* when the user does not have to worry about technical details. By appearing like a single system, the fact that the Grid consists of many resources at different physical locations should not result in a higher usage complexity.

4. Physics Performance of the ALICE Central Barrel

Introduction

The ALICE detector is designed to search for traces of the quark-gluon plasma in heavy-ion collisions. One of the signatures is a change in the yield of quarkonia states. Their rates are expected to show different behavior in dependence of certain variables like the temperature T.

Consequently, it is of major concern to know what signals are to be expected. So-called *performance studies* show which particle types ALICE will be able to measure. They also give information about the efficiency for detecting certain particles, which is the ratio of particles tracked by the detector and particles created. In turn the efficiency can be used to conclude the occurred number of particles from the detected number of particles. The efficiency varies for different particles and energies.

To enable these analyses, especially of quarkonia states, the physics performance of the ALICE central barrel will be determined. The analysis approach will be shown below. First an introduction to quarkonia measurements will be given and key terms such as *slow simulations*, *fast simulations* and *response functions* will be explained.

4.1 Quarkonia Measurements

The quarkonia states which are to be studied by ALICE are the charmonium ($c\bar{c}$) states J/Ψ and Ψ' and the bottomonium ($b\bar{b}$) states Υ, Υ' and Υ''.

These states decay in various hadronic and radiative channels that are not suitable for analysis because most of the branching ratios are very low. Furthermore the resulting sets of particles mostly consist of 3 or more particles which makes the reconstruction of the originating particle a complicated task. Therefore, the leptonic decay channels remain for analysis. ALICE will be able to track particles from the dielectron ($X \rightarrow e^+e^-$) and dimuon ($X \rightarrow \mu^+\mu^-$) decay channels. As mentioned before, leptons do not undergo strong final state interactions and thus pass the surrounding environment

State	Mass in GeV/c^2	Width in keV/c^2	$\rightarrow e^+e^-$ in %	$\rightarrow \mu^+\mu^-$ in %
J/Ψ	3.10	91.0	5.93 ± 0.10	5.88 ± 0.10
Ψ'	3.69	281	0.76 ± 0.03	0.73 ± 0.08
Υ	9.46	53.0	2.38 ± 0.11	2.48 ± 0.06
Υ'	10.0	43.0	1.34 ± 0.20	1.31 ± 0.21
Υ''	10.3	26.3	seen	1.81 ± 0.17

Table 4.1: Properties and branching ratios of different quarkonia states [Eid04]

without being influenced, which allows an undistorted measurement. The branching ratios of the leptonic channels can be seen in Table 4.1.

The states which are to be studied have lifetimes of the order of 10^{-20} s, implying that their decay takes place in the close proximity ($< 10^{-11}$ m) of the interaction point. The decay products have to be tracked by the detector to make it possible to reconstruct the originating particle.

The study of the dielectron channel is based on the measurement of electrons and positrons. These are detected by the central barrel of the ALICE detector. Most important in this process is the TRD which provides electron identification and trigger for high energetic electrons. For the measurement of J/Ψ and Ψ' the vertex capabilities of the ITS are similarly important. These allow distinction between particles created in the collision and secondaries from B meson decays.

ALICE will also study the dimuon decay channel which is made possible by the muon spectrometer and additional components. These are located at large rapidities and therefore capable of measuring decay products of quarkonia states with low momenta by detecting Lorentz-boosted muons above 4 GeV/c.

This thesis focuses on the measurement of the dielectron channel. The analysis of the dimuon channel is for example discussed in [ALI05c].

4.2 The Concept of Slow Simulations

The results in this thesis are based on detailed simulations. The straightforward approach to a simulation consists of the following steps:

- **Generation**: An interaction is simulated. Particles with initial momenta and energies are output of this step. The piece of software performing this step is called *event generator*. Some event generators use simple parameterizations; others use complex theoretical models. A typical event generator for heavy-ion collisions is the Heavy Ion Jet INteraction Generator (HIJING) which is based on the expectation of dominating hard or semi-hard parton scattering with transverse momenta of a few GeV/c [Gyu94].

- **Simulation**: The propagation of all particles created in the collision is simulated through the detector. In this process, particles can decay, interact with matter or create additional particles which are then simulated through the detector as well. The total number of particles after the simulation step of a typical event is of the order of three to four times the number of particles created at the interaction.

  Furthermore, all hits with detector parts are recorded. If the detector produced a signal when it is exposed to a hit, the corresponding digital output is calculated and recorded. The output of this step corresponds to the output of the detector in case of an interaction of this kind within the detector.

- **Reconstruction**: The combined output from all detectors is used to identify tracks. In this process momenta and vertices are determined. Particle identification is performed and has probabilities for several particle types as a result.

This simulation method covers the full chain of the simulation. It consumes a considerable amount of computing time and is therefore called *slow simulation*. On a typical CPU it needs between 2 and 7 hours for a Pb-Pb HIJING event at LHC energies, strongly dependent on the type of event and the question if all detectors or e.g. only the central barrel is simulated.

4.3 The Concept of Fast Simulations

The rate of produced quarkonia states in a heavy-ion collision is very low. Thus many events are needed to get a significant signal. The amount of particles that are created in average per event and decay to e^+e^- is shown in Table 4.2 for different centrality classes. Per year, which corresponds to one month of Pb-Pb interaction, ALICE will record of the order of 10^8 Pb-Pb events. The quarkonia rates are highest for most central collisions; of the order of 10^7 events are expected per year in the most central class. Simulating these, with an average slow simulation time of 3 CPU hours for one event, would lead to a total computing time of about 3,400 CPU years. Even with

Centrality in %	J/Ψ / Event	Ψ' / Event	Υ / Event	Υ' / Event	Υ'' / Event
$0-10$	$2.0 \cdot 10^{-2}$	$3.8 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$4.5 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$
$10-30$	$9.6 \cdot 10^{-3}$	$1.8 \cdot 10^{-4}$	$7.8 \cdot 10^{-5}$	$2.1 \cdot 10^{-5}$	$6.5 \cdot 10^{-6}$
$30-50$	$3.1 \cdot 10^{-3}$	$6.0 \cdot 10^{-5}$	$2.4 \cdot 10^{-5}$	$6.7 \cdot 10^{-6}$	$2.0 \cdot 10^{-6}$
$50-70$	$7.4 \cdot 10^{-4}$	$1.4 \cdot 10^{-5}$	$5.5 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$	$4.6 \cdot 10^{-7}$
$70-100$	$9.7 \cdot 10^{-5}$	$1.9 \cdot 10^{-6}$	$6.7 \cdot 10^{-7}$	$1.8 \cdot 10^{-7}$	$5.6 \cdot 10^{-8}$

Table 4.2: Quarkonia rates including branching to e^+e^-

Shown are the rates of states of the quarkonia family which decay to e^+e^- for different centrality classes [Som05a]. The rates are based on the Glauber model which calculates the number of nucleus-nucleus collisions in a Pb-Pb interaction. The rates take into account that nucleons have a different structure function when being part of a nucleus instead of moving freely (*nuclear shadowing*).

today's computer clusters, which consist of over 100 processors, the simulation would finish, at the earliest, years after ALICE has started collecting data.

The solution for this problem is a different type of simulation: A faster simulation which in contrast to the previously described slow simulation does not perform the whole cycle. The simulation and reconstruction steps are replaced by lookups in so-called *response functions*. This process is depicted in Figure 4.1. Basically, response functions answer the following questions:

- Is the particle detected?
- What is the resolution of the detection?
- Is the particle identification correct?

The response functions, also referred to by *lookup tables* (LUTs), are based on results of slow simulations and will be explained in detail in the next section.

A lookup in a response function is much faster than a complete simulation cycle. Therefore the average time for one event will be reduced to less than a second. The above mentioned 10^7 events can be created in a reasonable time of 0.32 CPU years. On a typical computing cluster this corresponds to a time span of slightly more than one day. This kind of simulation is thus called *fast simulation*.

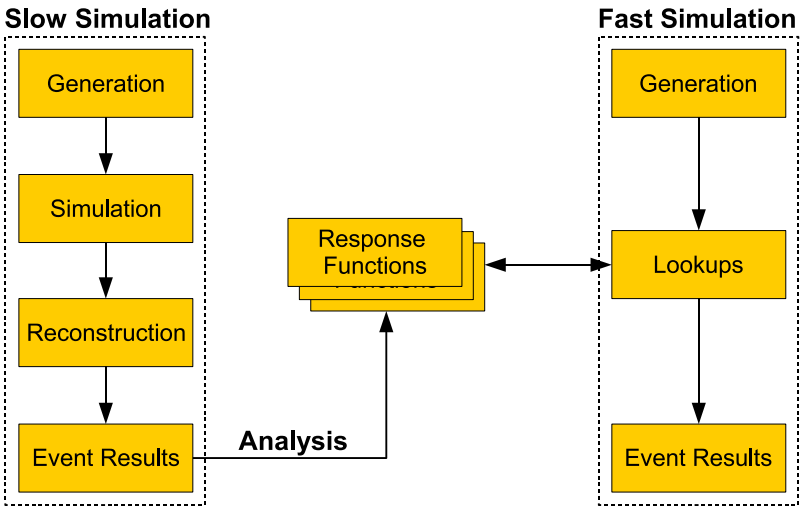

Figure 4.1: The scheme of slow and fast simulations

Response functions are created from slow simulations, which consist of three steps: generation, simulation and reconstruction. Fast simulations replace the simulation and reconstruction step with response function lookups. This process is significantly faster.

4.4 Response Functions

The study of the dielectron channel of quarkonia states necessitates response functions for electrons. The following functions are needed to parameterize the behavior of the detector:

- **Efficiency**: The efficiency describes the probability that a particle is detected and reconstructed by the detector. Theoretically all particles in the acceptance range[1] should be tracked by the detector. However, detectors have dead areas and the support structure might stop or deflect particles.
- **Resolution**: The resolution describes the accuracy of the reconstructed physical values and is based on $\Delta x = x_{Generated} - x_{Reconstructed}$, where x is a physical value. Naturally, these will rather turn out as probability distributions than as fixed values, which are then parameterized.
- **Electron-Pion Separation Efficiency**: The electron-pion separation efficiency gives the probability that an electron is falsely identified as pion and vice-versa. The pion background[2] is very intense and mis-identified pions have impact on the

[1] The *acceptance range* is the area that is covered with sensitive detector parts.
[2] The number of expected pions is of the order of 10^4 higher than the number of electrons.

reconstruction of particles. Energy loss measurements and time of flight information are not suitable to distinguish electrons and pions above $2 - 3$ GeV/c. This makes it one of the main tasks of the TRD to differentiate them by measuring transition radiation.

For the study of dielectron channels it is important to know the abilities of separating electrons and pions. Other particles, such as kaons or protons, may also be mis-identified as electrons. But their rate is much lower than the rate of pions and they are usually well-identified by the TPC and TOF. Therefore it is not necessary to take them into account for the response functions.

Response functions will be created in a transverse momentum range from 1 GeV/c to 10 GeV/c. This will cover the main range of decay products of quarkonia states. Below 1 GeV/c the background caused by mis-identified pions increases drastically and thus it is not advisable to take this region into account. Therefore most fast simulations for quarkonia states do not consider particles below 1 GeV/c. Approximations for transverse momenta above 10 GeV/c will also be created.

The initial momentum of a particle can be characterized by the variables[3] p_t, Θ and ϕ. The response functions will be parameterized using these variables.

4.4.1 Event Selection

The response functions are based on results of events produced by slow simulations. The heavy-ion interaction is simulated by the HIJING event generator. The transverse momentum spectrum of electrons and positrons[4] of a typical HIJING event is shown in Figure 4.2. Only a few particles have a transverse momentum above 1.5 GeV/c. After performing the necessary cuts, which will be explained in section 4.4.4 along with their effects, hardly any particle has a p_t above 0.5 GeV/c. But response functions are to be created for particles up to 10 GeV/c, so that electrons and positrons with higher transverse momentum are embedded into an HIJING event. Further below it will be made sure that these embedded particles do not falsify the results of the events which are then input for the response functions. The particles are added in the acceptance

[3]For the definition of the ALICE coordinate system see section A in the appendix. Furthermore section B defines kinematic variables like p_t.

[4]For clarity, in this section electrons and positrons are named explicitly. Subsequent sections will follow the convention again, that arguments for electrons can be extended to cover positrons, as well.

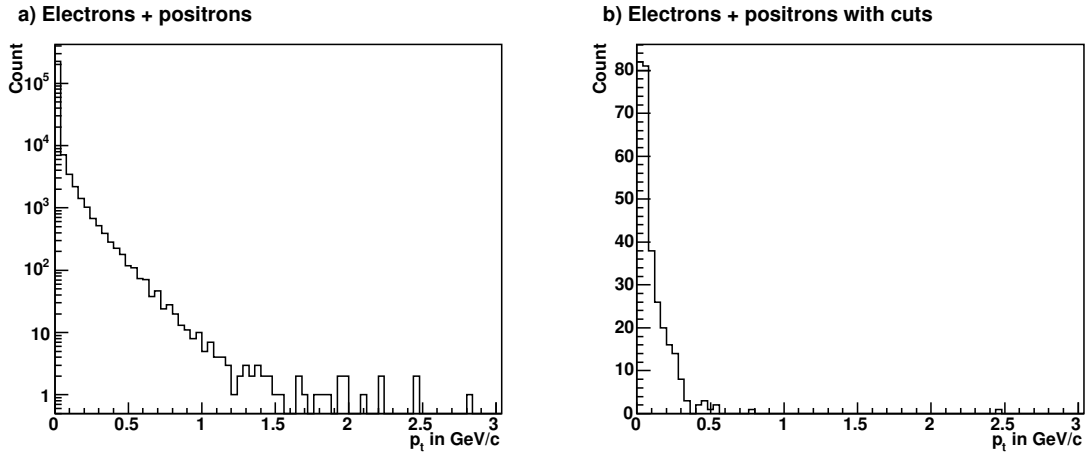

Figure 4.2: Transverse momentum spectrum of a HIJING event
All electrons and positrons of a HIJING event are shown in (a). After performing the necessary cuts (see section 4.4.4) the particles shown in (b) remain. It should be noted that the scales are different in orders of magnitude. The event was generated by AliGenHIJINGPara [Mor01] with $dN_{ch}/dy = 4000$.

range of the central barrel ($|\eta| < 1$)[5]. There is no need to add particles outside the acceptance range because reconstruction will not be possible in this region. Therefore, the efficiency response function is set to 0, and other response functions cannot have meaningful values there. These events with embedded electrons and positrons are used for the creation of the efficiency and resolution response functions. A different approach is needed for the creation of the electron-pion separation efficiency which requires also pions with high transverse momenta. Therefore, for this case, electrons and positrons as well as positive and negative pions are embedded into a HIJING event. The simulations will be performed using the designated magnetic field for the experiment of 0.5 T.

Particles in the transverse momentum range of 1 GeV/c to 10 GeV/c mainly originate from the embedded particles and not from the HIJING event. Consequently it can be understood as background that provides realistic occupancies in the detector. Combined with the fact that the simulation of the collision takes a considerable amount of computing time it was decided to use a parameterized version of the HIJING generator: a generator called *AliGenHIJINGPara*. It is based on parameterized pseudo-rapidity density and transverse momentum distributions of charged and neutral pions and kaons

[5]Actually, the requirement for the acceptance of the central barrel is $|\eta| < 0.9$. However, at the "borders" of the acceptance range some particles are still tracked, therefore these are included in the response functions.

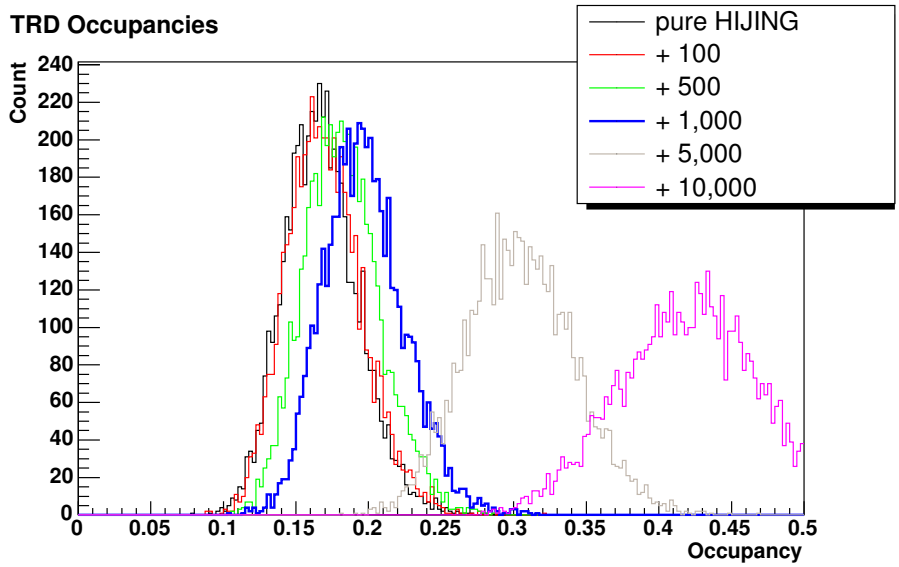

Figure 4.3: TRD occupancies for HIJING mixtures

The occupancy per detector module defined as the percentage of detector pixels (rows, columns, time bins) which have a signal. Different events are shown ranging from a pure HIJING event to a HIJING event with 10,000 embedded particles. It was decided to use the event with 1,000 embedded particles (shown in blue). The histograms contain one entry per module, the values of 10 events are accumulated.

[Mor01]. AliGenHIJINGPara is part of AliROOT [Car05], which is the analysis framework developed and used by the ALICE collaboration.

The response functions will be created for different multiplicities[6] dN_{ch}/dy. Multiplicities which are expected in heavy-ion collisions in ALICE are based on extrapolations; a dN_{ch}/dy between 3000 and 4000 seems to be the most promising candidate [ALI04a]. However, the accuracy of the estimations is unknown and the first LHC events will give an answer. For the time being $dN_{ch}/dy = 4000$ is used for most of the simulations. The maximum expected value is $dN_{ch}/dy = 6000$. Therefore, response functions for the multiplicities 4000 and 6000 will be created. In proton-proton collisions the multiplicity will be much lower. Response functions are created for this case by using events which contain only the embedded particles, and not the background in form of a HIJING event. This case will be referred to by "$dN_{ch}/dy = 0$". All results which are shown in the following sections are of the multiplicity 4000, if not noted otherwise.

Embedded particles are not supposed to falsify the response functions. This is verified in two different ways: Firstly the occupancies of the TRD detector are calculated for

[6] dN_{ch}/dy: Number of charged particles per unit of rapidity at mid-rapidity. In the following referred to by *multiplicity*.

Number of Embedded Particles	Occupancy in %	ΔOccupancy in %
0	16.8	
100	17.0	0.2
500	18.2	1.4
1,000	**19.7**	**2.9**
5,000	30.5	13.7
10,000	42.0	25.2

Table 4.3: Average occupancies of the TRD modules

The table shows the average occupancies for different events, the right column notes the difference in occupancy compared to a pure HIJING event. The case of 1,000 embedded particles is taken as input for the response functions.

a pure HIJING event and the mixtures. These values are then compared. Secondly it is shown that only a negligible amount of particles with high transverse momenta overlap.

TRD Occupancy

It is important that the embedded particles do not significantly increase the occupancies of the detectors, especially of the TRD acting as the centerpiece for the correct identification of electrons at high transverse momenta. Increased occupancy would result in a lower tracking efficiency and unclear results concerning particle identification. The occupancies of the TRD were calculated utilizing a method described in its Technical Design Report [ALI01] (section 11.3.2). This method defines the occupancy by the percentage of detector pixels having a signal above a threshold. For every pad, each time bin is considered and the occupancy is calculated for all 540 modules, which is shown in Figure 4.3. It shows the results of different events, these vary from a pure HIJING event to an HIJING event with 10,000 embedded particles. It was decided to use events with 1,000 embedded particles for the creation of the response functions. This on the one hand assures sufficient particles to have good statistics and on the other hand increases the average occupancy by only about 3%, which can be seen in Table 4.3.

500 electrons and 500 positrons are embedded in the events which are the basis of the efficiency and resolution response functions. For the electron-pion separation efficiency, electrons and positrons as well as positive and negative pions are embedded, 250 of each per event.

Overlapping

Usually the amount of electrons and positrons with high transverse momenta per event is very low. That is why, in a realistic event, the tracks of these particles propagating through the detector never overlap with each other. Although there is an unrealistically high number of particles with high transverse momenta in events with embedded particles, no tracks are to overlap. Overlapping tracks would e.g. imply an unrealistic reduction of the tracking efficiency. A track is considered overlapping in the TRD when its distance to another track is less than three full pad widths in ϕ and less than one full pad width in Θ [Ems05] during the passage trough the detector. Furthermore the transverse momenta p_t of probably overlapping track candidates must be close to each other, otherwise their bending radii are different and they only overlap for a small fraction of the path through the detector. The distance of two tracks in p_t is transformed to a distance in ϕ and also three full pad width distance are demanded.

With the mentioned conditions less than 10 tracks overlap per event. One event contains over 1,000 particles, so that there is a systematic error of less than 1%.

4.4.2 Dependency Considerations

Response functions in the form of $F(p_t, \Theta, \phi)$ are to be created. The values of the variables are stored in equidistant bins with typical bin widths of $W_{p_t} = 0.2$ GeV/c, $W_\Theta = \frac{\pi}{150}$ rad and $W_\phi = \frac{\pi}{45}$ rad. For each bin $(p_t, p_t + W_{p_t}; \Theta, \Theta + W_\Theta; \phi, \phi + W_\phi)$ the values of the response functions must be determined.

Assuming that for the efficiency response function about 1,000 particles are needed per bin, and that the values for the typical binning mentioned before are used, this would lead to an order of $10^8 - 10^9$ necessary particles. Even with distributed computing environments, it is only feasible to simulate of the order of $10^6 - 10^7$ particles. If the response functions separated, the number of needed particles would be significantly decreased. Therefore it will be analyzed if a separation is possible, in particular for the variables Θ and ϕ.

The procedure which is used will first be outlined in general. Assuming a function which separates

$$f(a, b) \equiv g(a)\, h(b), \tag{4.1}$$

and the following values as measurements:

$$f_1(b) := \int f(a,b)da, \tag{4.2}$$

$$f_2(a) := \int f(a,b)db. \tag{4.3}$$

It is possible to calculate $f(a,b)$ using f_1 and f_2 if Eq. (4.1) is valid:

$$\begin{aligned} f_1(b)\cdot f_2(a) &\overset{(4.2,4.3)}{=} \int f(a',b)da' \cdot \int f(a,b')db' \\ &\overset{(4.1)}{=} \int g(a')\ h(b)da' \cdot \int g(a)\ h(b')db' \\ &= g(a)\ h(b) \cdot \int g(a')\ h(b')da'db' \\ &\overset{(4.1)}{=} f(a,b) \cdot \underbrace{\int f(a',b')da'db'}_{=:c} \quad \text{with} \end{aligned}$$

$$\begin{aligned} c = \int f(a',b')da'db' &\overset{(4.2)}{=} \int f_1(b')db' \\ \text{or} \quad c &\overset{(4.3)}{=} \int f_2(a')da' \\ \Rightarrow f(a,b) &= \frac{f_1(b)\cdot f_2(a)}{c} \end{aligned} \tag{4.4}$$

To determine if Eq. (4.4) is valid, or in other words, if the function separates, an error function is defined as:

$$e = \int\int |\varepsilon(a,b)|\ da\ db \quad \text{with} \tag{4.5}$$

$$\varepsilon(a,b) = f(a,b) - \frac{f_1(b)\cdot f_2(a)}{c} \tag{4.6}$$

This sum of absolute errors is 0 if the function separates.

Obviously f has to be known to determine the error e. It was mentioned before that the separation is applied because the statistics are not good enough to calculate the response functions without separation. So it seems to be impossible to verify if the functions separate. However, an analysis if the separation is possible will turn out to be feasible because the response functions, contrary to the example above, have three variables.

The outlined procedure will be applied to analyze a possible separation of the response functions in Θ and ϕ. Therefore a and b are renamed Θ and ϕ, respectively, and the p_t-dependence is added in Eqs. (4.1 − 4.4). Results are:

$$F(p_t, \Theta, \phi) \equiv G(p_t, \Theta)\, H(p_t, \phi) \tag{4.7}$$

$$F_1(p_t, \phi) := \int F(p_t, \Theta, \phi)\, d\Theta \tag{4.8}$$

$$F_2(p_t, \Theta) := \int F(p_t, \Theta, \phi)\, d\phi \tag{4.9}$$

and

$$F(p_t, \Theta, \phi) = \frac{F_1(p_t, \phi) \cdot F_2(p_t, \Theta)}{C(p_t)} \quad \text{with} \tag{4.10}$$

$$C(p_t) = \int F_1(p_t, \phi)\, d\phi = \int F_2(p_t, \Theta)\, d\Theta =: F_3(p_t). \tag{4.11}$$

F is the response function which is to be gathered, F_1 and F_2 will be calculated on the basis of the simulated events and F_3 will appear as a projection of F_1 or F_2 on the p_t-axis. If the response functions separate in Θ and ϕ, Eq. (4.10) can be used to determine the value of a response function from the corresponding F_1 and F_2.

The error function, Eqs. (4.5 − 4.6), transform to

$$E(p_t) = \int\int |\epsilon(p_t, \Theta, \phi)|\, d\Theta\, d\phi \quad \text{with} \tag{4.12}$$

$$\epsilon(p_t, \Theta, \phi) = F(p_t, \Theta, \phi) - \frac{F_1(p_t, \phi) \cdot F_2(p_t, \Theta)}{C(p_t)}. \tag{4.13}$$

As stated before, the statistics are too bad to calculate $F(p_t, \Theta, \phi)$. But by integrating over p_t the statistics increase and an analysis is made possible. One could argue that the integration over p_t hides the fact that the separation is not possible, but this is not very intuitive. Applying the same procedure to the function F^* (the steps of this process are also shown in Figure 4.4), defined by

$$F^*(\Theta, \phi) = \int F(p_t, \Theta, \phi)\, dp_t, \tag{4.14}$$

leads to the error function

$$E^* = \int\int |\epsilon^*(\Theta, \phi)|\, d\Theta\, d\phi \quad \text{with} \tag{4.15}$$

$$\epsilon^*(\Theta, \phi) = F^*(\Theta, \phi) - \frac{F_1^*(\phi) \cdot F_2^*(\Theta)}{C^*} \quad \text{and} \tag{4.16}$$

F_1^*, F_2^*, C^* defined analogously to before. This in fact verifies if F^* separates to a G^* and H^*:

$$F^*(\Theta, \phi) = G^*(\Theta)\, H^*(\phi) \tag{4.17}$$

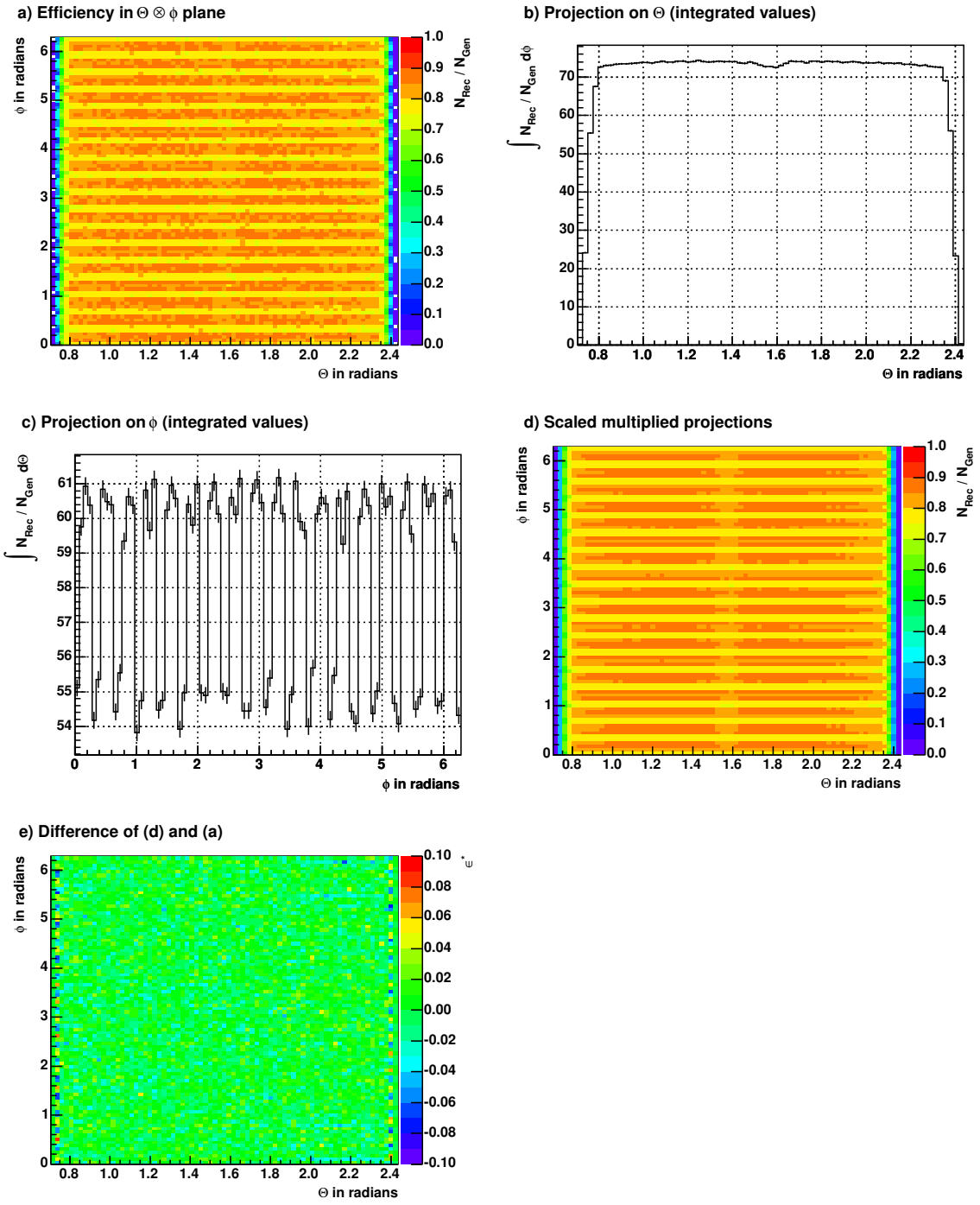

Figure 4.4: Separation verification

Shown are: (a) $F^*(\Theta, \phi)$ – the efficiency in the $\Theta \otimes \phi$ plane; (b) $F_2^*(\Theta)$ – the projection on the Θ-axis; (c) $F_1^*(\phi)$ – the projection on the ϕ-axis; (d) $\frac{F_1^*(\phi) \cdot F_2^*(\Theta)}{C^*}$ – the scaled multiplication of (b) and (c); (e) ϵ^* – the error function which shows that there are only minor differences between the function and the multiplied projections: It can be concluded that the function F^* separates in Θ and ϕ.

Function/Variables	$\tilde{E}^*$ in %
$\cos(x)\sin(y)$	$\sim 10^{-5}$
distorted $\cos(x)\sin(y)$	~ 1
statistical noise	50
p_t and ϕ	2.45
p_t and Θ	0.87
Θ and ϕ	0.78

Table 4.4: Error function analyzing the separation

The errors $\tilde{E}^*$ of artificial functions which separate or do not separate are shown along with the errors of the efficiency response function for different separation combinations of its three variables.

If F^* separates, and assuming that the integration over p_t does not generate or removes that the function separates in Θ and ϕ, it can be concluded that F separates as well. Thus Eq. (4.7) is valid. It should be noted, that this is not a strict mathematical proof, but can be seen as a strong hint that the function separates. However, response functions created by using the separation still need to be verified.

The integration over p_t resulted in a formula to analyze the separation of Θ and ϕ. Naturally, the integration could be also performed over Θ or ϕ which results in the analysis if p_t and ϕ, or p_t and Θ, respectively, separate.

In the shown calculations integrals were used. In practice these are replaced by sums and weighted according to the sum of absolute values of F^* to make sure that different functions can be compared.

$$\tilde{E}^* = \frac{1}{\sum_\Theta \sum_\phi |F^*(\Theta,\phi)|} \sum_\Theta \sum_\phi |\epsilon^*(\Theta,\phi)| \tag{4.18}$$

$\tilde{E}^*$ gives a weighted measure of the absolute deviation.

To estimate the error values which are to be expected when functions separate or do not separate, the errors of several artificial examples were calculated by using this method. Results are shown in Table 4.4. The error of the separating function $\cos(x)\sin(y)$, the latter function with a Gaussian distortion ($\sigma = 1\%$) embedded, and of statistical noise can be seen. Exemplarily, the analysis of a possible separation of the efficiency response functions is shown. Results are given for the three possible separation combinations of the variables p_t, Θ and ϕ.

The fact that the error of the Θ-ϕ separation, which is the smallest error, is of the order of the distorted separable function provides strong hints that this separation can be

used without implying an unacceptable systematical error. Another possibility would be to use the p_t-Θ separation. As pointed out before, the resulting response functions still need to be verified, which will be performed in section 4.4.9.

4.4.3 Simulation of Events

The events are generated, simulated and reconstructed with AliROOT in a distributed computing environment. The hardware is a cluster of the Institut für Kernphysik in Münster. Its features are 100 Opteron processors and 8 TB[7] of disk space. The operating system is Linux Redhat Fedora Core 3 [Fed05]. An event needs about 3 hours for generation, simulation and reconstruction, its results (generated particles, hit and digit information and reconstructed tracks) consume about 1.2 GB of disk space. Only the central barrel is simulated because the other detectors do not cover the whole acceptance range and give only minor input to the response functions.

For the creation of the efficiency and resolution response functions 5,000 events with embedded electrons are simulated for each multiplicity. This number of events corresponds to about $5{\cdot}10^6$ particles per multiplicity. Furthermore 750 events with embedded electrons and pions per multiplicity are simulated for the creation of the electron-pion separation efficiency response function. The overall process consumes for all multiplicities about 6 CPU years of computing time.

4.4.4 Extracting Information

The generated particles and the reconstructed tracks, including the information about their true values from the event generator, are extracted from the results of the events. The first ones will now be referred to by *particles*, the latter ones by *tracks*. Only information of electrons is extracted because only these particles are needed for the response functions. The latter is different for the creation of the electron-pion separation efficiency where information about pions is needed as well.

An event with the multiplicity 4000 and embedded electrons contains[8] about 230,000 particles. Out of these about 45,000 particles are created by the event generator during the primary interaction, others are created while the particles propagate and interact with the detector material. One event contains about 142,000 electrons and 11,000

[7] 1 Terabyte (TB) = 1,024 Gigabyte (GB).

[8] Exemplarily, in the following, the number of particles and tracks are given for the case of the processing of an event with multiplicity 4000 and embedded electrons.

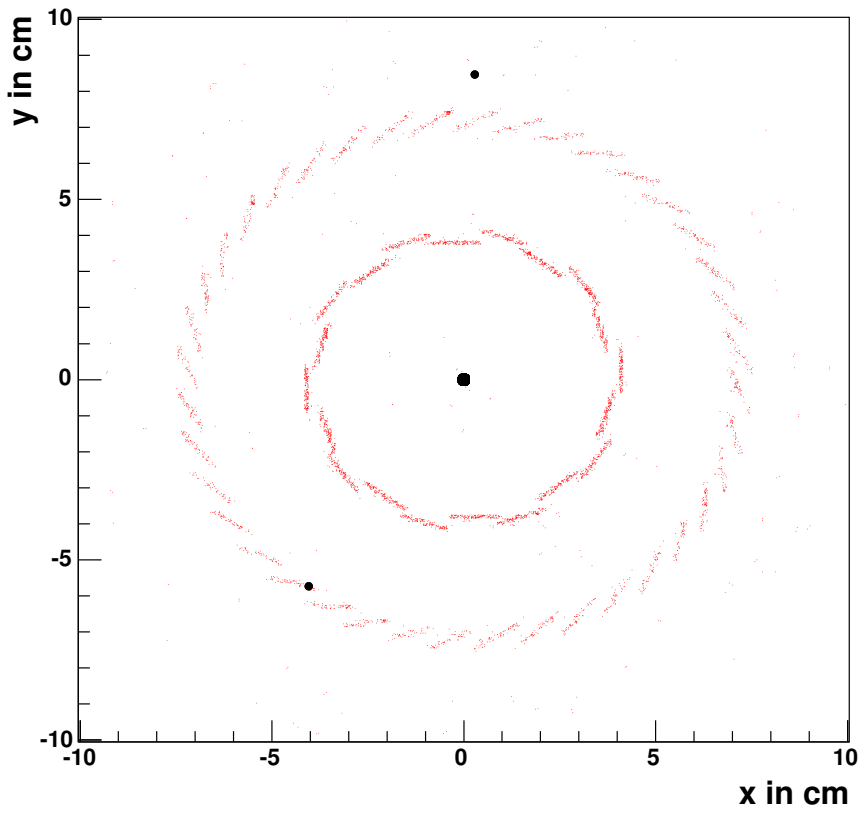

Figure 4.5: Production vertices of particles

Shown are the production vertices of particles of one event in the xy-plane. Only particles which are produced in a z-range of ±6 cm, but with any p_t are shown. Particles with a p_t below 1 GeV/c are marked in red, above 1 GeV/c in black (bigger markers for visibility). The interaction point and the structure of the ITS can be clearly seen. The latter is caused by conversion electrons, which are created while particles propagate through the detector. Most of the conversion electrons are low-energetic and removed by the p_t-cut. Only particles which originate from the interaction point are taken into consideration for the response function.

positrons. On average 10,000 tracks are reconstructed, 1,600 of these are electrons and positrons. Particles and tracks consume about 20 MB of disk space.

The extracted information is reduced by the following cuts.

- **p_t-Cut**: The transverse momentum region in which the response functions will be created begins at 1 GeV/c. Therefore only particles and tracks with a transverse momentum above this value are considered for further processing.

  This cut removes about 151,000 particles and 400 tracks.

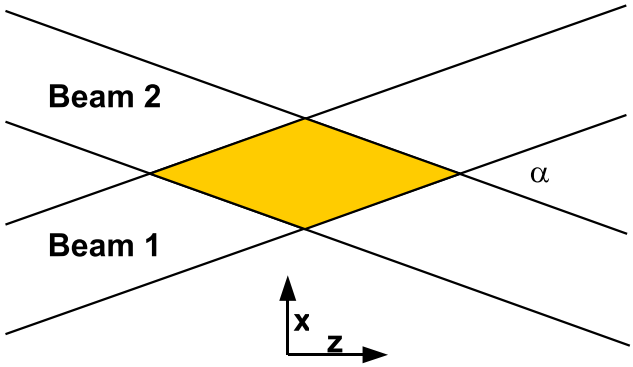

Figure 4.6: The interaction diamond
The LHC beams cross at an angle of $\alpha = 300$ μrad. This extremely small angle creates a diamond-shaped area of possible particle interactions point (colored area). For visibility, the angle between the beams in this figure is highly exaggerated.

- **Vertex Cut**: All particles and tracks which do not originate from the interaction point are removed, eliminating particles created when primary particles interact with the detector material. This cut is necessary because the response functions are to be a parametrization for particles which were created in the interaction point, therefore particles created elsewhere would falsify the results.

  The production vertices of the particles are plotted in Figure 4.5, which shows that particles are created in the interaction point and when interacting with detector material, producing mainly conversion electrons. Counting from the center the first interaction with material leading to particle creation is with the innermost layer of the ITS, located at a radius of 4 cm. The beam pipe is located at a radius of 3 cm, but due to the low atomic number Z of beryllium and the consequently small radiation length, the interaction with electrons is minor. The cut, performed on the x and y coordinates at ± 0.1 cm, removes all particles which were not created in the primary vertex.

  The two LHC beams, which cross at a very small angle of 300 μrad $= 0.0172°$ allow any position of the interaction point in the so-called *interaction diamond* (see Figure 4.6): Orthogonal to the beam direction – in x and y – the position is defined very well. On the contrary, parallel to the beam the possible interaction points are smeared out. This is also taken into consideration by the simulations: The z coordinate of the interaction point is randomized by a Gaussian distribution with $\sigma = 5.3$ cm, where the z coordinate is always in $\pm 1\sigma$. Therefore the cut removes all particles which are more than 6 cm off in their z coordinate.

Overall, no interaction with material is possible inside the vertex cut ranges. About 300 particles and 20 tracks per event are removed by the cut.

Another possibility than a vertex cut would be to consider only primary particles. However, the cut in this way can also be applied on reconstructed values and will be used later in the performed analysis.

- **Acceptance Cut**: Particles which cannot be tracked because they are not in the acceptance range and therefore do not hit the detector are removed. It is not necessary to gather information for the response functions outside acceptance range because results are either undefined or, like the efficiency, set to 0.

  Only a few particles per event are removed by this cut. This number would be significantly higher if all particles would be processed. However, all particles except electrons have been removed and other cuts have been performed before. No tracks are removed because particles can naturally not be tracked outside of the acceptance range.

- **Lost Tracks Cut**: Tracks which reconstructed transverse momenta differs more than 50% from their generated value are considered lost. The resolution of these tracks can be only badly parameterized and as they are of no use for reconstruction and analysis it is the best choice to remove them. This reduces the overall efficiency and therefore does not change the physical meaning of the response functions.

  This cut removes about 90 tracks.

- **No ITS and Double Track Cut**: The tracking is mainly performed by the TPC. Resulting track candidates are then matched with information from the ITS [Bel04]. However, some tracks cannot be identified with hits in the ITS and therefore have no ITS-specific information, such as the vertex position. This is a problem because in an interaction $J/\Psi s$ are not only produced in the primary collision; they can also originate from B meson decays. These have a lifetime of the order of 10^{-12} s and therefore decay at about 500 μm from the interaction point. Because the rate of B mesons is quite high, it is important to distinguish between primary J/Ψs and ones from B meson decay. If these would not be separated, the J/Ψ signal which is supposed to stem only from primary particles, would be distorted.

  The differentiation is performed by checking if the vertex of the J/Ψ is in the interaction point. Therefore, a very precise tracking is needed, which is only possible if the particle was also identified in the ITS.

The current version of AliROOT tracks a low percentage of particles several times. These tracks have to be removed, otherwise the efficiency values would be incorrect. It can be shown that the two track candidates have very similar tracking, therefore it is not important which candidate is kept. If there are two candidates of a track, usually only one of them is identified with information in the ITS.

This cut removes about 250 tracks per event.

After performing the cuts about 1,000 generated particles and 850 tracks per event remain. This consumes about 1 MB[9] of disk space.

The particles and tracks of all 5,000 events are merged, which amounts to about 5 GB of data. For performance reasons only necessary values are extracted and stored in so-called *tuples*. For each particle and track its momentum, parameterized by p_t, Θ and ϕ, the resolution of the reconstructed values, and the probability for being identified as electron and pion are saved. It should be noted that in case of tracks the momenta of the associated generated particles (true values from the event generator) are used. This is important because the response functions are to give information about particles which **have** given values, not for particles which **are reconstructed** at given values. Otherwise shifts in reconstructed values could falsify efficiency values.

One event consumes about 36 KB in tuples. For 5,000 events this corresponds to 180 MB, which can reasonably be processed in an acceptable period of time.

4.4.5 Analysis and Results

Efficiency

The efficiency is the percentage of particles tracked by the detector. As was outlined above it will be created in two planes: the $p_t \otimes \Theta$ plane and the $p_t \otimes \phi$ plane. For each of the planes two 2D-histograms are created. Their x-axis is p_t and their y-axis is Θ or ϕ (first or second histogram, respectively). Each $p_t, \Theta/\phi$-bin of the first histogram contains the number of particles which have been generated (Figure 4.7a); each bin of the second histogram the number of particles which have been reconstructed (Figure

[9] Although the number of particles was reduced by a factor of more than 100 and the number of tracks by a factor of 10, the used disk space reduced only by a factor of 20. This is due to the fact that tracks consume considerably more disk space than particles. Furthermore the data stored on the disk is compressed, which is less effective for small amounts of data.

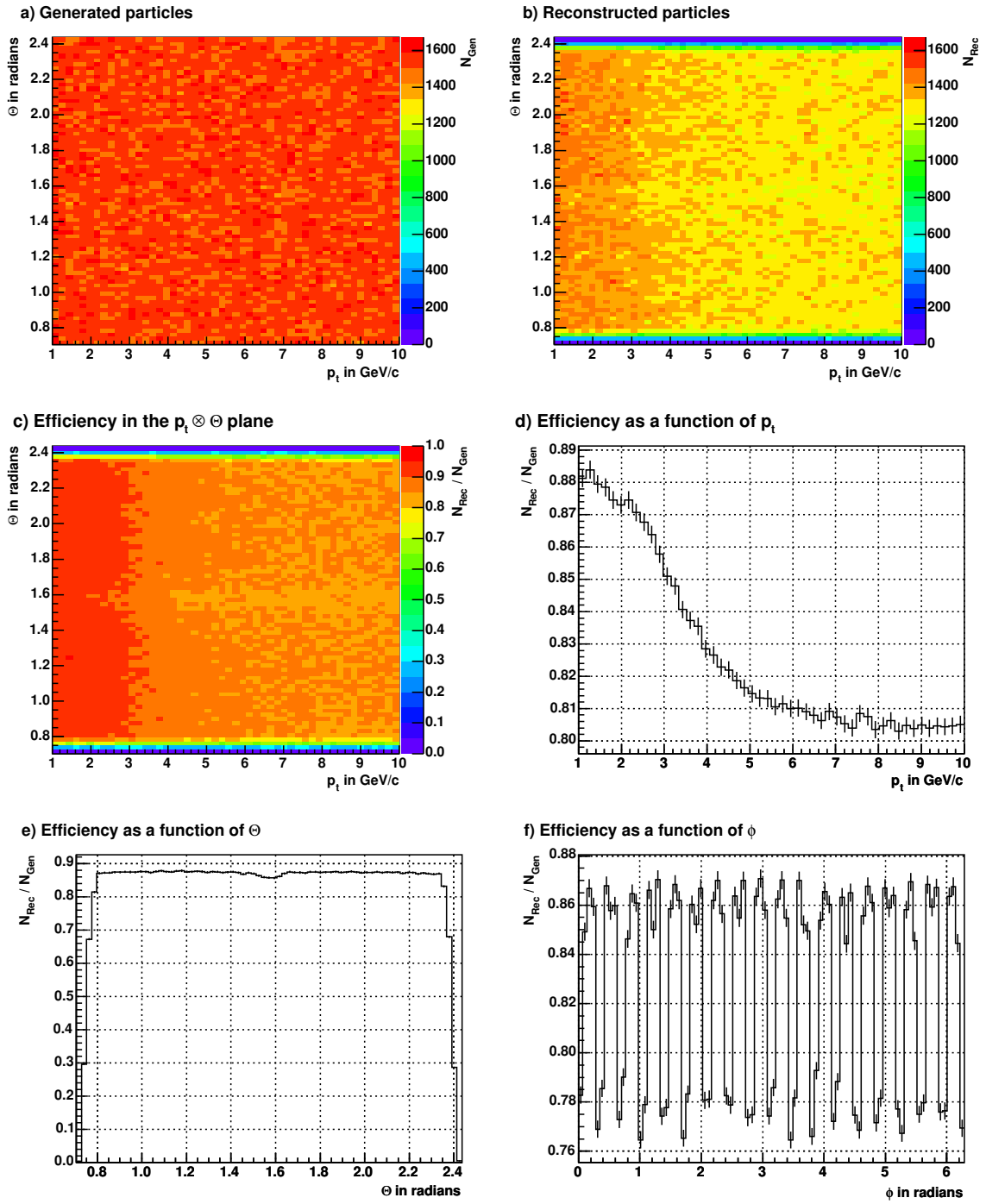

Figure 4.7: Efficiency response function

The process of creating the efficiency response function is illustrated exemplarily for the $p_t \otimes \Theta$ plane. (a) shows the number of generated particles and (b) the number of reconstructed tracks. Their quotient, which corresponds to the efficiency, is shown in (c). Furthermore the efficiency as function of p_t (d), Θ (e) and ϕ (f) is depicted.

4.7b). The second is divided by the first which corresponds to the efficiency N_{Rec}/N_{Gen} for each bin (Figure 4.7c).

Although 2D-histograms are needed for the response functions, plots as a function of only one variable are more suggestive. In Figure 4.7d-f the efficiency as a function of p_t, Θ and ϕ is shown. The following aspects can be noted:

- **In $\mathbf{p_t}$**: The efficiency decreases by about 7% between 1 and 5 GeV/c. This is caused by a higher influence of dead zones with increasing momentum. A track that propagates through such a region does not produce hits with sensitive detector parts and cannot be tracked. A particle with low momenta propagates out of a dead zone before it leaves the detector and can be tracked. This is different for particles with high momenta which are less bent by the magnetic field and thus have straighter tracks.

  This argument was verified by simulations without any magnetic field, which do not show the efficiency drop. Furthermore, Figure 4.12 on page 54 shows that the drop is mainly caused by the dead zones.

  If the dead zones and particles outside of the acceptance range are not taken into consideration the efficiency is about 92% at 1 GeV/c. If all tracks are considered reconstructed the efficiency improves to 99%. It is lowered by the performed cuts: About 3% of the tracks miss the ITS information (*no ITS cut*); in about 5% of the tracks the reconstructed p_t deviates more than 50% from the generated values (*lost tracks cut*).

- **In Θ**: The efficiency in Θ is flat and falling to 0 outside of the acceptance range. At mid-rapidity ($\Theta = \pi/2$) a broadened slight drop of about 2% can be seen. This is caused by particles absorbed or diverted by the central electrode of the TPC. The drop is broadened by the smearing of the interaction point.

- **In ϕ**: The dead zones of the TPC [ALI00], the support structure and borders between the modules of the TRD [ALI01] are revealed by the 18 visible drops of about 9%.

Resolution

The resolution of a physical value x is based on the distribution of

$$\Delta x = x_{Generated} - x_{Reconstructed}. \tag{4.19}$$

It is calculated for p_t, Θ and ϕ, e.g. $\Delta p_t = p_{t,Generated} - p_{t,Reconstructed}$.

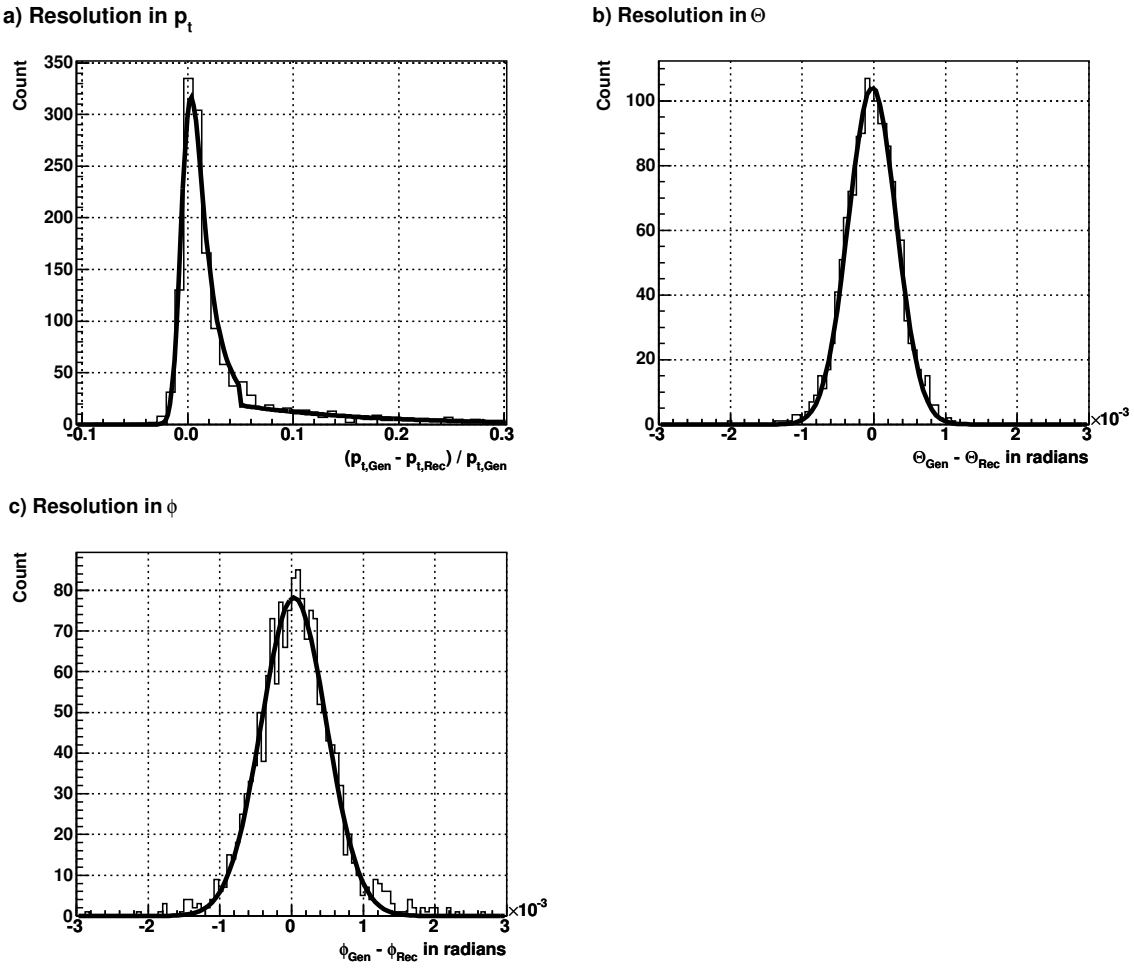

Figure 4.8: Resolution of p_t, Θ and ϕ

Shown are the resolution values in one bin for p_t (a), Θ (b) and ϕ (c). The fit functions whose parameters are saved for the response functions are also shown.

Analogous to the processing of the efficiency, the resolution is created in the $p_t \otimes \Theta$ and $p_t \otimes \phi$ planes, which are divided into bins. The resolution values of tracks which belong to a certain bin are filled into the histogram corresponding to this bin. For each bin, each plane and each of the variables p_t, Θ and ϕ one histogram is created.

A typical binning covering the three variables needs about 20,000 histograms which have to be saved. Instead of saving the histograms, fits are performed and their fit parameters are stored. Histograms including fits can be seen in Figure 4.8. For the resolution in Θ and ϕ a Gaussian fit corresponds quite well to the shape of the histogram. Out of the three fit parameters of the function only one has to be stored. Firstly, the Gaussian function is centered around 0 so that the mean does not have to be stored.

Furthermore the resolution response functions will be used as probability distributions and therefore the absolute scaling is not important. Thus only the Gaussian width σ is saved.

A convenient parametrization of the p_t-resolution is possible if $\Delta p_t/p_t$ is filled into the histograms. The resolution shows a tail which is caused by bremsstrahlung. This superimposes with the "usual" Gaussian smearing, from multiple scattering, of the reconstructed value. Therefore, a Landau distribution convoluted with a Gaussian function [Per05b] suits well for the fit in the region around 0. It is described by the following formula:

$$f_{k,\xi,\sigma,x_0}(x) = k(G_\sigma * L_{\xi,MPV})(x) = k \int G_\sigma(x-z) L_{\xi,MPV}(z) dz \tag{4.20}$$

$$\text{with} \quad MPV = x_0 - MPV_0 \cdot \xi \quad \text{and} \tag{4.21}$$

$$MPV_0 = \text{the maximum of the Landau distribution}$$

Its fit parameters are a constant k, the Landau width ξ, the Gaussian width σ, and the position of the maximum x_0 which is transformed by Eq. (4.21) to the most probable value known from the Landau distribution. However, the tail cannot be properly parameterized using this function. Therefore an exponential function is used which describes the distribution starting from a certain point.

The overall fit function is

$$g_{k,\xi,\sigma,x_0,b,l,\alpha}(x) = \begin{cases} f_{k,\xi,\sigma,x_0}(x) & \text{for} \quad x < b \\ l \exp(-\alpha x) & \text{for} \quad x >= b \end{cases} \tag{4.22}$$

The exponential function implies the two fit parameters l and α, the border between the functions another parameter b. Out of these 7 parameters, only 6 have to be saved because, analogous to before, the resolution response functions will be used as probability distributions and thus only the relative scaling between the two parts of the function is important. Therefore instead of saving k and l, only k/l is saved.

The fit parameters are filled into 2D-histograms, and projections to their axes are created to provide a more suggestive impression of the parameters. Exemplarily, the resolution response functions as functions of p_t and Θ are shown in Figure 4.9. The projection are in fact the averages of the fit parameters. These do not necessarily correspond to the fit parameters, which would be obtained if the response functions were created as a function of only one variable.

The following aspects can be noted:

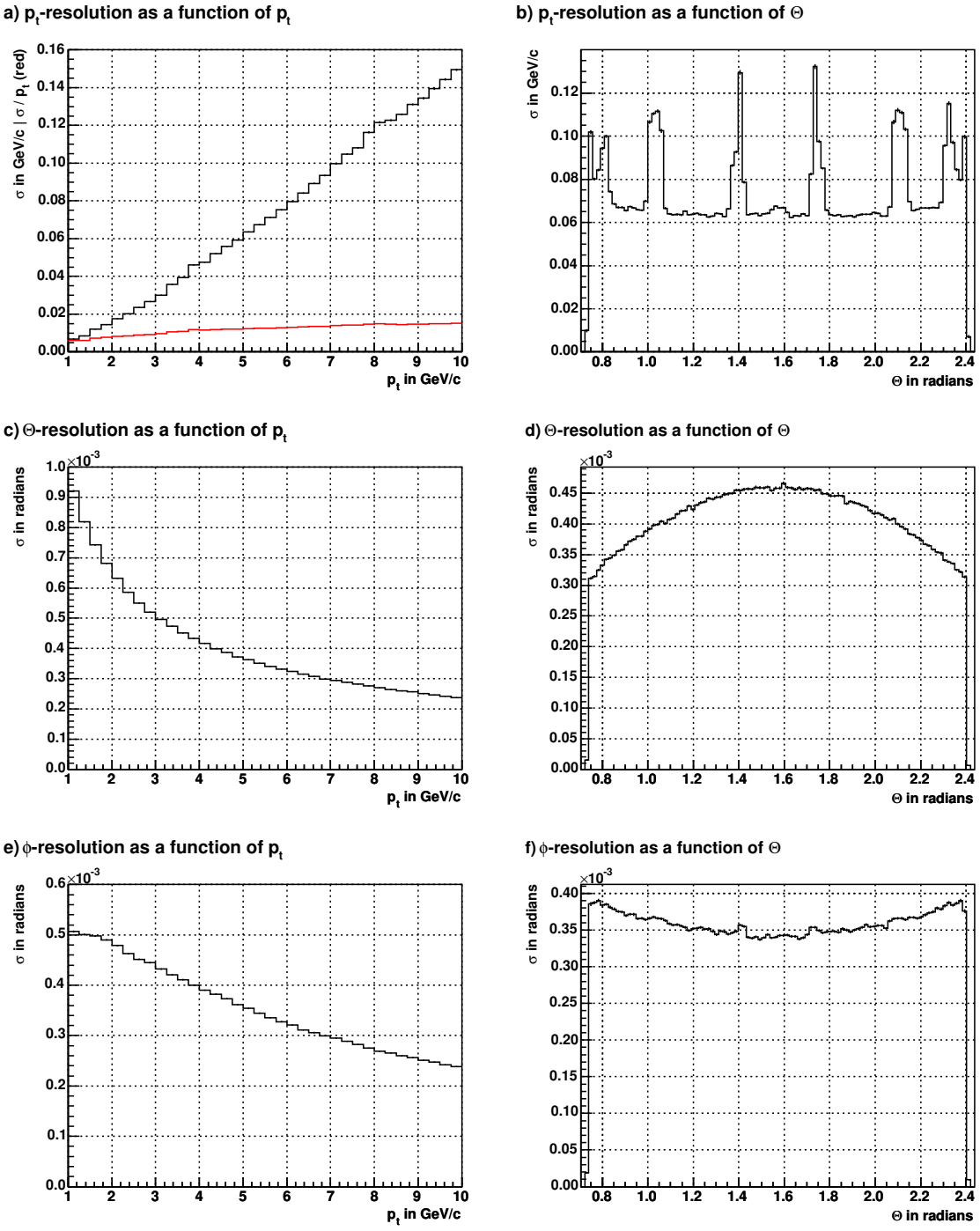

Figure 4.9: Resolution response functions

Shown is the resolution in p_t (a-b), Θ (c-d) and ϕ (e-f). These are shown as functions of p_t and Θ. It was outlined before that the fit to the p_t-resolution has 6 fit parameters, which when plotted are not very suggestive. Instead of that, the σ of Gaussian fits is shown for illustration in (a) and (b). Furthermore $\frac{\sigma}{p_t}$ can be seen in (a) (red). For Θ and ϕ also the fit parameter σ of the Gaussian fits is shown.

- **p_t-Resolution**: Displaying the six fit parameters is not very suggestive. Therefore Gaussian fits of the distributions of Δp_t are performed[10] and the fit parameter σ is shown. It increases (the resolution decreases) because σ shows the absolute deviation from the generated p_t value. Furthermore σ/p_t is depicted. It consists of a constant term and a slight, linear increase. The first is caused by multiple scattering, the latter by the increasing uncertainty of the p_t measurement. The p_t-determination is based on the measurement of the bending in the magnetic field whose uncertainty causes the increase of σ/p_t. The linear increase of σ/p_t was also verified in the range from 10 to 100 GeV/c.

  As a function of Θ, the σ is higher in the gaps between the modules of the TRD. This is caused by the missing TRD information, which implies that the p_t determination that is based on the determination of $\Delta\phi$ is less exact. Furthermore the resolution decreases slightly far from mid-rapidity caused by a longer path through the TPC which is subject to bremsstrahlung and multiple scattering.

- **Θ-Resolution**: The figure shows the fit parameter σ of the Gaussian fits. It decreases (the resolution increases) with higher transverse momenta due to less multiple scattering which scales approximately with $\frac{1}{p}$ [Eid04]. In Θ the resolution is less good for tracks near mid-rapidity, which is caused by diffusion. The previously mentioned effect of a longer path through the TPC, which would actually worsen the resolution of tracks far from mid-rapidity is also present here. But it is exceeded by the negative effect of diffusion which is worst at mid-rapidity: Charge deposited near mid-rapidity has to drift for a longer time than charge deposited near the end-caps and therefore bears a greater uncertainty when it arrives at the cathode. This is especially influencing the Δt-measurement of the drifting charge, which is the basis of the measurement of Θ.

- **ϕ-Resolution**: The figure also shows the fit parameter σ of the Gaussian fits. As a function of p_t the parameter decreases, but not as significant as in the Θ-resolution before. This is due to a smaller improvement by the reduction of multiple scattering because tracks are straighter at higher p_t and therefore produce less $\Delta\phi$ used for the p_t determination. This in return decreases the ϕ-resolution at higher p_t. As a function of Θ a larger fit parameter and therefore a decrease in resolution can be seen for tracks far from mid-rapidity. This can be explained by the same argument as before which is the fact that a longer path through the TPC bears more bremsstrahlung and multiple scattering.

[10] To provide a more suggestive view Δp_t is used in this context. Nevertheless $\Delta p_t/p_t$ is used for the parametrization.

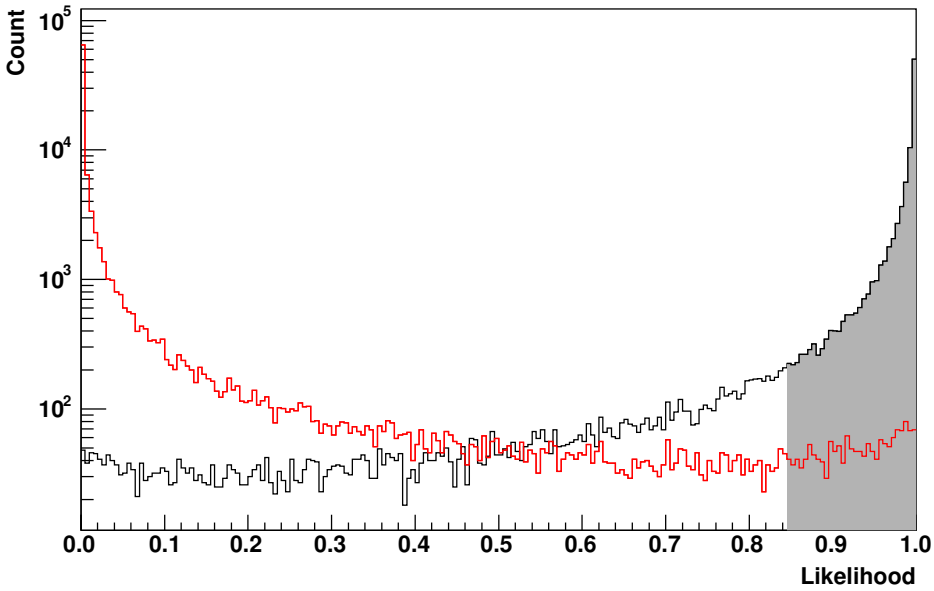

Figure 4.10: Likelihood distribution of electrons and pions

On the x-axis the probability that a particle was identified as electron is given; this is shown for electrons (black) and pions (red). The shaded area shows the integral which is picked in a way such that it contains p_e (usually and shown 90%) of the electrons. The fraction of pions in the shaded area is the pion efficiency p_π.

Electron-Pion Separation Efficiency

The electron-pion separation efficiency is parameterized by calculating the pion efficiency ϵ_π, which is the percentage of pions that have been falsely identified as electrons. It is directly dependent on the demanded electron efficiency ϵ_e, which gives the percentage of correctly identified electrons. Usually 90% is required for ϵ_e. The pion efficiency is derived from the electron and pion likelihood distributions, p_e and p_π, respectively. These are distributions of the probabilities of electrons and pions for being identified as electrons. An integral is created over p_e with an upper boundary of 1 and a lower boundary such that the integral contains a percentage of ϵ_e of the electron entries. The pion efficiency is the fraction of pions of p_π which are inside the integral boundaries. This method is illustrated in Figure 4.10 and described by the following formulas:

$$\text{L is defined by} \quad \epsilon_e \stackrel{!}{=} \frac{\int_L^1 p_e(x)dx}{\int_0^1 p_e(x)dx} \quad \text{and determines} \quad \epsilon_\pi = \frac{\int_L^1 p_\pi(x)dx}{\int_0^1 p_\pi(x)dx} \tag{4.23}$$

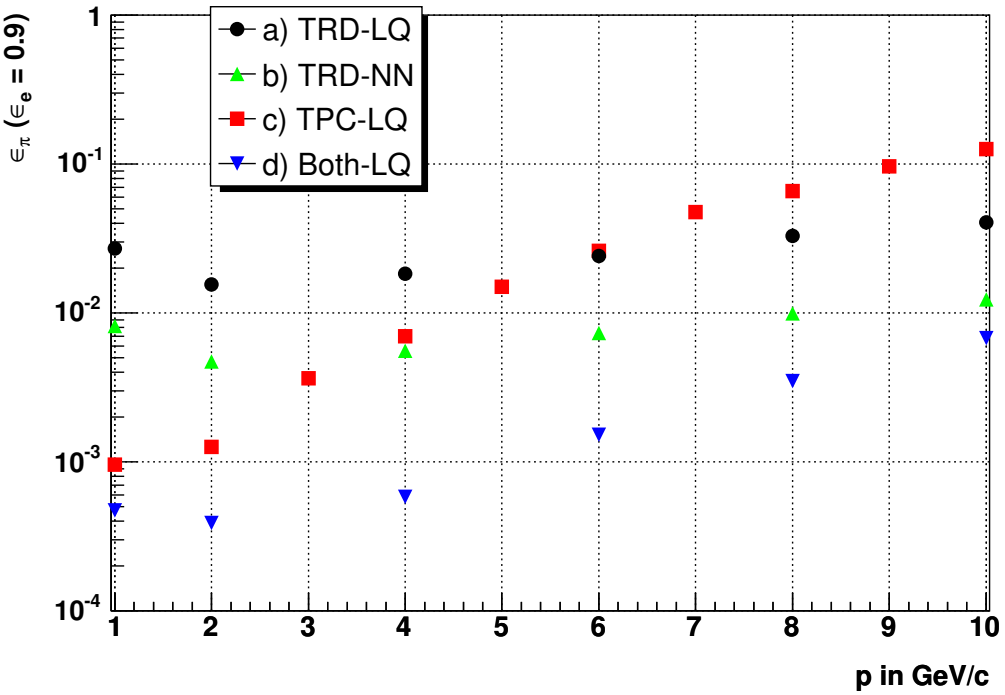

Figure 4.11: Pion efficiency

The pion efficiency at an electron efficiency of 90% is shown. The pion efficiency of the TRD based on test beam data can be seen using the LQ-method (a). For illustration, the pion efficiency, which is to be expected when neural networks are used, is shown in (b). The pion efficiency of the TPC, which was gained from simulations, can be seen in (c). The combined pion efficiency of the TPC and TRD is shown in (d). The latter is available as response function. The error bars are not visible because they are too small. It should be noted that in contrast to all other plots the pion efficiency is given as a function of p, instead of p_t.

In other words, when ϵ_e of the electrons are correctly identified, ϵ_π of the pions are mis-identified as electrons. The reciprocal of ϵ_π is therefore called the *pion rejection*. The design goal of the TRD is $\epsilon_\pi < 0.01$ for $\epsilon_e = 90\%$ and momenta exceeding 3 GeV/c.

The straightforward approach to the creation of probabilities for electron and pion identification is the *likelihood on total charge* (LQ-method). Charge deposition lookup tables are created which are probability distributions of the energy deposited in the detector for certain particle types. In case of the TRD these exist for electrons and pions. The charge that a particle deposits while traversing each layer of the TRD is compared with the distributions, resulting in probabilities for the identification as electron and pion. This can be extended to the *two dimensional likelihood on charge and position of the maximum cluster* (LQX-method). An additional probability distribution of the

position of the time bin, where most of the charge is deposited is taken into account. An alternative approach is the use of neural networks (NN) for the creation of the probabilities, which results in better identification. However, this method is still in an experimental stage. The "worst case" will be available as response function, which is the LQ-method. For a detailed description of the LQ-, LQX- and likelihood method and especially the use of neural networks for particle identification with the TRD see [Wil04, ALI05a].

The current version of AliROOT does not include a functional particle identification for the TRD. This is due to changes in the geometry that have not propagated to the particle identification algorithms yet and problems with the parametrization of transition radiation. Therefore the current results from AliROOT cannot be taken as input for the response function. Alternatively test beam data is used, which was gathered at CERN in 2004 [ALI04b]. The resulting pion efficiency at an electron efficiency of 90% using the LQ-method can be seen in Figure 4.11a. The data is available as a function of p, no Θ nor ϕ information is available. The significant improvement which is to be expected when neural networks are used was shown in [Wil04, ALI05a] on the basis of former test beam data. No results are available yet for the test beam data of 2004. For illustration, Figure 4.11b shows the results which are to be expected when neural networks are used. These are gained by scaling the results from the LQ-method with an improvement factor of 3.3, which is the average improvement that is expected.

Out of the other detectors of the central barrel, apart from the TRD, only the TPC provides significant input for electron-pion separation. The ITS and TOF do not have sufficient information about the energy loss of the particles. The pion efficiency of the TPC was gained by applying the LQ-method on results from simulations. The events with embedded electrons and pions are used. To adjust the binning of the TPC values to the values of the TRD from the test beam data, bins of $\Delta p = 1$ GeV/c are used. Thus the earlier mentioned p_t-*cut* is not performed for this data. The result is shown in Figure 4.11c. As it can be expected the pion efficiency increases drastically above $2 - 3$ GeV/c because electrons and pions cannot be separated by different amounts of deposited energy anymore.

The overall pion efficiency is gained by combining the results from the TRD and TPC. Unfortunately the overall pion efficiency cannot be simply calculated from the two pion efficiencies. Instead the probabilities, resulting from the TRD and TPC, of each particle have to be combined, separately. As mentioned before no information from the TRD concerning particle information is available in the simulated results. Therefore the likelihood distributions gained from test beam data are used. For each simulated particle a random probability from the likelihood distribution of the corresponding particle type and momentum is chosen. This value is combined with the (simulated)

probability created by the TPC. The overall probabilities are filled into likelihood histograms and, analogous to before, used to gain the pion efficiency. The result can be seen in Figure 4.11d.

This method of combining the probabilities does not take into account that correlations between the TRD and TPC probabilities may exist. E.g. an electron which is not identified well in the TPC may also have a bad identification in the TRD. Therefore, as soon as the particle identification algorithms for the TRD are properly integrated in AliROOT and the results correspond to the abilities of the TRD, new events should be simulated and the electron-pion separation efficiency should be extracted.

Improvements by Symmetries

It can be seen that all response functions are symmetrical in Θ around $\frac{\pi}{2}$. Therefore all values above $\frac{\pi}{2}$ can be mirrored to improve the statistics below $\frac{\pi}{2}$ by performing

$$\Theta' = \pi - \Theta \quad \forall \quad \Theta > \frac{\pi}{2}. \tag{4.24}$$

Furthermore it can be seen that the functions repeat in ϕ with a phase of $20° = \frac{\pi}{9}$ which corresponds to the 18 dead zones of the TPC, the support structure and the module boundaries of the TRD. Also here the statistics can be improved significantly by projecting all values to the first 20°-sector. This is done by

$$\phi' = \phi - \frac{\pi}{9} \text{ floor}(\frac{\phi}{\frac{\pi}{9}}). \tag{4.25}$$

Results using these improvements can be seen in Figure 4.12.

4.4.6 Response Functions at Higher Transverse Momenta

The response function are created for fast simulations of particles in a range of $1 - 10$ GeV/c. In these simulations a low fraction of particles has higher transverse momenta. Therefore the response functions will approximate transverse momenta up to 100 GeV/c. It can be expected, however, that e.g. the efficiency does not change significantly for higher p_t due to already nearly straight tracks at 10 GeV/c. Events with electrons up to 100 GeV/c have been simulated, though with lower statistics (250 events per multiplicity). Exemplarily the efficiency and the Θ-resolution at higher transverse momenta are shown in Figure 4.13.

Information about the performance of the particle identification of the TRD above 10 GeV/c is neither available in the simulations nor from test beam data. Therefore no information can be gained for the electron-pion separation efficiency at higher transverse

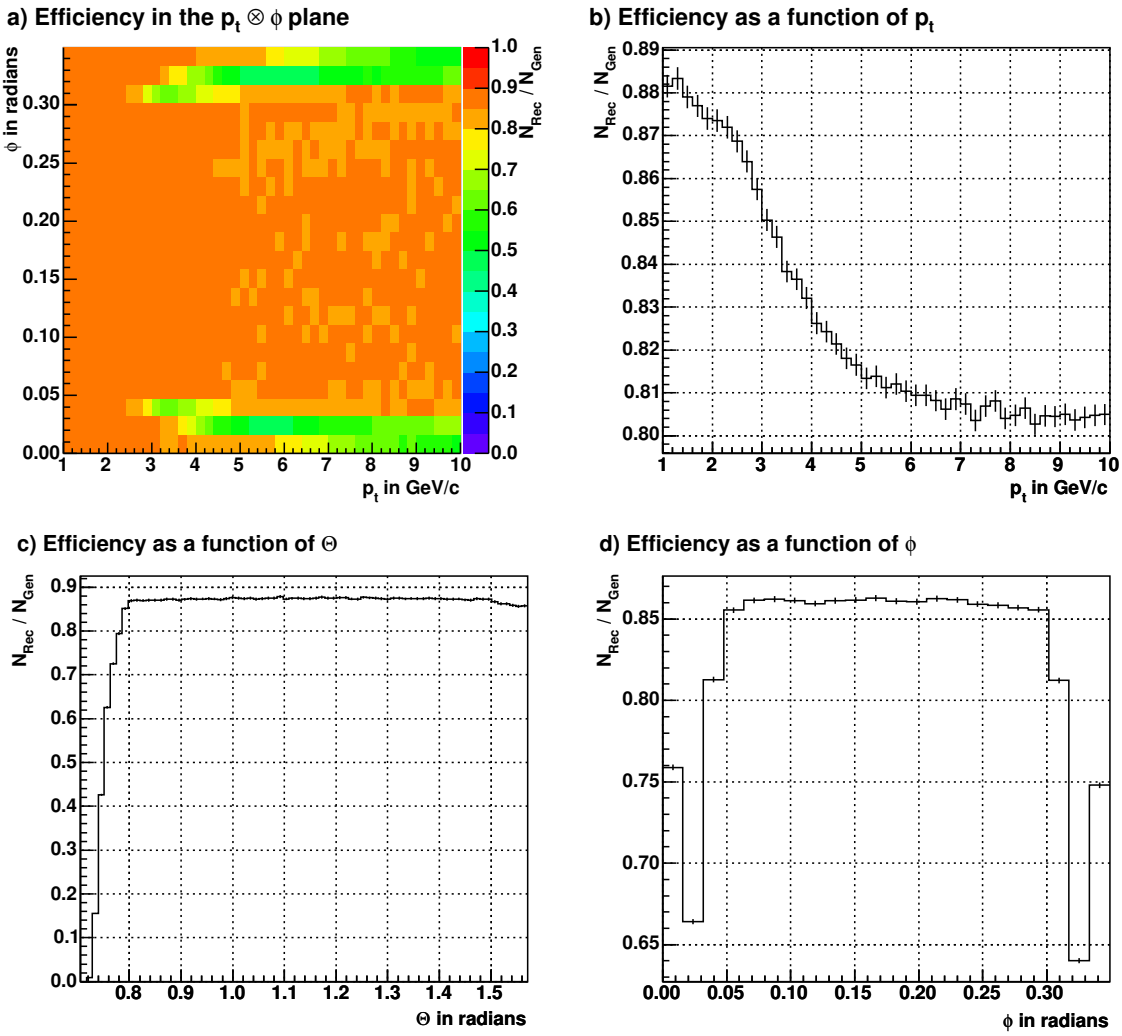

Figure 4.12: Efficiency response function applying symmetries

Shown is the efficiency response function in the $p_t \otimes \phi$ plane (a) applying the improvements mentioned above. Furthermore the efficiency as a function of p_t (b), Θ (c) and ϕ (d) plane is shown. The earlier mentioned efficiency drop (section 4.4.5) caused by the dead zones can be clearly seen at the borders of (a) ($\phi < 0.05$ radians and $\phi > 0.3$ radians).

momenta. An extrapolation from the results below 10 GeV/c may be possible but the physics behind the dependency of the electron-pion separation efficiency on p should be understood first. For the time being the worst value of the electron-pion-separation efficiency, which is at $p = 10$ GeV/c, is used as approximation for higher transverse momenta. The user of the response functions has to take this limitation into account. For the performance studies which will be performed in section 4.7 this is not a major concern because only a minor fraction of particles has momenta above 10 GeV/c.

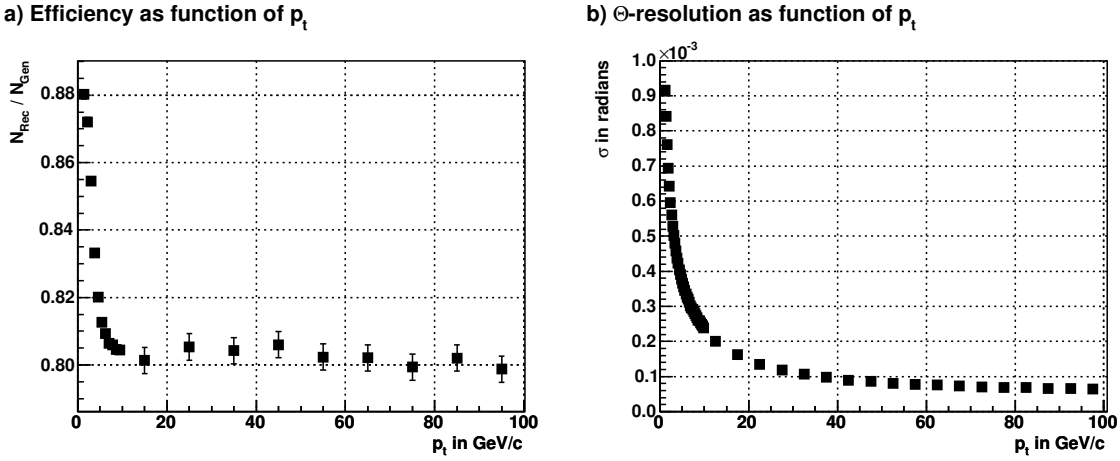

Figure 4.13: Approximation at higher transverse momenta

Exemplarily for the approximations at higher transverse momenta the efficiency (a) and the Θ-resolution (b) are shown as a function of p_t. The error bars in (b) are not visible because they are too small.

4.4.7 Multiplicity Dependence

It was mentioned in section 4.4.1 that the response functions will be created for the multiplicities 0, 4000 and 6000 (section 4.4.1). To this point, all results shown were created using the multiplicity 4000. Exemplarily, the comparison of the efficiency and the ϕ-resolution at different multiplicities can be seen in Figure 4.14. The lower efficiency due to the increased multiplicity can be clearly seen.

4.4.8 Integration into AliROOT

The response functions enable fast simulations for electron studies. To make the functions available in AliROOT the package *ACBRESPONSE* has been created. It contains the class *AliResponses* which can be used to easily access the response functions. It contains functions to provide the efficiency, resolution and electron-pion-separation efficiency. At initialization, the user decides which multiplicity is to be used. It is possible to access the response functions directly or to query random distributions based on the response functions. This can be exemplified by an efficiency value:

- **Direct Access**: The user calls the function *GetEfficiency*, providing the momentum of the particle, which returns the efficiency of the detector between 0 and 1.

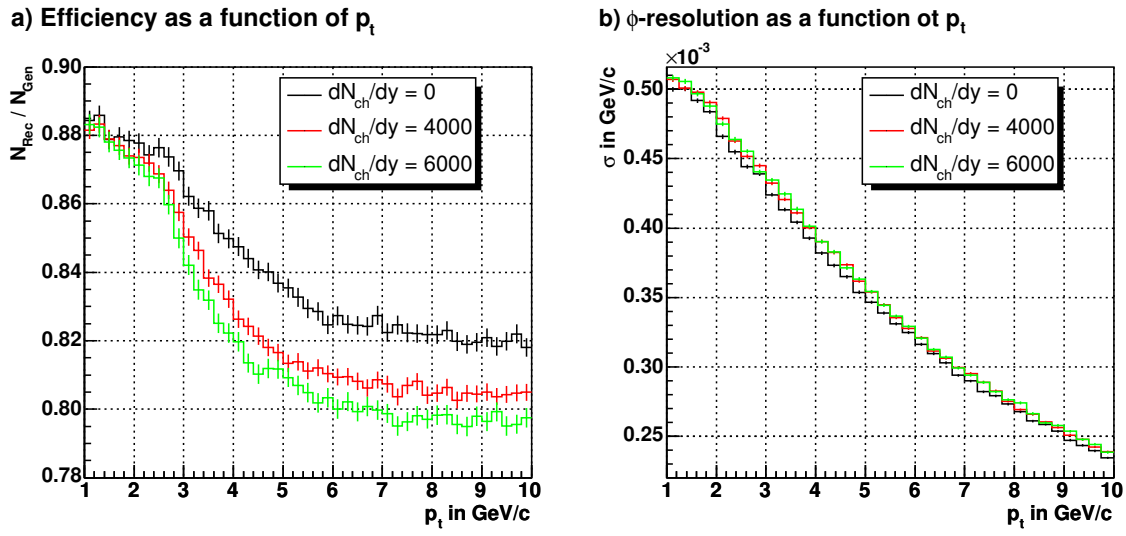

Figure 4.14: Response functions at different multiplicities

The efficiency (a) and the ϕ-resolution (b) as functions of p_t are shown for the multiplicities 0, 4000 and 6000.

- **Random Distribution**: The user calls the function *IsDetected* providing the momentum of the particle. It queries a random generator and returns a boolean value which determines if the particle was detected. This function is a more convenient solution when single particles are processed.

This is analogous for the electron-pion separation efficiency and the resolution. For the latter case the six fit parameters of the p_t-resolution can only be used if the corresponding function (Eq. 4.22) is created by the user. Therefore it is advisable to use the function querying the random distribution.

All functions of the class AliResponses are described in section C of the appendix.

4.4.9 Verification

To verify the correctness of the response functions, results from slow simulation are compared to results from fast simulations which are based on the response functions. The efficiency and electron-pion separation efficiency response functions can be easily verified because their values are percentages. In case of the resolution this is more complicated. Distributions have to be created and fitted, then the fit parameters compared. This is particularly difficult for the p_t-resolution, where 6 fit parameters have to be compared. Therefore Gaussian fits and the standard deviations of the distribu-

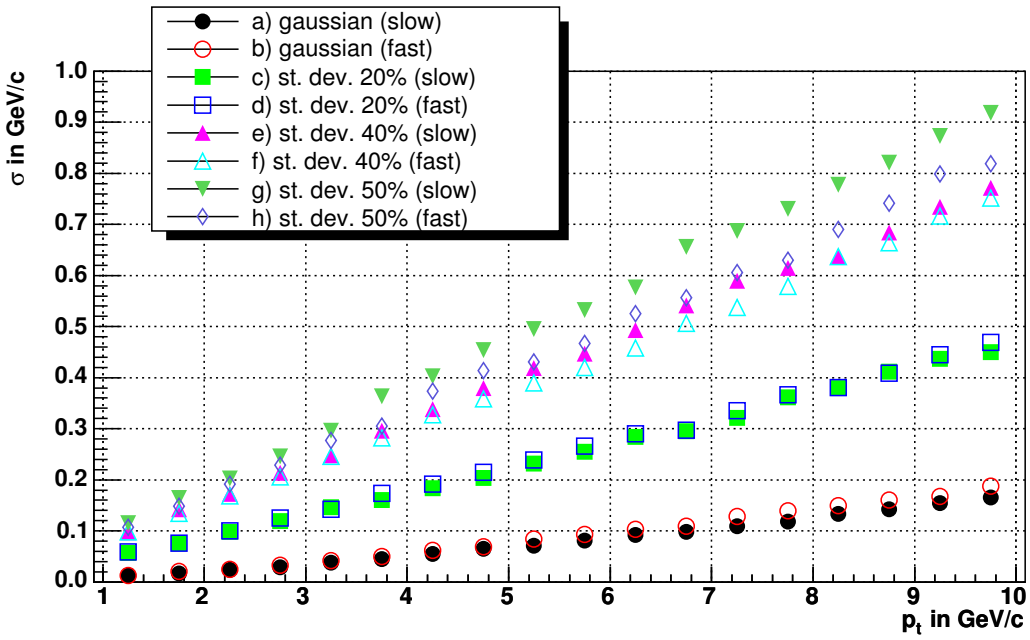

Figure 4.15: Verification of the response functions

The p_t-resolution of fast (open markers) and slow (filled markers) simulations is compared. Shown are the standard deviations and σs of Gaussian fits to data from both slow and fast simulations. The Gaussian fits (a,b) agree quite well. However, a Gaussian fit does not completely represent the resolution which has a tail due to bremsstrahlung. Therefore, the standard deviations taking different ranges into account are also shown. Ranges, in this context, denote the percentage of difference between the reconstructed and generated value: 20% (c-d), 40% (e-f) and 50% (g-h) are shown. The latter is the maximally occurring deviation, particles with a higher deviation have been cut out before (see section 4.4.4, *lost track cut*). Out of the three the first two agree quite well, the third is slightly off in higher p_t which is due to the fact that the parameterizations do not reproduce tracks well whose p_t is far off. However, this affects only a low percentage (1-2%) of tracks.

tions are compared. Exemplarily the evaluation of the p_t-resolution by several means is shown in Figure 4.15.

In [Mah04] a parametrization of the detector response of the ALICE central barrel for pions has been performed. Comparing these to the results in this thesis shows slight differences, especially in the p_t-resolution. These are expected because contrarily to pions, electrons produce bremsstrahlung. The pion efficiency can only be compared to [Mah04] on the level of the single values of the TPC and TRD. An overall pion efficiency is not given. Apart from the latter, which cannot be compared, the results agree with the results in this thesis.

To provide a direct comparison between slow and fast simulations, the following section will compare results of both on the analysis level.

4.5 Comparison of Slow and Fast Simulations

As a test of the correctness of the response functions, a simulation is performed using both slow and fast simulations. The results are then compared.

The response functions have been created for the analysis of quarkonia signals. Fast simulations are necessary because the rates of quarkonia states per event are very low. Thus a number of events is needed, whose computation exceeds the available computing power by far. Therefore these analyses cannot be taken as comparison between slow and fast simulations. A similar case will be used which assumes that the rates of quarkonia states, the J/Ψ was chosen in particular, are much higher. It should be pointed out that this is an unphysical case in the sense that this rate will never be observed. However, it will show that the response functions reproduce the results on the analysis level.

4.5.1 Event Selection

A HIJING event is used, represented by the also previously used parametrization *AliGenHIJINGPara*. 500 J/Ψs are embedded per event which exceeds the realistic rates (Table 4.2, page 28) by a factor of 10^4. The J/Ψs are created by the generator *AliGenParam* with the *CDF scaled* distribution [Bed03] implemented in *AliGenMUONlib*. They are forced to decay to e^+e^-. About 1,000 events have been simulated using both slow and fast simulations.

In the following sections the processing of the events for the cases of slow simulation and fast simulation will be shown. Then the analysis of the simulation results will be described.

4.5.2 Slow Simulation Workflow

The event is simulated and reconstructed with AliROOT. From the reconstructed tracks all electrons and pions which have ITS information are kept for further processing. As mentioned before, this information is necessary to remove secondary J/Ψs which do not originate from the interaction point. Therefore only tracks of particles are kept whose creation vertex is at the interaction point. To reduce the contamination with pions all tracks with a transverse momentum below 1 GeV/c are removed.

These cuts are similar to the processing of the events for the response functions (section 4.4.4). The cuts which are performed from that section are the p_t-*cut*, the *vertex cut* and the *no ITS cut*. However, in this analysis only reconstructed values are used to determine if a track is cut. The true values from the event generator cannot be used because these will not be available when real events are processed.

It was mentioned before that the particle identification of the TRD is not properly working in AliROOT yet. This necessitates the use of the electron-pion separation efficiency response function. This is unfortunate because thus the particle identification cannot be compared between slow and fast simulations. However, there is no other solution until AliROOT provides particle identification of the TRD. The particle identification is performed in the following way: The generated values (which is necessary in this case) determine if the particle is an electron or a pion. In both cases the response function is queried and decides if the particle is detected as an electron. Only particles remain which have been, correctly or falsely, identified as electrons or positrons. Naturally, if a pion is falsely identified, it is identified as an electron or positron depending if it is a negative or positive pion, respectively.

4.5.3 Fast Simulation Workflow

All generated electrons and pions that have a transverse momentum above 1 GeV/c are processed. The fast simulation chain is as follows:

1. The efficiency response function determines if the particle is detected, this takes acceptance and efficiency arguments into account.
2. The electron-pion separation efficiency response function determines if the particle was identified as electron. This utilizes different probability distributions for electrons and pions.
3. The resolution response functions smears the p_t, Θ and ϕ values.

Analogous to the slow simulation, only particles remain which have been, correctly or falsely, identified as electrons or positrons.

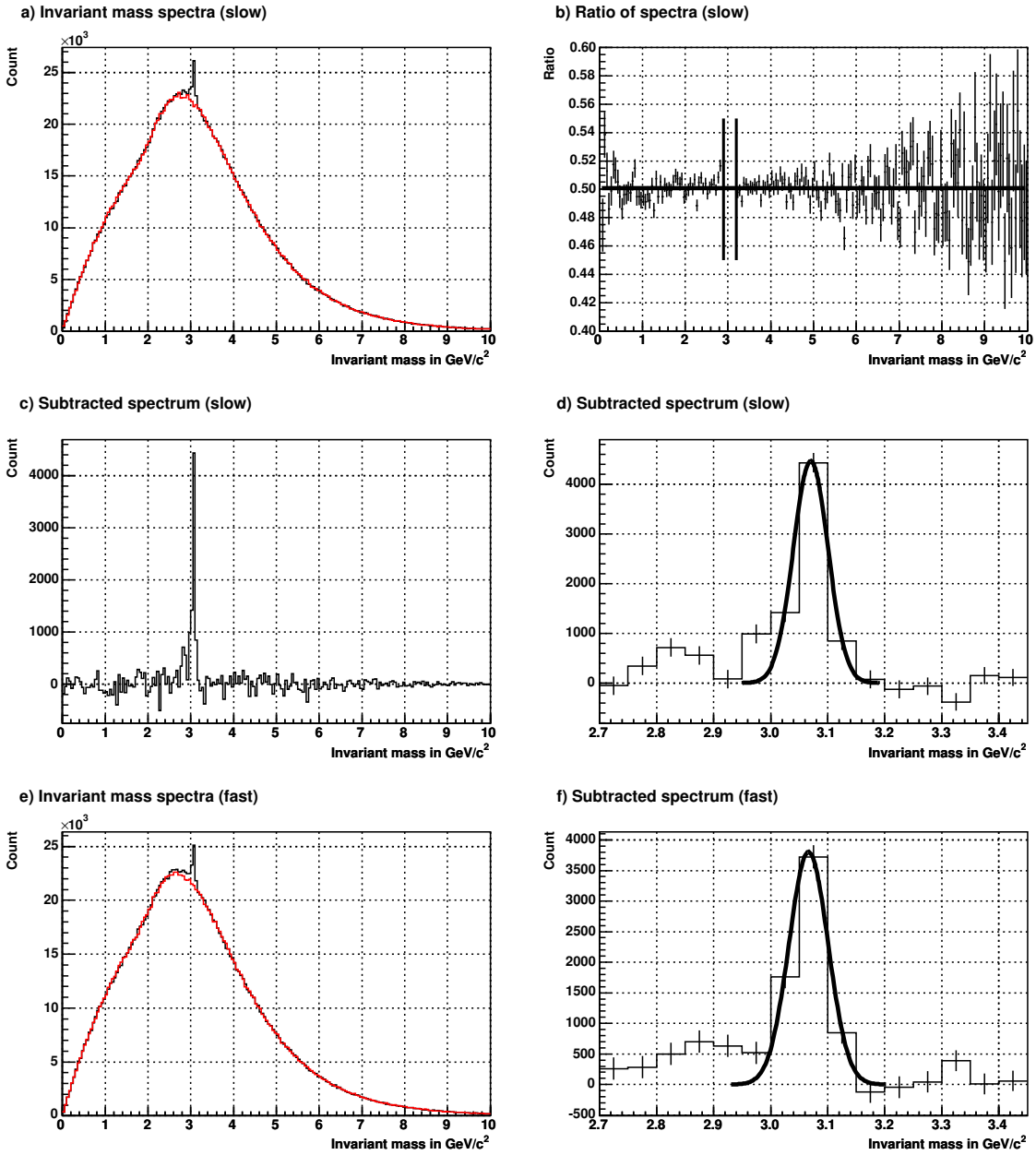

Figure 4.16: Analysis results of slow and fast simulations

Results from slow (a-d) and fast (e-f) simulations are shown. In (a,e) the unlike-sign distribution (black) and the like-sign background (red) can be seen. These are already scaled to each other based on their division, which is shown in (b) fitted with a constant. The lines show the region which is excluded from the fit. The subtracted spectrum is shown in the full range (c) and magnified to the mass region of the J/Ψ fitted with a Gaussian (d,f). The peak widths in (d) and (f) of about 30 MeV/c^2 are of the order of the p_t-resolution.

Simulation	$\mathbf{S} \cdot 10^3$	$\mathbf{B} \cdot 10^3$	S/B in %
Slow simulation (2σ interval)	10.82	106.0	10.2
Fast simulation (2σ interval)	11.91	125.6	9.48
Slow simulation (fixed interval)	17.42	850.0	2.05
Fast simulation (fixed interval)	17.93	842.2	2.13

Table 4.5: Results of slow and fast simulations
The signal and background values, which are the number of entries in the specified interval, are given. These result in the signal-to-background ratio. The values in the 2σ interval cannot be directly compared because slight differences in σ lead to significant differences between the values. For comparison the values taken from a fixed interval of 50 MeV/c^2 around the peak's center are also given. These agree quite well.

4.5.4 Analysis

The results from slow and fast simulations are separately processed, but in the same way. The J/Ψ is identified by its mass; therefore the invariant mass is calculated of each electron-positron pair. It is calculated as follows[11]:

$$m_{inv} = \sqrt{2p_{t,\pi}p_{t,e}\left[\cosh(\eta_e - \eta_\pi) - \cos(\phi_e - \phi_\pi)\right]} \tag{4.26}$$

The resulting invariant mass spectrum is called *unlike-sign distribution* below.

Of course only a low fraction of combinations are the two particles which originated from the same J/Ψ. Thus a large combinatorial background is overlaying the signal. The background is quantified by calculating the invariant mass for electron-electron and positron-positron pairs. These cannot originate from the same J/Ψ, but follow the same distributions like electron-positron pairs that do not originate from the same J/Ψ. The resulting spectrum is called *like-sign background* below.

The like-sign background is scaled to the unlike-sign distribution and subtracted (Figure 4.16a,e). For this purpose the latter two are divided by another. The result is fitted with a constant outside the mass region of the J/Ψ (Figure 4.16b). The subtracted histogram shows the J/Ψ peak which is fitted with a Gaussian (Figure 4.16c,d,f).

Signal and background values are extracted, counting the number of entries in the 2σ interval around the peak's center. From these the signal to background ratio is calculated. For comparison of the results from slow and fast simulations a fixed interval of 50 MeV/c^2 is also used. Results are shown in Table 4.5.

[11] See also section B of the appendix.

The spectra in Figure 4.16 resulting from slow and fast simulations correspond quite well. Also the signal and background values and ratios agree.

4.6 Conclusions

Response functions have been created which parameterize the behavior of the ALICE central barrel when tracking electrons and positrons. These have been made available in the analysis framework AliROOT and can be used for fast simulations. The difference in needed computing time between slow and fast simulations is of several orders of magnitude. For example one event used in the analysis which will be explained below needs approximately 3 hours in a slow simulation. In a fast simulation it needs about a second which corresponds to an improvement of 10^4.

Combined with the shown fact, that the response functions properly reproduce the results of slow simulations, analyses can be performed which were before lacking computing resources. An example are performance studies of quarkonia states which will be described in the following section. These have already been performed using the gained response functions.

4.7 Performance Studies for Quarkonia States

The extracted response functions have already been used in fast simulations for quarkonia states. A performance study which has been performed in [Som05b] along with results which are to be published in [ALI05c] will be outlined briefly in the following.

Similarly[12] to the previous section, a HIJING event with embedded quarkonia states is used. The states J/Ψ, Ψ', Υ, Υ' and Υ'' are embedded corresponding to their realistic rates (Table 4.2, page 28). The rates for the 10% most central events are used. 10^7 of these are recorded per year when the minimum bias trigger[13] is used. This number can be significantly increased with a trigger on central events.

The fast simulation chain and the analysis is similar to the previous section. For each electron and pion with a transverse momentum above 1 GeV/c the response functions are queried. It is first determined if the particle is detected using the efficiency response function. Secondly, the electron-pion separation efficiency response function determines

[12]The generators AliGenHIJINGPara and AliGenParam with the CDF scaled distribution [Bed03] from AliGenMUONlib are used.

[13]The *minimum bias trigger* sends a trigger signal in case of any collision in the detector. Therefore the recorded centralities are evenly distributed.

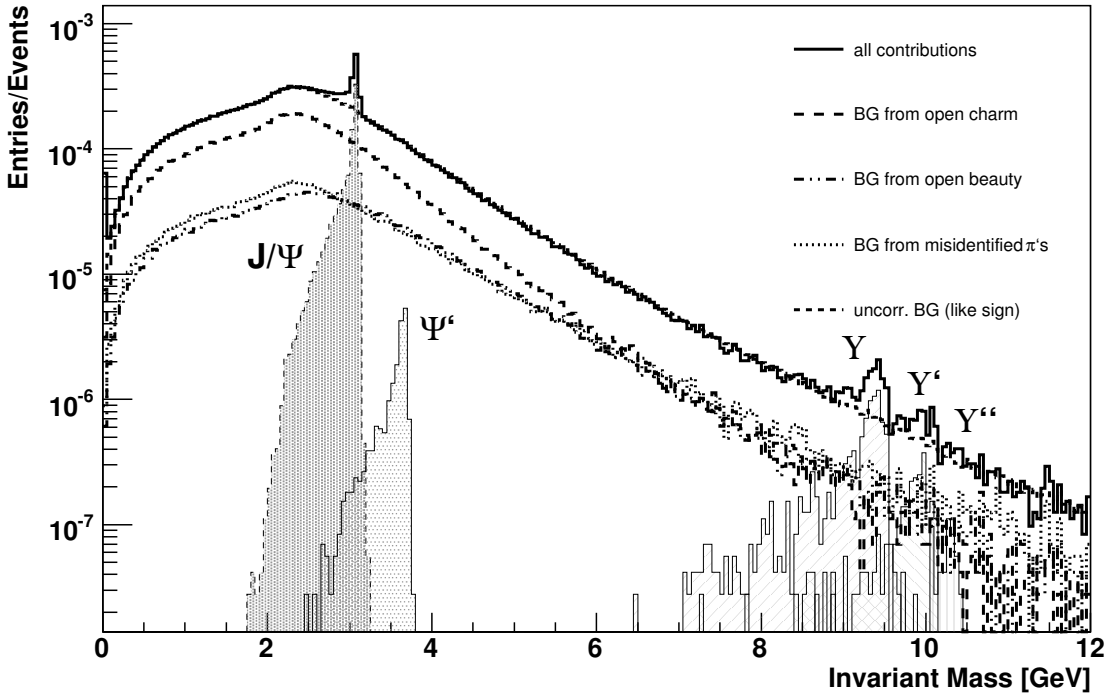

Figure 4.17: Results from quarkonia performance studies [Som05b]
Shown are the invariant mass spectra of performance studies which have been performed in [Som05b] using the gained response functions. The unlike-sign distribution (solid line) and the like-sign background of $7.5 \cdot 10^7$ events can be seen. Furthermore, the background arising from mis-identified pions, open charm and open beauty states are shown. Mis-identified pions are the main contribution to the background at lower masses. The shaded histograms in the bottom area represent the signals which would be measured, if there was no background. They are created by combining the reconstructed electron-positron pairs which originate from the same particle. The tail to lower masses of the peaks caused by bremsstrahlung can be nicely seen. The subtracted spectrum is shown in Figure 4.18.

if the particle was correctly or falsely identified as an electron.[14] Thirdly, the momentum of the particle is smeared using the resolution response functions.

In the analysis invariant mass spectra are created for unlike-sign (electron-positron) pairs and like-sign (electron-electron and positron-positron) pairs. The latter are treated as the background distribution. Subsequently, the two distributions are properly scaled and subtracted from each other.

[14]Due to consistency with other results in [ALI05c], especially the muon part, a different electron-pion separation efficiency response function than the one described before was used in this analysis.

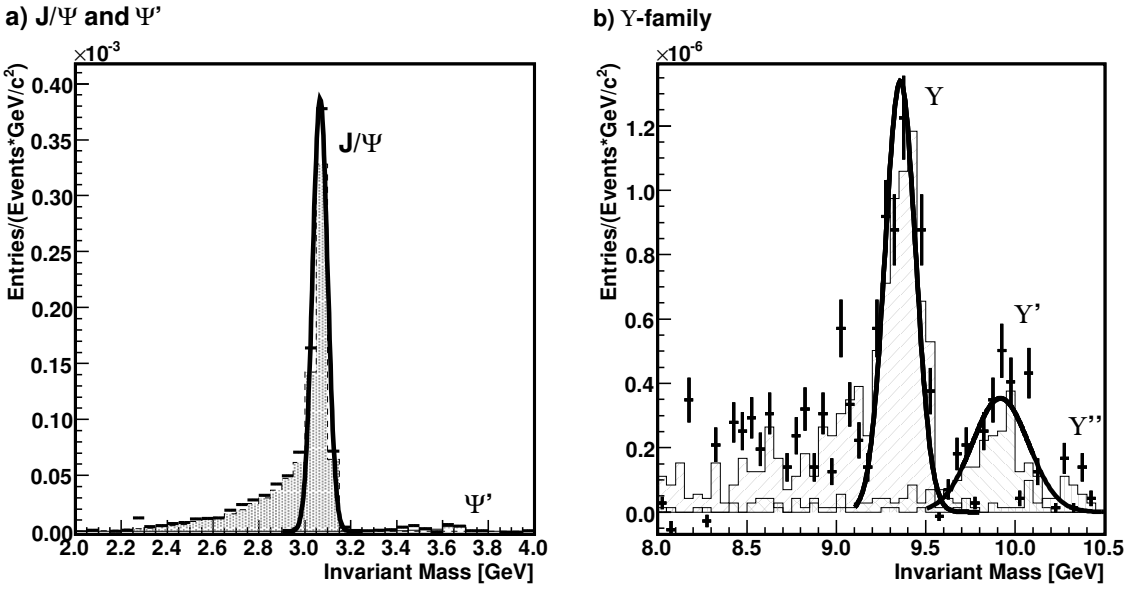

Figure 4.18: Subtracted invariant mass spectrum

The subtracted invariant mass spectrum in the mass regions of the J/Ψ and Ψ' (a) and the Υ-family (b) can be seen. The spectrum is the result of the subtraction of the unlike-sign distribution and the like-sign background properly scaled to each other. The shaded histograms correspond to the signals which would be measured, if there was no background. The J/Ψ and Υ can be nicely seen; the Υ' is indicated.

Results are shown in Figure 4.17. The mass regions around the J/Ψ, Ψ' and Υ-family are shown in Figure 4.18. The J/Ψ and Υ-peaks can be seen nicely; the Υ' is indicated. However, the Ψ' and the Υ'' do not show significant signals. The background mainly originates from mis-identified pions, a better pion rejection would significantly enhance the signals.

ALICE will be able to measure quarkonia states. The study of their rates as functions of variables like the average transverse momentum, the hadron rapidity distribution and the transverse energy can be studied. The measurement of changes in the quarkonia yields will answer which theoretical models apply and give hints to the properties of the new state of matter formed in these collisions.

These results will be used as an input for the second volume of the Physics Performance Report of the ALICE experiment [ALI05c].

5. Distributed Analysis and the Grid

Introduction

In the previous chapter simulations were performed using a distributed computing environment. Up to 100 processors simulated events in parallel. Distribution of programs and collecting of results were controlled by a batch system. Performing these tasks manually would have been very time-consuming, probably exceeding the time which was actually gained by using several machines. When after the start of LHC, analyzing and understanding of the physics behind the data begins, an extremely high amount of computing resources is needed. It has already been addressed in the introduction to this thesis that this amount cannot be made available by CERN or single institutes. Many institutes have to combine their resources to be able to analyze the data. The manual distribution of the data to institutes for reconstruction and analysis being utterly impossible led to the need for an automatic system. The concept of *Grid* was identified as a solution.

5.1 The Grid

The success of the World Wide Web has led to a tremendous amount of information which can be seamlessly accessed. The Grid is a new kind of infrastructure which provides seamless access to resources like computing power and storage distributed over the globe. The name Grid was chosen in analogy to the electric power grid [Fos04a]. In a power grid every outlet provides the same power, the user does not have to take into account where the power comes from. The Grid aims to be similar, computing power and storage can be accessed at one of many equal "outlets".

Distributed computing systems have been available for decades, but they have always tended to be very specialized systems for a limited group of users or even more crucial for a limited type of applications. The Grid goes further and takes different kinds of resources into account. It is dynamic, it can deal with frequent adding and removal of resources. The Grid will make numerous computer centers, which are distributed globally and are under different administrative domains, appear as a single computing environment.

The high energy physics community can take a lot of advantages from this concept. Computers in various distributed computer centers can be used to perform analysis tasks. Data which nowadays is only stored at distinct locations close to the corresponding experiments will be available globally. Therefore researchers distributed over the globe will then be able to access big amounts of experimental data which are supplied by many experiments. Cross checking between different experiments will be easy and allow types of analysis that have not been possible before. Moreover the amount of data in high energy physics experiments nowadays exceeds by orders of magnitude the amount which could be handled by a single institute. So does the amount of computing power needed for the analysis of the data. For example the LHC experiments will produce $10 - 15$ PB of data per year.

The software which provides seamless use of resources, like computing power and storage, is called *Grid Middleware*. The name originates in the fact that the software is the glue between the hardware and the user. The design of a typical Grid middleware will be shown in the following sections. It should be noted that today's typical Grid middleware is still far from the rather idealized description in this introduction.

5.2 The Grid Middleware

The Grid middleware interacts between all components of the Grid. These include the hardware, e.g. different kind of computers and storage types, the software, e.g. other Grid softwares and the Grid's users. These can be data providers like experiments, applications like analysis tools, or scientists. This is schematically shown in Figure 5.1.

The main task of the Grid middleware is to accept the user's tasks, to map them against the available Grid components, to find appropriate locations to perform the tasks, send the tasks to these locations, wait for the results, retrieve them and return them to the user who made the request. The Grid middleware must be able to deal with any kind of failure which could happen in this cycle.

The complete interaction with the Grid should be completely transparent: it should appear like a single system. The part of the Grid which executes the tasks, the location of the data which is necessary to perform the tasks or the location of the user does not imply any changes in the user's behavior when interacting with the Grid middleware.

A crucial issue is security. It has to be assured that data stored in the Grid can only be accessed by users with the necessary access rights. This is even more important for other fields of science for which the Grid is used, like medical applications. To prevent

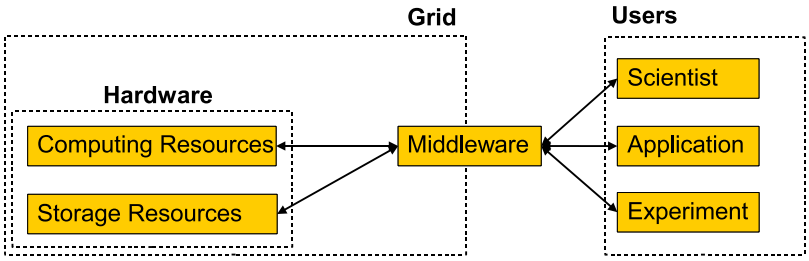

Figure 5.1: The Grid middleware

The Grid middleware is the glue between the Grid's hardware and the users, these include scientists, applications and experiments.

abuse only users with the correct credentials should be able to access specific parts of the Grid.

Exemplarily the structure of the Grid middleware gLite will be introduced below.

5.3 Virtual Organizations

The Virtual Organization (VO) has been introduced as an organizational unit. Users belong to Virtual Organizations which in turn have contracts with Grid Service Providers. The latter ones own resources which are thus made available to the users. This concept allows an association between providers of resources and users, which is necessary because resources financed and maintained by for example ALICE are supposed to be mainly available for ALICE users. Naturally, users can be part of several VOs and therefore access different sets of resources.

At CERN, each LHC experiment will form one Virtual Organization. In practice, these have resources of their own and do not use structures involving Grid Service Providers.

5.4 The Grid Middleware gLite

One of today's main Grid projects is the project *Enabling Grids for E-SciencE (EGEE)* [EGE05] which is funded by the European Union. The EGEE project involves more than 70 institutes in 28 countries and is headed by CERN. It started in April 2004 and combines the knowledge of former Grid projects which have been developed at the collaborating institutes. Its main goal is to put a production quality Grid in place which is able to cope with the needs of its users, the LHC experiments in particular. Based

on earlier experience the project should create a prototype of a new generation Grid middleware software. The prototype is then to be extended to a final Grid middleware which is to satisfy the users' requirements.

The prototype developed by EGEE as well as the final Grid middleware is called *gLite* [EGE04]. gLite is based on experience from the projects ALICE Environment (AliEn) [Sai03], European DataGrid (EDG) [TDG00], LHC Computing Grid (LCG) [LCG05], NorduGrid [Nor05], Virtual Data Toolkit (VDT) [VDT05] and many more.

5.4.1 Structure

The Grid middleware gLite implements a service-orientated architecture, in other words the software is structured in several components called services where each is responsible for certain tasks. This model has many advantages: firstly it allows independent development of the services. Secondly deployment and maintenance on different computers and at different locations are much easier when the different parts of the software are not hard-wired together. Furthermore services can be used in different contexts, and even components of different implementations of Grid middleware could be combined (*interoperability*).

The services of gLite are depicted in Figure 5.2 and essential ones described in the following sections. The gLite services are implemented as web services. A web service is a software which exposes its features over computing networks using standard internet protocols.

5.4.2 Security Services

Security in Grid environments is a vital issue in many ways: Firstly the access to Grid resources should be restricted to users who are known. This guarantees a reasonable, responsible and cooperative use of the resources. In addition a more fine-grained model allows association of resources with user groups. This makes sure that for example a collaboration which acquired resources is given preferential right of using them. This is also supported by the concept of Virtual Organizations which was previously introduced.

The second important security issue is privacy. Users and user groups should be able to protect their data against access by others. In gLite Access Control Lists (ACLs) are used which allow to specify detailed access privileges for specific users and user groups. Furthermore data which is transmitted between Grid elements needs to be encrypted to prevent eavesdropping. In high energy physics privacy may not seem a

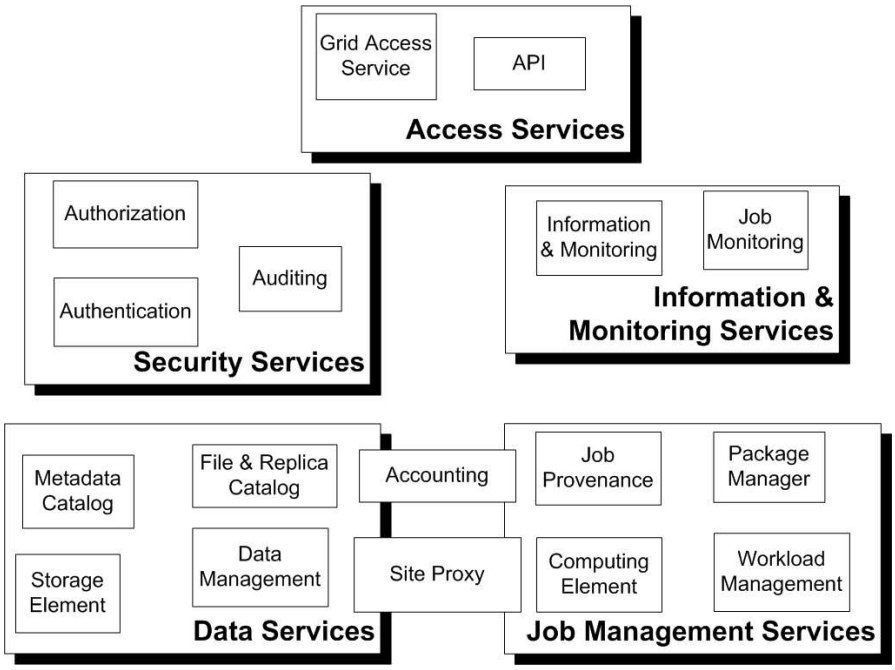

Figure 5.2: The services of the Grid middleware gLite [EGE04]

major concern. On the other hand it is vital for other user groups like the previously mentioned medical applications.

These requirements are fulfilled by the following means:

Authentication

The most important issue is the correct identification of a user. Once a user is correctly identified her access rights can be exactly determined. Nobody should be able to claim somebody else's identity and consequently misuse user accounts. For practical reasons all services should use the same authentication mechanisms, so that users do not need different credentials for different services or sites. Authentication is also needed for the process of accounting, which is the tracking of the amount of resources a user consumes.

In gLite the Public Key Infrastructure (PKI) [Hou02] is used. The Public Key Infrastructure provides security by asymmetric encryption and certificates. In [Pac04] it is described as *"a technology that enables practical distribution of public keys using certificates"*. The following paragraphs will give a brief introduction.

The authentication process with the use of PKI is based on the exchange of asymmetric encrypted messages. In contrast to symmetric encryption, asymmetric encryption uses

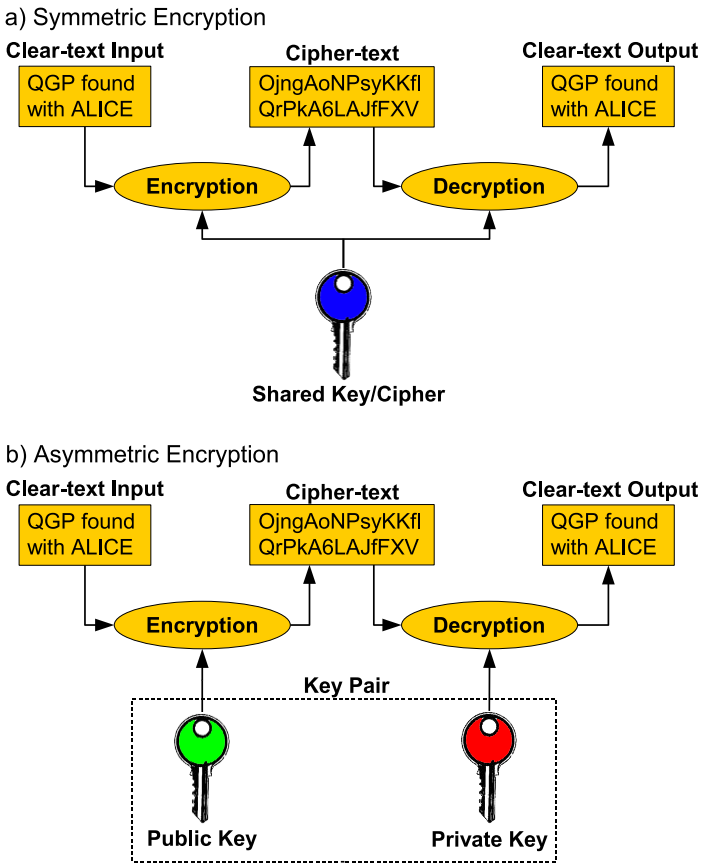

Figure 5.3: The contrast between symmetric and asymmetric encryption
In symmetric encryption (a) the same key is used for encryption and decryption. Therefore the key is also called shared cipher. In asymmetric encryption (b) a pair of keys is used, one of which is the public key and used for encryption, the other the private key and used for decryption. The model shown in the figure is simplified: In practice, due to the fact that asymmetric encryption requires significantly more computing power than symmetric encryption, asymmetric encryption is only used to encrypt a cipher. This cipher is used to symmetrically encrypt the message. In this way the advantages of both approaches, the security of asymmetric encryption and the speed of symmetric encryption, are used together.

---

different keys for encryption and decryption (see Figure 5.3). Each user owns a pair of keys, one is considered the *public key*, the other the *private key*. The public key is made available to everybody and used to encrypt messages directed to the user. Messages encrypted with the public key can be sent freely using any, even insecure, communication channel because only the user owning the corresponding private key is able to decrypt the message.

However, the question may be raised how a user A knows that a given public key belongs to a given user B. Another user C could create a key and claim it to belong to user B. User A would use the key to encrypt sensitive information and user C could read it. This problem, the so-called *man in the middle attack*, is solved by *certificates*: A simple certificate contains a public key and information about the owner. Furthermore it is signed by a trusted entity. Trusted entities, so-called *certificate authorities* (CAs), are institutes, governments or agencies. For example CERN or the Universität Münster take over such functions.

The authentication process using PKI, which is in use by gLite, can be outlined as follows:

- The user, who wants to authenticate to the system, sends her certificate.
- The system verifies the digital signature of the certificate. This makes sure that the certificate is valid and that the public key in the certificate belongs to the user whose name is given in the certificate. This does not necessarily mean that the user who has sent the certificate is the user named in the certificate.
- The system provides some random data to the user and asks to encrypt it using her private key.
- The user encrypts the data and sends it back to the system. If she owns the private key corresponding to the certificate, she is able to encrypt the data in a way such that the system is able to decrypt it using the public key in the certificate.
- The system decrypts the data and checks if it matches the random data sent to the user. This is only the case when the user's private key corresponds to the public key in the certificate. If the data matches, the system can be sure that the user is who she claims to be. In other words the user is authenticated.

The certificates used in gLite are X.509 certificates [Hou02] which contain additional information, for example an expiration date.

It should be pointed out that the privacy of the private key is vital to the authentication system. Once a private key is exposed to malicious persons, they could claim to be the user owning the private key and misuse her account. For this case special revocation methods exist which will not be explained in this context as they would exceed the scope of this thesis.

Summarizing, each user owns a private key and a certificate. The latter contains the user's public key and is signed by a certificate authority. It is given to all users and

services wishing to communicate with the user. The certificate and the private key form the user's *credentials*.

Authorization

The Authorization service decides if a user is allowed to access certain Grid elements. These can be data, resources or services. The decision is based on the user's membership in certain groups. For private data like files it is checked if the user owns a given file or was given access by the owner of the file. For this purpose Access Control Lists (ACLs) are used.

Delegation

Some of the services which execute users' tasks involve other services, for example a job which relies on a number of files owned by the user. It must be possible for the service which deals with the job management to receive these files, although they are not owned by the service itself. Therefore the user has to give the service the right to act on her behalf. This is called *to delegate credentials*. In turn the service's behavior is called *to impersonate the user*.

Technically speaking the act of delegation could be performed by giving the user's private key to the service. This contradicts the claim mentioned above that the private key should be kept strictly private and therefore implies a dangerous risk: If a service gets compromised[1] private keys could be stolen. The user's credentials would not be secure anymore. This problem can be solved by the creation of temporary valid credentials, which are signed with the user's private key. These are called *proxy certificates* [Tue04], their lifetime can be defined by the user and usually lasts between a few hours and a few days, depending on its purpose. The risk coming from a stolen proxy certificate is low because it will expire soon.

The risk of compromised credentials is also tackled by the introduction of *credential stores* [Gus04] which is depicted in Figure 5.4. A user, who wants to delegate her credentials to a service, does not directly give a proxy certificate to the service. Instead she stores it in a credential store and protects it with a password. Then she tells the service where to pick it up. When a service requests a proxy certificate, the credential store creates a new proxy certificate, signs it with the user's proxy certificate, and returns it to the service. This serves the purpose that an even lower lifetime can be

[1] A system is called *compromised* when unauthorized entities are able to access it, e.g. due to a successful attack by hackers.

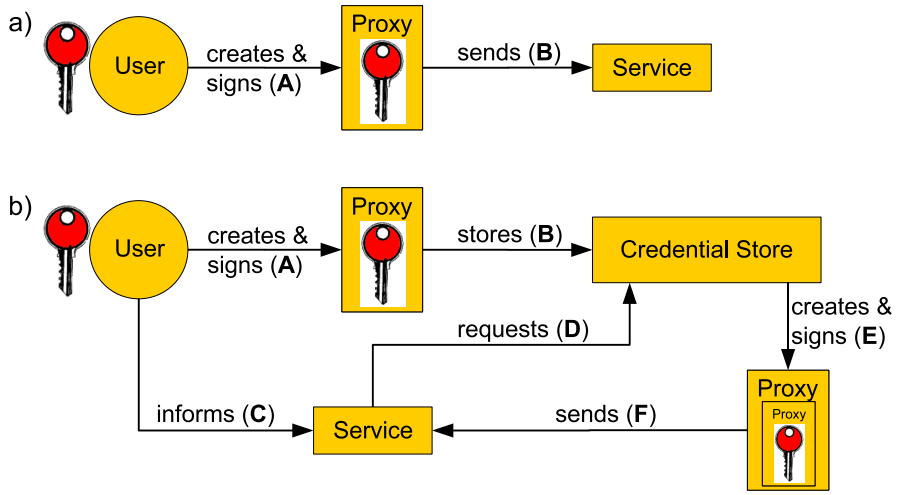

Figure 5.4: Delegation of credentials using credential stores
(a) Delegation of credentials without credential store: the user creates a proxy certificate signed with her private key (A) and sends it to the service (B).
(b) Delegation using credential stores: the signed proxy certificate (A) is stored in the credential store (B). The user informs the service where to pick the certificate up (C). It requests it (D) and obtains a proxy certificate of the proxy certificate (E,F).

associated with it (usually of the order of hours). As mentioned earlier, the proxy certificate stored in the credential store has a short lifetime. In turn, the proxy certificate of the proxy certificate, which is exposed to the service, has an even lower lifetime. If a service gets compromised the time frame in which the credentials can be abused is thus very short. If delegated credentials are needed for a longer period than the initial lifetime, the service can regularly demand new proxy certificates from the credential store. This process is called *renewal*.

Credential stores which store many proxy certificates of users' credentials are high-value targets for directed attacks. These services should only be hosted in trusted computing environments and should not be central. In practice these are hosted by the user's home organization. This limits the impact of a compromised machine on that single organization, and, in addition, increases the trust of users in the service.

gLite uses the credential store *MyProxy* [Nov01] which is developed by the National Center for Supercomputing Applications (NCSA) [NCS05].

5.4.3 Data Services

File & Replica Catalog and Data Management

Files are stored at various locations in the Grid. Users who want to use the data should not have to take care of connecting to the specific location where the data is stored. Instead they should be able to ask for a file and the system should transparently locate the data and deliver it to the users. The File Catalog service keeps a directory of files stored in the Grid and consequently provides a file system like view of all files in the Grid.

Some data is used more frequently than other. It is the task of the Data Management service in cooperation with the Replica Catalog service to replicate data, which is often used, to other locations. These have to be carefully selected to speed up the access to the data.

Metadata Catalog

The Metadata Catalog stores additional (meta) information about data stored in the Grid. It allows users to select data by providing search criteria to the Metadata Catalog. An example in high energy physics would be to ask for events with a given centrality.

Storage Element

The Storage Element service provides the logic for data storing. The storage itself can be realized in many ways, for example by disk or tape. Storage Elements can have different policies of data safety and are therefore divided into *tactical* and *strategic* storage. Tactical storage is suited for temporarily needed files which have to be accessible quickly. Since the data safety of tactical storage is low, only data which can be regenerated should be stored. Strategic storage comes with a higher quality of service and offers reliable and timewise unlimited storage. Usually it uses mass storage systems to be able to store large quantities of data.

Raw experimental data which is saved in strategic storage, and results from preliminary analysis which are saved in tactical storage, are examples for the use in high energy physics.

5.4.4 Job Management Services

Workload Management & Job Provenance

The task of the Workload Management service is to take care of the whole cycle of jobs[2] being submitted to the Grid. Firstly it accepts jobs from users. Then it matches the jobs' specifications to the available resources and assigns the jobs to suitable resources. It submits the necessary applications and data to the resources and monitors the execution of the jobs. The monitoring data is also made available to the user. When the jobs are finished the results are retrieved and stored for the users to be picked up.

The Job Provenance service provides information about the environment of completed and failed jobs which can be used for debugging.

Computing Element

The Computing Element service is the front-end to an arbitrary number of computers which can execute Grid jobs. In most cases a Computing Element interfaces with the local batch system at a site[3] which is part of the Grid infrastructure. Consequently, it is the glue between the Grid and the local queues which deliver the jobs to the computer nodes.

Package Manager

The Package Manager service distributes applications which are needed by Grid jobs to the computers which run the jobs. This is especially important in the context of nowadays fast development cycles of applications which results in the need for an easy and fast distribution system for updates and new applications.

5.4.5 Information & Monitoring Services

The Information & Monitoring services are a vital component of the middleware. They are aware of all services running in the Grid environment, their free resources and utilization. All other services report to the Information Services publishing their state

[2]Users send *jobs* to the Grid middleware to have tasks performed. Apart from the tasks which are to be executed, a job includes information about dependencies to programs and criteria for input data. An example for the case of a high energy physics analysis job will be given below.

[3]A *site* denotes any participating entity which provides services of the Grid middleware. This can range from a complete set of services to a simple computing power or storage resource provider.

and additional important information which can be queried by users and services. These services can be used to find elements with certain properties or to get a general overview. An example would be to locate an element with a certain amount of free storage space.

Services, like the Workload Management and the Data Management, rely on the information provided by the Information & Monitoring services.

5.4.6 Access Services

API

Services expose interfaces to give applications and users the possibility to access them. Web services expose their interface in the form of a Web Service Description Language (WSDL) file [Chr01]. Based on this file an Application Program Interface (API) can be generated for different programming languages and platforms. However, for more complex cases, e.g. if some logic is needed on the user's side, libraries are provided which contain APIs for these specific services.

The API can be used to explicitly access a single service. When users interact with many Grid services, it can be inconvenient to access each of them separately by their API. Instead the counterpart of the API which is the Grid Access Service should be used. An explanation of this service will follow below.

It should be noted that the API is not a service of its own, nevertheless, it is shown in the diagram of services (Figure 5.2, page 69) so that this contains all ways of accessing a service.

Grid Access Service

The Grid Access Service offers access to all Grid services by a unified interface. Therefore the user only needs to talk to that service which can be personalized to her needs. It can be seen as an entry point to the Grid offering transparent access to the functionality of all underlying Grid services.

The Grid Access Service will be described in detail in section 6.2.

5.5 Distributed Analysis

High energy physics experiments produce data at very high scales, for the five LHC experiments the data volume will approximately be $10 - 15$ PB per year. Apart from storing the raw data, reconstruction has to be performed and results will be analyzed by many physicists. This will require enormous amounts of storage capabilities and computing power which need to be managed and their usage organized. However, all of these processes are performed on a "one-event" basis; storing, reconstruction and analysis of an event are not dependent on other events. After analysis, the results are merged. The order in which the events have been analyzed has no effect on the merged results. This makes the Grid a very suitable candidate for addressing the requirements of high energy physics.

To support so-called *distributed analysis*, the analysis frameworks, for example AliROOT, will be interfaced with the Grid. The most striking example of distributed analysis in the Grid is the Parallel ROOT Facility (PROOF) [Bal03], which enables the submission of an analysis to the Grid giving criteria for events which have to be processed. These events are automatically located at sites in the Grid and the analysis is sent to the sites. In case of several locations of identical events, the amount which is to be processed per site is shared corresponding to the computing power at each site. Results of the analysis are retrieved and merged as soon as sites finish their set of events. The resulting histograms visually improve in the course of time. A detailed description of PROOF can be for example found in [LHC03].

As an example, the submission of a high energy physics analysis job to the Grid will be shown. The services of the Grid middleware which are needed for each step will also be listed up. The steps of such a job are:

- The user defines job criteria specifying the program to be run and the applications it relies on. An example is a macro which has to be run with a certain AliROOT version. Furthermore the user provides selection criteria for the data the program is to process. These can point to distinct files available by the **File Catalog** or be results from a query to the **Metadata Catalog**. An example for the latter case is a query for central events with a trigger on at least one electron above 3 GeV/c.

- The job is submitted to the **Workload Management** using the **Grid Access Service** or an **API**. This implies the use of the **Authentication** and **Authorization services**. A possible **Computing Element** to run the program is determined. This takes the requirements for installed applications into account

(**Package Manager**) as well as the location of the input data (**File Catalog** and **Data Management**).

- The input data is transferred from the corresponding **Storage Elements** to the designated execution site, then the job is executed at the **Computing Element** which is monitored by the **Job Monitoring**.
- After successful execution the results are stored in a **Storage Element**. If applicable metadata describing the results is generated and stored in the **Metadata Catalog**.
- During the whole process the user can gain information about the status of the job from the **Information & Monitoring services**.

6. Developing the Grid

6.1 Introductory Overview

In the work performed for this thesis distributed computing was intensively used for high energy physics analysis. This thesis also contains the development of parts of a Grid middleware software. The development was performed in the context of the earlier mentioned EGEE project which develops the Grid middleware gLite. In the following paragraphs, an overview of the newly developed elements of gLite is given. Furthermore their position in the overall context is shown.

The first component which has been developed is the Grid Access Service. It was already introduced as entry point to the Grid in the previous section. It provides access to all services of the Grid middleware and could even be extended to access other implementations of Grid middleware. It hides the Grid's complexity and therefore, as far as changes to underlying services are concerned, there is no need for the user to change her behavior when interacting with the Grid, or to change applications using the Grid.

ROOT [Bru03], the framework for high energy physics analysis, was extended to be able to access the gLite Grid middleware. Consequently it can be used to submit analysis tasks to the Grid. Apart from a new module for ROOT containing classes to access the Grid, a service had to be written residing between ROOT and the Grid. It was named APIService and could be described as the glue between the ROOT classes and the Grid. ROOT is e.g. the basis of AliROOT.

The high energy physics analysis tasks submitted to the Grid usually cover huge amounts of data. For faster processing it is useful to split the input data and process these parts on different computers in parallel. In the context of the integration of gLite into ROOT, splitting algorithms for input data were also developed.

To host the two latter mentioned services, the Grid Access Service and the APIService, a web service container was developed. A container is a piece of software which can be contacted and asked to create a service for a user. It takes care of the state and lifetime of the hosted services. Figuratively it could be described as a habitat for services. Furthermore the container is aware of other containers. Consequently it can provide service discovery: It determines transparently the most suitable location for creating new services. A container has been developed which is named gContainer.

The development of two components is described in detail in the following sections: the Grid Access Service and the gContainer. They should be clearly seen as parts of the prototype which is to be developed by EGEE and are to show the concept and feasibility of the approach. Of course, many improvements will still be possible. It was mentioned before that gLite is based on several other Grid projects. Partly the implementation of the Grid Access Service and the gContainer will show proximity to AliEn [Sai03] which is one of these Grid projects. However, the components are developed as part of the whole gLite middleware.

The Grid Access Service and the gContainer are also described in [Bun04].

6.2 The Grid Access Service

Introduction

The Grid middleware is structured in a number of services. Each serves a specific task, for example data or job management. At the same time, there might exist several implementations for one specific service, which solve a given problem in different ways. The Grid Access Service simplifies the usage of all these services. It serves two major purposes: Firstly it hides the complexity of the Grid middleware, secondly it allows easy replacement of components of the Grid middleware.

The Grid Access Service virtually resides between the user and the services of the Grid middleware (see Figure 6.1). Every task is sent to the Grid Access Service which delegates it to that service which is responsible for this specific task. The latter one will now be called *underlying service*. This model implies a great advantage: An underlying service can be replaced without the user having to change her behavior or parts of application code. The replacement could even be performed without the user noticing it. This is of great value when it comes to the deployment of the Grid middleware.

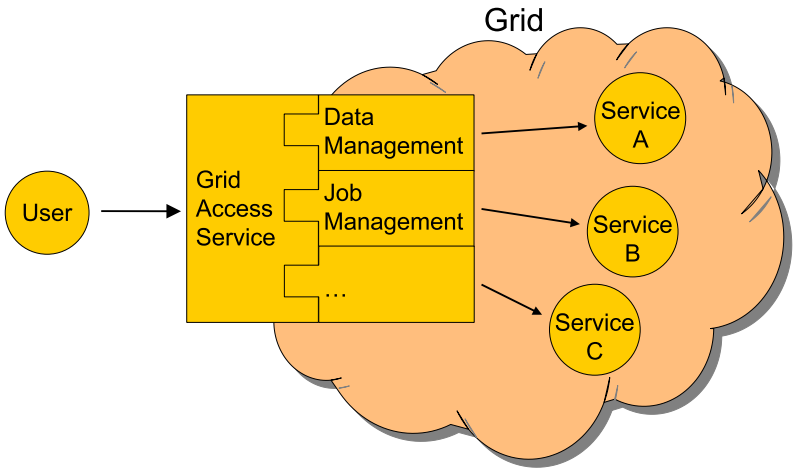

Figure 6.1: The Grid Access Service

The properties of the Grid Access Service (GAS) can be summarized as follows:

- The GAS is the main entry point to the Grid. It provides access to all major components like data and job management.
- The GAS deals with all management, communication and security issues. Therefore it is a simple way to use all functionality offered by the Grid middleware.

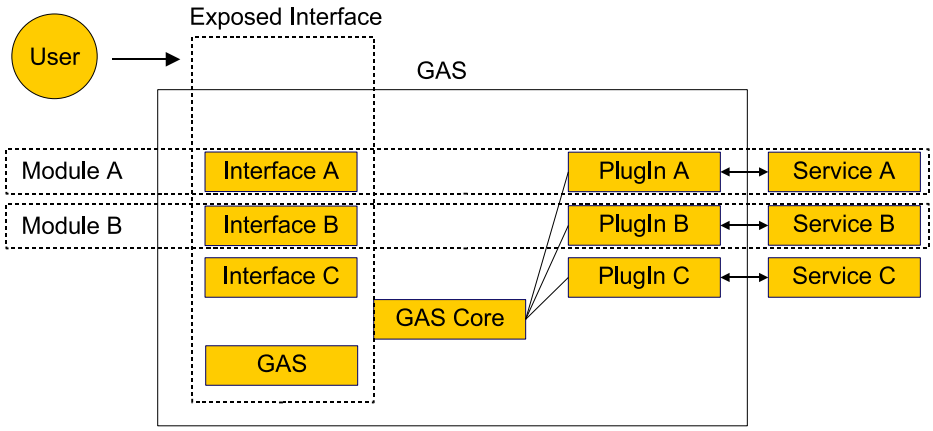

Figure 6.2: GAS architecture

The GAS is composed of several modules. Each module consists of an interface and a PlugIn. Each is connected to one service. The interface of each module is added to the exposed interface.

- The GAS is composed by a set of services. This composition is chosen by the user. Therefore a GAS suitable for a specific purpose can be configured by the individual user.

Its position in the overall context of the gLite middleware can be seen in Figure 5.2, page 69. All services in this figure are candidates for the usage with the GAS.

6.2.1 Architecture

The architecture of the GAS is shown in Figure 6.2.

It is made up of the following major components:

- **Interfaces**: These contain all functions which are made available. They are the connection to the user.
- **PlugIns**: These control the communication to the underlying services. They are the connection to the Grid middleware.
- **GAS Core**: This is the glue between the two previously mentioned parts. It receives tasks from the user and delegates them to the corresponding PlugIn.

The GAS consists of several individually configured modules. Each defines all necessary information concerning one underlying service. A module consists of the following components:

Figure 6.3: GAS multiplicities
One GAS contains an arbitrary number of modules. Each module belongs to one GAS and is connected to one underlying service. One service is connected to an arbitrary number of modules.

- **The Interface**: This contains the functions available in the module. The unification of the interfaces of all modules are exposed to the user. Due to the fact, that the interface of one module makes up only a part of the whole interface, it is also referred to by *interface snippet*.
- **The PlugIn**: It forwards calls to the underlying service. Details of the access method are given by the module's configuration.

It is important to understand the following relationship: one GAS has several modules. Each module contains the functions for a specific service type, for example the access to a file catalog. For each of these modules there are many services which can solve the corresponding tasks. One is picked by the user when the GAS is created and associated with this module. This individually chosen service is referred to as underlying service. The multiplicities which underlie this relationship are shown in Figure 6.3, examples of the complete structure are presented in Figure 6.4.

The user who requests the GAS selects a set of modules. Therefore a GAS suited to her personal needs can be created. However, compositions which are commonly used, can also be selected from a list of predefined configurations.

6.2.2 Workflow

There are three main periods in the lifetime of the GAS:

- **Authentication**: The first step for a user requesting a GAS is to authenticate herself. This means she has to prove that she is who she claims to be by exchanging certain keys and information.
- **Creation**: After successful authentication the GAS is created corresponding to the user's needs. This includes contacting the underlying services.
- **Usage**: When the GAS has been created tasks can be sent. The GAS sends them to underlying services and returns the results to the user.

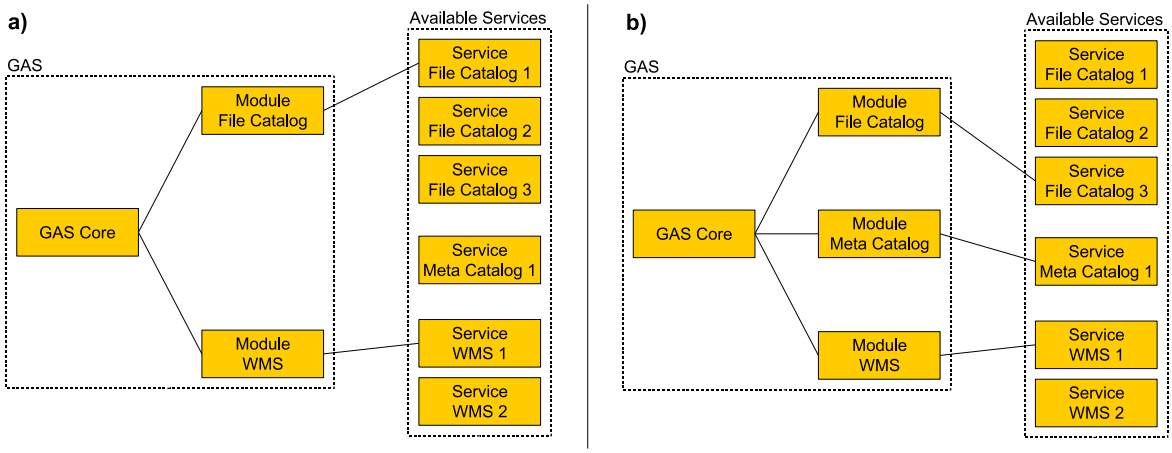

Figure 6.4: Examples of GAS structures

(a) Example of a GAS: It consists of two modules (File Catalog and Workload Management Service (WMS)), each of the modules is connected to one service (File Catalog 1 and WMS 1, respectively). (b) Another example of a GAS: An additional module can be seen, the Meta Catalog. Analogous to before, each module is connected to one service, but in this example the module File Catalog is connected to a different service providing the File Catalog functionality (File Catalog 3).
There are always many services which provide a given functionality. But only one is picked for each module.

Authentication

When the GAS is requested the user has to authenticate herself. Furthermore it is supposed to act on behalf of the user, e.g. submit a job. Therefore it needs a proxy certificate signed with the user's private key. This is solved by using credential stores. The authentication methods are explained in section 5.4.2: *Security Services*.

The authentication functionality is implemented in the gContainer which hosts the Grid Access Service. Implementation details can be found in section 6.3: *gContainer*.

Creation

When the GAS is created, the interface snippets of each module, whose functions are only able to perform a specific type of task, are combined to one flat interface. This interface is described by Web Service Description Language (WSDL) [Chr01] and exposed to the user. A set of generic management functions is added, e.g. to stop the GAS. The flatness of the interface implies that the user does not have to specify which

module she wants to communicate with. The mapping between a specific function and the corresponding module is done by the GAS.

The underlying service of each module is contacted. If necessary the GAS authenticates to the service impersonating the user. This is needed for sending tasks on behalf of the user.

Usage

The user can submit tasks to all modules which she requested with calls to the GAS. The GAS delegates the call to the specific module. In turn the module contacts the underlying service. If necessary the input of the call is transformed. The reply of the underlying service is received by the module, transformed if necessary, and sent back to the user.

If an underlying service is replaced, only the configuration of the GAS has to be changed. The interface of the GAS does not change, therefore no changes are necessary in applications or the user's behavior.

6.2.3 Implementation

The GAS has been implemented using the script language Perl [Per05a]. It was developed with Perl 5.8.5 on a 32-Bit machine running Scientific Linux CERN (SLC) [SLC05]. Due to the platform independency of script languages like Perl there should be no issues to run it on other machines or operating systems, which is also made sure by the testing team of the gLite project. For details how to retrieve the code see section D of the appendix: *Retrieving the Code*.

The GAS is implemented as a web service following the Web Service Resource Framework (WSRF) standard. It is hosted by the gContainer. These terms and relationships are explained in detail in section 6.3.1 in the context of the gContainer. For the time being it should be sufficient to understand that it inherits from the class *WSRFService*.

Class Diagram

The class diagram of the GAS is shown in Figure 6.5. The classes are divided into the groups: core classes, interface classes and communication objects. Furthermore the group of underlying AliEn services is shown which were the first services being interfaced with the GAS. However, more services have been interfaced which are not shown in this figure.

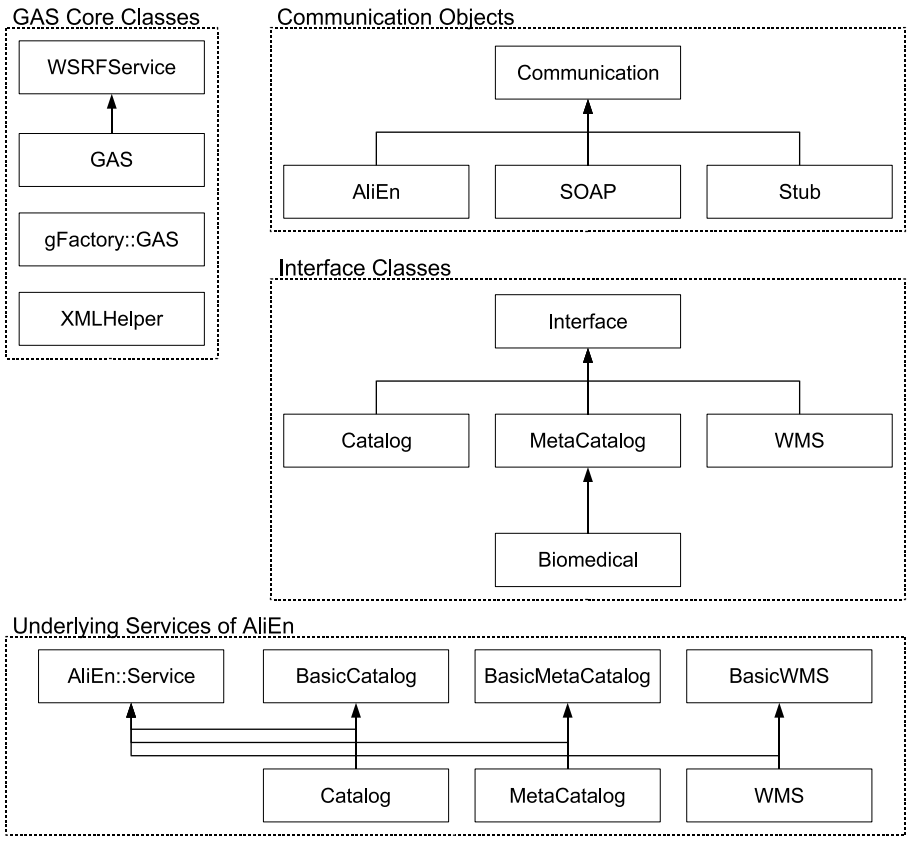

Figure 6.5: GAS class diagram

Module Structure

A module completely defines the access to one service of the Grid middleware. It consists of an interface class and a communication object. The interface could be described as the logic of the connection to the underlying service, whereas the communication object is the protocol. The structure of a module is shown in Figure 6.6. Its components will be outlined in the following paragraphs.

Communication Objects Communication objects are used to communicate with underlying services. Their base class contains three functions: *new*, *shutdown* and *delegate*. The first two connect and disconnect, respectively, to the underlying service. The *delegate* function sends the calls to the service.

Three communication objects are implemented:

Figure 6.6: The structure of a module
The colored boxes (Interface, PlugIn and Communication Object) are part of the module.

- **Communication::SOAP**: Its function is to connect to services using the Simple Object Access Protocol SOAP) [Gud03]. SOAP is a protocol for executing commands between different machines being independent of programming languages and operating systems. It is in use by web services and therefore the communication object which will be used for most of the services being integrated.
- **Communication::Stub**: Its function is to connect to services providing a Perl stub[1]. The stub is provided by the creator of the service and has to be integrated in the GAS system. This is used for special cases which do not fit into the usual communication schemes.
- **Communication::AliEn**: Its function is to connect to AliEn services. AliEn is a Grid middleware which is being developed in the context of the ALICE experiment. It is one of the projects which gLite is based on, therefore former AliEn services will be part of gLite and the GAS must be able to interface these.

Interface Classes An interface class has two purposes: firstly it defines which functions a module offers. Furthermore it acts as an adapter. An adapter provides a mapping between each function offered in its exposed interface to a function of the underlying service. In practice this is often a one-to-one mapping because the exposed interface is identical with the interface of the underlying service. However, transformations can be implemented if functions or parameters differ. Due to this dual-use architecture an interface class which is used in the first case is called *interface*, in the latter case it is referred to by *PlugIn*.

The base class consists of three functions: *new*, *shutdown* and *delegate*. The first can perform special initialization and destruction sequences, respectively. The *delegate* function sends the calls to the communication object which sends them to the underlying service. The interface class always uses the communication object to talk to the service.

[1] A *stub* is a piece of software which acts as a bridge to another software component. The latter can even be on a different machine. The communication between the stub and the component is transparent to the user.

The question may be raised why there are both an interface class and a communication object as both look very similar. The reasons is that this model allows a maximum reusability of the classes: Every interface class can be used with every communication object. E.g. a specific interface class like the file catalog could be accessible by SOAP, another by a Perl Stub. Then again another interface class like the metadata catalog could also use the SOAP communication object.

Three interfaces classes have been implemented so far:

- **Interface::FileCatalog**: An interface and PlugIn for File Catalog services.
- **Interface::MetaCatalog**: An interface and PlugIn for Metadata Catalog services.
- **Interface::WMS**: An interface and PlugIn for Workload Management services.

If an underlying service needs a modified interface class, it can be derived from a specialized interface. E.g. a Metadata Catalog designed for biomedical application was interfaced in this way: the class Interface::MetaCatalog::Biomedical inherits from Interface::MetaCatalog (see Figure 6.5).

Summarized the interface class fulfills two tasks. Firstly it is the interface snippet which is exposed to the user. Secondly it acts as an adapter to access the underlying service. This model reduces the amount of lines needed to be written for a new service that is to be integrated into the GAS.

Workflow

The workflow of the creation and the usage of the GAS were outlined abstractly in the previous section. Their implementation will now be explained in more detail:

Creation The function *GAS::initialize* is called and performs the following tasks:

- Initialize helper classes like XMLHelper.
- Setup the secure communication: Load the user's proxy certificate and allow only the user of the GAS and the hosting gContainer to talk to it.
- Determine lifetime and set timer for destruction at expiration.
- Read the configuration which contains a list of modules which are to be loaded into the GAS. If the configuration was not passed by the user (this could be either

explicitly, with the request for the GAS, or defined in the user's environment) it is read from the global configuration of the Virtual Organization.

- Expand the information about the modules: Read all information necessary for creating the modules from the global configuration of the GAS system.
- Parse the interface snippet of each module. A mapping is created between the functions of the interface snippets and the modules. This allows to associate a function call with a module, which is needed because the GAS exposes a flat interface to the user.
- Create the requested modules. For each module the needed communication object is created. Then the PlugIn is created.
- If all mandatory modules could be created, and these could contact the underlying services, report successful initialization.

Usage The interface of the GAS is composed of management functions and interface snippets, one for each module. Only the first are implemented in the GAS class itself, a call to one of the others has to be forwarded to the corresponding module. In the latter case the call is forwarded by using the *AUTOLOAD* functionality of Perl. This allows to react to calls to functions which are unknown to the class in which they are called. The mapping between functions and modules, which was created in the initialization phase, then determines which module the requested function belongs to. The call is forwarded to the corresponding module and the result returned to the user.

6.2.4 Configuration & Integration of a New Module

A module consists of an interface, a PlugIn and a communication object (Figure 6.6). All three elements have to be provided for the integration of a new module.

1. **Interface**: The GAS is equipped with interfaces and PlugIns for all common services, like file catalogs, metadata catalogs and workload management. So usually the interface which is needed for the service being integrated is already available. However, if a new type of service is integrated, a new interface has to be created.
2. **PlugIn**: If the interface of the new service is identical with an already existing one also its PlugIn can be used. If the interface differs a new PlugIn has to be created, which is derived from the basic PlugIn for this service type. Only if a new service type is integrated a new PlugIn has to be created. In this case,

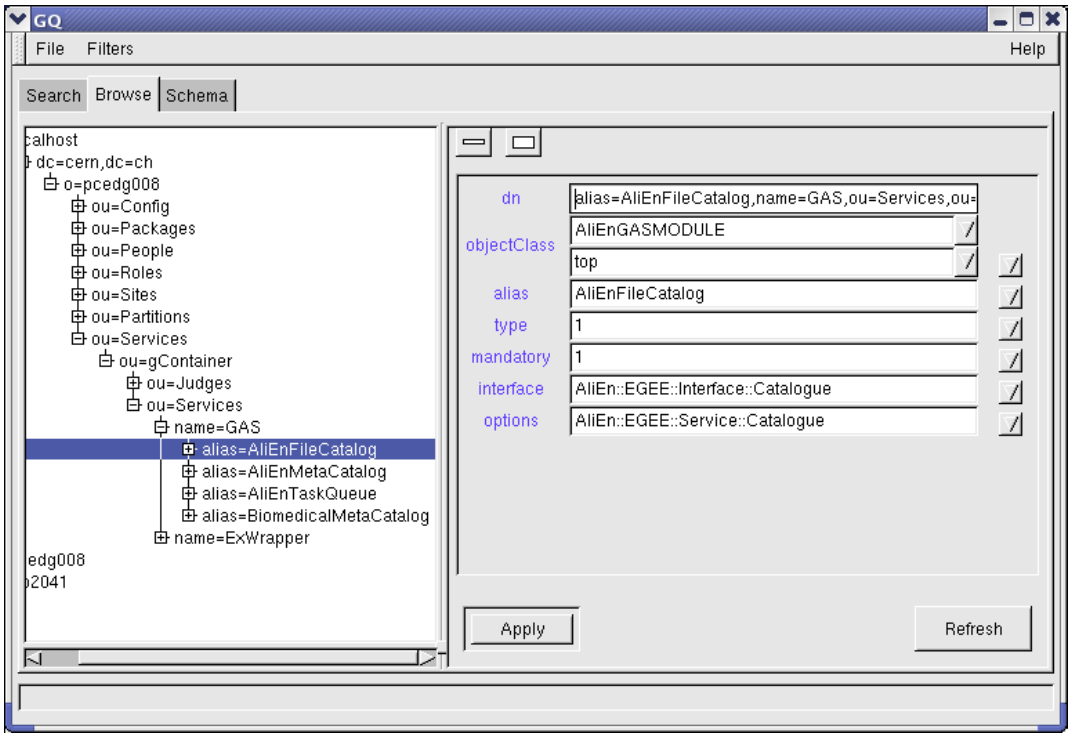

Figure 6.7: A module's configuration in LDAP
The configuration of the module AliEnFileCatalog can be seen which interfaces the File Catalog service of AliEn.

---

the interface and the PlugIn are implemented in the interface class of the new module.

3. **Communication Object**: A communication object has to be selected and corresponding options have to be provided. This includes information on how to access the service.

The most common case is that a new service for an already existing type is available, e.g. a file catalog designed for a special application. Usually this service has the same interface as existing ones, so no new classes have to be created. Only the information about the module has to be put into the configuration of the GAS.

Currently the configuration is stored using the Lightweight Directory Access Protocol (LDAP) [Hod02]. For each module an entry is created in the *Services/gContainer/Services/GAS* tree. An example of a module configuration can be seen in Figure 6.7.

The entry contains the elements:

- **Alias**: A name for the module.

- **Type**: The ID of the communication object used.
- **Mandatory**: If this flag is set the module is marked as mandatory. The mandatory state implies that in case a user requests this module and there is a problem during the startup of this module, the GAS startup will fail completely. This has to be seen in contrast to the fairly common situation where a single missing module is not followed by a complete failure.

  The mandatory flag can be set if a GAS is used by an application which relies on the availability of certain modules.
- **Interface**: The interface class which is used.
- **Options**: Options which are passed to the communication object. Usually this contains information on how to access the module.

6.3 The gContainer

Introduction

It was stressed before that the Grid middleware consists of many services where each is responsible for a specific purpose. The services can be taken up by many users at different physical locations. Therefore they are distributed across different sites and thus, in their complexity, form a diversified and distributed service pool. Although the services have different functionality, all of them have a number of qualities in common. Each of the services must have an authentication mechanism which allows verification of user's credentials. They have to report to an information system which is able to keep an overview of the Grid environment. It is also desirable that services react to changes of the load[2] of the system which is hosting it.

It would be a waste of effort to implement this functionality with every new service. Therefore a container has been created which is able to host various types of services. The container offers functionality that is generally needed and is therefore well suited for offering the previously identified tasks like authentication, information and load management.

The container which has been created is called *gContainer*. A schematic view can be seen in Figure 6.8.

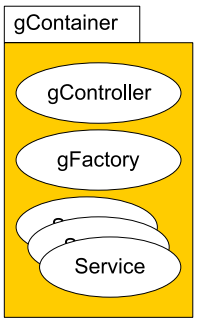

Figure 6.8: The gContainer – A web service container

6.3.1 Architecture

The services of gLite appear as web services. Many of them follow the standard defined by the Web Service Resource Framework (WSRF) [WSR04]. This recently published

[2]The *load* is a measure of the utilization of a system.

standard for web services defines terms like *state* and *resources*. The gContainer is able to host all services following the WSRF standard. However, there are also means of hosting services which are not compliant to this standard.

The Web Service Resource Framework (WSRF)

The Web Service Resource Framework is a comprehensive standard. Outlining the whole standard would by far exceed the scope of this thesis. Therefore only terms which are necessary for the architecture of the gContainer will be shown.

The gContainer is a web service container. In the WSRF a web service is defined as: *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards"* [Fos04b].

To put it in a nutshell, a web service is a software component at a specific location. It offers a set of functions which are exposed in the form of an interface. Web services can be stateless or stateful.

According to the WSRF *"a stateless service implements message exchanges with no access or use of information not contained in the input message."*
The result of a call to a stateless service only depends on the input. No other data is taken into account. Therefore all calls to a stateless service with identical input lead to identical output at any given time.

A *stateful* service is defined as *"a service that acts upon stateful resources [and therefore] provides access to, or manipulates a set of logical stateful resources (documents) based on messages it sends and receives."*
A resource is a set of data. It can for example appear as a file or a database. The service processes a call using this resource. Consequently it takes more information into account than the caller provides as input. Therefore identical calls to the service do not necessarily lead to identical output.

The implementation model which is used in the gContainer combines a service with the associated resource. Therefore what is called *"a service that acts upon stateful resources"* in the previous definition will be simply called *stateful service*.

WSRF and the gContainer The gContainer will be able to support stateful and stateless services. It is important to differentiate between these two types. In the first case the output is just determined by the input. In case of several instances of a stateless service it is therefore insignificant which instance replies to a call. Consequently the gContainer is able to react on changing load. If many users want to use the same service it can create additional ones (see *Load Management* below).

This is different for stateful services. Subsequent calls have to be replied by the same instance. Otherwise they would not interact with the same state and would, from the user's point of view, produce indeterministic results. Therefore no load management can be applied to stateful services. However, in certain cases several instances of a stateful service can be created if each of them is associated with one user and therefore each of the users interacts with their own state. Usually an instance of a stateful service is used by only one user, and an instance of a stateless service by several users.

If a service acts on behalf of a user it needs a proxy certificate (see section 5.4.2). In this case it is called *impersonating service*.

Functionality

The gContainer hosts services and offers the following functionality to them:

- **Authentication and Authorization**: For most services it is necessary to authenticate the user because sensitive data is to be accessed. The authentication and authorization is based on the user's certificate which was explained in detail in section 5.4.2.
- **Service Discovery**: A typical scenario is a user requesting a service to perform a certain task. This service could be offered at different locations: physical locations like different institutes and logical locations like different machines in the same computing center. It would be quite troublesome if a user had to take it upon herself to pick the best location. The concept of service discovery implies that the system determines transparently the most suitable location to create the service. One example of aspects which are taken into account are the distances between the user and the potential locations. Another example is the load. The requested service is then created at the most suitable location. This process is called *service discovery*. The gContainer should be created in a way that the arguments which are taken into account can be extended or modified easily.
- **Load Management**: Stateless services are used by many users. One instance of a service can only handle one call at a time. Therefore it is convenient to

create additional ones when the number of users exceeds a certain limit. In turn instances should be removed to free resources when the number of users decreases significantly.

- **Lifetime Management**: Usually every user requesting a service should trigger its destruction when she does not need it anymore. However, this cannot be guaranteed because it could be forgotten or fail due to e.g. network problems. Abandoned services would pile up and congest the system. Therefore a lifetime is associated to each service. When the lifetime expires the service is destroyed to free resources. However, if a service is used constantly, the lifetime is prolonged.

All the points mentioned above are integrated in the gContainer and therefore do not have to be integrated in each service anymore. This allows creators of new services to concentrate on the development of the core functionality of the new services.

Structure

The gContainer consists of three components: the *container* itself, a management service called *gController* and a factory service called *gFactory*. Each gContainer has exactly one gController and one gFactory service (see Figure 6.8). Different instances of the gContainer communicate with each other in a structure which is outlined below.

Container The container hosts WSRF services. It contains technical functionality like accepting connections and provides secure communication. It removes services whose lifetimes expire.

gController The gController is the management service of the gContainer. It keeps track of the running services, which includes checking if all services are alive and removing those which do not respond anymore. The gController advertises its capabilities, in particular the services it can start, to the parent gContainer. It accepts information from child gContainers advertising their capabilities and also propagates them to its parent. For each user requesting a service the gController can use this information to determine which location is best for creating the requested service. The gController monitors the load on specific services. It dynamically creates additional or removes existing instances if the load exceeds certain thresholds, which as mentioned before can only be performed with stateless services. The gController is a stateful service, necessitated by the information it has to store.

gFactory The gFactory is the factory service of the gContainer. It creates services which are requested by users. In this process it contacts the gController to determine the best possible location for creating the service. The possible locations can include the gContainer itself or any child gContainer. Like described above the gController also decides which location would be best suitable for the user. If the best location is the gContainer itself the gFactory creates the requested service. In turn if the best location is another gContainer, it is contacted and the call is forwarded to the associated gFactory.

If the new service is an impersonating service the gFactory picks up the needed user's proxy certificate at a credential store.

The gFactory is a stateless service. It retrieves needed information for the creation process from the gController.

6.3.2 Implementation

Analogous to the GAS, the gContainer is developed in Perl 5.8.5 on a 32-Bit machine running Scientific Linux CERN (SLC). For details how to retrieve the code see section D of the appendix.

Several software projects have turned out to be able to host services following the WSRF standard. Some of these are open source projects and allow reusing without the need to pay license fees. It has been decided to build the gContainer on top of the open source project WSRF::Lite [MK05]. This project offers hosting of services following the WSRF standard and provides security by the Secure Socket Layer (SSL) protocol [Fre96]. The main goal of the development was to add functionality, like parts of authentication, service discovery and load management, without modifying the code of the WSRF::Lite project. In case of modifications to the original code, migration to a newer version of WSRF::Lite would be very complicated.

Class Diagram

The class diagram of the gContainer can be seen in Figure 6.9. It is divided into classes belonging to the gController, the gFactory and the container itself (called *gContainer core classes* in the figure). Exemplarily, two services which are hosted by the gContainer are also shown: the GAS and the Executable Wrapper (ExWrapper). The ExWrapper is a service which is able to start arbitrary applications[3] which then can be managed by the gContainer.

[3]This gives the possibility to host services which do not follow the WSRF standard.

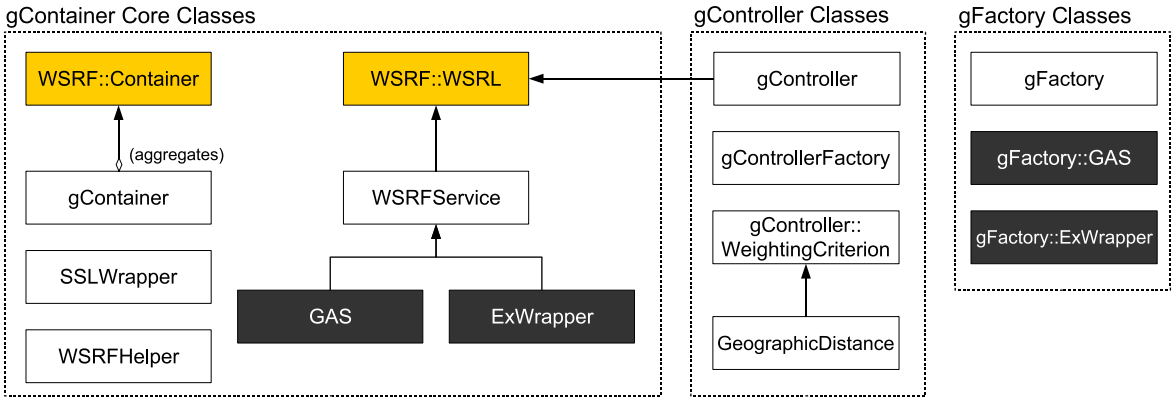

Figure 6.9: gContainer class diagram

The classes marked in yellow originate from the WSRF::Lite framework. The classes marked in black are examples for hosted services. In addition to each hosted service (e.g. GAS) there is a factory class in the gFactory namespace (e.g. gFactory::GAS). Not shown in the diagram is that the classes GAS and ExWrapper are wrapped by the SSLWrapper class. This enables the use of certificates as security method.

Communication among gContainers

The gContainer offers service discovery by means of communicating with other instances of gContainer. Each gContainer has one parent and an arbitrary number of children. Regularly, it advertises its capabilities and the capabilities of its children to its parent. A hierarchical tree structure evolves which is depicted in Figure 6.10. The relationships of this structure are shown in Figure 6.11.

The instances of gContainer communicate the services which they are able to create between each other. For this purpose each gContainer keeps two lists of services. The first contains the services which the gContainer itself is able to create. The second list contains the services which its children can create.

The unification of both lists, which contains all services which the instance itself and the children can create, is reported to the parent. The resulting list does not include information about the specific location where the services could be created. Therefore the parent only knows that one or more gContainers in a child branch can create a service, not where in the branch it could be created.

When a service is requested a list of possible locations is built up. This list can include the gContainer itself and an arbitrary number of children. Then these are weighted. It should be pointed out that within the weighting process there is no information if a child creates the service directly or sends the creation request to one of its children.

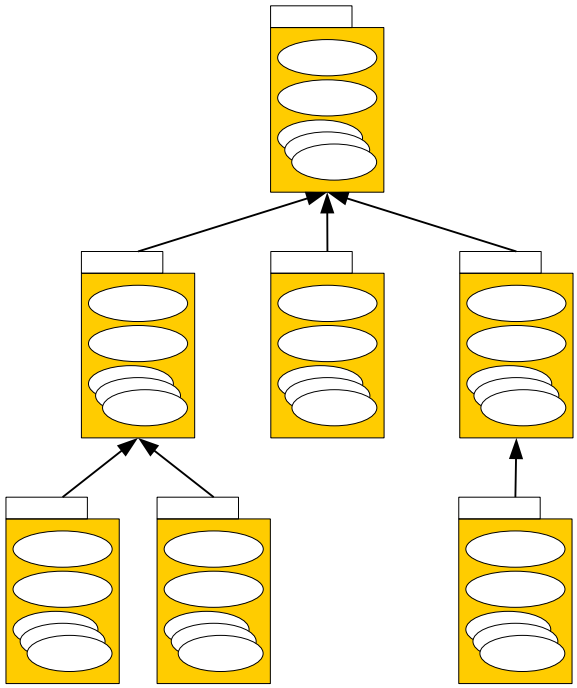

Figure 6.10: Hierarchical tree structure of several gContainers

The communication and weighting model has been chosen to be simple and lightweight. This allows the code and the process to be very understandable and transparent. It implies that the hierarchy of the different instances of gContainer is well-chosen.

To experts this hierarchical model may seem to hold a single point of failure: the gContainer at the very top. But this is not the case because a user must not necessarily talk to the top level gContainer. Several gContainers at the second level can be used as a fallback solution. These are not able to create services in the whole tree of gContainers, but cover an adequate part of it.

Startup

First the container is started, which, after startup, accepts calls to factories. The factory of the gContainer is called and creates the gContainer. Its startup undergoes a process of three steps:

- A logging object is created and the secure connection is initialized.

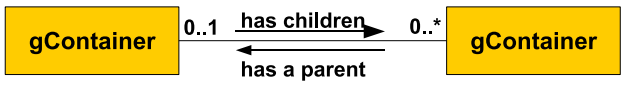

Figure 6.11: Relationship diagram of the gContainer
One gContainer has an arbitrary number of children. In turn every gContainer has a parent, but there is one exception: the top level gContainer.

- The configuration is read. This includes the services the gContainer can start, the available classes containing criteria for service discovery and the address of the parent gContainer.
- A process is created which performs regular tasks like load management, reporting to the parent gContainer and alive checking.

After initialization the gContainer consists of the following processes:

- **Container**: The main process of the gContainer. It accepts calls and forwards them to the corresponding service.
- **gController**: The manager service of the gContainer.
- **gController – Process for regular tasks**: a subprocess of the gController. It performs regularly executed tasks, which are explained in the next paragraph.

For each new service a new process is created. However, this process can trigger further processes. The gFactory, as stateless service, is created upon a user request and therefore is not denoted here as active process.

Regular tasks

The mentioned process for regular tasks performs the following operations:

- **Reporting**: The gController reports to the parent of its gContainer which services it is able to create. Therefore it builds up a list of the services which can be created by its children and the gContainer itself. This list is sent to the parent.

  Reporting also serves the purpose of informing the parent that the operating gContainer is still running. In case that an instance does not report for a certain period of time it will be removed from the list of children.

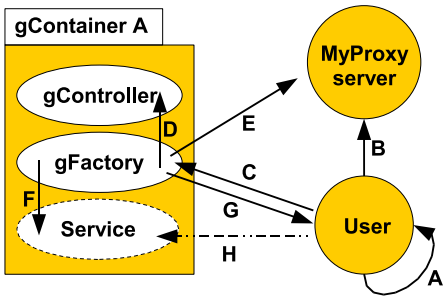

Figure 6.12: Use case: service request, example 1

- **Load Management**: The utilization of all stateless services is checked. If certain thresholds are exceeded the creation of additional services is triggered. The load management model, which is used, is described below.
- **Alive checking**: The gController checks every service which is hosted by the gContainer. Each service implements a method for this task. If a service is not responding anymore, its resources are freed and it is removed from the list of running services.

Service Request

If a user requests a service, the gContainer creates a list of potential locations which are able to create the requested service. This list can contain the gContainer itself and any child which is able to create it. These locations are weighted to find the most suitable one. Criteria can be the distances between the user and the locations, distances between other components the requested service relies on, or the load at the locations. Afterwards the service is created at the most suitable location and the address of the new service is returned to the user. However, if a location should unexpectedly be unable to create the service, the second best location is tried and so on. Naturally, the process includes contacting of other instances of the gContainer.

Two examples will illustrate this use case *Service Request*. In the first example the service is created at the gContainer which received the service request from the user. The latter one shows the involvement of other gContainers.

The first example can be seen in Figure 6.12. Its steps are explained below:

- **A**: The user creates a proxy certificate.

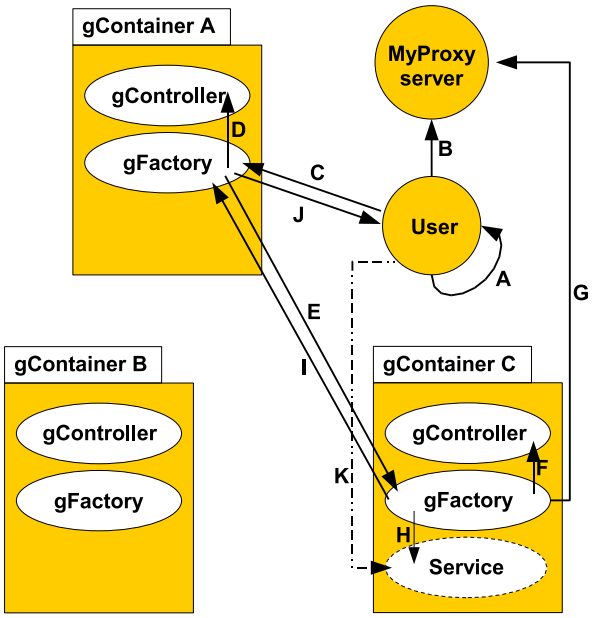

Figure 6.13: Use case: service request, example 2

- **B**: The proxy certificate is stored in a credential store. gLite uses a MyProxy server.
- **C**: The user submits a service request to the gFactory of a gContainer. The service request contains details about the requested service and the user.
- **D**: The gFactory queries the gController for the best location to create the service. In this case the gController advises to create the service locally.
- **E**: The gFactory picks up the proxy certificate from the MyProxy server. The certificate is passed to the service which is created in the next step.
- **F**: The requested service is created and initialized.
- **G**: The address of the service is returned to the user.
- **H**: The user talks to the service to have the needed tasks performed.

Steps A, B and E which are related to the creation and storing of the user's proxy certificate are optional. They are needed for impersonating services which act on behalf of the user. For other services these steps are skipped.

The second example is shown in Figure 6.13. In contrast to the first example, the best location is not considered the gContainer A which received the service request by the

user. Instead of that the call is forwarded to another gContainer C. The steps in detail are presented below:

- **A**: The user creates a proxy certificate.
- **B**: The proxy certificate is stored in a MyProxy server.
- **C**: The user submits a service request to the gFactory of gContainer A. The service request contains details about the requested service and the user.
- **D**: The gFactory queries its gController for the best location to create the service. In this case the gController advises to forward the request to gContainer C.
- **E**: The gFactory of gContainer A forwards the service request to the gFactory at gContainer C. gContainer C now follows the same pattern as gContainer A did before.
- **F**: The gFactory of gContainer C queries its gController for the best location to create the service. The gController decides to create the service locally. However, it could also have decided to forward the call to one of the children of gContainer C.
- **G**: The gFactory picks up the proxy certificate from the MyProxy server. The certificate is passed to the service which is created in the next step.
- **H**: The requested service is created and initialized.
- **I**: The address of the created service is returned to the requestor, which is the gFactory of gContainer A.
- **J**: The gFactory of gContainer A returns the address of the new service to the user.
- **K**: The user talks to the service to have the needed tasks performed. It should be noted that the user contacts the service directly, without the need of gContainer A.

Comparing the two examples it can be seen that in the second example a few steps are inserted after step D: the call is forwarded to gContainer C. gContainer C basically performs the same steps (F-I) as gContainer A did in the first example (D-G). Then the address is returned to gContainer A which returns it to the user. This process is completely transparent to the user.

Example of Weighting Criteria – Geographic Distance

In the process of service creation a list of potential locations is created. These are weighted on the basis of certain criteria. One example is the criterion called *Geographic Distance* which determines the geographic distance between locations in the internet. It uses a database which contains a mapping between IP address ranges and geographic coordinates. A trivial calculation results in the distance between two coordinates.

It should be noted that the geographic distance is only a rough estimation of the latency between two systems in the internet. Other much more sophisticated methods could also be implemented.

Requesting an Existing Service

When a user does not need a service anymore she can request its destruction. She could also leave the service running to reuse it at a later time. For this purpose a possibility is provided to ask for existing services. Upon a request the gController searches in the list of running services for a service which belongs to the requesting user and, if applicable, returns it.

Load Management

A simple load management model for stateless services has been implemented. The point in time of each request for a stateless service is recorded. If the number of requests in a certain period of time exceeds a threshold the creation of additional instances is triggered. In turn instances are removed if the value drops below a certain limit. It has to be made sure that instances which are to be removed are not in use anymore. The thresholds can be configured separately for every service.

If several instances of a given kind exist the instance which is returned to a user upon a request is selected round-robin[4].

The implemented model can clearly be improved in many ways. The actual load of each service should be determined by asking the service. This functionality could be added to the base class *WSRFService* which is inherited by all services.

[4] *Round-robin* is a simple scheduling model which assigns the same amount of resources to every requestor. In this context round-robin means that the instances are returned sequentially in the order in which they are listed.

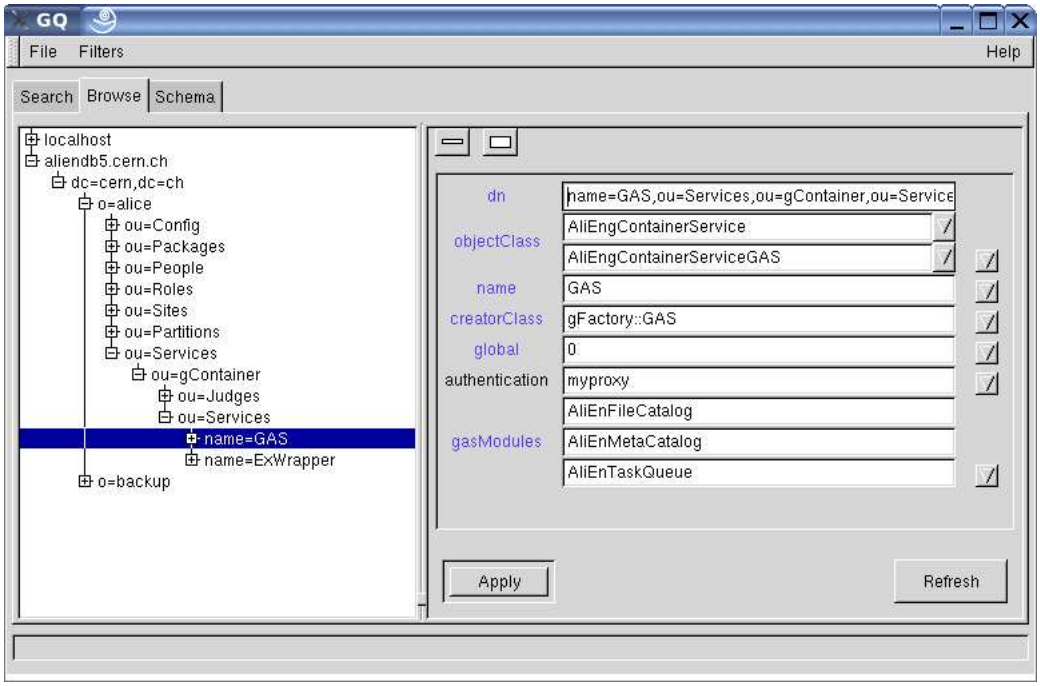

Figure 6.14: Configuration of a service which is hosted by the gContainer
The configuration of the Grid Access Service can be seen. Its factory class, the authentication method and further specific options are visible.

6.3.3 Configuration & Integration of a New Service

All services which are integrated into the gContainer have to inherit from the class *WSRFService* which implements the functions needed for the communication with the gController. Furthermore the class implements basic functionality which is demanded for all services following the WSRF standard.

A factory class for the new service has to be provided and put into the namespace *gFactory*. Additionally, an entry has to be added to the configuration. If necessary a new set of criteria, different from the standard one, can be provided, which is used to determine the most suitable location when the service is requested. If the service needs a proxy certificate which is available at a MyProxy server, this can also be set in the configuration. The gContainer will pick up the proxy upon a service request, so the new service does not have perform this by itself.

An example configuration can be seen in Figure 6.14.

6.4 Conclusions

The Grid Access Service which has been developed interfaces different services. These range from services of the AliEn framework to newly developed services of gLite. It has turned out that it is very important to develop interfaces which support all functionality the underlying services have or might have. This should be done in cooperation with the development teams of the services to make sure that their needs are taken into consideration; so that in a final stage the GAS supports all necessary functions and fulfills all users' requirements.

The gContainer, which has been engineered, hosts the Grid Access Service and the APIService. The latter is used as a bridge between the high energy physics analysis framework ROOT and gLite. Furthermore the idea came up that the gContainer could be used to manage most of the services of the Grid middleware.

Both the Grid Access Service and the gContainer, offering the functionality described in the previous sections, have been successfully deployed in the gLite testing environment.

They can now be evaluated by the users and feedback for the next version can be collected. In the next version of the gContainer the load management model and the weighting process for service discovery should be enhanced. Furthermore, the communication between the gContainer and the information services of the Grid should be implemented.

7. Summary

From 2007 on, data taken by ALICE will give hints if the quark-gluon plasma was found and support its understanding. One of the signatures is a change of the quarkonia yields. A detailed view of this new phase will prove or disprove theories which describe strongly interacting matter, and provide a deeper understanding of the early evolution of the universe. To be able to interpret the measurements a detailed understanding of the detector and the signals which are to be expected is needed. For this purpose simulations are an important tool to learn about the performance of the detector. However, simulating a number of events, which corresponds to e.g. one month of data taking, is not feasible when the straightforward way, called slow simulation, is used. The amount of needed computing time exceeds the available resources by far.

Nevertheless, one way to make these simulations, e.g. for quarkonia states, possible is to parameterize the performance of the ALICE central barrel. In this thesis, it has been described by response functions which represent the behavior of the detector for electrons and positrons. They provide information about the detector efficiency, its resolution and the particle identification in respect to electrons and pions. Thus the parametrization gives a detailed view of the capabilities of the central barrel. A result is that the tracking is efficient in sensitive detector areas; however, about 5% of the tracks, in an event with the multiplicity 4000, are reconstructed at significantly wrong momenta. The impact of dead zones, especially on high momentum tracks, can be seen. This deeper understanding of the detector can also be used to optimize the tracking routines.

A direct comparison between slow and fast simulations, based on the response functions, shows that results from slow simulations are properly reproduced on the analysis level. The response functions make fast simulations possible that involve electrons and positrons, at a processing time which is about 10,000 faster than slow simulations. Therefore fast analyses are enabled, even for rare particles which have very low rates.

The gained response functions solve the problem of lacking computing resources, for simulations which are performed before the experiment starts. However, the amount of data, which will be taken from 2007 on, and its analysis makes enormous demands on the computing infrastructure. Many institutes will have to combine their resources and perform so-called distributed analysis. The software which supports and organizes the use of the resources is called Grid. Two components of the Grid middleware gLite, that simplify the usage of the Grid, have been developed and deployed in the context of this thesis. Furthermore, interfacing of the ROOT framework with the Grid has been

introduced. The Grid is on a good way and it is to hope that in 2007 it will provide the necessary infrastructure to fulfill all demands from physicists for high energy physics analysis.

An analysis of the quarkonia signals, in particular for the particles J/Ψ, Ψ', Υ, Υ' and Υ'', has already been performed in [Som05b] using the response functions determined in this thesis. Preliminary results indicate that these states will be visible. However, an increase in the pion rejection capabilities would significantly improve the signals. The use of neural networks and a weighted combination of the PID probabilities of the TPC and TRD could improve the pion rejection on the analysis level. Further investigation on how to combine the particle identification from the different detectors is necessary.

A. The ALICE Coordinate System

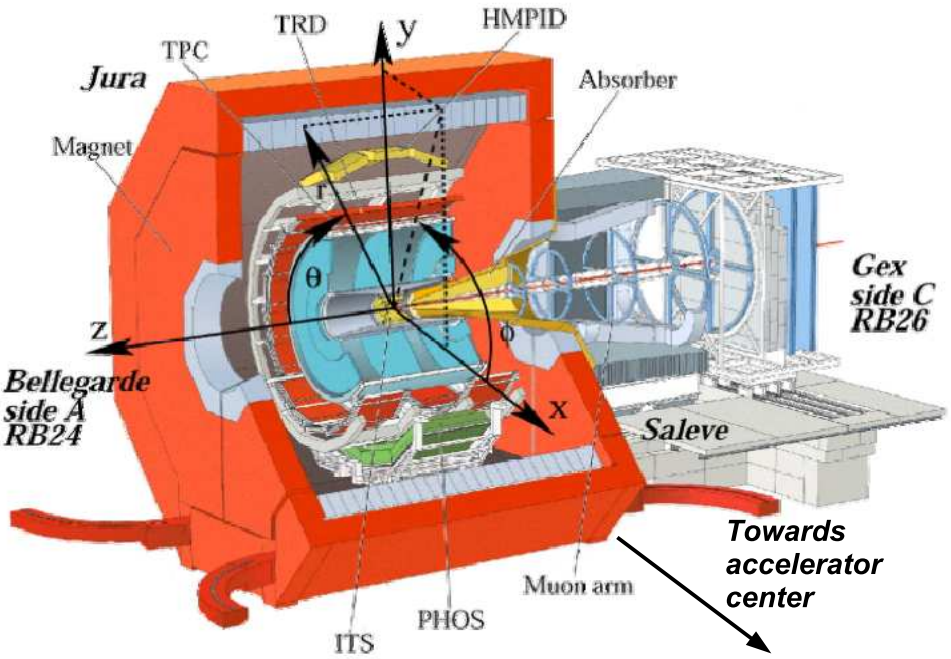

Figure A.1: The coordinate system of ALICE [ALI03]

The coordinate system of ALICE (Figure A.1) defines

- **the point of origin** $x = y = z = 0$ at the interaction point;
- **the x-axis** perpendicular to the mean local beam direction, aligned with the local horizontal and pointing to the accelerator center. Positive x is directed from the point of origin towards the accelerator center;
- **the y-axis** perpendicular to the x-axis and the mean local beam direction, pointing upward. Positive y is directed from the point of origin upward;
- **the z-axis** parallel to the mean local beam direction. An observer looking to positive z has the accelerator center on the left. The muon arm is at negative z;
- **the polar angle** Θ which increases from z ($\Theta = 0$) passing the x-y plane ($\Theta = \pi/2$) to $-$z ($\Theta = \pi$); at $\Theta = \frac{\pi}{2}$ the rapidity y, explained in the following section, is 0. This is called *mid-rapidity*.

- **the azimuthal angle** ϕ which increases clockwise from x ($\phi = 0$) passing y ($\phi = \pi/2$) to x ($\phi = 2\pi$) with the observer standing at negative z and looking towards the point of origin.

B. Kinematic Variables

The description of ultra-relativistic heavy-ion collisions uses several specific variables.

A particle, with the energy E and the momentum $\vec{p}$, is described by its *four-momentum*[1]

$$P^\mu = (E, \vec{p}) = (E, p_x, p_y, p_z). \tag{B.1}$$

This allows to calculate its *invariant mass* m_{inv}, which is invariant under Lorentz transformation and thus the same in all reference frames.

$$m_{inv}^2 = P^2 = E^2 - \vec{p} \cdot \vec{p} \tag{B.2}$$

The invariant mass of a free particle is equal to its *rest mass* m_0.

In high energy physics, the energy of a collision is given in the center of mass system of the colliding particles with

$$\sqrt{s} = \sqrt{P_1 + P_2} \tag{B.3}$$

where P_1 and P_2 are the four-momenta of the colliding particles. In case of collisions of ions, the energy of collision is given per nucleon pair and denoted with $\sqrt{s_{NN}}$.

The momentum of a particle is divided into its *longitudinal momentum* p_l and *transverse momentum* p_t:

$$p \quad = \quad |\vec{p}| = \sqrt{p_l^2 + p_t^2} \tag{B.4}$$

$$p_l \quad \overset{\text{(ALICE}^2\text{)}}{=} \quad p \ \cos(\Theta) = p_z \tag{B.5}$$

$$p_t \quad \overset{\text{(ALICE)}}{=} \quad p \ \sin(\Theta) = \sqrt{p_x^2 + p_y^2} \tag{B.6}$$

The transverse momentum is invariant under Lorentz transformations. Contrary the longitudinal momentum is not invariant, which led to the definition of the *rapidity* y.

$$y = \frac{1}{2} \ln \left(\frac{E + p_l}{E - p_l} \right) \tag{B.7}$$

y is not invariant under Lorentz transformation, but additive. However, the determination of the rapidity is complicated because p_l cannot be easily measured. Thus the *pseudorapidity* η is defined for the case of $E \gg m_0$:

$$\eta = \frac{1}{2} \ln \left(\frac{p + p_l}{p - p_l} \right) \overset{\text{(ALICE)}}{=} - \ln \tan \frac{\Theta}{2}. \tag{B.8}$$

[1] The commonly adopted convention of $\hbar = c = 1$ is used.

[2] The ALICE coordinate system was outlined in the previous section.

For ultra-relativistic particles the rapidity y is identical to the pseudorapidity η.

Eq. (B.2) can also be used to calculate the rest mass of a particle which decayed into several others, whose momenta have been measured. The calculation for the case of two particles (momenta $\vec{p}_1$ and $\vec{p}_2$) and in the limit of $E \gg m_0$, and thus $E \approx p$, results in

$$\overset{\text{(ALICE)}}{\Rightarrow} m_{inv} = \sqrt{2p_{t,\pi}p_{t,e}\left[\cosh(\eta_e - \eta_\pi) - \cos(\phi_e - \phi_\pi)\right]}. \tag{B.9}$$

C. ACBRESPONSE Package

The response functions, which have been created in this thesis, are available in the package ACBRESPONSE. It can be compiled on any system having AliROOT installed. The resulting shared library can be loaded into AliROOT to have access to its classes.

The main class is the class AliResponses. At initialization the multiplicity the user likes to use has to be picked. This is done in form of passing the file which contains the response functions. The default behavior is to use response functions representing heavy-ion collisions with $dN_{ch}/dy = 4000$.

The functions of the class AliResponses are listed below. Their definition uses the platform-independent ROOT variables. The variables *pt*, *theta*, *phi* are of the type Double_t which is omitted for visibility.

- **AliResponses(const char* *filename*)**: Constructor of the class. Creates it using the response functions contained in the file given in *filename*.
- **Bool_t IsElectronRecognizedAsElectron(*pt*, *theta*, *phi*)**: Returns if an electron with the given momenta is identified as electron by querying a probability distribution based on the electron-pion separation efficiency response function.
- **Bool_t IsPionRecognizedAsElectron(*pt*, *theta*, *phi*)**: Returns if a pion with the given momenta is falsely identified as electron by querying a probability distribution based on the electron-pion separation efficiency response function.
- **Bool_t IsRecognized(*pt*, *theta*, *phi*)**: Determines if a particle with the given momentum is detected by querying a random distribution based on the efficiency response function.
- **Double_t GetEfficiency(*pt*, *theta*, *phi*)**: Returns the efficiency at the given momentum as a value of [0,1].
- **Double_t GetElectronEfficiency(*pt*, *theta*, *phi*)**: Returns the electron efficiency at the given momenta.
- **Double_t GetPionEfficiency(*pt*, *theta*, *phi*)**: Returns the pion efficiency at the given momenta.
- **Double_t GetSmearingPhi(*pt*, *theta*, *phi*)**: Returns the fit parameter σ of the fit to the ϕ-resolution histogram at the given momenta (see Figure 4.8).

- **Double_t GetSmearingPt(*pt*, *theta*, *phi*, Int_t *par-id*)**: Returns the fit parameters of the fit to the p_t-resolution histogram at the given momenta (see Figure 4.8). Out of the 6 existing fit parameters, the one given in *par-id* is returned. *par-id* can be 0 for ξ, 1 for x_0, 3 for σ, 4 for b, 6 for α, 7 for k/l (see section 4.4.5 for details).

- **Double_t GetSmearingTheta(*pt*, *theta*, *phi*)**: Returns the fit parameter σ of the fit to the Θ-resolution histogram at the given momenta (see Figure 4.8).

- **void SetPIDModeLQ()**: Enables that the electron-pion separation efficiency values gained by the LQ-method are used. See section 4.4.5 for details.

- **void SetPIDModeNeuralNetwork()**: Enables that the electron-pion separation efficiency values gained by neural networks are used. These values are not yet available. See section 4.4.5 for details.

- **void SetSeed(UInt_t *seed*)**: Sets the seed *seed* of the random generator used for the probability distributions.

- **void Smear(*pt*, *theta*, *phi*)**: Smears the given momentum values based on the resolution response function. The variables have to be passed to the function as references.

D. Retrieving the Code

Released packages of gLite can be retrieved at its web site *http://www.glite.org*.

Code which is under development is available in the gLite CVS server *jra1mw.cvs.cern.ch*. The GAS and the gContainer can be found in the repository *org.glite.alien.gas*. Although the developed Grid middleware is free software and its released components can be freely downloaded, the CVS with components being still under development is only accessible for authorized users.

The gContainer is based on WSRF::Lite which is available at *http://www.sve.man.ac.uk/Research/AtoZ/ILCT*.

E. Bibliography

[ALI99] ALICE Collaboration, **ALICE Technical Design Report of the Inner Tracking System (ITS)**, 1999, CERN/LHCC 99-12, *https://edms.cern.ch/document/398932/1*

[ALI00] ALICE Collaboration, **ALICE Technical Design Report of the Time Projection Chamber**, 2000, CERN/LHCC 2000-001, *https://edms.cern.ch/document/398930/1*

[ALI01] ALICE Collaboration, **ALICE Technical Design Report of the Transition Radiation Detector**, 2001, CERN/LHCC 2001-021, *https://edms.cern.ch/document/398057/1*

[ALI02] ALICE Collaboration, **ALICE Addendum to the Technical Design Report of the Time of Flight System (TOF)**, 2002, CERN/LHCC 2002–016, *https://edms.cern.ch/document/460192/1*

[ALI03] ALICE Collaboration, **Definition of the ALICE Coordinate System and Basic Rules for Sub-detector Components Numbering**, 2003, ALICE-INT-2003-038, *https://edms.cern.ch/document/406391/2*

[ALI04a] ALICE Collaboration, **ALICE: Physics Performance Report – Vol 1**, J. Phys. G: Nucl. Part. Phys. **30** (2004) 1517

[ALI04b] ALICE TRD Collaboration, **First beam test with a real size, six layer, series production detector stack for the ALICE TRD**, GSI Scientific Report 2004

[ALI05a] ALICE TRD Collaboration, **Electron/Pion Identification with ALICE TRD Prototypes using a Neural Network Algorithm**, 2005, arXiv:physics/0506202

[ALI05b] ALICE Collaboration, **ALICE Technical Design Report of the Computing**, 2005, CERN/LHCC-2005-018

[ALI05c] ALICE Collaboration, **ALICE Performance Report Vol. 2**, 2005, to be published

[And03] A. Andronic, P. Braun-Munzinger, K. Redlich and J. Stachel, **Statistical hadronization of charm in heavy-ion collisions at SPS, RHIC and LHC**, Phys. Lett. B **571** (2003) 36

[Bal03] M. Ballintijn, G. Roland, R. Brun and F. Rademakers, **The PROOF distributed parallel analysis framework based on ROOT**, eConf **C0303241** (2003) TUCT004,
General information: *http://root.cern.ch/root/PROOF.html*

[Bed03] M. Bedjidian *et al.*, **Hard probes in heavy ion collisions at the LHC: Heavy flavour physics**, 2003, arXiv:hep-ph/0311048

[Bel04] I. Belikov, P. Hristov, M. Ivanov, T. Kuhr, K. Safarik, **Track reconstruction in high density environment**, Conference proceeding, CHEP 2004

[Bru03] R. Brun, F. Rademakers, P. Canal and M. Goto, **ROOT Status and Future Developments**, eConf **C0303241** (2003) MOJT001

[Bun04] P. Buncic, J. F. Grosse-Oetringhaus, A. J. Peters, P. Saiz, **The Architecture of the AliEn System**, Conference proceeding, CHEP 2004

[Car05] F. Carminati, P. Buncic, P. Hristov, A. Morsch, F. Rademakers, K. Safarik, **The AliRoot framework, status and perspectives**, Conference proceeding, CHEP 2004,
Used software version of AliROOT: HEAD version (10.03.05) with Geant3 v1.0 and ROOT v4.02.00,
General information: *http://aliweb.cern.ch/offline/*

[Chr01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, **Web Services Description Language (WSDL) 1.1**, 2001,
http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[EGE04] EGEE JRA1: Middleware Engineering and Integration, **EGEE Middleware Architecture**, 2004, EGEE-DJRA1.1-476451-v1.0,
https://edms.cern.ch/document/476451/1.0

[EGE05] **Enabling Grids for E-SciencE**, 2005,
General information: *http://public.eu-egee.org/*

[Eid04] S. Eidelmann *et al.*, **Particle Physics Booklet**, Physics Letters B592, 1 (2004)

[Ems05] David Emschermann, Universität Heidelberg, **Personal Communication**, 2005

[Fed05] **The Fedora Project**, 2005,
General information: *http://fedora.redhat.com*

[Fos04a] I. Foster, C. Kesselmann, **The Grid – Blueprint for a New Computing Infrastructure**, 2nd edition, Morgan Kaufmann Publishers, 2004

[Fos04b] Ian Foster *et al.*, **Modeling Stateful Resources with Web Services**, Version 1.1, 2004,
http://devresource.hp.com/drc/specifications/wsrf/ModelingState-1-1.pdf

[Fre96] A. O. Freier *et al.*, **The SSL Protocol**, 1996,
http://wp.netscape.com/eng/ssl3/draft302.txt

[Gud03] M. Gudgin *et al.*, **SOAP Version 1.2**, 2003,
http://www.w3.org/TR/soap/

[Gus04] D. Gustafson, M. Just, M. Nystrom, **RFC 3760 – Securely Available Credentials (SACRED) – Credential Server Framework**, 2004,
http://www.faqs.org/rfcs/rfc3760.html

[Gyu94] M. Gyulassy, X. N. Wang, **HIJING 1.0: A Monte Carlo program for parton and particle production in high-energy hadronic and nuclear collisions**, Comput. Phys. Commun. **83**, 307 (1994)

[Har96] J. W. Harris, B. Müller, **The search for the quark-gluon plasma**, Ann. Rev. Nucl. Part. Sci. **46** (1996) 71

[Hod02] J. Hodges, R. Morgan, **RFC 3377 – Lightweight Directory Access Protocol (v3): Technical Specifications**, 2002,
http://www.faqs.org/rfcs/rfc3377.html

[Hou02] R. Housley, W. Polk, W. Ford, D. Solo, **RFC 3280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile**, 2002,
http://www.faqs.org/rfc/rfc3280.txt

[LCG05] LHC Computing Grid, 2005,
General information: *http://www.cern.ch/lcg*

[LHC03] LHC Computing Grid Project's Requirements and Technical Assessment Group (RTAG), **Architectural Roadmap towards distributed analysis**, CERN, 2003, CERN-LCG-2003-033

[Mah04] T. Mahmoud, **Development of the Readout Chamber of the ALICE Transition Radiation Detector and Evaluation of its Physics Performance in the Quarkonium Sector**, PhD thesis, Universität Heidelberg, 2004

[MON99] The MONARC Architecture Group, **Regional Centres for LHC Computing**, 1999,
http://www.fnal.gov/projects/monarc/task2/rcarchitecture_sty_1.htm

[Mor01] A. Morsch, **Generation of reference events for comparative background studies**, 2001 (last modified: Jan 12th, 2001),
http://aliweb.cern.ch/people/morsch/AliGenerator/GeneratorHIJING.html

[MK05] Mark Mc Keown, **WSRF::Lite – An Implementation of the Web Services Resource Framework**, 2005 (last modified: Jul 19th, 2005), Used Software Version: WSRF::Lite 0.1,
http://www.sve.man.ac.uk/Research/AtoZ/ILCT

[NCS05] **The National Center for Supercomputing Applications**, 2005,
General information: *http://www.ncsa.uiuc.edu/*

[Nor05] **Nordugrid**, 2005,
General information: *http://www.nordugrid.org*

[Nov01] J. Novotny, S. Tuecke, V. Welch, **An Online Credential Repository for the Grid: MyProxy**, Conference proceeding, HPDC-10, 2001

[Pac04] A. Pace, **An Introduction to Public Key Infrastructure (PKI)**, 2004,
http://csc.web.cern.ch/CSC/2004/This_year_school/Programme/Handouts_PDF_Files/grid_intro_pki_updt.pdf

[Per00] D. H. Perkins, **Introduction to High Energy Physics**, 4th edition, Cambride University Press, 2000

[Per05a] **PERL – Practical Extraction and Report Language**, 2005,
General information: *http://www.perl.org/*

[Per05b] H. Pernegger, M. Friedl, **Convoluted Landau and Gaussian Fitting example**, 2005 (last modified: Jul 5th, 2005),
http://root.cern.ch/root/html/examples/langaus.C.html

[Sai03] P. Saiz, L. Aphecetche, P. Buncic, R. Piskac, J.-E. Revsbech, V. Sego, **AliEn – ALICE environment on the GRID**, Nucl. Instrum. Methods **A502** (2003) 437-440

[Sat90] H. Satz, **Color Screening And Quark Deconfinement In Nuclear Collisions**, Adv. Ser. Direct. High Energy Phys. **6** (1990) 593

[SLC05] **Scientific Linux CERN 3 (SLC3)**, 2005,
General information: *http://linux.web.cern.ch/linux/scientific3/*

[Som05a] W. Sommer, Universität Frankfurt, **Personal Communcation**, 2005

[Som05b] W. Sommer, Universität Frankfurt, PhD thesis, 2005, to be published

[Spe99] Standard Performance Evaluation Corporation, **SPEC Releases CPU-2000 Benchmarks**, 1999,
http://www.spec.org/cpu2000/press/release.html, 1999

[TDG00] The DataGrid Project, **Technical Annex**, 2000,
General information: *http://www.edg.org*

[Tue04] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, **Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile**, 2004,
http://www.faqs.org/rfc/rfc3820.txt

[VDT05] **The Virtual Data Toolkit**, 2005,
General information: *http://www.cs.wisc.edu/vdt*

[WSR04] **The WS-Resource Framework**, 2004,
General information: *http://www.globus.org/wsrf*

[Wil04] A. Wilk, **Elektronen-Pionen-Separation im ALICE TRD**, Diploma thesis, Universität Münster, 2004

[Won94] C.-Y. Wong, **Introduction to High-Energy Heavy-Ion Collisions**, World Scientific Publishing Co. Pte. Ltd, 1994

F. List of Figures and Tables

List of Figures

List of Tables

Acknowledgements

At this point I would like to thank the people who made this thesis possible and who supported me in many ways. Especially:
Johannes P. Wessels for an interesting topic in a nice working group and collaboration environment, for the support of my CERN applications and the possibility to work with a Grid computing cluster.
Predrag Buncic and *Federico Carminati* for supporting my CERN applications and the possibility to work for the AliEn team.
Roberta Faggian Marque for the nice supervision during my technical student stay at CERN, and *Pedro Andrade*, *Paolo Palazzi* and the GRACE collaboration for the friendly atmosphere.
Karen Koop and *Manfred Reinecker* for their linguistic expertise and the provided corrections.
Predrag Buncic, *Roberta Faggian Marque*, *Michael Grevenstette*, *Christian Klein-Bösing*, *Andreas Köpf*, *Mareike Rosin*, *Baldo Sahlmüller*, *Pablo Saiz*, *Wolfgang Sommer* and *Alexander Wilk* for giving important feedback.
Peter Hristov, *Marian Ivanov*, *Andreas Joachim Peters* and *Pablo Saiz* for answering many Grid, AliEn and AliROOT questions.
The TRD collaboration, especially *Anton Andronic*, *David Emschermann*, *Prashant Shukla* and *Wolfgang Sommer*, for answering many TRD-related questions.
Michael Grevenstette, *Christian Klein-Bösing*, *Klaus Reygers* and *Alexander Wilk* for answering even more of the many questions.
The PROVISIO GmbH, *Andreas Köpf* and *Christoph Niehus* in particular, for giving me the possibility to work at their company and their ability of making me solve unsolvable tasks.
Damian Bucher for the good teamwork during the installation of the cluster.
For the nice atmosphere in my group, I would like to thank *Jan Auffenberg*, *Christoph Baumann*, *Damian Bucher*, *Richard Glasow*, *Holger Gottschlag*, *Norbert Heine*, *Melanie Hoppe*, *Christian Klein-Bösing*, *Karen Koop*, *Markus Rammler*, *Klaus Reygers*, *Baldo Sahlmüller*, *Wolfgang Verhoeven*, *Johannes P. Wessels*, *Alexander Wilk* and *Oliver Zaudtke*.
Alexander Ferling and *Oliver Winkelmann* for making the physics understandable.
Matthias Böcker, *Michael Grevenstette*, *Uwe Hehmann*, *Katrin Horstmann*, *Christian Ripperda*, *Tobias Heil* and *Alexander Winnemöller* for fighting the physics together.
My parents, *Cornelia* and *Hans-Joachim*, for their support which made my studies possible.
And *Mareike* for all her support.

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Münster, 1. August 2005

Jan Fiete Grosse-Oetringhaus