Design and Implementation of the Object-Oriented Fast Simulation Program for the ATLAS Experiment and its Use to Determine the Discovery Potential of the Higgs Boson via the Channel $\mathrm{h}_0 \rightarrow \mathrm{Z}^0\mathrm{Z}^{0*} \rightarrow b\bar{b}\ l^+l^-$

Richard M. Steward

University College London

Submitted to the University of London

in fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy.

April 2004.

Contents

List of Figures

List of Tables

Abstract

The design and implementation of the object-oriented fast simulation program Atlfast is described for the ATLAS experiment at the CERN particle physics laboratory in Switzerland.

Fast simulations use parametrised energy and momentum smearing in order to recreate the detection efficiency and particle identification of a real experimental detector, without the time-consuming computation required for full detector simulation.

Additionally, an object-oriented program for performing user-defined physics analyses is described. This program is released for general use by the ATLAS collaboration and is designed for use with, but not restricted to, physics output from the Atlfast fast simulation program.

These programs are demonstrated in a physics study of the feasibility of discovering the Higgs boson at the ATLAS experiment, using the discovery channel $\mathrm{h}_0 \rightarrow \mathrm{Z}^0\mathrm{Z}^{0*} \rightarrow b\bar{b}\ l^+l^-$ via weak vector boson fusion in the mass range 150 GeV - 200 GeV. It is found that this channel does not significantly increase the discovery potential over this mass range, achieving the observation threshold of 3σ only after 3 years high luminosity running at the upper end of the mass range.

Acknowledgements

This thesis is dedicated to my daughter, Milly.

This work would not have been possible without the guidance and tuition of many people in the UCL HEP group. I would like especially to thank Peter Clarke, Peter Sherwood, and Jon Couchman for their patience, advice and encouragement. I am grateful to PPARC for their generous sponsorship which made possible a memorable experience working at the CERN laboratory.

There are also a number of people whose friendships and support have vastly improved the quality of my life throughout. My fondest thoughts go firstly to Isabel for her constant faith and companionship, and for making my time in Geneva richer and more memorable than I could have hoped for; Matt and Adam for being the best neighbours both in Switzerland and thereafter; all the London contingent, but most notably Leigh, Tony and Paul for not forgetting me while I was away; Simon, Nick and Richard for sharing the powder, and Pierre for being there in spirit. I would also like to thank my family for their love and support in all that I do.

Chapter 1

Introduction

1.1 The Standard Model

The Standard Model[1] is the best theory in existence for describing the phenomenology of the electromagnetic, weak and strong forces. It is a gauge quantum field theory based on the SU(3)xSU(2)xU(1) symmetry group, describing both the spin-$\frac{1}{2}$ fermion fields which constitute fundamental matter, and the integer spin gauge bosons which mediate the fundamental forces.

The matter fermions can be divided into three 'generations' of progressively more massive families, consisting of lepton and quark doublets. These particles are shown in figure 1.1. For each fermion there exists an antiparticle partner which has identical mass and opposite quantum numbers.

$$\text{Leptons} \quad \begin{pmatrix} e \\ \nu_e \end{pmatrix} \quad \begin{pmatrix} \mu \\ \nu_\mu \end{pmatrix} \quad \begin{pmatrix} \tau \\ \nu_\tau \end{pmatrix}$$

$$\text{Quarks} \quad \begin{pmatrix} u \\ d \end{pmatrix} \quad \begin{pmatrix} c \\ s \end{pmatrix} \quad \begin{pmatrix} t \\ b \end{pmatrix}$$

Figure 1.1: The Standard Model Fermion Families

The leptons, such as the electron, exist in nature as single free particles, while the quarks are bound within tri-quark hadrons such as the proton(uud), or quark-antiquark mesons such as the pion ($\overline{u}d$).

The gauge bosons are grouped in table 1.1 according to the interactions which they mediate. The electromagnetic interaction is described by the U(1) gauge symmetric theory of quantum electrodynamics (QED). It affects all charged particles and is mediated by the photon.

The electromagnetic and the weak interaction are unified into the SU(2)xU(1) 'electroweak' model which describes the charged $W^{\pm}$ bosons and the neutral Z^0 boson, as well as the photon. An example of a weak interaction is shown in figure 1.2.

The helicity of a particle is the component of spin in the direction of its motion. Fermions have spin-$\frac{1}{2}$ so can have helicity $\pm\frac{1}{2}$; fermions with negative helicity are called left-handed. The weak interaction couples to all left-handed fermions.

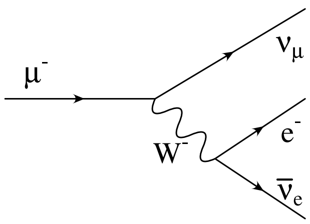


Figure 1.2: *A Typical Leptonic Weak Interaction*

The strong force is described by quantum chromodynamics (QCD) based on the SU(3) symmetry group. It is felt only by the quark fermions and is responsible for binding the quarks in $q\overline{q}$ meson and qqq hadron states. It is mediated by eight self-interacting tri-coloured gluons.

The nature of the gluon self-interaction causes the strength of the strong force to increase with distance. This phenomenon is called 'asymptotic freedom'. As coloured objects are separated, the potential energy increases until new quarks are formed which combine to preserve overall colour neutrality. This is why single quarks are never observed and is known as 'quark confinement'.

Interaction	Boson	Mass (GeV)
Electromagnetic	γ	0
Weak	$W\pm$	80.41 ± 0.10
	Z^0	91.187 ± 0.007
Strong	8 gluons	0

Table 1.1: *The Standard Model Bosons*

The origin of the particle masses in the Standard Model is unconfirmed. According to the quantum field theory, all particles in the Lagrangian must be massless in order to preserve local gauge invariance. A solution to this problem is provided by the Higgs Mechanism, which allows particles to acquire masses while preserving invariance of the Lagrangian. This mechanism gives rise to an additional spinless scalar particle called the Higgs boson. It is the discovery of this particle, and therefore the origins of the particle masses, which is partly the subject of this thesis.

1.2 The Higgs Mechanism

In a gauge symmetric quantum field theory, such as the SU(2) group of rotations in weak isospin space, the Lagrangian which describes the system is required to be invariant under local (space-time dependant) transformations of the wavefunction.

When this 'local gauge invariance' is imposed on a Lagrangian with fermionic fields, a number(dependant on the number of generators for that group) of gauge fields A^i_μ are required to maintain this gauge symmetry.

Adding mass-like terms for these gauge vector fields violates the symmetry, so they cannot therefore be directly related to the massive $W^\pm$ and Z^0 vector bosons of the weak interaction.

The mechanism of 'spontaneous symmetry breaking'[2] can be used to generate gauge boson mass terms in the Lagrangian while maintaining symmetry under local gauge transformations. This mechanism is introduced using an example of spontaneous breaking of a U(1) local gauge symmetry.

A complex scalar field $\phi = (\phi_1 + i\phi_2)/\sqrt{2}$ is described by the Lagrangian 1.1, which is invariant under the U(1) local gauge transformation $\phi \rightarrow e^{i\alpha(x)}\phi$.

$$\mathcal{L} = (D_\mu\phi)^*(D_\mu\phi) - \mu^2\phi^*\phi - \lambda(\phi^*\phi)^2 - \frac{1}{4}F_{\mu\nu}F^{\mu\nu}$$
$$\text{where } D_\mu = \partial_\mu - ieA_\mu \tag{1.1}$$

Considering the case where $\lambda > 0$ and $\mu^2 < 0$, the potential, shown in figure 1.3, has minima which are degenerate about a circle in the ϕ_1, ϕ_2 plane of radius ν, such that

$$\phi_1^2 + \phi_2^2 = \nu^2 \quad \text{with} \quad \nu^2 = -\frac{\mu^2}{\lambda} \tag{1.2}$$

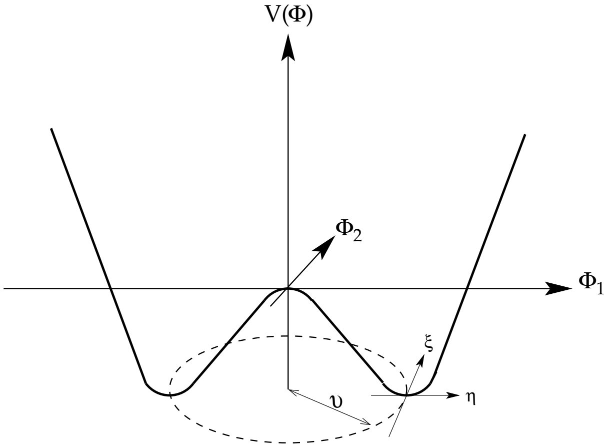


Figure 1.3: *A complex scalar field potential where* $\mu^2 < 0$

Translating the field ϕ to a potential minimum at $\phi_1 = \nu$, $\phi_2 = 0$, the Lagrangian can be expanded about the vacuum in terms of fields η, ξ; excitations in the ϕ_1 and ϕ_2 directions. Substituting the expansion 1.3, the Lagrangian becomes 1.4[3].

$$\phi(x) = \sqrt{\tfrac{1}{2}}[\nu + \eta(x) + i\xi(x)] \tag{1.3}$$

$$\begin{aligned} \mathcal{L}' = \frac{1}{2}(\partial_\mu \xi)^2 + \frac{1}{2}(\partial_\mu \eta)^2 - \nu^2 \lambda \eta^2 + \frac{1}{2}e^2\nu^2 A_\mu A^\mu \\ -e\nu A_\mu \partial^\mu \xi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \textit{interaction terms} \end{aligned} \tag{1.4}$$

The Lagrangian contains terms representing the desired massive vector boson A_μ, a massive scalar η, but also a massless boson ξ, known as a Goldstone boson. This Goldstone boson is shown to be unphysical by using relation 1.5 and choosing a gauge where the h field in 1.6 is real. Substituting this into the original Lagrangian of equation 1.1 one arrives at 1.7.

$$\phi = \sqrt{\tfrac{1}{2}}(\nu + \eta + i\xi) \approx \sqrt{\tfrac{1}{2}}(\nu + \eta)e^{i\xi/\nu} \tag{1.5}$$

$$\phi \to \sqrt{\tfrac{1}{2}}(\nu + h(x))e^{i\theta(x)/\nu} \tag{1.6}$$

$$\mathcal{L}'' = \frac{1}{2}(\partial_\mu h)^2 - \lambda\nu^2 h^2 + \frac{1}{2}e^2\nu^2 A_\mu^2 - \lambda\nu h^3 - \frac{1}{4}\lambda h^4 \\ + \frac{1}{2}e^2 A_\mu^2 h^2 + \nu e^2 A_\mu^2 h - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} \tag{1.7}$$

This Lagrangian contains only two massive particles, the vector boson A_μ with $m_A = e\nu$, and the massive scalar h with $m_h = \sqrt{2\lambda\nu^2}$, called a Higgs particle. The Goldstone boson is no longer apparent, providing the longitudinal polarization state of the massive gauge boson.

This procedure for eliminating the unwanted massless Goldstone bosons is called the Higgs Mechanism, and is extended to the SU(2) local gauge symmetry using a doublet of complex scalar fields, shown in 1.8. Similarly, choosing the potential minimum to be at $(\phi_1 + i\phi_2) = 0, (\phi_3 + i\phi_4) = \nu$ in equation 1.9 and substituting expansion 1.10 into the SU(2) invariant Lagrangian containing three W_μ gauge bosons, one is left with three massive gauge fields with $m = \frac{1}{2}g\nu$ where g is the gauge coupling, and one Higgs scalar with $m = \sqrt{2}\mu$.

$$\phi = \begin{pmatrix} \phi_\alpha \\ \phi_\beta \end{pmatrix} = \sqrt{\tfrac{1}{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix} \tag{1.8}$$

$$\phi = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ \nu \end{pmatrix} \tag{1.9}$$

$$\Phi = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ \nu + h(x) \end{pmatrix} \tag{1.10}$$

The boson masses of the electroweak interaction are obtained by using the same complex Higgs doublet to break the symmetry of an SU(2)xU(1) invariant Lagrangian. The SU(2) group corresponds to the non-Abelian group describing rotations in isospin space, while the U(1) group represents weak hypercharge transformations. Hypercharge Y is related to the electromagnetic charge Q by equation 1.11, where T^3 is the third component of weak isospin.

$$Q = T^3 + Y \tag{1.11}$$

The U(2) group with coupling g and gauge bosons W_μ^a is related to the U(1) group with coupling g' and boson B_μ by the weak mixing angle, θ_W, which determines the relative two strengths of the interactions according to 1.12.

$$g^{'} = gtan\theta_W \tag{1.12}$$

The physical electroweak bosons, W, Z, and γ are linear superpositions of these gauge fields as shown in 1.13, of which only the Z and W acquire mass through the Higgs mechanism. The photon is massless because the Lagrangian remains invariant under the local $U(1)_{em}$ transformations with generator Q. The mass of these bosons are related as in 1.14, where the parameter describing the relative strengths of the neutral and charged current processes $\rho \approx 1$.

$$\begin{aligned} A_\mu &= cos\theta_W B_\mu + sin\theta_W W_\mu^0 \\ Z_\mu &= cos\theta_W W_\mu^0 - sin\theta_W B_\mu \\ W_\mu^\pm &= \frac{1}{\sqrt{2}}(W_\mu^1 \pm iW_\mu^2) \end{aligned} \tag{1.13}$$

$$M_W^2 = \rho M_Z^2 cos^2\theta_W \tag{1.14}$$

The minimal choice of a Higgs doublet is sufficient to generate the masses of the weak bosons as well as all the fermion masses in the Standard Model. These masses however are just parameters of the theory and are not predicted. The mass of the Higgs boson itself is also not predicted, however the fact that the fermionic coupling is proportional to the fermion mass is a prediction which could be tested if a Standard Model Higgs boson is discovered.

Chapter 2

The LHC and the ATLAS detector

This chapter gives an overview of the Large Hadron Collider and the ATLAS detector, including the layout of the accelerator and its specifications, and an explanation of the various subdetectors comprising the experiment.

2.1 The Large Hadron Collider

The Large Hadron Collider[4] is a proton-proton synchroton particle accelerator in construction at the CERN particle physics laboratory in Geneva, Switzerland. It will be the largest and most energetic synchrotron in the world, providing centre of mass energies of up to 14 TeV. Physics studies over a mass range up to 1 TeV will be possible, providing many opportunities to study standard model predictions for new physics, especially in the electroweak symmetry breaking sector.

The LHC will utilise the existing 27km LEP tunnel within which two opposing beams of protons or heavy ions will be collided[5].

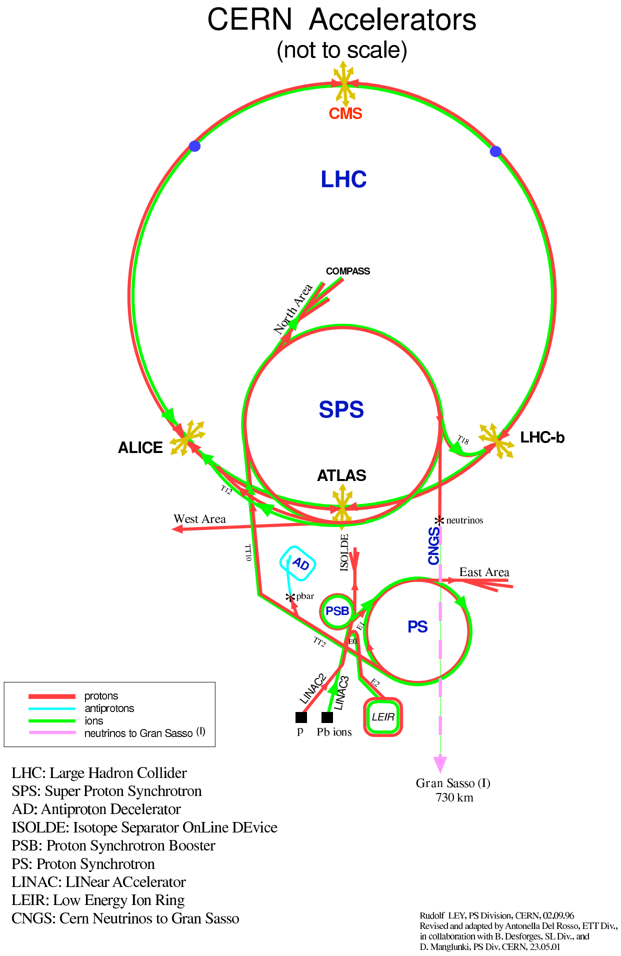


Figure 2.1: *The LHC injection layout*

The protons are accelerated in four stages, shown in figure 2.1, commencing with the Proton linac. Here, protons are accelerated to 50 MeV and injected into the Proton Synchroton Booster(PSB) where they reach 1.4 GeV. The Proton Synchroton(PS) then accelerates the protons to 26 GeV where they are delivered in 135 bunches, spaced at 25ns, to the Super Proton Synchroton(SPS). After being accelerated to 450 GeV, they are injected in to the LHC. Protons are accelerated to 7 TeV in the main collider, completing 400 million revolutions in the luminosity lifetime of 10 hours.

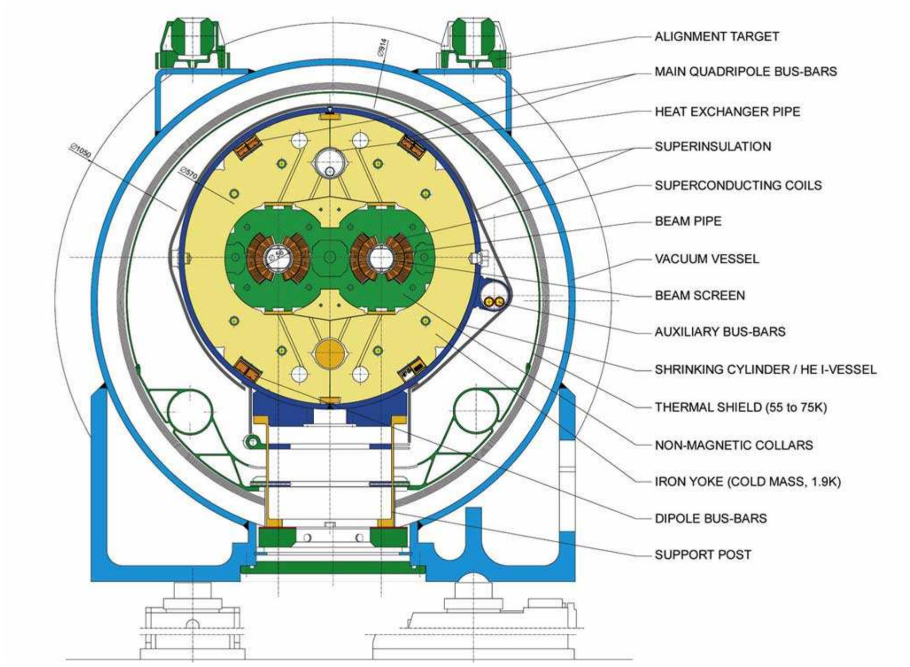


Figure 2.2: *Cross section of the LHC beam-pipe superconducting magnets*

The LHC consists of single twin bore superconducting magnets which provide fields of up to 8.65 T. Coupled with the radius of the accelerator beam, a maximum proton centre of mass energy of 14TeV can be reached. Figure 2.2 shows a cross-section of the beam pipe. The twin-apertures lie side by side, 194mm apart, in the cold yokes of the main dipole and quadrupole magnets which are cooled to 1.9K with superfluid Helium.

Although the energies reached by the LHC allow investigation of physics at higher mass ranges, the cross-sections of many processes are inversely proportional to the square of the beam energy[1]. Therefore, to achieve the high production cross-sections necessary for the study of interesting physics at the LHC, a very high luminosity is required. Initially, the LHC will run

[1] $\sigma(e^+e^- \to \mu^+\mu^-) = \frac{4\pi\alpha^2}{3s}$ where $\alpha = e^2/4\pi$ and $\sqrt{s}$ is the beam energy

with a luminosity of $2 \times 10^{33} cm^{-2}s^{-1}$, increasing to $10^{34} cm^{-2}s^{-1}$ as soon as possible. The lower luminosity running is more suited to B-physics studies due to smaller pile up effects, where large numbers of particle collisions make vertex detection of B-decays near the interaction point more difficult.

The basic design parameters of the LHC are summarised in table 2.1[5].

Parameters		p-p	pb
Beam energy	TeV	7.0	7
Centre of mass energy	TeV	14	1262
Injection energy	GeV	450	190.6
Initial luminosity	$cm^{-2}s^{-1}$	2×10^{33}	$1.95X10^{27}$
Luminosity	$cm^{-2}s^{-1}$	10^{34}	$1.8X10^{27}$
Bunch spacing	ns	25	124.75
Particles per bunch		$1X10^{11}$	$6.2X10^{7}$
R.M.S bunch length	m	0.075	0.075
Number of bunches		2835	608
Luminosity lifetime	h	10	10
Dipole field	T	8.3	8.3
Coil aperture	mm	56	56
Distance between apertures	mm	194	194

Table 2.1: *Parameters of the LHC*

There will be four experiments on the LHC ring. ATLAS[6] and CMS[7] are multi-purpose detectors which will be used to study a wide range of physics processes. ALICE[8] will study heavy ion collisions and LHCb[9] will specialise in B physics. They are located on the LHC ring as shown in figure 2.1

2.2 The ATLAS Detector

The ATLAS detector is an all-purpose pp detector which is designed to exploit the full discovery potential of the Large Hadron Collider. A three dimensional view of the ATLAS detector is shown in figure 2.3. It has a total length of 42m, a radius of 11m, weighs 7000 tonnes and comprises four major detector sub-systems; the inner detector, the electromagnetic and hadronic calorimeters, the muon spectrometer and the magnet systems.

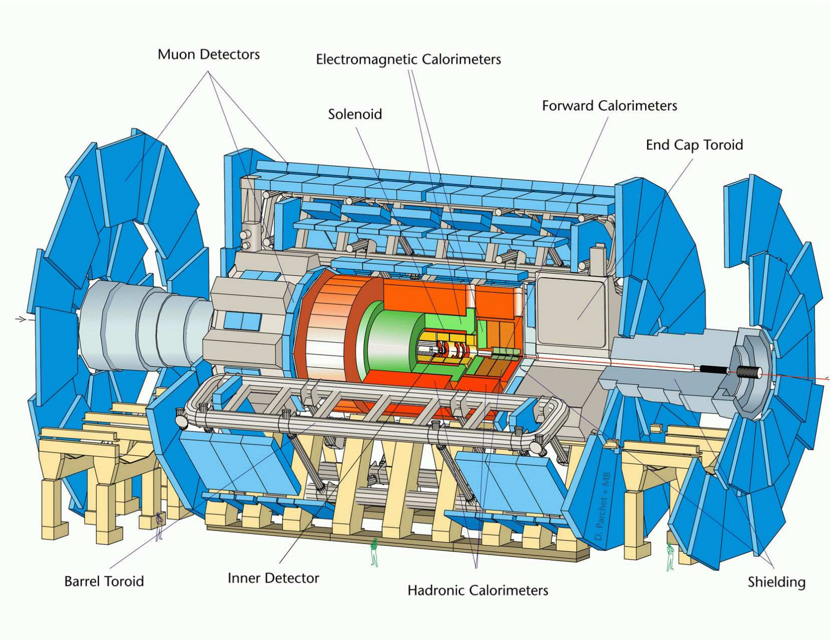


Figure 2.3: *The ATLAS Detector*

One of the main goals for the LHC is to investigate the nature of Electroweak Symmetry Breaking in the Standard Model. Searches for the SM Higgs Boson, or a family of Higgs particles as described by the Minimal Supersymmetric Standard Model, are therefore used as a first benchmark for detector optimisation.

Research into other important phenomena such as fermionic composites, CP violation in B-decays, and precision top quark measurements also require robust detector performance from all of these detector subsystems. The basic design requirements are summarized as follows[10]:

- Very good electromagnetic calorimetry for electron and photon identification and measurement.
- Hermetic jet and missing E_T calorimetry.
- Efficient tracking at
  - high luminosity for lepton momentum measurements, b-quark tagging, and enhanced electron and photon identification.
  - low luminosity for tau and heavy-flavour vertexing and reconstruction capability of B-decay final states.
- Stand alone, precision muon-momentum measurements up to high luminosity, as well as very low p_T trigger capability at lower luminosity.

Rigorous requirements are placed on the ATLAS trigger systems by the high interaction rate of 10^9 Hz at design luminosity. The bunch crossing rate of 40 MHz requires decisions to be made every 25ns, while small cross sections for interesting physics processes require that high rates of background rejection are achieved.

The trigger system comprises three levels, each applying additional selection criteria. No tracking information is used in level 1, with decisions based upon local and global trigger objects exceeding various thresholds, while level 2 triggering makes use of calorimetry information to identify regions of interest. Complete reconstruction is performed by the event filter, using more complex algorithms to reject background. The final data rate is expected to be 100 MB per second.

2.2.1 The Inner Detector

The inner detector[11] provides high-precision vertex and momentum measurements of charged particles from the LHC beam-pipe to the electromagnetic calorimeter system. It comprises three distinct subsystems; high-resolution pixel detectors at innermost radii, fine-granularity silicon microstrips in the intermediate Semi-Conductor Tracker(SCT), and straw tube technology in the outermost Transition Radiation Tracker(TRT). These subsystems are contained within a 2T solenoidal magnet.

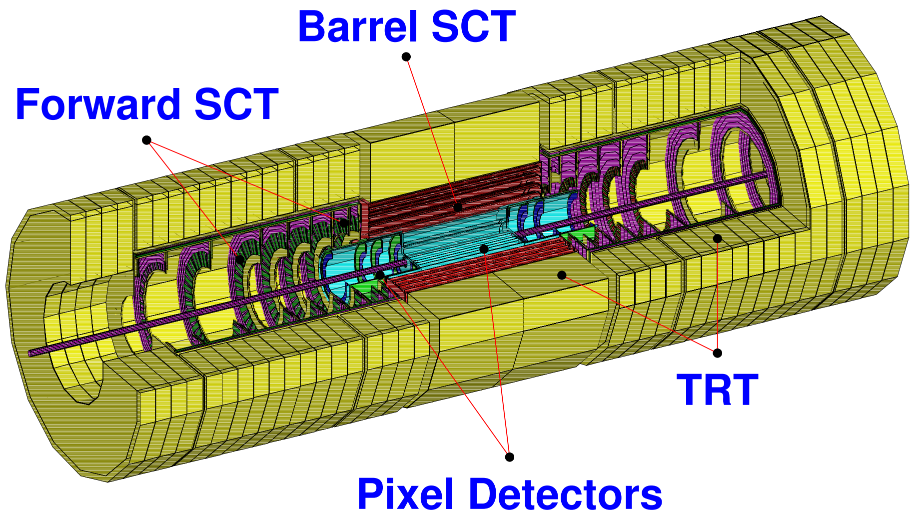


Figure 2.4: *The ATLAS Inner Detector*

A three-dimensional view of the inner detector is shown in figure 2.4. It is 7m long with a radius of 115 cm. The barrel part of the detector extends over ± 80 cm along the beam axis, with two end-caps covering the rest of the cylindrical cavity. The central pixel and SCT tracking elements are contained within a radius of 56 cm.

The barrel detector layers are arranged on concentric cylinders around the beam axis, covering the region $|\eta| < 1.4$ where η is the pseudorapidity defined as $\eta = -ln[tan\frac{\theta}{2}]$. The end cap detectors are mounted on disks perpendicular to the beam axis, with the tracking elements located in planes parallel to the disks.

The Pixel Detector

The pixel detector provides very high-granularity, high-precision measurements as close as possible to the interaction point, determining the impact parameter resolution and the ability of the inner detector to identify b-quarks, τ leptons, and other such short-lived particles.

The system consists of three barrels at average radii of 4cm, 11cm, and 14cm, and four disks in each end-cap between radii of 11cm and 20cm. It consists of 140 million detector elements with granularities of 50 μm in the $R\phi$ direction and 300μm in z, making the pixel detector highly effective at track separation.

The Semiconductor Tracker

The SCT system provides four precision measurements per track in the intermediate radial range, contributing to the measurement of momentum, impact parameter, and vertex position.

The barrel uses four layers of silicon microstrip detector at radii 300mm, 373mm, 447mm, and 520mm to provide points in the $r-\phi$ and z co-ordinates within the pseudorapidity range $|\eta| < 2.5$. The end-caps consist of modules mounted in rings onto nine wheels, with a barrel/end-cap transition region lying within $1.16 < \eta < 1.64$.

The detector contains $61m^2$ of silicon detectors, with 6.2 million readout channels. It gives a spatial resolution of 16 μm in $R\phi$ and 580 μm in z. Tracks separated by more than 200μm can be distinguished with this detector arrangement.

The Transition Radiation Tracker

The TRT uses straw detectors which, by virtue of their small diameter and isolation, can operate at the very high rates needed at the LHC. They typically provide 36 measurements per track in the barrel region, with a precision of 170 μm in $R\phi$, but with a drift time of up to 40ns; significantly larger than the LHC bunch crossing period.

Each straw is 4mm in diameter and 150cm long, giving a fast response and good mechanical properties. The barrel contains about 50,000 straws divided in two at the centre to reduce occupancy and readout at each end. The straws run parallel to the beam-pipe and are 0.68cm apart, covering the radial range from 56cm to 107cm. The first six radial layers are inactive over the central 80cm of their length, providing extra coverage of the crack between barrel and end-caps.

The two end-caps consist of 18 wheels. The straws are oriented radially with between 576 and 768 straws per layer giving a straw spacing of 1.1 cm at the outer radius. The most central 14 wheels cover the radial range from 64cm to 103cm, while the last four extend to an inner radius of 48cm.

The TRT contributes to the accuracy of the momentum measurement in the inner detector by providing a set of measurements equivalent to a single point of 50 μm precision. Pattern recognition is aided by the large number of hits per track, and should allow a fast level-2 track trigger to be implemented[12]. Additionally, Xenon gas in the TRT aids electron identification through the detection of transition radiation photons created in the radiator between the straws.

2.2.2 Calorimetry

The physics requirements of the calorimeter system includes accurate measurement of energy and position of electrons and photons, as well as the energy and direction of high energy hadronic jets. Calorimeters also aid particle identification, separation of electrons, photons and hadronic τ decays, missing transverse momentum measurement, and event triggering.

The ATLAS calorimetry[13] consists at inner radii of an electromagnetic calorimeter made of lead-liquid argon (LAr) providing coverage up to $|\eta| < 3.9$, and an outer dual-technology hadronic calorimeter covering up to $|\eta| < 4.9$. The barrel region uses iron scintillator and steel absorber tiles, while the end-caps use LAr technology. A three dimensional view of the ATLAS calorimetry is shown in figure 2.5.

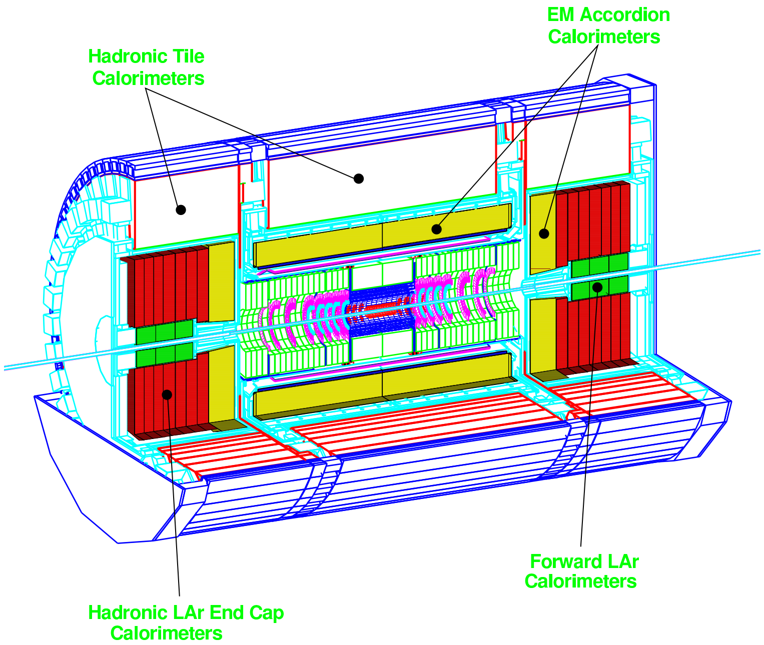


Figure 2.5: *A three dimensional view of the ATLAS calorimetry*

The Electromagnetic Calorimeter

Higgs benchmark channels(see section 6.1) place stringent requirements on acceptance, energy resolution, energy range and particle identification. The 'golden' Higgs channel, $\mathrm{h}_0 \rightarrow \mathrm{Z}^0\mathrm{Z}^{0*} \rightarrow 4e$, covering the mass range 90-180 GeV requires electron identification down to 5 GeV with energy resolutions of 1%, while certain heavy boson channels require lepton reconstruction up to the TeV scale.

In order to meet these requirements, the electromagnetic calorimeter uses LAr active scintillator plates with stainless-steel coated absorber material, arranged in an accordian shape, providing complete symmetry in ϕ with no azimuthal gaps. The barrel covers the pseudorapidity range $|\eta| < 1.475$ with two identical end-caps covering $1.375 > |\eta| < 3.2$. The thickness of the absorbers varies as a function of pseudorapidity, optimising the energy resolution of the calorimeter.

The Hadronic Calorimeter

The purpose of the hadronic calorimeter is to identify, reconstruct, and measure jets caused by hadronic activity in the detector, as well as contributing to the missing energy measurement.

Standard Model Higgs channels require high performance in both high and low mass regions. For low mass searches, reconstruction of b jet pairs from $h \rightarrow b\bar{b}$ decays are important, while for higher mass Higgs decays via high-p_T $W \rightarrow jj$, good jet-jet mass reconstruction as well as accurate forward jet tagging is required. MSSM Higgs decays to $\tau\tau$ pairs require high resolution in $p_T^{miss}$and accurate reconstruction of E_T^{miss}.

The calorimetry uses a combination of two technologies, collectively covering a pseudorapidity range of $|\eta| < 3.2$, from 2.3m to 4.23m in the radial direction. The sampling calorimeters are constructed from iron scintillator tiles in an extended barrel region of $|\eta| < 1.7$. In the range $1.5 < |\eta| < 3.2$ liquid argon calorimetry is employed, with higher density LAr calorimetry in the remaining $3.2 < |\eta| < 4.9$.

2.2.3 The Muon Spectrometer

The design of the muon spectrometer is influenced by SM and MSSM Higgs decays as well as new vector boson decays with muons in the final state. Its requirements are summarised as follows[14]

- Identify, reconstruct and measure the momenta of muon tracks, associating them with measurements from the inner detector. It must cover the pseudorapidity range of $|\eta| < 3$ and maintain good transverse momentum resolution down to 5 GeV.
- Unambiguously associate muons with the correct bunch crossing using timing information provided by the trigger chambers.
- Selectively trigger on both single and multi-muon events, with a wide acceptance.

A wide pseudorapidity coverage of up to $|\eta| = 3$ is essential for track reconstruction, while momentum and mass resolutions of order 1% are crucial for the reconstruction of narrow final states with two or four muons as well as background rejection and charge identification.

The muon system also plays an important role in triggering, where the trigger pseudorapidity coverage of $|\eta| < 2.4$ is determined by good acceptance for Higgs processes and the high statistics needed to study asymmetries in CP violating beauty decays.

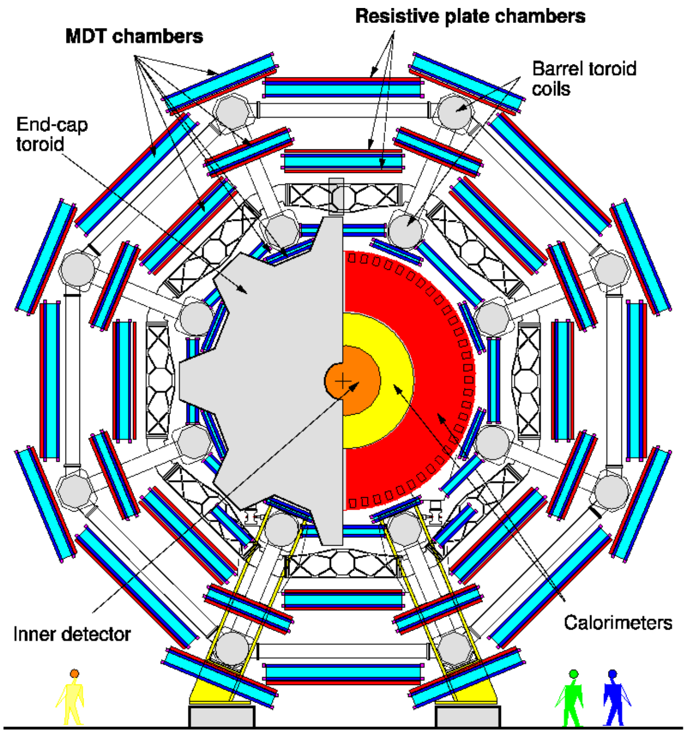


Figure 2.6: *End view of the muon spectrometer*

The system consists of three large super conducting air core toroid magnets, providing a large volume field covering a psuedorapidity range of $|\eta| < 2.7$, as well as a fast trigger system and high precision tracking chambers. Side and end view schematics of the muon chambers are shown in figures 2.6 and 2.7.

The chambers are arranged such that particles from the interaction traverse three chambers. In the barrel, they are arranged on three concentric cylinders about the beam axis at 5m, 7.5m, and 10m radii. In the end caps they are arranged over four disks at distances 7m, 10m, 14mm, and 22m from the interaction point.

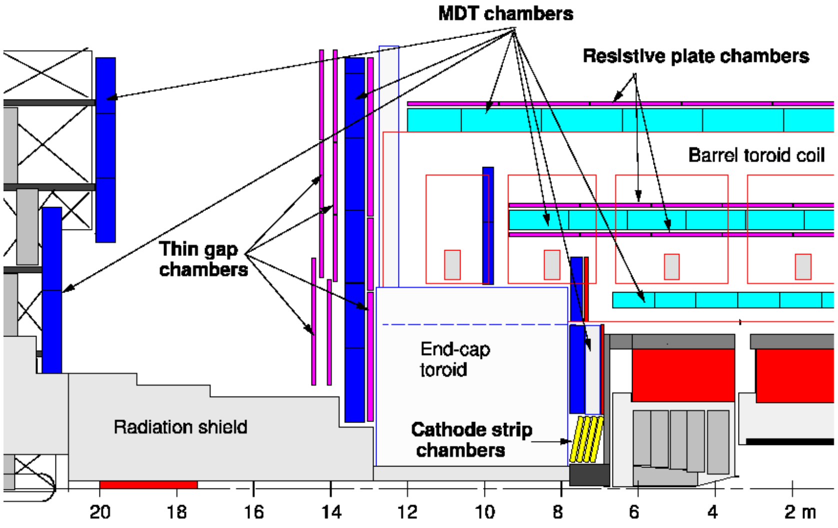


Figure 2.7: *Side view of the muon spectrometer*

2.2.4 The Magnet System

The ATLAS magnet systems consist of the central solenoid and the outer air-core toroids. The 5.3m central solenoid provides the axial 2T magnetic field in the inner detector volume, and is positioned in front of the electromagnetic calorimeter. To avoid degradation of the calorimeter performance, the single layer superconducting coil is integrated into the vaccuum vessel of the calorimeter barrel cryostat.

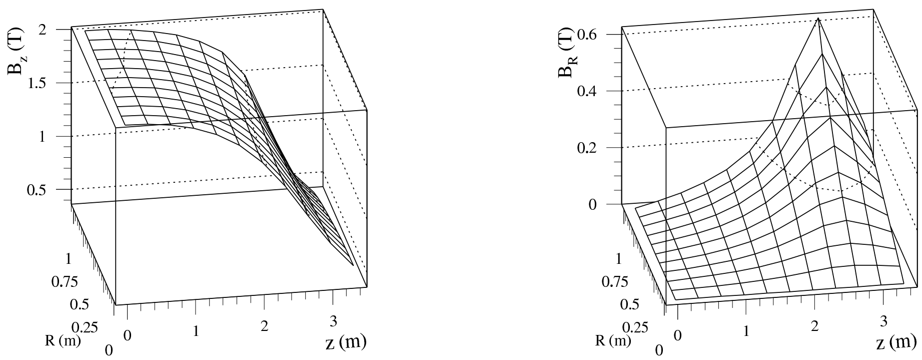


Figure 2.8: *Solenoidal Axial and Radial Field Strengths*

The iron absorber of the tile calorimeter provides the yoke for the magnetic flux return, which is separate from the solenoid coil itself, resulting in a non-uniform field in the inner detector. This causes deviation of track trajectories from perfect helices, impacting pattern recognition algorithms and reduced bending power resulting in degradation of p_T resolutions. The deviations in field uniformity are shown in figure 2.8.

The superconducting air-core toroid magnet system consists of a 26m long barrel consisting of eight flat coils arranged about the beam axis and two 5.6m end caps inserted at each end of the barrel. This arrangement, shown in figure 2.9 provides a pseudorapidity coverage of $|\eta| < 3$ and an integrated field strength increasing from 3Tm at $\eta = 0$ to 8Tm at $\eta = 3$. The whole magnet system represents a cold mass of 700 tons, with a total weight of 1400 tons.

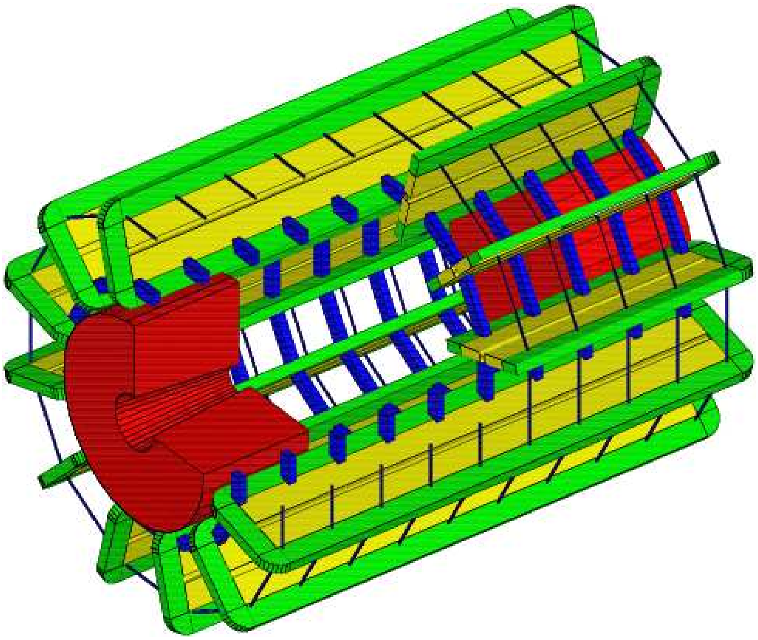


Figure 2.9: *The Superconducting Air-Core Toroid Magnet System*

Chapter 3

Design of the Atlfast Fast Simulation Package

This chapter describes the design and functionality of the Athena-Atlfast fast simulation package. It was written in C++ within the Athena software framework by a small team of which the author was a member.

3.1 The Athena Software Framework

Athena is the software control framework used by the ATLAS collaboration and is based on the Gaudi software architecture, with ATLAS specific enhancements[15]. A software architecture describes software components and the interactions which are possible between these components.

An architectural framework provides services and defines the interactions, or interfaces, between software components. These components can then be extended to make useful software which 'plugs' in to the architecture.

Software developed by physicists consists of specialisations of a few specific components. For example, to develop an application such as a fast simulator one develops software by adding functionality to a number of framework application base classes.

3.1.1 Application Component Base Classes

In order to develop a software application within Athena, there are three base classes which should be extended.

- DataObject
- ContainedObject
- Algorithm

Algorithms

The Algorithm is a base class intended for computational, 'function' type objects. It provides three methods which are called by the framework, and must be overloaded by derived algorithm classes.

$$\mathsf{StatusCode\ initialize()} \tag{3.1}$$

$$\mathsf{StatusCode\ execute()} \tag{3.2}$$

$$\mathsf{StatusCode\ finalize()} \tag{3.3}$$

Algorithms are scheduled by the architecture, and also instantiated by it. State variables can therefore not be set in the constructor, so instance specific parameters are set using a steering 'job options' file which is interpreted at run-time by Athena. Algorithms also have access to the Athena services, via service interfaces. Any classes which inheret from the Algorithm class can also access these services.

Method 3.1 is used to carry out initialising procedures, such as instantiating member variables, and is called by Athena once at the beginning of a job. Method 3.2 is called recursively during the running of the job and carries out the function of the extended Algorithm. The number of executed 'events' is set by a job options parameter. Method 3.3 is called at the end of the job and generally is used to execute any final tasks and cleaning up required by the algorithm.

Data and Contained Objects

Data objects and Contained objects represent physical quantities, such as momenta, particles or tracks. All objects which are intended for storage individually in Athena's 'Transient Data Store' should inheret from the Data Object class. Athena also provides container class ObjectVectors for storing groups of objects. These types of data classes should inheret from the ContainedObject class.

3.1.2 The Athena Services

Athena provides a number of services which are useful to software developers who are creating programs within the control framework. The main services control algorithm configuration, data storage, persistency and messaging.

Job Options Service

As mentioned above, Algorithms cannot be configured via their constructors because the Algorithm classes are scheduled and instantiated by the framework. Algorithm data members of type int, float, string, or vectors of these types, can be declared as job options properties. These data members can be set via a job options file before method 3.1 is called. More complicated member variables can then be initialised using these variables supplied at run-time.

Event Data Service

Athena's transient event store is used to store data which can be accessed by any Algorithm scheduled within a single job. The store has a lifetime of one event(i.e. one execute cycle), and in the context of Atlfast, holds the monte carlo event record input, and all output data. All classes which inheret the Algorithm base class have access to this store, and the entries are keyed on a string identifier, or by object type.

The Histogram Service

The histogram data store is a more persistent area for storing statistical based data which has a lifetime of more than one event. Athena provides histogram classes which conform to the AIDA(Abstract Interfaces for Data Analysis) standard[16]. These can be booked and manipulated by Athena's histogram service, accessible to Algorithm classes. Histograms are persistified at the end of a job as an HBOOK or ROOT file[17].

The Ntuple Service

The ntuple service interface provides Algorithms with access to the ntuple store, where ntuples can be created and manipulated, or read in from files. Ntuples are formatted data structures which can be used to store event data, and have a lifetime of more than one event. Ntuples are persistified in HBOOK or ROOT format.

The Messaging Service

The message service provides facilities for the logging of information, warnings and errors. Algorithms can output information to a message stream via this service, and specify an 'output level' which determines the priority of the message(eg. debugging, information, errors). A run-time output level can be set both globally and for individual Algorithms that determines which messages are displayed and which are ignored.

3.2 The Atlfast Package

The Athena-Atlfast package aims to simulate the response of the ATLAS detector in a quick and practical way to avoid CPU and time consuming sophisticated full detector simulation. This 'fast simulation' is useful for making estimates of signal and background rates for specific channels and high-statistics background studies at the LHC.

It includes the most crucial aspects of the detector response; lepton and photon momentum resolution, jet reconstruction and tagging, magnetic field effects, track reconstruction and expected missing momentum. It includes energy calibration and detection efficiencies for jet reconstruction, although it does not attempt to produce efficiencies for lepton or photon isolation. For previous ATLAS studies, these efficiencies are not parameterised in any way and are generally applied as a percentage reduction in signal efficiency per isolated particle. It is therefore preferable to leave control of this to the analysis of Atlfast data.

3.2.1 Overview

The Athena-Atlfast simulation software is an Athena package which slots into a chain comprising three main steps; event generation, detector simulation, and event analysis.

The input for the Athena-Atlfast package is the HepMC event record[18], an object oriented event storage structure containing Monte Carlo generated four-vectors. Relationships between the four-vectors are preserved, and the vertices can be traversed using iterators.

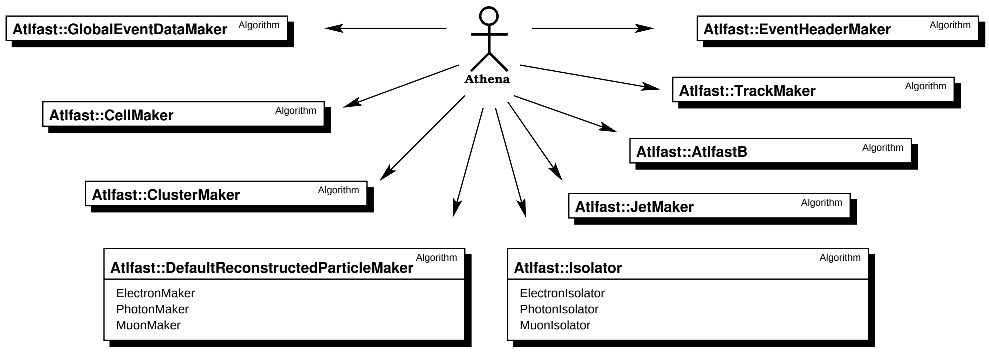


Figure 3.1: *The Core Atlfast Modules*

The output of the Athena-Atlfast package is a number of analysis objects, such as isolated leptons and photons, reconstructed jets and general event information such as missing momentum.

The Atlfast package consists of a number of modules which perform the particle, track and calorimeter parameterisations and jet reconstruction. The core Atlfast modules are shown in figure 3.1. These Algorithms are scheduled by Athena as shown, and are initialised and executed in this order.

The GlobalEventDataMaker is responsible for making available the data which are common to all Algorithms, such as luminosity and magnetic field strength. The Monte Carlo particle energies are then deposited in calorimeter cell objects by the CellMaker Algorithm. These are formed into Clusters by the ClusterMaker for jet reconstruction.

There are three DefaultReconstructedParticleMaker algorithms responsible for electron, muon and photon production. Three instances of the Isolator algorithm carry out particle isolation for electrons, photons and muons respectively.

The JetMaker class is responsible for making Jet objects from the clusters and the AtlfastB algorithm performs jet energy recalibration and estimates b-tagging efficiencies.

Finally, an EventHeaderMaker creates an object containing other event information available to the user, such as missing energy. There also exist optional Algorithms which create tracks and write ntuples for analysis.

Each algorithm's input and output objects must be stored on the Athena Transient Event Store, thus all Algorithms aggregate a service class called TesIO which encapsulates the mechanism for storage and retrieval.

3.2.2 Interfaces

Interfaces are used extensively in both the Atlfast and AtlfastAnalysis packages, allowing powerful generality via the mechanism of polymorphism. Interfaces specify a programming contract between objects, defining functionality while keeping the implementation encapsulated, thus providing a level of indirection which keeps the client class independant of the service class. This decoupling of functionality and implementation is extremely powerful in designing robust, maintainable, and extendable software.

In C++, interfaces are implemented as pure abstract base classes, which define the interface methods but contain no implementation. These methods are over-ridden by concrete implementations of this interface using the inheretence mechanism.

3.2.3 Design Patterns and Generic Algorithms

A number of design patterns[19] and STL algorithms[20] are used throughout the Atlfast and AtlfastAnalysis design. These are described here.

The Filter Pattern

The Filter pattern allows objects with compatible interfaces that perform different operations on data to dynamically connect in order to perform arbitrary operations. Figure 3.2 shows the Filter pattern. The AbstractSource interface is used by the DataClient to retrieve the data. While the ConcreteSource object simply provides data, the ConcreteSourceFilter extends the behaviour of the obtainData() method, performing some data transformation, whilst remaining completely transparent to client objects.

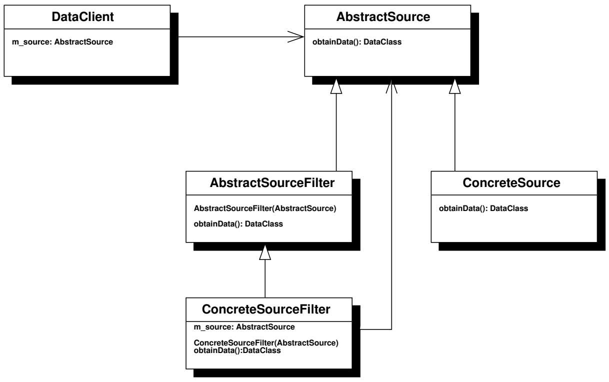


Figure 3.2: *The Filter Pattern*

An example of its use in Atlfast is the IMCSelector Monte Carlo filter suite(section 3.3.1), where a number of filter objects can be combined to form complex filtering algorithms which extract specific particles from the event record.

The Factory and Prototype Patterns

Many algorithms or classes aggregate service provider classes via interfaces, providing a level of indirection between the client and service, allowing different implementations of the service to be selected dynamically.

The factory pattern provides a way to keep the client-service independance intact by providing a class which creates the concrete service classes for the client. Figure 3.3 shows the factory pattern.

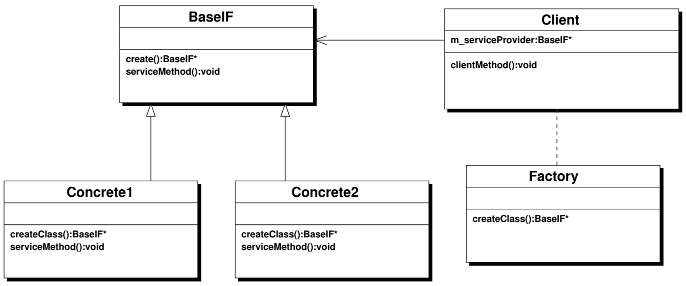


Figure 3.3: *The Factory/Prototype pattern*

The client class is service independent, and as a result does not know which specific concrete class is providing its service. The service classes can therefore not be copied or reproduced by the client which may be necessary, for example, in a client copy constructor.

Interfaces therefore provide a 'prototype' method which is overridden by concrete implementations in order to recreate instances of the concrete class.

An example of the factory and prototype pattern is found in the AtlfastAnalysis package where the IDManager factory(see section 5.4.2) creates concrete instances of the IEntityID base class for the AnalysisVariable classes.

The Singleton Pattern

The Singleton pattern ensures that only one instance of a class can be created, and any objects which use the class use this same instance.

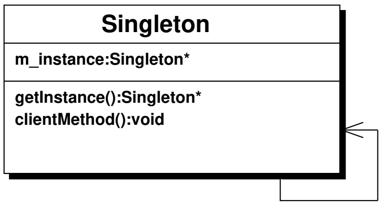


Figure 3.4: *The Singleton Pattern*

Figure 3.4 shows the singleton class pattern. It has a static variable which refers to its one instance and private constructors, which prevent clients from creating new singleton classes.

The class is accessed via a static method which either returns the pre-existing static reference, or creates and returns the new class if it hasn't yet been created. An example of the singleton class in Atlfast is the GlobalData class(see section 3.3.3) which contains common information used by all the Atlfast algorithms.

STL Generic Algorithms

The STL(Standard Template Library) provides a number of generic algorithms which perform operations on STL containers. These algorithms normally depend on a client-supplied predicate condition class, which can make a boolean comparison of two objects.

An example which is used throughout Atlfast is the sort function. This sorts a container of objects according to a supplied predicate class and takes the form 3.4, while the predicate class must, in the case of a binary algorithm, supply the 3.5 method.

```
std::sort(container.begin(), container.end(), Predicate);    (3.4)
bool operator()( *(container::iterator) a, *(container::iterator) b );    (3.5)
```

A number of predicate function objects are provided in Atlfast for the most common tasks. These function objects work on containers of IKinematic objects(see next section), and can therefore be used to sort, copy, or partition almost all Atlfast output classes.

Sort by ascending and descending predicates are provided for particles' kinematic properties. Binary predicates are provided which take a reference to another object as a constructor argument, for example if a sort is required for objects closest in pseudorapidity to a reference particle. Partition conditions are also supplied which partition containers according to a positive or negative predicate result, such as objects above a momentum threshold. These sort and partition objects are listed in appendix A.1.

A Cluster partition predicate is also provided, ClusterIsAssoc, which partitions a container into Clusters which are associated with ReconstructedParticles, and those which are not.

3.2.4 Atlfast Output Objects

The aim of the Atlfast package is to take generated Monte Carlo particles, apply a parameterised detector response and create output objects which a physicist can use to perform an analysis. The output classes and interfaces are described below.

The IKinematic Interface

In most practical analyses, objects such as isolated electrons, muons or photons are viewed as little more than labelled four vector objects. These objects, although different, are all kinematic objects and should honour an appropriate interface. Not only does this standardise method names for different types of object, it provides the type abstraction required for generic kinematic functions.

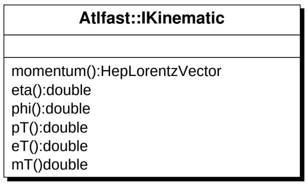


Figure 3.5: *The Atlfast IKinematic Interface*

The IKinematic interface guarantees a number of standard methods which return commonly used quantities, such as transverse momentum and pseudorapidity. These are shown in figure 3.5

The HepLorentzVector class[18] represents the momentum four-vector and provides many methods which return useful associated quantities. The transverse momentum, azimuth and pseudorapidity methods are included for convenience.

Particles

The Atlfast class representing simple particles, i.e. electrons, muons and photons, is called ReconstructedParticle. Although these particles are physically different, they are treated as a single object due to their identical kinematic nature, whose instances are only differentiated by a label.

Figure 3.6 shows the class diagram for the ReconstructedParticle. It inherets from IKinematic, and therefore honours the standardised kinematic interface described previously.

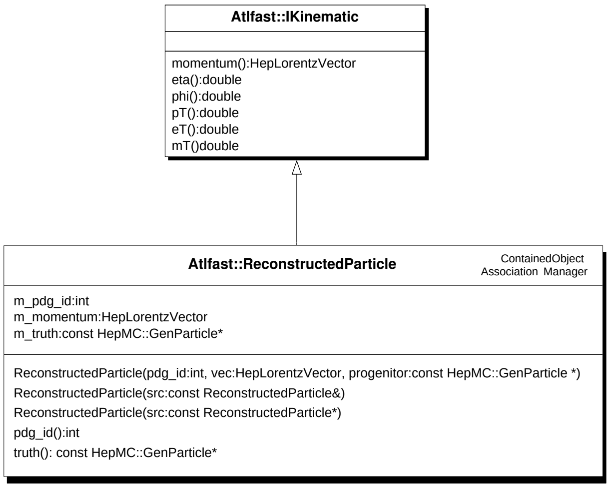


Figure 3.6: *The Atlfast ReconstructedParticle Class*

It is instantiated with an integer PDG(Particle Data Group) ID[21], a momentum four-vector, and a pointer to the Monte Carlo particle from which it was created. As well as the kinematic interface methods, the ReconstructedParticle provides a method to return this monte carlo particle for interrogation.

ReconstructedParticle inherets from two further classes. For an object to be storable in a container on the Athena data store, it must inheret from the

Athena component class ContainedObject. The AssociationManager class is responsible for associations between Atlfast objects. This is described in full in section 3.3.6

Cells

Atlfast Cell objects represent calorimeter cells. In an event, final state particles hit the calorimeter and deposit their energies in these cells. These are then used to determine particle isolation, and in cluster forming for jet reconstruction. It is possible that these objects may also be useful to a physicist in an analysis.

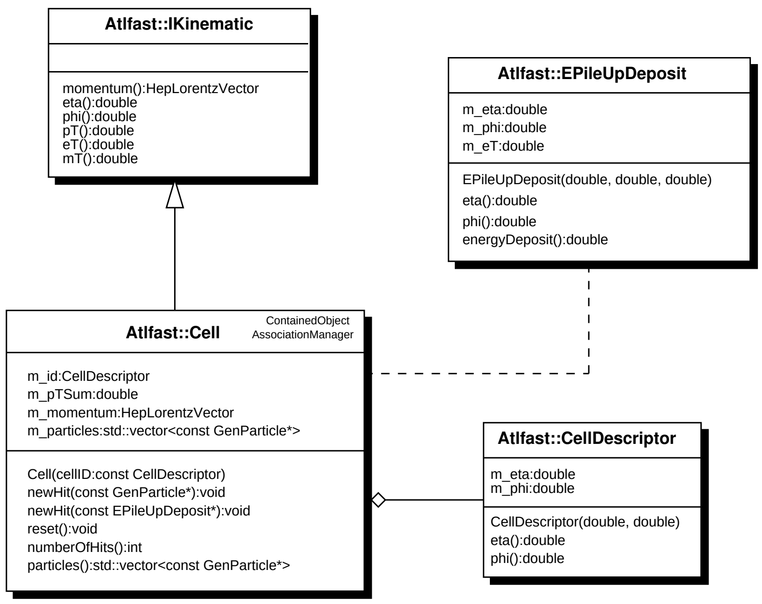


Figure 3.7: *The Atlfast Cell Class*

Figure 3.7 shows the class diagram for the Atlfast Cell. It contains a cell descriptor which encapsulates the cell identification via its $\eta \times \phi$ geometry. An accessible container of the Monte Carlo particles which have deposited energy in the cell is also aggregated.

Although a cell is only in a limited sense a four-vector kinematic object, it honours the IKinematic interface which is useful in STL sort function algorithms(see section 3.2.3). A Cell is also a ContainedObject, which allows storage within an Athena data store container, and inherets from AssociationManager for particle associations(see section 3.3.6).

The cell collects energy in the form of 'hits' from monte carlo particles, as well as EPileUpDeposit objects. These objects represent depositions from pile up events where, due to the high interaction rate at the LHC, particles from previous bunch crossings deposit energy in the calorimeters.

Clusters and Jets

The Cluster class represents energy summations centred at a 3-vector position. They are constructed from calorimeter Cell objects and are used to reconstruct jets.

Although very simple objects, the strategy by which these clusters are formed, and therefore how jets are reconstructed, has a great impact on the reconstructed event.

Figure 3.8 shows the Cluster class diagram. Clusters are constructed from the energy four-vector and iterators which cover all of the Cells used in the formation of the cluster. The Cluster is associated with these cells via the AssociationManager, described in section 3.3.6.

Clusters are ContainedObjects in order to be stored in Athena TES containers. They also honour the IKinematic interface and therefore implement the standard kinematic methods.

The Jet class represents reconstructed jets in Atlfast. Atlfast Jets are constructed from a Cluster, and proximal non-isolated muons.

Figure 3.9 shows the class diagram for the Jet. It honours the IKine-

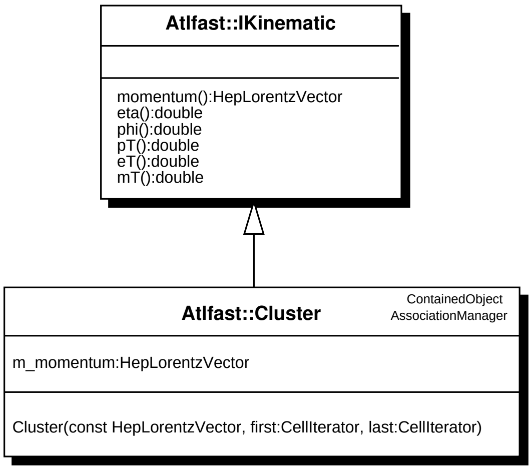


Figure 3.8: *The Atlfast Cluster Class*

matic interface thus providing the standard kinematic methods and, via the AssociationManager described in section 3.3.6, is associated with the Cluster from which it is made, and any muons which may be included.

Tracks

The Atlfast track class represents the track left by charged particles traversing the Transition Radiation Tracker(TRT). Atlfast tracks are described in detail in section 4.

MissingMomentum and EventHeader

The EventHeader is the last analysis object to be constructed by Atlfast and contains useful event information such as the multiplicity of particles and jets. The MissingMomentum object represents all the unused energy deposits in the calorimeter, and is used to calculate the escaped and missing momentum.

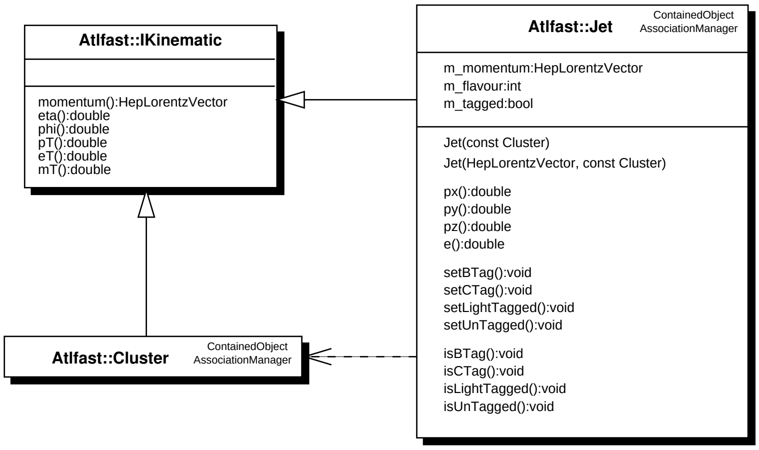


Figure 3.9: *The Atlfast Jet Class*

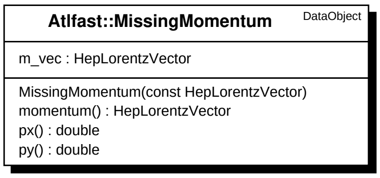


Figure 3.10: *The MissingMomentum Classes*

3.3 Atlfast Components

The Atlfast package comprises a number of Algorithm modules which are scheduled by the control framework. Each individual module is responsible for each different aspect of detector simulation, such as calorimetry, cluster forming, jet reconstruction, particle isolation or track simulation.

In most cases the modules perform an algorithm on a set of input entities, such as Monte Carlo particles or simulated objects such as Cells, and store the end products for use either by the user or another Atlfast module. The individual modules, and their service classes, are described in this section.

3.3.1 Accessing the Transient Event Store

The only method of communication between Atlfast modules, and eventually the analyst, is via the Transient Event Store. Each algorithm retrieves its input objects from the TES and, after having performed its task, stores the end products on the TES for use by other modules or the analyst user.

The TesIO class

The TesIO class is responsible for storage and retrieval of objects from the TES. Every class which communicates with the TES aggregates a TesIO object with which to do this.

This encapsulation of the access to the store assures no reproduction of common access code within the Algorithms, and provides ease of maintenence, with changes in TES access protocols only requiring a modification of the internal implementation of the TesIO class.

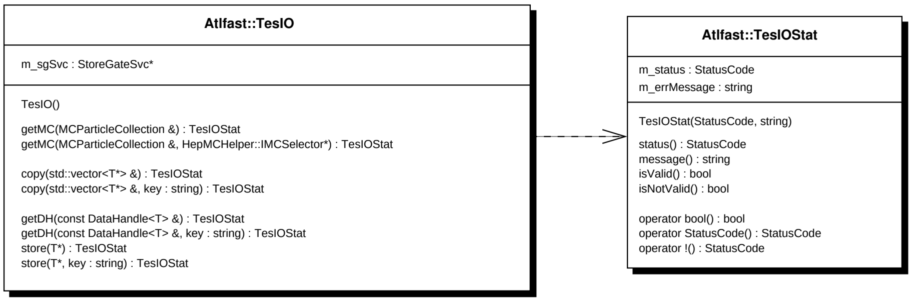


Figure 3.11: *The TesIO Class Diagram*

Figure 3.11 shows the class diagram for the TesIO. It provides methods to retrieve vectors or single objects, and store objects on the TES. The methods are templated so that all types of object can be stored and retrieved. A special method to retrieve Monte Carlo particles from the event record is provided, which can filter the Monte Carlo particles via the IMCSelector interface.

These storage and retrieval methods return a TesIOStat class which represents the status of any communications with the event store, such as the success or failure of a store request as well as any debug or warning messages.

The IMCSelector Interface

In most cases a subset of the HepMC event record is required by an Atlfast module, for example all charged particles or all particles with a certain particle ID. The IMCSelector interface is developed for the application of these criteria to the event record. The interface provides a clone `create` method, allowing an IMCSelector concrete instance to be recreated as described in section 3.2.3. The `operator()` method selects or rejects the HepMC particle according to the relevant criteria encapsulated in the concrete implementation.

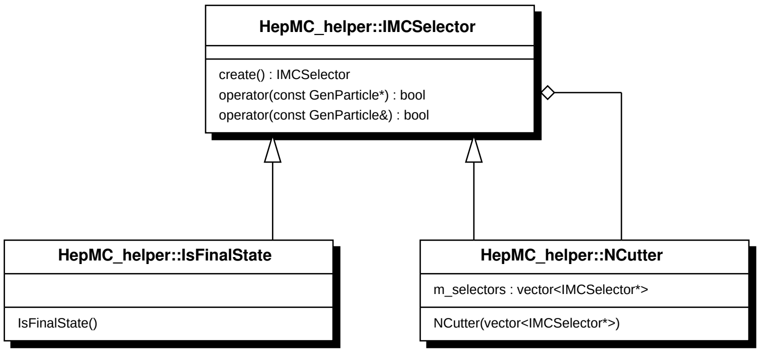


Figure 3.12: *The IMCSelector interface*

Figure 3.12 shows the interface and two example concrete implementations. The IMCSelector suite is based on the filter pattern described in section 3.2.3.

The NCutter concrete class is itself an IMCSelector, and aggregates a container of IMCSelector instances and can therefore perform any combination of single selection criteria.

The client object is completely independent of the selector service and can therefore use a single filter, or any possible combination using the NCutter, without any modification to the client. The concrete implementations of this interface are listed in section A.2.

3.3.2 Smearing

Fast simulation involves 'smearing' monte carlo four-vectors according to detector performance parameterisations. ReconstructedParticles, Cell energy depositions, Jets and Tracks are smeared in different ways in order to emulate the performance of the detector.

The ISmearer Interface

Although the implementation of electron, muon, photon, cell and jet smearing is quite different, the smearing functionality itself depends only on an entity's four-vector. The ISmearer interface is therefore developed as an abstract base class for smearing four-vector objects and one public method as shown in 3.13. The smearing of all four-vector type objects can therefore be carried out via this interface, without explicitly needing to know the implementation of the smearing.

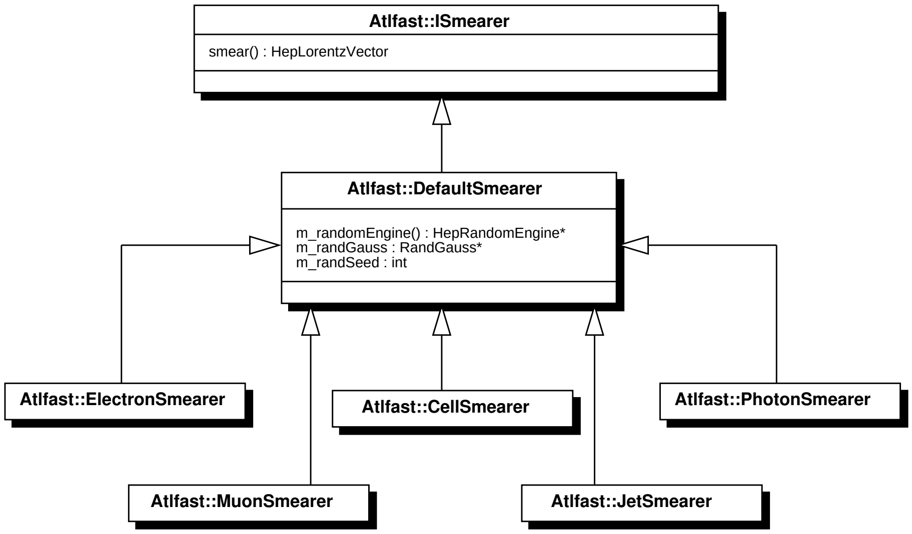


Figure 3.13: *The ISmearer Concrete Classes*

A ParticleMaker class can, for example, aggregate any ISmearer and use its implementation via the `smear()` method, irrespective of the type of smearer it aggregates. Thus, any number of existing smearers can be selected for use at run-time, and new smearing implementations can be developed and slotted in without the need for modification.

The four-vector smearers are shown in figure 3.13. DefaultSmearer stores common member variables for each smearer, such as the random seed, and random engines.

The ElectronSmearer and PhotonSmearer implement simple parameterised Gaussian smearing which is sensitive to luminosity. The MuonSmearer is more complex, invoking an external fortran function to calculate the smear resolution of the muon from persistent histograms.

The Cell smearing is based on a simple Gaussian energy smearing, with randomised, luminosity sensitive pile-up simulation. Jet Smearing is a historical predecessor of the cell smearing based on the same luminosity sensitive Gaussian parameterisation with a more primitve pile-up simulation, and is included for completeness. The smearing parameterisations are described in more detail in appendix A.

Track smearing does not deal with four-vector IKinematic objects, but five parameters which describe the helix path that a charged particle follows in a magnetic field. This smearer hierarchy is therefore not relevant to track smearing, which is described in detail in section 4.2.1.

3.3.3 Global Event Data

The Atlfast modules are Athena Algorithms and therefore cannot be instantiated with arguments, so all state variables must be passed to the algorithms via job options. To avoid having a long list of parameters which are common to all Algorithms, a GlobalEventData class exists which provides common data, such as luminosity, to the Algorithms.

The TES is only accessible to Algorithms during Athena execution cycles, so this cannot be used to distribute the data amongst the Algorithms at initialisation. The GlobalEventData class follows the singleton pattern described in section 3.2.3 and is therefore accessible without the use of the TES. The GlobalEventDataMaker Algorithm and GlobalEventData class are shown in figure 3.14.

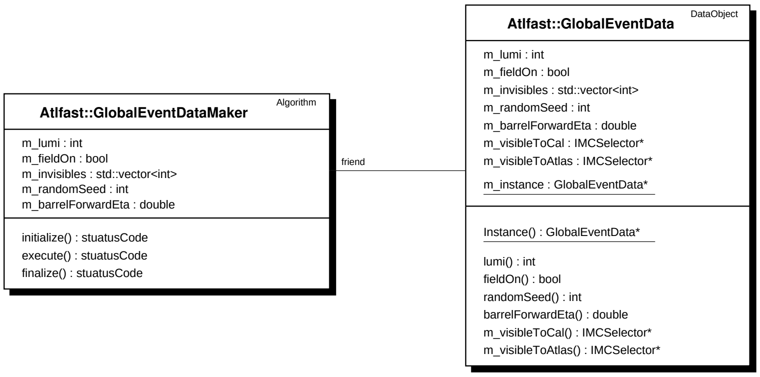


Figure 3.14: *The GlobalEventDataMaker and Global Event Data Class*

The singleton GlobalEventData is created via the static instance method by the GlobalDataMaker before any other algorithms are initialised. The GlobalDataMaker is a friend class, initialising the GlobalEventData via a private method. These values can therefore not be altered by unauthorised clients.

3.3.4 Particle Simulation

The Algorithm responsible for particle production in Atlfast is called DefaultReconstructedParticleMaker, shown in figure 3.15. It creates ReconstructedParticle objects which represent electrons, muons and photons.

It aggregates a TesIO class for communication with the TES, an IMCSelector base class to filter its input Monte Carlo particles, and an ISmearer base class to perform the smearing.

The particle type which will be produced is set via job options and at initialisation the correct Smearer object is created for this particle species. An NCutter object is instantiated which accepts only final state Monte Carlo

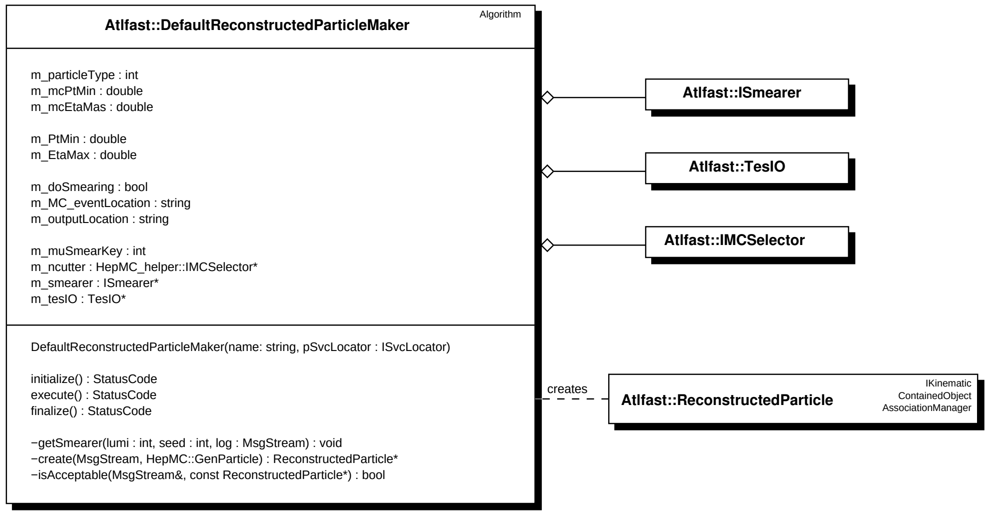


Figure 3.15: *The DefaultReconstructedParticleMaker Classes*

particles of the correct species, passing kinematic cuts also set via job options.

During an execution cycle, the DefaultReconstructedParticleMaker retrieves the relevant Monte Carlo particles from the event record and creates ReconstructedParticle objects from them.

If smearing is activated, the ISmearer then smears the particle four-vectors. Acceptance cuts are applied, and successful candidates are stored on the TES. Job Options parameters, smearing details, and the execution sequence diagram are described in A.3.

3.3.5 Calorimeter Simulation

The CellMaker algorithm is responsible for simulating the Calorimeter response. It aggregates a Calorimeter class which encapsulates the energy deposition details, and a MagField class which is responsible for bending particles in the magnetic field. An EnergyPileUpMap is responsible for simulating energy pile up in the cells. These classes are shown in figure 3.17.

The Calorimeter class consists of three instances of a delegate class called CalSection, which represent the barrel and end cap calorimeter sections. These CalSection classes consist of a map of Cell objects.

The TesIO retrieves the relevant Monte Carlo particles and the MagField instance creates TransportedParticles(particles positioned at the calorimeter, rather than their production vertex). These particles are passed to the Calorimeter, where the CalSectionReject function object tests them for kinematic acceptance. If acceptable, their energies are deposited in the Calorimeter's Cell objects.

If smearing is activated, the CellMaker's EPileUpMap reads a randomly chosen data file and creates a number of EPileUp instances which are also deposited in the Cells by the Calorimeter. The ISmearer instantiation is passed to the Calorimeter and the hit cells (smeared if smearing is activated) are returned for storage on the TES. Job Options parameters, smearing parameterisation, and a detailed execution sequence diagram are shown in appendix A.4.

Clusters

Clusters represent groupings of energetic calorimeter Cells. The strategy used to make Clusters subsequently determines how jets are reconstructed. Figure 3.16 shows the ClusterMaker class diagram. It aggregates an IClusterStrategy abstract base class, which provides the client interface for all ClusterStrategy objects.

Currently, only a simple cone strategy is implemented, but the use of the interface allows other strategies, once they are implemented, to be selected dynamically. The class diagram shows the ClusterConeStrategy which uses a PreCluster helper class to form simple clusters around initiator cells.

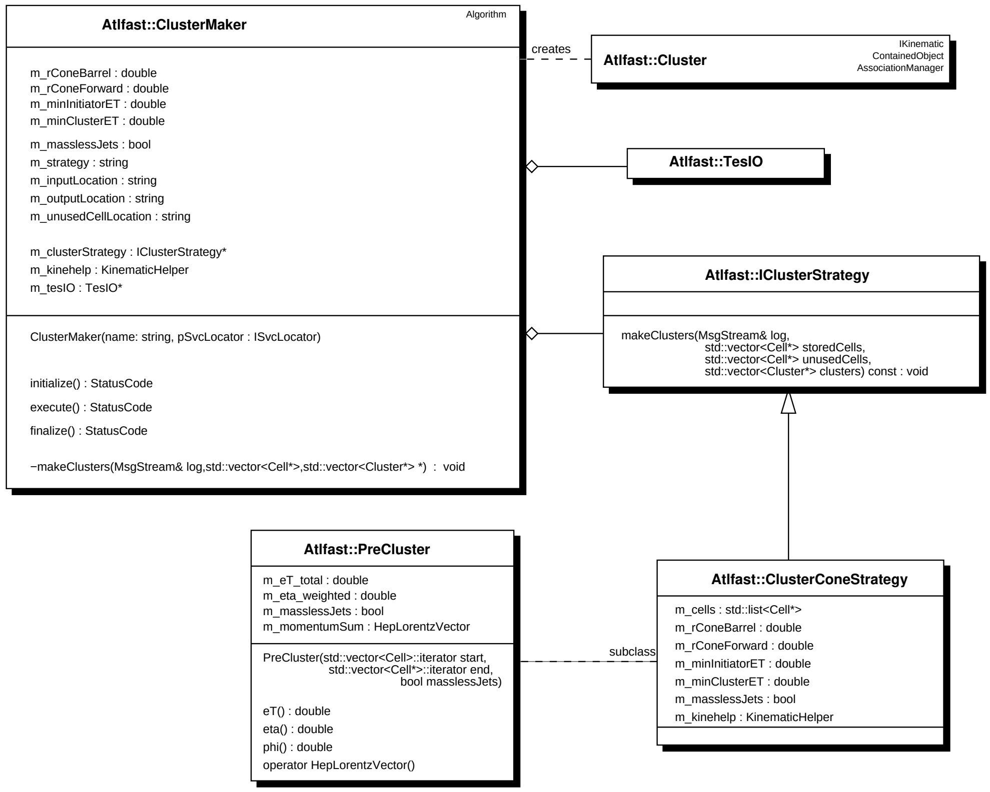


Figure 3.16: *The ClusterMaker Class*

The ClusterMaker execution sequence is very simple. The TesIO object copies the Cell objects from the TES, which are then passed to the IClusterStrategy object. This creates a vector of Cluster objects and a vector of unused Cells(energetic cells which are not included in any clusters), which are stored on the TES for use by the JetMaker Algorithm. Job Options parameters, detailed execution sequence diagrams, and details of the implemented strategies are described in appendix A.5.

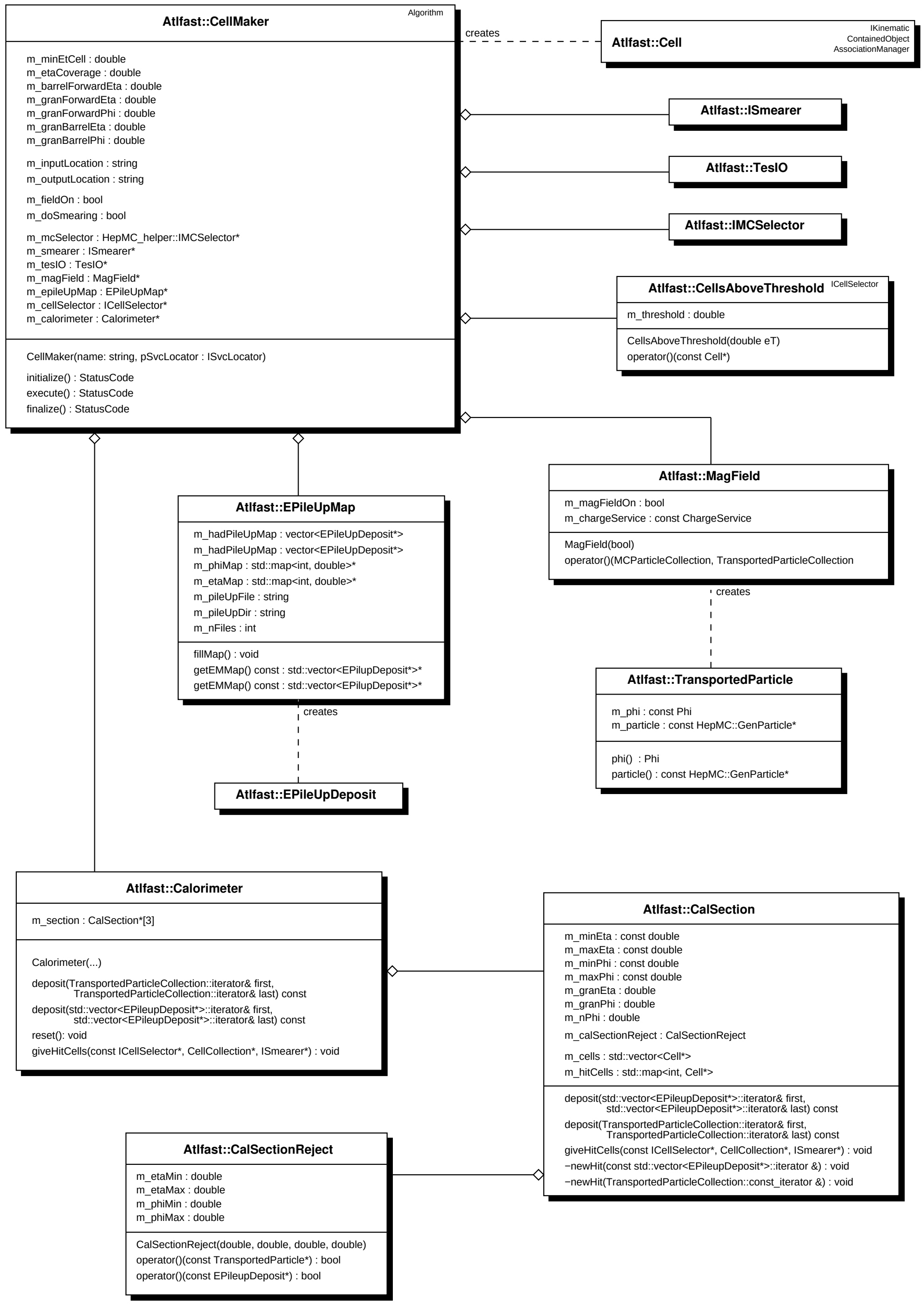


Figure 3.17: *The CellMaker Class*

3.3.6 Particle Isolation and Cluster Association

The Isolator Algorithm is responsible for determining the isolation of ReconstructedParticles and associating them with Cluster objects. Three Isolator objects are instantiated in a run, one for electrons, one for muons, and one for photons.

Only Clusters which are not associated with other particles are considered for the isolation and association algorithm. If an unclaimed cluster exists within a $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ cone of 0.15 of the particle, this cluster is considered associated with the particle. This particle is deemed non-isolated if there are any other energetic clusters within a 0.4 ΔR cone of the particle.

The particle is then tested for cell isolation. If there is a total cell energy deposit above $E_{cell} = 10\ GeV$(not including the particle's own energy deposit) within a ΔR cone of 0.2, the particle is also deemed non-isolated. Only isolated particles are finally associated with clusters.

Figure 3.18 shows the Isolator class diagram. It aggregates a TesIO object and collects an IMCSelector object from the GlobalEventData which verifies particle visibility in the calorimeter. A ClusterIsAssoc predicate class(see section 3.2.3) is also used to check pre-existing associations of the Clusters.

ReconstructedParticles, Cells, and Clusters are retrieved from the TES and the isolation algorithm described above is executed. Isolated and non-isolated particles are then written separately to the TES.

Associations are handled by the AssociationManager base class. Classes which can have associations inheret from the AssociationManager. Associations to objects are stored as ContainedObjects, so all entities which can be contained in an Athena container(i.e. all Altfast output entities) can be associated. Methods are provided which return all of an entity's associations, as well as templated methods which only return associations of a specific

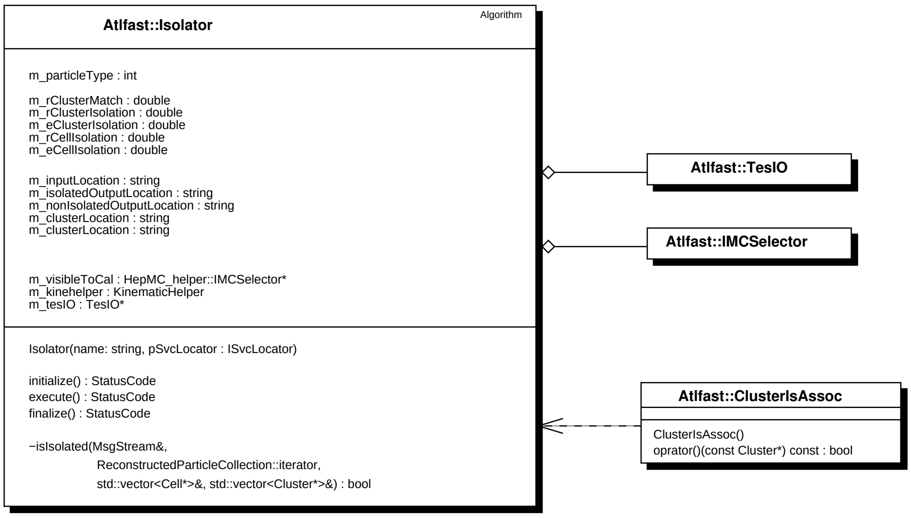


Figure 3.18: *The Isolator Class*

type(eg. only associated ReconstructedParticles).

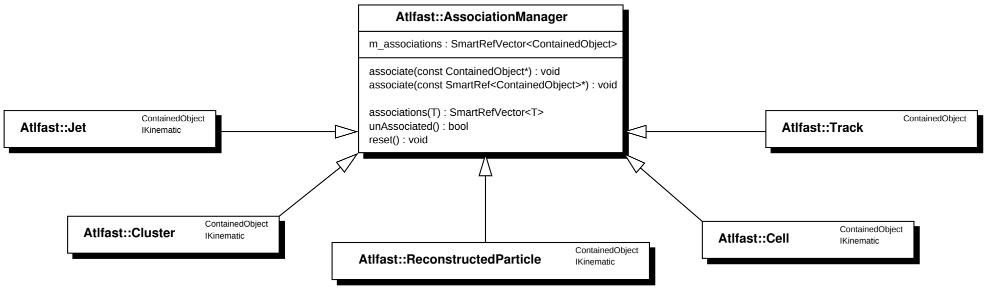


Figure 3.19: *The AssociationManager Class*

3.3.7 Jet Making

The JetMaker class creates Atlfast Jet objects from the Clusters, and also performs the necessary book-keeping for missing momentum calculation. It aggregates a TesIO object, an ISmearer instantiation, and also depends on the SelectTauTag and SelectJetTag IMCSelector classes to retrieve particles for jet tagging.

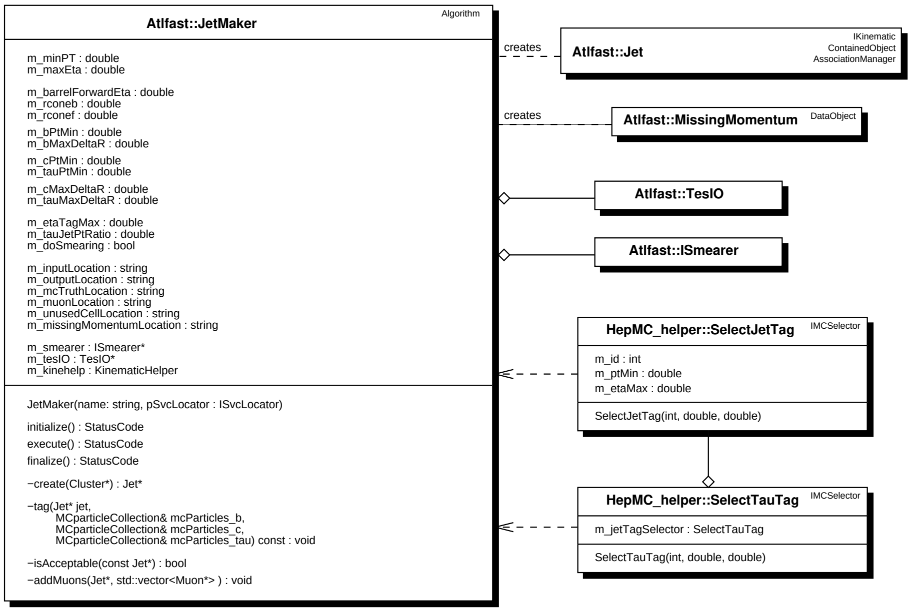


Figure 3.20: *The JetMaker Class*

Clusters, non-isolated muons, bottom quarks, charm quarks, and tau leptons are retrieved from the TES. The Cluster collection is iterated over, and for each Cluster which is not associated with a particle, a Jet is created. If smearing is activated, the jet momentum is smeared by the JetSmearer class.

If a non-isolated muon is within a minimum $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ cone of the Jet, its momentum is added, and the muon is associated with the jet. The Jet is then tested for kinematic acceptance, and if acceptable, the jet is tested for tagging, otherwise it is deleted.

A Jet can be tagged as either a b-tag, c-tag or tau-tag; it is tagged as a bottom or charm jet if there are any Monte Carlo quarks within a ΔR cone, or a tau jet if there is a tau lepton within a ΔR cone, and the ratio of the visible tau decay products momentum and the jet momentum are above a certain level.

Once all the Clusters have been dealt with, the acceptable Jets are stored to the TES. A MissingMomentum object is created with the momenta of the discarded jets and, after all the unused Cell energies have been added, is also stored on the TES. Job Options parameters and a detailed execution sequence diagrams are shown in A.6.

3.3.8 Jet Calibration

The AtlfastB Algorithm is responsible for adding jet tagging efficiencies to, and recalibrating energies of, Atlfast Jets. It was converted to C++ from the Fortran atlfastb algorithm[22] and is included here for completeness. The AtlfastB class aggregates a TesIO object to extract Jets from the TES, and then to re-store the Jets to a separate location after calibration. Calibration data are read in from files and during execution, depending on a number of switches, randomised jet tagging efficiencies are applied and jet energies are recalibrated.

3.3.9 Track Simulation

The module responsible for parameterisation of charged tracks is called TrackMaker, and creates tracks with differing parameterisations for electrons, muons and pions. Track Simulation is described in full in the next chapter.

Chapter 4

Design of the Track Simulation Module in Atlfast

This chapter describes the simulation of tracks in the Atlfast software package. The module was designed by the author to carry out parameterised smearing of tracks in accordance with the ATLAS experiment's inner detector. The smearing is based on calculated parametrisations of the inner detector performance from full simulation[23, 24].

4.1 Simulated Tracks

The simulation of particle tracks is performed by the TrackMaker suite. Monte Carlo particle four momenta are converted to track helix parameters and parametrised smearing is performed according to particle type. Three types are treated; electrons, muons and pions. All other charged particles are treated as pions.

4.1.1 Track Parameters

Tracks traversing the inner detector are described by a set of five parameters

$$p = d_0, z_0, \phi_0, cot\theta, Q/p_T \quad (4.1)$$

The parameters illustrated in figure 4.1 are defined as follows

- impact parameter, d_0 - the distance of closest approach to the beamline
- z perigee, z_0 - the z-axis projection of the impact parameter
- azimuth, ϕ_0 - The angle between the track and the x-axis at closest approach
- cot theta, $cot\theta$ - The inverse slope of the track in the r-z plane.
- inverse p_T charge, q/p_T - the particle charge multiplied by the inverse of the transverse momentum.

The radius r_T (the transverse x-y plane projection of the distance of closest approach) and the curvature $R_{curv} = p_T/0.3qB$ are also considered useful quantities.

4.1.2 Parameterisation from Full Simulation

Three different parameterisations are included in the track smearing package. Within the achievable precision of a parametric description, muon resolutions are considered as Gaussian distributions. Electrons are smeared using the muonic description, after a parameterised bremstrahlung process has been performed. Pions are described by the sum of two Gaussian distributions representing a core gaussian distribution and a tail.

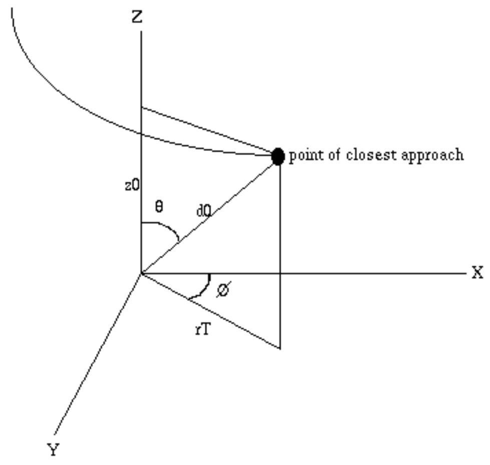


Figure 4.1: *The Track Helix Parameters*

Covariance Matrices

The quantities measured with Gaussian errors are not independent, so they are described by a square covariance matrix of dimension five.

The diagonal terms are the squared sigma values $C_{aa} = cov(p_a) = \sigma_a^2$ of the five parameters in 4.1, while the off-diagonal terms represent correlations between track parameters, $C_{ab} = cov(p_a, p_b) = \rho_{ab}\sigma_a\sigma_b$.

To good approximation, the variables in the x-y plane and r-z plane are independent, so only four of the possible ten correlations are non-zero. A covariance matrix thus consists of nine non-zero parameters; the diagonal elements, $cov(Q/p_T, \phi_0)$, $cov(Q/p_T, a_0)$, $cov(\phi_0, a_0)$, and $cov(cot\theta, z_0)$.

Muon Parameterisation

The muon errors are parameterised in transverse momentum and pseudorapidity. There are 26 bins spread evenly in pseudorapidity over the range $0 < |\eta| < 2.5$, while 14 bins in p_T span the region from 0.5 to 1000 GeV.

For each value of p_T and η, a single covariance matrix is obtained from an average of approximately fifty different matrices, calculated from full simulation with a Gaussian distribution of the z_0 track parameter, centred around $z_0 = 0$.

These matrices are calculated using two choices of detector geometry, magnetic field configuration and vertex constraint.

- detector b-layer inclusion / exclusion
- real solenoidal magnetic field / ideal 2T magnetic field
- vertex constraint / no vertex constraint

The resulting data are input by the fast simulation package, and the covariance matrix for a given momentum and pseudorapidity can then be calculated by two-dimensionally interpolating between these supplied matrices. The resulting matrix is then used to calculate the five smear parameters.

Electron Parameterisation

The electron smearing is applied in two stages; firstly bremstrahlung energy loss is accounted for, and then the track is smeared according to the muonic parameter smearing. The bremstrahlung loss can be characterised by a single radiation at a radius R_{brem}, independent of p_T. A distribution of energy loss scale factors for values of R_{brem} and η is derived from electrons with $p_T = 20GeV$.

For electrons in the fast simulation, a bremstrahlung radius is chosen at random from twelve possible values. Depending on the electrons pseudorapidity and bremstrahlung radius, a p_T loss scale factor is selected at random from the appropriate distribution. This factor is used to generate a new momentum value according to 4.2.

$$
\begin{aligned}
p_T^{brem} &= S\ p_T^{gen} \\
S &= \alpha(1+\beta(S_{20}-1))
\end{aligned}
\tag{4.2}
$$

where S_{20} is the scale factor and $\alpha,\ \beta$ are constants determining energy loss and p_T deviations from 20 GeV[1].

It is assumed that parameters in r-z plane are not affected by bremstrahlung, so only new d_0 and ϕ_0 values are generated from the smeared momentum, according to 4.3

$$
\begin{aligned}
\Delta d_0 &= grad\ \Delta(1/p_T) \\
grad &= \gamma(m_{(R_{brem})} + \Gamma r_{(R_{brem})})
\end{aligned}
\tag{4.3}
$$

where γ allows for $p_T \approx 20\ GeV$, $m_{(R_{brem})}$ and $r_{(R_{brem})}$ are the mean and r.m.s of a distribution, and Γ is a Gaussian random number.

These new helix parameters are then subjected to the same smearing as the muons, using b-layer geometry, an ideal constant b-field and no vertex constraint.

Pion Parameterisation

The pion errors are parameterised in three variables, pseudorapidity, transverse momentum, and radius at the point of closest approach.

For a given bin in η, p_T, and R_T the resolution of each track parameter is described as the sum of two Gaussian distributions of the five track parameters, according to equation 4.4.

$$
F(x) = \frac{f_1}{\sqrt{2P\ det(C)}} e^{-\frac{1}{2}x^T C^{-1} x} + \frac{f_2}{\sqrt{2P\ det(T)}} e^{-\frac{1}{2}x^T T^{-1} x}
\tag{4.4}
$$

where C and T are the covariance matrices of the core and tail Gaussians and x is the track parameter vector 4.1. The fractions and matrices are

[1] $\alpha = 0.994,\ \beta = 1 + log(p_T/20)$

extracted from a fit to full simulation data, and the fractions and matrices are parameterised.

In order to smear the tracks, each track parameter is assigned randomly to the core or tail with probability $f_1/(f_1+f_2)$. These fractions show a slight p_T dependance, and are parameterised according to 4.5

$$f_1/(f_1+f_2) = C_0(\eta) + C_1(\eta)/p_T \tag{4.5}$$

Each diagonal covariance matrix element is calculated from the track's momentum and either the core or tail data parameter coefficients, according to 4.6.

$$\sigma_{aa} = C_0(\eta) + C_1(\eta)/p_T + C_2(\eta)/\sqrt{p_T} \tag{4.6}$$

The off-diagonal correlation coefficients are parameterised as shown in equation 4.7.

$$\rho_{ab} = C_0(\eta) + C_1(\eta)/p_T + C_2(\eta)/p_T^2 \tag{4.7}$$

The C_n coefficients are interpolated using the tracks position within the bin ranges and the resulting matrix is used to calculate the five smear parameters.

4.2 The Tracking Module

The TrackMaker class is the Athena Algorithm responsible for generating and smearing tracks in the Altfast package. It uses HepMC::GenParticle objects from the event record to calculate the tracks, expressed in terms of the five standard track parameters in expression 4.1. These parameters are smeared according to the track's position in smear parameter space.

4.2.1 Design

The TrackMaker suite was designed primarily as a maintainable track smearing package, which would be flexible enough to be easily modified for improved track smearing parametrisations. Generality was also important within this design, as the TrackMaker must be able to handle in a similar way all different species of particles which leave tracks in a detector.

In a fast simulation, a track is generated by calculating helix parameters from a particles production vertex, three-momentum, and magnetic field strength of the detector. Unlike a full simulation or real tracking device, there is no concept of detector 'hits' which are interpolated to form a trajectory, so the calculated tracks are smeared with parameterised Gaussian errors generated from full simulation.

Tracks in Atlfast are represented by the Track class shown in figure 4.2. The two TrackTrajectory classes aggregated by a track represent the helix trajectory calculated from the monte carlo truth, and the smeared trajectory which represents the path measured by the detector. An HepMatrix is also aggregated which is the 5x5 covariance matrix used to smear the track.

The Trajectory itself contains a TrackParameters class which encapsulates the representation of the helix parameters and provides accessor methods for

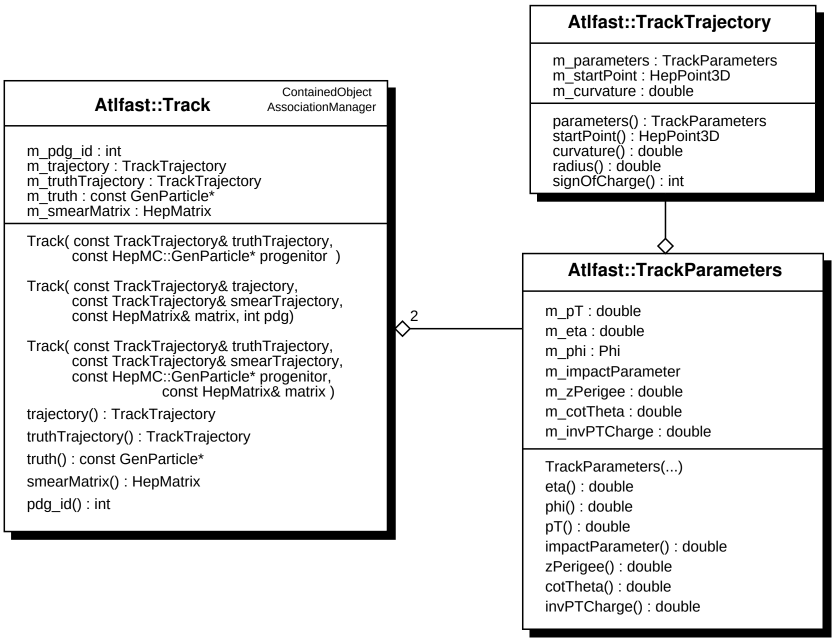


Figure 4.2: *The Atlfast Track Class*

each parameter, the start point of the trajectory, the radius of curvature, and the transverse radius of the start point.

The TrackMaker Algorithm

Figure 4.3 shows the TrackMaker class. It is responsible for making tracks from the HepMC::GenParticle objects, smearing them, and storing them on Athena's Transient Event Store. It aggregates a TesIO object for accessing the TES, an IMCSelector object for selecting the relevant HepMC generator particles, and a TrackSmearer class which provides the smearing service via method 4.8.

$$\text{TrackTrajectory smear(const Track\&) const} \tag{4.8}$$

The IMCSelector selects only charged Monte Carlo particles passing certain momentum and pseudorapidity cuts. These values, and all other parameters

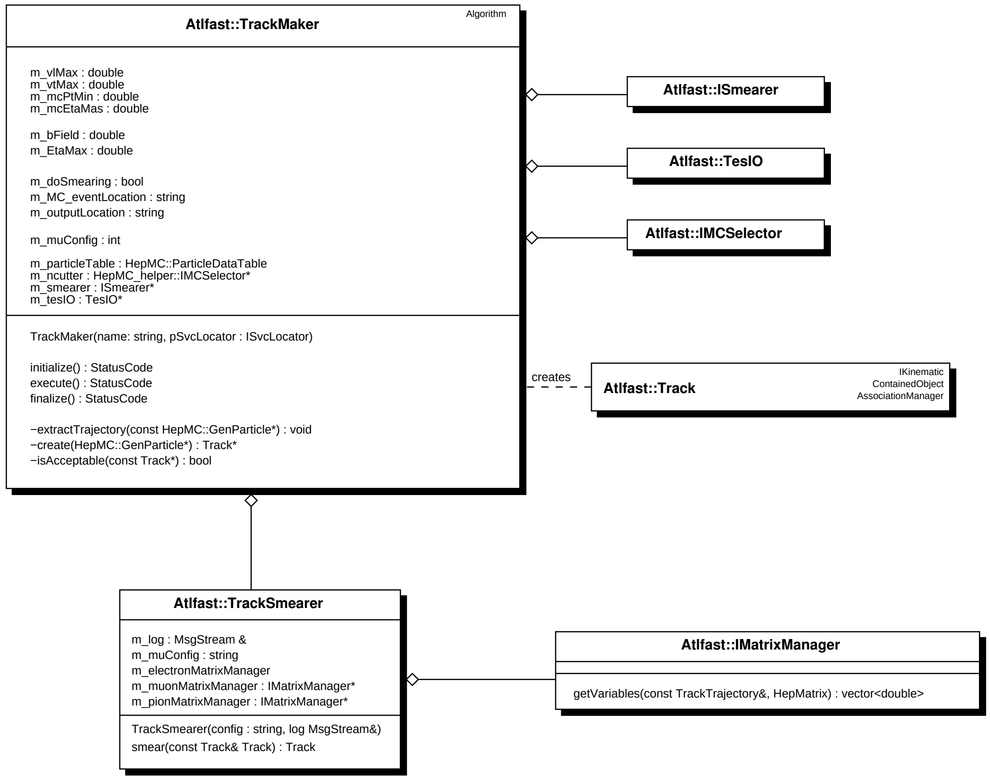


Figure 4.3: *The TrackMaker Algorithm*

for track making, are set here via the job options steering file.

Firstly, the TesIO object retrieves the relevant Monte Carlo particles from the TES for making tracks, filtered by the IMCSelector object. A Track object is created from each HepMC particle which, if smearing is switched on via job options, is passed to the TrackSmearer. The smeared trajectory is added to the track which is then checked for kinematic acceptance. If acceptable, the tracks are added to a storage vector.

Once all the particles have been dealt with, the vector of Track objects is stored on the TES via the TesIO object. The job options parameters and UML seqence diagrams for the TrackMaker suite are described in appendix A.7.

Track Smearing

The smearing of tracks is parameterised in a number of discretely binned variables(eg. p_T, η). For each n-dimensional variable bin there exist parameter values from which, using interpolations from individual track parameters, a covariance matrix is calculated. The covariance matrix is then used to calculate the five smear parameters for the track.

The classes which are responsible for calculating these parameters for the TrackSmearer are called matrix managers, whose functionality is defined by the IMatrixManager class.

The parameterisations are specific to particle species, so the TrackSmearer object aggregates three IMatrixManager objects; one for electrons, one for muons and one for pions. The concrete implementations of IMatrixManager are responsible for calculating the appropriate covariance matrix for the track, and consequently the track smear parameters.

Figure 4.4 shows the matrix manager classes for each smeared species of track. The IMatrixManager interface provides the method definition for returning the smear parameters and the covariance matrix from which they were derived. The CorrelatedData class calculates the square root of the covariance matrix and from this generates the five deviate smear parameters which are returned to the TrackSmearer. The MuonMatrixManager and ElectronMatrixManager are specialisations of a single LeptonMatrixManager class, and a separate PionMatrixManager also exists.

The calculation of the covariance matrix is delegated to BinData classes, which represent the n-dimensional parameterisation variable bins(i.e. bins in p_T, η *etc.*). The IBinData interface, shown in figure 4.5, defines the functionality of these BinData classes. Each BinData has its own exclusive BinID object, which contain the bin edges in n-dimensional parametrised variable

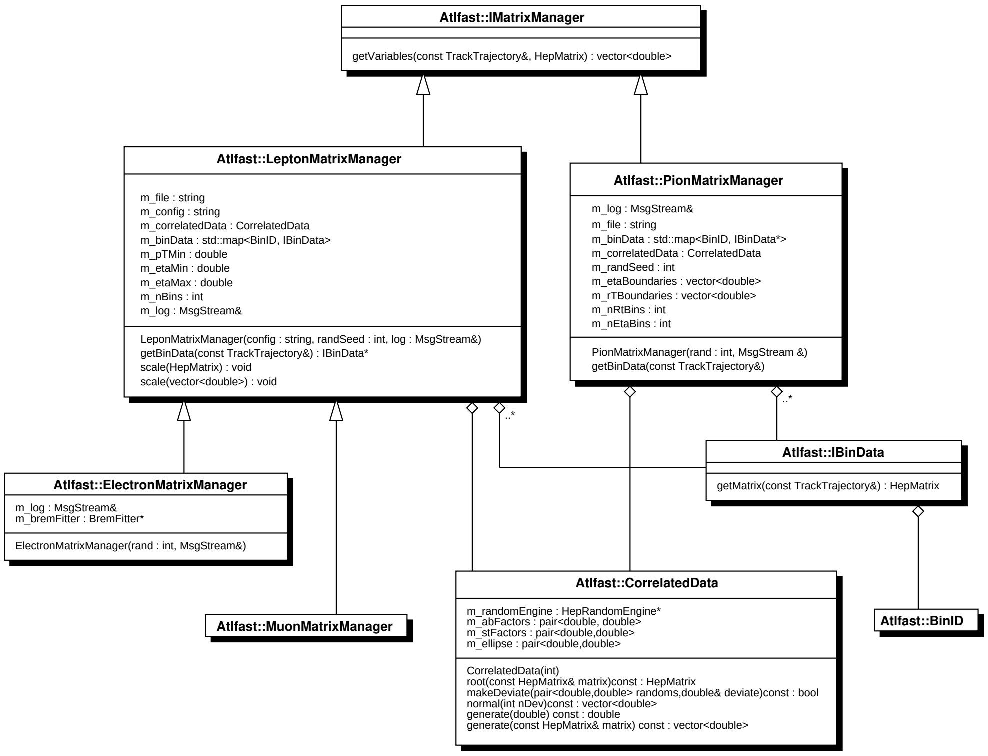


Figure 4.4: *The MatrixManager classes*

space. They calculate the covariance matrix by interpolating the track's kinematic parameters between their own bin edges.

The BinData objects are stored by the MatrixManagers in a map, keyed on their BinIDs, which provide an isInBin method to determine whether an n-dimensional variable falls within its boundaries or not.

The relevant BinID for a track(i.e. the bin in which the track lies) can therefore easily be selected to retrieve the BinData object which calculates the track-specific covariance matrix. Appendix A.7 shows the detailed sequence diagrams for the track smearing classes.

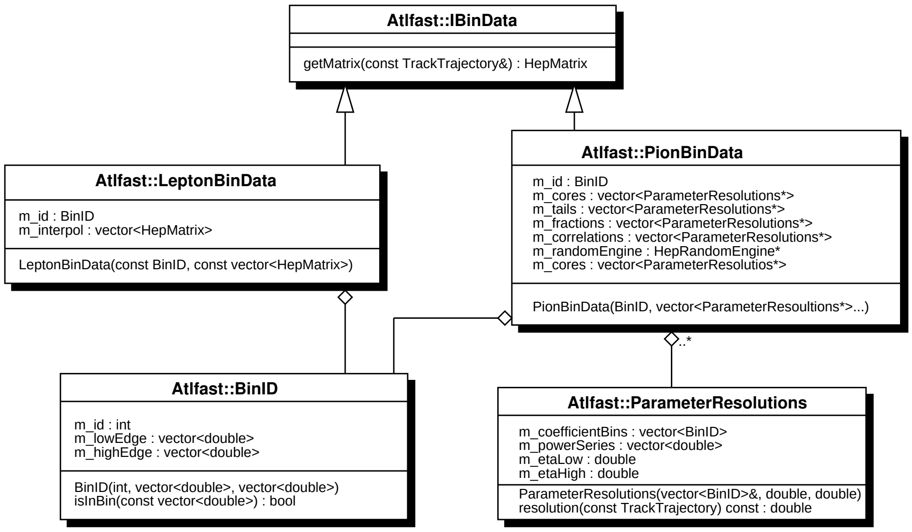


Figure 4.5: *The BinData classes*

The PionMatrixManager

The PionMatrixManager class is responsible for the PionBinData objects which calculate the non-leptonic covariance matrices. The pion track resolutions are parameterised in three variables, transverse momentum, pseudorapidity and transverse radius of the track production vertex. Furthermore, the parameterisations of the track parameter errors are separated into Gaussian core and tail distributions. The core fractions and error correlations are also parameterised in the above three track variables.

Each PionBinData therefore represents a three dimensional bin which contains four sets of parameter coefficients. They aggregate helper classes called ParameterResolutions which hold these coefficients and calculate the parameter resolutions from them, using formulae 4.5, 4.6, and 4.7.

The fraction resolutions determine whether to select a core or tail Gaussian distribution for the diagonal elements of the covariance matrix, and the correlation resolutions are used to calculate off-diagional elements.

The LeptonMatrixManager

The LeptonMatrixManager is responsible for the LeptonBinData objects which calculate the leptonic covariance matrices. It can exist in seven possible states, representing detector configurations, specified by a 3-digit binary number.

The lepton track smearing is parameterised in transverse momentum and pseudorapidity. Each LeptonBinData object contains nine 2x2 interpolation matrices corresponding to each non-zero covariance matrix element and a unique BinID object which represents the two-dimensional bin edges. The MuonMatrixManager is a straightforward derivation of the LeptonMatrix-Manager, providing no extra functionality.

The ElectronMatrixManager is a specialisation of the LeptonMatrixManager. It performs soft transverse momentum scaling to simulate bremstrahlung energy loss using the BremFitter class shown in figure 4.6 before calculating the covariance matrix. It has a fixed configuration comprising inclusion of b-layer geometry, an ideal constant magnetic field and no vertex constraint.

This energy loss can be characterised by a single radiation at a radial distance R_{brem} chosen at random from a set of twelve values. According to this radial distance and binning in track pseudorapidity, a p_T scale factor is chosen, from which new p_T, d_0 and ϕ_0 values are calculated.

In order to implement this scaling, a number of BremEtaBin classes are aggregated, representing the geometrical divisions in pseudorapidity. Each BremEtaBin aggregates a number of BremRBin classes which represent binning in transverse radial distance R_{brem} from the origin.

Each of these η x rT bins aggregates a number of BremPTScaleBin objects which represent the pT scale values, and are responsible for calculating the

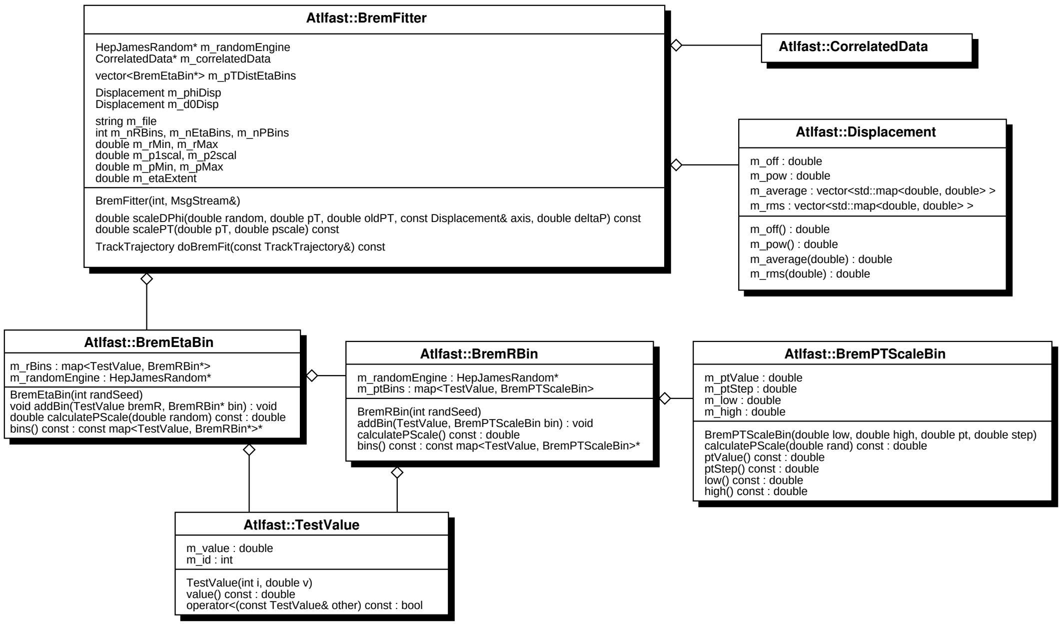


Figure 4.6: *The BremFitter class*

actual interpolated pT scale.

Scaling the transverse parameters is performed by the BremFitter class itself, according to the expressions 4.2 and 4.3. The Displacement classes are responsible for providing the γ, mean and rms parameters in these expressions; one for the impact parameter, the other for the azimuthal angle. The CorrelatedData class provides the Gaussian random number.

All data are read from a single file when the BremFitter class is instantiated. This file contains the bin dimensions and contents, as well as the scaling constants. Appendix A.7 shows the execution sequence of the BremFitter suite.

4.2.2 Tracking Results

The tracking software was based on parameterisations developed by Armin Nairz and tested in Fortran. Comparisons between results from the Athena TrackMaker suite and the Fortran were performed to demonstrate equivalent functionality.

The Fortran implementations have previously been shown to produce good agreement with full-simulation, so the current comparison need only show agreement of the smearing between the two implementations.

Track Trajectory

The calculation of the helix parameters was refactored in Athena-Atlfast and therefore required confidence testing. To test the conversion of production vertices and three momenta into helix parameters, the Fortran and C++ conversion algorithms were compared using 30,000 charged pions.

Figure 4.7 shows the resultant five track parameters for both Fortran and C++. The Athena results are shown on the left, the Fortran results in the centre, and a superimposition of the two is shown on the right. The new implementation shows exact agreement with the previous Fortran calculation of track helix parameters apart from minor differences at the origin caused by differences in precision between languages.

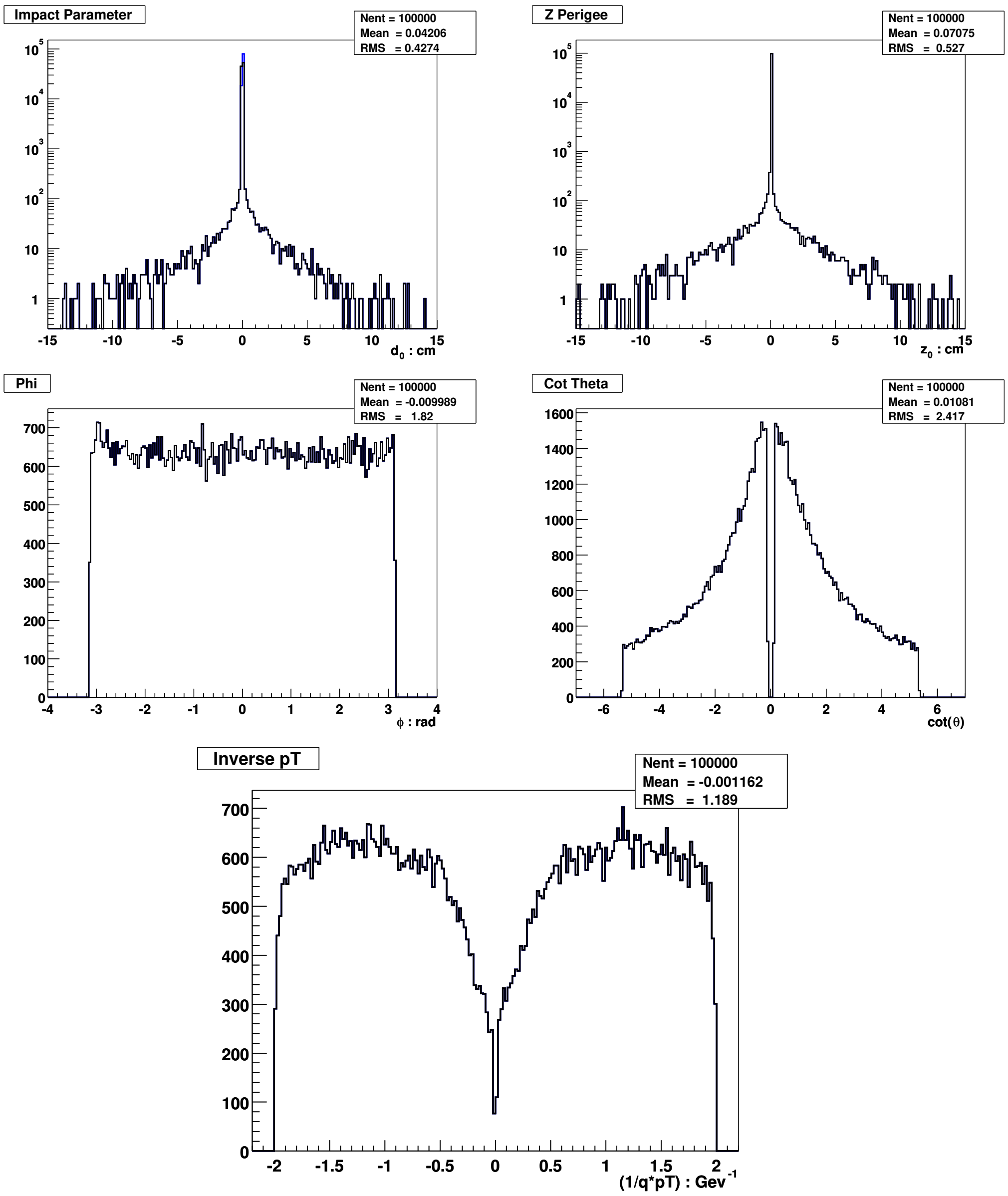


Figure 4.7: *Comparison of Track Parameters: Athena and Fortran results superimposed*

Electrons

Electron tracks use the same smearing as Muons, with an additional bremstrahlung parameterisation. Only p_T, impact parameter and azimuthal angle are smeared to account for bremstrahlung energy loss.

In order to test the bremstrahlung energy loss classes, the Fortran and C++ implementations were compared using 20,000 generated electrons. An identical sequence of random numbers was used for smearing in the first run and both algorithms produced identical results.

The distributions of the bremstrahlung smear parameters using unrelated random numbers are shown in figure 4.8, with statistical comparisons using both chi-squared and Kolmogorov statistical tests, described in appendix C.

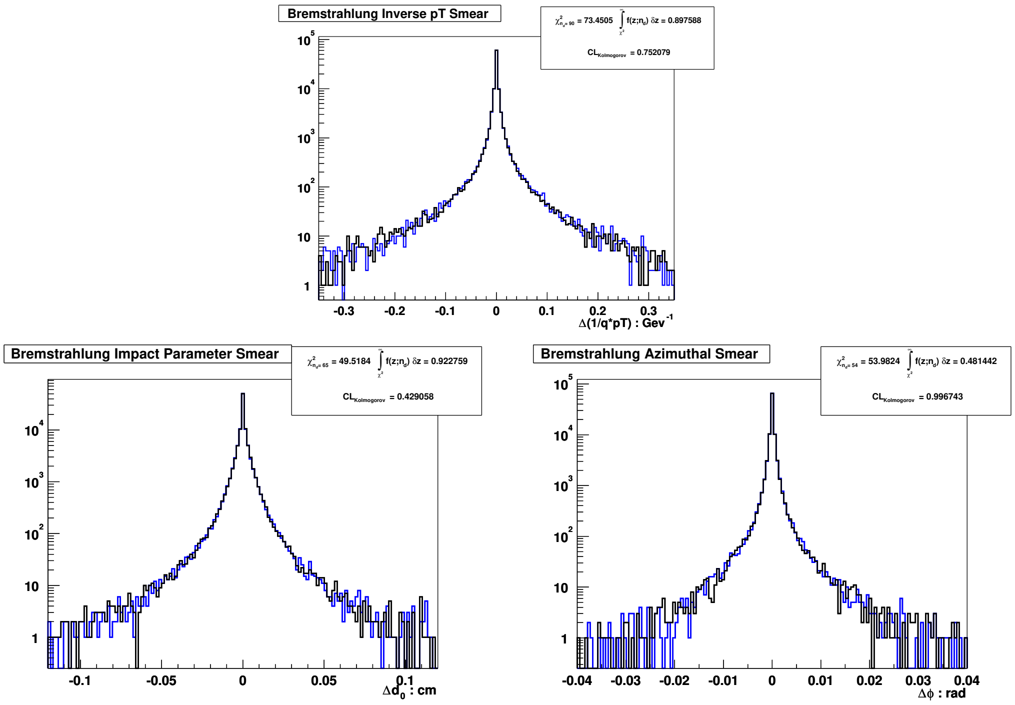


Figure 4.8: *Comparison of Bremstrahlung Smear Parameters: The C++ data are shown in black, the Fortran data in blue*

Muons and Pions

The muon and pion smearing suites were tested using 20,000 generated muons and pions. The Muon smearing was tested using data representing b-layer geometry, a realistic solenoidal B-field, and vertex constraint.

An identical sequence of random numbers was used for smearing in the first run and both algorithms produced identical results. The distributions of the smear parameters using unrelated random numbers are shown in figure 4.9 and figure 4.10. Athena results are shown in black, Fortran results are shown in blue.

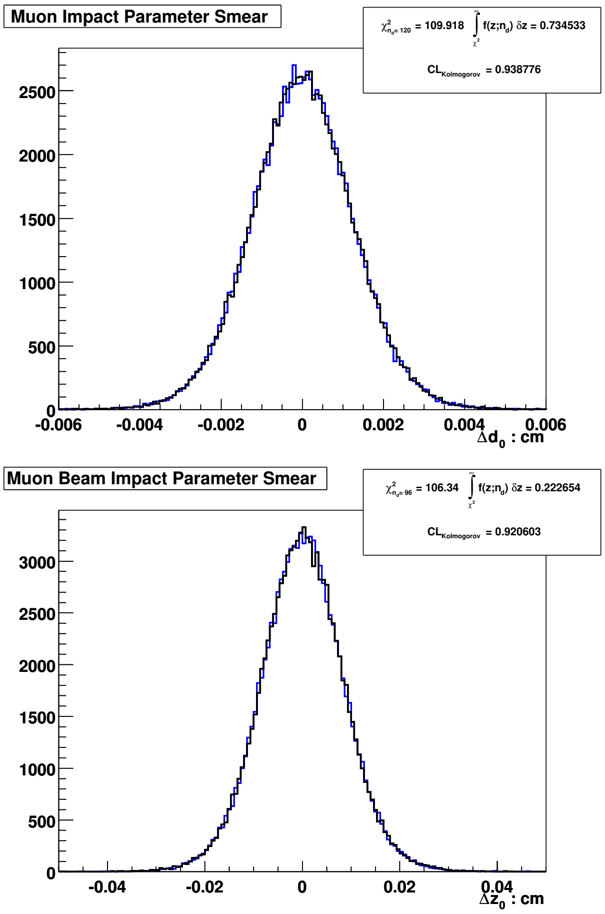


Figure 4.9: *Comparison of Muon Track Parameter Smearing*

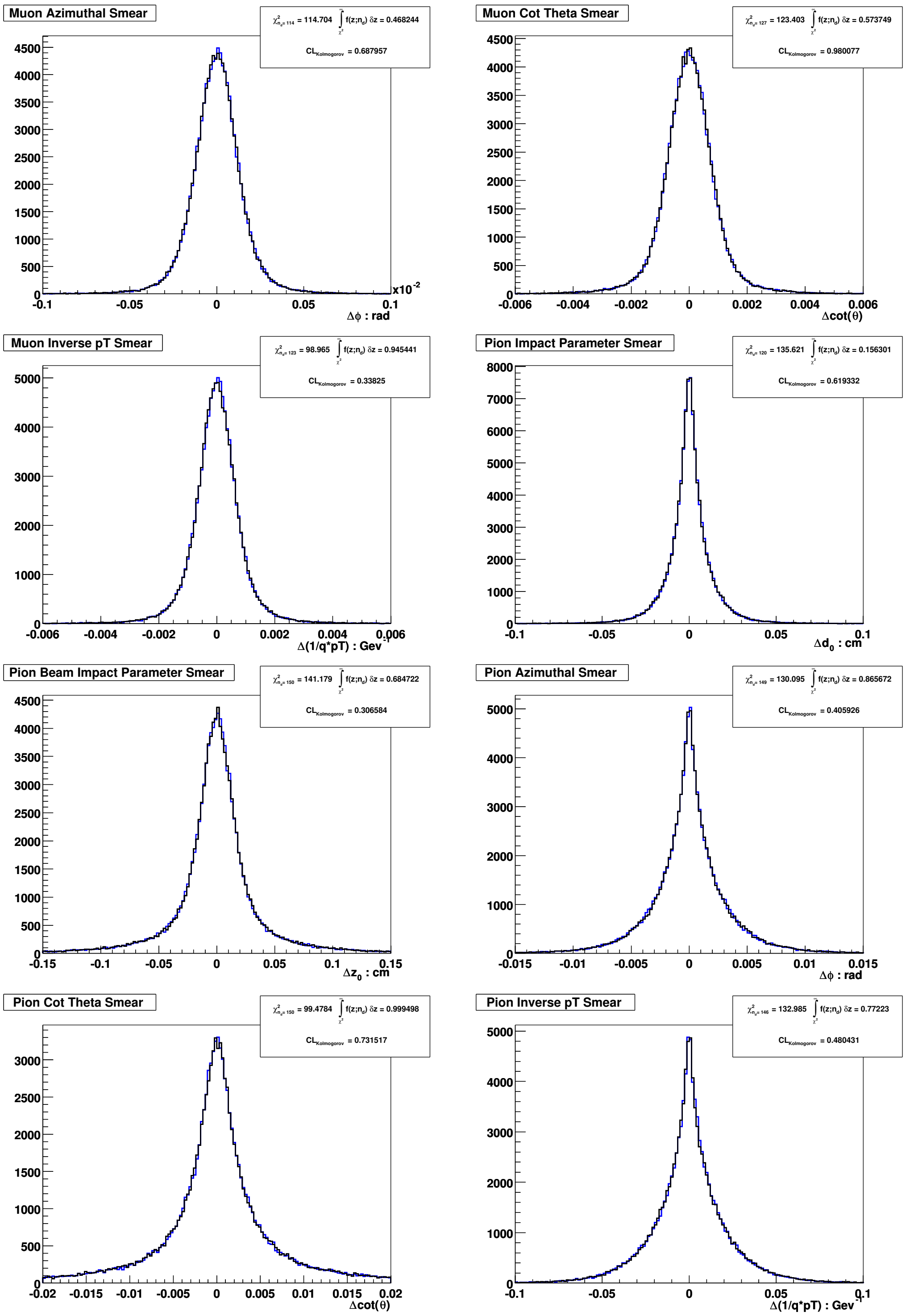


Figure 4.10: *Comparison of Muon and Pion Track Parameter Smearing*

Chapter 5

Development of an Object-Oriented Analysis Package in the Athena Framework

This chapter describes the design, implementation and operation of the AtlfastAnalysis software package, created and developed in Athena for general use by the author.

5.1 Requirements of an OO Analysis Package in Athena

The motivation behind designing and implementing a CMT[25] Analysis Package for use with Atlfast in Athena arose due to the need for physicists to develop analyses within the new software framework using the Object-

Oriented Fast Simulation package. A separate package which uses the Atlfast data to perform user-defined, arbitrarily complex analyses would allow the physicist, generally unfamiliar with the technical details of the software framework and the new Atlfast output objects, to produce physics studies easily within Athena.

The use of AtlfastAnalysis was originally intended for, but not necessarily restricted to, analysis of fast simulation data from the Atlfast package. An attempt was made during the design to avoid fast simulation specificity and could feasibly be extended for use with data produced by more complex reconstruction packages.

The design requirements of the package concern framework compatibility as well as complex functionality and adaptability.
AtlfastAnalysis was required to be a CMT package which runs in the Athena framework with access to all the services provided by Athena. It must be executable in a Generator-Fast Simulation-Analysis chain, with the added possibility of reading some form of persistency store to remove the time-consuming generation/simulation stage from an analysis.
It must be able to extract information from a simulated event, perform variable cuts, produce physics histograms and should allow further implementation of complex analysis functionality if so required. It must also have a simple, user-friendly mechanism for specialising analyses to fit individual physics requirements and display results in a useful manner.

5.2 Design

A good object-oriented design allows simple development, extension and maintainability. The design makes extensive use of interfaces and design patterns which allow functional extendability, generality and flexibility. The inputs objects are encapsulated, allowing the code to be easily extended or adapted for use with non-fast simulation data.

5.2.1 Overview

Design Concept

There are a number of conceptual entities which make up a physics analysis; the physics event, particles, jets, tracks, variables, cuts, histograms. To a physicist, the concept of an event is a topology represented by a number of detected entities such as calorimeter energies, particles, jets or tracks.

The physicist is interested in the properties of these analysis entities; for example their momenta, their pseudorapidities, their energies or their impact parameters. In an analysis, operations are performed on these quantities, such as cuts or histogramming in order to distinguish the interesting physics channel from background events.

Using these ideas the following design concept was developed: There exists an **AnalysisEvent** which contains the **AnalysisEntities**(jets, leptons etc.). **AnalysisVariables** calculate **AnalysisQuantities**(pT, η etc.) from these entities. **FunctionObjects**(Cuts, Histograms) then perform operations on these quantities. The most important data object in the analysis which contains all the information needed by the function objects is therefore the Analysis Event.

Interfaces and Patterns

In order to utilise these generic concepts, a number of 'pure abstract base classes' are developed(figure 5.1). These pABCs provide interfaces that must be honoured by all classes which are specialisations of these generic entities.

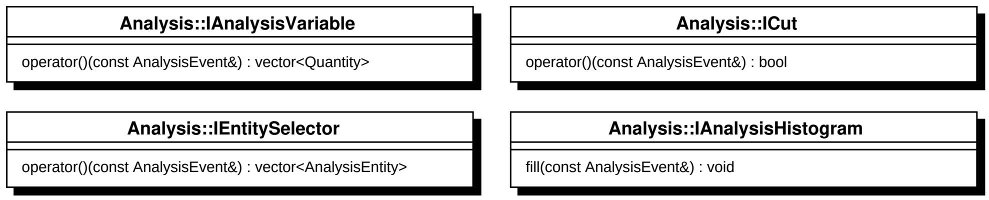


Figure 5.1: *The AtlfastAnalysis Interfaces*

Therefore, all AnalysisVariable objects will calculate quantities from the AnalysisEvent, all Cut objects will return a pass or fail from the AnalysisEvent and all Histogram objects will fill themselves from the AnalysisEvent.

AnalysisVariables generally calculate quantities from different AnalysisEntities within the event in the same manner, thus an IEntitySelector interface is also developed. Concrete implementations of this interface will extract the appropriate entities from the AnalysisEvent. A number of design patterns are used throughout AtlfastAnalysis which are described in section 3.2.3.

5.2.2 Analysis Objects

There are a number of data-like entities described in the Analysis model in section 5.2.1. Firstly, there is the event, which is the main data store representing the complete event topology. The analysis entities represent most of the objects within the event, such as jets, tracks and particles. Finally, analysis quantities are the variable values calculated from these entities.

AnalysisEvent

The AnalysisEvent class represents the physics event, containing all the information needed to perform an analysis. Figure 5.2 shows the collaboration diagram, highlighting its simple data structure of aggregated containers of AnalysisEntity objects. It also aggregates a pointer to the HepMC event record, and a Signature object.

The Signature object is a user defined class which contains special entities or quantities of particular interest within a certain analysis. For example, it may be useful to select entities which pass certain acceptance cuts, jets which originate from a leptonic decay, or tracks above a certain energy. It is often more efficient, and more helpful, to store these topological signatures separately in the Signature class, than to calculate them every time they are used in a cut, or a plot.

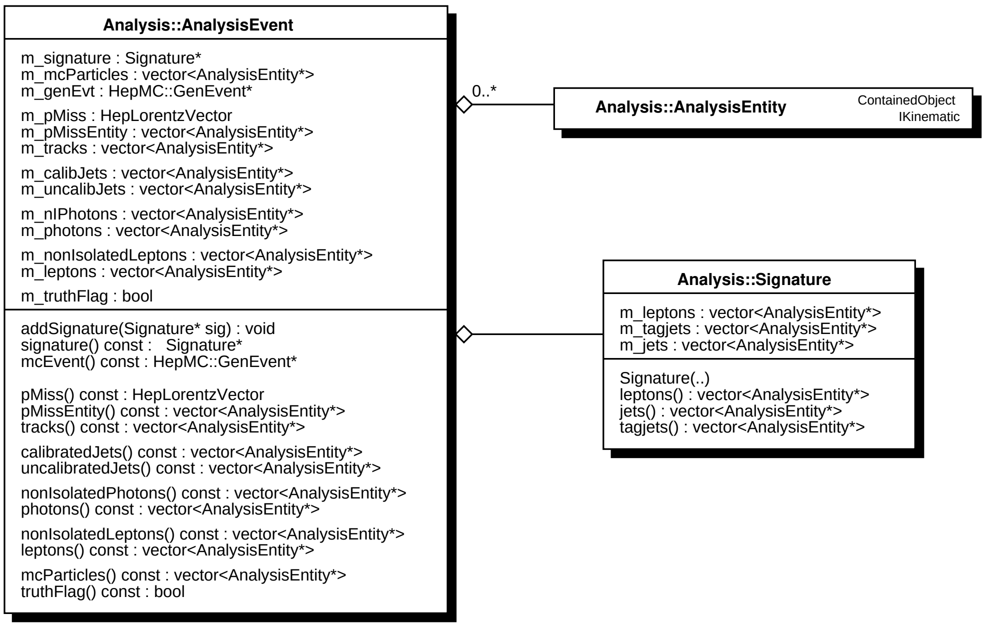


Figure 5.2: *The AnalysisEvent Class*

AnalysisEntity

The AnalysisEntity class represents the objects in the event, such as jets, particles and tracks. For the current fast-simulation use case, only particles, jets and tracks are output for use, so the AnalysisEntity class encapsulates these entities.

Figure 5.3 shows the structure of the AnalysisEntity class. It can be instantiated from the following objects

- Atlfast::ReconstructedParticle(Atlfast output jets, particles)
- Atlfast::Track(Atlfast output tracks)
- HepMC::GenParticle(Monte Carlo event record)
- CLHEP::HepLorentzVector(Ntuple record)

The AnalysisEntity provides methods common to these objects for simplicity, such as transverse momentum, pseudorapidity etc. It also provides methods which return the non-generic objects, such as Trajectory, GenParticle, or momenta, for extracting more complex information.

If an AnalysisEntity client, such as an AnalysisVariable, tries to call a method which is not appropriate for a particular AnalysisEntity, an AnalysisException is thrown, containing a message describing the inappropriate action.

This allows the code to be extended in a controlled manner, such that any type of output entity could be used in the analysis. For example if a physicist needed to analyse Atlfast::Cell calorimeter objects, these could be easily included in the AnalysisEntity class, which would be added to the AnalysisEvent, and AnalysisVariable instances could extract information about them.

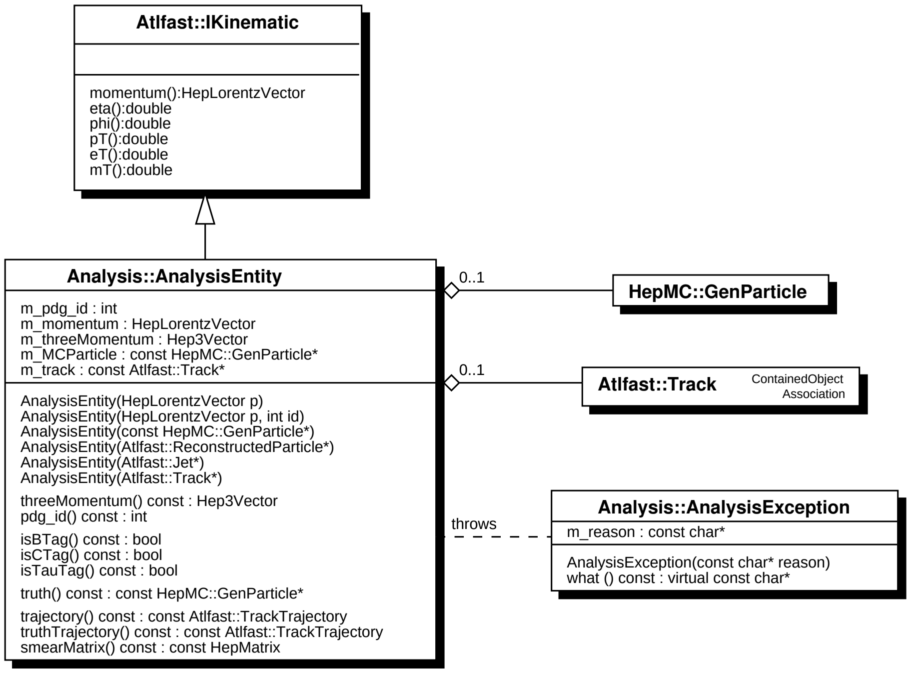


Figure 5.3: *The AnalysisEntity Class*

Quantity

The Quantity class represents the values calculated from the variable classes, and hence cut values and histogram axes.

Quantities in an analysis are generally of two types, integer for multiplicities, floating points for most others. Thus the quantity class encapsulates these separate types in order that all variables can return a general class. Casting, comparison and arithmetic operators are provided which are responsible for encapsulating the quantities identity, internalising all integer or floating point operations.

Figure 5.4 shows the collaboration diagram for the Quantity class. Quantities, in addition to simply representing values, also include the entities from which they are calculated. This is useful in an analysis when one requires calculation of a variable from a subset of entities, for example, if you want to identify the pseudorapidity of the lowest pT lepton.

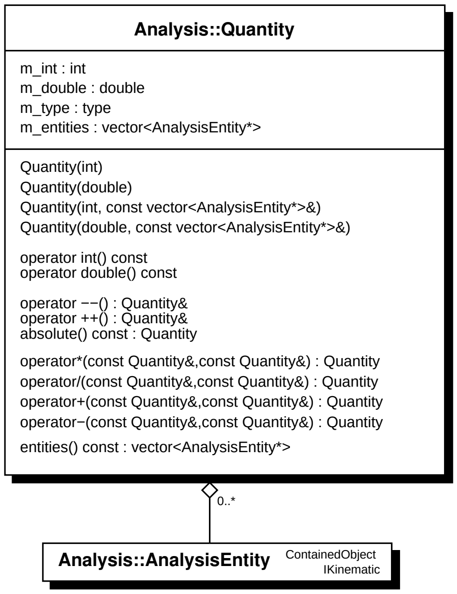


Figure 5.4: *The Quantity Class*

5.2.3 Entity Selection

As mentioned above, variables will often execute the same operation for all AnalysisEntities. Calculating transverse momentum is exactly the same whether one is concerned with leptons, high pT jets, or just electrons within a pseudorapidity range.

The IEntitySelector interface provides a single method which, given the AnalysisEvent, returns a container of AnalysisEntity objects. All classes which honour the IEntitySelector interface can therefore be used, via this method to extract entities from the event, without the client ever having to know explicitly which type of selector it is using. This is an example of the filter pattern described in section 3.2.3.

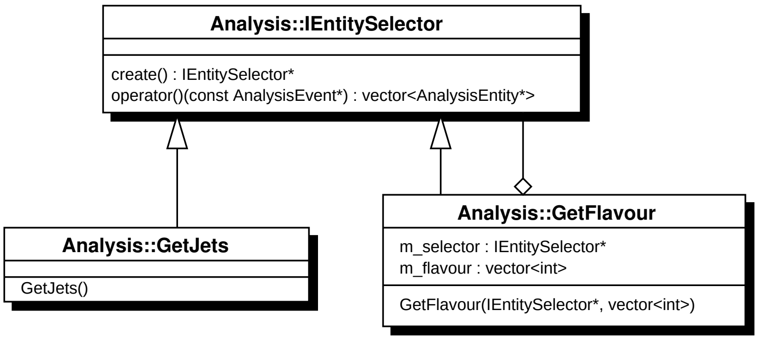


Figure 5.5: *Example Selector*

There exist a number of selector class which inherit from the IEntitySelector interface. Figure 5.5 shows an example of two concrete implementations.

GetJets simply implements the interface method, extracting the jets from the AnalysisEvent. GetFlavour is more complicated, aggregating an IEntitySelector which is used to extract AnalysisEntities; these are then filtered according to their flavour.

The IEntitySelector interface is developed further for use by its clients, the AnalysisVariable objects. Figure 5.6 shows the extended inheritance tree with two concrete implementations of the IEntityID interface, EntityID and EntityExtractor. A complete list of IEntitySelector objects is given in appendix B.1.

Entity ID

The EntityID class is developed further, where the AnalysisEntity objects selected do not just provide the method for Entity retrieval, but can also return a short description of themselves. This is useful when analysis information is displayed for view by the user.

The AnalysisVariables are set at run-time by the user, so the EntityID

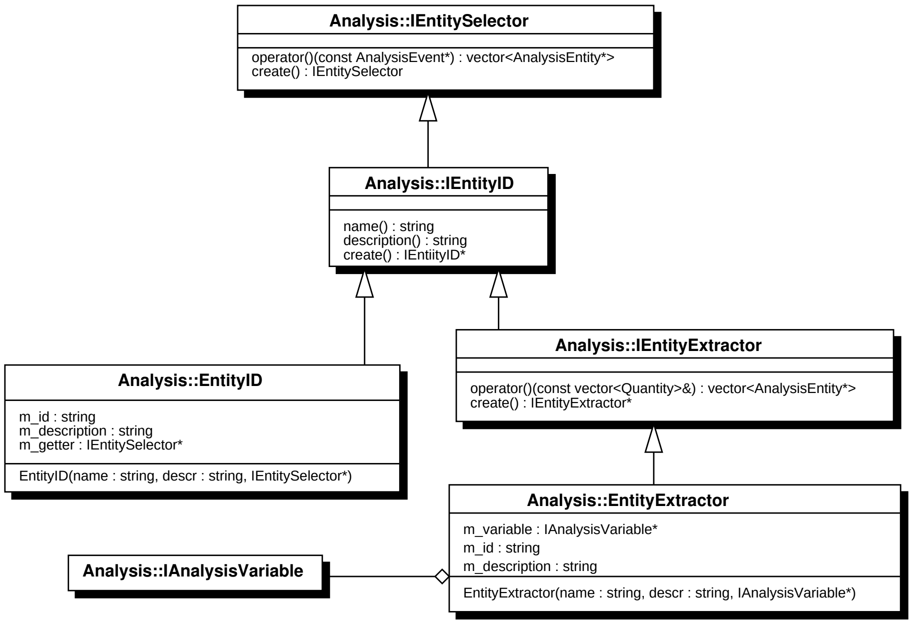


Figure 5.6: *Two concrete implementations of the IEntityID interface.*

objects also have an exclusive string key by which they can be selected from a map by an EntityID factory.

Entity Extractor

The IEntityExtractor interface is developed for use in selecting 'filtered' AnalysisEntity objects. The concept is based on an IEntityID which can select a subset of entities, using an aggregated AnalysisVariable.

For example, an AnalysisVariable object may calculate the maximum pT of electrons; an EntityExtractor object would use this AnalysisVariable to return the electron with this maximum pT. This design allows arbitrarily complex AnalysisVariables to be created.

5.2.4 Analysis Variables

AnalysisVariables form the substance of any analysis. The analysis operations that a physicist performs, such as event rejection and histogramming, are operations on quantities which result from calculations of different variables.

An AnalysisVariable is conceptually made up of two distinct parts. The operation, i.e. calculating the transverse momentum, and the operand, i.e. the AnalysisEntity objects, such as the detected electrons. The selection of these entities from the event is relegated to the IEntityID classes.

A number of abstract base classes representing generic types of AnalysisVariable are developed, providing common functionality for concrete implementations. These variable base classes are shown in figure 5.7.

Unary Variables

Variables which are calculated from a single AnalysisEntity, such as transverse momentum, or impact parameter, are of type UnaryVariable. They are calculated from a single collection of AnalysisEntity objects, such as electrons. They therefore aggregate a single IEntityID object, which is responsible for retrieving the relevant AnalysisEntity objects from the event.

All UnaryVariable concrete implementations will aggregate a KinematicFunction object, which provides functionality for kinematic operations. In addition a string descriptor, constructed in part from the IEntityID description, is provided which is useful for user displays of results.

When the IAnalysisVariable `operator()` method is called, the variable retrieves the relevant AnalysisEntities via its IEntityID object. These are passed to method 5.1.

This method iterates over the contents of the AnalysisEntity container and supplies the iterator to method 5.2, using the exception try command. Exceptions which may arise from inappropriate method calling of the AnalysisEntity object are therefore handled here so as not to be repeated in all concrete implementations.

$$\textsf{vector}<\textsf{Quantity}>\ \textsf{calculate(const vector}<\textsf{AnalysisEntity*}>\textsf{\&) const} \tag{5.1}$$

$$\textsf{Quantity calculateQuantity(vector}<\textsf{AnalysisEntity*}>\textsf{::const\_iterator) const} \tag{5.2}$$

Any classes which extend the UnaryVariable base class need therefore only implement the create() method, and method 5.2 in order to perform their function. If an exception is thrown when calculating the variable, a warning message is displayed and an empty Quantity container is returned.

Pair Variables

Variables which are calculated from a pair of AnalysisEntity objects, such as pseudorapidity separation, or invariant mass, are of type PairVariable. They are calculated from either a single collection or a pair of collections of AnalysisEntity objects, such as simply electrons, or for example electrons and jets.

The PairVariable class aggregates a pair of IEntityID objects, which enables the variable to choose two different types of AnalysisEntity objects. Like UnaryVariables, PairVariables also aggregate a KinematicFunction object and a string descriptor.

When the IAnalysisVariable operator() method is called, the variable retrieves the relevant AnalysisEntities from one or both of the IEntityID member variables. These are passed either to method 5.3 or method 5.4.

Method 5.3 pairs the AnalysisEntity objects with no double counting, and each pair of iterators is passed to the abstract method 5.5 using the exception try command.

Method 5.4 pairs each AnalysisEntity from the first container with each AnalysisEntity from the second container, but not with each other. The pairs of iterators are then passed to the abstract method 5.5 using the exception try command.

Exceptions which may arise from inappropriate method calling of the AnalysisEntity object are therefore handled here so as not to be repeated in all concrete implementations.

$$\textsf{vector<Quantity> calculate(const vector<AnalysisEntity*>\&) const} \tag{5.3}$$

$$\begin{aligned} &\textsf{vector<Quantity> calculate(const vector<AnalysisEntity*>\& ,} \\ &\qquad\qquad \textsf{const vector<AnalysisEntity*>\&) const} \end{aligned} \tag{5.4}$$

$$\begin{aligned} &\textsf{Quantity calculateQuantity(vector<AnalysisEntity*>::const\_iterator,} \\ &\qquad\qquad \textsf{vector<AnalysisEntity*>::const\_iterator) const} \end{aligned} \tag{5.5}$$

Any classes which extend the PairVariable base class need only implement the create() method, and method 5.5 in order to function. If an exception is thrown when calculating the variable, a warning message is displayed and an empty container is returned.

Event Variables

There also exist variables which do not calculate themselves from AnalysisEntity objects, but rather from the event itself. Examples are missing transverse momentum, or sphericity. These are of type EventVariable.

Extended classes of EventVariable must implement the create() method and the operator() method in order to function.

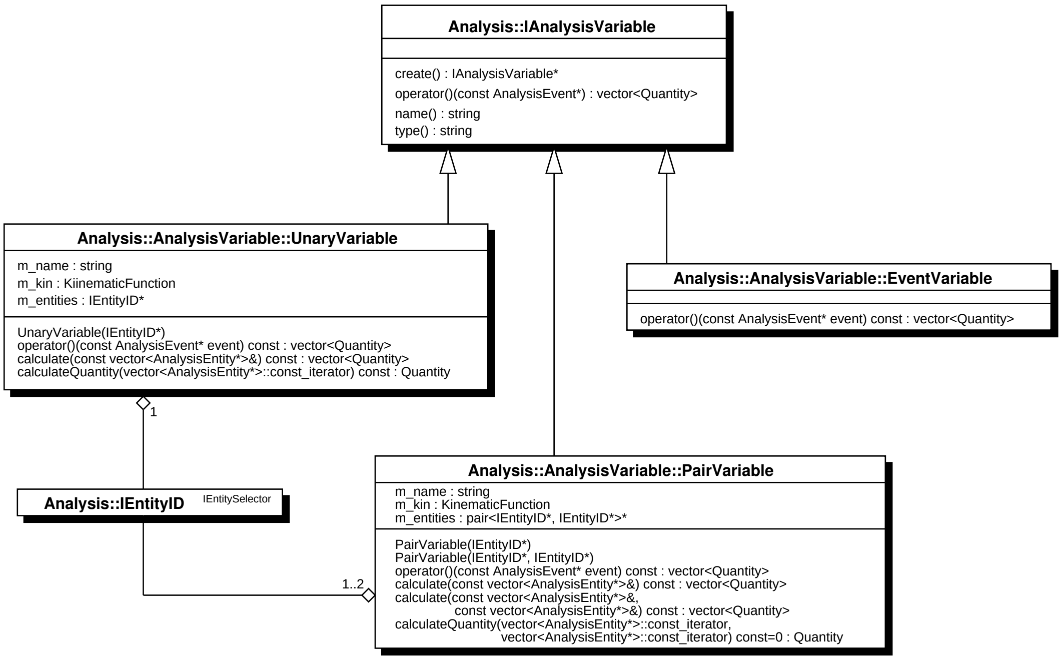


Figure 5.7: *The AnalysisVariable Base Classes*

Variable Operators

In many analyses more than just simple calculated quantities are required. Useful operations such as finding the highest jet momentum, or finding the invariant mass closest to the Z peak of a pair of leptons, are very important. IAnalysisVariable objects which perform these functional tasks are called VariableOperators and come in two flavours; UnaryOperators and Binary-Operators. These are illustrated in figure 5.8.

Unary Operators

The UnaryOperator class is designed for operations which are dependant solely on one set of Quantities, i.e. they can be calculated without reference to any other objects or quantities. Examples are the maximum quantity, or the absolute value of quantities.

UnaryOperators aggregate an IAnalysisVariable object is aggregated, which calculates the Quantity objects to be operated on, and a string descriptor for result documentation purposes.

The operator() method uses the aggregated variable to retrieve a container of Quantity objects. These are then passed to the operate() method which performs the operation. Derived classes of UnaryOperator need only implement the create() method and operate() method in order to perform their function.

Binary Operators

The BinaryOperator class is designed for operations which are calculated from more than one set of values, i.e. the operation is dependant on other values. Examples are the product of two quantities, or selecting a quantity closest to another value.

BinaryOperators aggregate an IAnalysisVariable which represents the left-hand side of the operation. A container of IAnalysisVariables represents the dependencies; the right-hand side of the operation.

The operator() method uses all aggregated variables to retrieve two containers of Quantity objects; the operands and the dependencies, which it then passes to the operate() method. Derived classes of BinaryOperator need only implement the create() method and the operate() method in order to perform their function.

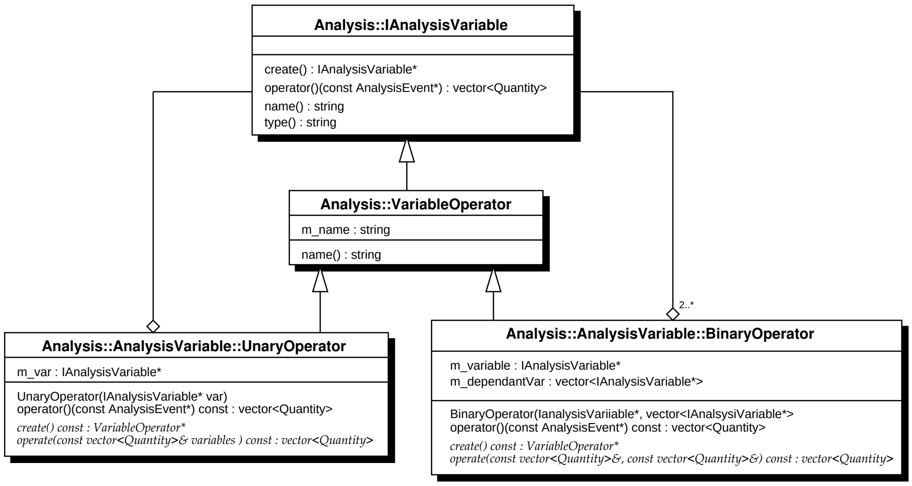


Figure 5.8: *The VariableOperator Base Classes*

5.2.5 Cuts

Cuts in a physics analysis are based on an event either passing or failing, depending on whether a specific event variable falls within(or without) a certain range or not. The ICut pure abstract base class is extended by a number of concrete implementations. The base class SimpleCut provides common functionality for a number of classes shown in figure 5.9. This ABC aggregates an IAnalysisVariable which is the cut variable, and an IAnalysisVariable which acts as the condition for the cut.

The concrete classes fall into two broad categories, InRange cuts and OutsideRange cuts. At instantiation they are supplied with a cut variable, and two variables representing the low and high edge of the cut. InRange or OutsideRange variable operators, described in appendix B.2, are created from them and passed to the simple cut object. These cuts return a pass or fail depending on whether the variable in question falls within or without a given range.

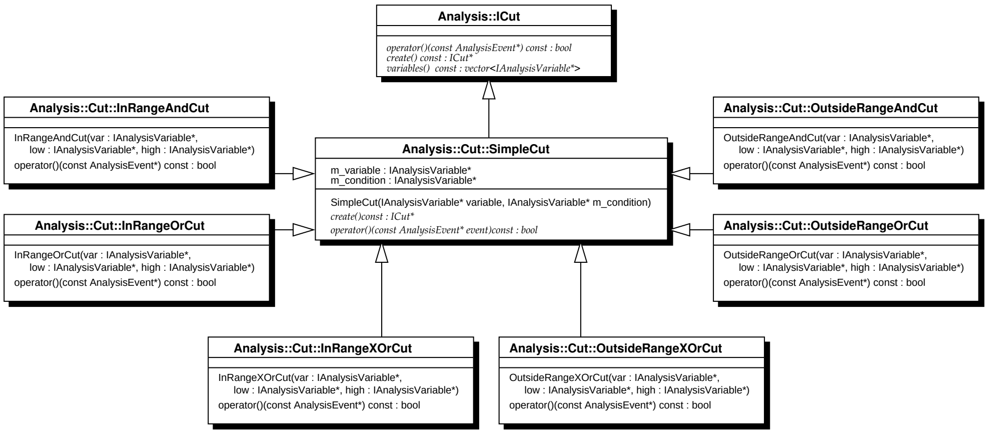


Figure 5.9: *The SimpleCut classes*

The test variables will often calculate a number of quantities, so each of these two cut categories are subdivided into booleans. For example, if an InRangeAndCut instance were created with an AnalysisVariable, Jet pT and given a range, 0 to 80, the cut would only return a pass if all the Jets had a momentum in this range. Similarly, an OR cut would return a pass if any one Jet satisified this condition, and the XOR cut would pass if only one were to fall into this range.

The ranges for these cuts may not only be constant values, but also variables themselves, such as minimum to maximum jet eta. The cut condition is also clearly not limited to InRange, or OutsideRange objects and could be extended for any VariableOperator.

Multiple Cutting

A cut may also depend on more than one condition, for example, particles must be in a momentum and eta range, or an event must have two leptons or two b-jets in order to be accepted.

Figure 5.10 shows the Multi-Cut classes. They aggregate a container of

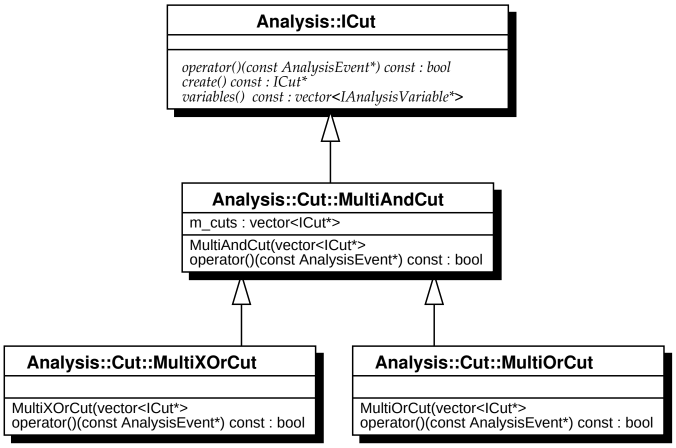


Figure 5.10: *The MultiCut classes*

any number of ICut objects, and also come in boolean varieties. The AND cut is passed if all constituent cuts are passed, the OR cut if any one of the cuts are passed, and the XOR cut if only one of the constituent cuts is passed.

All cuts, including multi-cuts, honour the ICut interface. This means that any complicated arrangement of cuts can be formed using these classes. This is an example of the filter pattern described in section 3.2.3.

AnalysisCuts

Cuts in an analysis involve more than just accepting or rejecting events; the physicist is concerned with efficiencies, the rate at which events pass a cut. Therefore a cut must also involve some sort of internal book-keeping.

The AnalysisCut class is shown in figure 5.11. It honours the ICut interface and aggregates an ICut class which is responsible for testing the event. It also aggregates a helper class, CutEfficiency, which keeps track of the pass rate.

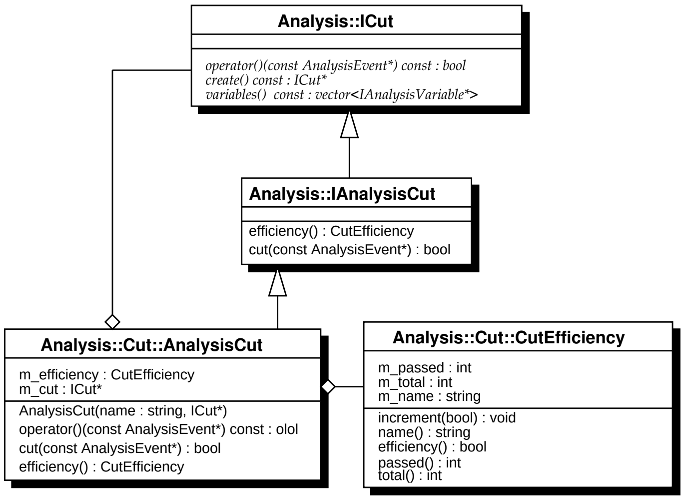


Figure 5.11: *The AnalysisCut class*

The operator() method is a const(it cannot change the state of the class), and can therefore not be used for book-keeping. Another non-const method is provided which tests the cut and updates the CutEfficiency.

5.2.6 Histograms

Histogramming quantities in an analysis is a simple procedure of calculating a variable quantity and filling pre-defined bins with their values. All histograms, irrespective of their dimension should be able to fill themselves from an event.

The two concrete implementations are AnalysisHistogram1D and AnalysisHistogram2D, for one and two dimensional binning respectively, shown in figure 5.12. The histograms aggregate at least one AnalysisVariable and an ICut object which is used to filter the events. If the event passes the filter cut, the histogram is filled by the AnalysisVariable.

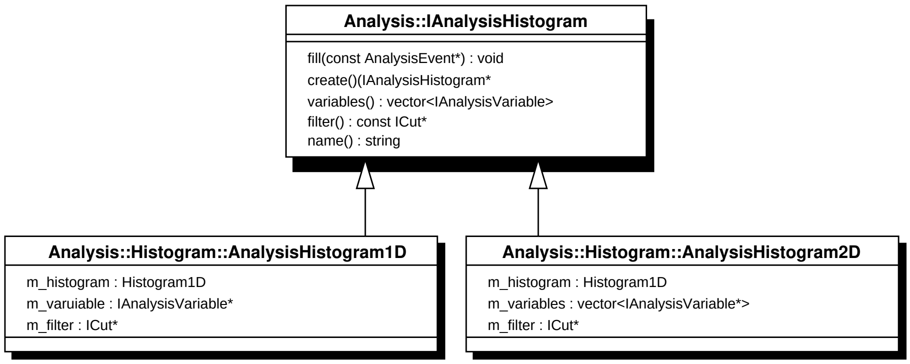


Figure 5.12: *The AnalysisHistogram classes*

The AnalysisHistogram1D class is instantiated with a single IAnalysisVariable and an ICut filter. The AnalysisHistogram2D is instantiated with one or two IAnalysisVariable objects and an ICut filter. Depending on the number of AnalysisVariables, the variable is plotted against itself in pairs or each variable is plotted against the other.

5.3 Analysis Execution

A number of Athena Algorithms and Manager classes exist which perform the execution of the analysis, controlling the input of fast simulation data, the application of cuts, the plotting of histograms, and the output of analysis results in a useful format for the analyst.

5.3.1 The Athena Algorithms

The Athena Algorithms are the main function objects scheduled by the framework to perform the analysis. They are created, initialised, and executed by the framework. They also have run-time steering properties that can be set in job options files.

The AtlfastAnalysis package has four Athena Algorithms; AtlfastReader, StandardNtupleReader, TrackNtupleReader, and OOAnalyser, only a subset of which are scheduled in a single job.

The AtlfastReader algorithm is responsible for retrieving Atlfast output objects from the TES, creating AnalysisEntity objects from them and storing these on the TDS for the OOAnalyser to use. The locations of the input and output particles are specified in the job options parameters.

The StandardNtupleReader algorithm is responsible for reading an HBOOK ntuple file, creating AnalysisEntity objects from the ntuple data and storing these on the TES for the OOAnalyser to use. The locations of the input and output particles are specified in the job options parameters. The TrackNtupleMaker is a similar algorithm for reading the Atlfast track ntuple.

The OOAnalyser is the main analysis class which creates the AnalysisEvent, and controls the variable monitor, cut manager, and histogram manager classes.

OOAnalyser

The OOAnalyser Algorithm is the main Algorithm which runs the analysis. It reads the AnalysisEntity objects from the TES and creates the AnalysisEvent object, and aggregates all the manager classes responsible for making the Signature object, monitoring variables, applying cuts and plotting histograms.

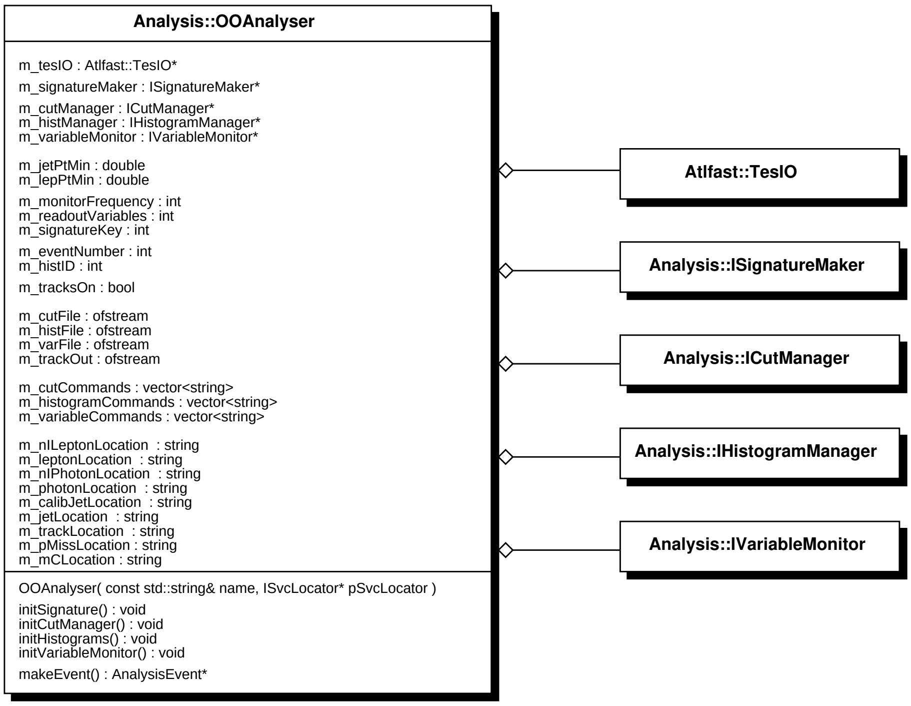


Figure 5.13: *The OOAnalyser Algorithm*

At initialisation SignatureMaker, CutManager, HistogramManager, and VariableMonitor classes are instantiated. These service classes are responsible for the main functions of the analysis suite. Cuts, histograms, and variables for monitoring are set at run-time via a string command job options mechanism, described in section 5.4.2.

In order to create the AnalysisEvent, the AnalysisEntity objects are read from the TES from the locations set in the job options file. Once this is created, it is passed to the individual service classes which perform the analysis based on the event data.

The analysis steering parameters define the locations of the AnalysisEntity objects in the TES, as well as defining the cuts, histograms and variables to be monitored. The number of events which are monitored is set here as well as whether tracks are to be analysed or not. Job options parameters and detailed sequence diagrams for the OOAnalyser Algorithm are shown in appendix B.3.

5.3.2 The Manager Classes

The manager classes are responsible for the event by event processing of the analysis. AnalysisVariables, Cuts and Histograms are all initialised, organised and monitored by the following function objects.

SignatureMaker

The SignatureMaker is responsible for creating the Signature object for each event. This is a user defined class, as the signature is specific to individual analyses. An ISignatureMaker interface is provided which is aggregated by the OOAnalyser to allow any SignatureMaker classes to be selected at run-time.

The class diagram is shown in figure 5.14. The `operator()` method is called by the OOAnalyser and is overridden by concrete instances in order to create the Signature object, which is then added to the AnalysisEvent. Concrete SignatureMaker classes will generally aggregate IEntitySelector objects in order to extract the relevant entities from the analysis event with which to construct the Signature object.

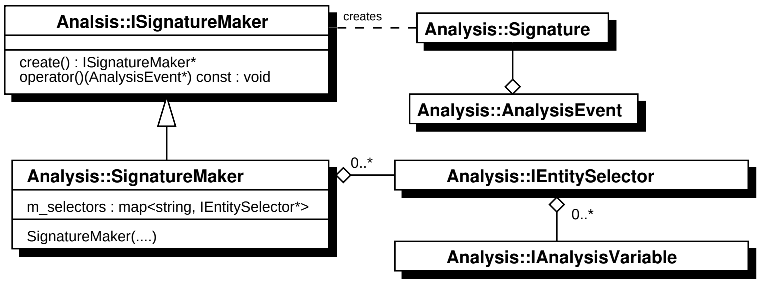


Figure 5.14: *The SignatureMaker classes*

Cut Management

The CutManager classes provide all the functionality for management of the analysis cuts, from initialising the cuts, applying the cuts to each event and displaying the results at the end of an analysis. Figure 5.15 shows the class diagram for the CutManager classes.

Most of the functionality is contained in the CutManager class. It aggregates a container of IAnalysisCut objects, and a container of CutEfficiencies. The individual cuts are responsible for their own cut statistics, while the CutManager's CutEfficiency objects keep track of the combined statistics.

The UserCutManager constructor creates a CutReader object which creates IAnalysisCut objects from a string command vector, read in from a job options file(see 5.4.2). An appropriate amount of ensemble CutEfficiency objects are then created, and the job options cut settings are output to the screen for debugging purposes.

The CutManager class implements a number of methods for applying its set of cuts. The applyCuts method applies the cuts, updating each cut's efficiency statistics and modifying the CutManager ensemble statistics accord-

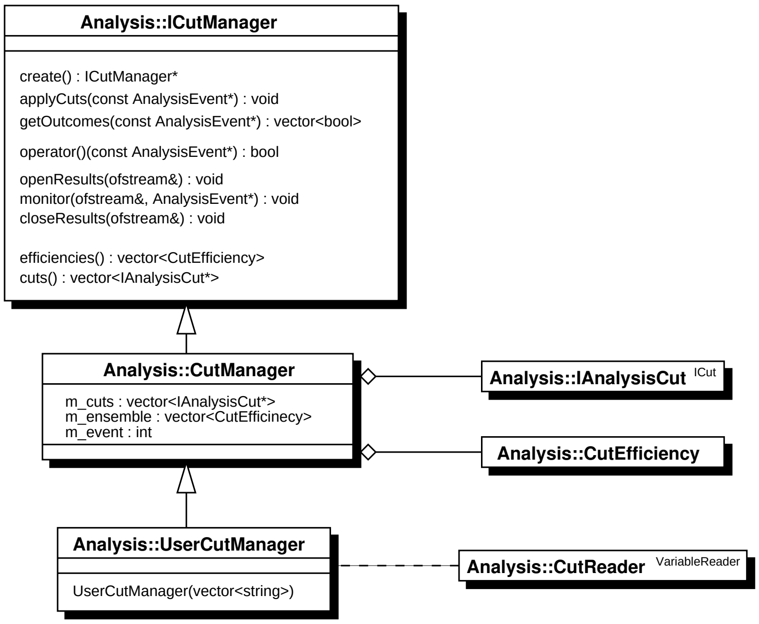


Figure 5.15: *The CutManager classes*

ingly. The operator() and getOutcomes methods are 'try it' style methods which leave the statistics unaffected and return either an event outcome, or individual outcomes for each cut.

The OOAnalyser algorithm calls the monitor method which outputs event information to an HTML file, including cut variable values and whether the event passed or failed each cut. Appendix B.3 shows the CutManager execution sequence diagram.

Histogram Management

The HistogramManager classes provide all the functionality for management of the analysis histograms in a similar way to the CutManager. Figure 5.16 shows the class diagram for the HistogramManager classes.

Again, most of the funcionality is contained in the HistogramManager

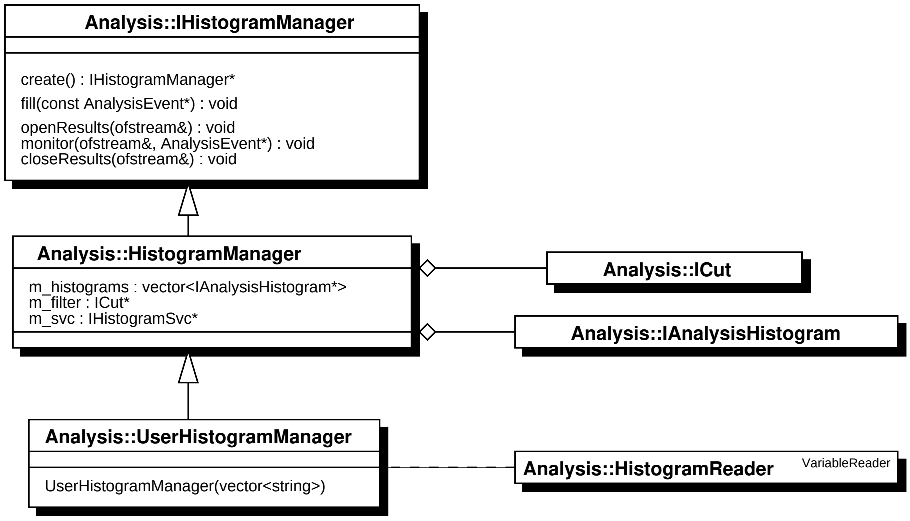


Figure 5.16: *The HistogramManager class*

class. It aggregates a container of IAnalysisHistogram objects and an ICut concrete instance for filtering. It also aggregates an Athena IHistogramSvc pointer, which allows access to the histogram services.

The individual histograms each have an event filter, while the Histogram-Manager's cut acts as a global filter for all histograms. The histograms are filled using the `fill` method, while the `monitor` method prints out event by event information to an HTML file, including histogram values and filter information.

The UserHistogramManager class implements a constructor which fills the container of IAnalysisHistogram objects from a string command vector read in from a job options file(see 5.4.2). Appendix B.3 shows the CutManager execution sequence diagram.

Variable Monitoring

Figure 5.17 shows the VariableMonitor classes responsible for displaying AnalysisVariable values for examination by the user.

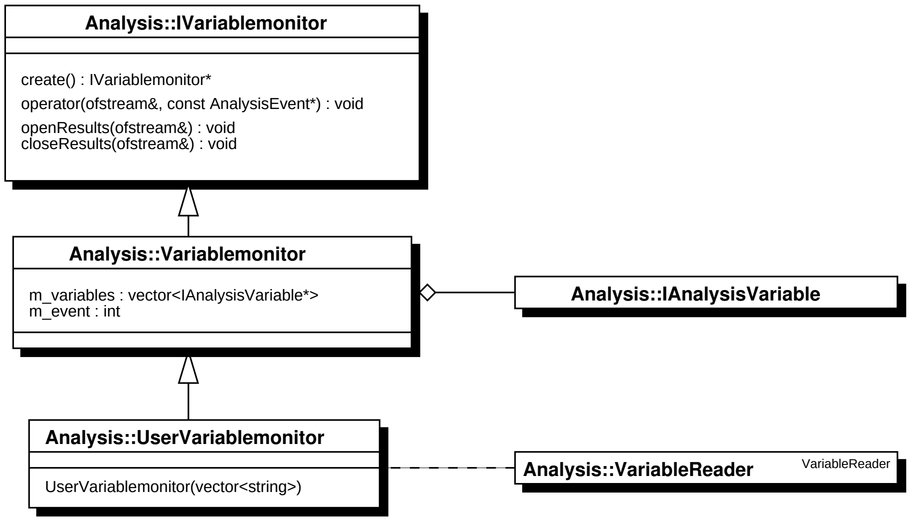


Figure 5.17: *The VariableMonitor class*

It is a simple class, aggregating a container of IAnalysisVariable objects, which prints out AnalysisVariable quantities event by event to an HTML file via the `operator()` method.

The UserVariableMonitor class simply implements a constructor which fills the container of IAnalysisVariable objects from a string command vector read in from a job options file(see 5.4.2).

5.4 Adapting the Analysis

The AtlfastAnalysis package performs user-defined functionality in two ways; compiled code modifications and job options steering parameters.

5.4.1 Compiled Modifications

There are several areas in which one can adapt the AtlfastAnalysis package to perform the analysis required by a user. The user adaptable classes are gathered in a user directory and subdivided into three parts, ObjectSelection, Signature and UserVariables.

The adaptation of the Signature object involves three steps. The Signature class must be defined and the SignatureMaker must be modified to construct this Signature from the event. New IEntitySelector classes must also be defined to extract Signature entities, and new EntityIDs corresponding to these EntitySelectors should be created for use by Analysis Variables.

New AnalysisVariables can also be created. These should inheret from the provided variable base classes described in section 5.2.4. These new variables should be added to the list in the job options reader.

The repository version of AtlfastAnalysis contains a simple user set-up for example purposes, to illustrate the packages functionality. This example, and how to construct new analysis objects, is described in appendix B.3.1.

5.4.2 Analysis Attributes via Job Options

Once an appropriate Signature object and any specific AnalysisVariables have been defined, the analysis attributes can be completely controlled by the job options file read at runtime.

This facility is a powerful feature of the AtlfastAnalysis package, allowing considerable modification of the analysis via the job options file without the need for regular and time-consuming compilation of the software.

Cuts, Histograms and Monitored Variables are all initialised via job options commands. There are a number of classes which are responsible for interpreting these commands correctly and returning the appropriate objects to the manager classes.

Figure 5.18 show the ensemble of classes which read the options. VariableReader is the base class, responsible for creating the appropriate AnalysisVariables. CutReader and HistogramReader inherit this functionality to supply the Cuts and Histograms which they create with the relevant AnalysisVariables.

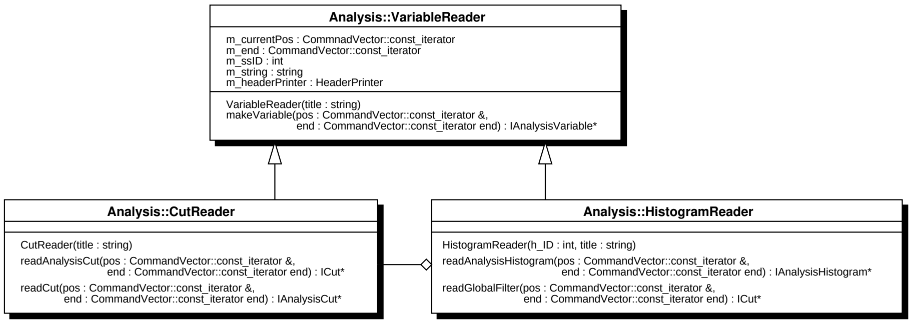


Figure 5.18: *The Job Options interpreters*

The Job Options themselves are a vector of string commands, which are parsed using a single white space as a delimiter. These string commands are read individually and interpreted according to a defined syntax, described in appendix B.2.1.

The VariableReader class contains a list of string identifiers for the AnalysisVariables. When reading a variable, it creates the appropriate IAnalysisVariable defined by this string and uses the singleton IDManager factory(described in the following section) to create the appropriate EntityID.

The IDManager

The IDManager is the class responsible for creating the IEntityID objects for use by AnalysisVariables. It is based on the factory and singleton design patterns, described in section 3.2.3.

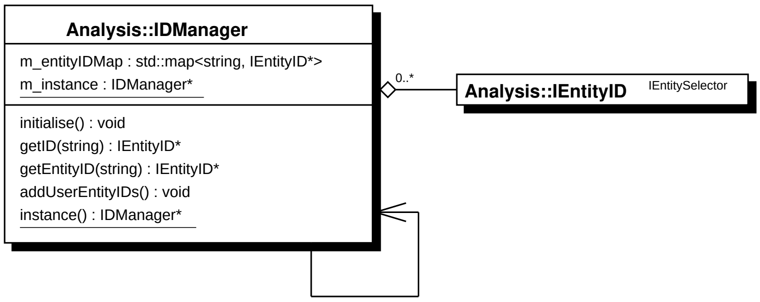


Figure 5.19: *The IDManager class*

IEntityIDs are used by IAnalysisVariable objects to extract AnalysisEntities from the event and are identified by unique string identifiers. The IDManager, when given a string identifier, creates the corresponding IEntityID object.

As shown in figure 5.19, the IDManager aggregates an STL map of the IEntityIDs, keyed on their string identities. When the singleton instance is retrieved, the map can be accessed using the two get methods.

These methods create a new IEntityID object corresponding to the supplied key. If the key is not found, the `getEntityID` returns a dummy IEntityID and outputs a warning message, while `getID` returns a NULL pointer, allowing a success test if a dummy return would not be suitable.

The map is filled when the IDManager's only instance is created and its contents are detailed in table B.2. The IDManager then calls a user-modifiable method to add further IEntityIDs to the map. Once the map is filled, the aggregated HeaderPrinter displays the contents of the map for user verification.

5.5 The Implementation of a Multi-Variate Likelihood Function

The implementation of a multi-variate likelihood function is described in this section, demonstrating the ability of the AtlfastAnalysis package to accomodate easily more complex methods of analysis. The use of this extension to improve the efficiency of a physics analysis is demonstrated in chapter 6.

5.5.1 The Uncorrelated Likelihood Method

The Likelihood Method uses the probability distributions of multiple variables to calculate a likelihood estimator with which to reject background. The probability of an event belonging to either signal or background is given by

$$\mathcal{P}(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i) \tag{5.6}$$

and is used to form a likelihood estimator

$$\mathcal{L} = \frac{P_{signal}(\mathbf{x})}{P_{signal}(\mathbf{x}) \ + \ P_{background}(\mathbf{x})} \tag{5.7}$$

The estimator is constructed from probabilities calculated from individual distribution functions, and therefore does not model any possible correlations between the variables.

5.5.2 The Projection-Correlation Approximation

The PCA method[26] enables an event likelihood estimator to be calculated which accounts for correlations between variables.

A Probability Distribution Function $\mathcal{P}(\mathbf{x})$ of n variables x_i is approximated using an n-dimensional Gaussian 5.8, described by an $n \times n$ covariance matrix V.

$$\mathcal{G}(\mathbf{y}) = (2\pi)^{-n/2}|V|^{1/2}\ exp(-\frac{1}{2}\mathbf{y}^{\mathbf{T}}V^{-1}\mathbf{y}) \tag{5.8}$$

The event variables, $\mathbf{x}$, are not in general Gaussian, so for the approximation to be valid, Gaussian parameter transformations of these variables are used. The transformation is given in equation 5.9, where erf^{-1} is the inverse error function, and F(x) is the cumulative distribution of x.

$$y(x) = \sqrt{2}\ erf^{-1}[2F(x) - 1] \tag{5.9}$$

The probability of an event belonging to either signal or background is given by equation 5.10 and can be used to form a likelihood estimator as shown previously in equation 5.7.

$$\mathcal{P}(\mathbf{x}) = |V|^{1/2}\ exp[-\frac{1}{2}\mathbf{y}^{\mathbf{T}}(V^{-1} - I)\mathbf{y}]\prod_{i=1}^{n} p_i(x_i) \tag{5.10}$$

The 'likesel'[27] program can be used to calculate correlation matrices and generate monte carlo probability distributions following the n-dimensional Gaussian from event variable distributions.

The AtlfastAnalysis program generates these distributions, and uses the resulting matrices and distributions to calculate likelihood efficiencies for background rejections.

5.5.3 Design and Integration

This section describes the modeling of the two likelihood methods within the design concept of the AtlfastAnalysis package, and how the calculations are incorporated into the execution process of a standard analysis.

Likelihood Variables

In order to implement both likelihood functions, the concept of a likelihood variable is introduced. This variable consists of a measured value per event as well as an associated probability distribution function of the variable. For each event therefore, the likelihood variable can supply its probability and cumulative distribution value.

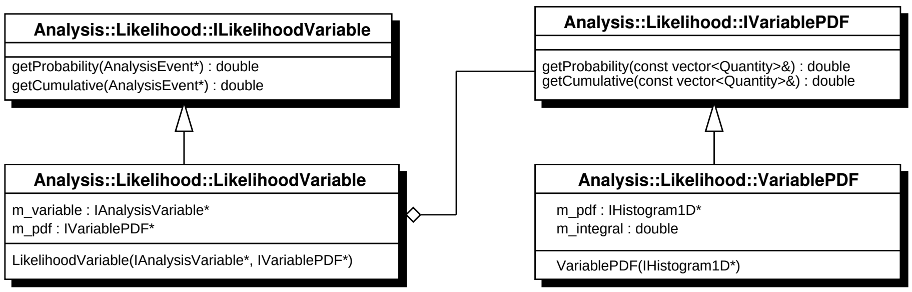


Figure 5.20: *The Likelihood Variable Classes*

The ILikelihoodVariable supplies the interface, shown in figure 5.20. The concrete instantiation aggregates an IAnalysisVariable which calculates the variable from the AnalysisEvent and an IVariablePDF instantiation which represents the variable's probability distribution.

The concrete VariablePDF class aggregates an IHistogram1D class, representing the variables PDF, from which it calculates the probability and cumulative distribution of the variable value. This PDF histogram is supplied at instantiation.

Likelihood Managers

The ILikelihood Manager interface is developed which provides a method to calculate the event likelihood, given an AnalysisEvent object.

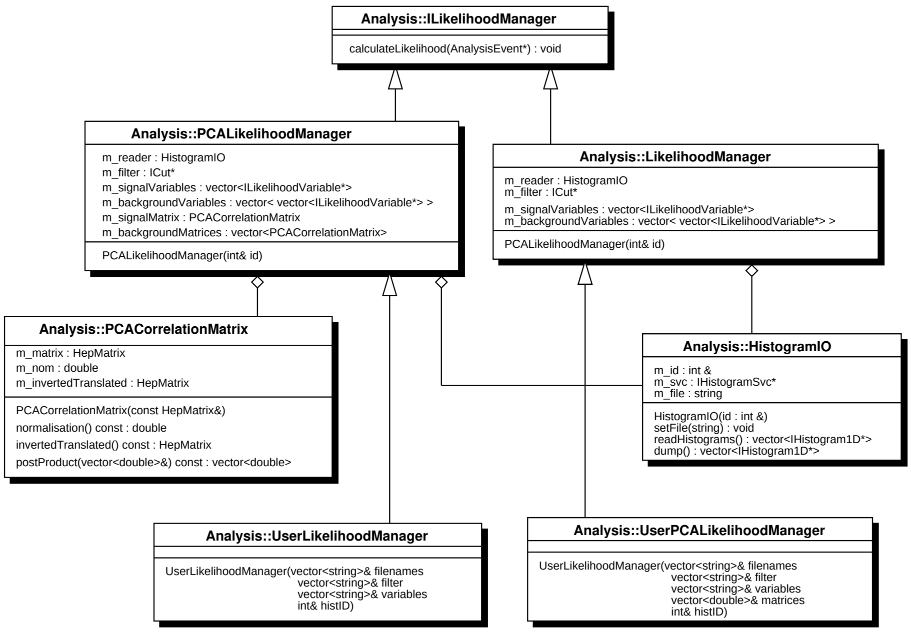


Figure 5.21: *The Likelihood Manager Classes*

Two concrete instantiations of the ILikelihoodManager exist, which provide both the uncorrelated, and PCA likelihood functionality. A HistogramIO service class is responsible for reading the PDF histogram data from file and booking the histograms in Athena.

Both of these intermediate Manager classes have a user extended class which is instantiated with string commands which specify the analysis variables, pdf histogram data files, and in the case of the PCA manager, the correlation matrix. The manager classes then create the relevant Likelihood-Variable objects from these data.

During the execution cycle, the manager is supplied with the AnalysisEvent, which it passes to its individual LikelihoodVariables. From the individual variable probabilities, the event likelihood value is calculated and added to the event signature.

The likelihood function is integrated into the OOAnalyser Algorithm. New job options parameters are added for the likelihood variable string commands, and the ILikelihoodManager is executed after the Signature is constructed. Initialisation and Execution sequence diagrams are shown in appendix B.4.

Chapter 6

Study of h $\rightarrow$ b$\bar{\text{b}}$ll via Weak Boson Fusion

This Chapter describes an analysis to determine the discovery potential of a Standard Model Higgs boson using the channel $\mathrm{h}_0 \rightarrow \mathrm{Z}^0\mathrm{Z}^{0*} \rightarrow b\bar{b}\ l^+l^-$. The $\mathrm{h}_0 \rightarrow \mathrm{W}^+\mathrm{W}^- \rightarrow e^{\pm}\bar{\mu}^{\mp}\ p_{Tmiss}$ channel study[28] is reproduced as a benchmark test for the simlulation and analysis software.

The Athena-Atlfast package was used to simulate detector response for all processes and the analysis was performed using the AtlfastAnalysis package described in the previous chapter.

6.1 The Higgs Discovery Potential at the LHC

Throughout the running lifetime of the four LEP experiments at CERN, 750 pb^{-1} of luminosity were accumulated, reaching a centre of mass energy of 206 GeV. Analysis of these data, using the Higgs search channel shown in figure 6.1, show within a confidence limit of 95% that $m_h >\ 114.4\,GeV$[29]. Precision fits to electroweak data predict a low Higgs mass of $m_h = 81^{+52}_{-33}\ GeV$,

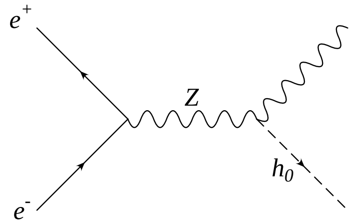


Figure 6.1: *The Higgs Search Channel used at the LEP experiments.*

therefore complete coverage of the low Higgs mass region is a primary goal at the LHC.

Table 6.1 shows the cross-sections of the three main Higgs production mechanisms at the LHC in the low to intermediate mass range. Gluon fusion is the dominant production mechanism over the entire region. The Higgs couples proportionally to the mass of particles, and therefore decays to the most massive accessible state as shown in figure 6.3

The large QCD backgrounds produced by initial and final state radiation from the gluon fusion process prohibit the low mass discovery channels involving quark final states, such as $H \to b\bar{b}, \tau^+\tau^-$. These decays are only viable in the context of associated Higgs production(see figure 6.1) where the associated weak boson(b) or one of the associated top quarks(c) decays leptonically, with a cross-section fifty times lower than the gluon fusion process.

$m_h(GeV)$	140	150	160	170	180	190
$\sigma_h(pb)$ gg	14.27	12.59	11.19	10.02	9.04	8.21
$\sigma_h(pb)$ WW	2.98	2.71	2.63	2.41	2.35	2.19
$\sigma_h(pb)$ ZZ	1.01	0.92	0.90	0.86	0.83	0.78

Table 6.1: *Higgs Production Cross-Sections (140-190 GeV)*

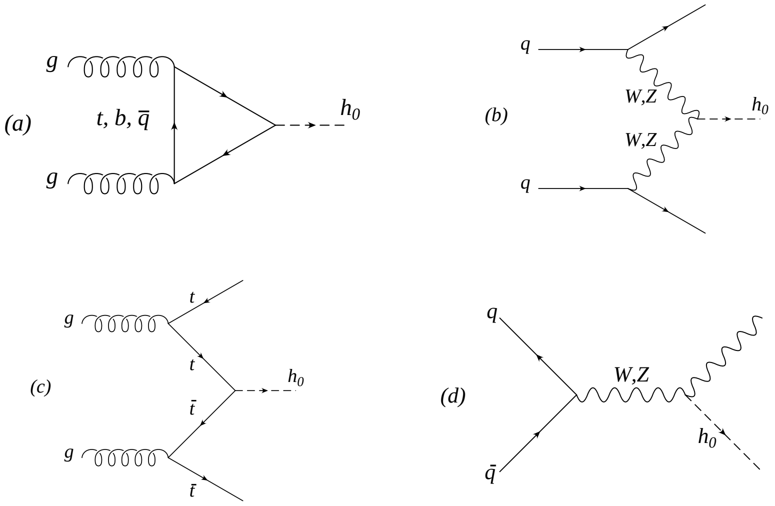


Figure 6.2: *Direct and Associated Higgs Production at the LHC: Gluon fusion(a), weak boson fusion(b), and production with associated top quarks(c) or weak bosons(d).*

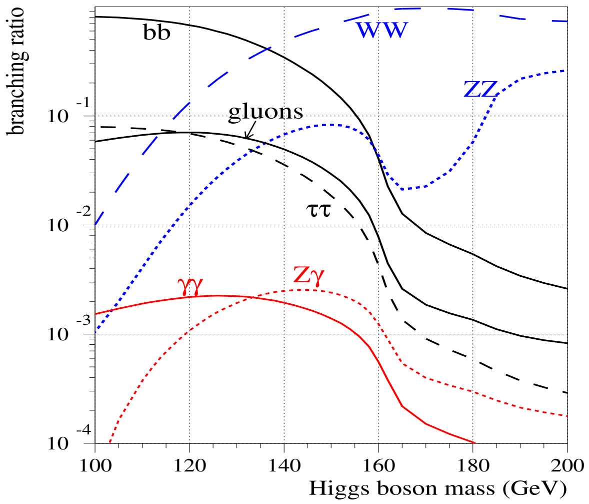


Figure 6.3: *Branching Ratios of the SM Higgs (100-700GeV).*

A number of Higgs physics channels have been investigated, covering the low, intermediate, and high mass ranges which are accessible to the LHC[30].These form the benchmark signatures for the ATLAS detector.

The low mass range extends the LEP2 mass limit up to 120 GeV in which $h \to \gamma\gamma$ and $h \to b\bar{b}$ are the most important decay channels. The $h \to \gamma\gamma$ channel is limited by the irreducible prompt $\gamma\gamma$ background and requires excellent energy and angular resolution in order to observe the narrow mass peak. For QCD background and the resonant background from $Z \to ee$ where $m_H \approx m_Z$, powerful particle identification capabilities are required.

Below $m_H \approx 180$ GeV, the $h \to ZZ^* \to 4l$ channel is limited due to the narrow width of the Higgs. The width of a particle represents the variance in its mass, so a narrow width means less probability of decay to heavier particles(e.g. a real Z^0). Although the signature is clean, the cross-section is low due to the virtual Z^0. The $h \to WW \to l\nu l\nu$ channel opens up at the $m_H \approx 160$ GeV weak boson resonance to supplement the discovery potential in the intermediate mass range, although the decay cannot be completely reconstructed due to the neutrinos.

In the high mass region, the 'gold-plated' $h \to ZZ \to 4l$ channel has a large signal and low background rate, and is considered the most reliable signal for discovering a Standard Model Higgs. It is limited at the top end of this mass range by a low production rate.

Weak boson fusion is an important production mechanism at higher Higgs masses, where its cross-section is comparable with the gluon fusion process. The features which distinguish vector boson fusion from other production mechanisms are the presence of two energetic forward jets in the final state and the lack of colour interchange between initial state quarks. This causes suppressed hadronic activity in the central rapidity region compared with the gluon fusion process, allowing the use of powerful jet vetos to reduce large QCD backgrounds.

The search is extended to $m_H \approx 1$ TeV by boson fusion channels such as $h \to ZZ \to ll\nu\bar{\nu}$, $h \to WW \to l\nu jj$, and $h \to ZZ \to lljj$. These channels have cross-sections many times larger than the 'gold-plated' channel, but place strong requirements on the missing energy resolution due to escaping neutrinos in the first case, while calorimeter performance and jet-tagging are important in the latter.

Figure 6.4 shows the discovery potential of a standard model Higgs boson over the full mass range[30]. It can be seen that 5σ discovery is possible with 30 fb^{-1} of integrated luminosity using a number of combined discovery channels. On top of its importance in high mass Higgs discovery channels, recent low mass studies[28, 31] have shown that weak vector boson fusion production is a powerful supplement to the existing discovery potential in the low/intermediate mass range.

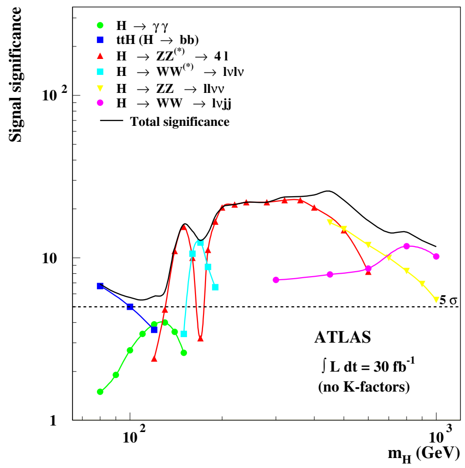


Figure 6.4: *ATLAS Discovery Potential of the Standard Model Higgs Boson at 30* fb^{-1} .

6.1.1 Weak Boson Fusion Backgrounds

The main backgrounds to weak Higgs boson decays in the two vector boson fusion channels $h_0 \to W^+W^- \to e^{\pm}\mu^{\mp}\ p_{Tmiss}$ and $h_0 \to Z^0Z^{0*} \to b\bar{b}\ l^+l^-$ studied in this thesis are shown in figure 6.5.

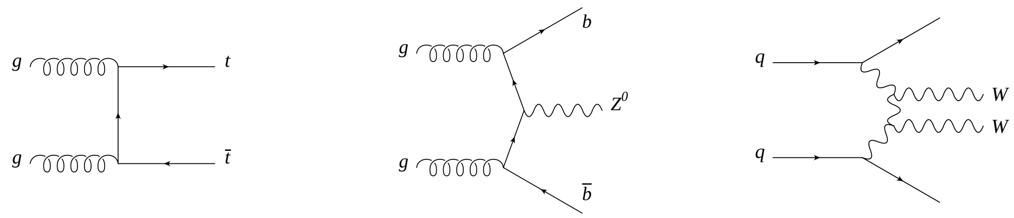


Figure 6.5: *The main backgrounds to the VBF Higgs processes*

The $t\bar{t}$ background is the most problematic due to the huge LHC cross-section for top production(1 nb). The top quark decays almost exclusively to a Wb pair, imitating the two lepton two jet signal for both $h \to l\nu l\nu$ and $h \to b\bar{b}ll$ channels. This background is reducible for the $h \to b\bar{b}ll$ signal due to the missing Z peak in the lepton invariant mass, and the elevated missing momentum.

The WW+jets background affects both channels, having a major impact on the $h_0 \to W^+W^- \to e^{\pm}\mu^{\mp}\ p_{Tmiss}$ signal. This background is much less problematic for the $h \to b\bar{b}ll$ signal due to the missing lepton mass peak, elevated missing momentum and reduced high pT jets in the event.

The $Z^0b\bar{b}$ channel is the most problematic background for the $h \to b\bar{b}ll$ signal due to the presence of the real Z. Reduction of this background will therefore rely heavily on forward tagging, jet vetos and good resolution of the b-jet mass peak.

The $t\bar{t}$ and WW+jets backgrounds were generated using the Pythia event generator[32], version 6.203 with the CTEQ5L structure functions, initial and final state radiation, fragmentation and decay, and multiple interactions turned on.

The $Z^0 b\bar{b}$ background is not implemented in the Pythia program, and so was simulated using the AcerMC[33] Monte Carlo generator which provides a library of massive matrix elements and phase space modules for the generation of this process. The hard process event is then completed by Pythia, which calculates the initial and final state radiation and hadronisation of the event.

Table 6.2 shows the cross-section of these background channels, exhibiting production rates of up to three orders of magnitude larger than the cross-sections in the range 10-100 fb for the weak boson fusion signals.

Background	$t\bar{t}$	WW+jets	$Z^0 b\bar{b}$
Cross-Section	22.8 pb	3.3 pb	65.8 pb

Table 6.2: *Background Cross-Sections*

6.2 Atlfast and AtlfastAnalysis validation

In order to test the functionality of both the Atlfast simulation and the AtlfastAnalysis package, a previous study of the channel $\mathrm{h}_0 \rightarrow \mathrm{W}^+\mathrm{W}^- \rightarrow e^{\pm}\bar{\mu}^{\mp}\ p_{Tmiss}$ was reproduced to test signal and one background. The same event selection criteria, and cuts were implemented in order to follow the benchmark study using both Athena-Atlfast and Fortran-Atlfast simulation.

6.2.1 Direct Comparison of Atlfast Programs

Events for both the Fortran and Athena simulation were generated using Pythia 6.203, with initial and final state radiation, fragmentation and decay, and multiple interactions turned on. Atlfast-Fortran version-00-02-21 and Atlfast-Athena version-00-00-10 were used with the same Pythia library and the standard Fortran pythia interface settings equal to produce the results for comparison. Both simulators produced identical results using the same generator random seed and simulation smearing switched off.

Figures 6.6 and 6.7 show the distributions of particles and jets from 100,000 $t\bar{t}$ events generated using both Fortran and Athena Atlfast with smearing activated. The distributions are compared using both chi-squared and Kolmogorov statistical tests, described in appendix C.

The multiplicities for particles are identical in each case as only the four-vectors are smeared. The transverse momenta with respect to the beamline and pseudorapidities show good agreement between smeared Fortran and Athena distributions. The jets are constructed from smeared cells which are grouped according to energy deposition. Therefore, differences in smearing are likely to produce differences in the number of jets reconstructed as well as their momenta and pseudorapidities. The missing momentum is constructed from the sum of transverse momenta of all particles which are not detected as isolated particles or do not contribute to reconstructed jet energies.

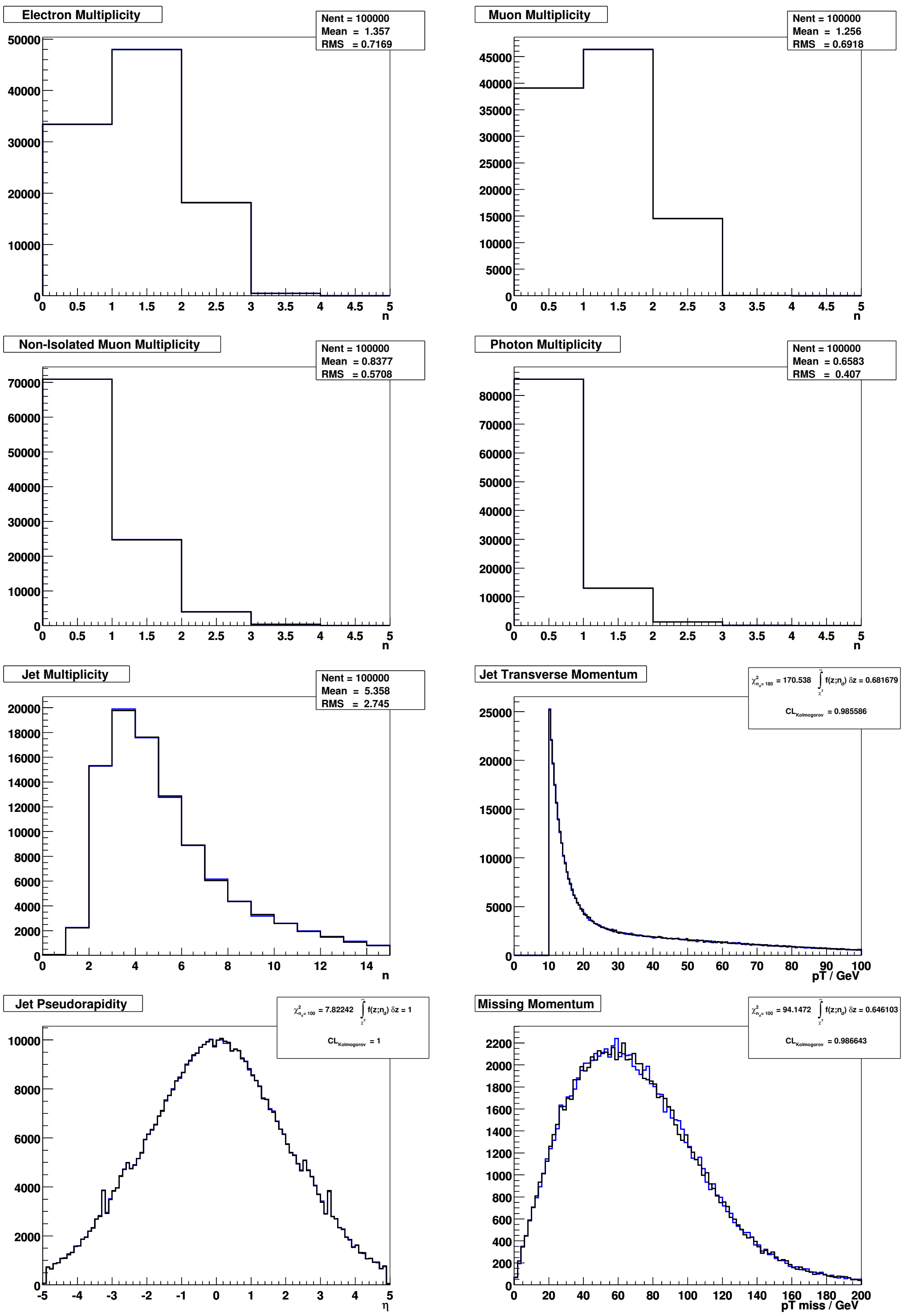


Figure 6.6: *Comparison of Atlfast particle, jet, and missing momentum distributions for 100,000 $t\bar{t}$ events generated using Fortran and Athena Atlfast with smearing activated. Athena distributions are shown in black, Fortran distributions are shown in blue.*

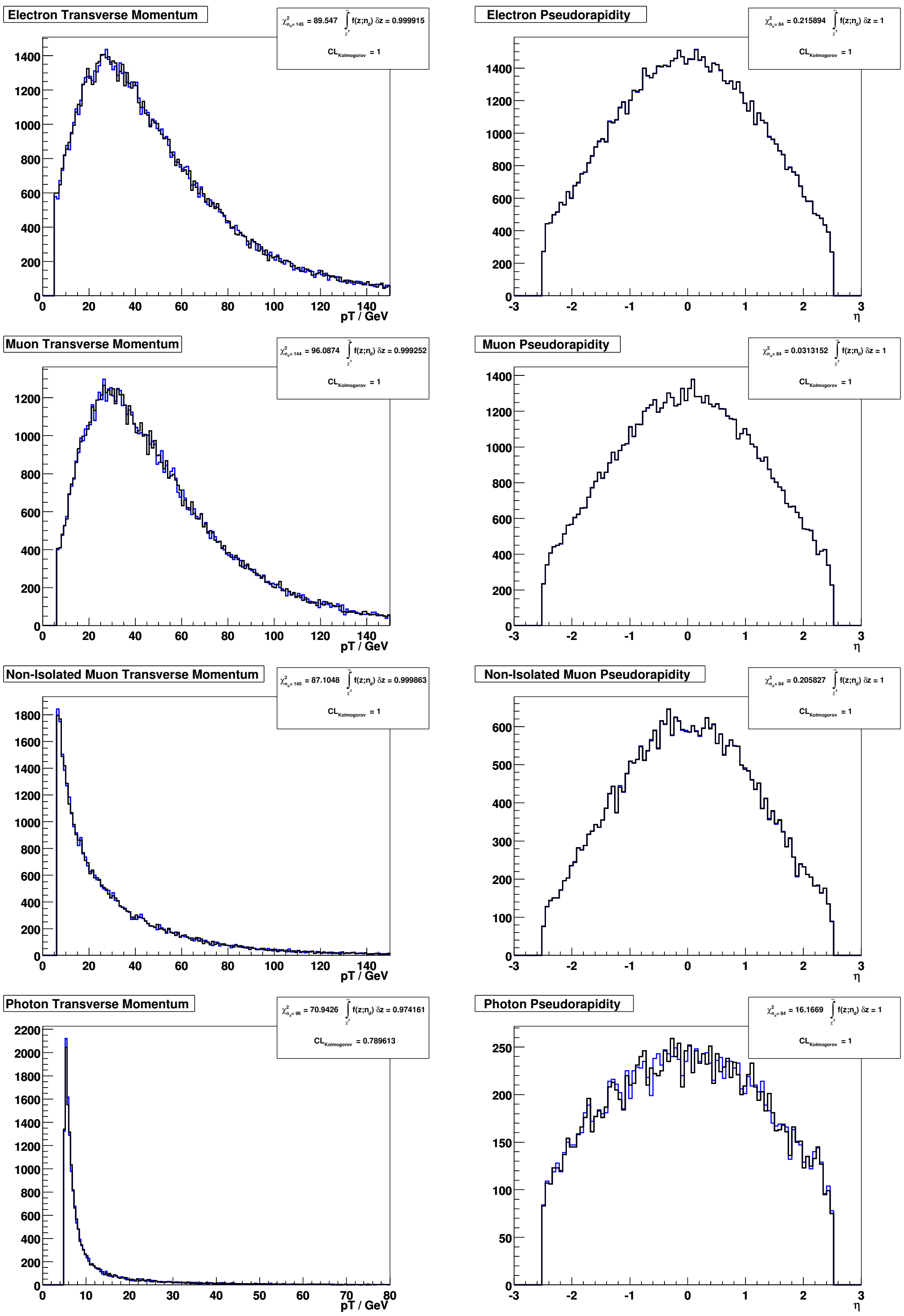


Figure 6.7: *Comparison of Atlfast particle transverse momenta and pseudorapidity distributions for 100,000 $t\bar{t}$ events generated using Fortran and Athena Atlfast with smearing activated. Athena distributions are shown in black, Fortran distributions are shown in blue.*

6.2.2 Analysis

The AtlfastAnalysis program was used to analyse 100,000 signal and $t\bar{t}$ background events from both Athena and Fortran ntuples, produced from identical Pythia events. Event selection and cut criteria are taken verbatim from the original study, and the original fortran analysis routine(courtesy of Rob Harper) was used to verify the cut statistics from the AtlfastAnalysis program using these ntuples.

6.2.3 Topology of $\mathrm{h}_0 \to \mathrm{W}^+\mathrm{W}^- \to \mathbf{e}^\pm\bar{\mu}^\mp$ $\mathbf{p_T}$

The $\mathrm{h}_0 \to \mathrm{W}^+\mathrm{W}^- \to e^\pm\bar{\mu}^\mp$ p_{Tmiss} process is shown in figure 6.8 and has the characteristic signature of weak vector boson fusion channels.

The two recoiling quarks from the emission of the weak bosons cause very high energy 'tagging' jets which are both forward in the detector and have high pT with respect to the beam line, and there is reduced central jet activity due to the lack of colour flow between initial and final states. The tagging jets, reduced central jet activity, and the spin-correlation of the bosons provide scope for a number of effective cuts and vetos.

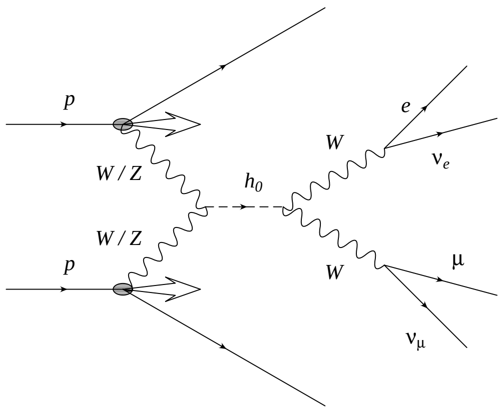


Figure 6.8: *The Higgs Production Channel.*

Event Selection and Cuts

To select events a number of basic acceptance criteria are defined. The two scattered quarks must be observed as forward tagging jets and an electron and muon must be observed. The following cuts must also be satisfied.

$$P_{T j,l} \geq 20.0 GeV; \eta_j \leq 5.0; \eta_{ll} \leq 2.5 \tag{6.1}$$

The tag jets are chosen as the two highest p_T jets in the event. If these requirements are met by an event, the cuts detailed in table 6.3 are applied. The cut results from both simulations are shown and compared in table 6.4.

The forward tagging cuts assert that the two tagging jets are indeed those created from the recoil quarks by demanding that they are in opposite hemispheres of the detector, with a large spacing in rapidity. The leptons must also be between these jets in pseudorapidity. Tagging jet distributions are shown in figure 6.9.

The tagging jet separation requirement is particularly effective at reducing the QCD background. It can be seen from the cut results that the application of the forward tagging requirements leads to a massive reduction of nearly 98% in the $t\bar{t}$ background while leaving over 55% of the signal.

The lack of gluon radiation in the initial state leads to suppressed hadronic activity in the central region, and applying a central jet veto reduces the QCD background by about half. The invariant mass of jets from background QCD events tend to be small compared with electroweak processes, therefore cutting on the invariant mass of the tag jets is also effective. The application of these two cuts reduces the $t\bar{t}$ background a further 95%, while leaving just under 50% of the signal.

Separation	$\Delta R_{jj} \geq 0.7; \Delta R_{lj} \geq 0.7$
Forward Tagging	$\Delta\eta_{tags} =\mid \eta_{tag1} - \eta_{tag2} \mid \geq 3.6$ $\eta_{tag1} \cdot \eta_{tag2} < 0.0$ $\eta^{min}_{tag} + 0.7 \leq \eta_{e,\mu} \leq \eta^{max}_{tag} - 0.7$
Central Jet Veto	$\eta^{min}_{tag} \geq \eta_j \geq \eta^{max}_{tag}; P_{Tj} \geq 20GeV$
Tag Jet Mass	$W_{jj} \geq 800GeV$
Lepton Angular	$\phi_{ll} \leq 95°; \cos\theta_{ll} \geq 0.2; \Delta R_{ll} \leq 1.8$ $W_{ll} \leq 90GeV; P_{Tlmax} \leq 120GeV$
Lepton Mass/PT	$W_{ll} \leq 90GeV; p^{ll}_T \leq 120GeV$
Real Tau Rejection	$x_{\tau 1}, x_{\tau 2} \geq 0.0; M_Z - 25 \geq W_{\tau\tau} \geq M_Z + 25GeV$

Table 6.3: *Analysis Cuts for* $h_0 \rightarrow W^+W^- \rightarrow e^{\pm}\mu^{\mp}\ p_{Tmiss}$ *Channel*

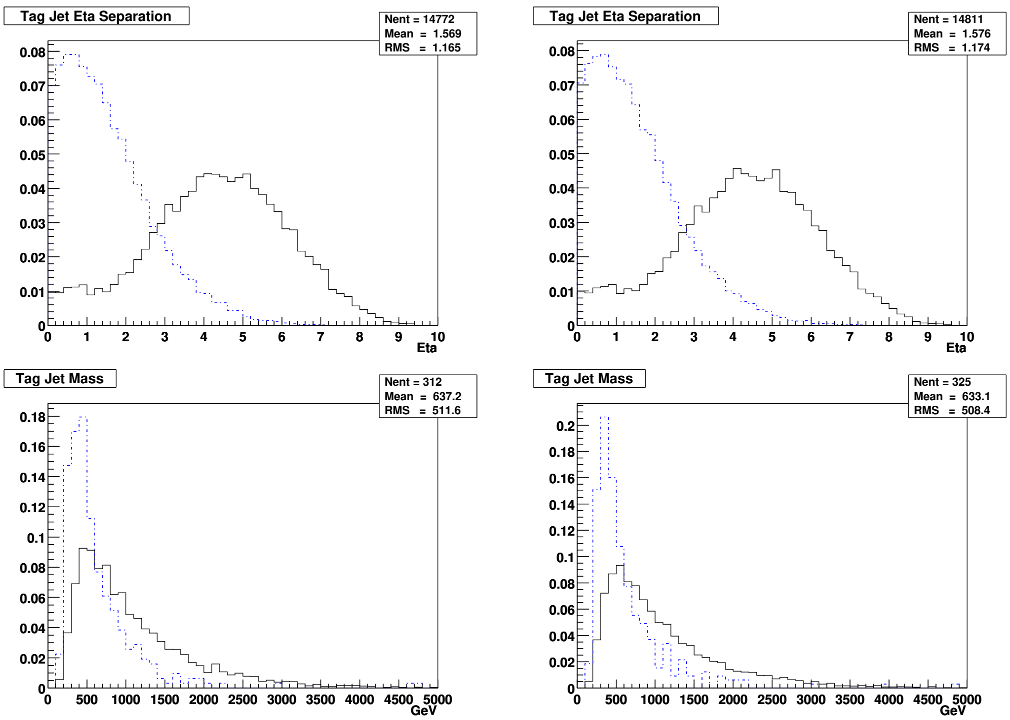


Figure 6.9: *Higgs signal and* $t\bar{t}$ *background pseudorapidity gap and invariant mass of forward 'tagging' jets. The Fortran distributions are on the left and Athena on the right, with signal shown in black and background in blue.*

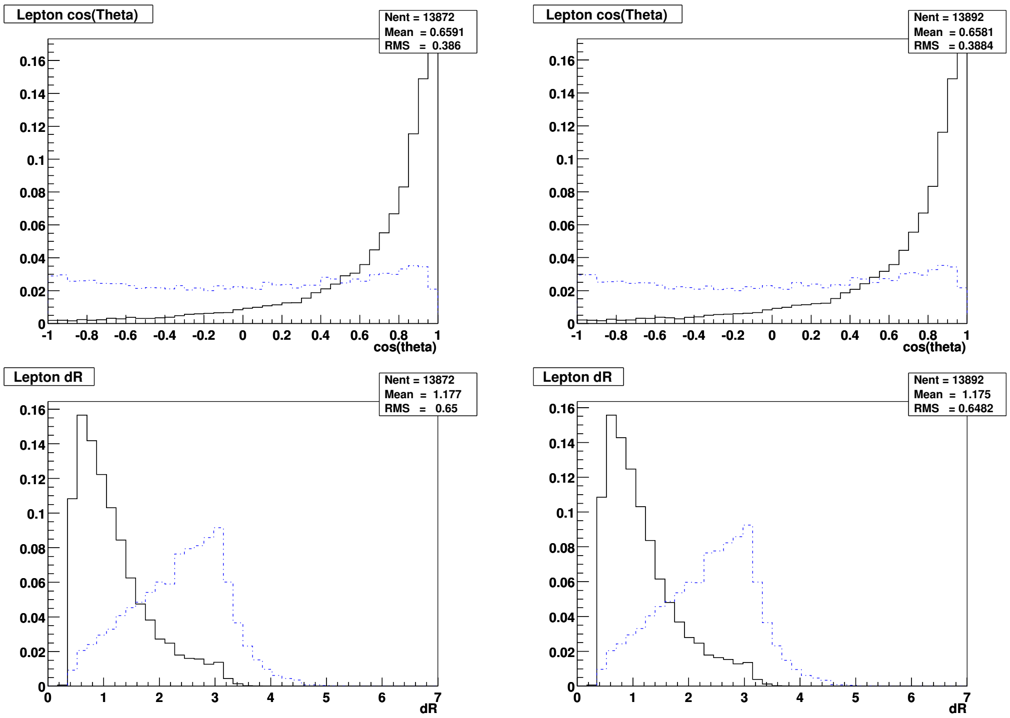


Figure 6.10: *Higgs signal and* $t\bar{t}$ *background lepton angular distributions. The first plots show the cosine of the separation angle* θ_{ll} *of the two leptons and the second plots show the separation in* R *where* $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$. *The Fortran distributions are on the left and Athena on the right, with signal shown in black and background in blue.*

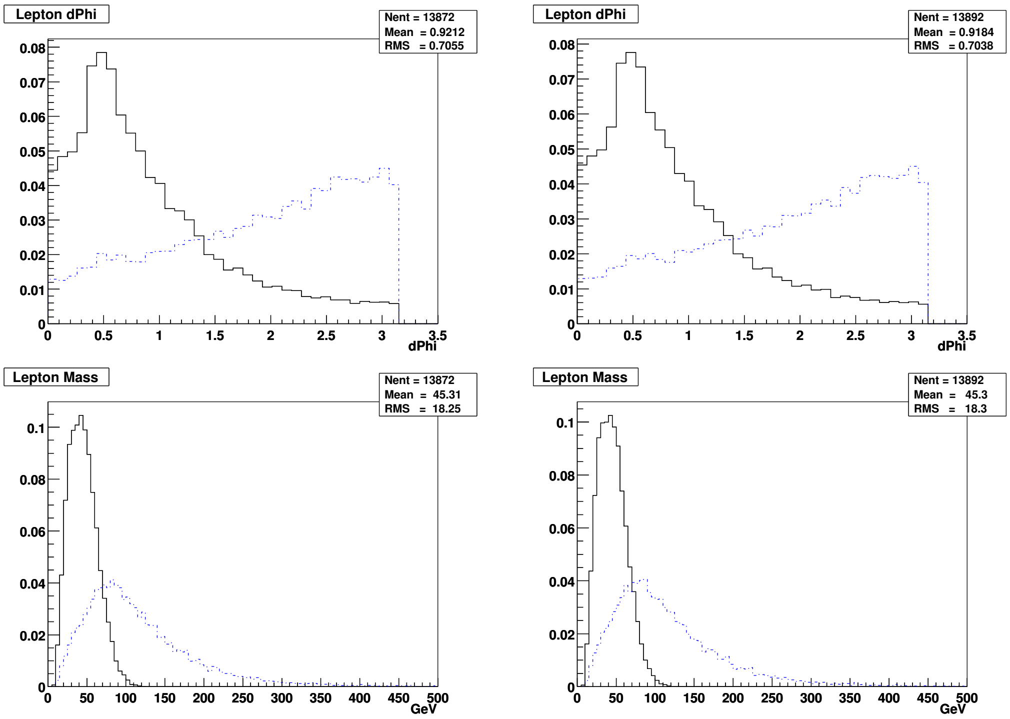


Figure 6.11: *Higgs signal and $t\bar{t}$ background lepton angular distributions. The first plots show the separation in azimuthal angle between the two leptons and the second plots show the lepton system invariant mass. The Fortran distributions are on the left and Athena on the right, with signal shown in black and background in blue.*

Due to the scalar nature of the SM Higgs, the spins of the W bosons are anti-correlated. In the Higgs frame, this causes the leptons to be emitted preferentially in the same direction, which is still observable in the lab frame(see figure 6.10). A number of lepton angular cuts are therefore used to further reduce background.

As the di-lepton system and neutrino system are emitted back-to-back in the Higgs frame, the invariant mass is restricted to $\frac{m_h}{2}$(see figure 6.11). A cut is therefore placed on the invariant mass of the visible leptons, and the maximum lepton transverse momentum is also restricted. These cuts reduce the background a further 10%.

The $\tau\tau$ background is not modeled as no generator interface exists to produce these events in the Athena framework. Despite this a 'real tau rejection' cut was included in order to fully test the variable calculation of the AtlfastAnalysis program. Using equation 6.2 describing the fraction of tau momenta taken by each lepton in an actual $\tau\tau$ event, it is possible to differentiate background from signal using $x_\tau^{1,2}$ and $W_{\tau\tau}$.

$$\begin{aligned} x_\mu &= \frac{D^T_{e\mu}}{D^T_{eP} + D^T_{e\mu}} \\ x_e &= \frac{e_{px}}{P_{px} + e_{px} + \mu_{px} - (\mu_{px}/x_\mu)} \qquad (6.2) \\ \text{where } D^T_{ab} &= a_{px}b_{py} - a_{py}b_{px} \\ W_{\tau\tau} &= W_{(x_e e \,+\, x_\mu \mu)} \end{aligned}$$

These cuts have little effect on the signal or $t\bar{t}$ background studied here, but would give large reductions in $\tau\tau$ background, as one of the tau fractions x tends to be negative for signal events.

No mass peak can be reconstructed from the observable signal, so the transverse mass must be used to extract information on the Higgs mass. M_T

is given in equation 6.3 where p_T^{ll} is the transverse momentum of the visible lepton system and $\Delta\phi$ is the azimuthal angle between the missing momentum vector and the visible lepton system. Figure 6.12 shows the transverse mass distributions of signal and background for Fortran and Athena Atlfast simulations.

$$M_T = \sqrt{2p_T^{ll}\, p_{Tmiss}(1 - cos\Delta\phi)} \tag{6.3}$$

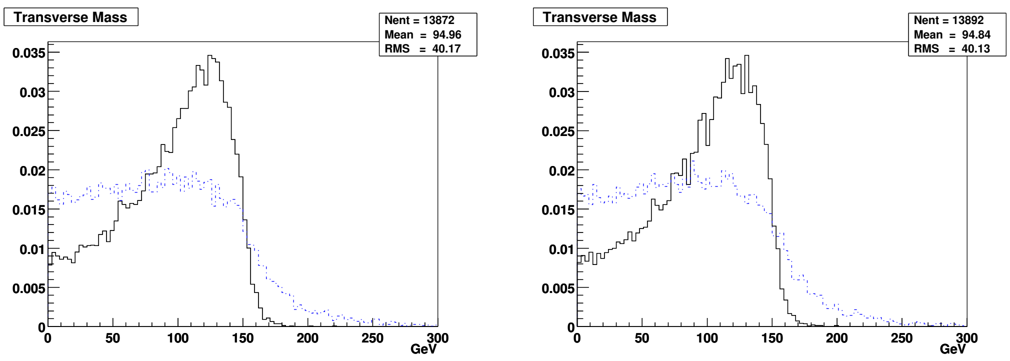


Figure 6.12: *Higgs signal and $t\bar{t}$ background transverse mass of the Higgs boson calculated from the signal leptons and the missing transverse momentum. The Fortran distributions are on the left and Athena on the right, with signal shown in black and background in blue.*

The AtlfastAnalysis cut results and histogramming showed exact agreement with the original fortran analysis routine, confirming its ability to reliably calculate complex variables, cut on a number of quantities and produce accurate histograms. Comparative results between analysis of Athena and Fortran fast simulation in table 6.4 show good agreement, with χ confidence levels of 99.99% and 92.79% for signal and background respectively.

	Signal			$t\bar{t}$		
Cuts	Fortran	Athena	ΔN	Fortran	Athena	ΔN
No Cuts	100,000	100,000	-	100,000	100,000	-
Basic Acceptance	15,820	15,811	0.07 σ	20,114	20,071	0.30
+ Separation	13,892	13,872	0.16 σ	14,811	14,772	0.32 σ
+ Forward Tagging	7,774	7,758	0.18 σ	325	312	0.72 σ
+ Central Jet Veto	6,847	6,857	-0.1 σ	169	156	1 σ
+ Tag Jet Mass	3,749	3,727	0.35 σ	16	15	0.25 σ
+ Lepton Angular	3,032	3,020	0.21 σ	3	2	0.57 σ
+ Lepton Mass/PT	2,882	2,875	0.13 σ	2	1	0.70 σ
+ Real τ Rejection	2,732	2,729	0.05 σ	2	1	0.70 σ

Table 6.4: *Cut Results for the* $\mathrm{h}_0 \rightarrow \mathrm{W}^+\mathrm{W}^- \rightarrow e^\pm \bar{\mu}^\mp\ p_{Tmiss}$ *Study*

6.3 Feasibility Study of $\mathrm{h}_0 \to \mathrm{Z}^0\mathrm{Z}^{0*} \to b\bar{b}\ l^+l^-$

The Higgs decay to Z bosons becomes important at 180 GeV due to the mass of the Z-boson. While the gold-plated $\mathrm{h}_0 \to \mathrm{Z}^0\mathrm{Z}^{0*} \to 4e$ channel is extremely successful, the branching ratio of the Z^0 to bottom quarks is some five times greater than that of the leptons.

Although quark final states are normally only viable in the context of associated Higgs production, it is possible that the $\mathrm{h}_0 \to \mathrm{Z}^0\mathrm{Z}^{0*} \to b\bar{b}\ l^+l^-$ channel via vector boson fusion could be a useful addition to the discovery channels in this mass range.

6.3.1 Event Characteristics

The topology of the $\mathrm{h}_0 \to \mathrm{Z}^0\mathrm{Z}^{0*} \to b\bar{b}\ l^+l^-$ channel shows similar features to the $\mathrm{h}_0 \to \mathrm{W}^+\mathrm{W}^- \to e^\pm\bar{\mu}^\mp\ p_{Tmiss}$ channel described earlier. The two recoiling quarks from the emission of the weak bosons cause very forward high p_T 'tagging' jets, and pairs of high p_T leptons and b-jets should be present.

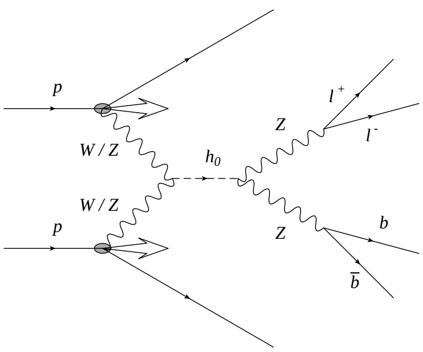


Figure 6.13: *The Higgs Signal.*

Again, the lack of colour flow between initial and final states causes suppressed hadronic activity in the central region, which may allow the b-jet pair from the hadronic Z^0 decay to be isolated.

Due to the low cross-section of the process, any cuts made must be extremely conservative and a multi-variate likelihood method of reducing background is developed to increase the discovery potential at such low production rates.

6.3.2 Event Generation

Signal events are generated using Pythia 6.203[32] with the standard Athena Generator package settings, and initial and final state radiation, fragmentation and decay, and multiple interactions turned on. Detector performance is simulated by the Athena-Atlfast package.

The branching ratios for Z decay to electrons or muons is 3.37% and b quark pairs is 15.13%, thus reducing $h \to ZZ$ events by 97.8%. Coupled with the lower cross-section of the vector boson fusion process, and suppression of the Higgs decay at the W resonance, this signal is particularly challenging due to its low cross-section over the intermediate mass region as shown in table 6.5

m_h (GeV)	120	130	140	150	160	170	180	190	200
σ_{signal} (fb)	1.0	2.6	4.3	5.5	2.9	1.4	3.29	12	13

Table 6.5: *Cross Sections of* $h \to ZZ$ *for* $100 \geq m_h \leq 200$ *GeV*

The generation of the backgrounds to the VBF channels are described in section 6.1.1. The backgrounds investigated in the following study are the QCD $t\bar{t}$ and $Z^0b\bar{b}$ processes, as well as the QCD WW+jets background. Their cross-sections are extremely large in comparison with the signal as shown in table 6.6.

Process	Cross-Section	events per year$(30fb^{-1})$
$h \to b\bar{b}ll$	13 fb	390
$t\bar{t}$	22,820 fb	684,600
$Z^0b\bar{b}$	65,820 fb	1,974,600
$WW + jets$	3,290 fb	98,700

Table 6.6: *Comparison of Background Cross Sections*

6.3.3 Event Selection and Topology

The event signature is two high pT like-flavour leptons, two central b-jets and two forward tagging jets. Considering the low branching ratio of the signal channel the efficiency of detecting this signature is extremely important.

The kinematic distribtutions of the generated Monte Carlo leptons and b-partons are shown in figures 6.14 and 6.15. The $t\bar{t}$ background is shown in blue, the $Z^0b\bar{b}$ in red, and the W^+W^-j in green.

In order that a lepton is detected by ATLAS, it must be in the pseudorapidity range -2.5 to +2.5 with a transverse momentum of 5 GeV. Of the signal leptons generated, 89% pass this veto, but only 75% are isolated by Atlfast. A minimum bias cut at 20 GeV reduces this to 69%, therefore only 47% of events contain two isolated leptons.

For real data, or full simulation studies, the effective tagging of b-jets relies on the relatively long lifetimes of b-hadrons, which give rise to displaced vertices of the jet. This vertex can be explicitly reconstructed, or impact parameters of the daughters can be examined. For this reason, a good impact parameter resolution $\sigma(d_0)$ is crucial to achieve good b-jet tagging. Other jets are rejected on the basis that they appear to originate from the primary vertex at the interaction point.

The b-jet direction is not uniquely defined as b-quarks in the final state of an interaction can radiate gluons and change direction. It is found in TDR studies that the b-quarks are found within a small ΔR cone of the jet reconstructed from the calorimeters and the charged tracks. This can therefore be modeled by a fast simulation program using ΔR distance between the Monte Carlo b-quark and the reconstructed jet as a b-tagging criterion.

For Atlfast to reconstruct a b-jet, there must exist a b-quark in the pseudorapidity range -2.5 to +2.5 with a transverse momentum of 5 GeV within a $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} = 0.2$ cone of a jet. Of the signal b-partons generated, 88% have the above kinematic properties, and 69% are also within the required proximity of a jet. Due to a 50% b-tagging efficiency, only 35% of the b-quarks are reconstructed as b-jets. With a 20 GeV minimum bias cut, only 12% of events contain 2 b-jets. Table 6.7 shows the efficiencies of detecting the leptons and b-jets.

N Leptons	0	1	2	3+
Efficiency	11.4 %	41.3 %	47.1 %	0.2%
N BJets	0	1	2	3+
Efficiency	43.1 %	44.7 %	11.9 %	0.3 %
2Leptons +	0 Bjet	1 Bjet	2 BJets	1+ Bjets
Efficiency	19.5 %	21.5 %	6.0 %	27.6 %

Table 6.7: *Accepted lepton/b-jet detection effficiencies. The tables show percentage of events with a number of leptons, b-jets, as well as a combination of 2 leptons and a number of b-jets.*

The $t\bar{t}$ distributions are very similar to signal, while the $Z^0 b\bar{b}$ and W^+W^-j are most noticeably different in the b-quark distributions. Both $Z^0 b\bar{b}$ and W^+W^-j distributions display a much lower peak in b-quark transverse momenta, while the W^+W^-j quarks are peaked in the very forward region in pseudorapidity and will therefore generally fail to be associated with a jet.

The first four columns of table 6.8 shows the background b-jet efficiencies of events with 2 leptons and the 20GeV minimum bias requirement. Due to the low cross-section of the signal process, choosing as conservative an event signature as possible is very important. A signature of two like flavour leptons and at least one b-tagged jet preserves the most signal events while retaining a b-jet condition.

Although only 40% of the events contain two leptons, detecting both high p_T leptons is absolutely necessary to select a signal event and resolve the Z mass in order to reject the huge $t\bar{t}$ background.

Large background rejection can be achieved by requiring the existence of at least four high pT jets, two from the hadronic decay and two forward tagging jets, which should not be present in the background processes. Along with the like-flavour lepton condition, these acceptance critera reduce further the backgrounds as shown in table 6.8, while only reducing the signal efficiency to 21%.

	0 Bjet	1 Bjet	2 BJets	1+ Bjets	Acceptance
Efficiency(tt)	16.5%	22.5%	7.9%	30.5%	6.5%
Efficiency(zbb)	27.5%	10.0%	1.2%	11.2%	4.2%
Efficiency(wwj)	35.7%	0.2%	0.01%	0.21%	0.014%

Table 6.8: *B-Jet and Full Acceptance Efficiencies for Background Events*

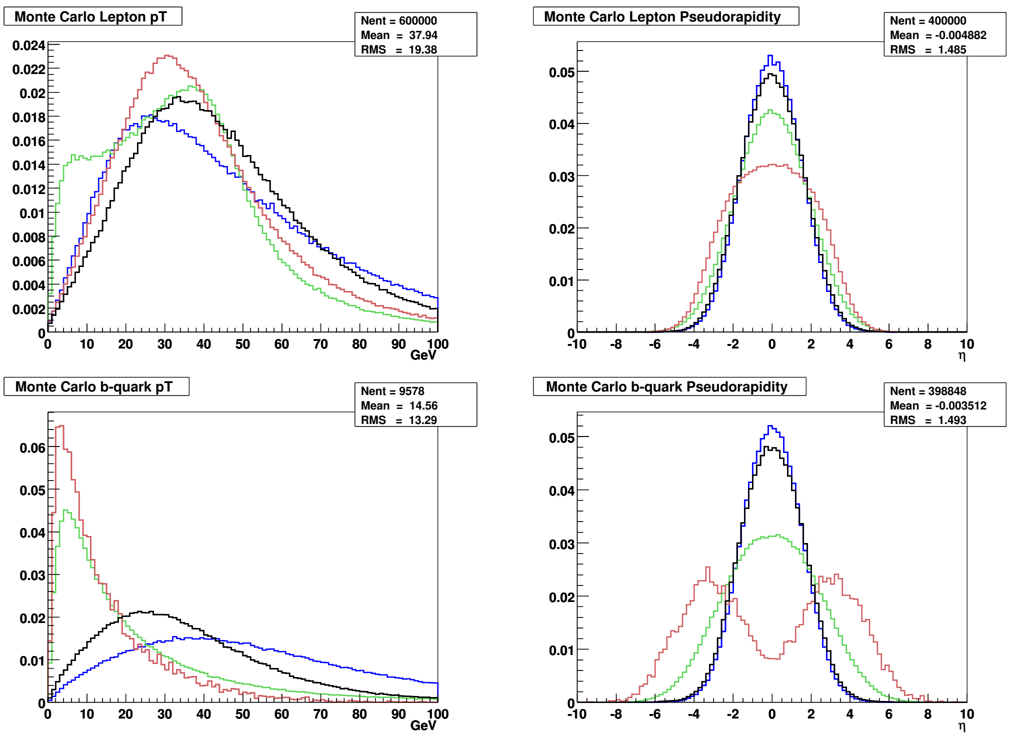


Figure 6.14: *Monte Carlo transverse momenta with respect to the beam line and psuedorapidity of all leptons and b-quarks for signal and backgrounds. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, the $Z^0b\bar{b}$ background in green, and the W^+W^-j background in red.*

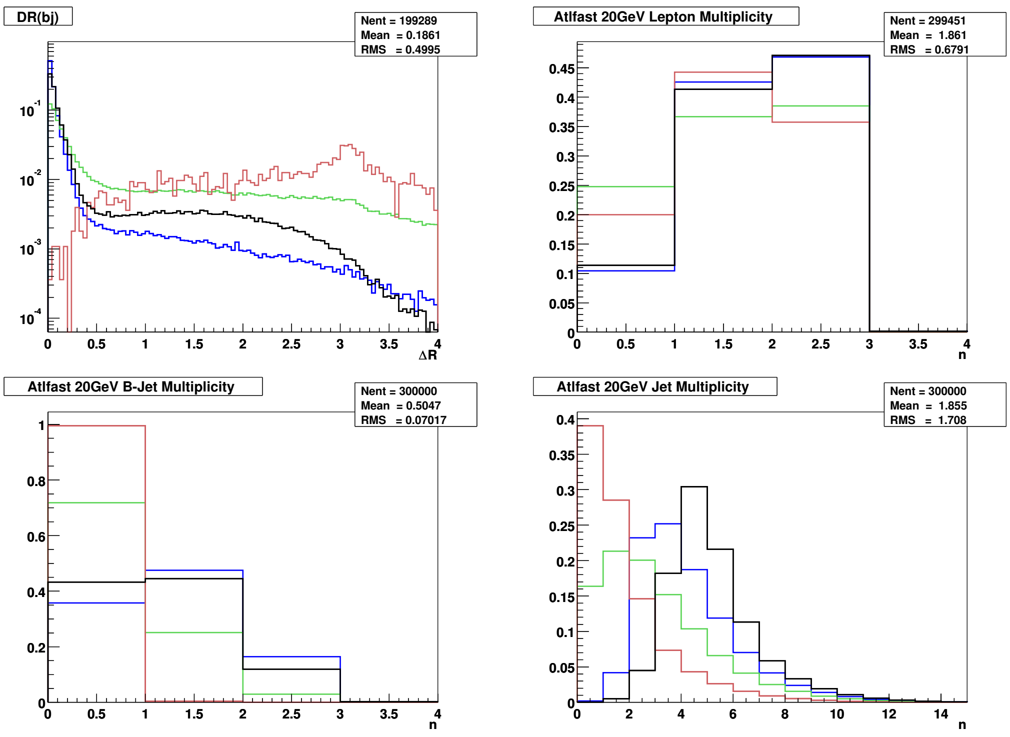


Figure 6.15: *The first diagram shows the R separation of each b-quark and the nearest Atlfast reconstructed jet for signal and backgrounds. The Monte Carlo multiplicities of all leptons, Atlfast reconstructed jets and Atlfast jets tagged as b-jets with a pT > 20 GeV are also shown for signal and backgrounds. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, the $Z^0 b\bar{b}$ background in green, and the W^+W^-j background in red.*

The lepton kinematics and missing momentum of the event should vary significantly between the $t\bar{t}$ and W^+W^-j backgrounds exhibiting W boson decays and the Z boson signal. Figure 6.16 shows the distributions of the di-lepton invariant mass, the lepton momenta boosted into the Z rest frame(constructed from the two lepton 4-vectors) and the missing momentum of the event.

The signal lepton invariant mass is peaked at the Z boson mass, while the boosted transverse momentum exhibits a peak at $\frac{m_Z}{2}$, effectively distinguishing the leptons which are produced from a Z decay. The backgrounds also show elevated missing momentum due to the neutrinos from the W decay. The $Z^0b\bar{b}$ background is irreducible in terms of these event kinematics.

The effects of cutting on these quantities are shown in table 6.9. As expected, the signal and $Z^0b\bar{b}$ background are not greatly affected while the $t\bar{t}$ is reduced considerably and the W^+W^-j is rendered insignificant. This background is therefore not considered further.

Figure 6.17 shows the topology of the hadronic event entities for the signal (shown in black), $t\bar{t}$ (shown in blue) and $Z^0b\bar{b}$ (shown in green). It is demonstrated that in the signal events, the highest pT jet not tagged as a b-jet is found in the forward region. This is most likely to be a recoil jet in the signal process, while the background jets tend to be more central with lower transverse momentum. The two tagging jets are chosen as this highest momentum jet, and the jet furthest in pseudorapidity. The pseudorapidity separation is shown to be more prominent in the signal than in the backgrounds.

The jets produced by the hadronic decay of the Z are expected to be high pT and in the central region. Due to the low b-tagging efficiency, the 2 btag requirement is relaxed, and the b system is chosen as the highest pT bjet and the jet with the combined invariant mass closest to the Z mass. The invariant mass and boosted transverse momentum with respect to the reconstructed Z frame of reference show a sharper peak in the signal as shown in figure 6.18, but does not offer high background rejection.

Cut	$h \rightarrow b\bar{b}ll$	$t\bar{t}$	$Z^0b\bar{b}$	W^+W^-j
Acceptance	2.75 fb	1401 fb	2669 fb	0.45 fb
$p^T_{miss} < 40GeV$	2.652	337.3	2547	0.23
$p_l^{T\ boosted} < 50GeV$	2.645	213.2	2504	0.17
$75 < m_{ll} < 100GeV$	2.527	59.4	2307	0.05

Table 6.9: *Preliminary Cut Cross-Sections*

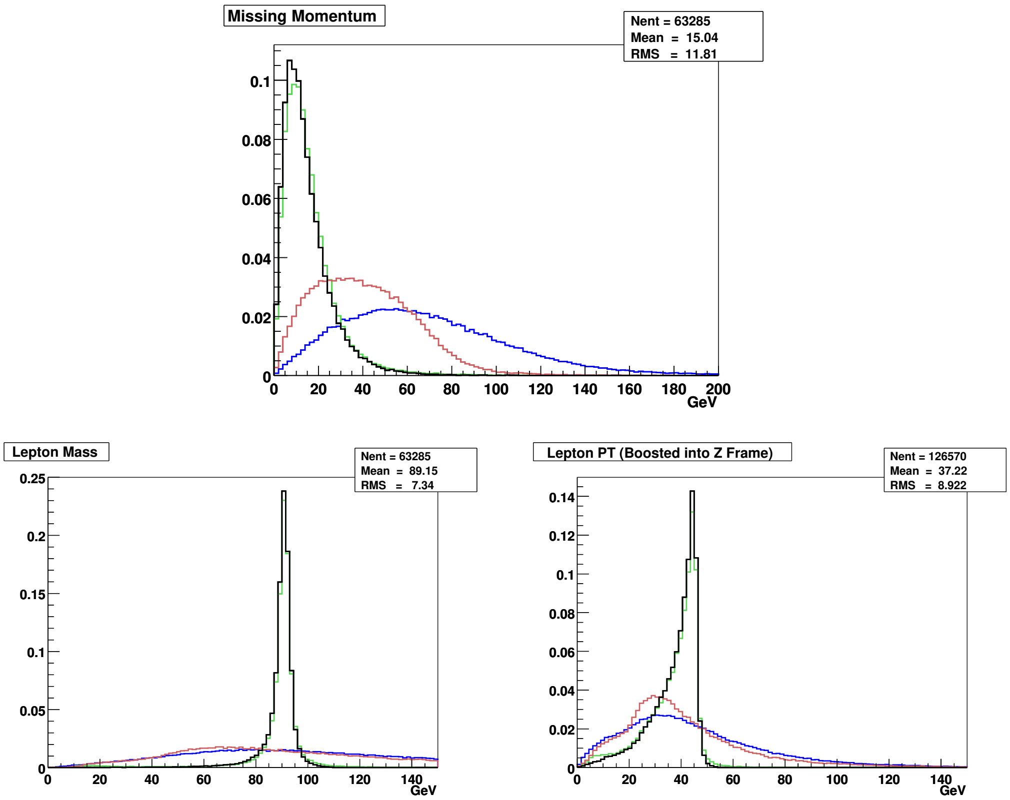


Figure 6.16: *The missing momentum, di-lepton invariant mass, and transverse momentum with respect to the reconstructed Z frame of reference for signal and backgrounds are shown above. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, the $Z^0 b\bar{b}$ background in green, and the W^+W^-j background in red.*

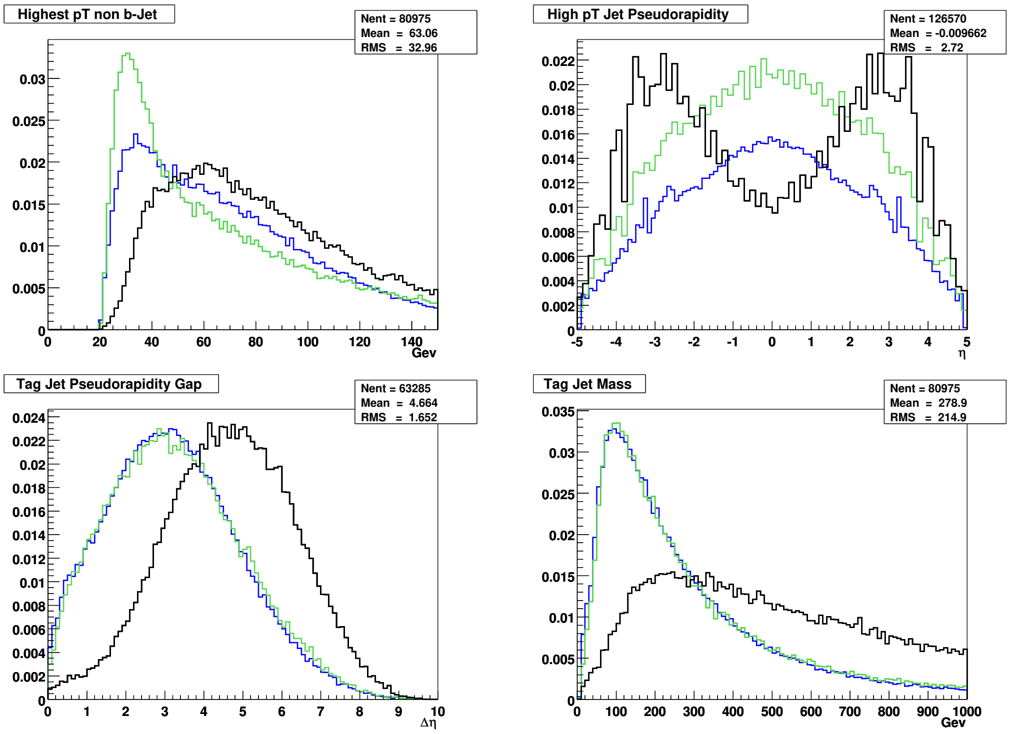


Figure 6.17: *These plots show the jet distributions of signal and background. The first two plots show the transverse momentum and pseudorapidity of the maximum pT Atlfast jet not tagged as a b-jet, while the lower plots show the pseudorapidity gap and invariant mass of the two jets chosen as the forward 'tagging' jets. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, and the $Z^0 b\bar{b}$ background in green.*

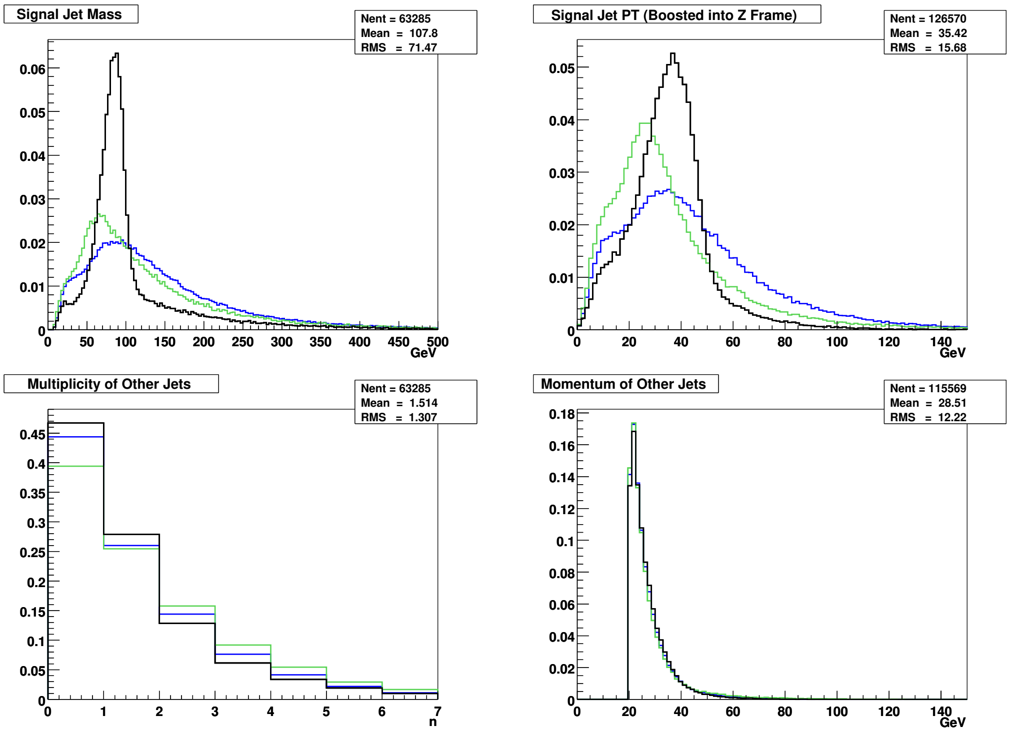


Figure 6.18: *The first two plots show kinematic distributions of the signal jets from the hadronic Z boson decay. The invariant mass of the jet system is shown, along with the transverse momentum with respect to the reconstructed Z frame of reference. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, and the $Z^0b\bar{b}$ background in green.*

6.3.4 The Likelihood Method

The Likelihood Method described in section 5.5 is employed to distinguish and reject background with minimal signal reduction. A number of variables with significantly different probability distribution functions between signal and background are chosen to perform the likelihood analysis. These variables are shown in figure 6.19 and 6.20, with the signal in black, the $t\bar{t}$ background in blue and the $Z^0b\bar{b}$ background in green.

The spatial separation of the Z boson four vectors calculated from the lepton and b-jet can be exploited using the likelihood method. The ΔR distribution of the signal has a fairly even distribution peaking slightly at 1, while the backgrounds both exhibit sharp peaks at π, indicating a tendency for back-to-back emission in the ϕ angle.

The tag jets also exhibit favourable kinematics for likelihood rejection. The pseudorapidity separation distribution of the signal tag jets is shifted significantly from both backgrounds, while the tag jet invariant mass is peaked lower in the backgrounds due to the higher invariant mass of weak processes, as shown in figure 6.17.

Boosting the Z decay products into the frame of the other Z boson(ie. leptons boosted into the b-jet system frame, and b-jets boosted into the lepton system frame) allows their angular distributions to be used as effective likelihood variables. The cosines of the angles between both the boosted leptons and the boosted b-jets show pronounced peaks toward perpendicular seperation at zero, while the backgrounds are peaked towards small angular separation at 1.

The pseudorapidity distributions of the Z decay products in relation to the tag jets contribute to both the likelihood method and an effective jet veto for reducing background, with only about 40% of background events having both leptons and both b-jets between the tag jets.

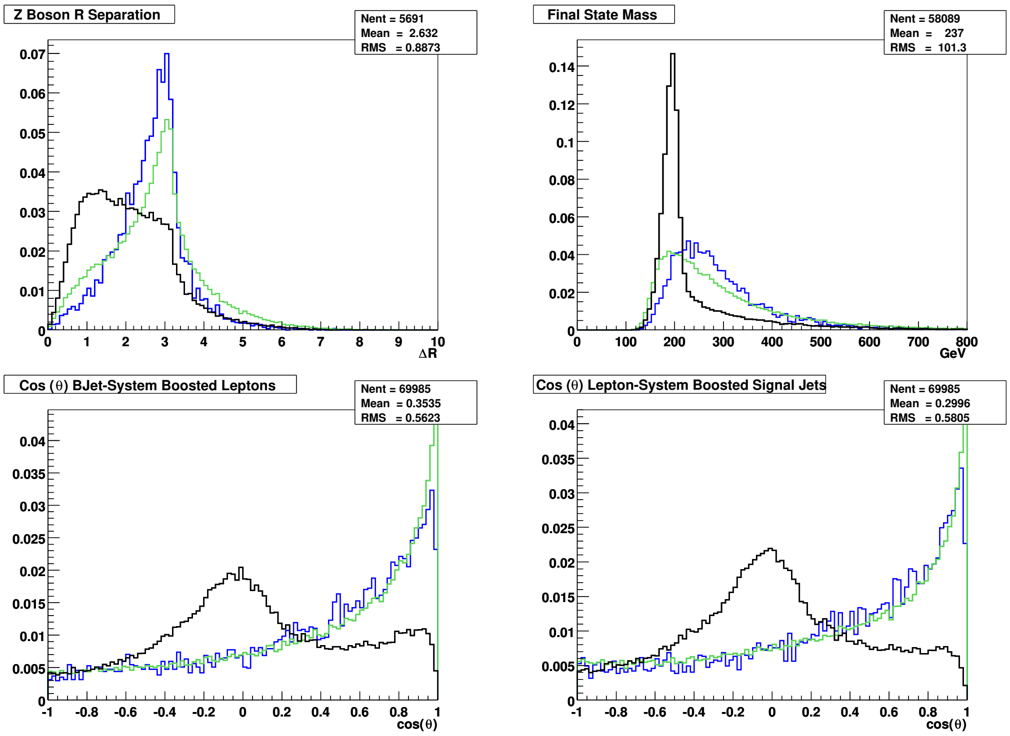


Figure 6.19: *These plots show signal and background distributions of variables chosen for the likelihood fit. The first plot shows the spatial separation of the two Z bosons four vectors, constructed from the combined vectors of the signal jets and leptons. The final state invariant mass, also constructed from these four vectors, is shown. The remaining plots show the cosine of the angle between the signal leptons boosted into the b-jet system frame, and the cosine of the angle between the signal b-jets boosted into the lepton system frame. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, and the $Z^0b\bar{b}$ background in green.*

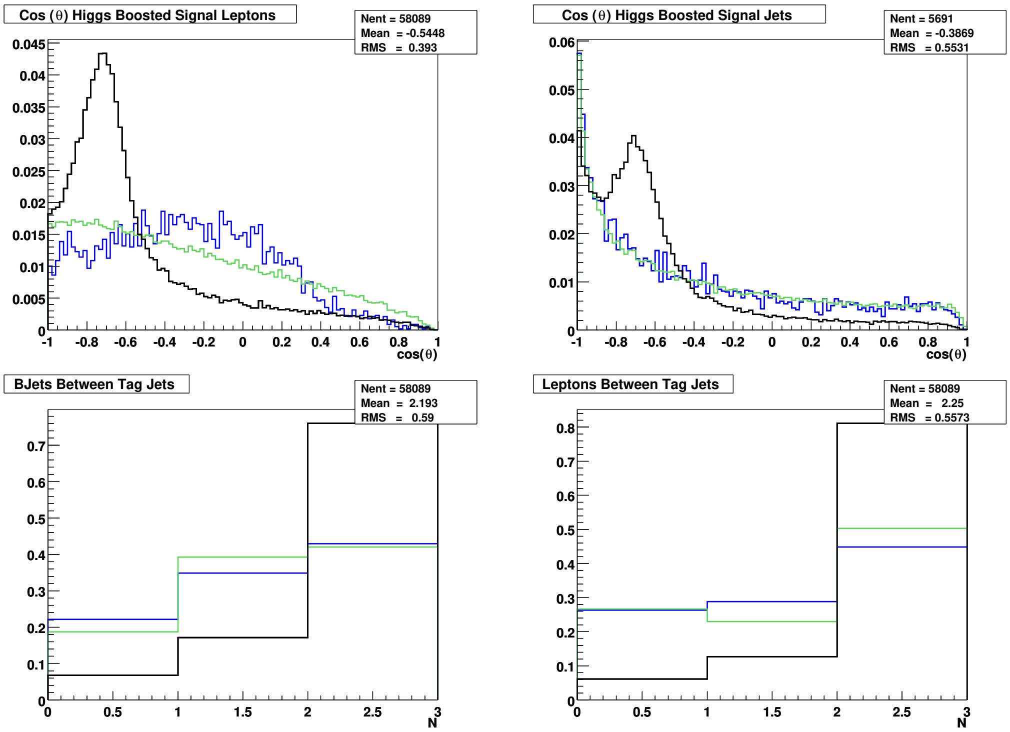


Figure 6.20: *These plots show further signal and background distributions of variables chosen for the likelihood fit. The first plots show the cosine of the angle between the signal leptons and the cosine of the angle between the signal b-jets boosted into the Higgs system frame, calculated from the four vectors of the signal leptons and b-jets. The remaining plots show the multiplicities of both signal b-jets and signal leptons found in the central pseudorapidity gap between the forward 'tagging' jets. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, and the $Z^0b\bar{b}$ background in green.*

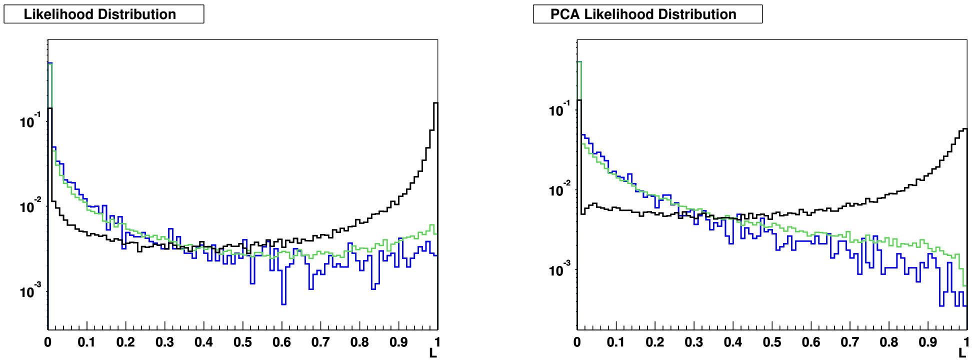


Figure 6.21: *Signal and background likelihood probabilities, with number of events normalised to unity. The left hand plot shows the non-correlated value, while the right hand plot shows the correlated value. The Higgs signal is shown in black, the $t\bar{t}$ background in blue, and the $Z^0b\bar{b}$ background in green.*

Event likelihoods for different sets of likelhood variables are calculated and compared using the two methods. The liksel[27] program is used to calculate correlation matrices for each set of variables, and produce Monte Carlo distributions representing the transformed n-dimensional Gaussian. These matrices and distributions are then used to perform the likelihood analysis.

Figure 6.21 shows example event likelihoods using both the uncorrelated and correlated likelihood methods. For both distributions the signal, shown in black, is peaked towards 1, while the $t\bar{t}$ and $Z^0b\bar{b}$ backgrounds(shown in blue and green respectively) tend towards 0.

Differences in background rejection efficiency are investigated for different values of the event likelihood over a number of variable groups. It is found that using a number of variables described in table 6.10 in the transformed Gaussian distribution maximises the rejection efficiency.

The integer multiplicity variables are not included in the correlation matrices or generated Gaussian distribution, but are added untransformed to the overall probability, in the same way that variables are used in the uncorrelated likelihood method.

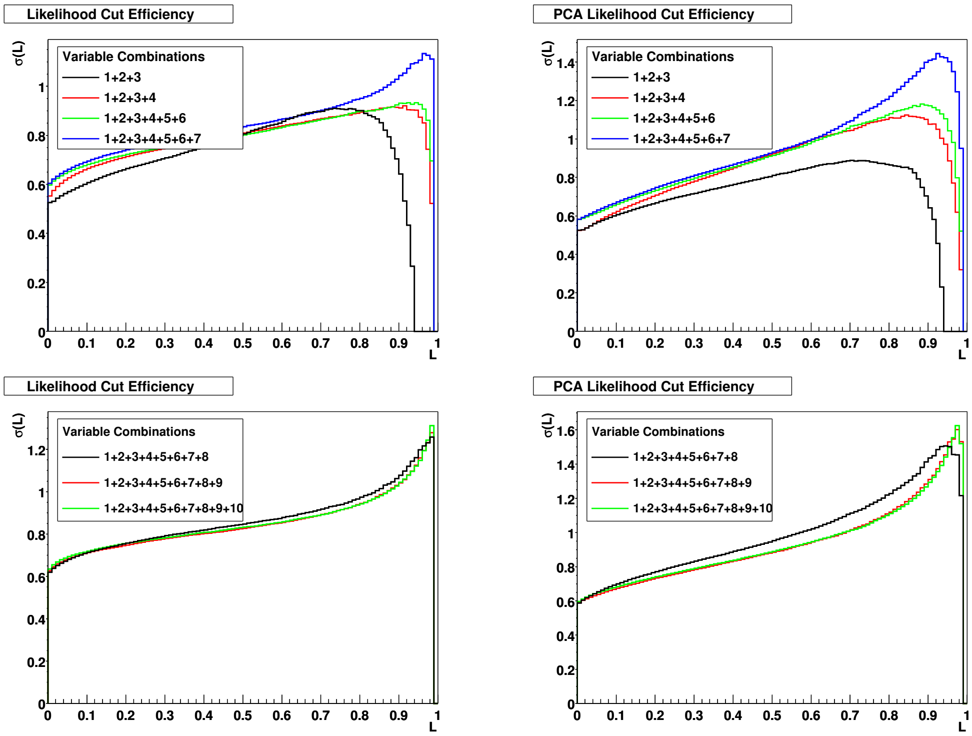


Figure 6.22: *Rejection efficiencies($S/\sqrt{B}$ at 1 years high luminosity) shown against likelihood probability cut value for a number of combinations of likelihood variables, for both non-correlated and correlated likelihood value. The key shows the combination of variables used to calculate the efficiency, referenced in the table below.*

ID	Variable
1	ΔR Z Bosons(Boosted leptons and b jets)
2	Invariant Mass of Tag Jets
3	$\Delta\eta$ of Tag Jets
4	Invariant Mass of Higgs(Boosted Z Bosons)
5	Multiplicity of B-Jets between Tag Jets in Pseudorapidity
6	Multiplicity of Leptons between Tag Jets in Pseudorapidity
7	$\cos(\theta)$ Leptons boosted into b-jet system
8	$\cos(\theta)$ b-jets boosted into lepton system
9	$\cos(\theta)$ leptons boosted into Higgs system
10	$\cos(\theta)$ b-jets boosted into Higgs system

Table 6.10: *Variables used for the likelihood efficiency calculation as shown in the plots above.*

Figure 6.22 shows the rejection efficiency($S/\sqrt{B}$ at 1 years high luminosity running) of the event likelihood for a number of combinations of likelihood variables from table 6.10, using both the uncorrelated and correlated likelihood methods. Using the correlated method with all the variables and a likelihood threshold of 98% gives the highest rejection of 1.6σ, reducing the signal to a little over 20% of its cross-section at the signature cut level.

Figure 6.23 shows these variable distributions after the likelihood cut for the signal and $Z^0b\bar{b}$ background. The kinematic distributions are almost identical, although the multiplicity of other event jets between the tag jets provides further background rejection, as shown in figure 6.24 for signal and both backgrounds. A jet-lepton veto which requires both signal jets and at least one lepton between the tag jets in pseudorapidity is also applied.

Table 6.11 shows the event rejection for the 200 GeV Higgs signal using the preliminary, likelihood and jet vetos. The discovery efficiency is shown for 1 year at high luminosity, reaching only 1.76σ, and achieving an observation efficiency of 3σ only after 3 years at $100fb^{-1}$.

Cut	hzz	tt	zbb	$S/\sqrt{B}$
Signal	2.747	650.0	2664	0.477
Preliminary Cuts	2.521	27.57	2302	0.522
Likelihood (98%)	0.557	0.053	11.77	1.620
Jet-Lepton Veto	0.543	0.048	11.02	1.632
2 Jets Between Tags	0.506	0.043	8.586	1.722
1 Jet Between Tags	0.461	0.029	6.777	1.767

Table 6.11: *Analysis Results* $m_h = 200$ *GeV*

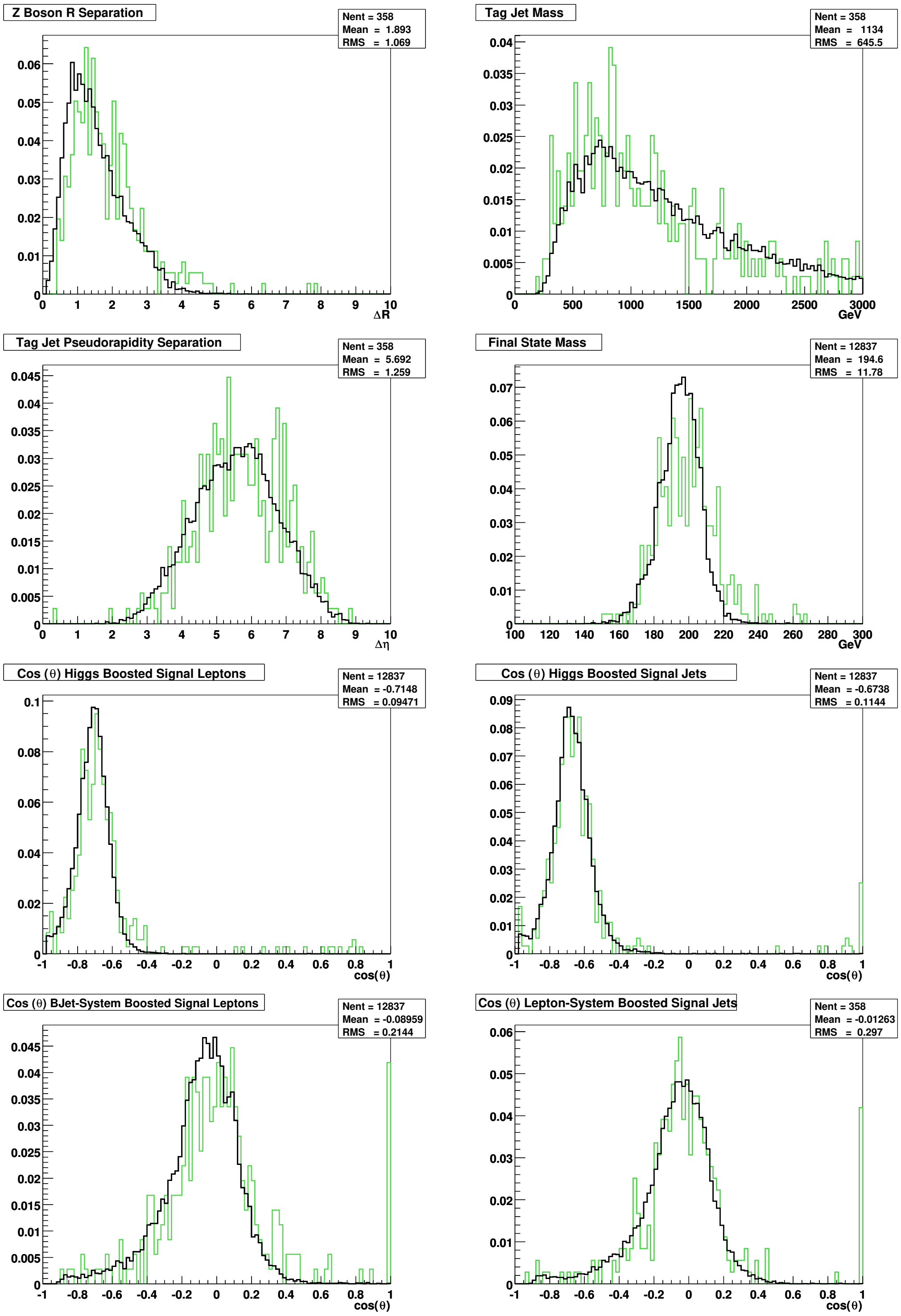


Figure 6.23: *Likelihood variable distributions of signal and $Z^0 b\bar{b}$ background after the likelihood cut has been applied. The Higgs signal is shown in black, the $Z^0 b\bar{b}$ background in green.*

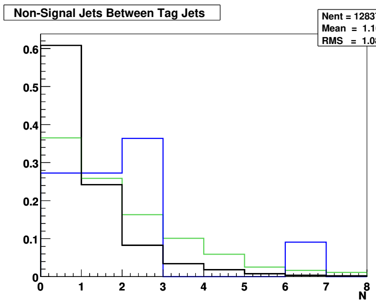


Figure 6.24: *Multiplicity of remaining non-signal jets in central pseudorapidity gap between forward tagging jets after the likelihood cut.*

Further analyses were performed over a Higgs mass range of 150 GeV to 190 GeV. The variable distributions for the tagging jets do not differ across the entire mass range, while the lepton and signal jet distributions are affected by one of the Z bosons from the Higgs decay being off-shell.

Figure 6.25 shows kinematic distributions for the signal from 150 to 190 GeV. At low mass, the presence of the off-shell Z boson causes a double peak in the lepton momentum and invariant mass distributions, while the signal jet distribution shows a wider spread skewed below the Z mass.

The final state mass constructed from the leptons and signal jets is peaked at the respective Higgs mass, and the separation of the Z bosons follows a similar distribution to the 200 GeV signal, which can again be exploited using the projection and correlation likelihood method.

The lower mass boosted jet and lepton separation cosine distributions are not peaked away from background as predominantly as the higher mass distributions, such as the 180 and 190 GeV distribution, as shown in figure 6.26. This lower potential background rejection coupled with the lower cross-section at these masses will reduce the discovery potential in the region below the Z mass threshold.

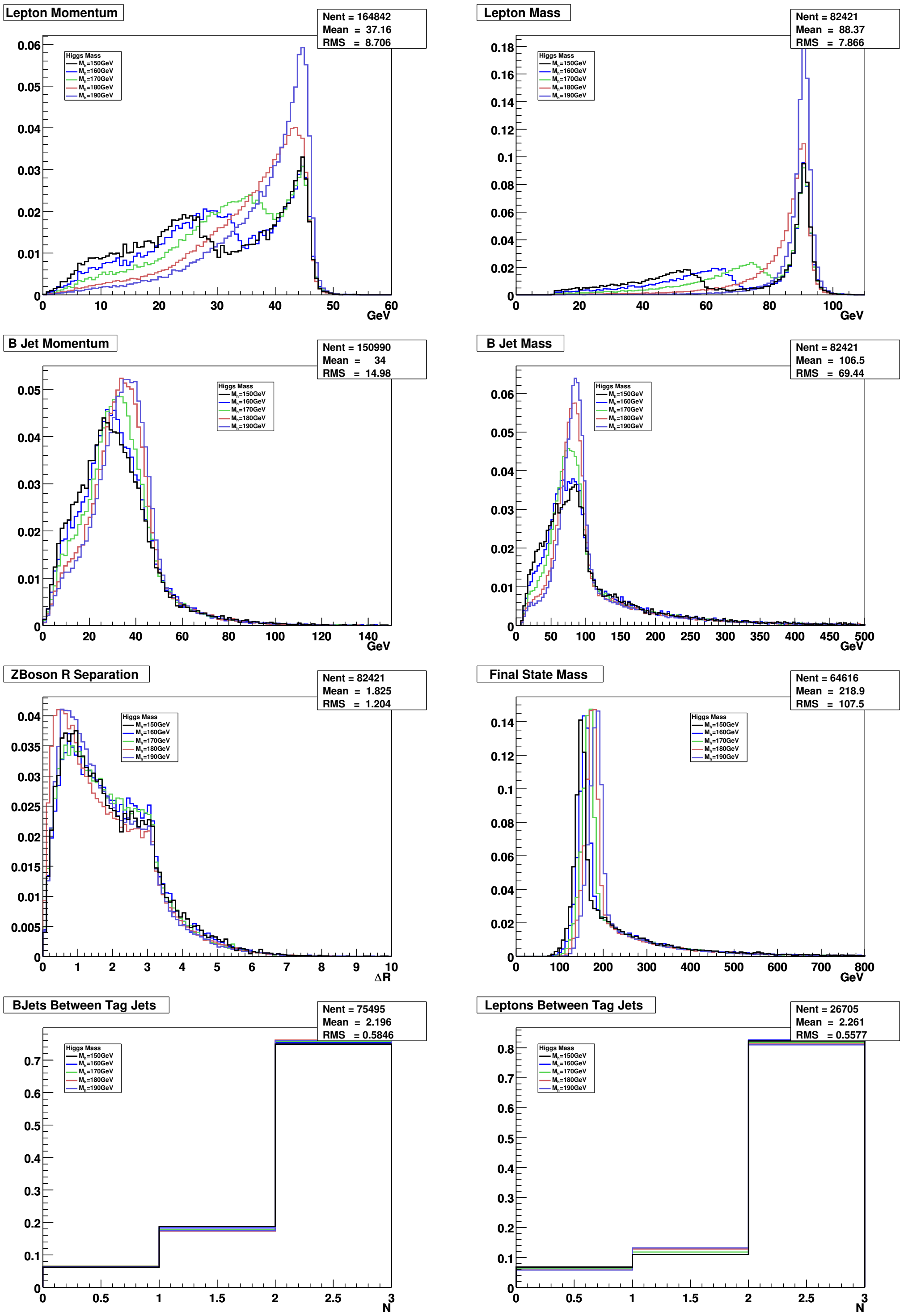


Figure 6.25: *Likelihood variable distributions of the signal process over the mass range 150-190 GeV.*

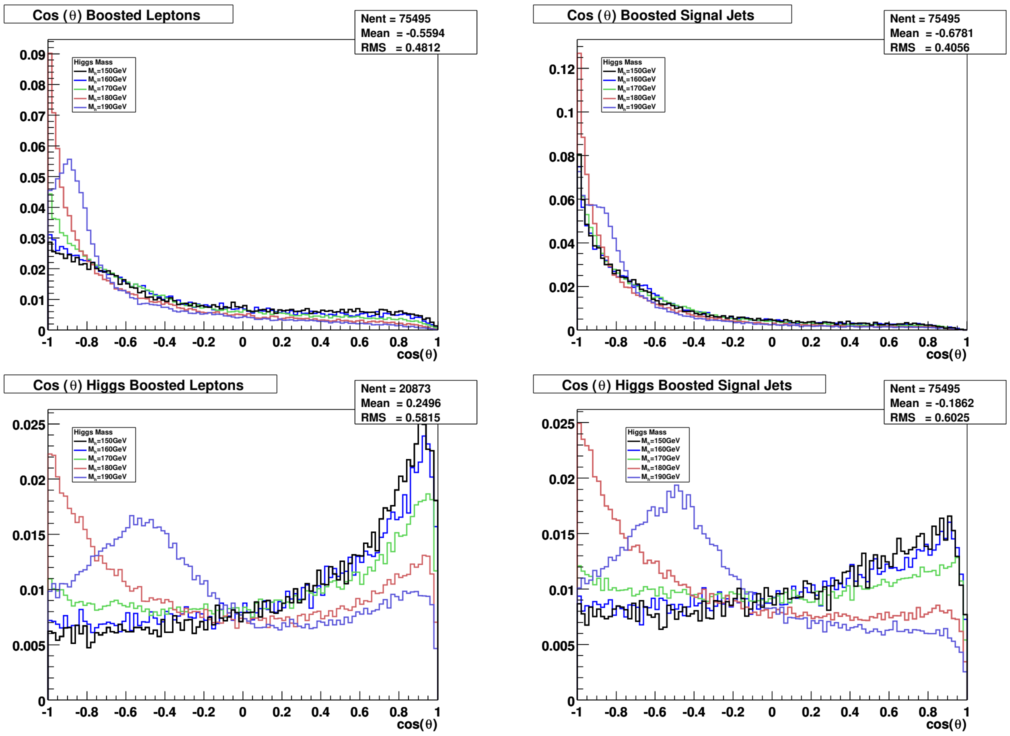


Figure 6.26: *Likelihood variable distributions of the signal process over the mass range 150-190 GeV.*

The distributions of the lepton mass and boosted momentum are not well modeled by the projection correlation method and, when used without transformation as extra likelihood variables, do not contribute significantly to the background rejection provided by the established likelihood variables.

The preliminary cut for the lepton invariant mass is relaxed from a lower limit of 75 GeV, to zero. While having little effect on the $Z^0b\bar{b}$ rejection in this area, the rejection efficiency of this cut on the $t\bar{t}$ background is reduced by a factor of 4 in the lower mass regions.

Using all variables from table 6.10 rejects backgrounds most effectively in the lower mass range. The likelihood rejection efficiencies over the mass range 150-190 GeV are shown in figure 6.27. The lower cross-section of the signal

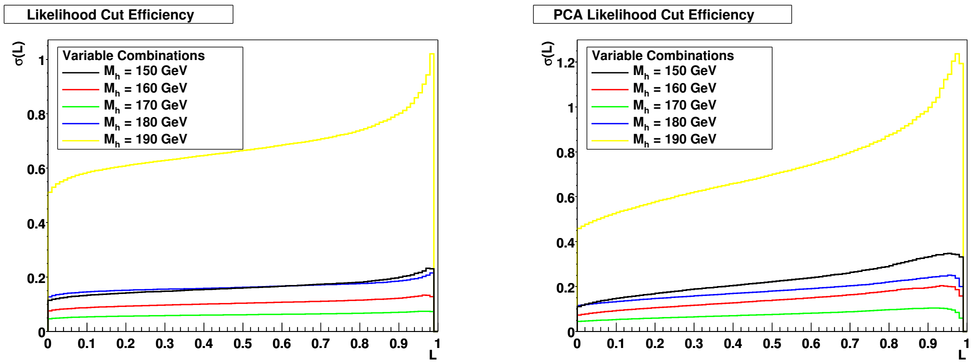


Figure 6.27: *Rejection efficiencies($S/\sqrt{B}$ at 1 years high luminosity) shown against likelihood probability cut value using all likelihood variables over the mass range 150-190 GeV, for both non-correlated and correlated likelihood value.*

process below the Z boson mass threshold coupled with the poor rejection provided by the boosted lepton and signal jet separation distributions causes the significant drop in discovery potential below the 190 GeV mass.

The optimal likelihood cut for each mass is chosen, while supplementing the likelihood rejection efficiency with central jet vetos increases the rejection efficiency to a small degree. The event numbers and statistical significances of the Higgs signal over this mass range are shown in table 6.12 and plotted in figure 6.28. The lowest potential occurs between 160 and 180 GeV, where the branching ratio of the Higgs to ZZ drops dramatically at the real W mass threshold, as shown in figure 6.3.

m_h GeV	N_{signal}	N_{tt}	N_{Zbb}	$S/\sqrt{B}$
150	6.1	12.5	187.5	0.437
160	5.1	51.3	361.8	0.255
170	2.8	56.6	361.8	0.138
180	10.6	41.1	1658	0.257
190	53.8	26.6	2059	1.178
200	46.1	2.9	677.7	1.767

Table 6.12: *Numbers of events and statistical significance for the Higgs signal over the studied mass range.*

Below 150 GeV, the Higgs decays predominantly to b-quark pairs, so the signal is not expected to improve significantly below the mass range considered, whereas the signal cross-section drops off slowly with increasing Higgs mass, remaining at around 10fb up to $m_h \approx 270$ GeV. Above this threshold, the signal decreases rapidly as Higgs production via vector boson fusion becomes less significant.

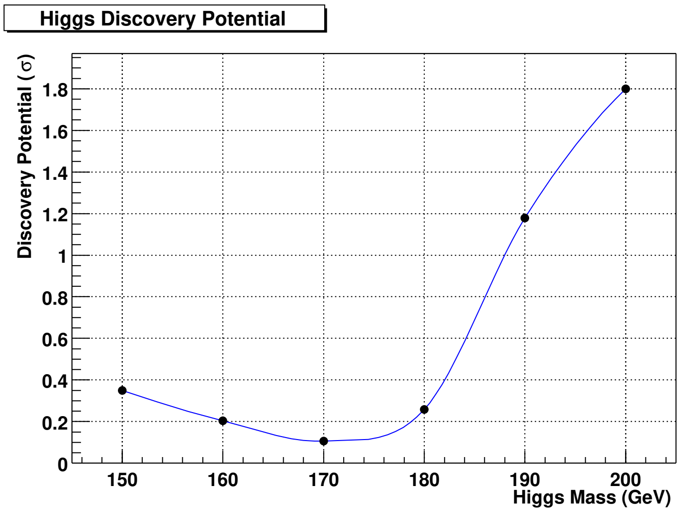


Figure 6.28: *The Higgs Discovery Potential using the $h \rightarrow ZZ$ channel*

6.3.5 Conclusions

The Athena Atlfast software was shown to produce reliable simulation of the ATLAS detector compared with previous software models using kinematic distributions of 100,000 $t\bar{t}$ events produced by the Pythia event generator. The fast simulation program in combination with the Atlfast Analysis package reproduced results for signal and one background for the $\mathrm{h}_0 \to \mathrm{W}^+\mathrm{W}^- \to e^{\pm}\bar{\mu}^{\mp}\ p_{Tmiss}$ Higgs discovery channel, showing excellent agreement with results using previous software.

The capability of the ATLAS detector in discovering the standard model Higgs boson in the mass range 150-200 GeV via the $\mathrm{h}_0 \to \mathrm{Z}^0\mathrm{Z}^{0*} \to b\bar{b}\ l^+l^-$ channel was investigated. The two main backgrounds were evaluated, and a projection-correlation likelihood method employed to improve the discovery potential. This channel was not, however, found to significantly increase the total discovery potential of the Higgs over this mass range, contributing a minimum of 0.1σ at 170 GeV and a maximum of 1.8σ at 200 GeV.

The signal can only be distinguished from the main $Z^0b\bar{b}$ background by investigating the origin of the b-jets, produced from a Z boson in the signal and a gluon in the background. For this reason, b-tagging using track vertex studies may produce additional rejection of the $Z^0b\bar{b}$ background, increasing the discovery potential of the Higgs using this channel. More accurate evaluation of the contributing backgrounds, including full evaluation of uncertainties in the background production cross-sections would need to be carried out, should vertex studies prove to significantly increase signal rates and rejection of the main irreducible background.

Bibliography

[1] J.Iliopoulos, L.Maiani, S.Glashow. Physical Review D, 2(7):1285, 1970.

[2] F.Mandl, G.Shaw. Quantum Field Theory, Wiley & Sons, 1996.

[3] F.Halzen, A.D.Martin. Quarks and Leptons: An Introductory Course in Modern Particle Physics, Wiley & Sons, 1984.

[4] The LHC Study Group. Design Study of the Large Hadron Collider, a multi-particle collider in the LEP tunnel, CERN/91-03, May 1991.

[5] L.R.Evans. LHC Status and Plans. Technical Report CERN-LHC Report-101, May 1997.

[6] The ATLAS Collaboration. ATLAS Technical Proposal for a General Purpose pp Experiment at the Large Hadron Collider at CERN, CERN/LHCC/94-43, LHCC/P2, December 1994.

[7] The CMS Collaboration. The Compact Muon Solenoid - Technical proposal, CERN/LHCC/94-38, 1995.

[8] ALICE Collaboration, Technical Proposal for a Large Ion Collider Experiment at the CERN LHC, CERN/LHCC/95-71, 1995.

[9] LHC-B Collaboration, Letter of Intent for a Dedicated LHC Collider Beauty Experiment for Precision Measurements of CP-Violation, CERN/LHCC/95-5, 1995.

[10] The ATLAS Collaboration. ATLAS Physics Technical Design Report, CERN/LHCC/99-4, LHCC/I2, 1992.

[11] The ATLAS Inner Detector Community. The ATLAS Inner Detector Technical Design Report, CERN/LHCC/97-16, April 1997.

[12] The ATLAS Trigger Performance Group, ATLAS Trigger Performance Status Report, CERN/LHCC/98-15, 30 June 1998.

[13] The ATLAS Calorimeter Community. The ATLAS Calorimeter Performance Technical Design Report, CERN/LHCC 98-13, 1998.

[14] The ATLAS Muon Collaboration. ATLAS Muon Spectrometer Technical Design Report, CERN/LHCC/97-22, June 1997.

[15] Athena User Guide, http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture.

[16] AIDA, Abstract Interfaces for Data Analysis group, http://wwwasd.web.cern.ch/wwwasd/lhc++/AIDA.

[17] R.Brun, HBOOK, Statistical Analysis and Histogramming; version 4.22, CERN Program Library Long Writeup Y250, 1994.

[18] CLHEP group, http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep.

[19] Design Patterns, Elements of Reusable Object-Oriented Software; E.Gamma, R.Helm, R.Johnson, J.Vlissides. Addison Wesley Ed 16, 1998, ISBN 0-201-63361-2.

[20] The Standard Template Library, http://www.cs.rpi.edu/projects/STL/htdocs/stl.html.

[21] D. Groom *et al.*, Review of Particle Physics, The European Physical Journal **C15**, 1 (2000).

[22] E. Richter-Was *et al.*, ATLFAST 2.0, a fast simulation package for ATLAS, ATL-PHYS-98-131 November 1998.

[23] E.J.Buis, R. Dankers, S. Haywood, A. Reichold. ATLAS Internal Note, INDET-No-195 December 1997.

[24] E.J.Buis, R. Dankers, A. Reichold, S. Haywood, F.G. Tartarelli, and N.labanca, ATLAS Internal Note, ATL-INDET-98-215).

[25] Configuration Management Tool, Christian Arnault, http://www.cmtsite.org.

[26] D. Karlen, Using projections and correlations to approximate probability distributions, Computers in Physics **12**, 4 (1998).

[27] PCA Likelihood liksel program, ftp.aip.org/cip_sourcecode/karlen_ja_98.

[28] C.Buttar, R.Harper, K.Jakobs, Weak Boson Fusion $\mathrm{h}_0 \rightarrow \mathrm{W}^+\mathrm{W}^- \rightarrow e^{\pm}\bar{\mu}^{\mp}\ p_{Tmiss}$ as a search mode for an intermediate mass Standard Model Higgs boson at ATLAS. Atlas Internal Note, ATLAS-PHYS-2002-033 (2002).

[29] LEP Collaborations, Searches for the Standard Model Higgs Boson at LEP, CERN-EP/2003-011 (2003), hep-ex/0306033.

[30] ATLAS Detector and Physics Performance Technical Design Report, vol. 2, CERN-LHCC-99-15, ATLAS-TDR-15.

[31] M.Klute, A study of the weak boson fusion with $H \to \tau\overline{\tau}$ and $\tau \to e(\mu)$. ATLAS internal note ATL-PHYS-2002-018 (2002).

[32] T.Sjostrand *et al.*, Pythia 6.2 Physics and Manual, hep-ph/0108264, LU TP 01-21, April 2002.

[33] Borut Paul Kersevan, Elzbieta Richter-Was. The Monte Carlo Event Generator AcerMC 1.0 with Interfaces to PYTHIA 6.2 and HERWIG 6.3, hep-ph/0201302 (2002).

[34] E. Richter-Was *et al.*, ATLFAST++ fast simulation package for ATLAS, http://atlasinfo.cern.ch/Atlas/GROUPS/PHYSICS/Physics.html

Appendix A

Atlfast Implementation Detail

This appendix describes the detailed implementation of the Atlfast modules, including the initialisation and execution sequence diagrams, job options steering parameters and smearing parameterisations. The various Atlfast helper classes are also listed here.

A.1 Sort and Partition Functions

The sort and partition condition classes provided by Atlfast are listed in table A.1. These classes are used with the STL generic sort and partition functions.

Class	**Description**
AscendingPhi	Ascending sort by azimuthal angle ϕ
DescendingPhi	Descending sort by azimuthal angle ϕ
AscendingEta	Ascending sort by pseudorapidity η
DescendingEta	Descending sort by pseudorapidity η
AscendingET	Ascending sort by transverse energy e_T
DescendingET	Descending sort by transverse energy e_T
AscendingPT	Ascending sort by transverse momentum p_T
DescendingPT	Descending sort by transverse momentum p_T
AscendingMT	Ascending sort by transverse mass m_T
DescendingMT	Descending sort by transverse mass m_T
DeltaPhi	Ascending sort by $\Delta\phi$ from a reference
DeltaEta	Ascending sort by $\Delta\eta$ from a reference
DeltaR	Ascending sort by ΔR from a reference
AboveThresholdET	Partition above transverse energy threshold
BelowThresholdET	Partition below transverse energy threshold
BelowThresholdDeltaR	Partition below ΔR threshold

Table A.1: *Sort and Partition Classes*

A.2 IMCSelector Concrete Classes

The IMCSelector classes are used by the TesIO to filter the Monte Carlo input particles. The concrete implementations of the IMCSelector objects are listed in table A.2. They are created by the Atlfast Algorithms in order to select the relevant particles for the algorithms specific function and are passed to the TesIO and used via their common IMCSelector interface.

Class	**Description**
NCutter	Multiple selector (filter pattern)
All	Selects all particles
BFieldCutter	All particles reaching barrel calorimeter
IsCharged	All charged particles
IsFinalState	All final-state particles
IsStatusxx	All particles with given status
MCCuts	All particles passing p_T and η cuts
MCselectorWrapper	Selects all particles
SelectType	Selects particles with given PID
RejectType	Reject particles with given PID
SelectJetTag	All particles valid for jet tagging
SelectTauTag	All particles valid for tau tagging
SelectZ0	All Z^0 particles
Unseen	All particles invisible to the calorimeter
AscendingEta	Order particles by pseudorapidity

Table A.2: *Concrete IMCSelector Classes*

A.3 DefaultReconstructedParticleMaker

This appendix shows the sequence diagrams and job options parameters for the DefaultReconstructedParticleMaker algorithm, and describes the smearing parameterisations of the particles which it creates.

The DefaultReconstructedParticleMaker is initialised and executed as shown in figure A.1. The available job options parameters are listed in table A.3 and in the standard release are supplied with ATLAS specific default values.

The smearing parameterisations used in Atlfast to simulate the ATLAS detector effects for three types of ReconstructedParticle are based on the Atlfast++ package parameterisations[34].

The photon smearing is parameterised for both high and low luminosity over a range of pseudorapidities. The polar angle is smeared first, followed by the photon energy.

The polar angle θ is smeared according to the following Gaussian resolution functions dependent on the pseudorapidity co-ordinate η.

$$\sigma(\theta_\gamma) = \frac{0.065}{\sqrt{E_\gamma}} \quad \text{for } |\eta| < 0.8 \tag{A.1}$$

$$\sigma(\theta_\gamma) = \frac{0.050}{\sqrt{E_\gamma}} \quad \text{for } 0.8 < |\eta| < 1.4 \tag{A.2}$$

$$\sigma(\theta_\gamma) = \frac{0.040}{\sqrt{E_\gamma}} \quad \text{for } 1.4 < |\eta| < 2.5 \tag{A.3}$$

The energy is then smeared using the θ adjusted momentum according to the following Gaussian parameterisations, dependent on pseudorapidity. The $\sigma_{pile-up}$ coefficient is zero for low-luminosity and is dependent on pseudorapidity for high luminosity.

$$\frac{\delta E_\gamma}{E_\gamma} = \frac{0.1}{\sqrt{E_\gamma}} + \frac{0.245}{E_\gamma^T} + \frac{\sigma_{pile-up}}{E^T} + 0.007 \qquad \text{for } |\eta| < 1.4 \tag{A.4}$$

$$\frac{\delta E_\gamma}{E_\gamma} = \frac{0.1}{\sqrt{E_\gamma}} + \frac{0.306(2.628-\eta)}{E_\gamma^T} + \frac{\sigma_{pile-up}}{E^T} + 0.007 \quad \text{for } |\eta| > 1.4 \tag{A.5}$$

$$\sigma_{pile-up} = 0.32 \quad \text{for } |\eta| < 0.6 \tag{A.6}$$

$$\sigma_{pile-up} = 0.295 \quad \text{for } 0.6 < |\eta| < 1.4 \tag{A.7}$$

$$\sigma_{pile-up} = 0.27 \quad \text{for } |\eta| > 1.4 \tag{A.8}$$

The electron smearing is parameterised for both high and low luminosity over a range of pseudorapidities in a similar manner to the photons. The polar angle is well measured using Inner Detector information, and is therefore not smeared. The pile-up coefficients are the same as in photon smearing.

$$\frac{\delta E_\gamma}{E_\gamma} = \frac{0.12}{\sqrt{E_\gamma}} + \frac{0.245}{E_\gamma^T} + \frac{\sigma_{pile-up}}{E^T} + 0.007 \qquad \text{for } |\eta| < 1.4 \tag{A.9}$$

$$\frac{\delta E_\gamma}{E_\gamma} = \frac{0.12}{\sqrt{E_\gamma}} + \frac{0.306(2.628-\eta)}{E_\gamma^T} + \frac{\sigma_{pile-up}}{E^T} + 0.007 \quad \text{for } |\eta| > 1.4 \tag{A.10}$$

The muon smearing is parameterised according to equation A.11, where σ is calculated by a dedicated fortran routine. The resolution is derived from the inner detector, the stand-alone muon-system, or a combination of the two as described in [22].

$$P^{\mu}_{smeared} = \frac{P^{\mu}_{true}}{1+\sigma} \tag{A.11}$$

Property	**type**	**Description**
ParticleType	int	PDG of particles to be created
McPtMinimum	double	Minimum p_T required to make particle
McEtaMaximum	double	Maximum η required to make particle
PtMinimum	double	p_T acceptance for smeared particle
EtaMaximum	double	η acceptance for smeared particle
DoSmearing	bool	Smearing switch
Bfield	double	Tesla value of magnetic field
MC_eventLocation	string	TES location of Monte Carlo data
OutputLocation	string	TES location for particle storage
MuonConfiguration	int	Configuration key for muon smearing

Table A.3: *DefaultReconstructedParticleMaker Job Options Parameters*

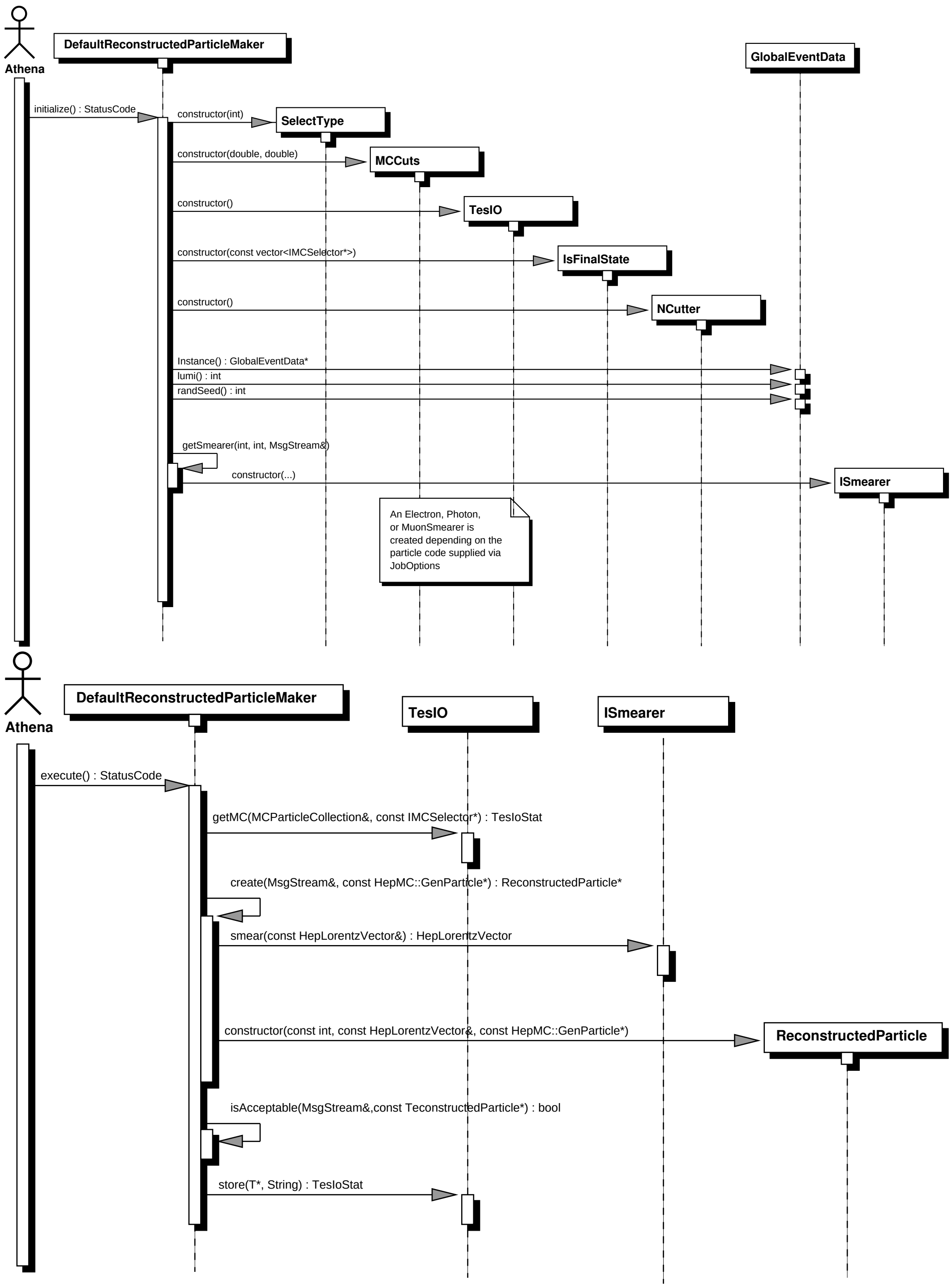


Figure A.1: *The DefaultReconstructedParticleMaker Initialisation and Execution Sequence Diagram*

A.4 CellMaker

This appendix shows the sequence diagrams and job options parameters for the CellMaker algorithm, and describes the cell smearing parameterisations and pile-up mechanism.

Property	**type**	**Description**
EtaCoverage	double	η coverage of calorimeter
MinETCell	double	Minimum E_T required to activate cell
GranBarrelEta	double	η granularity of calorimeter barrel region
GranBarrelPhi	double	p_T granularity of calorimeter barrel region
GranForwardEta	double	η granularity of calorimeter end-cap region
GranForwardPhi	double	p_T granularity of calorimeter end-cap region
DoSmearing	bool	Smearing switch
MC_eventLocation	string	TES location of Monte Carlo data
OutputLocation	string	TES location for Cell storage

Table A.4: *CellMaker Job Options Parameters*

The CellMaker is initialised as shown in figure A.2. The available job options parameters are listed in table A.4 and in the standard release are supplied with ATLAS specific default values. The execution sequence is shown in figure A.3.

The cells are smeared according to Gaussian distributions, dependent on the granularity of the detector calorimeter cells(i.e. whether the cells are in the forward or barrel region).

$$\frac{\delta E}{E} = \frac{0.50}{\sqrt{E}} + 0.03 \quad \text{for } \Delta\eta \times \Delta\phi = 0.1 \times 0.1 \tag{A.12}$$

$$\frac{\delta E}{E} = \frac{1.0}{\sqrt{E}} + 0.07 \quad \text{for } \Delta\eta \times \Delta\phi = 0.2 \times 0.2 \tag{A.13}$$

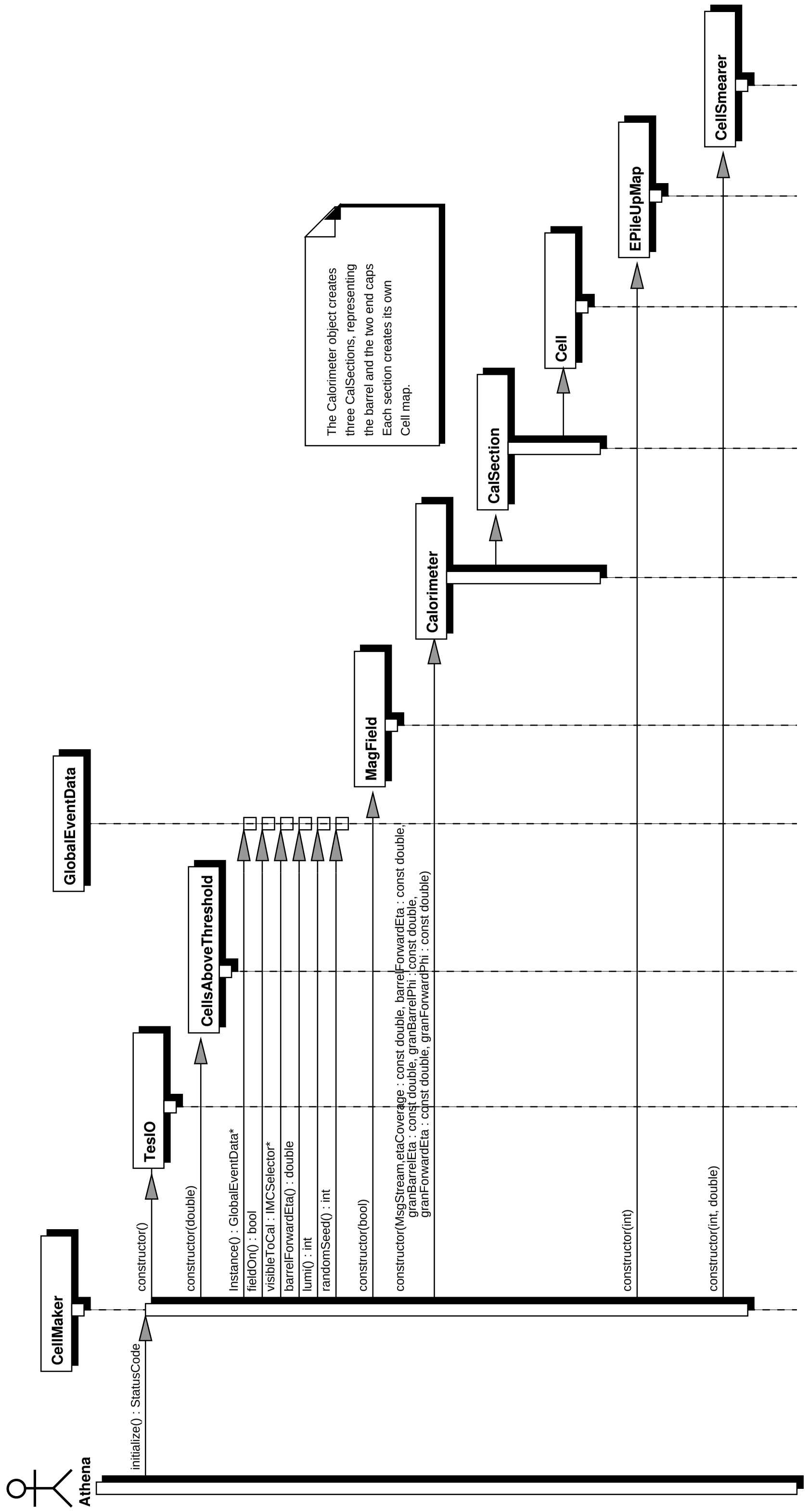


Figure A.2: *The CellMaker Initialisation Sequence Diagram*

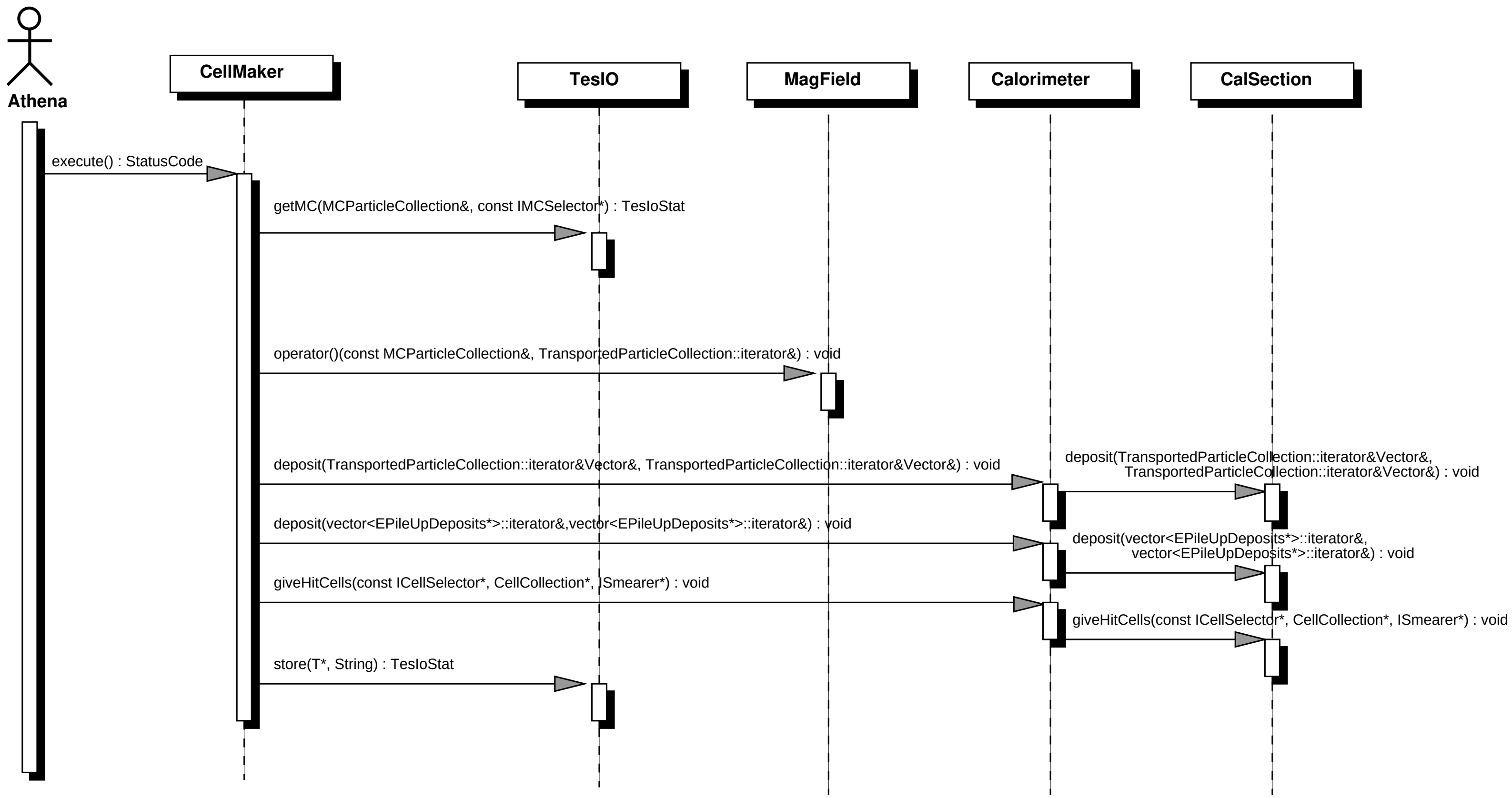


Figure A.3: *The CellMaker Execution Sequence Diagram*

A.5 ClusterMaker

This appendix shows the sequence diagrams and job options parameters for the ClusterMaker algorithm.

Property	**type**	**Description**
RConeBarrel	double	Cone size for clustering in barrel region
RConeForward	double	Cone size for clustering in end-cap region
minInitiatorET	double	Minimum required E_T for an initiator Cell
minClusterET	double	Minumum E_T acceptance for a Cluster
Strategy	string	Strategy choice flag for IClusterStrategy
InputLocation	string	TES location of input Cell objects
OutputLocation	string	TES location for Cluster storage
UnusedCellLocation	string	TES location for unused Cell storage
MasslessJets	double	Determines whether Cluster masses are zero or not

Table A.5: *ClusterMaker Job Options Parameters*

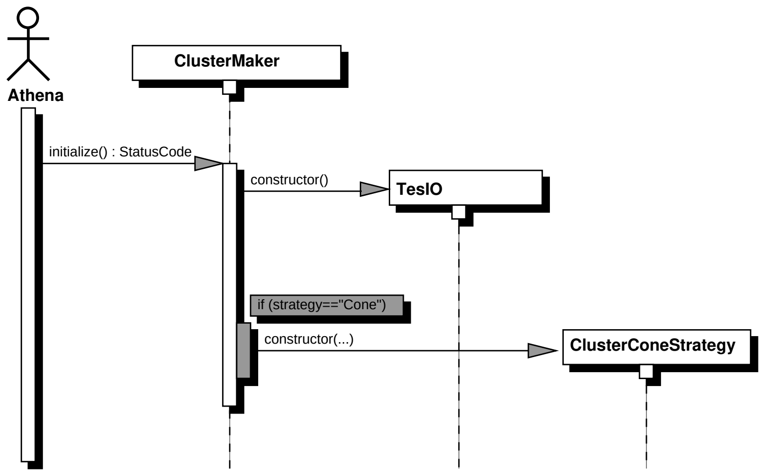


Figure A.4: *The ClusterMaker Initialisation Sequence Diagram*

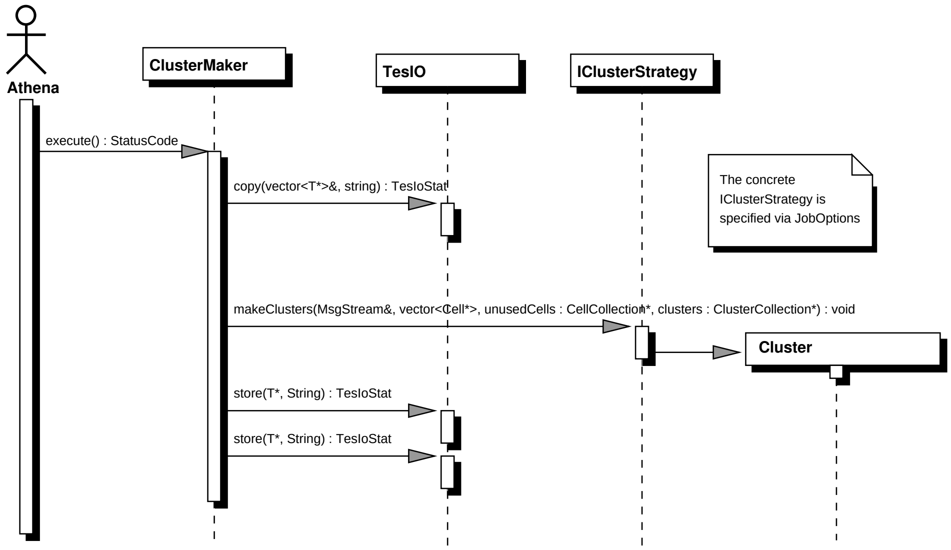


Figure A.5: *The ClusterMaker Execution Sequence Diagram*

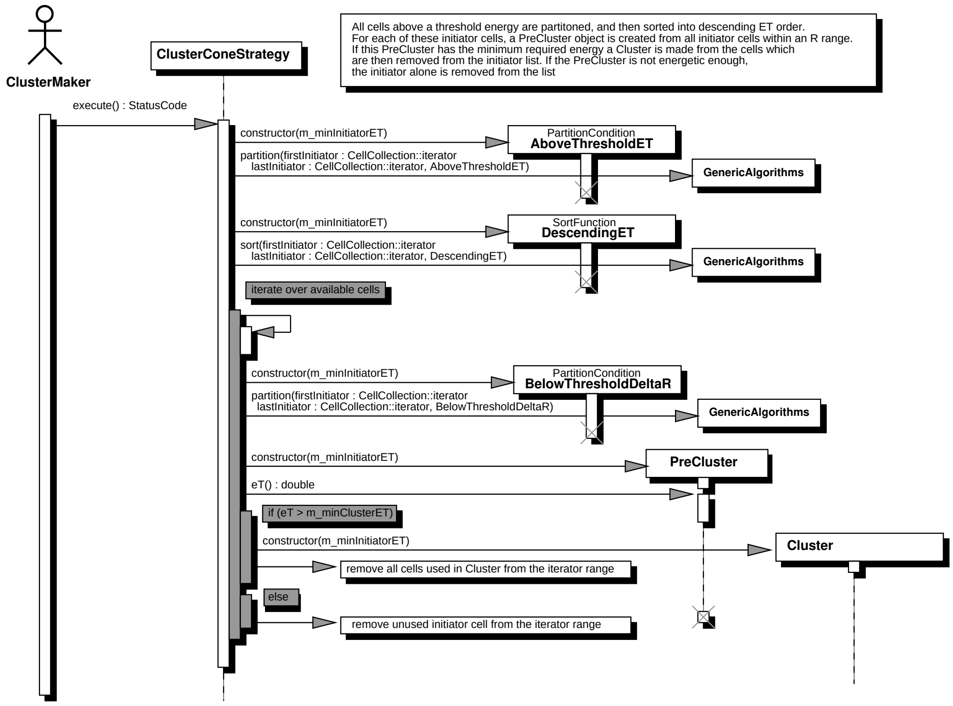


Figure A.6: *The ClusterConeStrategy Sequence Diagram*

A.6 JetMaker

This appendix shows the sequence diagrams and job options parameters for the JetMaker algorithm, and describes the smearing parameterisation used by the JetSmearer.

Property	**type**	**Description**
MinimumPT	double	Minimum p_T of particle to make track
MaximumEta	double	Maximum η of particle to make track
DoSmearing	bool	Smearing switch
RConeB	double	Cone size for Jets in barrel region
RConeF	double	Cone size for Jets in end-cap regions
bPtMin	double	Minimum b-quark p_T acceptance for b-tagging
bMaxDeltaR	double	Maximum Jet-quark dR for b-tagging
cPtMin	double	Minimum c-quark p_T acceptance for c-tagging
cMaxDeltaR	double	Maximum Jet-quark dR for c-tagging
tauPtMin	double	Minimum tau p_T acceptance for tau-tagging
tauMaxDeltaR	double	Maximum Jet-lepton dR for tau-tagging
etaTagMax	double	Maximum absolute Jet η value for tagging
tauJetPtRatio	double	Minimum acceptable p_{T_τ}/j_T ratio for tau-tagging
InputLocation	string	TES location for input Clusters
OutputLocation	string	TES location for Jet storage
UnusedCellLocation	string	TES location for input Unused Cells
MuonLocation	string	TES location for input Muons
McTruthLocation	string	TES location of Monte Carlo record

Table A.6: *JetMaker Job Options Parameters*

The JetMaker is initialised as shown in figure A.7. The available job options parameters are listed in table A.6 and in the standard release are supplied with ATLAS specific default values. The execution sequence is shown in figure A.8.

The Jet smearing is identical to the cell smearing parameterisation except for the addition of a pile up coefficient $\sigma_{pile-up}$ which is dependent on the

$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ cone size of the jet finder. This is therefore only relevant for a cone algorithm and is in normally de-activated in favour of the more generic Cell smearing.

$$\frac{\delta E}{E} = \frac{0.50}{\sqrt{E}} + \frac{\sigma_{pile-up}}{P_T} + 0.03 \quad \text{for } \Delta\eta \times \Delta\phi = 0.1 \times 0.1 \tag{A.14}$$
$$\frac{\delta E}{E} = \frac{1.0}{\sqrt{E}} + \frac{\sigma_{pile-up}}{P_T} + 0.07 \quad \text{for } \Delta\eta \times \Delta\phi = 0.2 \times 0.2 \tag{A.15}$$

$$\sigma_{pile-up} = 0.4 \quad \text{for } \Delta R < 0.4 \tag{A.16}$$
$$\sigma_{pile-up} = 7.5 \quad \text{for } \Delta R = 0.4 \tag{A.17}$$
$$\sigma_{pile-up} = 12.0 \quad \text{for } 0.4 < \Delta R < 0.5 \tag{A.18}$$
$$\sigma_{pile-up} = 18.0 \quad \text{for } 0.5 < \Delta R < 0.7 \tag{A.19}$$
$$\sigma_{pile-up} = 20.0 \quad \text{for } \Delta R > 0.7 \tag{A.20}$$

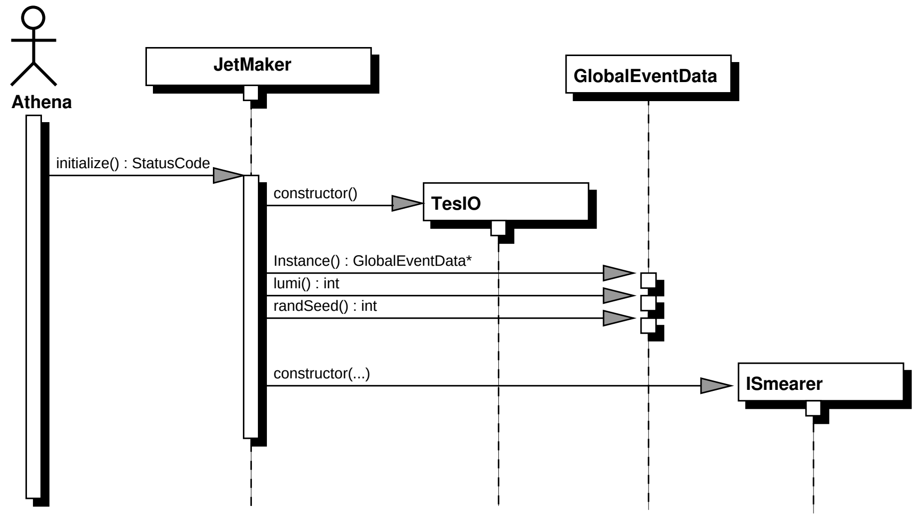


Figure A.7: *The JetMaker Initialisation Sequence Diagram*

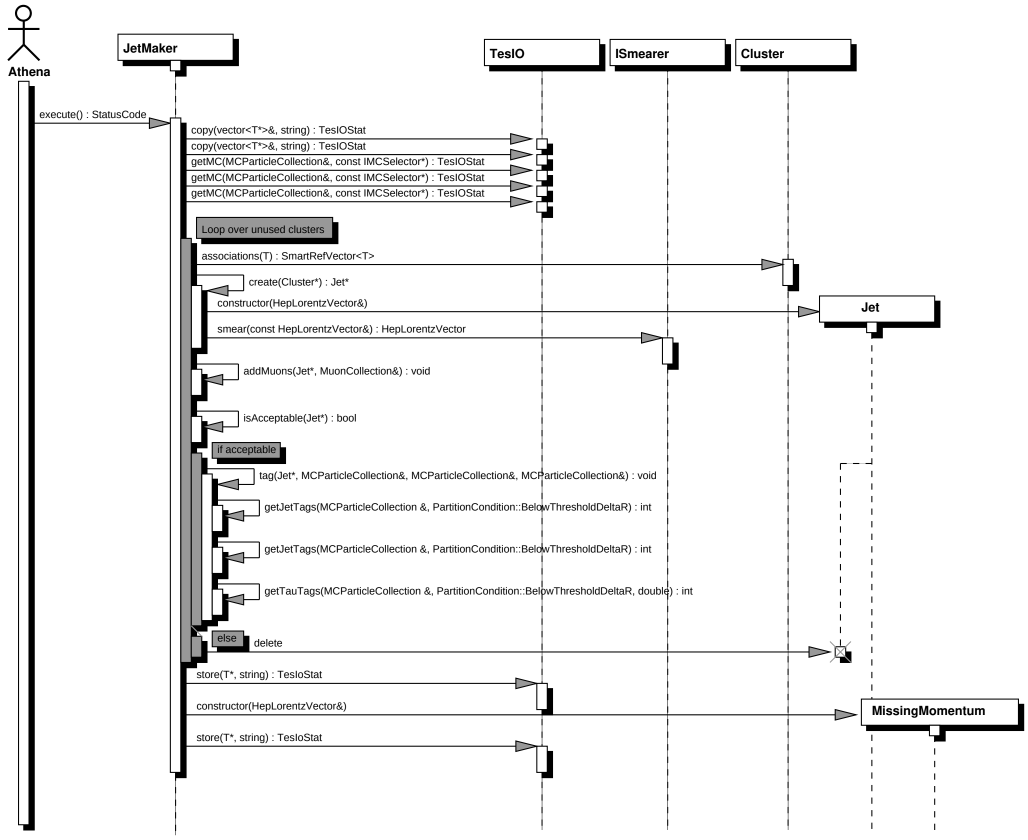


Figure A.8: *The JetMaker Execution Sequence Diagram*

A.7 TrackMaker

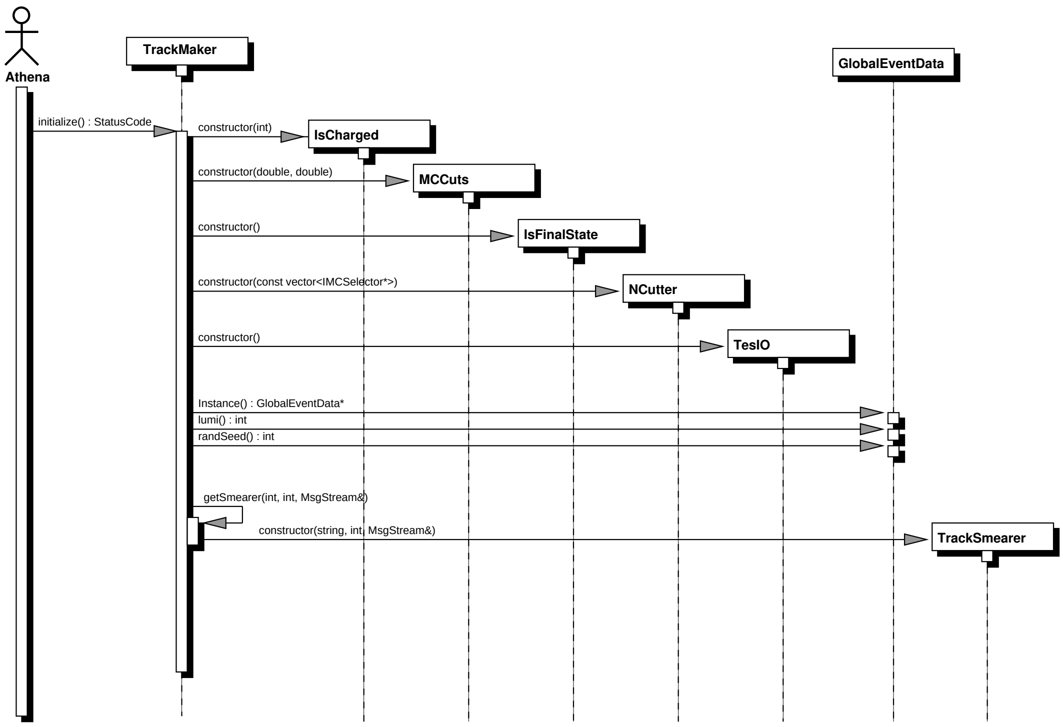
This appendix shows the sequence diagrams and job options parameters for the TrackMaker suite, including the smearer subclasses.

Property	**type**	**Description**
McPtMinimum	double	Minimum p_T of particle to make track
McEtaMaximum	double	Maximum η of particle to make track
SmearedPtMinimum	double	p_T acceptance for smeared track
SmearedEtaMaximum	double	η acceptance for smeared track
vlMaximum	double	Maximum longitudinal vertex
vtMaximum	double	Maximum transverse vertex(leptons only)
DoSmearing	bool	Smearing switch
Bfield	double	Tesla value of magnetic field
MC_eventLocation	string	TES location of Monte Carlo data
OutputLocation	string	TES location for track storage
MuonConfiguration	string	Configuration key for muon smearing

Table A.7: *TrackMaker Job Options Parameters*

The TrackMaker is initialised and executed as shown in figure A.9. The available job options parameters are listed in table A.6 and in the standard release are supplied with ATLAS specific default values.

The TrackSmearer class creates three IMatrixManager classes, one for Electrons, one for Muons, and one for Pions. The Electron and Muon classes are extensions of the LeptonMatrixManager class. Figure A.10 shows the initialisation diagram for the LeptonMatrixManagers which are instantiated with a three-bit configuration code representing the detector configuration(see table A.8).



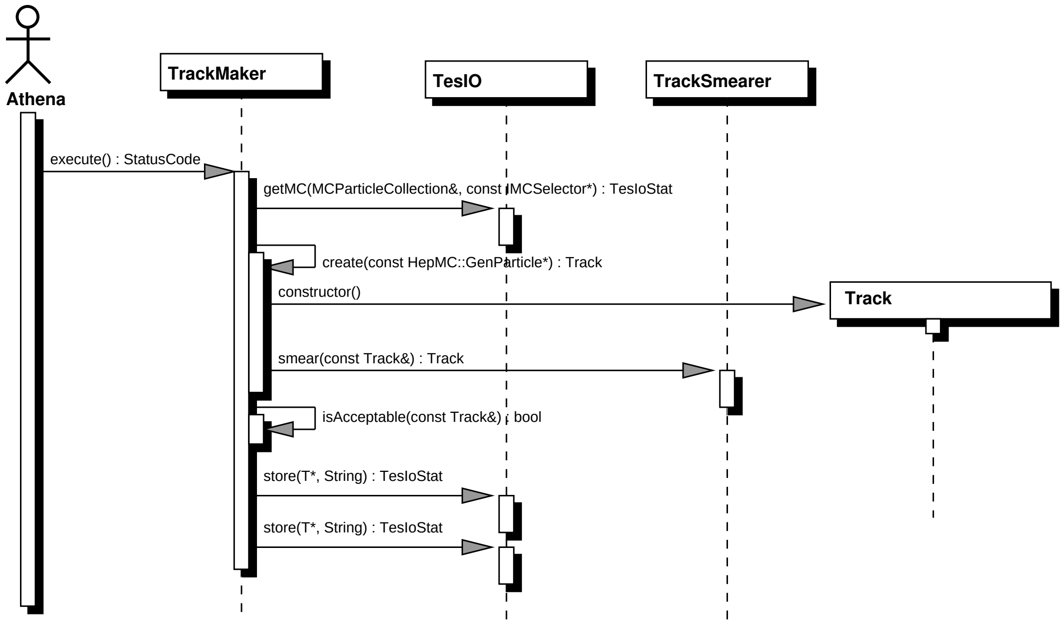


Figure A.9: *The TrackMaker Initialisation and Execution Sequence Diagrams*

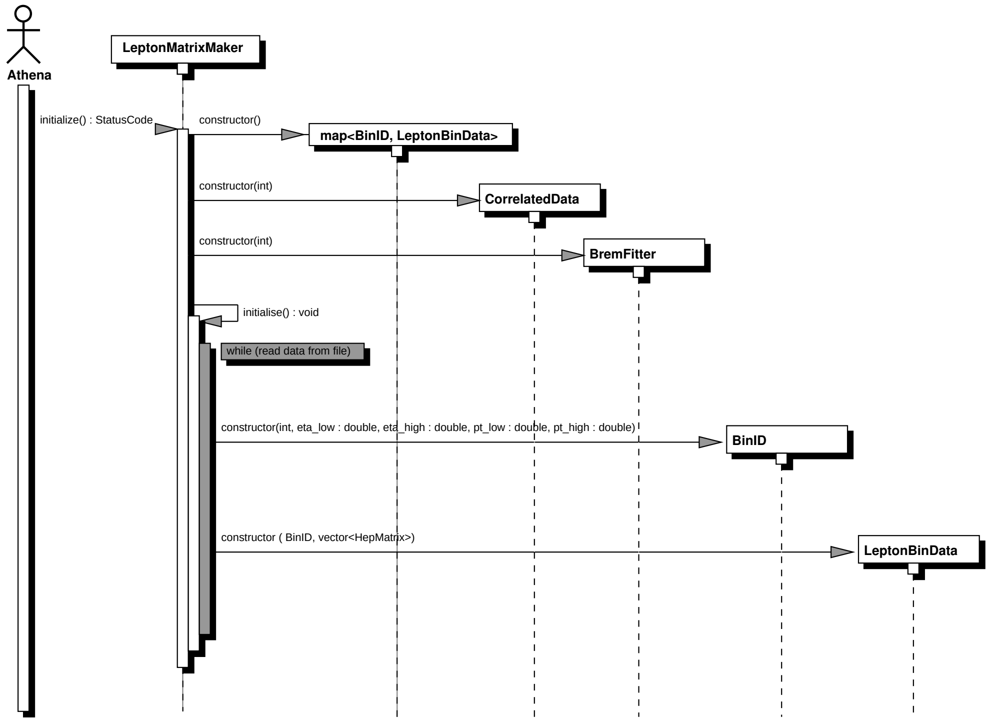


Figure A.10: *The Initialisation Sequence Diagram for the MatrixManager class*

	BIT 1	BIT 2	BIT 3
0	B-Layer Geometry	Realistic Solenoidal B-Field	Vertex Constraint
1	No B-Layer Geometry	Ideal Constant 2T B-Field	No Vertex Constraint

Table A.8: *LeptonMatrixManager Configuration*

Figure A.11 shows the detailed execution sequence of the TrackSmearer class. Depending on the particle id of the track, the smear parameters are requested from the relevant Matrix Manager. If the ElectronMatrixManager is chosen, the track is smeared according to soft p_T scaling parameterisations. From this point, the execution sequences for each matrix manager are the same.

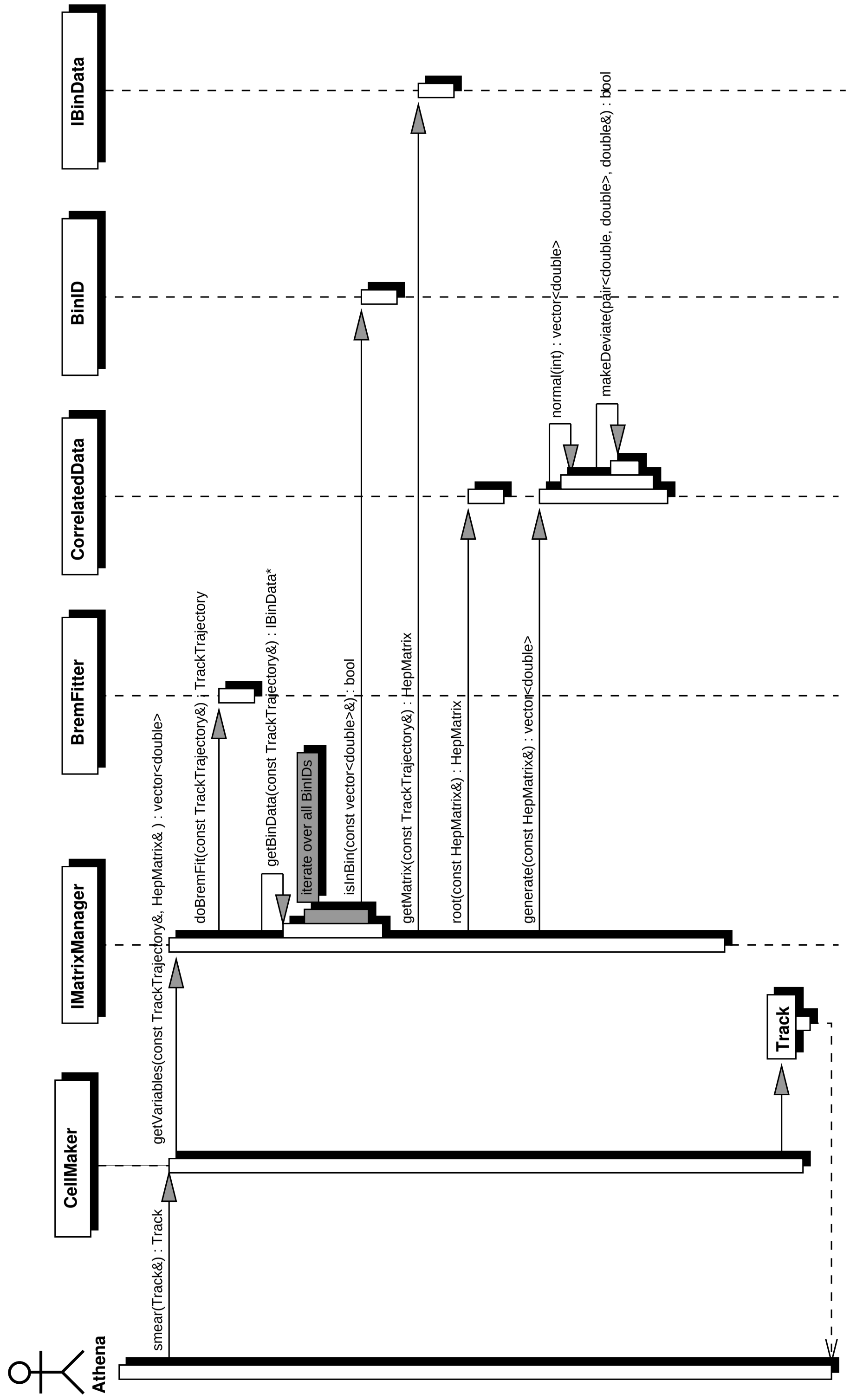


Figure A.11: *The MatrixManager Execution Sequence. This is the same for all three matrix managers apart from the inclusion of the BremFitter class, which is only present in the ElectronMatrixManager*

Figure A.12 shows the execution sequence of the BremFitter class which is responsible for soft p_T scaling of electron tracks. It is passed a TrackTrajectory object to smear and, using the trajectory's pseudorapidity, chooses the relevant BremEtaBin object which smears the p_T, impact parameter, and azimuthal angle of the track.

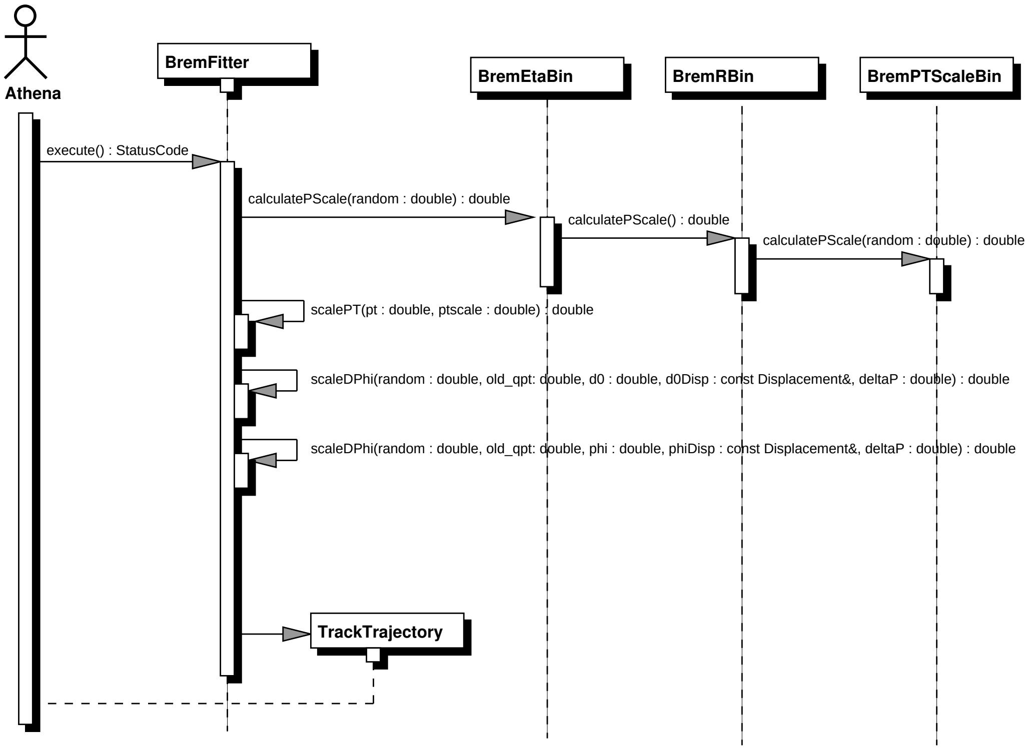


Figure A.12: *The BremFitter Class Execution Sequence*

Appendix B

AtlfastAnalysis Implementation Detail

This appendix describes the detailed implementation of the AtlfastAnalysis classes, including the initialisation and execution sequence diagrams, job options steering parameters, and customisation details.

B.1 Entity Selection

The entity selector concrete implementations(IEntitySelector, IEntityID) are service classes which select AnalysisEntity objects from the AnalysisEvent. The execution sequences of the classes used by AnalysisVariables(EntityID and EntityExtractor) are shown in figure B.1.

IEntitySelector concrete instances are aggregated by the EntityID objects and do the work of retrieving the AnalysisEntities on their behalf. Table B.1 lists both the basic concrete EntitySelectors and the classes which take IEntitySelectors as arguments and extend the versatility of object retrieval.

Elaborate combinations of the above classes are then aggregated by the

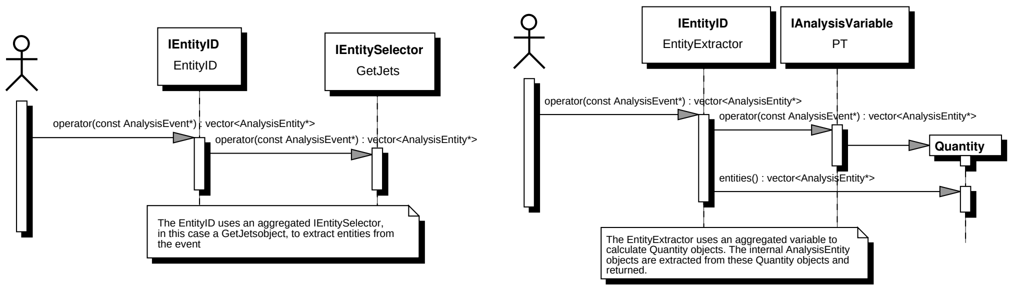


Figure B.1: *EntityID Execution Sequences*

EntityID objects via the IEntitySelector interface. These EntityIDs are stored in a map when the IDManager is created. EntityID objects are retrieved from the IDManager when AnalysisVariables are instantiated using a string id key. The default keys, which can be supplemented by the user, are shown in table B.2.

IEntitySelector	**Description**
GetNothing	returns empty vector
GetLeptons	returns isolated leptons
GetNILeptons	returns non-isolated leptons
GetPhotons	returns isolated photons
GetNIPhotons	returns non-isolated photons
GetJets	return calibrated jets
GetUncalibratedJets	return uncalibrated jets
GetMC	returns particles from event record
GetFlavour	filters aggregated IEntitySelector for pdg ID
GetAllButFlavour	filters aggregated IEntitySelector for pdg ID
GetBTags	filters aggregated IEntitySelector for pdg ID 5
GetTruth	extracts HepMC truth from AnalysisEntities
GetFourVectorSum	calculates four-vector sum of AnalysisEntities
GetBoosted	boosts AnalysisEntities into another frame of reference
MultiEntitySelector	returns all AnalysisEntities multiple IEntitySelectors

Table B.1: *Standard EntityID Objects*

String ID	**Description**
NoEntities	dummy entities
Leptons	Atlfast isolated leptons
NILeptons	Atlfast isolated leptons
Electrons	Atlfast isolated electrons
NIElectrons	Atlfast non-isolated electrons
Muons	Atlfast isolated muons
NIMuons	Atlfast non-isolated muons
Photons	Atlfast isolated photons
NIPhotons	Atlfast non-isolated photons
Jets	Atlfast calibrated jets
BJets	Atlfast calibrated b-jets
UncalibratedJets	Atlfast uncalibrated jets
UncalibratedBJets	Atlfast uncalibrated b-jets
Tracks	Atlfast tracks

Table B.2: *IDManager map entry keys with descriptions*

B.2 AnalysisVariables

This appendix presents details of the AnalysisVariable Classes. Table B.3 shows the concrete implementations for all five types of AnalysisVariable and Operators, and example sequence diagrams are presented for the UnaryVariable, BinaryVariable, UnaryOperator and BinaryOperator classes.

UnaryVariable	**Description**
PT	transverse momentum of entities
Eta	pseudorapidity η of entities
Phi	azimuthal angle ϕ of entities
Multiplicity	returns multiplicity of entities
Mass	invariant mass of each of a group of entities
TotalMass	invariant mass of a total group of entities
ImpactParameter	track impact parameters
CotTheta	track cot thetas
ZPerigee	track z-perigees
InvPTCharge	track inverse pT charges
Radius	track radii(not curvature)
PairVariable	**Description**
DEta	pseudorapidity distance $\Delta\eta$ between pairs of entities
DPhi	azimuthal angular distance $\Delta\phi$ between pairs of entities
DR	distance $\Delta R = (\sqrt{\Delta\eta^2 + \Delta\phi^2})$ between pairs of entities
PairMass	invariant mass of pairs of entities
CosTheta	cosine of the angle between pairs of entities
EventVariable	**Description**
PTMiss	missing transverse momentum of event
Constant	constant value for use in BinaryOperators
UnaryOperator	**Description**
Min	minimum value of a variable
Max	maximum value of a variable
Absolute	absolute values of entities
BinaryOperator	**Description**
ClosestTo	value of a variable closest to a target
InRange	values of a variable within a given range
OutsideRange	values of a variable without a given range
Product	product of two variables
Quotient	quotient of two variables
Sum	sum of two variables
Subtract	difference between two variables

Table B.3: *Concrete IAnalysisVariable Classes*

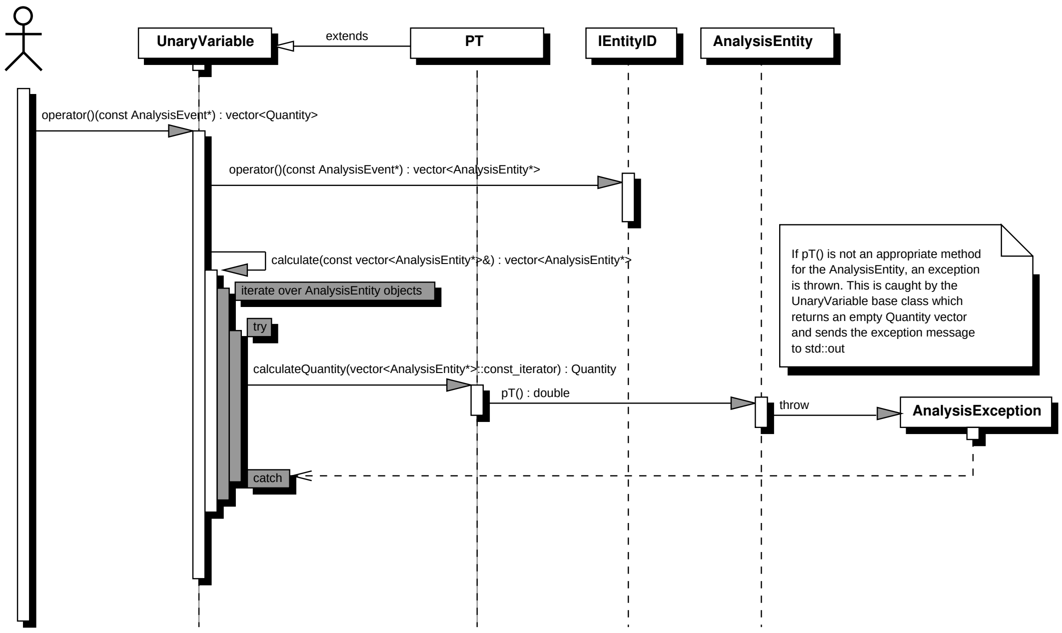


Figure B.2: *The UnaryVariable Calculation Sequence Diagram for the PT class*

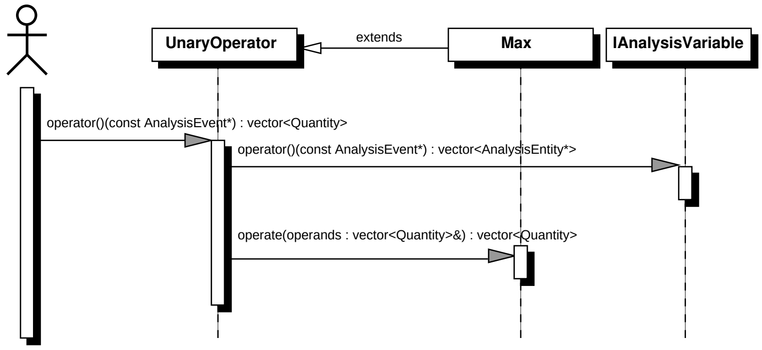


Figure B.3: *The UnaryOperator Calculation Sequence Diagram*

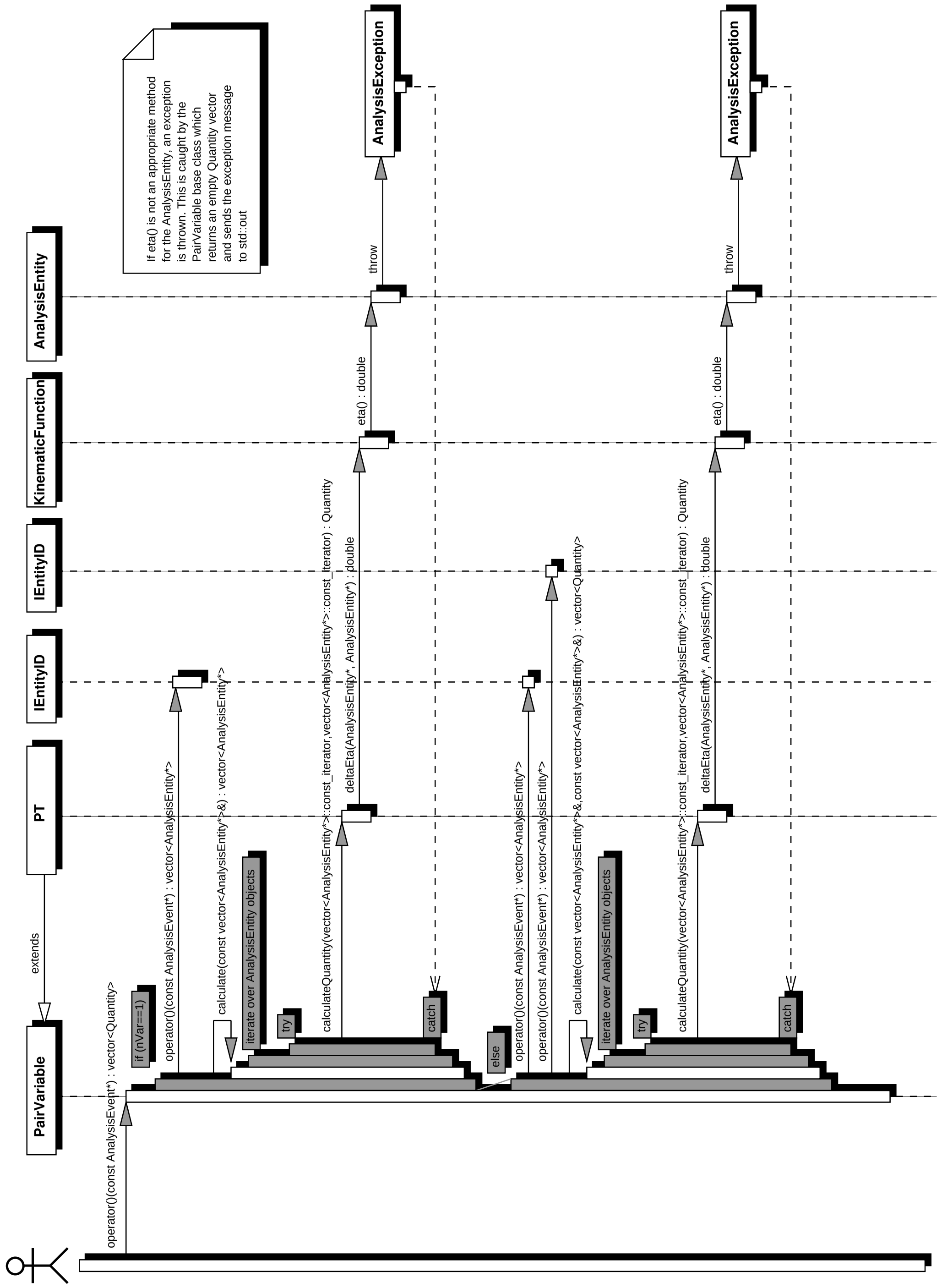


Figure B.4: *The PairVariable Calculation Sequence Diagram*

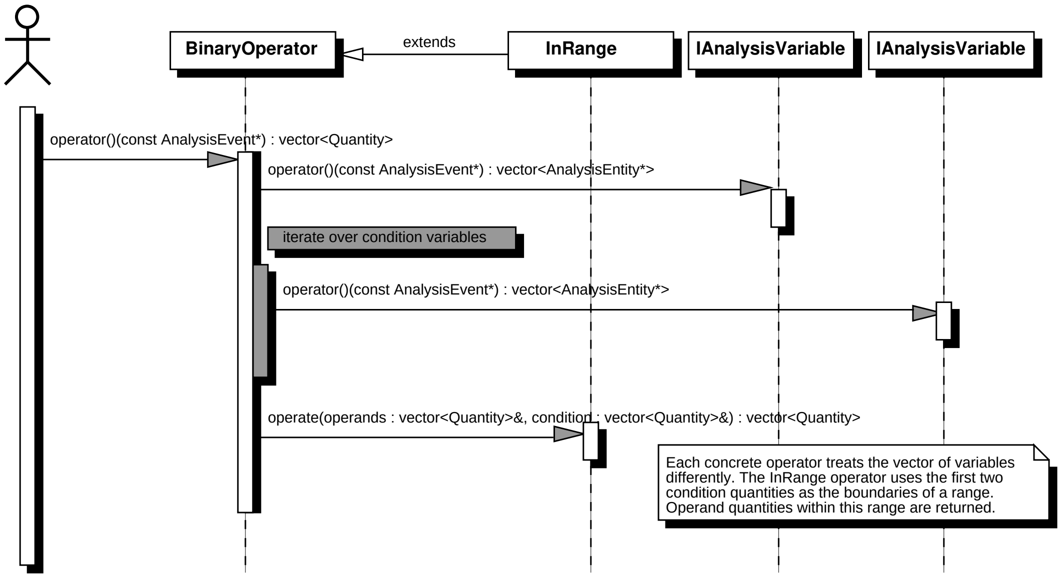


Figure B.5: *The Binaryoperator Calculation Sequence Diagram*

B.2.1 Instantiating Variables, Cuts, and Histograms

This appendix describes the class execution and syntax used for creating variables, cuts, and histograms in an analysis via the job options mechanism. Three Reader classes are responsible for interpreting instructions from the job options file and creating variables, cuts, and histograms. This run-time specification allows quick and easy adjustment of analysis parameters.

Interpreting Variables

The execution sequence for the VariableReader class is shown in figure B.6. It interprets a command vector from the job options steering file and creates the relevant variables according to the following syntax.

For UnaryVariables, the name of the variable is declared first, followed by the IEntityID string identifier which corresponds to the required AnalysisEntities, separated by a single white space.

For PairVariables, the name of the variable is declared first, followed by either one or two IEntityID string identifiers, separated by single white spaces. If the second argument doesn't exist, or is not an IEntityID, the variable is therefore instantiated with one IEntityID.

EventVariables are instantiated with simply the name of the variable, except for the Constant variable, which is followed by the type(int/double) and a value.

To instantiate UnaryOperators, the operator name is declared first, followed by the relevant operand variable declaration in the same string.

To instantiate BinaryOperators, the operator name is declared first, followed by the relevant number of operand variable declarations(depending on operator) in the same string. The ClosestTo operator is declared with a test variable followed by a target variable; the range operators are declared with a test variable, and two boundary edge variables; Product, Quotient, Sum and Subtract are all instantiated with two IAnalysisVariables.

The EntityExtractor class can also be interpreted by the variable reader, replacing an IEntityID identifier, allowing more complex filtering of AnalysisEntities.

Variable type	**Example declaration**
EventVariable	"PTMiss"
EventVariable	"Constant int 2"
UnaryVariable	"Multiplicty Jets"
PairVariable	"PairMass Leptons"
PairVariable	"DR Jets Leptons"
UnaryOperator	"Max PT Jets"
BinaryOperator	"ClosestTo PairMass Leptons Constant double 91.1"
EntityExtractor	"Eta Extract MaxPT Leptons"

Table B.4: *Example AnalysisVariable JobOptions Declarations*

Table B.4 shows examples for each variable type. Complex variables can therefore be constructed using combinations of the given variable types, for example the decalaration

"Max PT Extract InRange Eta Jets Min Eta Leptons Max Eta Leptons"

would create a variable to return the maximum transverse momentum of the jets which lie between the two most forward and backward leptons in pseudorapidity.

Interpreting Cuts

The Cuts are interpreted by the CutReader, whose sequence diagram is shown in figure B.7. It reads the job options command vector, interprets the cut type details and uses its inhereted VariableReader functionality to parse the variable on which the cut acts.

There are two types of cuts which can be parsed; single cuts, and multiple cuts. The single cuts are simplest and follow the pattern; cut name, cut type, cut variable, low edge, high edge, all separated by a single white space.

Multiple cuts are parsed over a number of command strings. The first string dictates the cut name and cut type. The following strings correspond to the constituents of the multicut, and adhere to the syntax of the individual cuts above, with the cut name absent. The last string contains an end delimeter flag.

Cut type	**Example declaration**
Single - LeptonPT	"LeptonPT InRangeAndCut PT Leptons +" "Constant double 20 Constant double 1000000"
Single - HighPTLepton	"HighPTLepton InRangeOrCut PT Leptons +" "Max PT Jets Constant double 1000000"
Multiple - Isolation	"Isolation MultiAndCut" "OutsideRangeAndCut DR Leptons +" "Constant double 0.0 Constant double 0.8" "OutsideRangeAndCut DR Jets +" "Constant double 0.0 Constant double 0.8" "OutsideRangeAndCut DR Leptons Jets +" "Constant double 0.0 Constant double 0.8" "EndMultiCut"

Table B.5: *Example AnalysisCut JobOptions Declarations*

Table B.5 shows examples for each cut type. The LeptonPT cut is passed if all leptons have $p^T > 20GeV$; the HighPTLepton cut is passed if any one lepton has a higher transverse momentum than the highest p^T jet; the Isolation cut is passed if all leptons and jets are separated by a $\Delta R = (\sqrt{\Delta\eta^2 + \Delta\phi^2})$ distance of at least 0.8.

Interpreting Histograms

The HistogramReader interprets the analysis histograms. The execution sequence is shown in figure B.8 and uses its inhereted VariableReader functionality as well as an aggregated CutReader to interpret the histograms.

Histogram management involves plotting histograms as well as filtering. The first item to be declared when setting the histograms is the global filter. The filter is declared with a "GlobalFilter" string flag, followed by a cut.

The histograms themselves fall into two categories, one and two dimensional. The one dimensional histograms are declared using the following pattern; dimension, title, number of bins, range and variable. The two dimensional histograms follow a similar pattern; dimension, title, number of bins in x, x axis range, number of bins in y, y axis range, and one or two variables

Each histogram can also have a filter of its own which is declared immediately after the histogram in a separate string, using a "Filter" string delimiter flag

Declaration type	**Example declaration**
Global filter	"GlobalFilter" "Multiplicity Leptons Constant int 2 Constant int 2"
1D histogram	"1D Lepton_PT 100 0 150 PT Leptons"
1D filtered histogram	"1D Lepton_Mass 100 0 150 PairMass Leptons" "Filter" "InRangeAndCut PT Jets +" "Constant double 30.0 Constant double 1000000"
2D histogram	"2D Jet_eta/phi 50 -3 3 50 -3.2 3.2 Eta Jets Phi Jets"

Table B.6: *Example AnalysisHistogram JobOptions Declarations*

Table B.6 shows example histogram delcarations. The global filter ensures that only events with 2 leptons will be considered for histogramming. The

Lepton_PT histogram will display the momentum of both leptons up to 150 GeV with 100 bins; the Lepton_Mass histogram displays the invariant mass of the two leptons for events where both leptons have $p^T > 30GeV$; the Jet_eta/phi histogram will display a 2D plot of jet pseudorapidity versus azimuthal angle.

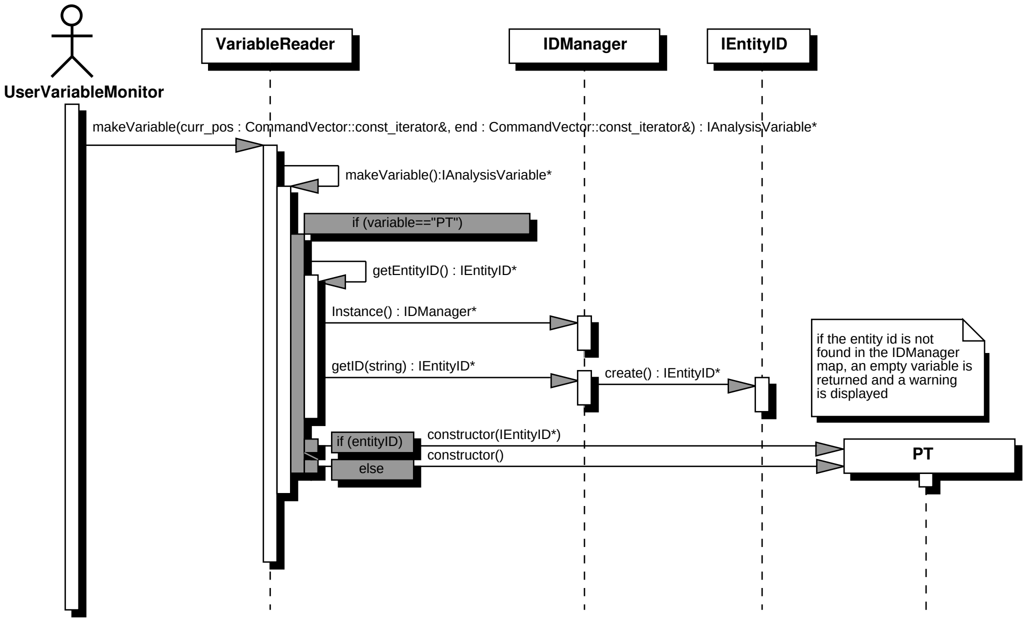


Figure B.6: *The VariableReader Operation Sequence Diagram*

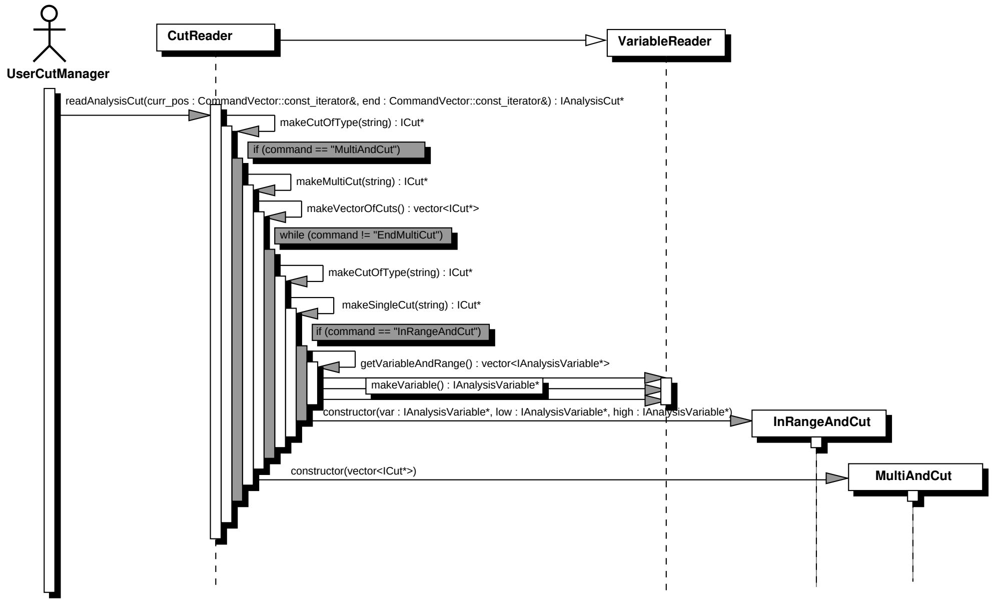


Figure B.7: *The CutReader Operation Sequence Diagram*

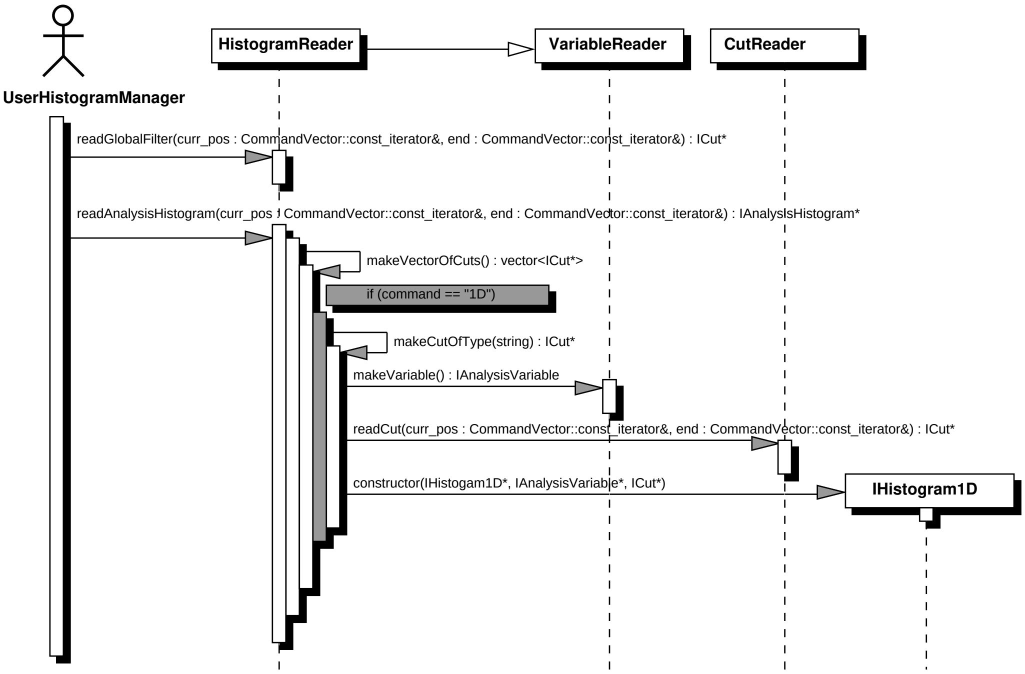


Figure B.8: *The HistogramReader Operation Sequence Diagram*

B.3 AtlfastAnalysis Execution

This appendix describes the implementation details of the OOAnalyser algorithm which controls the execution of the analysis, and the manager class sequence diagrams for applying cuts and plotting histograms

Property	**type**	**Description**
MonitorFrequency	int	Frequency of event number output
ReadoutVariables	int	Number of events to monitor
TracksOn	bool	ignore or read tracks from TDS
CutCommands	vector<string>	cut command vector
HistogramCommands	vector<string>	histogram command vector
VariableCommands	vector<string>	variable command vector

Table B.7: *OOAnalyser steering parameters*

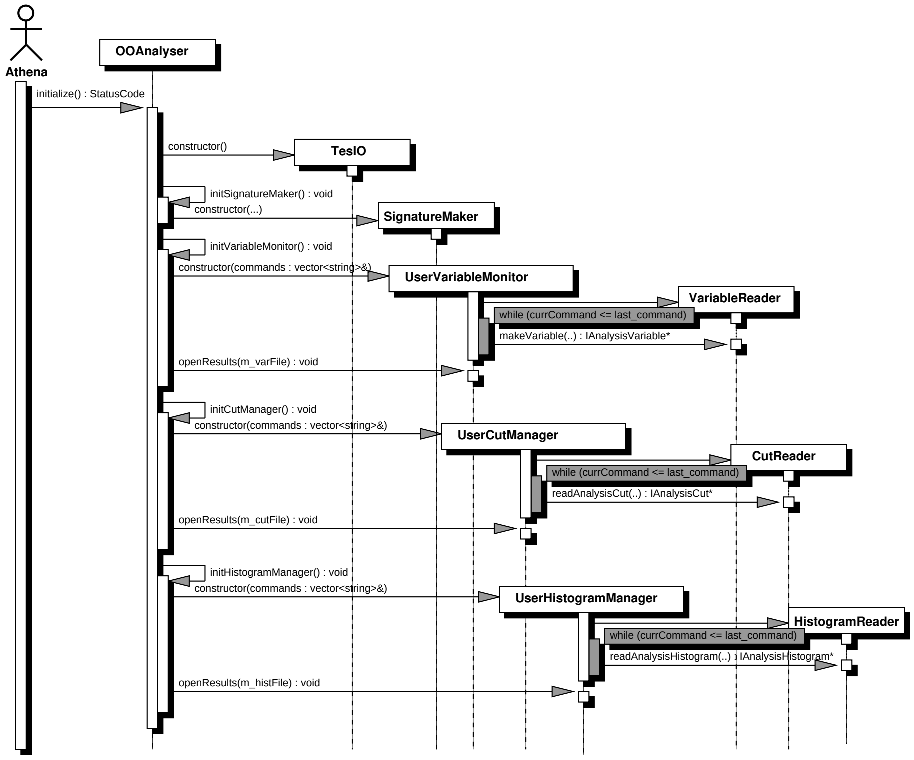


Figure B.9: *The OOAnalyser Initialisation Sequence Diagram*

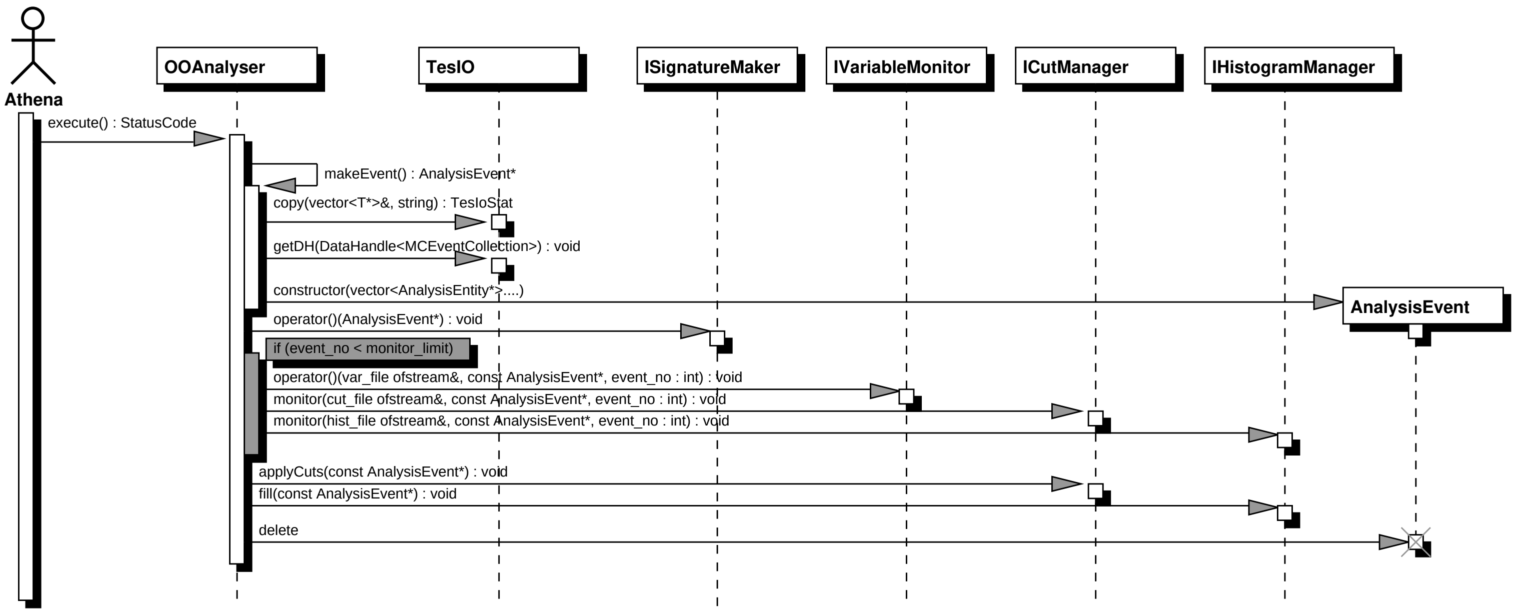


Figure B.10: *The OOAnalyser Execution Sequence Diagram*

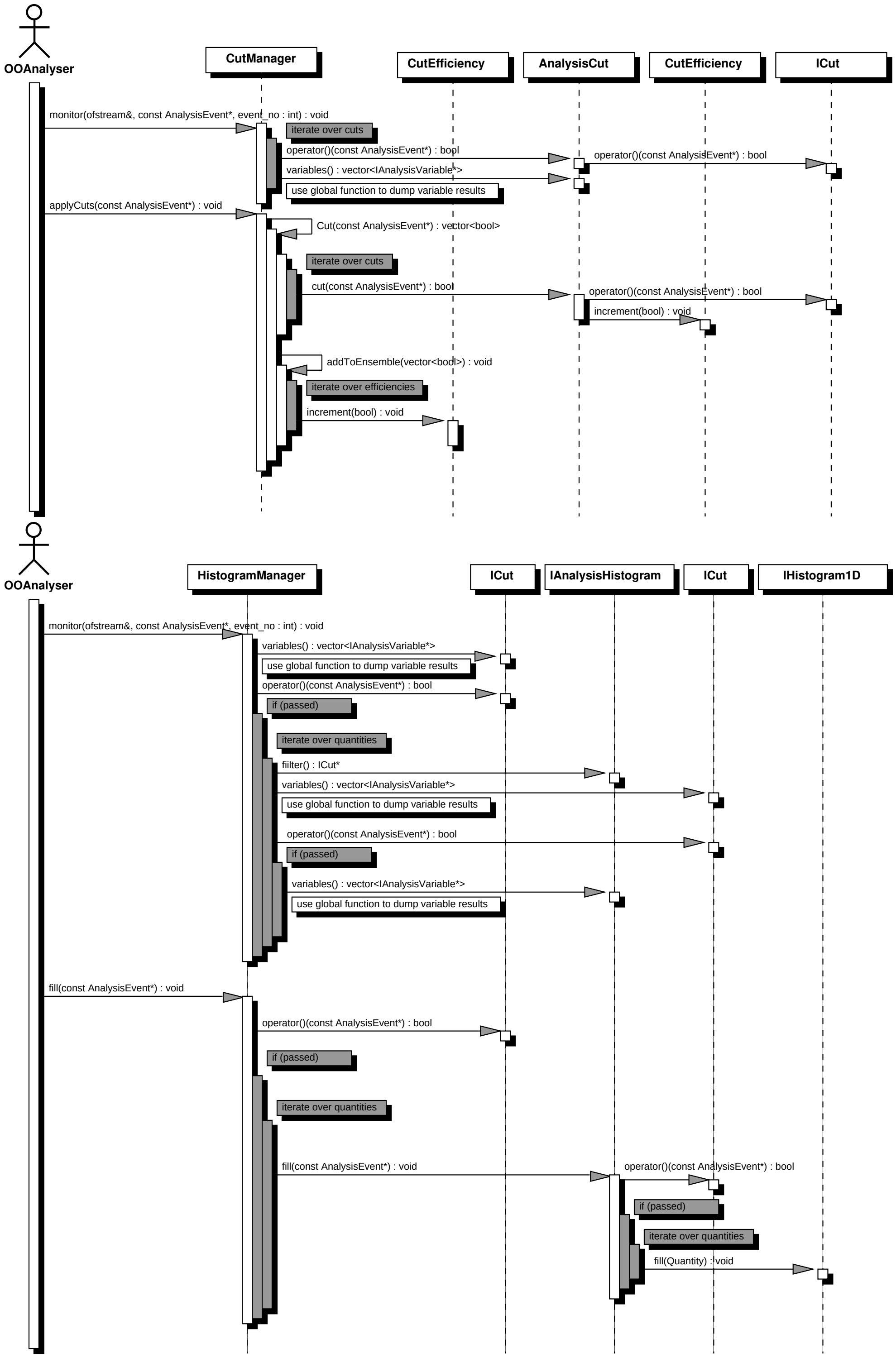


Figure B.11: *The CutManager and HistogramManager Execution sequences diagrams*

B.3.1 Customising the Analysis Package

This appendix describes the way in which the analysis package can be customised using compiled modifications of the signature classes, entity selectors and variables.

The Signature class is the user-defined container class for special or frequently selected AnalysisEntities or event quantities, such as signal event topologies. The Signature is created every event by the SignatureMaker class.

Figure B.12 shows the diagram for the simple user example provideed in the release. The signature is defined here as all leptons and jets above a momentum threshold, and two high pT 'tag' jets. New IEntitySelector classes are provided with which to extract these quantities from the Signature.

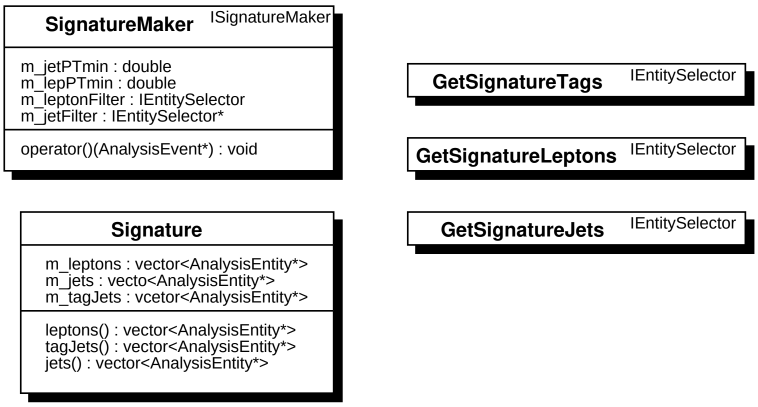


Figure B.12: *Example User Signature*

The signature maker is constructed by the OOAnalyser algorithm, and is instantiated with a lepton and jet minimum momentum value, set in the OOAnalyser's job options. It aggregates a number of IEntitySelectors which act as filters for the leptons and jets, using these values.

Every event the SignatureMaker creates the Signature object which is added to the AnalysisEvent. The concrete IEntitySelector classes are provided to retrieve the signature entities for use in AnalysisVariables and AnalysisHistograms. EntityIDs for each selector are entered in the IDManager's IEntityID map so that they can be instantiated by the VariableReader.

User-Defined IAnalysisVariable instances can also be constructed, using the AnalysisVariable base classes. The VariableReader class method makeUserVariable must be modified for each new variable to allow the variable to be read from job options. The example supplied is a UnaryVariable class, shown in figure B.13, which calculates the transverse mass B.1 of a four vector X and is instantiated with an IEntityID* object.

$$M_T = \sqrt{2p_T^X \; p_{Tmiss}(1 - cos\Delta\phi)} \tag{B.1}$$

The three aggregated IAnalysisVariables are used internally to calculate the result. Although, any IEntityID object can be used by this variable, it is intended for calculation using single four vector sums, such as a lepton system.

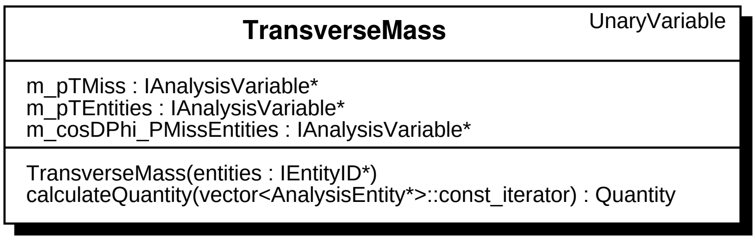


Figure B.13: *Example User AnalysisVariable: TransverseMass*

B.4 Likelihood Implementation

This appendix shows the sequence diagrams for the initialisation and execution of both ILikelihoodManager instances which calculate uncorrelated and correlated event likelihoods

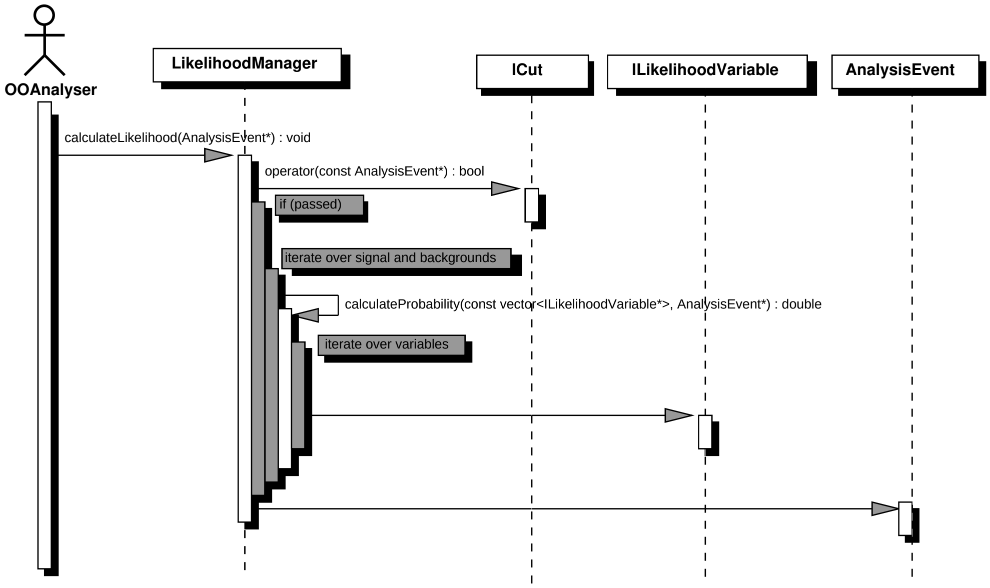


Figure B.14: *The LikelihoodManager Execution Sequence Diagram*

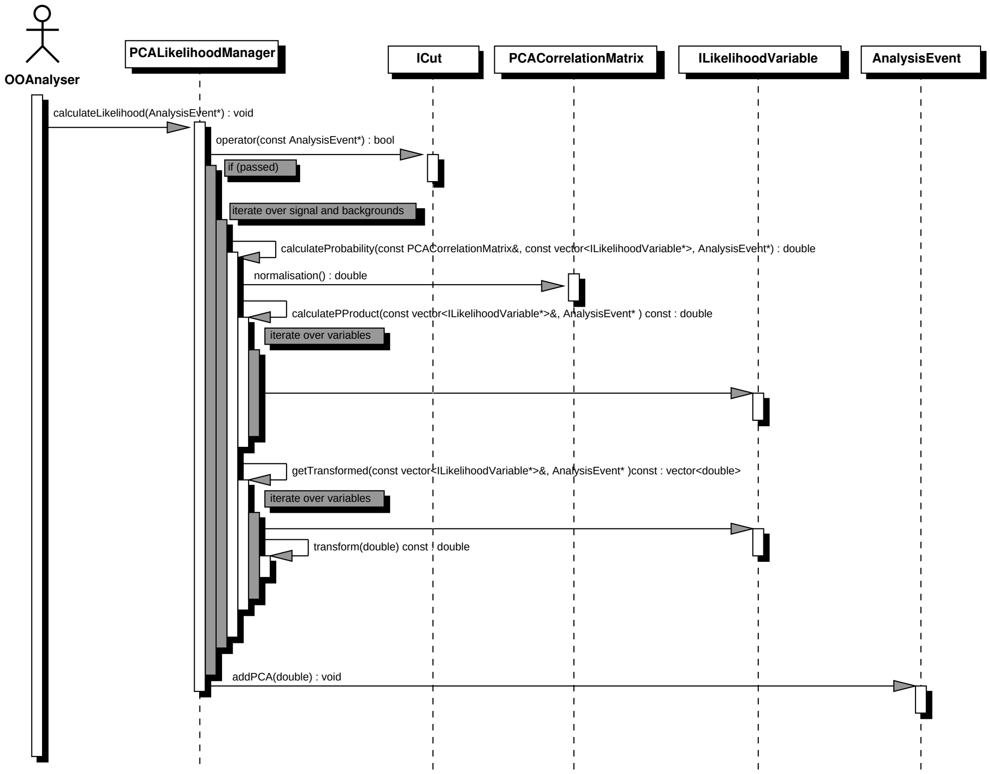


Figure B.15: *The PCALikelihoodManager Execution Sequence Diagram*

Appendix C

Statistical Tests

Two statistical tests are used to compare distributions of continuous histogrammed variables within this thesis; the Pearson's Chi Squared test and the Kolmogorov-Smirnov test.

C.1 Pearson's Chi Squared Test

The most common test statistic which reflects the agreement between two histograms of a continuous variable x, histogrammed within N bins, is the Pearson's χ^2 statistic. If the number of entries n_i in each bin are Poisson distributed with mean values ν_i, then the statistic C.1 will follow the χ^2 distribution C.2 for N degrees of freedom.

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \nu_i)^2}{\nu_i} \tag{C.1}$$

$$\begin{aligned} f(z;n) &= \frac{1}{2^{n/2}\Gamma(n/2)} z^{n/2-1} e^{-z/2}, n = 1, 2.... \\ \Gamma(x) &= \int_0^\infty e^{-t} t^{x-1}\ dt \end{aligned} \tag{C.2}$$

The χ^2 test is distribution free, as this holds regardless of the distribution of the variable x, as long as the bin entries $n_i \geq 5$ (i.e. they are approximately Gaussian distributed). The standard deviation of a Poisson variable is given by $\sqrt{\nu}$, thus the χ^2 statistic gives the sum of the squares of the deviations between observed and expected values, in units of the standard deviations. The significance level is given by the integral of the χ^2 distribution from the observed χ^2 to infinity.

$$P = \int_{\chi^2}^{\infty} f(z; n_d)\ dz \tag{C.3}$$

C.2 The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is another distribution independant test which can be used to compare two histograms. It is an exact test which does not depend on a large data sample, taking into account not only of the differences between corresponding bins, but also the sign of the difference, and in particular it is sensitive to a sequence of consecutive deviations of the same sign.

The test involves comparing the cumulative distributions of the two samples, and taking the maximum deviation C.4 of the two curves. This maximum can be used to formulate a test statistic C.5 which relates to a P-value determining the probability of the two samples being compatible.

$$D_{MN} = max|S_N(X) - S_M(X)|\ over\ all\ x \tag{C.4}$$

$$\sqrt{[MN/(M+N)]}D_{MN} \tag{C.5}$$