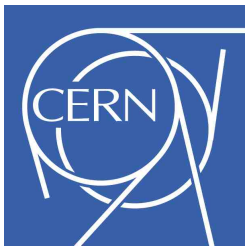

Etude des mésons $D^{*\pm}$ avec le détecteur ATLAS

Alexandre FAURE

sous la direction du Dr. David CALVET

Juin-Août 2010

Résumé

Ce rapport est issu des travaux effectués durant le stage de recherche effectué au Laboratoire de Physique Corpusculaire de Clermont-Ferrand (LPC) sur le sujet : *Etude des mésons* $D^{*\pm}$ *avec le détecteur ATLAS* sous la direction du Dr. David CALVET de fin juin à début août 2010. Les données utilisées sont issues des toutes premières collisions *p-p* d'une énergie de 7 TeV, dans le centre de masse, obtenues au Large Hadron Collider (LHC) du Centre Européen de la Recherche Nucléaire (CERN), situé à Genève, de mars à mai 2010.

Table des matières

Première partie

Introduction

Chapitre 1

Motivation

La physique des particules est actuellement en plein essor notamment due à la mise en service du *Large Hadron Collider* ou LHC en 2008. Malgré le retard des premières collisions à cause d'un incident technique dans le collisionneur [1], la communauté mondiale espère faire de nombreuses découvertes permettant de confirmer, ou d'infirmer, certaines théories actuelles. La recherche du fameux *boson de Higgs* est une des motivations de la création de ce collisionneur géant.

Le présent rapport est basé sur l'étude d'hadrons particuliers, les mésons $D^{*\pm}$, à l'aide d'un des détecteurs du LHC : le détecteur ATLAS. Le travail effectué s'inscrit dans la continuité des travaux effectués par M. Loïc Valéry qui a travaillé sur ce sujet dans le cadre du Master de Physique-Chimie à l'Université Blaise-Pascal de Clermont-Ferrand. Les programmes utilisés et modifiés de ce rapport sont issus à l'origine des premiers travaux qu'à effectué mon prédécesseur.

Nous réaliserons une étude encore plus poussée afin d'approfondir nos connaissances sur les mésons $D^{*\pm}$ et d'accroître la précision de l'analyse en optimisant la pureté du rapport signal sur bruit.

1. Une fuite d'hélium, qui permet le refroidissement des aimants supraconducteurs, a conduit à un retard du projet.

Chapitre 2

Le Modèle Standard

Le Modèle Standard est une théorie qui permet de décrire toutes les particules fondamentales et leur interactions : l'interaction électromagnétique, l'interaction faible et l'interaction forte. Elle a été élaboré dès les années 1950 à l'aide de la *Théorie Quantique des Champs* qui permet de relier deux branches majeures de la physique : la *Physique Quantique* et la *Relativité*. Après unification de l'interaction faible et la prise en compte de la théorie de jauge dans les années 1970, le Modèle Standard devint celui que l'on connaît actuellement.

Rutherford, pionnier de la description de particules atomiques avec sa célèbre expérience de diffraction qui porte son nom, a permis de mettre en évidence la subdivision de ce que l'on appelle *atome* (dont l'étymologie remonte au grec *atomos* qui signifie *brique*). Celui-ci est composé en réalité d'autres particules à savoir les protons et neutrons, qui forment le noyau atomique autour duquel "tournent" des électrons.

Depuis, les physiciens essayent de connaître toujours plus précisément les briques fondamentales de ces protons, neutrons et électrons en faisant notamment percuter les particules entre elles pour en extraire de possibles particules filles.

2.1 Particules de matières et particules de champs

On distingue en effet les particules qui composent la matière et celles qui composent les champs (quantiques).

Il existe douze particules de matière qui sont appelées les *fermions*. Leur particularité est d'avoir un spin demi-entier. Ils sont eux-même séparés en deux catégories :

- 12 Quarks et 12 Anti-Quarks : Les différentes combinaisons de ces particules permettent de former des *hadrons*.

- 6 Leptons et 6 Anti-Leptons.

Enfin, les particules de champs sont appelées *bosons* et sont considérés comme les vecteurs de transport des interactions. A la différence des fermions, leur spin est entier.

– 8 Gluons qui transmettent l'interaction forte.

– Les W^+, W^- et Z^0 qui transmettent l'interaction faible.

– Le Photon qui transmet l'interaction électromagnétique.

On obtient alors le tableau de la figure 2.1 qui dresse un véritable bilan des particules utilisées dans le cadre du Modèle Standard.

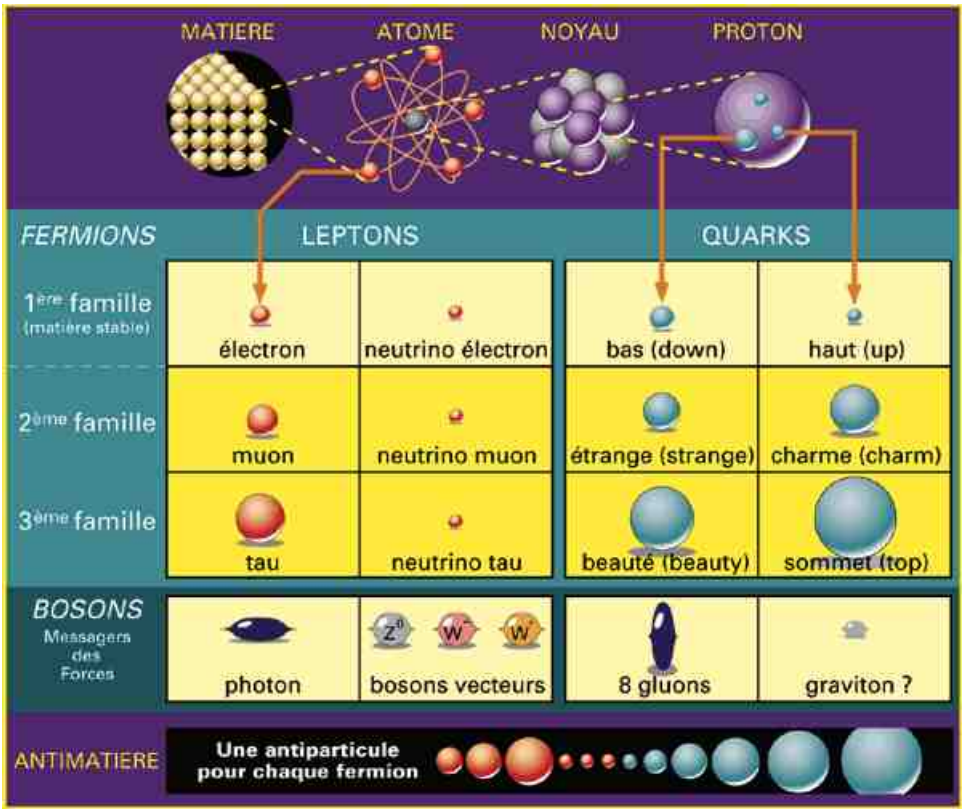

FIGURE 2.1 – Crédit : Laboratoire de Physique Corpusculaire de Clermont-Ferrand

2.2 La théorie de l'*Electrodynamique Quantique* (QED)

Cette théorie est bâtie à partir de la théorie de l'électromagnétisme à laquelle on rajoute la mécanique quantique à l'aide d'un Lagrangien relativiste [1]. Elle stipule que les fermions munis d'une charge [2] utilisent les photons comme vecteurs d'interaction de la force électromagnétique.
De plus, comme nous le savons, les photons n'ont pas de charge électrique ce qui empêche des

1. Ces prédictions se sont toutes avérées particulièrement précises au point que Richard Feynman, lui-même, la surnomma *"bijou de la physique"*.

2. Pour écarter le neutrino qui est une particule sans charge et donc non soumis à l'interaction électromagnétique.

perturbations électriques avec les fermions.Il faut également noter que la masse du photon étant nulle, sa portée devient donc mathématiquement infinie.

Enfin, d'un point de vue mathématique, cette théorie utilise le groupe de jauge [3] suivant :

$$SU(2)_L \otimes U(1)_Y \tag{2.1}$$

avec $SU(2)_L$ pour l'interaction d'isospin faible et $U(1)_Y$ pour l'interaction d'hypercharge faible.

2.3 La théorie de la *Chromodynamique Quantique* (QCD)

La QCD a la vertu d'expliquer l'interaction forte, toujours par l'échange de particules vecteurs d'interactions, les gluons. Elle s'exerce toujours entre les fermions, cette fois-ci munis d'une *couleur* : bleue, verte ou rouge. Les gluons étant également munis d'une couleur, ils échangent leur couleur avec celle des autres particules lors de l'interaction.

Seuls les quarks sont sensibles à cette interaction forte. En effet, les leptons ne sont pas munis de couleur et sont donc considérés comme neutres de ce point de vue.

D'autre part, il est important de noter que l'interaction forte croît avec la distance. C'est ce que l'on appelle la *liberté asymptotique*. C'est la raison pour laquelle nous n'observons jamais de quarks isolés mais bien des hadrons lors de la production de jets de collisions.

Enfin, d'un point de vue mathématique, la chromodynamique quantique modélise l'interaction forte à l'aide du groupe de jauge :

$$SU(3)_C \tag{2.2}$$

2.4 Les lacunes du Modèle Standard

D'après la théorie, toutes les particules mises en jeu (matière et champs) doivent systématiquement avoir une masse nulle. Ce n'est actuellement pas le cas d'après les mesures. Le modèle a donc prévu l'existence d'une nouvelle particule de spin nul : le *boson de Higgs*. Dès lors, c'est l'interaction entre les fermions et le boson de Higgs qui devrait donner la masse que l'on connaît à ces fermions. On postule donc également l'existence d'une cinquième interaction.

Nous l'avons vu, cette théorie permet d'expliquer les interactions électromagnétique, faible et forte mais pas l'interaction gravitationnelle ! Avec les nouveaux accélérateurs, et notamment le LHC, les théoriciens espèrent découvrir la particule vecteur associée à cette force : le *graviton*. Cependant, à l'heure actuelle, elle n'a jamais encore été observée.

La recherche actuelle en physique des particules, qu'elle soit orientée théorique ou expérimentale, cherche à combler ces lacunes afin de faire du Modèle Standard, une théorie encore plus conforme aux observations.

3. Avec $S(U)$ pour *Special Unitary group*

2.5 Les alternatives au Modèle Standard

Plusieurs autres théories sont actuellement développées :

- *La Supersymétrie* (*SuSy*) : il s'agit d'une théorie basée sur une symétrie qui relie les fermions aux bosons par l'intermédiaire d'un opérateur Q. Chaque fermion serait associé à son *superpartenaire* de spin entier et chaque boson à son superpartenaire à spin demi-entier.

- *La Technicouleur* : basé sur la condensation de deux champs fermioniques en un seul et unique champ. Elle suppose également l'existence d'une nouvelle famille : les *technifermions*.

- *Les dimensions supplémentaires* : d'après les modèles de Kaluza-Klein, elle est basée sur une manière élégante de lier l'interaction électrofaible et la gravitation. Ainsi, la gravitation qui se propagerait dans $4+n$ dimensions s'en trouverait tout naturellement diluée dans notre espace-temps de sorte que nous ne pourrions pas l'apprécier.

- *La Théorie des Cordes* : c'est une théorie très en vogue actuellement pour décrire de manière quantique, la gravitation [4]. Les particules sont considérées comme des objets de dimension 1 apparentés à des cordes. Un des problèmes sous-jacent est l'apparition d'une particule qui n'a pas de signification physique : le *tachyon* [5]. En revanche, certains problèmes ont déjà été résolus de façon théorique en la combinant avec la supersymétrie pour former la *Théorie des Supercordes* ou, avec des dimensions infinies, la *Théorie des Branes*.

- *La Théorie de la Grande Unification* (GUT) : qui permettrait de résoudre les problèmes liés à la brisure de symétrie en cherchant le bon groupe de jauge [6]. Toutes les interactions seraient décrites à l'aide de la même constante de couplage.

- *La Composivité* : basée quant à elle, sur l'excitation des particules fondamentales.

4. Sa concurrente directe est la *Théorie Quantique de la Gravitation à Boucle*.
5. Avec $m^2 < 0$ ce qui pose évidemment un problème.
6. Le groupe $SU(5)$.

Chapitre 3

Le Large Hadron Collider (LHC)

Le "*Grand Collisionneur de Hadrons*" est un accélérateur de particules d'une circonférence de 27 kilomètres situé à 100 mètres sous terre sous la frontière franco-suisse, près de Genève. C'est actuellement le plus puissant accélérateur au monde qui permet des collisions proton-proton d'une énergie attendue à terme de l'ordre de 14 TeV, dans le référentiel du centre de masse.

Les protons peuvent voyager dans le LHC grâce à l'intervention de 1296 aimants dipolaires supraconducteurs (refroidis à l'hélium liquide) générant un champ magnétique de 8,6 Teslas. La fréquence de croisement des faisceaux protoniques est de 40 MHz soit un croisement de faisceau toutes les 25 nanosecondes.

Enfin, notons que 4 instruments sont déployés sur le LHC pour observer les collisions entre particules :

- ALICE (A Large Ion Collider Experiment) : utilisé pour observer les collisions d'ions lourds.
- ATLAS (A Toroidal LHC ApparatuS) : concentré sur les recherches actuelles du Modèle Standard comme la recherche du Higgs et l'étude de nouvelles théories physiques précédemment évoquées.
- CMS (Compact Muon Solenoid) : dont les recherches sont semblables à celles d'ATLAS mais avec une conception technique du détecteur assez différente.
- LHCb (Large Hadron Collider beauty experiment) : dédiée à l'étude de l'asymétrie matière-antimatière et la violation de la symmétrie Charge-Parité (CP).

3.1 Les étapes de l'accélération

La figure 3.1 montre le complexe d'accélérateurs du CERN. Chaque accélérateur permet d'augmenter la vitesse du paquet de protons que l'on cherche à collisionner ensuite.

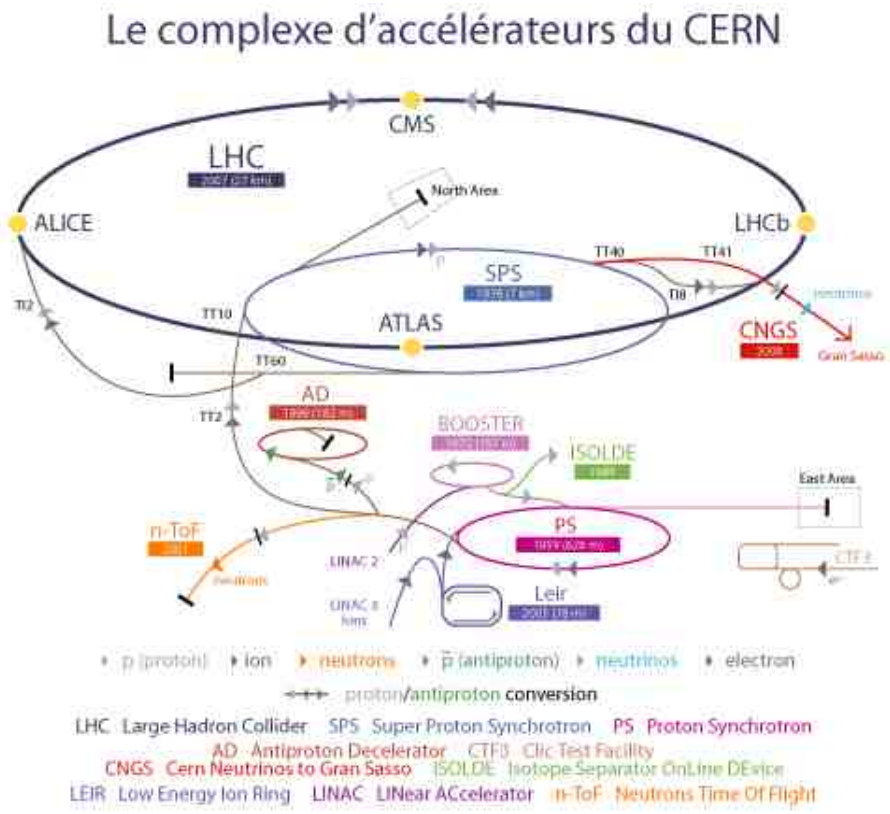

FIGURE 3.1 – Schéma du complexe d'accélérateurs du CERN (Crédit : CERN)

Dans un premier temps, les protons sont produits par ionisation d'hydrogène gazeux. Ils sont ensuite injectés dans une cavité radiofréquence où ils atteindront une énergie de l'ordre de 750 keV.

Puis, ils sont injectés dans le *LINAC* où ils atteindront 50 MeV. Les protons passent ensuite dans le *PS Booster* qui va définir la section du faisceau. Ensuite, ils passent dans le *PS* ou Synchroton à Proton qui permet de définir la taille et l'espacement des paquets de protons.

Dans le *SPS*, on configure le remplissage des paquets de protons qui atteignent une énergie de 450 GeV. Il s'agit du second plus grand accélérateur avant le LHC.

L'ultime accélération est réalisée dans le *LHC* où chacun des paquets de protons va pouvoir atteindre une énergie de l'ordre de 7 TeV. Ainsi, lorsque les deux paquets entrent en collision à de telles énergies, on obtient au final une énergie de collision dans le centre de masse de l'ordre de 14 TeV.

3.2 Le détecteur ATLAS

Dans ce rapport, nous utiliserons les données en provenance du détecteur ATLAS. Il s'agit du plus grand détecteur installé au LHC. Ses dimensions sont en effet importantes : 44 mètres de long, 25 mètres de diamètre et une masse de 7000 Tonnes. Il couvre un angle solide de 4π

stéradians.

ATLAS doit mesurer la trajectoire, la charge et l'impulsion des particules qui traversent ses détecteurs. Comme nous le verrons, plusieurs sous-détecteurs composent ATLAS comme le montre la figure 3.2.

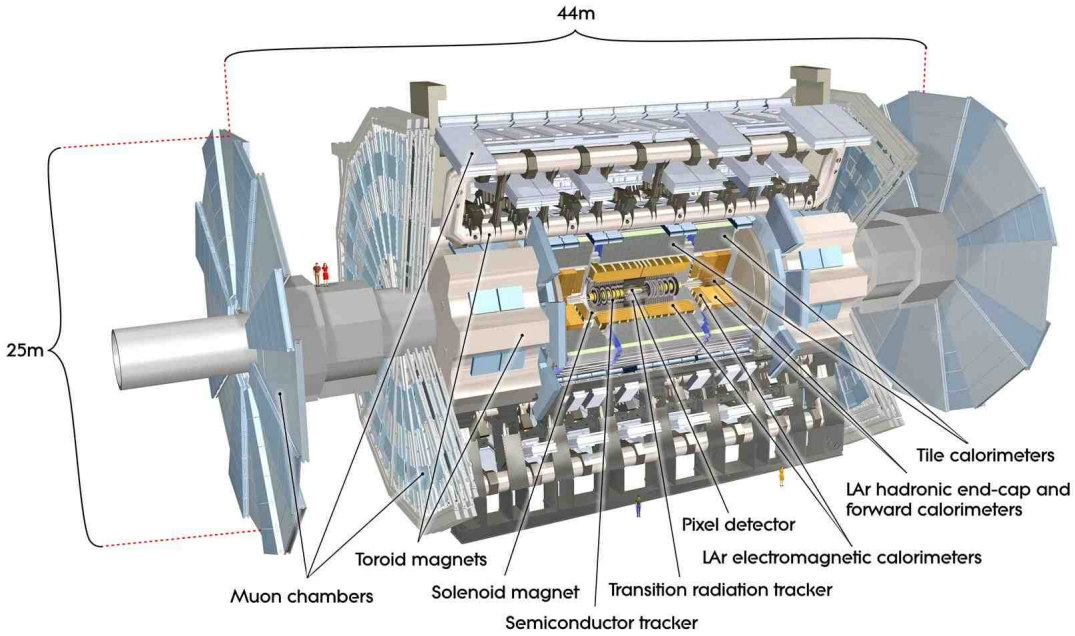

FIGURE 3.2 – Architecture du détecteur ATLAS (Crédit : CERN)

Des trajectographes permettent de déterminer les vertex primaires et secondaires. Autour de ceux-ci, nous trouvons un solénoïde supraconducteur afin de plonger la partie centrale dans un champ magnétique. Puis un calorimètre permet de mesurer l'énergie des particules qui traversent le détecteur. Enfin, 8 aimants toroïdaux surplombent l'ensemble de ces structures pour courber la trajectoire des muons lorsqu'ils sortent du calorimètre : on utilise alors le *spectromètre à muons*.

3.2.1 Les trajectographes

Cylindriques, ils ont une longueur de 690 cm et un diamètre de 115 cm. Le champ magnétique selon l'axe Oz est de 2 Teslas. Plusieurs sous-trajectographes sont présents.

Le détecteur a pixel Ses faibles dimensions (1,4 m de long et 50 cm de diamètre) en font le détecteur le plus proche du faisceau et de la collision. Ce détecteur est composé de 80 millions de pixels d'une dimension de 50 μm par 400 μm. Il est chargé d'assurer la reconstruction des vertex et des mesures des paramètres d'impacts. Le processus physique de détection est basé sur des jonctions P-N polarisées. C'est donc bien évidemment le trajectographe le plus fiable.

Le détecteur à micropistes de silicium ou SCT Ce détecteur est également à base de semi-conducteurs et composé, comme son nom l'indique, de micropistes de silicium de 80 μm de large pour 126 mm de long. Il sera utile pour la détection de vertex multiples et la mesure des leptons isolés de plus de 5 GeV/c.

Le trajectographe à rayonnement de position ou TRT Le TRT va jusqu'à un diamètre interne de 160 cm et permet, combiné au calorimètre électromagnétique, d'obtenir une bonne efficacité d'identification des électrons et un bon rejet des jets produits lors des collisions, qui ne sont pas intéressants.

3.2.2 Les calorimètres

Le but de la calorimétrie est de mesurer l'énergie et la position des particules qui le traversent. On distinguera trois calorimètres dans ATLAS.

Le calorimètre électromagnétique Le calorimètre électromagnétique est chargé de mesurer l'énergie des électrons et des photons. De plus, il mesure également la partie électromagnétique des jets, reconstruit la direction prise par les particules et identifie certaines particules.

Le calorimètre est maintenu à une température de 97 K et il s'agit d'un calorimètre constitué de deux milieux : l'argon dans lequel les signaux d'ionisation sont collectés et le plomb qui sert d'absorbeur. On dit qu'il est à échantillonnage *plomb/argon*.

Le calorimètre hadronique Le calorimètre hadronique permet, quant à lui, de mesurer l'énergie des hadrons et de contenir les gerbes hadroniques. Il est à échantillonnage *fer/scintillateur*. En effet, lorsqu'un hadron va traverser le milieu fer, il va déclencher une gerbe hadronique qui excite les molécules des tuiles scintillantes. La lumière recueillie grâce aux fibres optiques est ensuite photomultipliée pour convertir l'énergie des photons en signaux électriques.

Le calorimètre à l'avant Celui-ci sera utile pour l'étude des canaux où l'on s'attend à avoir une production de bosons de Higgs par fusion de bosons vecteurs. En effet, deux jets devraient être émis vers l'avant ou l'arrière du détecteur.

3.2.3 Le spectromètre à muon

Cet appareil sert à identifier les muons, mesurer précisément leur impulsion et leur trajectoire. L'efficacité d'identification est assez fiable pour des muons d'impulsion transverse d'énergie suffisante. Trois composantes sont importantes dans ce détecteur : l'aimant toroïdal qui courbe la trajectoire pour les mesures, la chambre de reconstruction de trace et la chambre de déclenchement.

3.3 L'informatique

La partie informatique est une composante essentielle en physique des particules. On verra qu'elle occupe la majeure partie du présent rapport.

En ce qui concerne les données recueillies au LHC, il faut savoir que l'espace de stockage doit être de 15 Pétaoctets (15000 To) par an ce qui nécessite l'utilisation d'autres machines réparties dans le monde pour le traitement des données. L'organisation mondiale de la répartition des machines est appelée la *grille de calcul*. Le LPC fait parti de cette grille de calcul en allouant un certain espace disque pour l'analyse et le calcul des données. La vitesse de transmission des données doit être aussi très importante de l'ordre de 1 To par seconde.

3.3.1 Les outils pour l'analyse

Parmi les outils utilisés pour l'analyse des données, nous citerons le logiciel ATHENA et le logiciel ROOT, tous deux codés en langage C++ par le CERN. En fonction du type de données brutes que l'on a, nous sommes amenés à choisir entre l'utilisation de ces deux logiciels.

Dans ce rapport, nous utiliserons l'outil d'analyse ROOT qui va nous permettre de programmer et d'effectuer des analyses de collisions.

Chapitre 4

Les processus physiques

Nous étudierons la désintégration d'une certaine catégorie de mésons appelés les mésons $D^{*\pm}$, composés d'un quark charm et d'un quark up ou down [1].

Les mésons D ont été découverts en 1976 avec les expériences *Mark 1* au Centre de l'Accélérateur Linéaire (SLAC) de Stanford. Plus particulièrement, les mésons $D^{*\pm}$ ont été découverts cette année, en 2010, grâce aux dernières données que nous étudierons.

4.1 Désintégration des mésons $D^{*\pm}$

Nous utiliserons les diagrammes de Feynman pour schématiser les désintégrations des mésons. Les deux cas du méson D^* sont représentés sur les figures 4.1 et 4.2 .

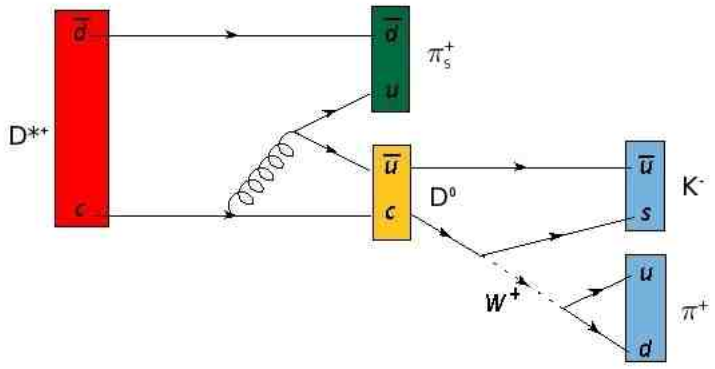

FIGURE 4.1 – Schéma de désintégration du méson D^{*+}.

Dans les deux cas, nous avons le méson $D^{*\pm}$ qui va se décomposer en méson D^0 et en pion mou (ou *soft* car son impulsion est faible étant donné la faible différence de masse entre $D^{*\pm}$ et

1. On symbolisera le quark et l'antiquark par la première lettre de leur saveur surmonté, ou non, d'une barre pour l'anti-particule.

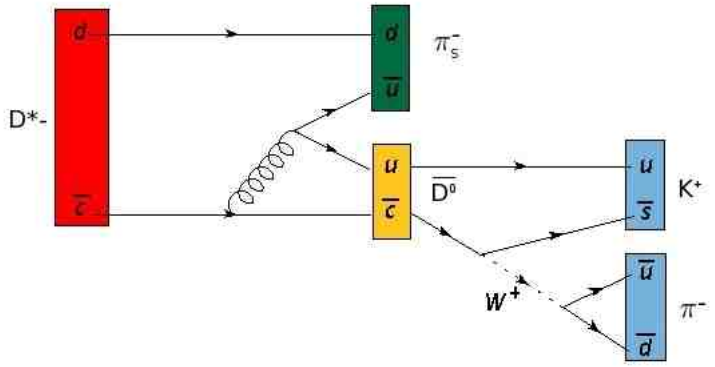

FIGURE 4.2 – Schéma de désintégration du méson D^{*-}.

D^0). Cette désintégration représente 67,7% de toutes les désintégrations connues. L'équation de désintégration s'écrit :

$$D^{*\pm} \rightarrow D^0 + \pi_S^{\pm} \tag{4.1}$$

Puis le méson D^0 va donner [2] à son tour deux autres particules, un kaon $K^{\pm}$et un pion $\pi^{\pm}$ selon l'équation suivante :

$$D^0 \rightarrow K^{\pm} + \pi^{\pm} \tag{4.2}$$

Nous noterons également lors de la désintégration du D^0, on obtient un changement de saveur du quark charm, transformé en anti-quark strange par le biais d'un boson $W^{\pm}$.

4.2 Reconstruction de la masse invariante

4.2.1 Etablissement du référentiel d'étude

Avant d'étudier la physique qui intervient dans le traitement des données, attardons-nous sur le référentiel utilisé pour l'étude des trajectoires de nos particules. Nous utiliserons le repère de la figure 4.3.

Par convention, l'axe z désignera l'axe du faisceau des paquets. Alors les axes y et z sont définis tels que le trièdre (Oxyz) soit direct.

Notons également que l'angle $\varphi \in [0 : 2\pi]$ et permet de déterminer la trajectoire à l'avant ou à l'arrière du faisceau du paquet ainsi que l'impulsion $\vec{p}$ tandis que l'angle $\theta \in [0 : \pi]$ compris dans le *plan transverse* contenant l'*impulsion transverse* $\vec{p}_T$, nous permettra d'extraire des données cruciales pour notre étude.

2. La durée de vie du méson D^0 est $(410,1 \pm 1,5) \times 10^{-15}$s, soit suffisante pour la détection dans ATLAS.

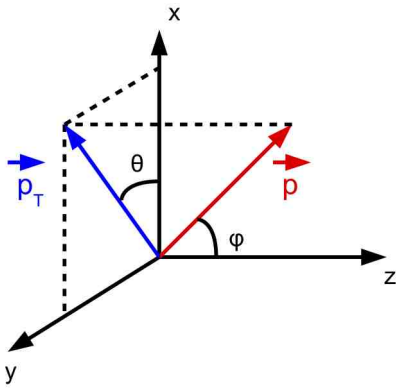

FIGURE 4.3 – Système de cordonnées du référentiel d'étude pour les collisions dans le détecteur ATLAS.

4.2.2 Les paramètres d'étude

Vecteur d'énergie impulsion Le vecteur d'énergie impulsion s'écrit par définition :

$$\vec{p} = \begin{pmatrix} \frac{E}{c} \\ p_x \\ p_y \\ p_z \end{pmatrix} \tag{4.3}$$

avec c la vitesse de la lumière dans le vide.

Dès lors la composante transverse de l'impulsion dans le plan (Oxy)s'écrit :

$$p_T = \sqrt{p_x^2 + p_y^2} \tag{4.4}$$

avec les composantes de l'impulsion de la particule suivant les axes (Ox) et (Oy).

La rapidité La rapidité se définit par la relation suivante :

$$y = \frac{1}{2} ln \left(\frac{E + p_z}{E - p_z} \right) \tag{4.5}$$

avec E, l'énergie de la particule et p_z la composante de l'impulsion de la particule suivant l'axe du faisceau Oz.

La pseudo-rapidité La pseudo-rapidité se définit par l'expression suivante :

$$\eta = -\ln \tan \left(\frac{\theta}{2} \right) \tag{4.6}$$

L'énergie transverse de la particule L'énergie transverse de la particule s'écrit par définition à l'aide de la pseudo-rapidité η et l'angle θ :

$$E_t = \frac{E}{\cosh \eta} \tag{4.7}$$

4.2.3 Détermination de la masse du méson $D^{*\pm}$

Equations fondamentales en formalisme relativiste Nous utiliserons les notions de relativité restreinte ainsi que les acquis concernant le quadrivecteur d'énergie-impulsion que nous avons définis plus haut. En effet, nous savons que la norme du quadrivecteur d'énergie-impulsion est invariante par changement de référentiel, c'est à dire, en vertu de l'équation d'Einstein :

$$\frac{E^2}{c^2} - p^2 = (mc)^2 \tag{4.8}$$

avec la norme du vecteur d'énergie-impulsion au carré :

$$\|\vec{p}\|^2 = p_x^2 + p_y^2 + p_z^2 \tag{4.9}$$

De plus, il est important de noter que le quadrivecteur se conserve pour un système isolé. C'est à dire que lors d'une désintégration d'une particule mère X en particules filles a et b, nous pouvons écrire la relation suivante :

$$\vec{p_X} = \vec{p_a} + \vec{p_b} \tag{4.10}$$

ainsi que la conservation en énergie :

$$E_X = E_a + E_b \tag{4.11}$$

Calcul de la masse de la particule mère Nous disposons alors de l'équation fondamentale suivante pour chaque particule i :

$$E_i^2 = m_i^2 c^4 + p_i^2 c^2 \tag{4.12}$$

Puis, avec les équations de conservations 4.10 et 4.11, nous remontons trivialement à la masse de la particule mère recherchée :

$$m_X^2 = \frac{1}{c^4}\left[E_X^2 - p_X^2 c^2\right] \tag{4.13}$$

On aboutit donc à la relation portant sur la masse de la particule mère X avec l'équation 4.14.

$$m_X = \sqrt{\frac{(E_a + E_b)^2}{c^4} - \frac{(\vec{p_a} + \vec{p_b})^2}{c^2}} \tag{4.14}$$

avec m_X en kg, E en J soit $kg.m^2.s^{-2}$ et p en $kg.m.s^{-1}$ et c en $m.s^{-1}$.

Application au cas du méson $D^{*\pm}$ Dans notre cas d'étude, nous travaillerons sur les hypothèses de masse du méson D^0 donc la masse sera calculée à partir des candidats pion π et kaon K avec l'équation 4.15

$$m_{D^0} = \sqrt{\frac{(E_\pi + E_K)^2}{c^4} - \frac{(\vec{p_\pi} + \vec{p_K})^2}{c^2}} \tag{4.15}$$

Ainsi que sur la masse du $D^{*\pm}$ calculé à l'aide des précédents candidats D^0 et pions mous $\pi_S^{\pm}$ que l'on écrira de façon semblable avec la relation 4.16.

$$m_{D^{*\pm}} = \sqrt{\frac{(E_{\pi_S^{\pm}} + E_{D^0})^2}{c^4} - \frac{(\vec{p_{\pi_S^{\pm}}} + \vec{p_{D^0}})^2}{c^2}} \tag{4.16}$$

Deuxième partie

Analyse des candidats $D^{*\pm}$ à 7 TeV

Chapitre 5

Etude des candidats $D^{*\pm}$

Nous passons directement à l'analyse des données et le détail des opérations effectuées pour arriver aux résultats présentés. Les lecteurs désirant obtenir plus d'informations sur la syntaxe de programmation et les caractéristiques du logiciel ROOT, peuvent se rendre sur le site internet dédié [1].

De plus, les programmes utilisés pour les analyses [2] seront fournis en annexes du présent rapport.

5.1 Les données à disposition

Les données que nous utilisons proviennent des collisions p-p effectuées au CERN du mois de mars à mai 2010. Nous utiliserons les runs suivants issus des données *D3PDMAKER* (419 GB) :

Période A

```
FiltSkim.MinBiasTree.data10_7TeV.00152166.f239_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152777.f243_p127/
FiltSkim.MinBiasTree.data10_7TeV.00153200.f249_p127/
FiltSkim.MinBiasTree.data10_7TeV.00153159.f249_p127/
FiltSkim.MinBiasTree.data10_7TeV.00153136.f249_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152214.f239_p127/
FiltSkim.MinBiasTree.data10_7TeV.00153134.f249_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152844.f243_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152221.f239_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152508.f241_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152845.f243_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152345.f239_p127/
```

1. Adresse URL : `http://root.cern.ch`
2. Nous partons des travaux initiaux de Messieurs L. Valéry et J. Kerneis pour les différentes phases de l'étude.

```
FiltSkim.MinBiasTree.data10_7TeV.00153030.f247_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152441.f239_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152409.f239_p127/
FiltSkim.MinBiasTree.data10_7TeV.00152878.f243_p127/
```

Période B :

```
FiltSkim.MinBiasTree.data10_7TeV.00155112.f255_p176/
FiltSkim.MinBiasTree.data10_7TeV.00155116.f255_p176/
FiltSkim.MinBiasTree.data10_7TeV.00155160.f255_p176/
```

Nous avons donc finalement des données sur environ [3] 28 millions d'évènements.

5.2 Précédents résultats

Dans la première partie, nous tenons à rappeler les précédents résultats obtenus [4] par M. Loïc Valéry sur lesquels nous nous basons pour notre étude [5].

Les données utilisées sont donc les mêmes que celles utilisées dans le présent rapport au nombre d'évènements près. Le programme commence par balayer donc toutes les traces avec la variable `trk_n` puis sélectionne les traces dont le nombre de coups dans le détecteur à pixel est de 1 avec `pixmin=1` et des conditions géométriques sur le $d0$ et le $z0sin$:

```
for(Int_t i=0;i<trk_n;i++)
  {
    if((*trk_nPixHits)[i]>=pixmin && fabs((*trk_d0_wrtPV)[i])<d0_max &&
  fabs(z0sin)<z0sin_max)
      {
```

Puis le programme analyse ensuite les traces dont le rapport $\frac{q}{p}$ est soit positif ou négatif, permettant ainsi de déterminer la charge de la particule. Pour la partie négative, le programme s'écrit :

```
    if((*trk_qoverp_wrtPV)[i]>0)
      {
        pionp.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],pionm);
        kaonp.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],kaonm);
        kplus.push_back(kaonp);
        pplus.push_back(pionp);
      }
```

Puis viens ensuite la sélection des candidats pion avec une coupure sur l'impulsion transverse des candidats sélectionnés jusque-là avec `ptmind0=500` en MeV/c.

3. 27 786 384 évènements exactement.
4. Voir le mémoire de stage de recherche *Reconstruction des mésons $D^{*\pm}$ avec le détecteur ATLAS - Juin 2010.*
5. Ces résultats ont été obtenus avec les données de la période A uniquement.

```
if(pplus[j].Pt()>ptmind0 && pmoins[k].Pt()>ptmind0)
```

Puis l'on associe les candidats pions sélectionnés avec les candidats kaon pour ensuite prendre ces paires de particules qui doivent satisfaire la condition suivante :

```
if(fabs(vkp.M()-d0m)<deltam)
```

Les paires de particules dont la différence de masse avec la valeur de D^0 attendue théoriquement, `d0m=1864.84` MeV, est inférieure à `deltam=200` MeV sont alors considérées comme des candidates de mésons D^0.

Puis en sélectionnant un candidat D^0 avec un autre [6] candidat pion, on forme un potentiel candidat $D^{*\pm}$ pris en compte, seulement s'il répond à la coupure sur son impulsion transverse qui doit être supérieure à `ptminds=2500` MeV/c :

```
if(vDSplus.Pt()>ptminds)
```

Enfin, les vrais candidats $D^{*\pm}$ sont finalement sélectionnés à l'aide de la condition sur la différence de masse :

```
if((vDSplus.M()-vkp.M())/1000<2 && (vDSplus.M()-vkp.M())/1000>0.05)
```

C'est à dire que les vrais candidats $D^{*\pm}$ sont ceux dont la différence de masse entre le méson $D^{*\pm}$ et le méson D^0 est supérieure à 50 MeV et inférieure à 200 MeV.

Le résultat obtenu d'après ce programme est exposé sur la figure 5.1.

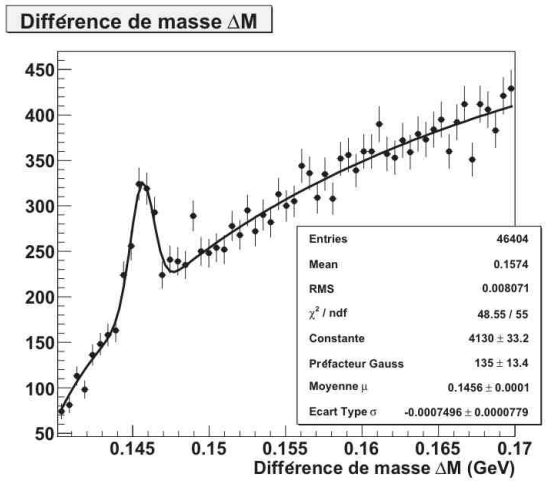

FIGURE 5.1 – Différence de masse entre le méson $D^{*\pm}$ et le méson D^0 et superposition du fit (Crédit : Loïc Valéry)

6. Différents des autres candidats pion. Attention à bien sélectionner la particule avec un indice différent de celui du premier pion.

5.3 Travail à effectuer

Dans un premier temps, le travail consiste à étudier la fenêtre de masse du méson $D^{*\pm}$. Nous sélectionnons ainsi des composantes du signal mais aussi de bruit de fond indésirable autour de la région située approximativement entre 140 MeV et 150 MeV comme le montre la figure 5.2

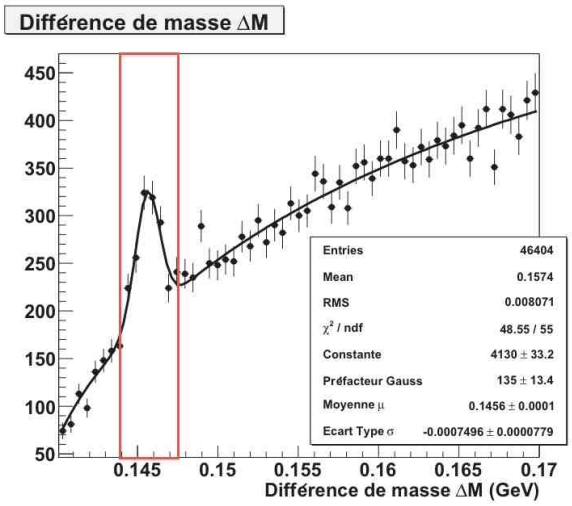

FIGURE 5.2 – Sélection de la fenêtre de masse de la différence de masse entre le méson $D^{*\pm}$ et le méson D^0 et superposition du fit (Crédit : Loïc Valéry)

Nous choisissons également de travailler également sur la composante bruit de fond seule, toujours cadrée sur la même fenêtre de masse, d'après la figure 5.3.

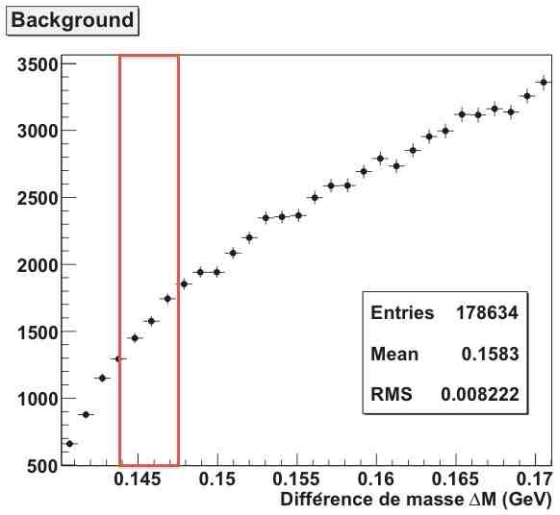

FIGURE 5.3 – Sélection de la fenêtre de masse du bruit de fond de la différence de masse entre le méson $D^{*\pm}$ et le méson D^0 et superposition du fit (Crédit : Loïc Valéry)

Cette différenciation entre la composante signal+bruit de fond (que nous appellerons SB par commodité) et bruit de fond seul (nommée B) s'effectue dans le programme `DStarRecTree.C`[7] avec les modifications que l'on détaillera dans la partie suivante.

Nous étudierons également de nouvelles variables pour l'étude du méson $D^{*\pm}$. Les comparaisons entre les histogrammes obtenus pour le SB, le B et les simulations numériques Monte-Carlo, nous permettront d'extraire les coupures intéressantes à effectuer sur le signal.

5.4 Nouvelles coupures et sélections SB et B

Nous avons modifié le programme `DStarRecTree.C` de la façon suivante pour différencier le SB du B.

Dans un premier temps, nous avons décidé de sélectionner uniquement les traces de particules issues du vertex primaire (le vertex primaire est caractérisé par la valeur 1 du `vx_type`) du détecteur ATLAS :

```
for(int v=0 ; v<vx_n ; ++v) {
      if ((*vx_type)[v]==1) {
```

Après les coupures[8] sur l'impulsion transverse des candidats pion et méson $D^{*\pm}$, nous sélectionnons le SB à l'aide de la variable `type_signal` qui sera notée comme égale à 1 pour SB et à 0 pour B.

Ainsi, nous programmons le remplissage de l'arbre `tree` dans la boucle contenant les coupures sur les masses du D^0 et du $D^{*\pm}$ avec des conditions beaucoup plus sévères :

```
if (fabs(vkp.M()-d0m)<deltam && diffds_d0>0.135 && diffds_d0<0.170) {
      h12->Fill((vDSplus.M()-vkp.M())/1000);
      type_signal=1;
      tree->Fill();
```

La condition sur la différence de masse entre le candidat D^0 et la masse théorique $d0\_m$ est beaucoup plus restrictive cette fois-ci avec :

```
  float_t deltam=20;
```

Dès lors, nous considérons la sélection SB avec une coupure sur la différence de masse du D^0 de 20 MeV et une coupure sur la différence de masse entre le méson $D^{*\pm}$ et du méson D^0 supérieure à 135 MeV et supérieure à 170 MeV.

En ce qui concerne la sélection du B, elle s'effectue de la même façon à l'aide de coupures similaires :

```
        } else if (fabs(vkp.M()-d0m)>50 && diffds_d0<0.170) {
      h13->Fill((vDSplus.M()-vkp.M())/1000);
      type_signal=0;
      tree->Fill();
```

7. Le programme est fourni en annexes du rapport.
8. Voire la partie des précédents résultats.

Nous considérons donc comme B, les particules dont la différence de masse entre le candidat D^0 et sa masse théorique est supérieure à 50 MeV. En revanche, si la différence de masse entre $D^{*\pm}$ et D^0 est inférieure [9] à 170 MeV.

5.5 Nouvelles variables d'étude

Afin d'opter pour les coupures les plus efficaces, nous avons choisi de travailler sur de nouvelles variables potentiellement intéressantes.

Somme des impulsions transverse des particules du vertex primaire $\sum P_T$

Nous insérons ce code immédiatement après la demande de sélection des traces correspondantes aux particules issues du vertex primaire.

```
    for(int v=0 ; v<vx_n ; ++v) {
      if ((*vx_type)[v]==1) {

  for(int t=0 ; t<(*vx_trk_n)[v] ; ++t) {
    const int i=(*vx_trk_index)[v][t];
    float trk_Pt=(*trk_pt)[i];

    trk_sumPt=trk_sumPt+trk_Pt;
```

Somme des impulsions transverse au carré des particules du vertex primaire $\sum P_T^2$

La même variable que précédemment est calculée cette fois-ci en étant élevée au carré.

```
    trk_sumPt_2=trk_sumPt_2+trk_Pt*trk_Pt;
```

Différence de pseudo-rapidité entre les mésons $D^{*\pm}$ et D^0, $\Delta\eta$

Nous étudions dans ce cas la différence relative de vitesse des deux particules après collision dans le vertex primaire.

```
  detadsd0=vDSplus.Eta() - vkp.Eta();
```

Différence de distance radiale $r$entre les mésons $D^{*\pm}$ et D^0, Δr

Elle regroupe les contributions des deux variables précédentes, $\Delta\varphi$ et $\Delta\eta$ qui lui sont reliée par la formule suivante :

$$\Delta r = \sqrt{\Delta\varphi^2 + \Delta\eta^2} \tag{5.1}$$

que l'on modélisera donc tout simplement à l'aide des fonctions pré-inclues dans ROOT par :

```
  drdsd0=vkp.DeltaR(vDSplus);
```

9. Il n'y a pas besoin de contraindre la différence de masse à être supérieure à 135 MeV ici car la différence de masse ne sera jamais inférieure à celle du pion qui est justement de 140 MeV.

Cosinus de l'angle θ^* entre le méson D^0 et le kaon $K^{\pm}$, $\cos\theta^*_K$

L'angle calculé est celui entre le kaon K et le méson D^0 lors de la désintégration de ce dernier mais dans le référentiel du centre de masse.

```
costhetaK=cos(kp1.Angle(tvpk));
```

L'angle est évidemment calculé dans le référentiel du méson D^0 ce qui implique la programmation du calcul de la transformée de Lorentz inverse du vecteur d'énergie-impulsion du kaon :

```
        TVector3 tvpk=vpk.BoostVector();
        TLorentzVector kp1=kplus[j];
        kp1.Boost(-tvpk);
      costhetaK=cos(kp1.Angle(tvpk));
```

Comme le lecteur pourra le voir, nous utilisons des `TLorentzVector` pour effectuer la transformation du vecteur. La transformation inverse de Lorentz (ou *boost*) s'effectuera grâce à la présence du signe $-$ dans le code `Boost`. On obtient ensuite notre variable `costhetaK`.

Cosinus de l'angle θ^* entre le méson D^0 et le kaon $K^{\pm}$, $\cos\theta^*_{D^0}$

Le même type de calcul est réalisé pour la désintégration du méson $D^{*\pm}$ en se plaçant, bien évidemment, dans son référentiel.

```
costhetaD0=cos(d0p.Angle(tvDSplus));
```

Chapitre 6

Etude des données de simulation Monte-Carlo

Nous utilisons les données de simulations numériques provenant d'une méthode statistique mathématique Monte-Carlo. Les données issues de ces simulations simulent parfaitement des collisions à 7 TeV et prennent en compte absolument tous les paramètres nécessaires à l'étude. Le détecteur ATLAS y est lui-même simulé avec une très grande précision.

La première version du programme de traitement de ces données Monte-Carlo a été programmée par M. Jérémy Kerneis. Afin de l'adapter à notre problème, nous avons dû effectuer quelques modifications majeures dans le programme que nous détaillerons dans la première partie de ce chapitre.

6.1 Modification du code source original

6.1.1 Signal des "vraies" données RMC simulées Monte-Carlo

A l'instar du programme utilisé pour le traitement des données réelles, nous travaillerons uniquement avec des particules issues du vertex primaire :

```
for(int v=0 ; v<vx_n ; ++v) {
      if ((*vx_type)[v]==1) {
```

Puis l'on s'occupe ensuite de calculer les nouvelles variables $\sum P_T$ et $\sum P_T^2$ sur toutes les particules sélectionnées dans le vertex primaire grâce au code suivant :

```
for(int t=0 ; t<(*vx_trk_n)[v] ; ++t) {
    const int w=(*vx_trk_index)[v][t];
    const int w2=(*trk_mc_index)[w];

    if (w2>0 && w2<mc_n-1){
      float trk_Pt=(*mc_gen_pt)[w2];
      trk_sumPt=trk_sumPt+trk_Pt;
      trk_sumPt_2=trk_sumPt_2+trk_Pt*trk_Pt;
    }
  }
```

La principale difficulté réside dans le fait de balayer toutes les traces du vertex primaire avec `vx_trk_n` puis de sélectionner l'indice de ces traces, noté `w`. Cet indice sert ensuite à identifier l'indice de ces mêmes traces mais correspondant aux simulations Monte-Carlo avec l'indice `w2`.

Il ne reste plus qu'a balayer ces traces Monte-Carlo pour en sortir les impulsions transverses des particules à l'aide de la variable nommée `trk_Pt`.

Utilisation de la variable `mc_mother_pdg` Nous utiliserons également la variable `mc_mother_pdg` qui permet de savoir si la particule MC sélectionnée, a pour particule mère une particule que l'on recherche. Cette variable nous sera forte utile car elle permettra de sélectionner encore plus précisément les bonnes particules dont nous aurons besoin.

On demande dans un premier temps de balayer toutes les particules et de chercher celles dont la mère est un D^0 :

```
for (int i=0 ; i<mc_n ; i++) {
    if ( (*mc_mother_pdg)[i]==421 || (*mc_mother_pdg)[i]==-421);
      nbre_part++;
```

Le variable `nbre_part` permet de compter le nombre de particules issues du D^0. Elle nous permettra de ne sélectionner que les particules Kaon et Pion qui, à deux uniquement, forment le D^0 que nous attendons. En effet, rappelons que la désintégration qui nous intéresse ne représente qu'une portion très faible sur toutes les désintégrations (seulement 3,80 %). Il est alors capital de s'assurer de sélectionner les bons candidats.

Puis nous cherchons un Kaon et un Pion à l'aide de la variable `mc_pdg` :

```
      if ((*mc_pdg)[i]== 321 || (*mc_pdg)[i]== -321) {
        w1=i;
      } else if ((*mc_pdg)[i]== 211 || (*mc_pdg)[i]== -211) {
        w2=i;
      }
```

Nous identifions également ces particules lorsqu'elles sont sélectionnées à l'aide des variables `wi`, avec $i \in [1, 2, 3]$ pour respectivement, le Kaon, le Pion et le Pion soft, calculés d'une manière identique.

Une fois que nous sommes sûrs que le nombre de particules issues du D^0 est bien égal à deux et que les particules sont bien un Kaon et un Pion, alors nous pouvons créer notre candidat D^0 :

```
if (nbre_part==2 && w1>-1 && w2>-1){
    VecK.SetPtEtaPhiE( (*mc_gen_pt)[w1] ,
    (*mc_gen_eta)[w1] , (*mc_gen_phi)[w1], (*mc_gen_energy)[w1] );
    VecPION1.SetPtEtaPhiE( (*mc_gen_pt)[w2] ,
    (*mc_gen_eta)[w2] , (*mc_gen_phi)[w2], (*mc_gen_energy)[w2] );
    VecD0=VecK+VecPION1;
```

Il est important de noter également que nous n'avons aucun moyen de savoir si les deux particules Kaon et Pion proviennent effectivement du même méson D^0. Dès lors, nous avons encore quelques artefacts de mauvaises combinaisons de D^0. Nous pouvons régler ce problème par une coupure sur la masse du D^0 :

```
if (VecD0.M()>1855 && VecD0.M()<1875) {
```

Puis, si toutes ces conditions sont effectivement vérifiées, alors nous pouvons créer le méson $D^{*\pm}$ uniquement si nous avons un seul Pion qui provient d'une mère $D^{*\pm}$. Une restriction sur la masse de la particule est également réalisée pour la même raison que précédemment.

Jusqu'à présent, nous avons donc obtenu un signal correspondant aux candidats $D^{*\pm}$ issus des "vraies" données Monte-Carlo que nous noterons par le sigle *RMC* (pour "Real Monte-Carlo") pour plus de clarté. Ce signal sera identifié à l'aide de la condition sur notre variable `type_signal=1`.

6.1.2 Signal "reconstruit" PMC simulé à l'aide de Monte-Carlo

Cependant, nous cherchons désormais à isolé le signal "reconstruit" de nos données Monte-Carlo, c'est-à-dire le signal pour lequel nous avons bien trois particules (que nous nommeront *PMC* pour "Particle Monte Carlo"). Un Kaon et un Pion pour construire notre D^0 ainsi qu'un Pion Soft pour l'associer au D^0 et construire finalement notre méson $D^{*\pm}$.

Dès lors, nous identifieront ce signal à l'aide de la condition `type_signal=2`. Cependant, il est nécessaire avant toute chose de reconnaître nos trois particules :

```
for(int tr=0 ; tr<trk_n ; ++tr) {
      const int mci=(*trk_mc_index)[tr];

      if (mci==i) w1=tr;
      else if (mci==j) w2=tr;
      else if (mci==k) w3=tr;
```

Le balayage des traces est effectué. Cependant, nous ne connaissons les particules intéressantes que grâce aux variables `mc_`. Il est donc nécessaire d'obtenir l'indice des traces MC à l'aide du paramètre `trk_mc_index`. Ainsi, il suffit ensuite de comparer ces indices par rapport aux indices du même type `i` correspondant au Kaon, `j` pour le premier Pion et `k` pour le second Pion dit soft.

Si les conditions sont effectivement vérifiées, il ne reste plus qu'à extraire l'indice de trace correspondant `tr` et l'identifier à une série de variable `w`.

Nous sélectionnons ainsi nos trois particules pour le signal PMC. Nous devons ensuite effectuer un nouveau calcul de variables en changeant les formules de calculs des *TLorentzVector* ainsi qu'en appliquant une sélection des particules avec les bonnes conditions géométriques sur le vertex primaire et sur le détecteur à pixel :

```
    if (((*trk_z0_wrtPV)[ww1] * sin((*trk_theta_wrtPV)[ww1]))<z0sin_max
ww1>0 && (*trk_nPixHits)[ww1]>=pixmin && fabs((*trk_d0_wrtPV)[ww1])<d0_max
        ) {
```

Chapitre 7

Comparaison

7.1 Coupures intéressantes

Nous décidons d'étudier les nouvelles variables afin de savoir si certaines coupures semblent intéressantes lorsque l'on compare le SB et le B.

Légende des histogrammes Les données SB issues des données du LHC sur les 28 millions d'évènements sont symbolisées en rouge. La composante bruitée, B sera dessinnée en bleue. Ces deux codes couleurs constituent donc les informations issues des données que nous étudions.

En revanche, la couleur vert symbolise le signal Monte-Carlo correspondant à des *vraies* particules intervenant dans le processus de reconstruction du méson $D^{*\pm}$, que nous avions appelés RMC. Le signal Monte-Carlo correspondant au signal *reconstruit PMC*, est quant à lui dessiné en violet.

Considérations techniques Il est important de noter que les histogrammes qui suivent ont tous subis une normalisation pour pouvoir être comparés entre eux. Il s'étendent donc tous en ordonnées de 0 à 1 nous permettant d'effectuer des conclusions assez rapides.

De plus, les informations issues de RMC et PMC ont subis exactement les mêmes coupures que celles appliquées sur SB et B, afin de pouvoir les comparer.

Ainsi, sur RMC et PMC sont appliqués les conditions sur les impulsions du premier pion (PION1) qui intervient dans la reconstruction du D^0 avec $P_{TPION1} > 1500$ MeV/c et l'impulsion sur le méson $D^{*\pm}$ avec $P_T D^{*\pm} > 4000$ MeV/c.

Etude de $Cos\theta^*_K$

La figure 7.1 présente une répartition spatiale entre -1 et 1 avec une hausse de la concentration vers 0.

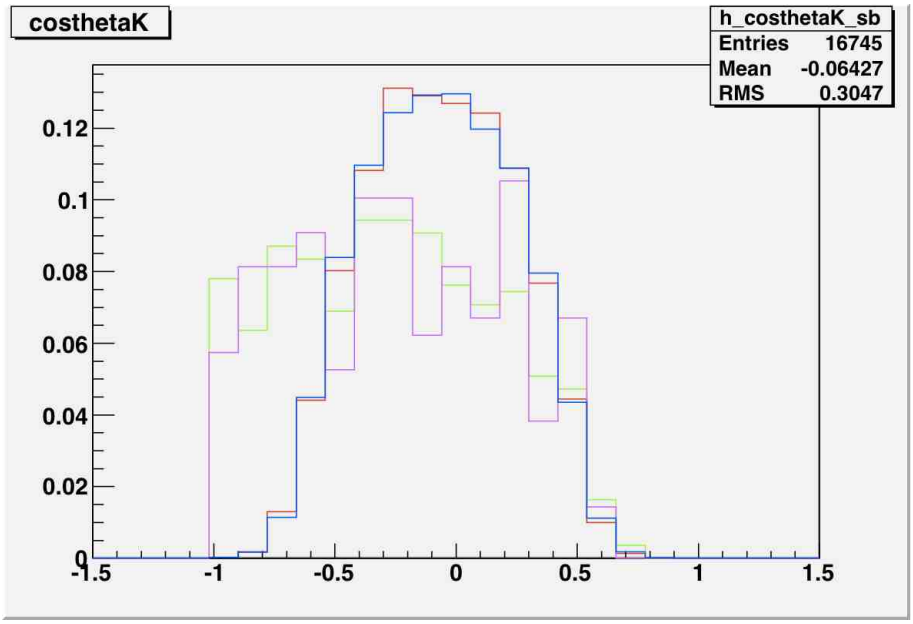

FIGURE 7.1 – Histogrammes pour différentes données de $Cos\theta^*_K$ pour le méson D^0.

La forme de *cloche* que nous observons sur le SB et B est dûe à la coupure sur l'impulsion du premier Pion (PION1) intervenant dans la reconstruction du méson D^0. Cependant, en appliquant cette coupure, on s'aperçoit bien que seule la partie droite de la figure (vers 1) suit effectivement cette forme de cloche après coupure, pour PMC. Mais nous n'avons pas réussi à obtenir la même forme que SB pour la partie gauche (vers -1).

En revanche, nous pouvons nous apercevoir que l'étendue du pic SB est un peu plus faible que celle du B. Nous pourrions définir le maximum d'entrées pour un cosinus compris entre $-0,4$ et $0,4$.

Etude de $Cos\theta^*_{D^0}$

Comme le montre la figure 7.2, SB et B suivent une progression linéaire continue mais qui s'étale de 3% à 6% environ.

Cette croissance linéaire devient en quelques sorte négligeable lorsque l'on fait la comparaison avec PMC qui, malgré quelques fluctuations statistiques, montre une valeur moyenne entre 4% et 5%.

Dans ces conditions, il devient difficile d'en conclure de l'efficacité de potentielles coupures. Cette variable n'est donc pas intéressante pour notre étude.

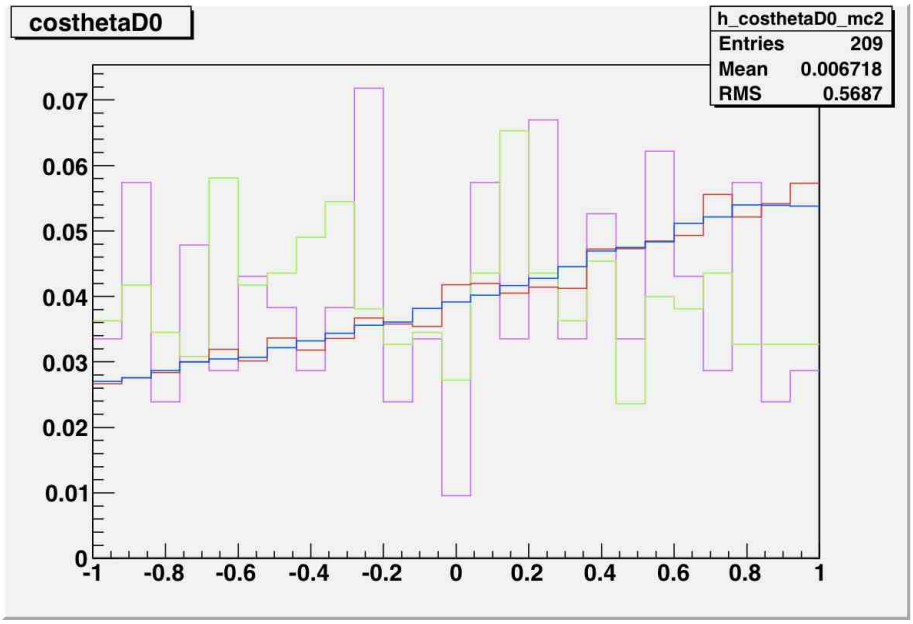

FIGURE 7.2 – Histogrammes pour différentes données de $Cos\theta^*_{D^0}$ pour le méson D^0.

Etude de $P_T D^0$

Ce que nous remarquons directement en observant la figure 7.3, c'est la différence du nombre d'entrées dans le pic commun à toutes les données.

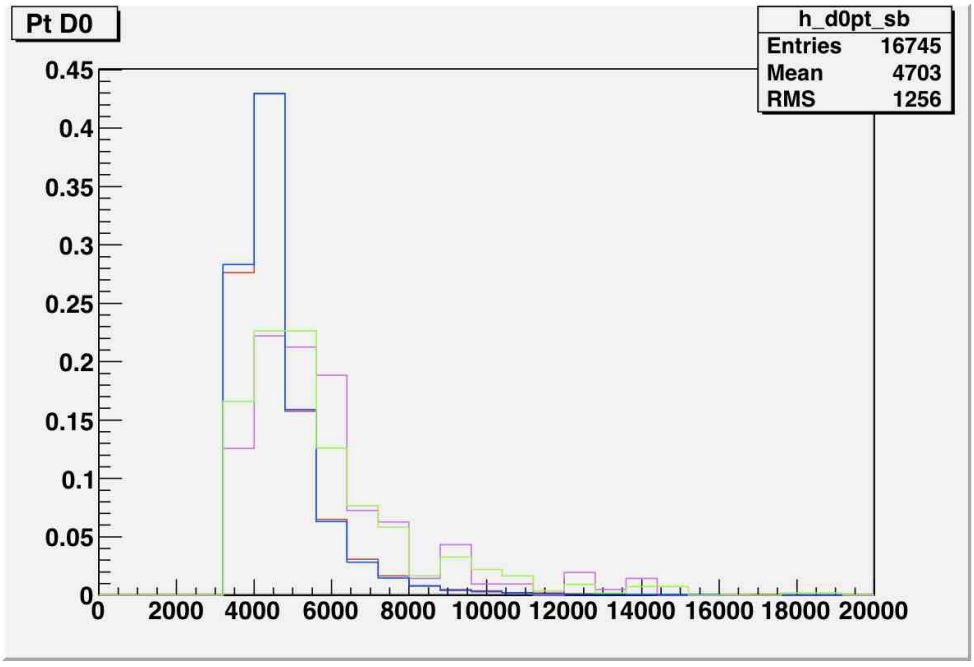

FIGURE 7.3 – Histogrammes pour différentes données de $P_T D^0$ pour le méson D^0.

A première vue, une coupure peut-être intéressante sur cette variable. En effet, le signal violet est décalé vers les impulsions supérieures à 4000 MeV/c jusqu'à 6000 MeV/c environ, contrairement au SB qui lui début environ vers 3500 MeV/c et s'arrête plus tôt, vers 5500 MeV/c.

Nous pouvons espérer donc une coupure intéressante vers 5000 MeV/c environ.

Etude de $P_T D^{*\pm}$

Encore une fois, nous observons une distribution semblable à celle de l'impulsion du D^0, pour l'impulsion $D^{*\pm}$ présent sur la figure 7.4.

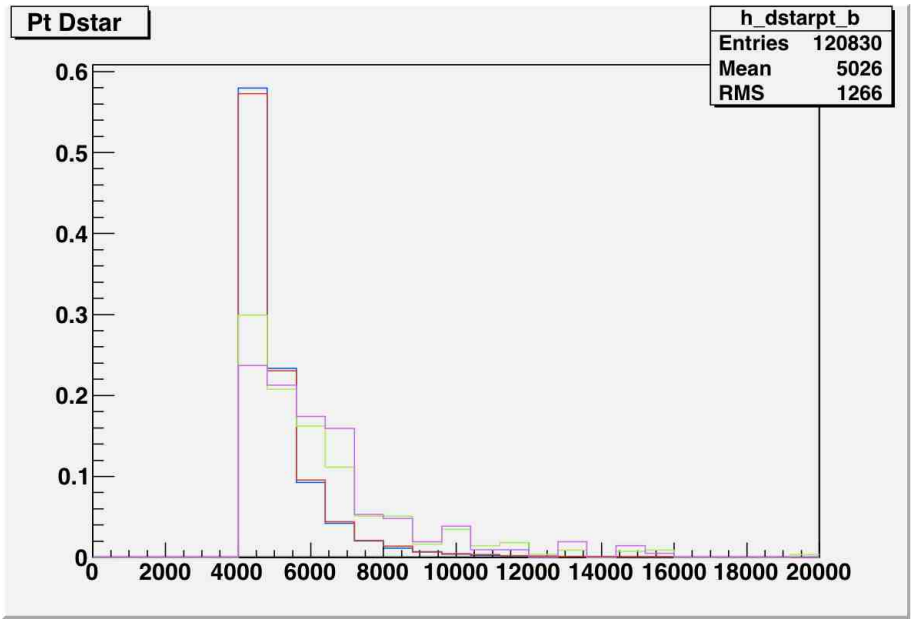

FIGURE 7.4 – Histogrammes pour différentes données de $P_T D^{*\pm}$ pour le méson $D^{*\pm}$

Une différence notable est le fait que le PMC s'étend jusqu'à environ 7000 MeV/c avec encore un nombre d'entrées assez important. Ce n'est pas le cas pour SB qui s'atténue assez rapidement dès 7000 MeV/c.

Encore une fois, nous nous attendons à une coupure exploitable aux alentours de 5000 MeV/c.

Etude de $P_T D^{*\pm} - P_T D^0$

Puis, observons la différence des impulsions transverses des mésons $D^{*\pm}$ et D^0 en figure 7.5.

Le décalage des données SB et PMC devient beaucoup plus clair sur cet histogramme. En effet, le maximum d'entrée pour les données PMC est obtenue pour un intervalle compris entre 300 MeV/c et 550 MeV/c.

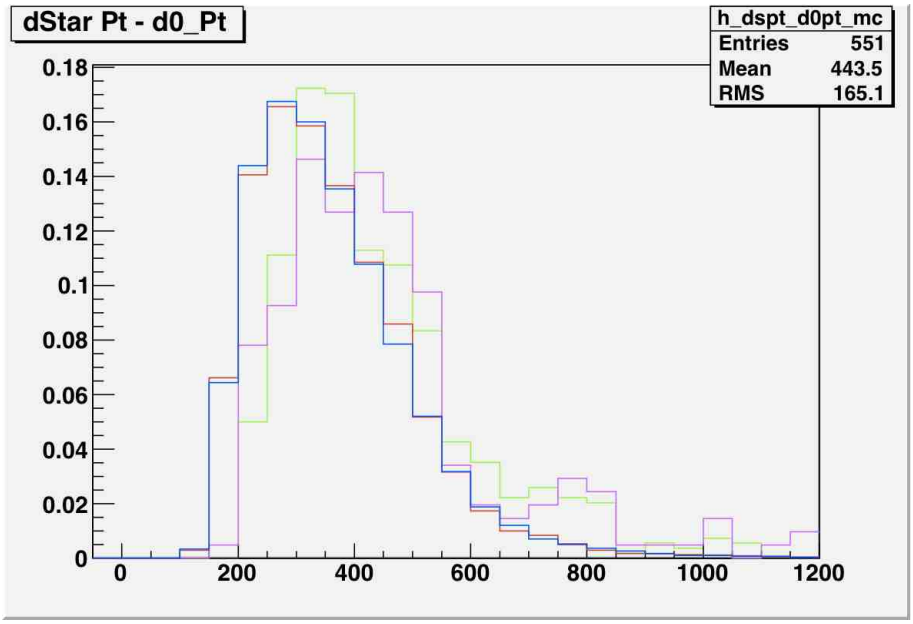

FIGURE 7.5 – Histogrammes pour différentes données de $P_T D^{*\pm} - P_T D^0$.

Ce n'est pas le cas pour SB qui s'étend de 200 MeV/c à 450 MeV/c. Nous voyons également que pour RMC, le maximum est atteint à environ 17% comme pour le signal SB. En revanche, avec le signal reconstruit PMC, nous constatons une légère baisse jusqu'à 14%.

On s'attend donc à une coupure aux alentour de 500 MeV/c.

Etude de $\frac{P_T D^{*\pm}}{\sum P_T}$

La figure 7.6 nous montre une différence entre histogrammes très nette.

L'histogramme PMC marque un décalage très notable de 0,08 à 0,26 avec un maximum à 14% environ contrairement à SB qui augmente plus tôt dès 0,06 à 29% pour chuter à 0,08% à 0,14.

Une coupure entre 0,1 et 0,2 serait sans doute très intéressante pour notre étude.

Etude de $\frac{P_T D^{*\pm}}{\sum P_T^2}$

Nous faisons des constations similaires sur la figure 7.7.

En effet, le décalage des données PMC est notable encore une fois et s'étend de $0,04^{-3}$ MeV^{-1}/c à $0,16 \times 10^{-3}$ MeV^{-1}/c environ avec un maximum à 15% contrairement au SB qui va de $0,02 \times 10^{-3}$ MeV^{-1}/c à $0,12 \times 10^{-3}$ MeV^{-1}/c environ pour un maximum atteint de 29%. La différence du nombre d'entrées entre les deux distribution est également semblable au cas $\frac{P_T D^{*\pm}}{\sum P_T}$.

Nous pouvons envisager une coupure aux alentours de $0,10^{-3}$ MeV^{-1}/c.

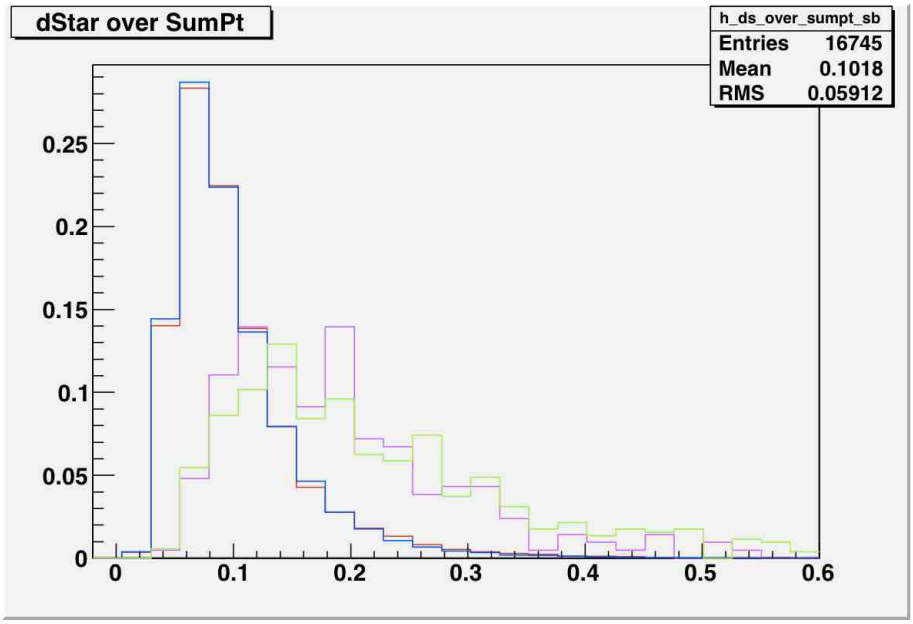

FIGURE 7.6 – Histogrammes pour différentes données de $\frac{P_T D^{*\pm}}{\sum P_T}$ pour le méson $D^{*\pm}$.

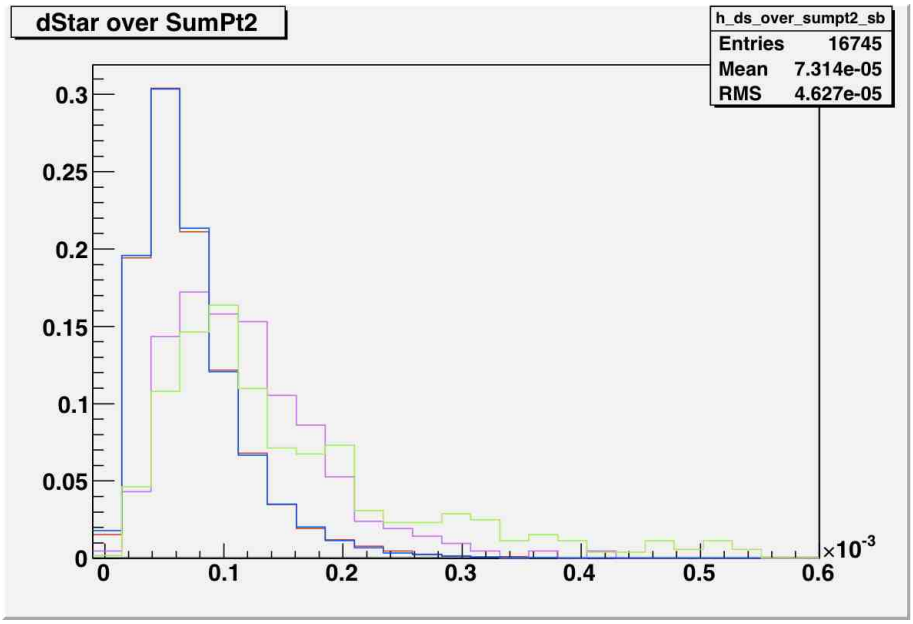

FIGURE 7.7 – Histogrammes pour différentes données de $\frac{P_T D^{*\pm}}{\sum P_T^2}$ pour le méson $D^{*\pm}$.

Etude de $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$

La figure 7.8, nous présente des différences encore plus évidentes que les cas précédents.

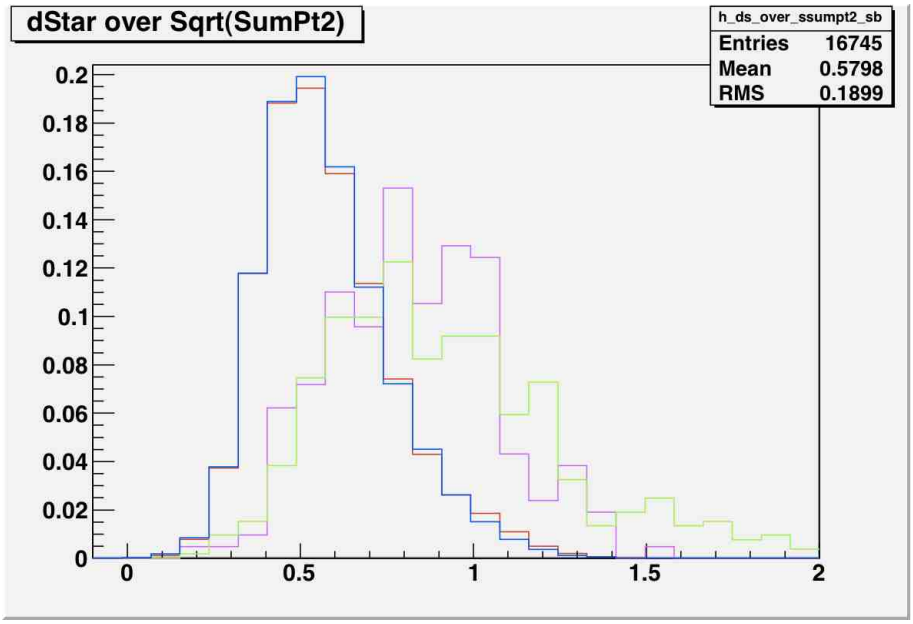

FIGURE 7.8 – Histogrammes pour différentes données de $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$ pour le méson $D^{*\pm}$.

La distribution PMC est beaucoup plus décalée vers les valeurs positives que SB. En effet, PMC s'étend, toujours pour un maximum d'entrée, de 0,6 à 1,1 environ avec un maximum à 15%.

Ce n'est pas le cas du SB qui s'étend quant à lui de 0,3 à 0,8 pour un maximum d'environ 20%.

On s'attend donc à une coupure raisonnable aux alentours de 0,7. Les trois dernières variables que nous venons d'étudier s'avèrent donc particulièrement utiles pour nos futures améliorations.

Etude de Δr

Nous obtenons ici un cas différent en ce qui concerne le nombre d'entrées pour chacune des distributions lorsque l'on observe la figure 7.9.

Nous avons en effet un maximum d'entrée pour PMC de 75% environ de 0, 005 MeV à 0,01 MeV. Une forte concentration d'entrées lorsque l'on effectue la comparaison avec SB qui s'étend plus largement de 0,005 MeV à 0,02 MeV avec un maximum d'entrée de l'ordre de 45% seulement.

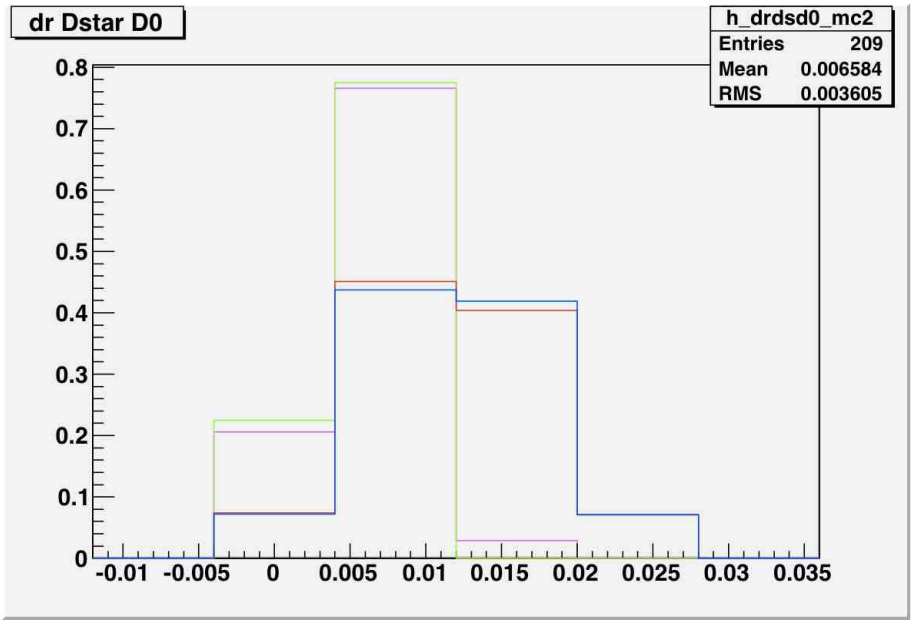

FIGURE 7.9 – Histogrammes pour différentes données de Δr entre les mésons $D^{*\pm}$ et D^0.

On en déduit la possible coupure vers 0,005 MeV, à confirmer avec les graphes d'efficacité/pureté.

7.2 Calcul de l'efficacité et de la pureté du signal

Le but est d'utiliser les variables précédentes afin de convenir de coupures potentiellement intéressantes. Cependant, nous l'avons vu dans la partie précédente, nous ne pouvons que donner des approximations de coupures.

Dès lors, il est nécessaire d'avoir des informations sur la variation de la pureté du signal en fonction de cette coupure ainsi que de son efficacité.

Nous utiliserons deux macros développées par M. David Calvet nommées `DStarPlot.C` et `EffPur.C` qui permettent respectivement de calculer la pureté du signal en fonction de la coupure associée et de tracer les graphes d'efficacité en fonction de la pureté du signal.

7.2.1 Calcul de l'efficacité

Dans une premier temps, il nous faut procéder au calcul de la pureté du signal. Nous décrivons l'algorithme utilisé pour ce calcul dans la partie suivante [1].

Etude du programme Pour pouvoir extraire les coupures les plus intéressantes, nous considérerons comme signal, les données issues du PMC et comme bruit, les données B. Ceci conduit à l'utilisation du code suivant :

1. Vous trouverez en annexes les codes originaux des programmes utilisés dans ce mémoire.

```
tree_mc2->Draw("Diff_masse >> h_diffmasse","type_signal==2 && pp_Pt>1500
&& dStar_Pt>4000");
tree_data->Draw("Diff_masse >> h_diffmasse_1","type_signal==1");
```

Evidemment, nous veillerons à comparer des informations comparables en appliquant les coupures sur les impulsions du premier pion (PION1) issu du D^0 (PION1) et du méson $D^{*\pm}$.

L'étape suivante consiste à effectuer une normalisation du bruit de fond afin de compenser l'éventuel (et probable) différence entre le nombre d'entrées du signal et du bruit de fond. Pour cela, nous utiliserons les données `type_signal==1` et `type_signal==0` des vraies données respectivement pour le signal et le bruit :

```
Double_t nh4=h_diffmasse_b->Integral(minBin,maxBin); // nh4 bruit
std::cout << "Number of events in h4=" << nh4 << std::endl;
Double_t nh3=h_diffmasse_1->Integral(minBin,maxBin); // nh3 signal
std::cout << "Number of events in h3=" << nh3 << std::endl;
```

Nous calculons donc ainsi le nombre d'entrées entre les `minBin` et `maxBin` précédemment définis :

```
Int_t minBin=h_diffmasse_b->FindBin(0.150);
Int_t maxBin=h_diffmasse_b->FindBin(0.170);
```

En veillant bien évidemment à se placer en dehors de la fenêtre de masse du méson $D^{*\pm}$ (i.e. après 145 MeV).

Le rapport de `nh3` sur `nh4` permet ainsi de calculer un bruit de fond normalisé :

```
Int_t nbins=h_diffmasse_b->GetNbinsX();
Double_t lowBin=h_diffmasse_b->GetBinLowEdge(1);
Double_t highBin=h_diffmasse_b->GetBinLowEdge(nbins+1);
TH1F *h5=new TH1F("h5","Normalized background",nbins,lowBin,highBin);
h5->Add(h_diffmasse_b,nh3/nh4);
```

Puis nous traçons également le signal PMC dans un autre canvas :

```
TCanvas *c15=new TCanvas ("c15","Histo15");
TH1F *h6=new TH1F("h6","Signal",nbins,lowBin,highBin);
h6->Add(h_diffmasse);
```

Puis vient le moment de sélectionner la fenêtre de masse du méson $D^{*\pm}$ qui nous intéresse :

```
minBin=h6->FindBin(145.4e-3-1.5e-3);
maxBin=h6->FindBin(145.4e-3+1.5e-3);
```

Puis nous calculons ensuite le nombre d'évènements présents dans cette fenêtre de masse pour le signal avec l'histogramme `h6` :

```
Double_t nSig=h6->Integral(minBin,maxBin)*wht;
std::cout << "Number of events in peak (h6)= " << nSig << " +/- "
<< sqrt(nSig) << std::endl;
```

Il est également capital de noter l'intervention du coefficient `wht` qui, lui, effectue une renormalisation du signal dû à la différence entre les statistiques PMC et B. En effet, rappelons-nous que nous utilisons pour ces calculs, 28 millions d'évènements de B et 5 millions d'évènements PMC [2].

Puis, nous calculons également le nombre d'entrées pour la même fenêtre de masse mais, cette fois-ci, pour le bruit :

```
Double_t nBkg=h5->Integral(minBin,maxBin);
  std::cout << "Number of events in scaled bkg (h5)=" << nBkg << " +/- "
      << sqrt(nBkg) << std::endl;
```

Puis, il ne nous reste plus qu'à calculer la pureté de notre signal qui s'écrit par définition :

$$Purity = \frac{nSig}{nSig + nBkg} \times 100 \pm \left[\frac{nSig}{nSig + nBkg} \times \sqrt{\frac{\frac{1-nSig}{nSig+nBkg}}{nSig}} \times 100 \right] \tag{7.1}$$

avec le valeur de l'erreur qui est située après le signe $\pm$.

Avec ce programme et en appliquant les coupures correspondantes au signal PMC, il nous est alors possible d'obtenir la pureté du signal en fonction de la coupure appliquée ainsi que le nombre d'évènements correspondants. Ces données seront utilisées ultérieurement pour le calcul de l'efficacité comme nous le verrons.

7.2.2 Calcul de la pureté

Maintenant que nous avons calculé la pureté de notre signal en fonction des coupures appliquées, il est intéressant de représenter l'évolution de l'efficacité de notre signal en fonction de sa pureté.

Ainsi, nous comprenons rapidement qu'il sera possible d'en conclure sur les coupures exploitables et la meilleure valeur à appliquer sur le signal pour avoir le meilleur compromis entre une **pureté adéquate du signal avec la meilleur efficacité possible**.

Ceci nous sera possible après avoir expliqué brièvement le principe de calcul de l'efficacité.

Etude du programme La structure de ce dernier programme est assez simple. Le calcul de l'efficacité du signal ainsi que son erreur se calculent à partir d'un signal de référence noté `signal0` :

```
    const double efficiency=nSig/signal0;
    const double effError=sqrt(efficiency*(1-efficiency)/signal0);
```

Dès lors, si le `signal0` est égal au `nSig`, ce qui est le cas lors de la première coupure, alors nous avons bien une efficacité de 1 comme nous le verrons dans les résultats qui suivent.

2. Le rapport `wht` est égal au rapport exact $\frac{27786384}{4998189}$.

7.3 Résultats obtenus

Dans cette partie, nous présentons les graphes d'efficacité et de pureté obtenus en fonction des différentes valeurs de coupures effectuées pour les variables qui nous intéressent.

Notons également que la pureté sera notée par une croix + et l'efficacité, par un point •.

Etude de $cos\theta^*_K$

Sur la figure 7.10, nous avons une coupure pour une valeur de 0 pour laquelle l'efficacité et la pureté seront de 35% environ chacune.

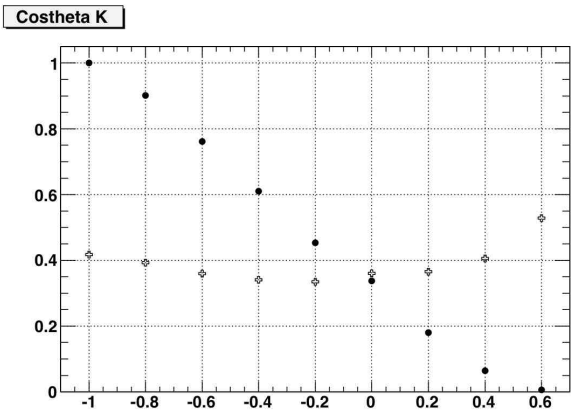

FIGURE 7.10 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $cos\theta^*_K$.

Etude de Δr

Sur la figure 7.11, nous n'observons pas de coupures intéressantes. En effet, la pureté chute très rapidement, sans augmenter.

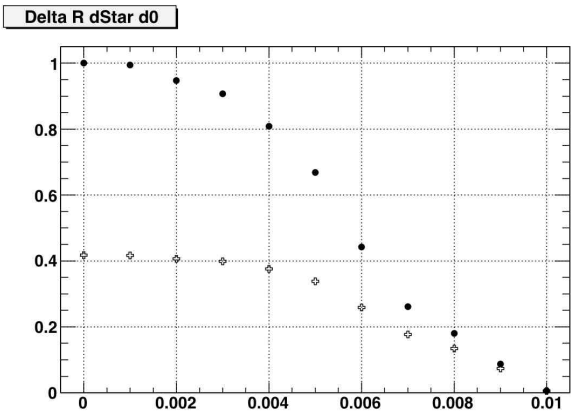

FIGURE 7.11 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable Δr.

Etude de $P_T D^0$

Avec la figure 7.12, nous observons une coupure intéressante à 5000 MeV/c qui nous permet d'avoir une efficacité/pureté de 65% environ. La coupure est de bonne qualité.

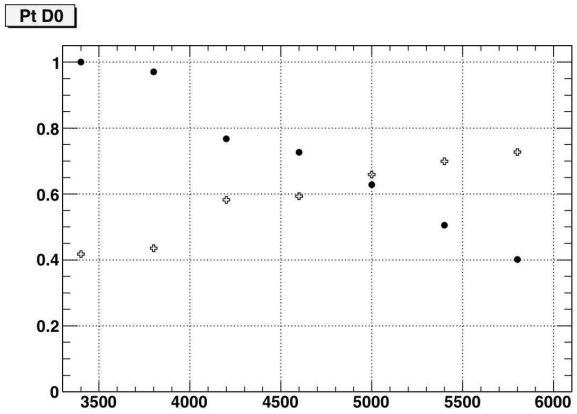

FIGURE 7.12 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $P_T D^0$.

Etude de $P_T D^{*\pm}$

Avec la figure 7.13, nous observons une coupure intéressante à 5200 MeV/c qui nous permet d'avoir une efficacité/pureté de 65% environ. La coupure est également de bonne qualité.

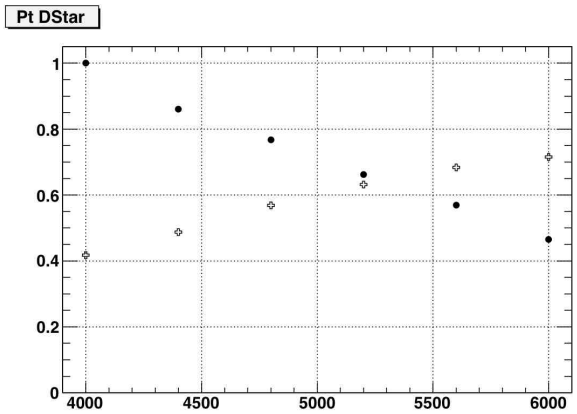

FIGURE 7.13 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $P_T D^{*\pm}$.

Etude de $P_T D^{*\pm} - P_T D^0$

Sur la figure 7.14, nous observons une coupure intéressante à 400 MeV/c qui nous permet d'avoir une efficacité/pureté de 55% environ.

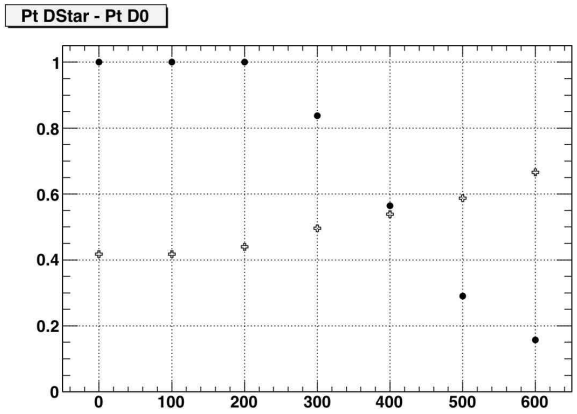

FIGURE 7.14 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $P_T D^{*\pm} - P_T D^0$.

Etude de $\frac{P_T D^{*\pm}}{\sum P_T}$

Comme nous le montre la figure 7.15, nous avons une efficacité et une pureté de 70% environ à 0,12 environ. Cette coupure est donc de bonne facture.

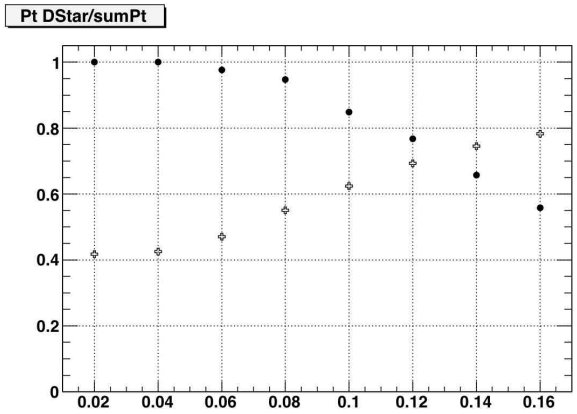

FIGURE 7.15 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $\frac{P_T D^{*\pm}}{\sum P_T}$.

Etude de $\frac{P_T D^{*\pm}}{\sum P_T^2}$

Sur cette figure 7.16, une observation similaire à 0,09 MeV^{-1}/c mais pour une efficacité/pureté un peu plus faible de l'ordre de 65% environ.

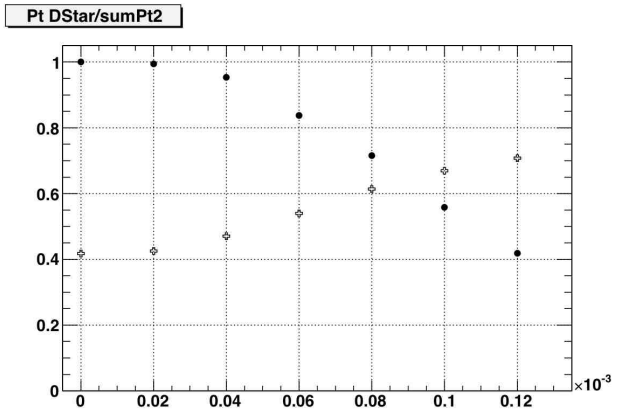

FIGURE 7.16 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $\frac{P_T D^{*\pm}}{\sum P_T^2}$.

Etude de $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$

Sur la figure 7.17, nous revenons à une efficacité/pureré de l'ordre de 70% aux alentours de 0,7. Encore une fois, nous avons ici une coupure qui peut-être assez bonne.

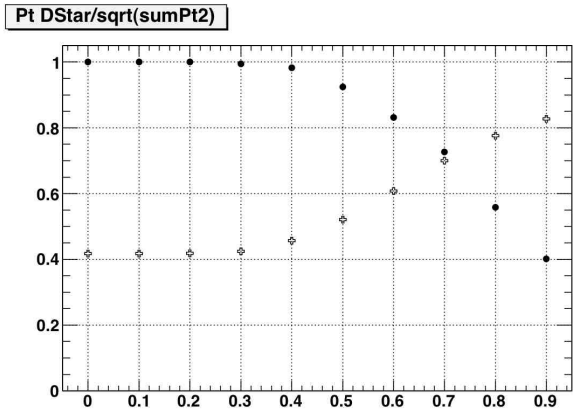

FIGURE 7.17 – Graphe d'efficacité et de pureté du signal pour plusieurs coupures sur la variable $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$.

7.3.1 Récapitulatif des coupures

Nous regroupons enfin ces graphes d'efficacité et de pureté du signal sur un graphe récapitulatif afin d'en extraire les coupures les plus judicieuses à utiliser. Evidemment, il s'agira de trouver les variables pour lesquelles les coupures présentent le meilleur compromis entre efficacité et pureté du signal à traiter.

Coupure sur les impulsions transverses La figure 7.18 nous permet de déduire les coupures les plus intéressantes.

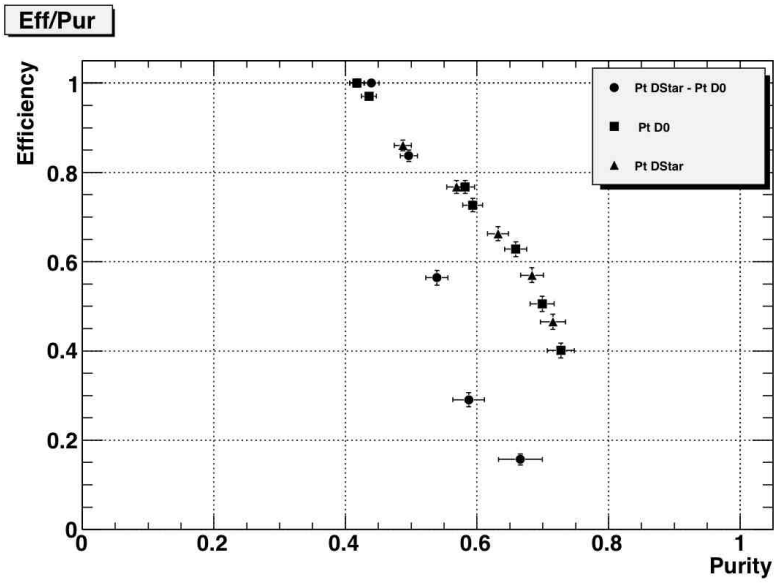

FIGURE 7.18 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion.

A priori, les deux variables sur lesquelles nous pourrions travailler sont les $P_T D^0$ et $P_T D^{*\pm}$. Les courbes, pour ces deux variables, sont très proches l'une de l'autre donc nous ne pouvons en choisir l'une plutôt que l'autre.

Nous pouvons en revanche écarter la variable $P_T D^{*\pm} - P_T D^0$.

Coupure sur les impulsions divisée par des sommes d'impulsions La figure 7.19 nous montre que les coupures les plus utiles doivent être appliquées sur les variables $\frac{P_T D^{*\pm}}{\sum P_T}$ et $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$.

7.4 Sélection des meilleures coupures

Enfin, nous décidons de tracer, sur le même graphe, les courbes correspondantes aux variables $P_T D^0$ et $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$ avec la figure 7.20.

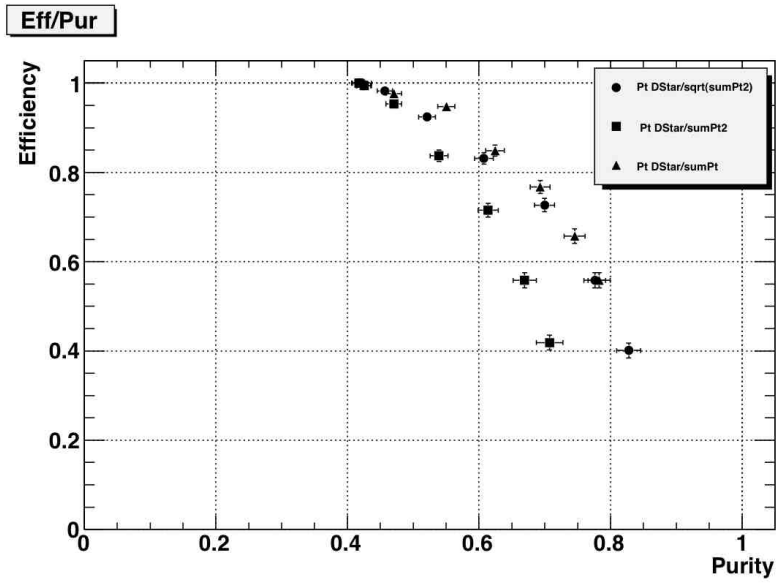

FIGURE 7.19 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion.

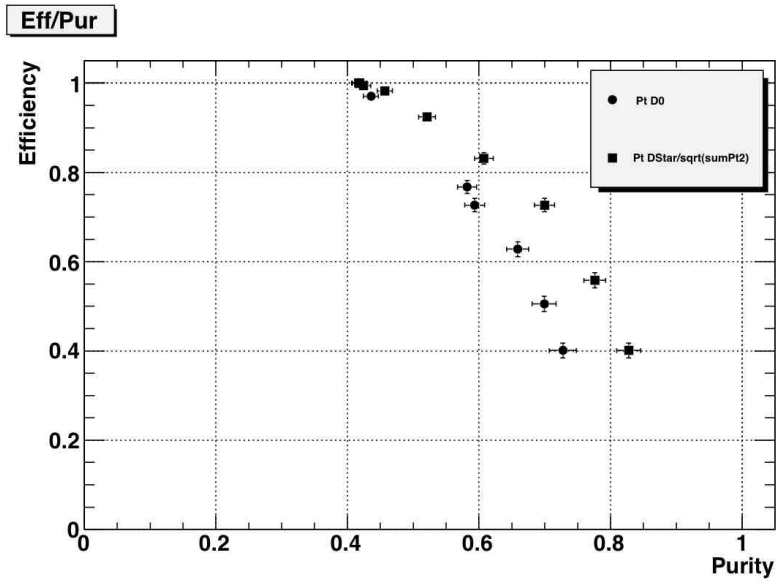

FIGURE 7.20 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion.

Dès lors, il devient clair que la variable prioritaire sur laquelle nous devrons travailler est la variable $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$ dont les coupures s'avèrent particulièrement efficaces.

La coupure que nous sélectionneront est celle qui fait perdre le moins d'efficacité pour la pureté la plus grande. Ainsi, pour une diminution de 20 % environ de l'efficacité, nous avons une pureté de 60% ce qui est satisfaisant. Cette coupure-ci sera importante pour la prochaine partie de l'étude.

Chapitre 8

Amélioration des coupures sur le signal

Comme nous l'avons vu dans le chapitre précédent, la variable sur laquelle il est le plus intéressant d'effectuer des coupures est la suivante :

$$\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6 \tag{8.1}$$

En effet, cette coupure nous permet d'avoir une diminution de l'efficacité du signal de l'ordre de 20% seulement pour une pureté d'environ 60% ce qui semble un bon compromis relatif.

Dès lors, nous effectuons des comparaisons en incorporant ces coupures par défaut, dans nos calculs d'efficacité et de pureté qui vont suivre afin de voir quelles coupures exploitables sont susceptibles d'être rajoutées pour améliorer le résultat.

Les variables que nous étudierons, en fonction de la coupure initiale rajoutée, seront les suivantes : $P_T D^0$, $P_T D^{*\pm}$ et $\frac{P_T D^{*\pm}}{\sum P_T}$. Ce seront celles sur lesquelles nous pensons que les coupures peuvent s'avérer exploitables.

8.1 Evolution des autres coupures

Dans cette partie, nous proposons d'observer si les coupures précédentes sur les variables qui nous ne semblaient pas intéressantes par défaut, le deviennent avec l'ajout de cette coupure $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6$.

La seule amélioration notable que nous avons constaté est celle concernant la variable $\Delta r D^{*\pm}/D^0$.

En effet, d'après la figure 8.1, nous constatons une augmentation de l'efficacité et de la pureté du signal d'environ 40% pour 0,006 MeV. L'amélioration est notable pour cette variable.

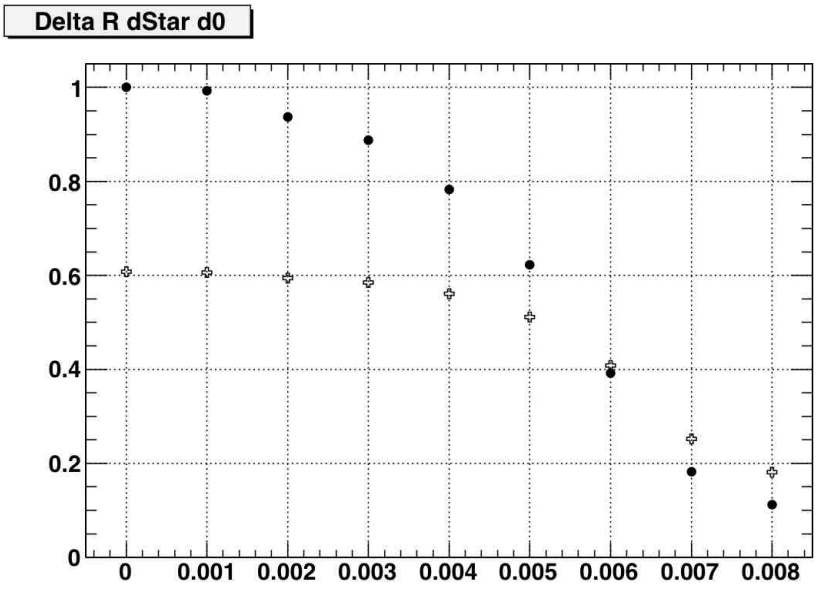

FIGURE 8.1 – Graphe d'efficacité et de pureté du signal pour la variable $\Delta r D^{*\pm}/D^0$ avec la coupure $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6$.

Cependant, une efficacité et une pureté d'environ 40% ne sont pas suffisantes pour pouvoir espérer une coupure qui ait de l'intérêt pour l'instant.
Nous continuons donc nos coupures sur les variables qui nous semblent pertinentes.

8.2 Evolution des coupures intéressantes après application

Nous retraçons de nouveau un graphe d'efficacité pureté en prenant en compte la coupure sur $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6$.

On obtient ainsi la figure 8.2 qui nous permet de constater qu'une coupure supplémentaire sur la variable $\frac{P_T D^{*\pm}}{\sum P_T}$ nous permettrait d'obtenir une pureté de 70% environ pour une diminution de l'efficacité de seulement 10% ! Un très bon compromis donc.

La prochaine coupure que nous devrons appliquer est donc la suivante :

$$\frac{P_T D^{*\pm}}{\sum P_T} > 0,1 \tag{8.2}$$

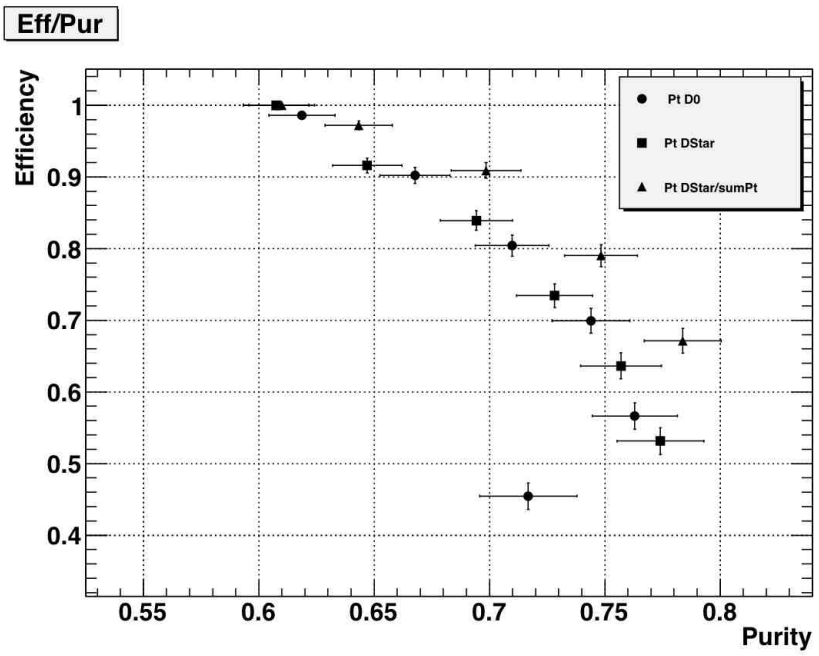

FIGURE 8.2 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6$.

8.2.1 Evolution des coupures après application de $\frac{P_T D^{*\pm}}{\sum P_T} > 0,1$

On constate des améliorations intéressantes sur la variables $P_T D^0$. Les autres variables subissent de trop faibles améliorations pour être prises en compte.

Variable $P_T D^0$ D'après la figure 8.3, on observe donc une augmentation de l'efficacité/pureté de l'ordre de 10%, puisque par rapport à la figure 7.12, nous passons de 65% à 75% à 4800 MeV/c.

Rajoutons donc cette coupure au programme afin de trouver une autre coupure intéressante.

8.2.2 Rajout supplémentaire de $P_T D^0 > 4800$ MeV/c

De la même manière que précédemment, le même mode opératoire est mis en place. Nous appliquons la coupure supplémentaire dans les calculs puis nous regardons l'évolution des coupures sur les variables.

Cette fois-ci, nous notons une nette amélioration des coupures sur les variables $cos\theta_K^*$, Δr, $P_T D^{*\pm}$, $P_T D^{*\pm} - P_T D^0$ et $\frac{P_T D^{*\pm}}{\sum P_T^2}$. Dans la majeure partie des cas observés, la pureté atteinte se situe aux alentours de 75%.

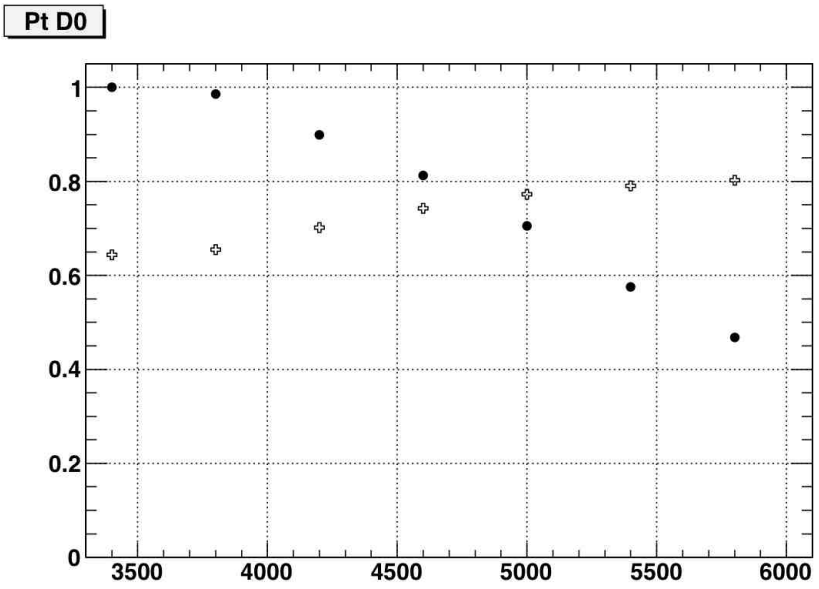

FIGURE 8.3 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $\frac{P_T D^{*\pm}}{\sum P_T} > 0,1$.

Cependant, après avoir tracé les meilleures coupures, nous obtenons la figure 8.4.

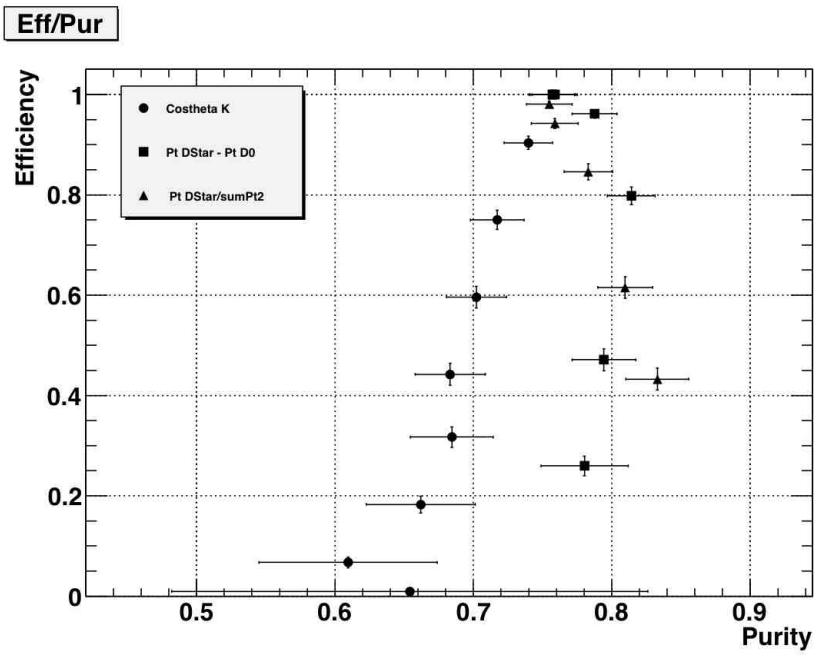

FIGURE 8.4 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $P_T D^0 > 4800$ MeV/c.

Il devient alors évident que la prochaine coupure que nous allons effectuer sur le signal sera :

$$P_T D^{*\pm} - P_T D^0 > 400 MeV/c \tag{8.3}$$

Ainsi pour une diminution de seulement 20% de l'efficacité du signal, nous somme à plus de 80% de sa pureté !

8.2.3 Rajout supplémentaire de $P_T D^{*\pm} - P_T D^0 > 400$ MeV/c

Toujours en utilisant le même mode opératoire, nous obtenons la figure 8.5 qui présente le résumé des meilleures coupures actuelles.

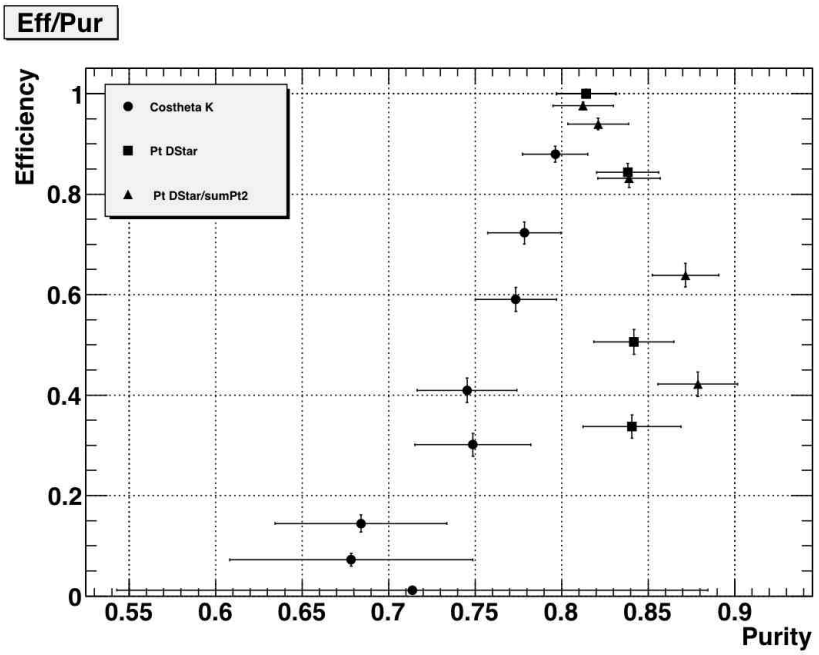

FIGURE 8.5 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $P_T D^{*\pm} - P_T D^0 > 400$ MeV/c.

Nous voyons désormais que le choix de coupure s'avère plus compliqué. Evidemment, nous pouvons tout de même en conclure que la prochaine meilleure variable sur laquelle couper est $\frac{P_T D^{*\pm}}{\sum P_T^2}$ mais pour quelle valeur ?

En effet, si l'on diminue d'environ 20% notre pureté, nous gagnons 2% de pureté et si l'on diminue de 35% la pureté, nous gagnons 5% de pureté du signal. La deuxième option n'est pas envisageable car nous ne souhaitons pas tomber à une efficacité d'environ 65% ce qui rendrait nos précédentes coupures inutiles.

Dès lors, la prochaine coupure à appliquer sera la suivante :

$$\frac{P_T D^{*\pm}}{\sum P_T^2} > 0,08 \times 10^{-3} MeV^{-1}/c \tag{8.4}$$

Puis, nous continuons donc notre optimisation dans la section suivante.

8.2.4 Rajout supplémentaire de $\frac{P_T D^{*\pm}}{\sum P_T^2} > 0,08 \times 10^{-3}\ MeV^{-1}$/c

Les améliorations deviennent maintenant délicates à détecter. Nous gagnons néanmoins quelques pourcents au-dessus des 80%. La figure 8.6 présente un zoom de la comparaison d'efficacité/pureté entre les deux dernières variables intéressantes, Δr, $P_T D^{*\pm}$ et $P_T D^{*\pm}$, dans la région qui nous intéresse.

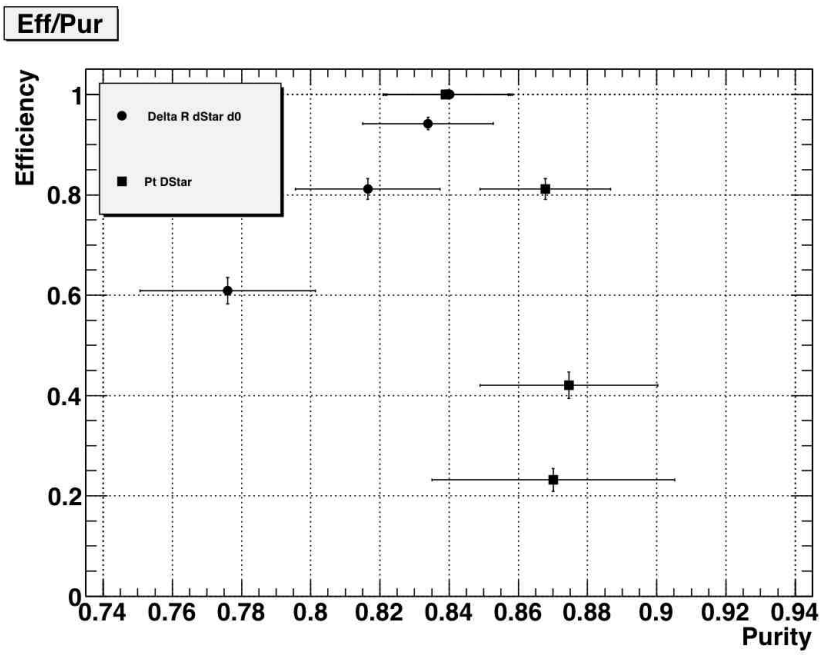

FIGURE 8.6 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $\frac{P_T D^{*\pm}}{\sum P_T^2} > 0,08 \times 10^{-3}\ MeV^{-1}$/c.

Notons également que la contribution de Δr, $P_T D^{*\pm}$ et $P_T D^{*\pm}$ présente une amélioration très faible au fur et à mesure des coupures. Nous pouvons donc à présent conclure, que la prochaine coupure qui suit, sera la dernière de notre étude :

$$P_T D^{*\pm} > 6000 MeV/c \tag{8.5}$$

8.2.5 Rajout supplémentaire de $P_T D^{*\pm} > 6000$ MeV/c

Enfin, en observant les deux dernières variables $cos\theta^*_K$ et Δr, $P_T D^{*\pm}$ sur la figure 8.7, nous nous rendons compte que l'amélioration significative n'est plus possible ici. On ne peut différencier véritablement la meilleure variable à utiliser.

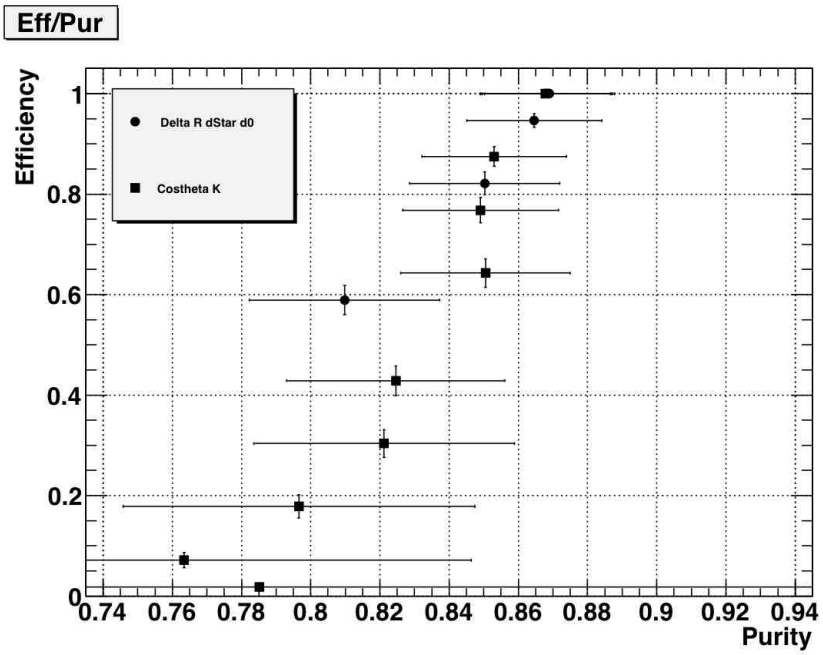

FIGURE 8.7 – Graphe d'efficacité et de pureté du signal pour plusieurs variables d'impulsion avec la coupure $P_T D^{*\pm} > 6000$ MeV/c.

Dès lors, nous considérons l'amélioration du signal terminée à ce stade de notre étude.

8.3 Résumé des coupures apportées au signal

Pour terminer notre étude sur l'optimisation du rapport signal sur bruit pour la reconstruction de méson $D^{*\pm}$, nous récapitulons ici, l'ensemble des coupures à effectuer pour améliorer notre signal :

$$\begin{gathered}
\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}} > 0,6 \\
\frac{P_T D^{*\pm}}{\sum P_T} > 0,1 \\
P_T D^0 > 4800 MeV/c \\
P_T D^{*\pm} - P_T D^0 > 400 MeV/c \\
\frac{P_T D^{*\pm}}{\sum P_T^2} > 0,08 \times 10^{-3} MeV^{-1}/c \\
P_T D^{*\pm} > 6000 MeV/c
\end{gathered} \tag{8.6}$$

Le tableau suivant permet de récapituler brièvement l'influence de ces coupures sur le signal :

Type	Without cuts	With cuts
Events	860 ± 29,3258	280 ± 16,7332
Purity	0,41736 ± 1,08633 %	0,867762 ± 1,88582 %

Les coupures que nous avons appliqué au signal PMC nous permettent donc d'augmenter notre pureté de plus de 40% environ. Le gain est donc bien significatif à l'aide de toutes ces contraintes appliquées sur notre signal.

Il s'agit maintenant de conclure notre travail en appliquant ces coupures au signal des données en provenance du CERN, sur nos 28 millions d'évènements.

Chapitre 9

Application des coupures au données réelles

Nous avons donc déterminé la totalité des coupures intéressantes à appliquer à notre signal issu des vraies données du LHC afin d'optimiser le rapport signal sur bruit.

Sans appliquer aucune coupure au signal, la différence de masse entre le méson $D^{*\pm}$ et le méson D^0, nous donne le résultat de la figure 9.1.

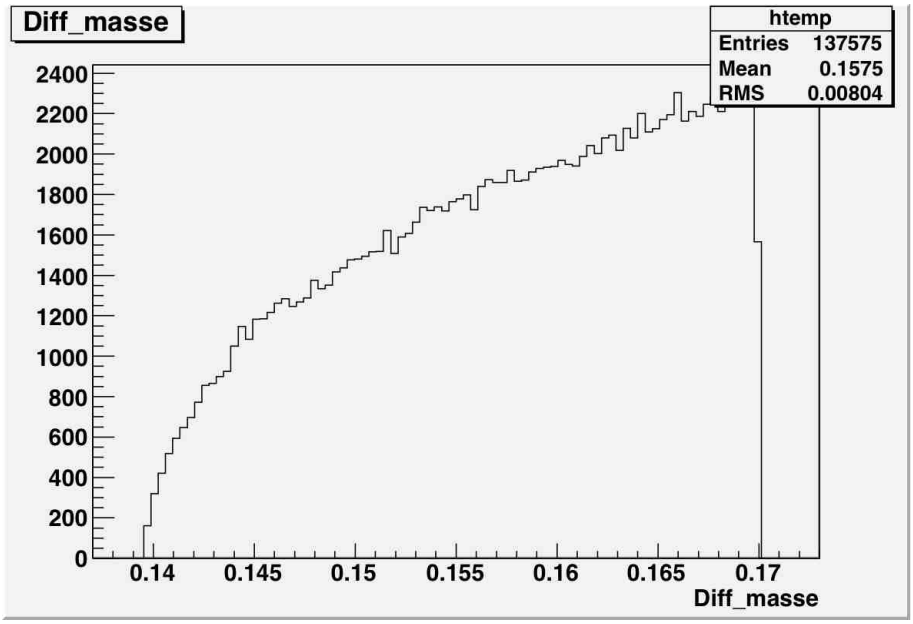

FIGURE 9.1 – Différence de masse entre le méson $D^{*\pm}$ et le méson D^0 sur 28 millions d'évènements, sans coupure.

Puis, en appliquant nos coupures successives, nous obtenons finalement la figure 9.2.

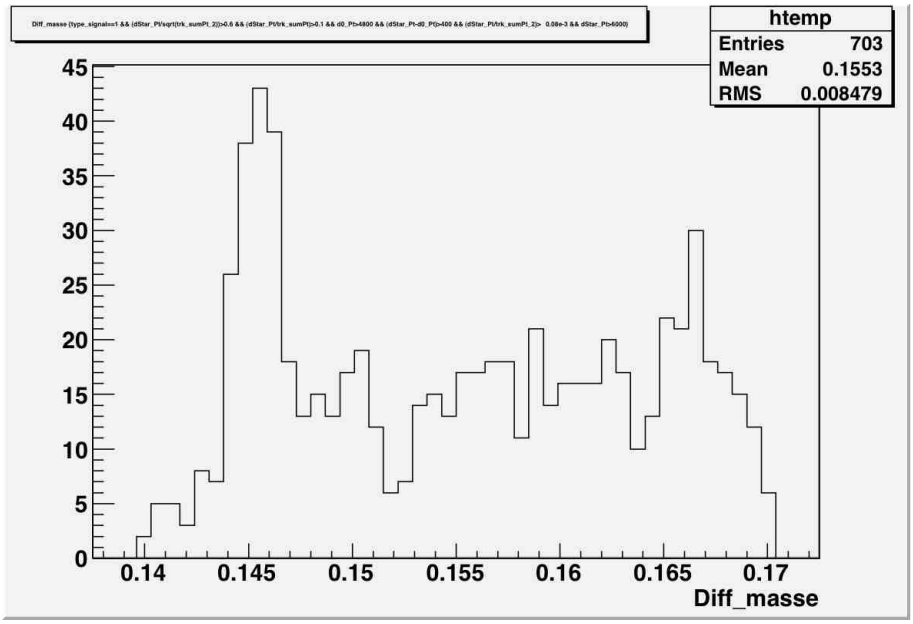

FIGURE 9.2 – Différence de masse entre le méson $D^{*\pm}$ et le méson D^0 sur 28 millions d'évènements, avec coupures.

Le pic à 145 MeV est clairement observable et le résultat est en conformité avec ce que nous attendons. Notons également la diminution très importante du nombre d'évènements (nous passons de 137 575 à 703 évènements avec nos coupures).

Pour terminer, il est maintenant utile de constater le gain en pureté acquise avec ces coupures lorsque l'on dénombre le nombre d'entrée situé dans le pic autour de 145 MeV avec et sans les coupures. Le tableau suivant résume les données :

Type	Without cuts	With cuts
Events in pic	2197 ± 46,8738	576,655 ± 24,0136
Purity	64,6655 ± 0,820053 %	93,1104 ± 1,01774 %

Finalement, nous constatons que nos coupures appliquées sur le signal des vraies données SB nous permet un gain en pureté de l'ordre de 28% environ pour une efficacité d'environ 73,75 %. Le gain est donc appréciable pour l'amélioration du rapport signal sur bruit du signal.

Chapitre 10

Conclusion

Les dernières données de mars jusqu'à mai 2010 en provenance du LHC du CERN nous ont permis d'étudier les premières collisions *pp* à 7 TeV, dans le référentiel du centre de masse.

La désintégrations des mésons $D^{*\pm}$ en mésons D^0 représente environ 68% des cas. Cependant, la désintégration du méson D^0 dans le canal étudié dans ce rapport ne représente que seulement 4% des cas. Dès lors, pour la reconstruction numérique et l'élaboration de modèles théoriques, il est nécessaire d'avoir le meilleur rapport signal sur bruit possible.

Nous avons travaillé sur une grande quantité de *vraies* données : environ 28 millions d'évènements à traiter. Il a fallu tout d'abord optimiser les précédents programmes afin qu'ils conviennent à notre tâche principale : l'optimisation du rapport signal sur bruit. Puis, nous avons cherché à analyser de nouvelles variables d'études afin de voir si des coupures sur celles-ci, pourraient engendrer d'autres coupures encore plus adéquates intéressantes.

En étudiant les 8 millions d'évènements des données de simulations numériques Monte-Carlo pour le détecteur ATLAS, issues du CERN, nous avons pu observer le comportement théorique, attendu, des distributions de ces variables. Ainsi, en modélisant les coupures intrinsèques du détecteur, nous avons pu observer l'évolution théorique du rapport signal sur bruit en fonction des variables utilisées.

Dans tous les cas, nous constatons que les meilleures variables sur lesquelles nous devons effectuer les coupures sont celles qui prennent en compte les impulsions transverses des mésons. C'était d'ailleurs déjà le cas avant cette étude. Cependant, l'apport de ce rapport est de montrer qu'il existe trois grandes nouvelles variables sur lesquelles il sera très intéressant d'appliquer des coupures : les variables $\frac{P_T D^{*\pm}}{\sum P_T}$, $\frac{P_T D^{*\pm}}{\sum P_T^2}$ et $\frac{P_T D^{*\pm}}{\sqrt{\sum P_T^2}}$.

Ainsi, avec les travaux effectués, nous pouvons prétendre à une amélioration théorique du rapport signal sur bruit d'environ 30% pour les futures études sur les mésons $D^{*\pm}$.

Actuellement, ces mésons $D^{*\pm}$ sont encore à l'étude et nous espérons que le présent rapport permettra d'affiner les modèles et les analyses numériques afin de percer un peu plus, les mystères de la création de notre Univers ...

10.1 Bilan du stagiaire

Lors de mon arrivée au LPC, le 21 juin 2010, j'ai été très agréablement accueilli par David et toute l'équipe ATLAS. J'ai eu l'occasion de me faire connaître de l'équipe par une petite présentation dans chaque bureau, ce que j'ai beaucoup apprécié pour me mettre dans l'ambiance de travail du laboratoire.

Les premiers temps ont consisté en des recherches documentaires sur le sujet de stage : lecture des thèses de David et de M. Defay ainsi que le rapport de mon prédécesseur, Loïc Valéry. J'ai apprécié notamment ce temps pour connaître encore plus de détails sur le LHC et son fonctionnement en détail. La visite de la partie *instrumentation* du laboratoire par David m'a également beaucoup plue me permettant de voir la grande implication du laboratoire dans la construction du LHC, au CERN. Evidemment, la partie informatique et notamment la gestion du système réseau m'a stupéfait. La puissance de calcul déployée est effectivement considérable et, malgré quelques pannes occasionnelles, l'organisation du réseau semble est très performante.

La partie programmation a été la plus ardue pour ma part. Je n'ai eu que quelques cours de programmation C/C++ durant mes études et j'ai été plongé directement dans ce langage de programmation par l'intermédiaire du logiciel ROOT du CERN. Ce fut donc la partie la plus difficile de ce stage, certes, mais qui m'a également permis de progresser en algorithmique. Les exercices de David m'ont permis d'appréhender une logique de programmation que je n'avais pas encore comprise. Je le remercie pour cette formation très efficace durant le stage.

Enfin, le reste du temps, j'ai effectué de l'analyse de données (des données du LHC ou de simulations numérique Monte-Carlo). Il est vraiment passionnant de pouvoir réfléchir sur les données d'une telle expérience et de les relier à des phénomènes physiques qui m'intéressent toujours autant.

Pour conclure, ce stage m'a énormément apporté étant donné l'absence de formation en physique des particules à Toulouse, jusqu'au Master 2. J'ai acquis des compétences en programmation que je dois poursuivre pour augmenter mon niveau et une expérience en physique des particules que je n'aurai pu apprendre aussi bien, que dans une équipe de recherche aussi performante que celle du LPC et avec un encadrant aussi rigoureux que David.

10.2 Remerciements

Tout d'abord, je tiens à remercier très chaleureusement celui qui m'a accepté en tant que stagiaire au LPC pour ce stage d'été, David Calvet. Tout a commencé par un échange de mail qui a duré plusieurs mois, assez fastidieux notamment à cause de problèmes d'administration. J'ai eu une véritable opportunité et une chance de travailler pour et avec David durant ce mois et demi. Je lui en suis très reconnaissant.

Je remercie également le directeur du Laboratoire, M. Alain Baldit qui m'a permis également de faire le stage dans son laboratoire, le LPC.

Loïc Valéry a pris environ trois jours de son temps de travail pour m'expliquer en quoi ont consisté ses travaux, me familiariser avec le logiciel ROOT et son environnement de travail, avant que je travaille avec David. Son travail m'a permis d'avancer beaucoup plus rapidement et j'ai été forcé de constaté l'excellent travail qu'il a pu réaliser au sein de l'équipe ATLAS lors de son stage de M1.

Enfin, je remercie également toute l'équipe ATLAS du LPC et notamment Dominique Pallin qui m'a permis de donner une nouvelle trajectoire à mon étude lorsque je m'étais égaré dans mes travaux, en plein milieu du stage. Merci aussi à Jérémy pour son travail et son aide.

Troisième partie

Annexes

10.3 Programmes de traitement des *vraies* données à 7 TeV

10.3.1 DStarRec.C

```
DStarRec() {
  gROOT->LoadMacro("MinBiasTree.C+");
  gROOT->LoadMacro("DStarTree.C+");
  TChain *ch=new TChain("MinBiasTree");

  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152166.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152777.f243_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00153200.f249_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00153159.f249_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00153136.f249_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152214.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00153134.f249_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152844.f243_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152221.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152508.f241_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152845.f243_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152345.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00153030.f247_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152441.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152409.f239_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00152878.f243_p127/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00155112.f255_p176/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00155116.f255_p176/*.root");
  ch->Add("./data/FiltSkim.MinBiasTree.data10_7TeV.00155160.f255_p176/*.root");

  DStarTree t(ch);
  t.Loop();
}
```

10.3.2 DStarTree.C

```
#define DStarTree_cxx
#include "DStarTree.h"
#include <iostream>
#include <bitset>
#include <TLorentzVector.h>
#include <TRotation.h>
#include <TLorentzRotation.h>
#include <TAxis.h>
#include <TCanvas.h>
#include <TH1.h>
#include <TH2.h>
#include <deque>
#include <TMath.h>

DStarTree::DStarTree(TTree *tree) : MinBiasTree(tree) {}

void DStarTree::Loop()
{

  float_t ptmind0=1500; //en MeV
  float_t ptminds=4000; //en MeV
  float_t pixmin=1;
  float_t deltam=20; //en MeV
  float_t d0_max=2.0;  //en mm
  float_t z0sin_max=2.0; //en mm

  if (fChain == 0) return;
  Long64_t nentries(fChain->GetEntries());
  //    nentries = 10;
  std::cout << nentries << std::endl;

  fChain->SetBranchStatus("*",0);
  fChain->SetBranchStatus("ei_EventNumber",1);
  fChain->SetBranchStatus("ei_RunNumber",1);
  fChain->SetBranchStatus("trk_n", 1);
  fChain->SetBranchStatus("trk_d0_wrtPV", 1);
  fChain->SetBranchStatus("trk_z0_wrtPV", 1);
  fChain->SetBranchStatus("trk_phi_wrtPV", 1);
  fChain->SetBranchStatus("trk_theta_wrtPV",1);
  fChain->SetBranchStatus("trk_pt", 1);
  fChain->SetBranchStatus("trk_nPixHits",1);
  fChain->SetBranchStatus("trk_qoverp_wrtPV",1);
  fChain->SetBranchStatus("vx_n",1);
  fChain->SetBranchStatus("vx_type",1);
  fChain->SetBranchStatus("vx_trk_n",1);
  fChain->SetBranchStatus("vx_trk_index",1);

  TLorentzVector kaonn;
  TLorentzVector pionp;
  TLorentzVector kaonp;
  TLorentzVector pionn;
  TLorentzVector vkp;
  TLorentzVector vpk;
  TLorentzVector vpp;
  TLorentzVector vpn;
  TLorentzVector vDSplus;
  TLorentzVector vDSmoins;
  TLorentzVector vDSplus_B;
  TLorentzVector vDSmoins_B;


  TH1F *h12 = new TH1F("Total_diff_sigb","Total Difference masse
  sig+bruit",30,0.135,0.17);
  TH1F *h13 = new TH1F("Total_diff_B","Total Difference de masse
  bruit",30,0.14,0.17);

  const float kaonm = 493.677;  //Mev
  const float pionm = 139.57018;//MeV
  const float d0m = 1864.84;    //MeV

  TCanvas *c4 = new TCanvas;
  TCanvas *c5 = new TCanvas;

  TFile *f = new TFile ("./output/dstars6.root","RECREATE");
  TTree *tree = new TTree("tree","arbre");

  float_t d0_P,d0_Ppos,d0_Pneg,d0_Ptpos,d0_Ptneg,d0_Pt,d0_m,d0_RunNbr,
  d0_EvtNbr,dStar_Pt,dStar_m,diffds_d0,drkpi,dphikpi,detakpi,dphid0ps,
  detad0ps,dphidsd0,detadsd0,drdsd0;
  float_t drd0ps,P_ps,trk_sumPt,trk_sumPt_2,costhetaK,costhetaD0;
  int type_signal;

  tree->Branch("d0_P",&d0_P,"d0_P/F");
  tree->Branch("d0_Ppos",&d0_Ppos,"d0_Ppos/F");
  tree->Branch("d0_Pneg",&d0_Pneg,"d0_Pneg/F");
  tree->Branch("d0_M",&d0_m,"d0_Masse/F");
  tree->Branch("d0_RunNbr",&d0_RunNbr,"d0_Run/F");
  tree->Branch("d0_EvtNbr",&d0_EvtNbr,"d0_Event/F");
  tree->Branch("d0_Ptpos",&d0_Ptpos,"d0_Ptpos/F");
  tree->Branch("d0_Ptneg",&d0_Ptneg,"d0_Ptneg/F");
  tree->Branch("d0_Pt",&d0_Pt,"d0_Pt/F");
  tree->Branch("dphikpi",&dphikpi,"dphikpi/F");
  tree->Branch("detakpi",&detakpi,"detakpi/F");
  tree->Branch("dphid0ps",&dphid0ps,"dphid0ps/F");
  tree->Branch("detad0ps",&detad0ps,"detad0ps/F");
  tree->Branch("dphidsd0",&dphidsd0,"dphidsd0/F");
  tree->Branch("detadsd0",&detadsd0,"detadsd0/F");
  tree->Branch("drdsd0",&drdsd0,"drdsd0/F");
  tree->Branch("type_signal",&type_signal,"type_signal/I");
  tree->Branch("trk_sumPt",&trk_sumPt,"trk_sumPt/F");
  tree->Branch("trk_sumPt_2",&trk_sumPt_2,"trk_sumPt_2/F");
  tree->Branch("costhetaK",&costhetaK,"costhetaK/F");
  tree->Branch("costhetaD0",&costhetaD0,"costhetaD0/F");
  tree->Branch("dStar_Pt",&dStar_Pt,"dStar_Pt/F");
  tree->Branch("dStar_M",&dStar_m,"dStar_Masse/F");
  tree->Branch("Diff_masse",&diffds_d0,"Difference");
  tree->Branch("drkpi",&drkpi,"Deltar_k_pi");
  tree->Branch("drd0ps",&drd0ps,"Deltar_d0_ps");
  tree->Branch("P_ps",&P_ps,"Impulsion_ps");

  for (Long64_t jentry=0; jentry<nentries;jentry++) {
    Long64_t ientry = this->LoadTree(jentry);

    if (ientry < 0) {
      cout<<"Nombre total d'evenememts: "<<jentry<<endl;
      break;
    }

    fChain->GetEntry(jentry);

    if(jentry%10000==0) {
      cout<<jentry<<" events processed"<<endl;

      c4->cd();
      h12->Draw();
      h12->GetXaxis()->SetTitle("Difference de masse signal+bruit (GeV)");
      h12->GetYaxis()->SetTitle("Nombre d'evenements");

      c5->cd();
      h13->Draw();
      h13->GetXaxis()->SetTitle("Difference de masse bruit seul (GeV)");
      h13->GetYaxis()->SetTitle("Nombre d'evenements");

      c4->Update();
      c5->Update();
    }

    std::deque<TLorentzVector> kplus;
    std::deque<TLorentzVector> kmoins;
    std::deque<TLorentzVector> pplus;
    std::deque<TLorentzVector> pmoins;

    trk_sumPt=0;
    trk_sumPt_2=0;
    for(int v=0 ; v<vx_n ; ++v) {
      if ((*vx_type)[v]==1) {

for(int t=0 ; t<(*vx_trk_n)[v] ; ++t) {
  const int i=(*vx_trk_index)[v][t];
  float trk_Pt=(*trk_pt)[i];
  trk_sumPt=trk_sumPt+trk_Pt;
  trk_sumPt_2=trk_sumPt_2+trk_Pt*trk_Pt;

  float tpm =(*trk_theta_wrtPV)[i];
  float tpm2=sin(tpm);
  float tpm3=(*trk_z0_wrtPV)[i];
  float z0sin = tpm3*tpm2;

  if ((*trk_nPixHits)[i]>=pixmin && fabs((*trk_d0_wrtPV)[i])<d0_max &&
  fabs(z0sin)<z0sin_max) {
    float tpn=(*trk_theta_wrtPV)[i]/2;
    float tpn2=tan(tpn);
    float_t eta=-log(tpn2);

    if ((*trk_qoverp_wrtPV)[i]>0) {
      pionp.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],pionm);
      kaonp.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],kaonm);
      kplus.push_back(kaonp);
      pplus.push_back(pionp);
    }

    if ((*trk_qoverp_wrtPV)[i]<0) {
      pionn.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],pionm);
      kaonn.SetPtEtaPhiM((*trk_pt)[i],eta,(*trk_phi_wrtPV)[i],kaonm);
      kmoins.push_back(kaonn);
      pmoins.push_back(pionn);
    }
  }
```

```
} //for (vx_trk_n)

for(unsigned int j=0;j<pplus.size();j++) {
  for(unsigned int k=0;k<pmoins.size();k++) {
    if(pplus[j].Pt()>ptmind0 && pmoins[k].Pt()>ptmind0) {
      vkp = pplus[j]+kmoins[k]; //TLV

      TVector3 tvkp=vkp.BoostVector();
      TLorentzVector kn1=kmoins[k];
      kn1.Boost(-tvkp);
      costhetaK=cos(kn1.Angle(tvkp));

      vpk = kplus[j]+pmoins[k];

      TVector3 tvpk=vpk.BoostVector();
      TLorentzVector kp1=kplus[j];
      kp1.Boost(-tvpk);

      costhetaK=cos(kp1.Angle(tvpk));

      if (fabs(vkp.M()-d0m)<200) {
for(unsigned int l=0;l<pplus.size();l++) {
  if (l!=j) {
    vDSplus = vkp+pplus[l];

    if (vDSplus.Pt()>ptminds) {

      d0_P=vkp.P();
      d0_Ppos=(pplus[j]).P();
      d0_Pneg=(kmoins[k]).P();
      d0_Ptpos=(pplus[j]).Pt();
      d0_Ptneg=(kmoins[k]).Pt();
      d0_m=vkp.M();
      d0_RunNbr=ei_RunNumber;
      d0_EvtNbr=ei_EventNumber;
      drkpi=kmoins[k].DeltaR(pplus[j]);
      dphikpi=pplus[j].DeltaPhi(kmoins[k]);
      detakpi = pplus[j].Eta() - kmoins[k].Eta();

      d0_Pt=vkp.Pt();

      drd0ps=vkp.DeltaR(pplus[l]);
      P_ps=pplus[l].P();
      dStar_Pt = vDSplus.Pt();
      dStar_m  = vDSplus.M();
      diffds_d0=(vDSplus.M()-vkp.M())/1000;
      dphid0ps=pplus[l].DeltaPhi(vkp);
      detad0ps=vDSplus.Eta() - pplus[l].Eta();

      drdsd0=vkp.DeltaR(vDSplus);
      dphidsd0=vkp.DeltaPhi(vDSplus);
      detadsd0=vDSplus.Eta() - vkp.Eta();

      TVector3 tvDSplus=vDSplus.BoostVector();
      TLorentzVector d0p=vkp;
      d0p.Boost(-tvDSplus);
      costhetaD0=cos(d0p.Angle(tvDSplus));

      if (fabs(vkp.M()-d0m)<20 && diffds_d0>0.135 && diffds_d0<0.170) {
h12->Fill((vDSplus.M()-vkp.M())/1000);
type_signal=1;
tree->Fill();
      } else if (fabs(vkp.M()-d0m)>50 && diffds_d0<0.170) {
h13->Fill((vDSplus.M()-vkp.M())/1000);
type_signal=0;
  tree->Fill();
      }
    }
  }
}
      }

      if(fabs(vpk.M()-d0m)<200) {
for(unsigned int n=0;n<pmoins.size();n++) {
  if (n!=k) {
    vDSmoins = vpk+pmoins[n];

    if (vDSmoins.Pt()>ptminds) {

      d0_P=vpk.P();
      d0_Ppos=(kplus[j]).P();
      d0_Pneg=(pmoins[k]).P();
      d0_Ptpos=(kplus[j]).Pt();
      d0_Ptneg=(pmoins[k]).Pt();
      d0_m=vpk.M();
      d0_RunNbr=ei_RunNumber;
      d0_EvtNbr=ei_EventNumber;
      drkpi=pmoins[k].DeltaR(kplus[j]);
      dphikpi=pmoins[k].DeltaPhi(kplus[j]);
      detakpi=pmoins[k].Eta()-kplus[j].Eta();

      d0_Pt=vpk.Pt();

      drd0ps=vpk.DeltaR(pmoins[n]);
      P_ps=pmoins[n].P();
      dStar_Pt = vDSmoins.Pt();
      dStar_m  = vDSmoins.M();
      diffds_d0=(vDSmoins.M()-vpk.M())/1000;
      dphid0ps=pmoins[n].DeltaPhi(vpk);
      detad0ps=vDSmoins.Eta()-pmoins[n].Eta();

      drdsd0=vpk.DeltaR(vDSmoins);
      dphidsd0=vpk.DeltaPhi(vDSmoins);
      detadsd0=vDSmoins.Eta() - vpk.Eta();

      TVector3 tvDSmoins=vDSmoins.BoostVector();
      TLorentzVector d0n=vpk;
      d0n.Boost(-tvDSmoins);
      costhetaD0=cos(d0n.Angle(tvDSmoins));


      if (fabs(vpk.M()-d0m)<20 && diffds_d0>0.135 &&  diffds_d0<0.170) {
h12->Fill((vDSmoins.M()-vpk.M())/1000);
type_signal=1;
tree->Fill(); // On remplit l'arbre.
      } else if (fabs(vpk.M()-d0m)>50 && diffds_d0<0.170) {
h13->Fill((vDSmoins.M()-vpk.M())/1000);
type_signal=0;
  tree->Fill();
      }
    }
  }
}
      }
    }
  }
}
      }
    }
  }
  f->Write();
  f->Close();
}

// this code has been copied from
https://twiki.cern.ch/twiki/bin/view/AtlasProtected/MinBiasD3PDDocu#CTP_Information
bool DStarTree::trigger_L1_MBTS_1()
{
  // The trigger decisions we are looking for
  bool trig_MBTS_1;
//   bool trig_MBTS_2;
//   bool trig_MBTS_1_1;
  // HARDCODE L1 bits corresponding to these triggers
  // THESE ARE THE 2009 NUMBERS
  int  L1_MBTS_1_bit   = 226;
//   int  L1_MBTS_2_bit   = 227;
//   int  L1_MBTS_1_1_bit = 228;

  // First find which BC the L1 accept come on
  unsigned int i = ctpRDO_lvl1aBC;

  // Set the offsets for this bunch crossing
  unsigned int trig_offset = 8*i; // 8 bits per BC
  // Loop over the bits. (There are 256 trigger bits.)
  unsigned int bit_pos = 0;
  unsigned int word_pos = 0;
  unsigned int bit_as_ulong = 1;

  // loop over all bits and fill the tavBits bitset
  bitset<256> tavBits;
  while(bit_pos < 256 && word_pos < 8) {
    if((*ctpRDO_tav)[word_pos+trig_offset] & bit_as_ulong) {
        tavBits[bit_pos] = true;
    }

    bit_pos++;
    if(bit_pos%32 == 0 && bit_pos != 0) {
      word_pos++;
      bit_as_ulong = 1;
    }
    else {
      bit_as_ulong <<= 1;
    }
  }// end loop over bits
  // make the decision on Trigger after veto
    trig_MBTS_1   = tavBits.test(L1_MBTS_1_bit);
//     trig_MBTS_2   = tavBits.test(L1_MBTS_2_bit);
//     trig_MBTS_1_1 = tavBits.test(L1_MBTS_1_1_bit);
    return trig_MBTS_1;
}
```

10.4 Programmes de traitement des données de simulations numériques Monte-Carlo

10.4.1 DStarRecTree.C

```
#define DStarRecTree_cxx
#include "DStarRecTree.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <iostream>
#include <TH1.h>
#include <TMath.h>
#include <TLorentzVector.h>
#include <set>
#include <vector>

DStarRecTree::DStarRecTree(TTree *tree) : MinBiasTree(tree) {}

void DStarRecTree::Loop()
{
  Int_t const  nbeventc=100000;
  Float_t const  kaonm=493.677;
  Float_t const  pionm=139.57018;
  Float_t pixmin=1;
  Float_t d0_max=2.0;  //en mm
  Float_t z0sin_max=2.0; //en mm

  TLorentzVector vecKstar;
  TLorentzVector TV2;

  TH1F *h12 = new TH1F("Total_diff_sigb","Total Difference masse
  sig+bruit",300,0.144,0.147);
  TCanvas *c1=new TCanvas;

  TFile *f = new TFile ("./output/mc_mother2.root","RECREATE");
  TTree *tree = new TTree("tree","arbre");
  int nbevent=0;

  //---------------------------------------------------

  if (fChain == 0) return;

  fChain->SetBranchStatus("*",0);
  fChain->SetBranchStatus("ei_EventNumber",1);
  fChain->SetBranchStatus("ei_RunNumber",1);
  fChain->SetBranchStatus("trk_n",1);
  fChain->SetBranchStatus("trk_pt",1);
  fChain->SetBranchStatus("trk_eta",1);
  fChain->SetBranchStatus("trk_phi",1);
  fChain->SetBranchStatus("trk_nPixHits",1);
  fChain->SetBranchStatus("trk_d0_wrtPV", 1);
  fChain->SetBranchStatus("trk_z0_wrtPV", 1);
  fChain->SetBranchStatus("trk_theta_wrtPV", 1);
  fChain->SetBranchStatus("vx_n",1);
  fChain->SetBranchStatus("vx_type",1);
  fChain->SetBranchStatus("vx_trk_n",1);
  fChain->SetBranchStatus("vx_trk_index",1);
  fChain->SetBranchStatus("trk_mc_index",1);
  fChain->SetBranchStatus("mc_gen_pt",1);
  fChain->SetBranchStatus("mc_gen_eta",1);
  fChain->SetBranchStatus("mc_gen_phi",1);
  fChain->SetBranchStatus("mc_gen_energy",1);
  fChain->SetBranchStatus("mc_begVtx_x",1);
  fChain->SetBranchStatus("mc_begVtx_y",1);
  fChain->SetBranchStatus("mc_begVtx_z",1);
  fChain->SetBranchStatus("mc_pdg",1);
  fChain->SetBranchStatus("mc_mother_pdg",1);
  fChain->SetBranchStatus("mc_n",1);
  fChain->SetBranchStatus("mcVx_n",1);

  TLorentzVector VecK;
  TLorentzVector VecPION1;
  TLorentzVector VecPION2;
  TLorentzVector VecD0;
  TLorentzVector VecDSTAR;

  float d0_P,d0_Pt,d0_m,d0_RunNbr,d0_EvtNbr,dStar_Pt,dStar_m,
  diffds_d0,drkpi,dphikpi,detakpi,dphid0ps,detad0ps,drd0ps,
  dphidsd0,detadsd0,drdsd0;
  float P_ps,trk_sumPt,trk_sumPt_2,part_sumPt,part_sumPt_2,
  costhetaK,costhetaD0, pp_Pt;
  int type_signal;

  tree->Branch("d0_P",&d0_P,"d0_P/F");
  tree->Branch("d0_M",&d0_m,"d0_Masse/F");
  tree->Branch("d0_RunNbr",&d0_RunNbr,"d0_Run/F");
  tree->Branch("d0_EvtNbr",&d0_EvtNbr,"d0_Event/F");
  tree->Branch("d0_Pt",&d0_Pt,"d0_Pt/F");
  tree->Branch("drkpi",&drkpi,"Deltar_k_pi");
  tree->Branch("dphikpi",&dphikpi,"dphikpi/F");
  tree->Branch("detakpi",&detakpi,"detakpi/F");
  tree->Branch("drd0ps",&drd0ps,"Deltar_d0_ps");
  tree->Branch("dphid0ps",&dphid0ps,"dphid0ps/F");
  tree->Branch("detad0ps",&detad0ps,"detad0ps/F");
  tree->Branch("drdsd0",&drdsd0,"drdsd0/F");
  tree->Branch("dphidsd0",&dphidsd0,"dphidsd0/F");
  tree->Branch("detadsd0",&detadsd0,"detadsd0/F");

  tree->Branch("type_signal",&type_signal,"type_signal/I");
  tree->Branch("trk_sumPt",&trk_sumPt,"trk_sumPt/F");
  tree->Branch("trk_sumPt_2",&trk_sumPt_2,"trk_sumPt_2/F");
  tree->Branch("part_sumPt",&part_sumPt,"part_sumPt/F");
  tree->Branch("part_sumPt_2",&part_sumPt_2,"part_sumPt_2/F");
  tree->Branch("costhetaK",&costhetaK,"costhetaK/F");
  tree->Branch("costhetaD0",&costhetaD0,"costhetaD0/F");
  tree->Branch("dStar_Pt",&dStar_Pt,"dStar_Pt/F");
  tree->Branch("dStar_M",&dStar_m,"dStar_Masse/F");
  tree->Branch("Diff_masse",&diffds_d0,"Difference");

  tree->Branch("P_ps",&P_ps,"Impulsion_ps");
  tree->Branch("pp_Pt",&pp_Pt,"Impulsion_transverse_pp/F");

  //---------------------------------------------------
  Long64_t nentries = fChain->GetEntriesFast();
  Long64_t nbytes = 0, nb = 0;
  Long64_t jentry=0;

  for (jentry=0; jentry<nentries; jentry++) {
    Long64_t ientry = LoadTree(jentry);
    if (ientry < 0) break;
    nb = fChain->GetEntry(jentry);   nbytes += nb;
    ++nbevent; //Compteur d'evenements
    if (!(nbevent%10000)) {
      cout<< "nombre d'evenements traites :"<<nbevent<<endl;
    }
    //if(nbevent==nbeventc){break;}

    trk_sumPt=0;
    trk_sumPt_2=0;
    part_sumPt=0;
    part_sumPt_2=0;

    for(int v=0 ; v<vx_n ; ++v) {
      if ((*vx_type)[v]==1) {

for(int t=0 ; t<(*vx_trk_n)[v] ; ++t) {
  const int w=(*vx_trk_index)[v][t];
  const int ww2=(*trk_mc_index)[w];

  if (w>=0 && w<trk_n){
  float part_Pt=(*trk_pt)[w];
  part_sumPt=part_sumPt+part_Pt;
  part_sumPt_2=part_sumPt_2+part_Pt*part_Pt;
  }

  if (ww2>=0 && ww2<mc_n){
    float trk_Pt=(*mc_gen_pt)[ww2];
    trk_sumPt=trk_sumPt+trk_Pt;
    trk_sumPt_2=trk_sumPt_2+trk_Pt*trk_Pt;
  }
}

Int_t w1=-1;
Int_t w2=-1;
Int_t w3=-1;

int nbre_part=0;
int nbre_part2=0;

for (int i=0 ; i<mc_n ; i++) {

  if ( (*mc_mother_pdg)[i]==421 || (*mc_mother_pdg)[i]==-421){

    nbre_part++;

    if ((*mc_pdg)[i]== 321 || (*mc_pdg)[i]== -321) {
      w1=i;
    } else if ((*mc_pdg)[i]== 211 || (*mc_pdg)[i]== -211) {
      w2=i;
    }
  } else if ( ((*mc_mother_pdg)[i]==413 || (*mc_mother_pdg)[i]==-413)){
```

```
    nbre_part2++;

    if ((*mc_pdg)[i]== 211 || (*mc_pdg)[i]== -211) {
      w3=i;
    }
  }
}

if (nbre_part==2 && w1>-1 && w2>-1){
  VecK.SetPtEtaPhiE( (*mc_gen_pt)[w1] , (*mc_gen_eta)[w1] , (*mc_gen_phi)[w1] ,
  (*mc_gen_energy)[w1] );
  VecPION1.SetPtEtaPhiE( (*mc_gen_pt)[w2] , (*mc_gen_eta)[w2] ,
  (*mc_gen_phi)[w2], (*mc_gen_energy)[w2] );
  VecD0=VecK+VecPION1;

  if (VecD0.M()>1855 && VecD0.M()<1875) {


    if (nbre_part2==1 && w3>-1){
      VecPION2.SetPtEtaPhiE( (*mc_gen_pt)[w3] , (*mc_gen_eta)[w3] ,
      (*mc_gen_phi)[w3], (*mc_gen_energy)[w3] );
      VecDSTAR=VecD0+VecPION2;

      if (VecDSTAR.M()>2005 && VecDSTAR.M()<2015) {
pp_Pt=VecPION1.Pt();

double mD0=VecD0.M();
double mDSTAR=VecDSTAR.M();


d0_P=VecD0.P();
//std::cout << "d0_P=" << d0_P << std::endl;
d0_Pt=VecD0.Pt();
d0_m=VecD0.M();
d0_RunNbr=ei_RunNumber;
d0_EvtNbr=ei_EventNumber;

drkpi=VecK.DeltaR(VecPION1);
dphikpi=VecPION1.DeltaPhi(VecK);
detakpi = VecPION1.Eta() - VecK.Eta();

drd0ps=VecD0.DeltaR(VecPION2);
dphid0ps=VecPION2.DeltaPhi(VecD0);
detad0ps=VecD0.Eta() - VecPION2.Eta();

P_ps=VecPION2.P();
dStar_Pt = VecDSTAR.Pt();
dStar_m  = VecDSTAR.M();
diffds_d0=(VecDSTAR.M()-VecD0.M());

drdsd0=VecD0.DeltaR(VecDSTAR);
dphidsd0=VecD0.DeltaPhi(VecDSTAR);
detadsd0=VecDSTAR.Eta() - VecD0.Eta();

TVector3 tvDS=VecDSTAR.BoostVector();
TLorentzVector d0p=VecD0;
d0p.Boost(-tvDS);
costhetaD0=cos(d0p.Angle(tvDS));

TVector3 tVecD0=VecD0.BoostVector();
TLorentzVector kp1=VecK;
kp1.Boost(-tVecD0);
costhetaK=cos(kp1.Angle(tVecD0));

type_signal=1;
tree->Fill();

h12->Fill((mDSTAR-mD0)/1000);//Remplissage histo sig+b

Int_t ww1=-1;
Int_t ww2=-1;
Int_t ww3=-1;

for(int tr=0 ; tr<trk_n ; ++tr) {
  const int mci=(*trk_mc_index)[tr];

  if (mci==w1) ww1=tr;
  else if (mci==w2) ww2=tr;
  else if (mci==w3) ww3=tr;
}

if (
    ((*trk_z0_wrtPV)[ww1] * sin((*trk_theta_wrtPV)[ww1]))<z0sin_max &&
    ((*trk_z0_wrtPV)[ww2] * sin((*trk_theta_wrtPV)[ww2]))<z0sin_max &&
    ((*trk_z0_wrtPV)[ww3] * sin((*trk_theta_wrtPV)[ww3]))<z0sin_max &&
    ww1>0 && (*trk_nPixHits)[ww1]>=pixmin && fabs((*trk_d0_wrtPV)[ww1])<d0_max &&
    ww2>0 && (*trk_nPixHits)[ww2]>=pixmin && fabs((*trk_d0_wrtPV)[ww2])<d0_max &&
    ww3>0 && (*trk_nPixHits)[ww3]>=pixmin && fabs((*trk_d0_wrtPV)[ww3])<d0_max
    ) {
VecK.SetPtEtaPhiM( (*trk_pt)[ww1] , (*trk_eta)[ww1] , (*trk_phi)[ww1] , kaonm);
VecPION1.SetPtEtaPhiM( (*trk_pt)[ww2] , (*trk_eta)[ww2] , (*trk_phi)[ww2], pionm);
VecPION2.SetPtEtaPhiM( (*trk_pt)[ww3] , (*trk_eta)[ww3] , (*trk_phi)[ww3], pionm);

VecD0=VecK +VecPION1;
VecDSTAR=VecD0 +VecPION2;

d0_P=VecD0.P();
d0_Pt=VecD0.Pt();
d0_m=VecD0.M();

drkpi=VecK.DeltaR(VecPION1);
dphikpi=VecPION1.DeltaPhi(VecK);
detakpi = VecPION1.Eta() - VecK.Eta();

drdsd0=VecD0.DeltaR(VecDSTAR);
dphidsd0=VecD0.DeltaPhi(VecDSTAR);
detadsd0=VecDSTAR.Eta() - VecD0.Eta();

drd0ps=VecD0.DeltaR(VecPION2);
dphid0ps=VecPION2.DeltaPhi(VecD0);
detad0ps=VecD0.Eta() - VecPION2.Eta();

P_ps=VecPION2.P();
dStar_Pt = VecDSTAR.Pt(); // Impulsion transverse du DStar
dStar_m  = VecDSTAR.M();
diffds_d0=(VecDSTAR.M()-VecD0.M())/1000;

TVector3 tvDS=VecDSTAR.BoostVector();
TLorentzVector d0p=VecD0;
d0p.Boost(-tvDS);
costhetaD0=cos(d0p.Angle(tvDS));

TVector3 tVecD0=VecD0.BoostVector();
TLorentzVector kp1=VecK;
kp1.Boost(-tVecD0);
costhetaK=cos(kp1.Angle(tVecD0));

type_signal=2;
tree->Fill();
}
      }
    }
  }
}
      }
    }
  }
  f->Write();
  f->Close();
}
```

10.5 Programmes de calculs d'efficacité et de pureté du signal

10.5.1 DStarPlot.C

```
DStarPlots()
{
  TCanvas *c1=new TCanvas;
  gPad->SetFillStyle(4000);
  c1->Divide(2,2);
  c1->cd(1);
  h1->Draw();
  c1->cd(2);
  h2->Draw();
  c1->cd(3);
  h3->Draw();// Signal + Bruit
  c1->cd(4);
  h4->Draw();// Bruit de fond

  TCanvas *c2=new TCanvas;
  Int_t minBin=h4->FindBin(180);
  Int_t maxBin=h4->FindBin(200);
  Double_t nh4=h4->Integral(minBin,maxBin);
  std::cout << "Number of events in h4=" << nh4 << std::endl;
  Double_t nh3=h3->Integral(minBin,maxBin);
  std::cout << "Number of events in h3=" << nh3 << std::endl;

  Int_t nbins=h4->GetNbinsX();
  Double_t lowBin=h4->GetBinLowEdge(1);
  Double_t highBin=h4->GetBinLowEdge(nbins+1);
  TH1F *h5=new TH1F("h5","Normalized background",nbins,lowBin,highBin);
  h5->Add(h4,nh3/nh4);
  h3->Draw();
  h5->Draw("same");

  TCanvas *c3=new TCanvas;
  TH1F *h6=new TH1F("h6","Signal-background",nbins,lowBin,highBin);// Histo du signal - bdf
  h6->SetOption("PE");
  h6->SetMarkerStyle(kFullCircle);
  h6->Add(h3);
  h6->Add(h4,-nh3/nh4);
  h6->Draw();

  minBin=h6->FindBin(145.4-1.5);
  maxBin=h6->FindBin(145.4+1.5);

  Double_t nSig=h6->Integral(minBin,maxBin);
  std::cout << "Number of events in peak (h6)= " << nSig << " +/- " // Nbre d'evenements signal seul
    << sqrt(nSig) << std::endl;
  Double_t nTot=h3->Integral(minBin,maxBin);
  std::cout << "Number of events in peak, incl. bkg (h3)= " << nTot << " +/- " //Nbre d'evenements signal+bruit
    << sqrt(nTot) << std::endl;
  Double_t nBkg=h5->Integral(minBin,maxBin);
  std::cout << "Number of events in scaled bkg (h5)=" << nBkg << " +/- " //Nbre d'evenements ds l'unite du bdf
    << sqrt(nBkg) << std::endl;
  std::cout << "Purity= (" << nSig/nTot*100 << " +/- "
    << nSig/nTot*sqrt((1-nSig/nTot)/nSig)*100 << ")%" << std::endl; //Purete obtenue
}
```

10.5.2 EffPur.C

```
EffPur()
{
  SetStyle();
  TCanvas *c3=new TCanvas;
  gPad->SetGridx(1);
  gPad->SetGridy(1);
  gStyle->SetOptStat(0);
  TH2 *hEPax=createEffPurAxes("Eff/Pur");
  hEPax->Draw();

  TLegend *pL=new TLegend(0.6,0.6,0.8,0.8);

  for(int cut=0 ; cut<2 ; ++cut) {
    std::string name;
    double step,min,max;
    double evts[100],purity[100];
    if (0==cut) makeDrdsd0(step,min,max,evts,purity,name);
    else if (1==cut) makeCosthetaK(step,min,max,evts,purity,name);

    const double start=min-step/2;
    const double end=max+step/2;
    const int nbBins=static_cast<int>((end-start)/step);

    TH1F *hEff=new TH1F("hEff",name.c_str(),nbBins,start,end);
    hEff->SetMarkerStyle(20);
    hEff->SetMinimum(0);
    TH1F *hPur=new TH1F("hPur","Purity",nbBins,start,end);
    hPur->SetMarkerStyle(28);

    double evtsTot[100];
    for(int b=0 ; b<nbBins ; ++b) {
      const double N=evts[0];
      const double n=evts[b];
      const double eff=n/N;
      const double p=purity[b];
      if (purity[b]>0) evtsTot[b]=evts[b]/purity[b];
      else evtsTot[b]=0;
      if (n>0) {
hEff->SetBinContent(b+1,eff);
hPur->SetBinContent(b+1,p);
      }
    }

    TCanvas *c1=new TCanvas;
    gPad->SetFillStyle(4000);
    gPad->SetGridx(1);
    gPad->SetGridy(1);
    hEff->Draw("p");
    hPur->Draw("same p");

    c3->cd();
    TGraphErrors *pEP=createEffPur(nbBins,evts,evtsTot);
    pEP->SetMarkerStyle(20+cut);
    pEP->Draw("same p");
    pL->AddEntry(pEP,name.c_str(),"P");
  }
  pL->Draw();
}

//   Costheta K
makeCosthetaK(double &step,double &min,double &max,double
*evts,double *purity,std::string &name)
{
  double step=0.1;
  double min=0.1;
  double max=0.5;
  const double s_evts[]={570,470,385,235,125};
  copydouble(5,s_evts,evts);
  const double s_purity[]={0.3455,0.3341,0.3402,0.3167,0.3280};
  copydouble(5,s_purity,purity);
  name="Costheta K";
}

//   Delta R dStar d0
makeDrdsd0(double &step,double &min,double &max,double
*evts,double *purity,std::string &name)
{
  double step=0.001;
  double min=0.003;
  double max=0.009;
  const double s_evts[]={780,695,575,380,225,155,75};
  copydouble(7,s_evts,evts);
  const double s_purity[]={0.6297,0.6138,0.5852,0.5064,0.4052,0.3563,0.2514};
  copydouble(7,s_purity,purity);
  name=" Delta R dStar d0";
}

copydouble(const int n,double *source,double *dest)
{
  for(int i=0 ; i<n ; ++i) {
    dest[i]=source[i];
  }
}

TH2 *createEffPurAxes(const std::string &name)
{
  std::string hName=name;
  hName+=";Purity;Efficiency";
  TH2I *pA=new TH2I("hEPax",hName.c_str(),10,0,1.05,10,0,1.05);
  return pA;
}

TGraphErrors *createEffPur(const int n,const double signal[],
const double total[],const double refSig=-1)
{
  TGraphErrors *pG=new TGraphErrors;
  pG->SetMarkerStyle(kOpenCross);
  if (n<1) return pG;

  double signal0=refSig;
  if (refSig<0) signal0=signal[0];
  if (signal0==0) {
    std::cout << "Error in createEffPur(..): reference signal is null !" << std::endl;
    return pG;
  }

  for(int i=0,p=0 ; i<n ; ++i) {
    const double nSig=signal[i];
    const double nTot=total[i];
    if (nSig>0 && nTot>0) {

      const double efficiency=nSig/signal0;
      const double effError=sqrt(efficiency*(1-efficiency)/signal0);

      const double purity=nSig/nTot;
      const double purError=purity*sqrt((1-purity)/nSig);

      pG->Set(p+1);
      pG->SetPoint(p,purity,efficiency);
      pG->SetPointError(p,purError,effError);
      ++p;
    }
  }
  return pG;
}

// the rest is taken from the ATLAS style wiki, and adjusted
that it actually compiles ...
TStyle* NewStyle()
{
  TStyle *newStyle = new TStyle("NEW","New style");

  // use plain black on white colors
  Int_t icol=0; // WHITE
  newStyle->SetFrameBorderMode(icol);
  newStyle->SetFrameFillColor(icol);
  newStyle->SetCanvasBorderMode(icol);
  newStyle->SetCanvasColor(icol);
  newStyle->SetPadBorderMode(icol);
  newStyle->SetPadColor(icol);
  newStyle->SetStatColor(icol);
  newStyle->SetPadTickX(1);
  newStyle->SetPadTickY(1);
  return newStyle;
}

void SetStyle ()
{

   TStyle* newStyle = NewStyle();
   gROOT->SetStyle("NEW");
   gROOT->ForceStyle();
}
}
```

10.6 Exemple de collision dans ATLAS à 7 TeV pour le RUN 152166

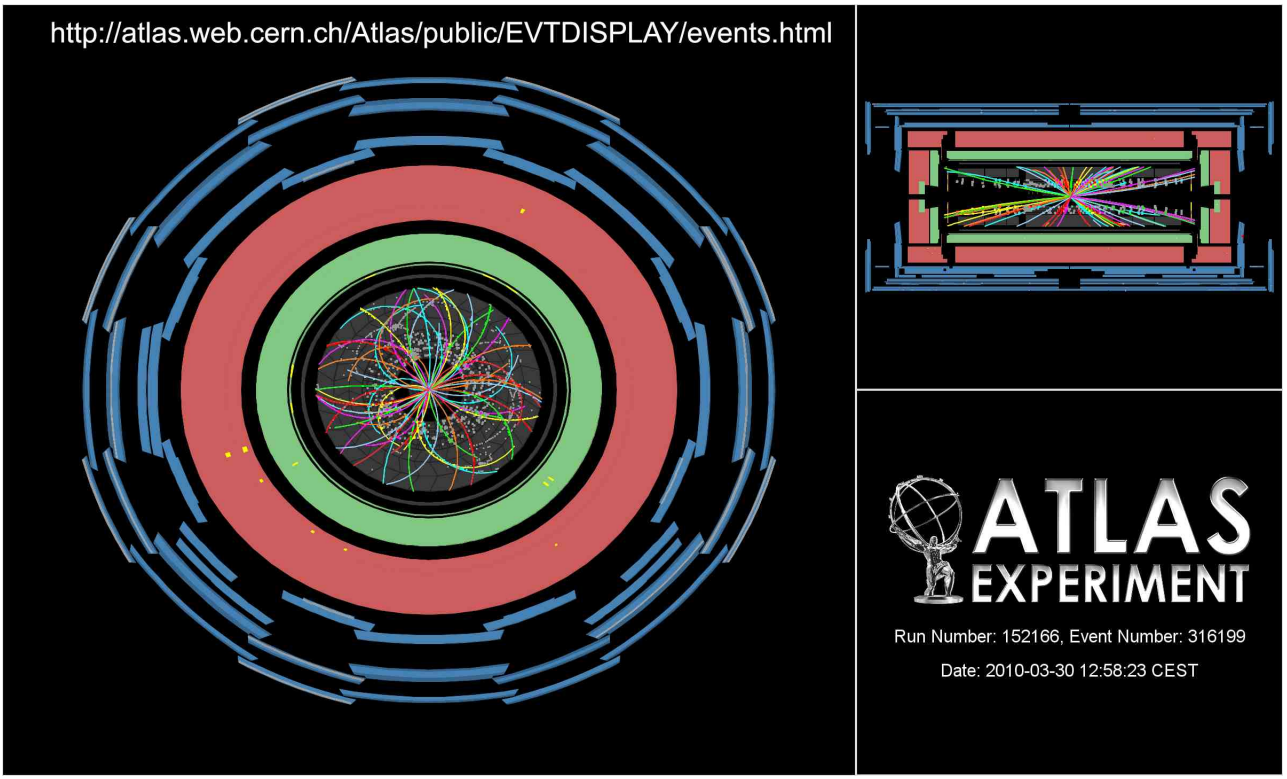

FIGURE 10.1 – Collision *pp* à 7 TeV,dans le centre de masse, dans le détecteur ATLAS du LHC, Run = 152166, Event = 316199

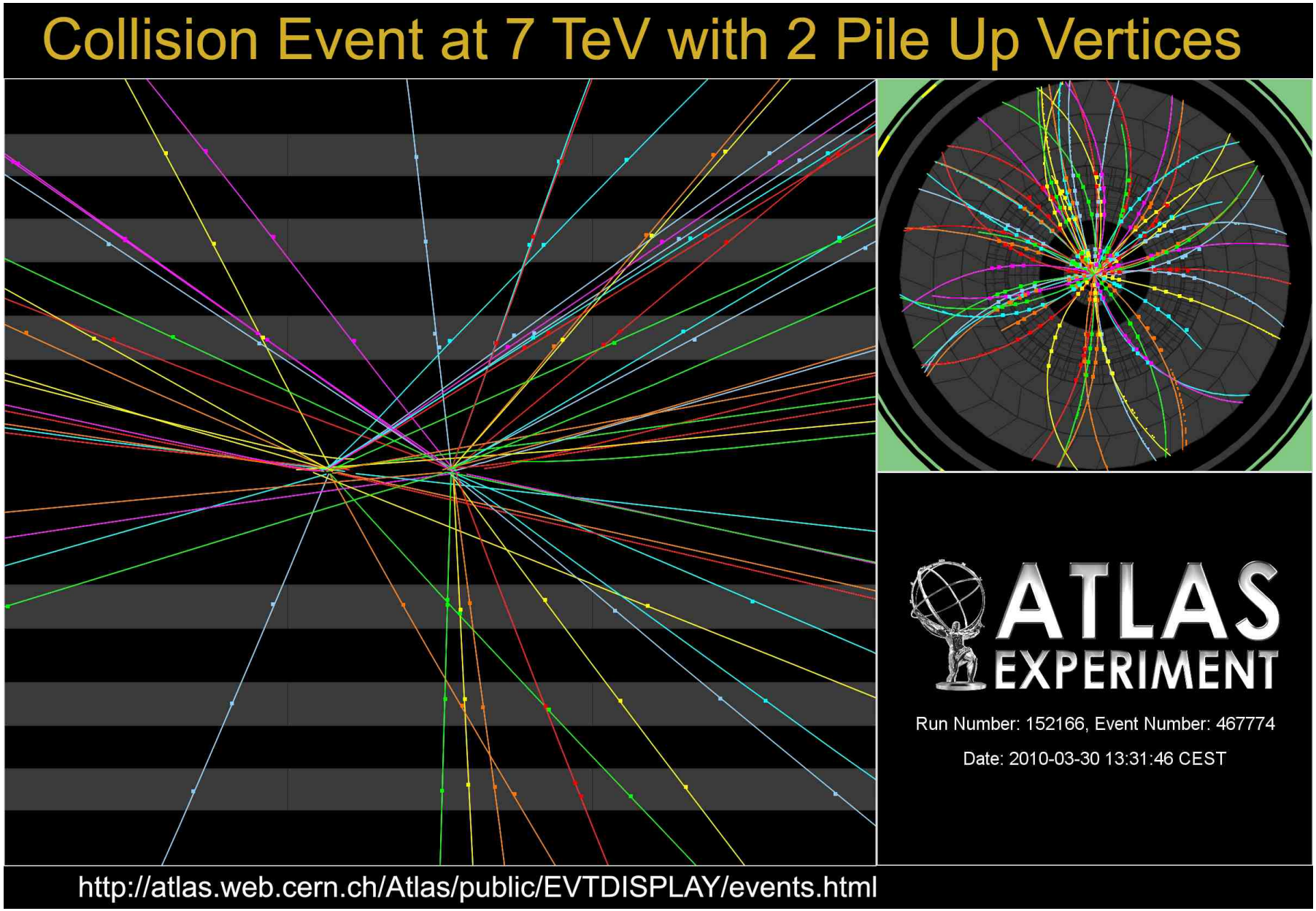

FIGURE 10.2 – Une autre vue de la collision pp à 7 TeV,dans le centre de masse, dans le détecteur ATLAS du LHC, Run = 152166, Event = 316199

Bibliographie

[1] CALVET David Mesure des durées de vie des mésons B^+ et B^0 par reconstruction exclusive avec le détecteur ALEPH. *Thèse*, 2004.

[2] DEFAY Pierre Olivier Prospectives de recherche du quark de quatrième génération avec le détecteur ATLAS auprès du LHC. *Thèse*, 2008.

[3] The ATLAS Collaboration $D^{(*)}$ mesons reconstruction in pp collisions at $\sqrt{s} = 7$ TeV. *ATLAS Note*. June 5, 2010.

[4] CALVET David Le modèle Standard. `http://voyage.in2p3.fr/` IN2P3.

[5] Particle Data Group Particle Physics booklet. July 2008.