TABLE DES MATIERES

Remerciements…….i

Table des matières…….iii

Liste des abréviations…….vi

Liste des figures……..vii

Liste des tableaux…….viii

INTRODUCTION…….1

PARTIE THEORIQUE

Chapitre I : LA VIOLATION CP 3

I-1 Généralités 3

I-2 La conjugaison de charge C et la parité P 4

I-3 Origine de la violation CP 4

Chapitre II : L'EXPÉRIENCE LHCb..... 7

II- 1 Généralités sur l'expérience LHCb 7

II-2 Le détecteur 9

II-2-1 Introduction 9

II-2-2 Tube à faisceau..... 10

II-2-3 Le VELO 10

II-2-4 L'aimant 11

II-2-5 Les RICHes 12

II-2-6 Le trajectographe..... 13

II-2-7 Les calorimètres 14

II-2-8 Le système à muon..... 14

II-3 Reconstruction de « trace » 14

II-3-1 Introduction 14

II-3-2 Différents types de trace 15

II-4 Indentification des particules 16

Chapitre III : LA SIMULATION 18

III-1 Introduction 18

III-2 Généralités sur Gauss 18

III-3 Structure de Gauss 19

III-3-1 Génération d'événements 20

III-3-2 Simulation du détecteur 21

III-3-3 Geant4 22

III-4 Contrôle de la simulation 24

PARTIE PRATIQUE

Chapitre I : CONFIGURATION DU TRAVAIL EN GAUSS 27

I-1 Introduction 27

I-2 Configuration et exécution d'un travail standard en Gauss 28

I-2-1 Structure du logiciel 28

I-2-2 Plateforme 28

I-2-3 Installation de l'environnement 28

I-2-4 Fichier d'options et contrôle de Gauss 29

I-2-5 Fonctionnement du travail 31

I-2-6 Fonctionnement de Gauss sur Linux 32

I-3 Configuration de notre travail de contrôle 32

I-3-1 Le code « MonitorTrackAction » 32

I-3-2 Création de la librairie 33

I-3-3 Options de travail 33

I-3-4 Introduction des coupures 35

Chapitre II : RESULTATS ET INTERPRETATION 36

II-1 Introduction 36

II-2 Valeur des coupures 36

II-3 Résultats généraux 38

II-3-1 Résultats avec LXBATCH 38

II-3-2 Résultats avec la même machine 40

II-4 Comparaison des histogrammes 41

CONCLUSION..... 45

BIBLIOGRAPHIE

LISTE DES ABREVIATIONS

CERN : Organisation Européenne pour la Recherche Nucléaire

LHC : Large Hadron Collider

LHCb : Large Hadron Collider beauty

LEP : Large Electron-Positon Collider

KEK : High Energy Accelerator Research Organisation

SLAC: Stanford Linear Accelerator Center

RICH : Ring Imaging Cherenkov

VELO : Vertex Locator

HPDs : Hybrid Photon Detectors

ECAL: Electromagnetic Calorimeter

HCAL: Hadronique Calorimeter

CPU : Central Processing Unit

GiGa : Geant Interface for Gaudi Application ou Gaudi Interface to Geant4 Application

LISTE DES FIGURES

Figure 1 : Triangle d'unitarité représentant l'équation
$V_{td}V_{ud}^{*}+V_{ts}V_{us}^{*}+V_{tb}V_{ub}^{*}=0$dans le plan complexe (ρ,η)……6

Figure 2 : Distribution angulaire des paires $b\bar{b}$ produites au LHC…….8

Figure 3 : Emplacement du détecteur LHCb dans la caverne au point 8……9

Figure 4 : Disposition du détecteur LHCb…….10

Figure 5 : Un détecteur prototype au silicium du LHCb……11

Figure 6 : Disposition verticale du détecteur RICH1…….13

Figure 7 : Schéma des traces reconstruites et les trajectoires y attribuées……17

Figure 8 : Structure de Gauss……..21

Figure 9 : Exemple d'équivalence portée-énergie……..25

Figure 10 : Effet de coupure d'énergie dans le calorimètre électromagnétique
avec un négaton de 30GeV……26

Figure 11 : Histogramme du travail par défaut……40

Figure 12 : Histogrammes correspondant à « Production cut »et
à « Tracking cut »…….41

Figure 13 : Rapport des histogrammes……42

LISTE DES TABLEAUX

Tableau 1 : Les valeurs des coupures pour chaque option de travail…….35

Tableau 2 : Résultats avec LXBATCH……36

Tableaux 3 : Résultats avec la même machine……38

INTRODUCTION

La physique des particules connue sous le nom physique des hautes énergies est devenue actuellement un des domaines les plus avancés dans le monde de la physique. Le CERN est un laboratoire reconnu à l'échelle internationale qui se concentre à réaliser des études dans ce domaine. Le LHC (*Large Hadron Collider*) est le nouveau collisionneur qui y est construit pour mener des expériences. Seulement il n'est pas encore opérationnel et à l'heure actuelle, les expériences utilisent les données produites par la simulation.

Le programme de recherche du CERN s'élargit en plusieurs expériences pour étudier tout aspect de la physique des particules et LHCb figure parmi eux. Ce dernier est dédié à l'étude de la violation CP et des autres phénomènes rares dans la désintégration des mésons B avec une grande précision.

Toute simulation est bâtie sur des considérations théoriques permettant, toutefois, de prévoir le comportement du détecteur. À part le faisceau test, la simulation se présente comme le moyen le plus efficace pour mener à bien la réalisation d'une expérience. Dans l'expérience LHCb, Gauss est un programme de simulation Monte-Carlo qui permet de générer des événements et de simuler le comportement du détecteur.

On remarque néanmoins que par rapport aux autres applications qui fonctionnent dans l'expérience, Gauss est très lent du fait que Geant4, le logiciel qu'il utilise pour reproduire ce qui se passe dans le détecteur, prend beaucoup de temps pour tracer les particules dans le détecteur. Le temps mis par Gauss pour générer un seul événement s'élève jusqu'à plusieurs minutes alors que la meilleure statistique exige des milliers, voire des millions d'événements. D'où vient l'idée que le temps de traitement devrait être sous contrôle et si possible amélioré sans nuire aux résultats.

Ainsi, le but de notre travail est de déterminer s'il y a des moyens pour pouvoir diminuer cette valeur de temps consommé par Geant4 lorsqu'il trace les particules dans le détecteur sans compromettre les résultats physiques de la simulation. Un moniteur est mis en place pour contrôler la performance de Gauss en termes du temps CPU consommé. Le temps CPU étant la valeur de temps avec laquelle l'unité centrale de traitement exécute réellement une instruction. Nous allons utiliser une variété de coupures afin de trouver celle qui sont jugées raisonnablement appropriées.

Notre travail est subdivisé en deux grandes parties : la partie théorique et la partie pratique. Dans la première partie, la violation CP sera développée. Puis nous allons voir la description de l'expérience LHCb suivie de la structure de Gauss et les aspects théoriques de la simulation.

Dans la deuxième partie, la configuration d'un travail standard en Gauss sera donnée. Après nous allons voir la modification que nous avons apportée lors de la configuration de notre travail de contrôle. Et finalement nous allons donner les résultats obtenus avec le système de *cluster* de PC du CERN et ceux avec la même machine. L'interprétation des ces résultats sera accompagnée d'une comparaison des histogrammes.

PARTIE THEORIQUE

Chapitre I : LA VIOLATION CP

I-1 Généralité[1]

Depuis des années, les physiciens admettaient que l'invariance des lois de la physique par la combinaison CP était une évidence. Ce n'était qu'en 1964 qu'un groupe de chercheurs a observé pour la première fois la violation de la symétrie CP dans la désintégration du kaon neutre $K°$. Son origine reste encore un des mystères pour la physique des particules. Si on trouve dans les interactions faibles une violation de C et de P séparément, la combinaison CP préserve la symétrie dans la plupart des processus d'interaction faible. Cependant, la symétrie CP est violée dans certains processus comme la désintégration du $K°$, et des mésons B.

Le Modèle Standard est une théorie qui décrit les interactions fortes, faibles et électromagnétiques, ainsi que l'ensemble des particules élémentaires. Ce modèle, avec trois familles de quark peut naturellement générer une violation CP, aussi bien dans l'interaction faible que dans l'interaction forte (jamais détectée). La violation CP dans l'interaction faible est décrite par la matrice 3x3 complexe et unitaire connue sous le nom matrice CKM (Cabibbo-Kobayashi-Maskawa) introduite par Kobayashi et Maskawa. Les phénomènes de violation CP observés dans le système des kaons neutres sont en accord avec ce mécanisme. Pourtant, cela ne veut pas dire que la physique au-delà du Modèle Standard ne contribue ni prend part pour les phénomènes observés.

La violation CP joue aussi un rôle très important en cosmologie. Ceci est un des trois éléments nécessaires pour expliquer l'excès de matière vis-à-vis d'antimatière dans notre univers. Au niveau de la violation CP pouvant être générée par le Modèle Standard, l'interaction faible est insuffisante pour expliquer la dominance de l'antimatière dans l'univers. Ceci introduit une nouvelle source de violation CP au delà du Modèle Standard.

La violation CP a été récemment détectée par les expériences Belle (au KEK, le *High Energy Accelerator Research.Organisation* au Japon) et BaBar (au SLAC, le *Stanford Linear Accelerator Center*, aux Etats-Unis d'Amérique) dans les désintégrations du méson $B°$. Dans le système de méson- B, il existe beaucoup plus de modes de désintégration disponibles, et le Modèle Standard permet la prédiction

de violation CP dans nombreux d'entre eux. Le système des mésons avec un quark b est, pourtant, un domaine intéressant pour étudier la violation CP.

I-2 La conjugaison de charge C et la parité P [2]

La conjugaison de charge C est une opération qui transforme une particule en son antiparticule ou l'inverse, en laissant inchangés sa masse, son spin et son impulsion. Elle agit sur tous les nombres quantiques additionnels en changeant leur signe : charge électrique, nombre baryonique, nombres leptoniques, étrangeté…

Dans l'espace euclidien la parité P est tout simplement une opération qui change le signe de toutes les coordonnées de chaque point d'espace : c'est la symétrie de l'espace par rapport à l'origine O des axes. En d'autres termes, la transformation de parité est équivalente à une symétrie par rapport au plan (O,x,y) suivie d'une rotation d'angle π autour de l'axe Oz.

Dans l'espace de Hilbert, cette opération P de l'espace euclidien est associée à un opérateur $\bar{P}$ qui permet d'obtenir le vecteur d'état dans lequel se trouve un système après l'application de l'opération P.

I-3 Origine de la violation CP [2][18]

On peut interpréter théoriquement le phénomène de violation CP à l'aide de la matrice 3x3 unitaire V_{CKM} appelé « matrice de Cabibbo-Kobayashi-Maskawa ». Cette matrice apparaît dans le lagrangien qui décrit l'interaction faible de courant chargé entre les quarks selon :

$$-L_{W^{\pm}} = \frac{g}{\sqrt{2}} \overline{u_{Li}}\, \gamma^{\mu}\, (V_{CKM})_{ij}\; d_{Lj} W_{\mu}^{+} + h.c \qquad (1.1)$$

Si on met les quarks en ordre suivant leur masse, c'est-à-dire (u_1, u_2, u_3) ⟶ (u, c, t) et (d_1, d_2, d_3) ⟶ (d, s, b) les éléments de V_{CKM} sont écrits comme suit :

$$V_{CKM} = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix} \quad (1.2)$$

Il existe plusieurs façons de paramétriser la matrice CKM. Dans ce devoir, nous allons prendre un seul exemple qui est le plus utilisé.

Dans cette paramétrisation, V_{CKM} est obtenue en faisant le produit de trois matrices de rotation, où les rotations sont caractérisées par les angles d'Euler θ_{12}, θ_{13} et θ_{23}, lesquels sont les angles de mélange des saveurs et une phase δ.

Alors on a :

$$V_{CKM} = \begin{pmatrix} c_{12}c_{13} & s_{12}c_{13} & s_{13}e^{-i\delta} \\ -s_{12}c_{23} - c_{12}s_{23}s_{13}e^{i\delta} & c_{12}c_{23} - s_{12}s_{23}s_{13}e^{i\delta} & s_{23}c_{13} \\ s_{12}s_{23} - c_{12}c_{23}s_{13}e^{i\delta} & -c_{12}s_{23} - s_{12}c_{23}s_{13}e^{i\delta} & c_{23}c_{13} \end{pmatrix} \quad (1.3)$$

Où $c_{ij} = \cos\theta_{ij}$ et $s_{ij} = \sin\theta_{ij}$. Ces angles décrivent le mélange et δ est la phase qui est responsable pour tous les phénomènes de violation CP dans les processus d'échange de courant de charge du Modèle Standard. La matrice obtenue est aussi unitaire.

Pour alléger cette matrice, Wolfenstein a suggéré le changement de variables suivant :

$s_{12} \equiv \lambda$,

$s_{23} \equiv A\lambda^{2,}$

$s_{13}e^{-i\delta} \equiv A\lambda^3(\rho - i\eta)$,

tout en conservant l'unitarité de la matrice.

Les éléments de la nouvelle matrice sont alors obtenus en faisant des développements en série de Taylor autour de λ. A l'ordre 4, on a :

$$V_{CKM} = \begin{pmatrix} 1 - \frac{\lambda^2}{2} & \lambda & A\lambda^3(\rho - i\eta) \\ -\lambda & 1 - \frac{\lambda^2}{2} & A\lambda^2 \\ A\lambda^3(1 - \rho - i\eta) & -A\lambda^2 & 1 \end{pmatrix} + O(\lambda^4) \quad (1.4)$$

La violation CP est décrite dans cette paramétrisation par la composante imaginaire iη.

L'unitarité de la matrice V_{CKM} (c'est-à-dire $V_{CMK}V_{CKM}\dagger=V_{CKM}\dagger\ V_{CMK}=1$) impose des restrictions sur les éléments de matrice. Parmi les neuf relations unitaires existantes, considérons les 3 équations d'orthogonalité indépendantes :

$$V_{ud}V_{cd}^{*}+V_{us}V_{cs}^{*}+V_{ub}V_{cb}^{*}=0$$
$$V_{td}V_{cd}^{*}+V_{ts}V_{cs}^{*}+V_{tb}V_{cb}^{*}=0$$
$$V_{td}V_{ud}^{*}+V_{ts}V_{us}^{*}+V_{tb}V_{ub}^{*}=0$$

Ces équations peuvent se représenter graphiquement par des triangles dans le plan complexe (ρ , η) en utilisant la paramétrisation de Wolfenstein. La troisième relation est la plus intéressante car le triangle qu'elle décrit possède 3 côtés avec des longueurs du même ordre ($\sim\lambda^3$) alors que les deux autres triangles sont relativement plats.

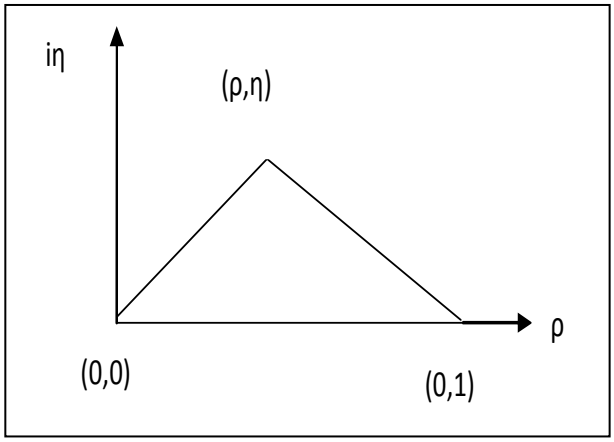


Figure 1 : Triangle d'unitarité représentant l'équation
$V_{td}V_{ud}^{*}+V_{ts}V_{us}^{*}+V_{tb}V_{ub}^{*}=0$ *dans le plan complexe* (ρ,η)

Heureusement, des expériences sont dédiées à l'étude de ce phénomène et nous allons décrire dans la section suivante une de ces expériences, LHCb, qui a comme objectif de faire des mesures de très haute précision sur ce sujet.

Chapitre II : L'EXPÉRIENCE LHCb

II- 1 Généralités sur l'expérience LHCb [4]

L'expérience LHCb s'inscrit dans le cadre du programme de recherche du CERN. Elle se traduit par la réalisation d'un détecteur qui va pouvoir bénéficier du faisceau de particules produites par le LHC. Il s'agit de l'un des quatre détecteurs annexés au collisionneur LHC.

Le but principal de LHCb est d'étudier l'origine de la violation CP, phénomène qui n'est pas complètement prédit par le Modèle Standard. Il permettra également l'étude des désintégrations rares. Il ouvre donc des perspectives de recherche au-delà du Modèle Standard.

Le LHCb sera mis en fonction en même temps que le collisionneur LHC et ses autres détecteurs. Les protons accélérés vont atteindre une énergie de 7 TeV, ce qui correspond à une énergie dans le centre de masse de 14 TeV.

La production des hadrons contenant un quark b ou $\bar{b}$ est prédominante dans un cône centré sur l'axe de collision. Les simulations données par le générateur Pythia [5][6] indiquent que l'angle moyen des particules b avec l'axe incident est concentré dans environ 400 mrd, et les deux quark b et $\bar{b}$ sont produits dans le même cône, conformément à la figure n° 2. C'est pour cette raison qu'on peut avoir un seul cône.

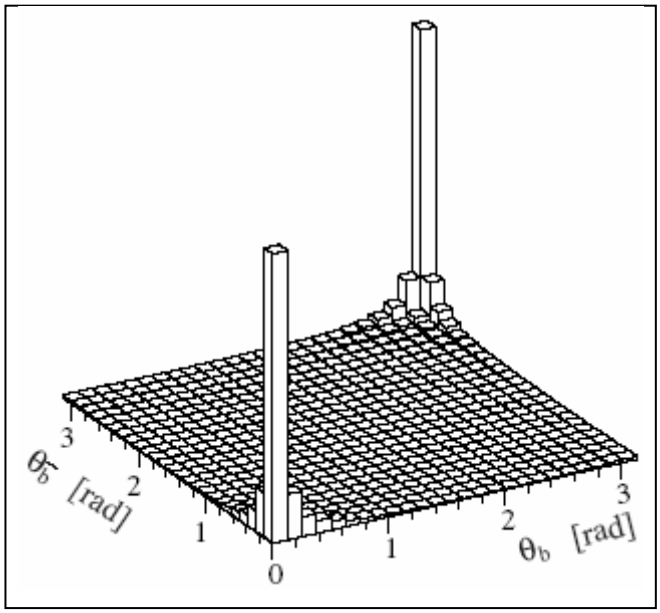


Figure 2 : Distribution angulaire des paires b$\bar{b}$ produites au LHC.

Ainsi, pour des arguments physiques, le détecteur est placé à l'avant de la collision. Un autre détecteur aurait pu être placé en arrière ce qui aurait permis d'augmenter le nombre d'évènements. Mais ce dispositif supplémentaire aurait fait doubler le coût total de l'expérience et surtout on aurait dû construire une caverne dédiée et pourtant on n'aurait pas gagné beaucoup dans les mesures.

Le détecteur LHCb est installé à 100m sous terre au Point 8 (figure 3) de l'anneau de collision. Cette place était auparavant occupée par l'expérience DELPHI, qui était en œuvre dans le précédant collisionneur LEP (*Large Electron-Positon Collider*). Le point de croisement des faisceaux a été déplacé du centre de la caverne vers l'entrée de la caverne afin de s'adapter à la configuration du détecteur.

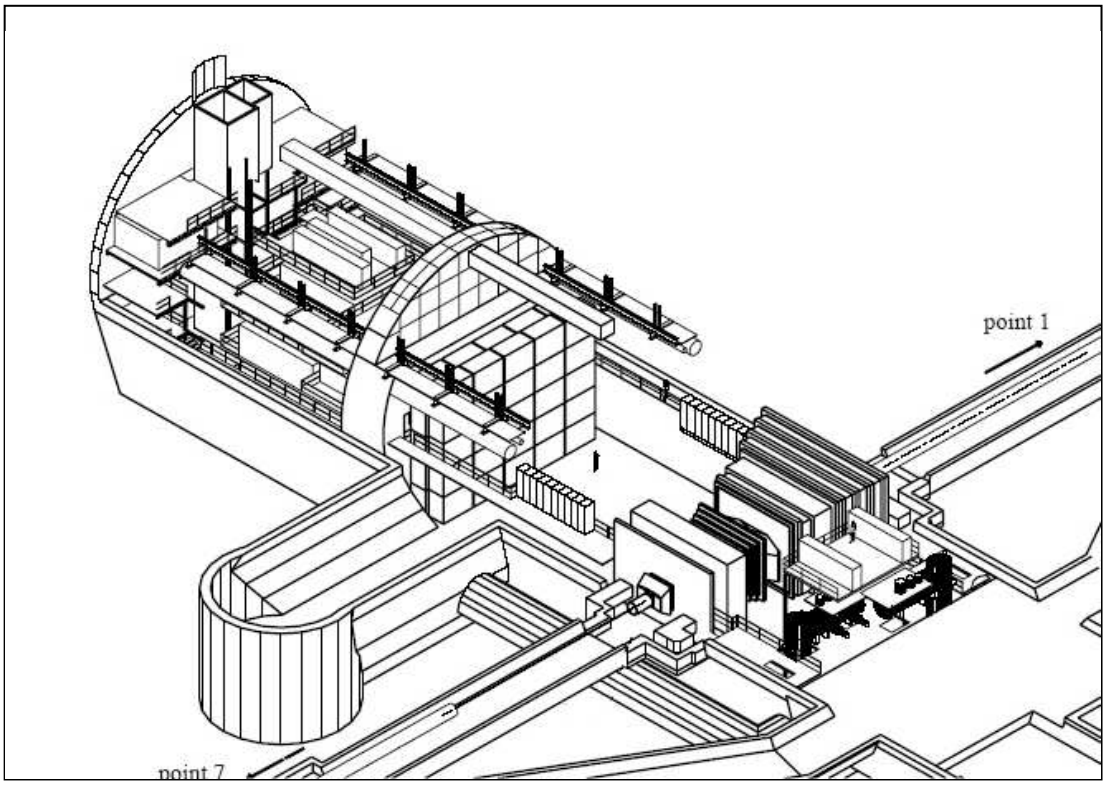


Figure 3 : Emplacement du détecteur LHCb dans la caverne au point 8

II-2 Le détecteur [4][7]

II-2-1 Introduction

Le détecteur est actuellement dans la phase finale d'installation. La première collecte des données est prévue pour l'été 2008. Le détecteur prend les 20m de longueur de la caverne. Afin de réaliser l'identification des mésons B et de voir les désintégrations rares, il est composé de sept éléments principaux :

- Le tube à faisceau
- Le Vertex Locator (VELO)
- L'aimant
- Les détecteurs Ring Imaging Cherenkov (RICHes)
- Les trajectographes
- Les calorimètres
- Le système à muon

La constitution de détecteur LHCb est montrée dans la figure n°4 où le système de référence est indiqué, avec z suivant la ligne de faisceau, du VELO au système à muon, y vers le haut et x (pas indiqué) qui rentre dans la page. Nous allons voir ces éléments principaux dans la section suivante.

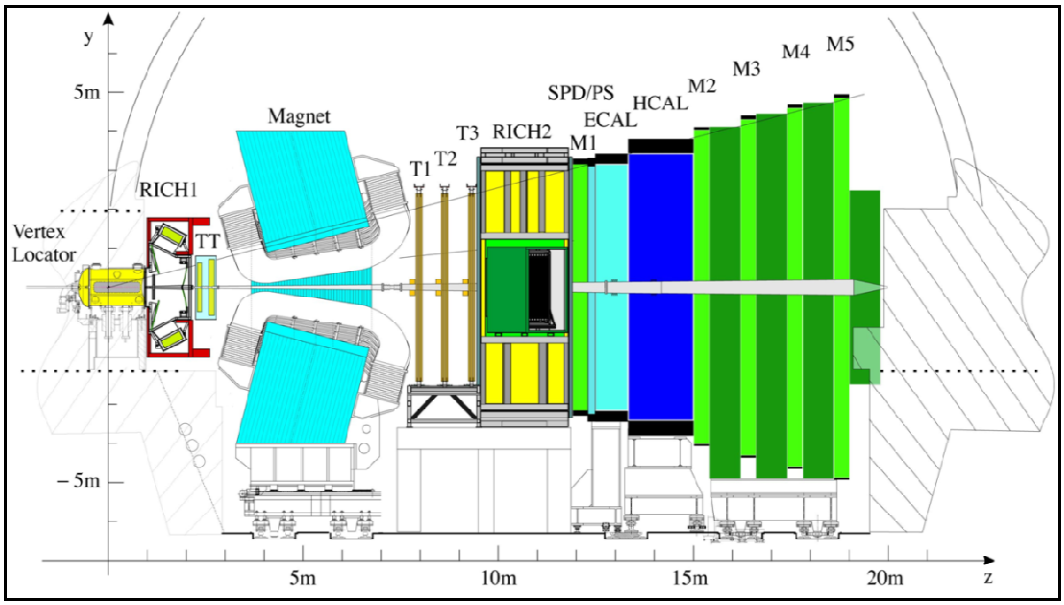


Figure 4 : Disposition du détecteur LHCb

II-2-2 Tube à faisceau

La conception du tube à faisceau devient particulièrement délicate depuis que la chambre à vide est placée dans la région à rapidité élevée du détecteur LHCb où la densité de particule est élevée et le nombre de particules secondaires dans l'événement dépend des matériels vus par les particules primaires incidentes. La masse relative du tube à faisceau comme étant une source de particule secondaire a été considérablement réduite. En plus, pour réduire la quantité de matériel traversé par les particules; le béryllium a été choisi pour presque 12 m.

Le tube à faisceau est constitué par une fenêtre de sortie mince scellée avec le réservoir à vide du VELO. Cette partie est suivie de deux sections coniques, la première est presque 1,5 m de long avec un angle d'ouverture 25 mrd, et la seconde est presque16 m de long avec un angle d'ouverture 10 mrd. Le système chambre à vide est conçu pour une pression totale moyenne de 10^{-8} à 10^{-9} mbar à l'intérieur du tube à faisceau.

Il est divisé en quatre sections consécutives : UX85/1 ; UX85/2 ; UX85/3 ; UX85/4 ; lieés entre elles par des systèmes des soufflets.

II-2-3 Le VELO

Le VELO ou « Vertex Locator » est formé par un ensemble de petits détecteurs (modules) au silicium placés perpendiculairement à la direction z. Chaque couche de silicium (*sensor*, deux par module) a une épaisseur de 220μm et un diamètre de 8,4 cm. Elles fournissent les coordonnés des points proches de la région d'interaction. Dans la figure 5 d'un des modules prototype au silicium de LHCb, deux régions particulières sont mises en évidence :

- Les « *strips* » : au nombre de 2048 dans un *sensor* ; ils sont répartis suivant la circonférence du *sensor* (figure 5) appelés « *r-strips* ». Mais il existe aussi un autre type de *sensor* où les *strips* appelés « *Φ-strips* » sont perpendiculaires à la circonférence du *sensor*.
- Les lignes de conduction ou *« routing lines »* : disposées radialement dans le détecteur

(c'est-à-dire perpendiculaires aux « strips »), elles ont pour rôle de conduire les signaux reçus par les « strips » lorsqu'une particule traverse le détecteur au silicium.

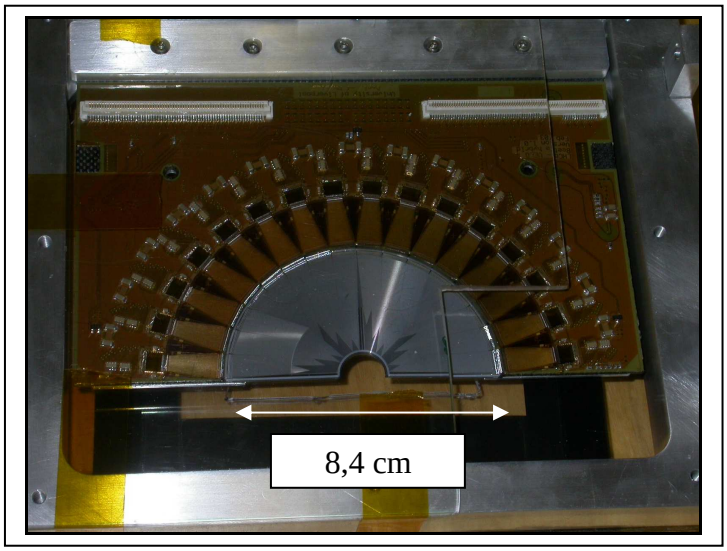


Figure 5 : Un détecteur prototype au silicium du LHCb

II-2-4 L'aimant

L'aimant est placé entre le VELO et le trajectographe, cela permet de garder sa taille petite.

Le champ magnétique est ici utilisé surtout pour calculer l'impulsion des particules chargées à l'aide du rayon de courbure de leur trajectoire dans le détecteur suite à la déviation sous l'action le champ.

L'aimant est un dipôle qui fournit un champ magnétique intégral élevé de 4T avec une courte longueur. Le champ est orienté verticalement et il a une valeur maximale de 1,1 T. La polarité du champ peut être changée pour réduire les erreurs systématiques dans la mesure de la violation CP .

Dans la région du VELO règne un champ magnétique faible pour trouver rapidement les traces et pouvoir les utiliser dans le déclencheur (*trigger*).

II-2-5 Les RICHes

Pour identifier les hadrons dans l'expérience LHCb, on utilise deux détecteurs *Ring Imaging Cherenkov* appelés RICH. Le premier détecteur RICH1, placé juste après le VELO sert à identifier les particules qui ont une impulsion comprise entre 1 et 60 GeV/c. Tandis que le deuxième détecteur RICH2, placé derrière l'aimant et les stations traceuses (*Tracking Stations*) sert à identifier les particules qui ont une impulsion plus élevée (jusqu'à 100 GeV/c).

RICH1 est composé de deux radiateurs : silica aérogel d'épaisseur 5 cm et fluorocarbone C_4F_{10}, 95 cm de long avec un angle polaire d'acceptation de 25 à 300mrd. L'aérogel sert à couvrir les traces correspondant aux quantités de mouvement inférieures. Les RICHes jouent un rôle très important pour distinguer les saveurs des particules dans les désintégrations des hadrons B

$$b \longrightarrow c \longrightarrow s$$

Il est à noter que l'aérogel, avec une épaisseur de 5 cm et un indice de réfraction n=1.03 permet d'identifier les kaons positifs au-dessus de 2 GeV/c et de séparer les pions et kaons jusqu'à environ 10 GeV/c.

Les photons produits par effet Cherenkov dans les radiateurs sont détectés par le détecteur à photons de RICH : les *Hybrid Photon Detectors* (HPDs) spécialement conçus par LHCb. Les photons sont réfléchis par le miroir sphérique, puis par le miroir plan avant d'arriver au détecteur. La disposition du détecteur RICH1 est montrée par la figure n°6. RICH 2 est rempli de CF_4 comme gaz radiateur avec une longueur environ de 180 cm.

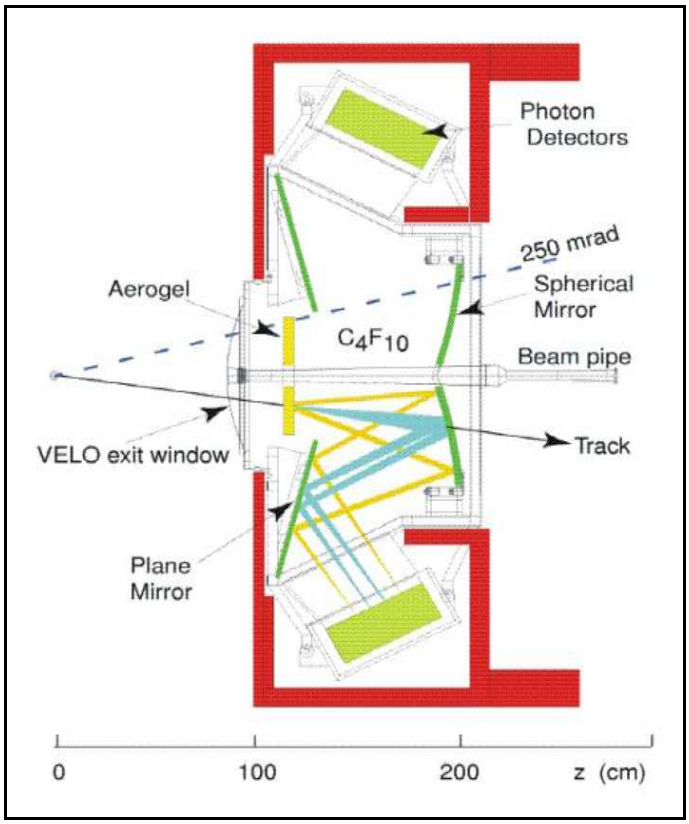


Figure 6 : Disposition verticale du détecteur RICH1

II-2-6 Le trajectographe

Le trajectographe est formé par différentes stations : l'une avant l'aimant (*TraggerTracker*) et trois autres (T1-T3) après. Comme son nom l'indique, le trajectographe est chargé d'enregistrer les points de l'espace où passent les particules chargées dont leurs trajectoires ont été courbées par le champ magnétique créé par l'aimant afin de mesurer leurs impulsions.

La première station appelée « station TT » se comporte en quatre couches de détection groupées en deux. La station TT est utilisée à reconstruire les trajectoires des produits de désintégration des particules neutres de longue vie qui se désintègrent à l'intérieur du volume de VELO et des particules avec une impulsion faible ayant dévié à l'extérieur de l'acceptation avant d'arriver à la deuxième station.

Les trois autres stations sont équidistantes, et chaque station se compose de « *Inner Tracker* » (IT) proche du tube à faisceau et de « *Outer Tracker* »(OT) dans la région à plus large angle.

Le TT et les IT sont constitués par un ensemble de détecteurs à silicium d'une épaisseur de 320 et 410 μm, tandis que le OT est composé de *straw* tubes.

Le trajectographe est constitué par un ensemble de détecteurs à silicium. Une épaisseur de 420μm de ce détecteur fournit une bonne efficacité de détection de particules.

II-2-7 Les calorimètres

Le calorimètre électromagnétique (ECAL) et le calorimètre hadronique (HCAL), avec le détecteur à pavés scintillants (SPD), et le détecteur de pied de gerbe (PS) sont là pour identifier les négatons, les photons et les hadrons en mesurant leur énergie et leur position.

Un sous-module du calorimètre électromagnétique est construit par 70 couches chacune contenant 2mm de plaque de plomb et 4 mm de scintillateur basé en polystyrène.
Un module du calorimètre hadronique est fabriqué par des scintillateurs encastrés dans le fer où les scintillateurs sont placés parallèlement à la direction du faisceau.

II-2-8 Le système à muon

Le système à muons sert principalement à identifier les muons. Il se compose de quatre stations M2-M5 placés entre des murs en bloc de fer pour arrêter toutes les particules qui ne sont pas des muons, et d'une station spéciale devant le calorimètre M1.

II-3 Reconstruction de « trace »[7]

II-3-1 Introduction

Dans le programme de reconstruction de trace, toutes les informations obtenues par le passage de particules dans le détecteur VELO, TT, IT et OT sont combinées pour former les trajectoires des particules à partir du VELO jusqu' au calorimètre. Le passage de la particule est traduit par ce qu'on appelle « hit ». Le programme a pour but de trouver toutes les traces dans l'événement qui laisse des

hits suffisants dans le détecteur, et pas seulement celles qui pourraient provenir d'une désintégration B. Après avoir reconstruit la trajectoire, une trace est représentée par des vecteurs d'état (x,y,dx/dz,dy/dz,Q/p) qui correspondent à une position z de l'expérience.

La performance de la reconstruction est exprimée en utilisant les quantités suivantes :

- L'efficacité de trouver une trace et le bruit de fond correspondant
- La précision du paramètre relevant de l'impulsion reconstruite
- La précision du paramètre d'impact reconstruit
- La précision des pentes de traces au niveau des détecteurs RICH

Ces trois premiers éléments sont très importants pour la production de la désintégration B. Tandis que l'importance de la dernière se révèle sur toutes les traces qui traversent le détecteur RICH et qui ont assez d'impulsion pour produire l'émission de la lumière de Cherenkov. Le schéma des traces reconstruites et des trajectoires y attribuées est montré dans la figure n°7.

II-3-2 Différents types de traces

Selon la nature de la trajectoire dont elles génèrent dans le spectromètre, les traces sont classées comme suit :

1. **Traces longues :** traversent tous les détecteurs, c'est-à-dire du VELO aux stations T. Elles servent à faire la reconstruction et sont très importantes pour la reconstruction d'une désintégration B.

2. **Traces en amont :** traversent seulement le VELO et les stations TT. Elles sont, en général, des traces correspondant à une impulsion faible et qui ne traversent pas l'aimant. Cependant, elles passent à travers le détecteur RICH1 et pourraient donner naissance à l'émission des photons de Cherenkov. Elles sont aussi utilisées pour comprendre les bruits de fond dans l'algorithme relatif à l'identification des particules pour le détecteur RICH. Elles pourraient être utilisées aussi pour la reconstruction de la désintégration B même si on n'a pas une bonne résolution en impulsion.

3. **Traces en aval :** traversent seulement la station TT et les stations T. Les cas les plus fréquents sont ceux qui proviennent de K_S^0 et Λ qui se désintègrent à l'intérieur de l'acceptation RICH.

4. **Traces VELO :** sont mesurées seulement dans le VELO et ont typiquement un large angle ou ne voyagent pas vers le reste du détecteur (*backward tracks*). Elles sont utiles pour la reconstruction du vertex primaire.

5. **Traces T :** sont mesurées seulement dans les stations T. Elles sont typiquement produites dans les interactions secondaires mais utiles pour la conception du modèle global d'identification dans RICH 2.

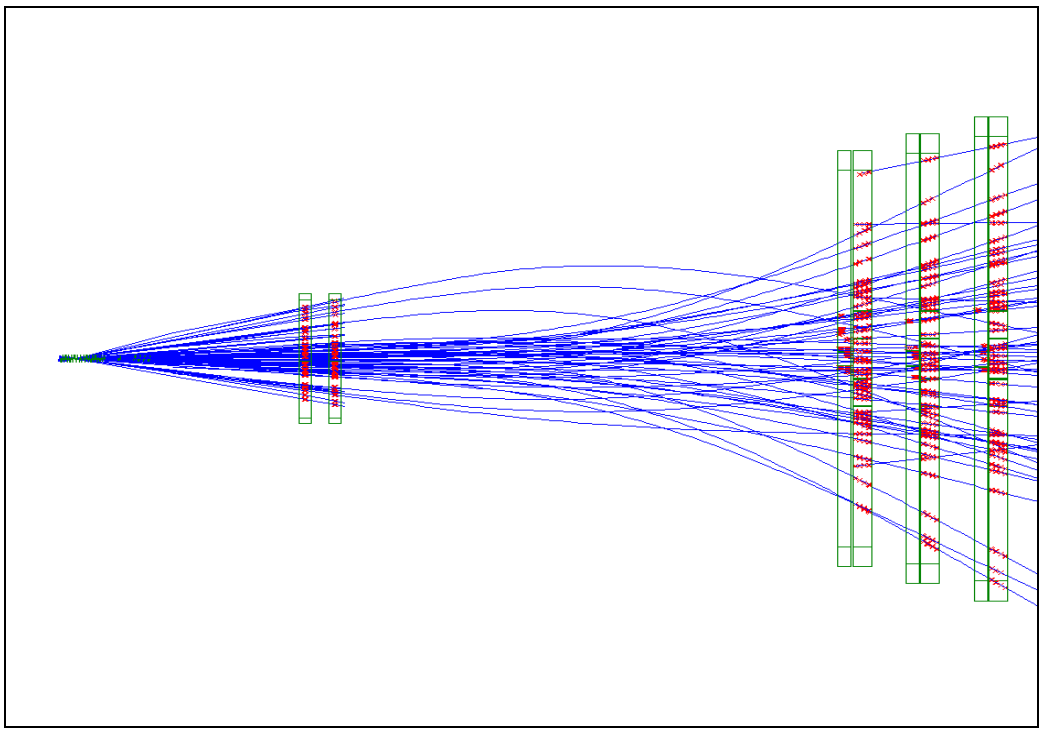


Figure 7 : Schéma des traces reconstruites et les trajectoires y attribuées

II-4 Indentification des particules[7]

La reconstruction des traces est la première partie essentielle pour pouvoir faire par la suite l'identification des particules. Une fois les traces reconstruites, les informations provenant des différents détecteurs tels que : les RICHes, les

calorimètres, le système à muon, permettent de faire cette identification des particules.

Ainsi, pour les particules chargées du type commun (négaton, muon, pion, kaon, proton), les négatons sont identifiés en premier lieu grâce au système calorimètre, puis les muons avec le système à muon et après les hadrons avec le système RICH. Cependant, les détecteurs RICH peuvent aussi aider à bien identifier les leptons ; les informations qui viennent des différents détecteurs sont donc combinées. Les particules électromagnétiques neutres telles que le gamma et le pion neutre (π^o) sont identifiées grâce au système calorimètre où la désintégration du pion neutre donne deux photons gamma bien distincts. Finalement, les particules comme le K_s^0 sont trouvés à partir de leur désintégration qui donne dans le cas du K_s deux pions de charge opposée.

Pour mener à bien cette expérience, l'équipe de LHCb a besoin de procéder à la simulation et nous allons la voir dans le prochain chapitre.

Chapitre III : LA SIMULATION

III-1 Introduction

Tant que le collisionneur LHC n'est pas encore opérationnel, il est indispensable de faire une simulation afin d'avoir des idées sur les conditions expérimentales et de comprendre et, si possible, d'améliorer la performance du spectromètre. Même si le détecteur LHCb n'est pas encore complètement installé, les physiciens prévoient déjà ce qui va se passer à l'intérieur de celui-ci pendant son fonctionnement normal grâce à une simulation préalable. Toutefois, les prévisions se basent sur des théories et leur comparaison avec des données existantes. Il y a donc des indéterminations sur les prévisions.

Ainsi, à part le faisceau test, la simulation se présente comme un des moyens les plus efficaces pour mener à bien la réalisation d'une expérience. Avant même la mise en place d'une installation, on peut procéder à des rectifications dans le but d'avoir une performance meilleure.

Tout ce qu'on prévoit se produire à l'intérieur d'un détecteur et qu'on s'attend à être enregistré par celui-ci peut être simulé à l'avance. Cela permet de faire une étude préalable sur un phénomène quelconque. Quand on collectionnera des données réelles, on pourra les comparer avec celles issues d'une simulation ; cette comparaison aidera à mieux comprendre la situation.

Dans le cas de l'expérience LHCb, le logiciel Gauss s'occupe de cette partie simulation que nous allons développer dans la section suivante.

III-2 Généralités sur Gauss [8]

Gauss est un programme de simulation dans l'expérience LHCb. Il est basé sur la structure de Gaudi (*Gaudi Framework*) et sur les classes du système LHCb.

Gaudi [9] est un « *Framework* » général d'objet orienté spécialement conçu pour donner une infrastructure et un environnement communs pour les différents logiciels d'application de l'expérience. Son architecture, implémentée en C++ et

Python, supporte les applications sur le traitement des données. Dans LHCb, ces applications fonctionnent dans les environnements de traitement, du début jusqu'à l'analyse physique finale. En plus de Gauss qui fait l'objet de notre étude, Boole est utilisé pour simuler la réponse électronique des détecteurs, Brunel pour la reconstruction de donnée et Da Vinci pour l'analyse physique.

Les données accessibles dans le système LHCb [10] sont réparties dans des classes. Une série de classes et la relation entre elles forment la représentation des événements, c'est ce qu'on appelle « LHCb Event Model »[11].

Gauss est construit sur le sommet de la structure Gaudi et il suit son plan architectural. Ainsi, Gauss est une collection de « codes utilisateurs » spéciaux pour la simulation physique.
Après avoir validé Gauss à la fois avec les données du faisceau test et par comparaison avec son prédécesseur, il est maintenant utilisé pour une production massive de données (*DC04* et *DC06 data challenge*). La performance de Gauss en terme de CPU pour une version de code utilisé varie selon la complexité et le type des événements: événements génériques ou évènements de signal.

III-3 Structure de Gauss [8][12][13]

Gauss comprend deux phases indépendantes qu'on peut faire tourner en même temps ou séparément :

- Génération d'événement
- Simulation du détecteur

Normalement, on fait tourner ces deux phases comme un seul travail. Toutes les deux utilisent des librairies et des programmes (*toolkits*) disponibles dans la communauté des physiciens. La structure schématique de ces deux phases est illustrée sur la figure n°8.
Nous allons voir ces deux phases dans les sous sections suivantes.

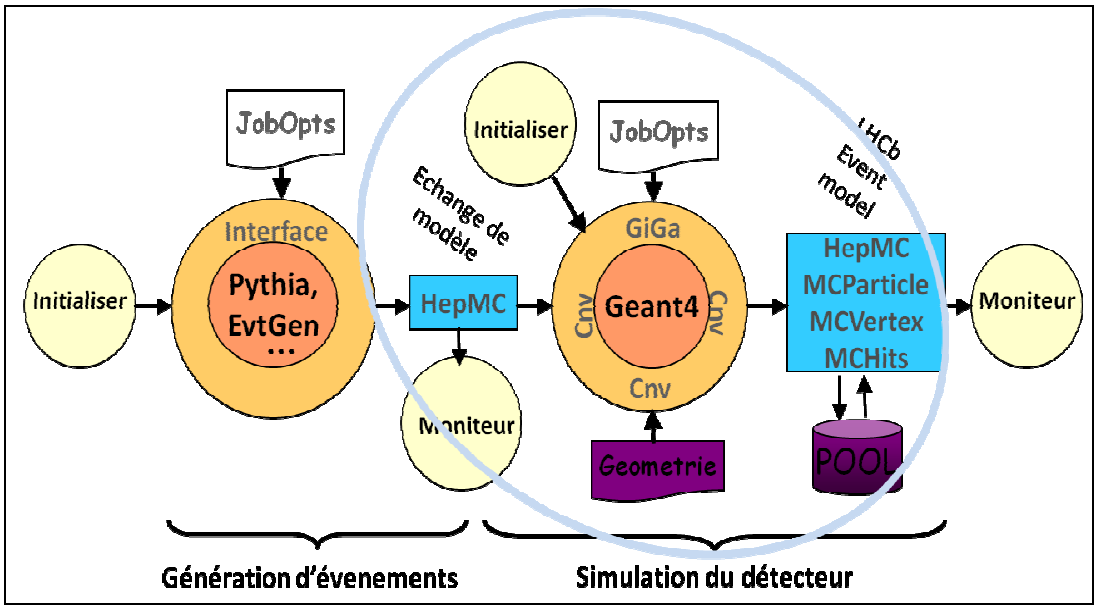


Figure 8 : Structure de Gauss

III-3-1 Génération d'événements

La première phase consiste à générer l'événement: collision entre deux protons et désintégration des particules, tout particulièrement des mésons B dans les canaux d'intérêt pour le programme physique de l'LHCb. Ceci a une interface en Pythia[5][6] pour la production d'événements et en un logiciel de désintégration spécialisé appelé « EvtGen »[14]. Les paramètres de Pythia sont réglés de sorte qu'on puisse reproduire les multiplicités des particules dans des expériences à énergie plus basse que celle de LHC. EvtGen est un logiciel spécial pour la désintégration des mésons B, conçu par l'expérience BaBar (au SLAC) pour décrire le modèle de désintégration des mésons B° et B^+ produits dans la désintégration du Y(4S); il a été adapté à son utilisation dans le contexte de machine hadronique par LHCb, on y ajoutait aussi les désintégrations des B_s, B_c, et Λ_b.La phase génératrice de Gauss simule la condition de fonctionnement et tout ce qui se passe dans la région d'interaction en tenant compte de la taille transversale et longitudinale des paquets de protons et le changement de la luminosité durant l'opération. Une ou plusieurs collisions p-p sont produites selon la luminosité de fonctionnement choisie.

Remarque :

Comme on a dit auparavant, Gauss fait tourner les deux phases normalement dans un seul travail. Mais le problème se pose sur le fait que la simulation physique du détecteur prend beaucoup de temps. On doit avoir une statistique raisonnable alors

que typiquement on ne peut produire que quelques centaines d'événements (500) dans un seul travail. Il faut donc faire tourner beaucoup de travaux en parallèle pour obtenir une statistique significative. C'est là que vient l'idée de créer un mode dit « stand-alone » avec lequel on fait des études au niveau du générateur. Autrement dit, on réalise un travail sans être passé à la deuxième phase. Voici les avantages :

- Le travail se fait très rapidement tout en produisant une statistique raisonnable
- C'est très utile lorsqu'une modification de générateur est envisagée ou on prépare un essai sur le mécanisme de production et sur les canaux de désintégration.
- L'introduction d'un nouveau canal de désintégration dans la chaîne de production est bien préparée.

Les particules produites dans la phase génératrice sont stockées dans le format générique HepMC [15][16] et peuvent être rendues persistantes dans le cas où on veut les utiliser dans des traitements successifs.

III-3-2 Simulation du détecteur

Cette deuxième phase de Gauss consiste à tracer les particules produites par la phase génératrice dans le détecteur LHCb. Elle sert, ainsi, à produire des « hits » quand les particules traversent les détecteurs sensibles. Les données produites sont étudiées directement ou après avoir subi un traitement quelconque. Parmi ces données on cite :

- Particules de Monte-Carlo : type de particule, son énergie, sa direction.
- Vertex de Monte-Carlo : position des interactions proton-proton initiale, toutes désintégrations consécutives, et interactions.
- Hits de Monte-Carlo correspondant au passage de la particule dans le VELO, TT, IT, OT, système de muon, RICH, et dans les calorimètres.

Les données sont liées entre elles avec des pointeurs qui représentent leur relation, par exemple les particules avec leur vertex d'origine.

Cette phase de Gauss utilise un « *toolkit* » appelé Geant4 que nous allons développer dans la sous section suivante.

III-3-3 Geant4 [17][18]

La simulation des processus physiques qui peuvent se produire quand les particules traversent la matière du détecteur dans la condition expérimentale réelle est confiée à un « *toolkit* » appelé Geant4. La navigation des particules dans le champ magnétique ainsi que les différents processus physiques susceptibles de se produire au niveau d'une particule à n'importe quelle énergie sont simulés grâce à ce logiciel.

Geant4 interagit avec Gauss en utilisant une série d'interfaces et convertisseurs encapsulés dans une structure de Gaudi spécialisée pour ce but (GiGa : *Geant4 Interface for Gaudi Application* ou *Gaudi Interface to Geant4 Application*). GiGa permet la conversion de la géométrie du détecteur LHCb dans la description de géométrie de Geant4. Il transforme aussi les particules à la sortie de la première phase de Gauss en format d'entrée de Geant4. Les « hits » produits dans les volumes sensibles de détecteur sortant de Geant4 sous forme de vérité Monte-Carlo sont aussi convertis, et constituent le modèle d'événement LHCb. Le comportement de Geant4 lors de la simulation du détecteur, les modèles physiques à utiliser, les détails de vérité Monte-Carlo à fournir sont contrôlés dans la configuration des options de travail.

Gauss a remplacé la précédente simulation en FORTRAN basée sur Geant3 à la fin de l'année 2003. Geant3 était le « t*oolkit* » standard mondial pour la simulation du détecteur dans la physique des hautes énergies. Le successeur Geant4, développé dans la communauté physique est écrit en C++. Il vient d'un mélange d'une théorie dirigée, paramétrée, et des formules empiriques qui décrivent les différents processus physiques.

Pendant la simulation, chaque particule est déplacée dans des pas (variant d'un micron à quelques centimètres) et à chaque pas Geant4 donne des informations sur :

- Le matériel où la particule se trouve
- La perte d'énergie
- Le changement de position et direction en tenant compte de l'effet du champ magnétique sur les particules chargées

- Les photons (de bremstrahlung ou de Cherenkov) éventuellement générés par la particule
- Les désintégrations pouvant se produire pendant que la particule passe le détecteur
- Le point où la particule entre ou sort d'un détecteur quelconque, etc.

Geant4 donne aussi des informations pour chaque particule au moment de sa création et à la fin de son vol comme :

- Le type de particule
- Son impulsion
- Le processus physique y donnant naissance, etc.

La simulation de Geant4 a été adaptée pour prendre soin aux spécialités de LHCb. Par exemple, la déposition de l'énergie dans les cellules du calorimètre électromagnétique est corrigée selon l'effet de saturation et la simulation complète du calorimètre électromagnétique est réglée avec les données du test faisceau. Le spectromètre LHCb se caractérise par l'identification des particules avec les détecteurs RICH et l'utilisation d'un radiateur à Aérogel pour une gamme d'énergie plus basse. Les processus physiques spécifiques aux RICHs, par exemple l'émission de Cherenkov et la dispersion de Rayleigh, ont été étudiées et validées dans la simulation en comparant avec les données du faisceau test. La simulation des productions de photoélectron dans le HPD a été implantée dans Gauss comme un utilisateur des processus physiques en Geant4 : ceci permet de manipuler directement aux valeurs quantiques efficaces.

Remarque :

Il est à noter que des options sont disponibles pour faire tourner la simulation en lisant juste les événements du générateur à partir d'un fichier qu'on a déjà produit précédemment.

En effet, on peut regarder les données sans les générer encore une fois, en lisant ces fichiers de simulation.

III-4 Contrôle de la simulation [13]

Geant4 possède des seuils de production pour les processus physiques électromagnétiques ; ceci nous permet de :

- Spécifier la portée à laquelle une perte continue d'énergie va s'effectuer. La portée est convertie en énergie pour chaque matériel.
- Créer des particules secondaires uniquement au-delà d'une portée bien spécifiée

Dans Gauss, on a introduit une coupure sur l'énergie cinétique des particules quel que soit leur type. C'est-à-dire on trace la particule jusqu'à ce que cette coupure d'énergie soit atteinte et on s'arrête à ce point. Ceci pourrait se faire par région et par volume dans le détecteur. L'action de Geant4 est ici contrôlée par GiGa.

Il existe une différence entre une coupure sur l'énergie cinétique et sur une coupure en portée mais on peut faire une conversion. Nous donnons, dans la figure 9 un exemple d'équivalence portée-énergie dans différents matériaux.

Les seuils de production et les coupures sur l'énergie cinétique influencent le temps en CPU de la simulation comme on peut le voir dans la figure 10 où est montré l'effet de coupure en énergie sur le temps de traitement d'une particule d'énergie donnée.

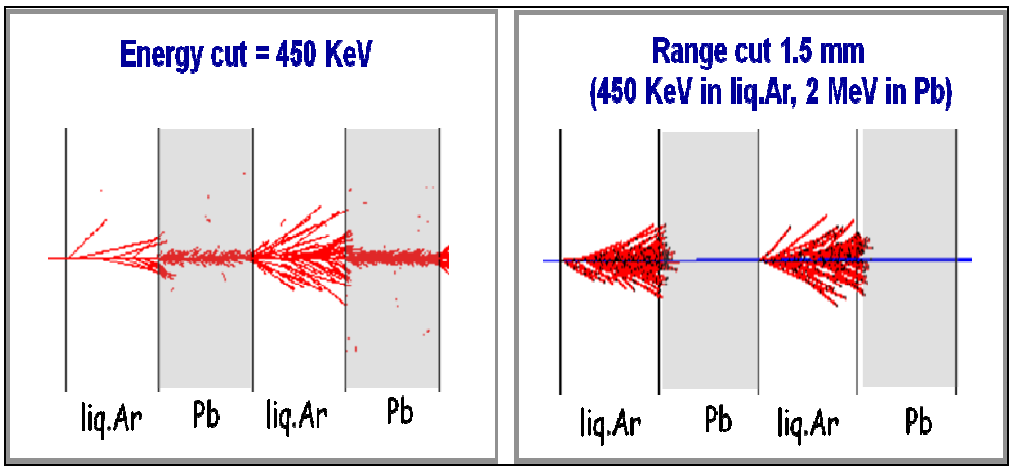


Figure 9 : Exemple d'équivalence portée-énergie

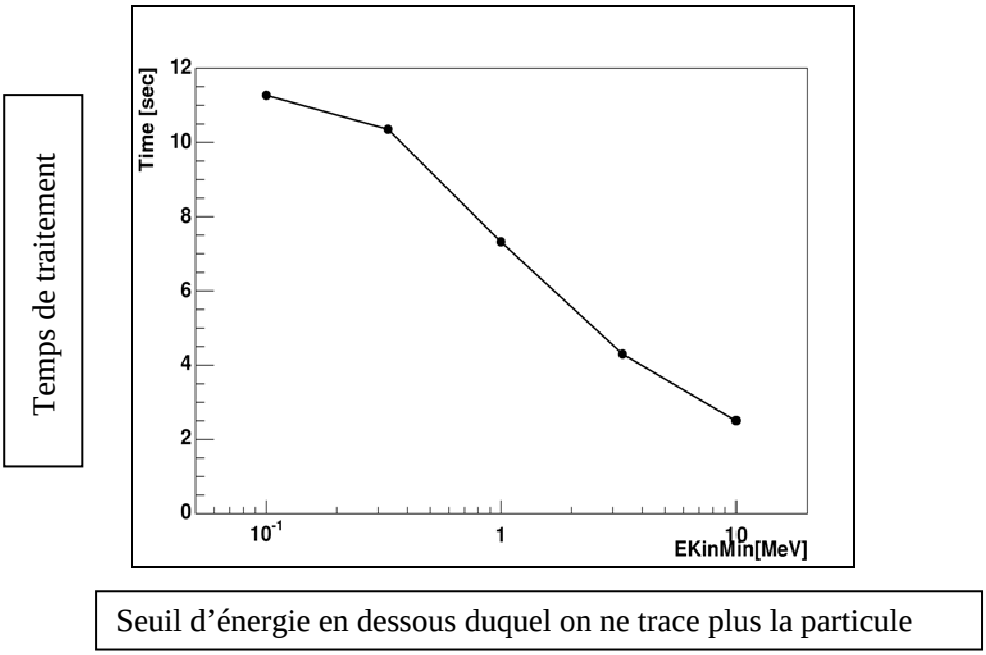


Figure10 : Effet de coupure d'énergie dans le calorimètre électromagnétique avec un négaton de 30 GeV

PARTIE PRATIQUE

Chapitre I : CONFIGURATION DU TRAVAIL EN GAUSS

I-1 Introduction

Gauss est construit sur la structure de Gaudi et fournit un mécanisme sur les algorithmes séquentiels dans cette structure. En effet, Gauss définit le mécanisme qui entre en jeu lorsque ces algorithmes sont exécutés. Les algorithmes exécutés en Gauss ont accès à tous les services déjà installés dans Gaudi au moment de l'exécution et à toutes les données dans le stockage. Les algorithmes placent les données résultant du traitement dans le stockage telles qu'elles soient disponibles aux algorithmes suivants.

Gauss utilise CMT (*Configuration Management Tool*) pour gérer les codes : ceux-ci sont organisés dans des paquets (*packages*) selon leur fonctionnalité qui, à leur tour, sont groupés dans des projets logiciels.

Dans ce travail, nous avons créé un code appelé « *MonitorTrackAction* » pour que nous puissions contrôler le bon fonctionnement de Gauss, et surtout le temps qu'il consomme pour réaliser une tâche. Ainsi, nous avons mis en place un moniteur de base qui sert à évaluer la performance de la simulation en termes de temps CPU consommé dans divers détecteurs et pour différents types de particules. Le temps CPU est la valeur de temps avec laquelle l'unité centrale de traitement exécute réellement une instruction.

Ce moniteur de base a été utilisé aussi pour étudier l'effet sur la performance des choix multiples disponibles pour le processus physiques électromagnétiques et leur réglage (*production cut*) ainsi que les seuils en énergie cinétique pour suivre les particules à travers le détecteur (*tracking cut*). Cette technique utilisée pour suivre les particules a été introduite par Gauss et lui est spécifique.

Dans les sections suivantes, nous allons voir les étapes à suivre quand on veut travailler en Gauss : sa configuration et l'exécution d'un travail standard. Puis, nous allons voir les modifications qu'il nous a fallu faire quand nous avons essayé de réaliser le projet de contrôle.

I-2 Configuration et exécution d'un travail standard en Gauss [12]

Une version publiée en Gauss contient la configuration nécessaire pour un travail de simulation standard de LHCb. Nous allons voir ici comment configurer et exécuter un travail standard.

I-2-1 Structure du logiciel

Toutes les publications faites au niveau du projet logiciel en Gauss se trouvent dans le domaine de publication de logiciels au sein de LHCb ou bien dans le directoire $LHCBREALEASE/GAUSS/GAUSS_vxry où les nombres x et y sont respectivement la version majeure et mineure de Gauss. Pour une version donnée, ce directoire contient tous les packages de Gauss propres à cette version.

Un projet en Gauss dépend du projet logiciel fondamental dans LHCb, du projet en Geant4 et de la structure de Gaudi. Les versions spécifiques des différents projets et packages utilisés sont fixés dans des packages dédiés.

L'application de Gauss réside dans le package ***Sim/Gauss*** du projet. Ce package est utilisé pour construire et exécuter l'application. Le directoire a comme structure :

- **doc** : notes de réalisation
- **job** : exemple de travail en Linux
- **options** : exemple d'option de travail et celle par défaut
- **cmt** : le fichier d'exigence CMT utilisé pour gérer les codes

I-2-2 Plateforme

Actuellement, Gauss est construit et supporté sur les plateformes suivantes :

- ***Scientific Linux CERN 3 (slc3)*** avec compilateur gcc3.2.3
- ***Scientific Linux CERN 4 (slc4)*** avec le compilateur gcc3.4 (32-bit et 64-bit)

I-2-3 Installation de l'environnement

Avant de commencer à travailler en Gauss, il faut activer le CMT. Quand on entre dans « LXPLUS »(le *cluster* des PCs pour le travail interactif au CERN), la variable d'environnement CMTPATH est normalement établie comme étant

$HOME/cmtuser. Ce directoire est automatiquement créé quand on entre pour la première fois comme un utilisateur au sein de l'expérience LHCb et c'est ici que CMT recherchera le code privé de l'utilisateur. Il est accompagné par un fichier spécial **cmt/project.cmt**.

Maintenant, on va dire à CMT quelle version de Gauss on veut travailler avec. Ceci est fait par la commande :

GaussEnv v25r10 ou **setenvGauss v25r10**

Cette commande dit à CMT de localiser dans le domaine publique tous les packages exigés par la version v25r10 de Gauss. Dans la commande ci-dessus et dans la suite du texte, on utilise comme numéro de version d'exemple v25r10, qui est celui qu'on a employé dans le travail.

Même si on veut faire tourner un travail standard, on aura besoin de procéder à quelques modifications. Si on veut définir par exemple le nombre et le type d'événement à simuler ainsi que le fichier de sortie à la fin du travail, on doit amener une copie du package de Gauss dans le directoire CMT du travail en faisant la commande :

```
cd  $HOME/cmtuser
getpack  Sim/Gauss v25r10
cd  Sim/Gauss/v25r10
```

I-2-4 Fichier d'options et contrôle de Gauss

Le directoire des options contient une variété de fichiers avec un suffixe *.opts* pour contrôler la configuration du travail en Gauss au moment de l'exécution. La configuration par défaut est dirigée par l'option de niveau le plus élevé appelée *Gauss.opts* où il existe une liste des paramètres fondamentaux à changer selon le besoin. On peut aussi configurer un travail à l'aide d'un fichier option spécifique.

Maintenant, nous allons voir quelques uns des paramètres de contrôle les plus courants. Avant de faire tourner un travail, il est toujours préférable de spécifier

combien d'événements on va simuler. On contrôle cela en spécifiant le nombre désiré dans la ligne suivante :

```
ApplicationMgr.EvtMax = 500 ;
```

La semence du nombre aléatoire utilisée par tous les générateurs en Gauss est uniquement déterminée pour chaque événement par le *RunNumber* et par l'*EventNumber* dans le traitement. Ceci permet de générer une série des données indépendantes ou bien de reproduire complètement les événements.

On peut spécifier le *RunNumber* et le nombre du premier événement dans les options. On peut aussi générer une série de données indépendantes ou refaire tourner des événements particuliers dans le but d'en faire une nouvelle analyse.

Quand on fait tourner un travail en Gauss, on a besoin de spécifier quel type d'événement on va simuler. Le type d'événement par défaut est le « *minimum bias* »(événements de collision proton-proton génériques). Dans le cas où on veut simuler des événements de signal particulier ou des événements qui contiennent $b\bar{b}$, on devra changer les options de travail. Des options prédéfinies, pour différents types d'événements qu'on peut générer sont disponibles dans des fichiers situés dans un package spécial appelé *DecFiles*.

On peut consulter les types d'événement disponible dans le directoire $LHCBREALEASES/DBASE/Gen/DecFile/vXrY/options/, où vXrY est une version donnée ou bien dans le site web de Gauss.

Gauss écrit, par défaut, les données d'événements dans le fichier de sortie avec une extension ***.sim*** . Pour contrôler ce fichier de sortie, on utilise les options suivantes :

```
1 : ApplicationMgr.OutStream += '' GaussTape.opts'' ;
2 : GaussTape . Output = ''DATAFILE='PFN :
GaussPool_30000000.sim'TYP='POOL_ROOTTREE' OPT='REC' '';
```

Si on ne veut pas avoir un fichier de sortie avec les données, il suffit de commenter la première ligne. Ce fichier est écrit dans le directoire où on a fait tourner le travail. Le contenu du fichier de sortie se trouve dans le fichier *$STDOPTS/SimContents.opts* inclus dans *GaussTape.opts.*

Tout contrôle dans Gauss se fait dans la séquence de contrôle. Dans cette séquence, histogrammes et *ntuples* prédéfinis sont remplis que ce soit pour la phase du générateur ou que ce soit pour la phase de simulation. Dans la production, la séquence de contrôle est mise en marche.

La configuration par défaut des histogrammes de contrôle est disponible dans *Monitor.opts*. Pour avoir des contrôles plus détaillés, un fichier d'option *MonitorInDetail.opts* est disponible et peut être ajouté aux options de travail.

I-2-5 Fonctionnement du travail

Une fois l'installation de Gauss terminée et quand on est dans un directoire approprié, on peut faire tourner le travail. On a comme directoire :

$HOME/cmtuser/Sim/Gauss/v25r10

L'action de faire tourner le travail comporte trois étapes :

- Construire l'exécutable
- Définition de l'environnement pour la version donnée du logiciel
- Exécution du travail suivant la directrice de l'option de travail appropriée

Remarque importante :

On tient à remarquer que Gauss est très lent. Si on fait tourner un travail en Gauss, ceci pourrait prendre jusqu'à plus d'une minute par événement. Cela vient du fait que Geant4 passe beaucoup de temps pour tracer les particules dans le détecteur. Ainsi, la simulation en termes de temps CPU consommé devrait être sous contrôle et, si possible, améliorée sans nuire aux résultats. Ceci est exactement le but de notre travail.

I-2-6 Fonctionnement de Gauss sur Linux

La séquence des commandes suivantes permet de faire tourner interactivement le travail sur Linux :

cd cmt
SetupProject Gauss v25r10
make (seulement la première fois qu'on veut construire l'exécutable)
cd…/job
$CMTCONFIG/Gauss.exe $GAUSSOPTS/v200601.opts >& Gauss.log &
où $GAUSSOPTS/v200601.opts est le fichier d'options principal pour un travail standard

I-3 Configuration de notre travail de contrôle

Comme on a dit auparavant, notre travail consiste à contrôler la performance de la simulation en termes de temps CPU consommé en introduisant des options qui pourraient avoir un effet sur la performance. Ces options contiennent des choix multiples disponibles pour les processus physiques électromagnétiques et leur réglage, y compris les seuils. Dans ce travail, nous nous sommes intéressé aux valeurs de temps consommées par la simulation avec chacune de ces options de travail. Nous avons soumis plusieurs travaux et chaque travail utilise une de ces options spéciales dans le but toujours de vérifier l'effet mentionné ci-dessus.
Mais avant tout, il nous fallait construire le code appelé « MonitorTrackAction » et nous allons décrire ce code dans la sous section suivante.

I-3-1 Le code « MonitorTrackAction »

Ce code est écrit en C++. Il inclut des fichiers à partir de :

- GiGa : pour l'interfaçage sur l'action de base au point de vue traçage des particules et sur la création des « *ntuples* »
- AIDA : pour le remplissage des histogrammes à une et deux dimensions

- CLHEP : pour les vecteurs utilisés
- Gaudi : pour les propriétés des particules, le système des unités et pour le remplissage des histogrammes et « *ntuples* »
- Geant4 : pour les processus physiques, la dynamique des particules, les informations sur la trace, la gestion du traçage, la définition des particules, la table des particules, les vecteurs utilisés et pour le minuteur.
- Gauss : pour la trajectoire des particules et l'information sur les traces

Il utilise des pointeurs, des vecteurs, des constantes de telle sorte qu'on obtient comme résultat :

- Une liste des processus physiques qui ont créé les particules
- Le nom de toutes les particules trouvées
- L'énergie des particules
- Le temps mis par Geant4 pour tracer chaque type de particule et chaque particule originaire d'un type de processus donné
- La durée totale pour tracer toutes les particules
- Le nombre total de particules trouvées
- Des histogrammes et « *ntuples* »

I-3-2 Création de la librairie

Le code a été mis dans un package propre à lui appelé *Sim/Monitor v1r0* dans le directoire *$HOME/cmtuser*. Après avoir créé ce code, il a fallu le compiler. On se met dans le directoire CMT où se trouve le code (c'est-à-dire $*HOME/cmtuser/Sim/Monitor/v1r0/cmt*) et puis on fait la commande :

```
cmt br make
```

Une fois le code compilé, la librairie est aussi créée automatiquement.

I-3-3 Options de travail

A ce niveau là, il nous reste à créer des fichiers options qui vont diriger le travail. Il est à noter qu'il existe un fichier option commun à tout travail en Gauss appelée *Gauss.opts*. Cette dernière sera incluse dans l'option principale appelée *v200601.opts* **.**Par convention, cette option principale est nommée avec la façon suivante : vAAAAMM.opts où AAAA est l'année et MM est le mois pendant lequel

on a déterminé la géométrie du détecteur. Nous allons donner ci-dessous un fichier option à titre d'exemple.

```
ApplicationMgr.ExtSvc += { "NTupleSvc" };
NTupleSvc.Output={"FILE1 DATAFILE='GaussTuples.root'
TYP='ROOT' OPT='NEW'"};
#include "$GAUSSOPTS/v200601.opts"
ApplicationMgr.OutStream -=  { "GaussTape" };// Ne pas écrire le fichier
simulation
ApplicationMgr.DLLs += { "Monitor" };// Charger la librairie monitrice
GiGa.TrackSeq.Members      += { "MonitorTrackAction" } ; // Executer
Moniteur de Trace
GiGa.TrackSeq.MonitorTrackAction.MakeNtuple = true; // Remplire
ntuple dans moniteur de trace
//GiGa.TrackSeq.MonitorTrackAction.ParticleStat = "blabla";// Changer le
nom du fichier de sortie pour la statistique des particles
//GaussGen.RunNumber = 2; // Introduire un autre "random number" , si
pas commenté
ApplicationMgr.EvtMax     = 10; // Nombre d'événements à simuler
#include "$DECFILESROOT/options/30000000.opts"  // Min Bias
//#include "$GAUSSOPTS/ParticleGun.opts"  // ParticleGun example
```

Dans cet exemple de fichier option, nous pouvons voir ligne par ligne les instructions nécessaires pour faire tourner le travail. Dans les trois premières lignes, nous indiquons que le travail va générer des *ntuples*. En effet, on appelle un service extérieur pour créer ces *ntuples* avec la façon dont on les mettra dans le fichier de sortie. La quatrième ligne signifie que l'option principale de Gauss est incluse. Dans la cinquième et sixième ligne, l'instruction dit qu'on n'écrit pas le fichier de simulation. L'instruction dans la septième ligne est utile pour changer la librairie monitrice. Dans la huitième ligne, l'instruction vise à exécuter le moniteur de trace indiqué entre parenthèse ({MonitorTrackAction}). L'instruction dans la dixième ligne est nécessaire pour remplir les *ntuples* dans le moniteur de trace. Si on veut

changer le nom de fichier pour la statistique des particules en « blabla. », la douzième ligne permet de le faire. Les deux instructions qui suivent permettent de changer le nombre aléatoire et le nombre d'événements à simuler. Enfin, les deux dernières instructions servent à spécifier le type d'événement à simuler (*minimum bias* ou *particle gun*).

Avec cet exemple de fichier option, on fait tourner un travail utilisant le moniteur de trace « *MonitorTrackAction* » avec dix événements de « *minimum bias* » générant des *ntuples.*

I-3-4 Introduction des coupures

Pour qu'on puisse manipuler les coupures, il faut avoir dans l'option de travail la commande suivante permettant de les contrôler dans Geant4 via GiGa :

```
GiGa.RunSeq.Members + = {''TrCutsRunAction/TrCuts''}
```

Nous avons soumis quatre travaux avec quatre options de travail différentes. Chaque option de travail contient un genre de coupure spécifié. Il suffit d'inclure dans cette option la série de commandes suivante et avec les valeurs voulues :

```
// Coupure de production par défaut pour tout le
détecteur
GiGa.ModularPL.CutForElectron = 10000. * mm;
GiGa.ModularPL.CutForPositon = 5.0 * mm;
GiGa.ModularPL.CutForGamma    = 10.0 * mm;
// Coupure de trace pour tout le détecteur
GiGa.RunSeq.TrCuts.MuonTrCut     = 10.0*MeV;
GiGa.RunSeq.TrCuts.pKpiCut       = 10.0*MeV;
GiGa.RunSeq.TrCuts.NeutrinoTrCut =  0.0*MeV;
GiGa.RunSeq.TrCuts.NeutronTrCut  = 10.0*MeV;
GiGa.RunSeq.TrCuts.GammaTrCut    =  1.0*MeV;
GiGa.RunSeq.TrCuts.ElectronTrCut =  1.0*MeV;
GiGa.RunSeq.TrCuts.OtherTrCut    =  0.0*MeV;
```

Chapitre II : RESULTATS ET INTERPRETATION

II-1 Introduction

Comme nous avons indiqué auparavant, nous avons soumis quatre types de travaux chacun correspondant à des coupures différentes. Dans ce chapitre, nous allons montrer l'effet de ces coupures sur la performance aussi bien de façon globale sur les travaux que plus en détails en analysant leurs effets sur le nombre et le type des particules produites au cours de la simulation.

II-2 Valeur des coupures

Puisque nous nous intéressons à la comparaison des valeurs de temps consommé par Geant4 pour tracer les particules dans le détecteur, il nous a fallu varier la valeur des coupures et regarder l'effet de cette variation sur la durée. En premier lieu, on doit avoir une référence, étant donné qu'il s'agit d'une comparaison. Pour cela nous avons créé une option de travail « par défaut » servant de référence. Puis, nous avons introduit trois autres types de coupure dans trois autres options afin que nous puissions voir leurs effets sur la performance. Ces options de travail comportent des valeurs de coupure reparties dans le tableau 1 suivant :

Tableau 1 : Les valeurs des coupures pour chaque option de travail

Option de travail	Natures de coupure	Types de particule	Portées (mm)	Energie (MeV)
« Par défaut »	Coupure de production	Electron	10000	
		Positon	5	
		Gamma	10	
	Coupure d'énergie cinétique des traces	Muon, pion, kaon, neutron		10
		Gamma, négaton, positon		1
		Autres		0
« Production cut »	Coupure de production	Electron, positon, gamma	0,5	
	Coupure d'énergie cinétique des traces	Toutes particules		Comme par défaut
« Tracking cut »	Coupure de production	Toutes particules	Comme par défaut	
	Coupure d'énergie des traces	Toutes particules		0
« Cut Filter »	Coupure de production	Toutes particules	Comme par défaut	
	Coupure d'énergie des traces	Muon+/-		10
		Autres particules dans le filtre à muon		500

II-3 Résultats généraux

Nous pouvons voir dans le tableau 2 suivant la répartition du temps passé durant l'application : la durée moyenne, minimum, maximum et la durée totale prise par GiGa pendant qu'il traite les algorithmes, ainsi que la durée totale prise par Geant4.

II-3-1 Résultats avec LXBATCH

Nous avons soumis ces quatre travaux pour 500 événements de « *minimum bais* » dans LXBATCH, le cluster des machines pour les longs travaux au CERN. Les résultats sont résumés dans le tableau suivant :

Tableau 2 : Résultats avec LXBATCH

	Facteur de vitesse de la machine	**GiGa Moyen (s)**	**GiGa Min (ms)**	**GiGa Max (s)**	**GiGa Total (s)**	**Durée Totale dans Geant4(s)**	**Nombre Total de traces dans Geant4**
Default		12.2	7.5	80.2	6160	6193	110875941
	2.04	24.9	15.3	163.6	12567	12634	
Rapport		**1**	**1**	**1**	**1**	**1**	**1**
Prod Cut		23.4	11.2	126.5	11704	11681	165827349
	1.61	37.7	18.0	203.7	18843	18807	
Rapport		**1.51**	**1.2**	**1.24**	**1.50**	**1.49**	**1.50**
Track Cut		24.3	7.5	154.2	12167	12191	225830638
	1.85	44.9	13.9	285.2	22509	22554	
Rapport		**1.81**	**0.9**	**1.74**	**1.79**	**1.79**	**2.04**
Cut Filter		29.5	16.6	185.7	14866	14851	108132723
	0.84	24.8	13.9	156.0	12488	12474	
Rapport		**1.00**	**0.91**	**0.95**	**0.99**	**0.99**	**0.98**

Dans ce tableau apparaissent les facteurs de vitesse de la machine et les différents rapports qui permettent de comparer les résultats obtenus. On voit aussi le temps moyen, minimal, maximal et le temps total pris par GiGa. Dans les deux dernières colonnes se trouvent le temps total pris par Geant4 et le nombre total des traces reconstruites. Les deux dernières colonnes sont celles que « MonitorTrackAction » a essentiellement ajouté.

Lorsque nous regardons le fichier de données, nous pouvons constater que ces valeurs de temps prises par GiGa sont très significatives. Ceci est tout à fait normal car la plus grosse tâche confiée à Gauss, c'est dans ces algorithmes où le travail de simulation est confié à Geant4. Si nous voulons diminuer le temps consommé ou bien si nous voulons que Gauss ne prenne pas beaucoup temps, il faut chercher le moyen pour diminuer ces valeurs. C'est pour cela que notre attention a été tirée sur l'éventuelle diminution qu'on puisse faire à ces valeurs de temps en utilisant une variété de coupures.

Si on change la coupure, cette action pourrait diminuer le nombre de traces dans Geant4. Ceci est pourtant utile dans le cas où ces traces ne produiront pas des signaux dans les détecteurs. Dans ce cas là, si on génère des milliers d'événements sans avoir bien spécifié une coupure appropriée, on risquera d'avoir une terrible perte de temps.

Dans LXBATCH, il existe une centaines de machines qui fonctionnent en parallèle. Ces machines sont dédiées à faire un gros travail. L'avantage est de pouvoir soumettre un tel travail et ceci est mis à part. On peut travailler interactivement sur la machine en attendant que ce travail soit terminé. Mais le problème est d'avoir des machines différentes avec, évidemment, une vitesse différente. C'est pourquoi il faut tenir compte du facteur de vitesse pour chaque machine. On doit donc multiplier la valeur de temps par ce facteur pour avoir sa valeur réelle. Lorsqu'on soumet les travaux dans la même machine, ce problème ne se pose pas.

II-3-2 Résultats avec la même machine

Cette fois ci, nous avons soumis les quatre travaux dans la même machine, c'est-à-dire que tous ces travaux on été soumis dans une même machine un à un. Les résultats sont résumés dans le tableau suivant :

Tableau 3 : Résultats avec la même machine

	GiGa Moyen (s)	**GiGa Min (ms)**	**GiGa Max (s)**	**GiGa Total (s)**	**Durée Totale dans Geant4(s)**	**Nombre Total de traces dans Geant4**
Default	31.4	16.6	193.6	15701.6	15680.1	114836408
Rapport	**1**	**1**	**1**	**1**	**1**	**1**
Prod Cut	42.8	22.2	29.2	21423.7	21351.6	170320760
Rapport	**1.36**	**1.34**	**0.15**	**1.36**	**1.36**	**1.48**
TrackCut	52.1	16.6	272.9	26072.7	25949.1	236446470
Rapport	**1.66**	**1.00**	**1.41**	**1.66**	**1.65**	**2.06**
CutFilter	31.0	16.7	203.1	15538.4	15505.7	111473777
Rapport	**0.99**	**1.01**	**1.05**	**0.99**	**0.99**	**0.97**

En comparant les valeurs, nous pouvons dire que la durée augmente si on réduit les coupures. L'augmentation est beaucoup plus grande dans le cas de « *Tracking cut* » que celui de « *Production cut* » . Le nombre de traces dans Geant4 augmente de la même façon et reste significatif pour les deux cas. Cela veut dire qu'en changeant la coupure mentionnée dans ces options on peut gagner du temps sans perdre assez de traces en tenant compte de leur effet sur les détecteurs.

Les deux options « *cut filter* » et celle de la référence sont presque les mêmes au point de vue temps consommé. On perd, cependant, assez de trace. Cela signifie que le fait de changer la coupure de cette manière ne présente pas beaucoup d'intérêt sur l'économie de temps sauf que si ceci est exactement ce qu'on veut faire.

Nous pouvons constater aussi que les résultats concernant la durée totale prise par GiGa et celle dans Geant4 sont compatibles du fait que l'interface ajoute très peu au temps passé dans les algorithmes de Geant4. Les valeurs dans ces deux colonnes sont presque les mêmes.

Il est à noter que le nombre de traces donné dans ces tableaux est pour 500 événements de *minimum bais* et pas un seul événement. Si on veut avoir le nombre de trace pour un seul événement ces valeurs devraient être divisées par 500.

II-4 Comparaison des histogrammes

Le code « *MonitorTrackAction* » fournit des histogrammes jugés utiles pour évaluer la performance en tenant compte du nombre de particules pour chaque type comparé à la valeur de temps consommé pour tracer ces particules dans le détecteur.

Les histogrammes sont sauvegardés dans le format ROOT une fois le travail terminé. Nous avons utilisé PyROOT (un contrôle en Python de ROOT) pour manipuler ces histogrammes de telle sorte que nous voyons clairement l'effet des coupures développées ci-dessus sur la performance.

Dans la figure 11, nous montrons les histogrammes dans le travail avec l'option par défaut à titre d'exemple. Dans cette figure on peut voir que Geant4 prend plus de temps pour quelques types de particules comme : gamma, négaton, photon optique, π^+, π^-. Cependant, le nombre de ces particules est aussi significatif. En divisant ces deux histogrammes, on obtient le rapport, c'est-à-dire : le temps mis pour tracer les particules divisé par le nombre de particules pour chaque type.

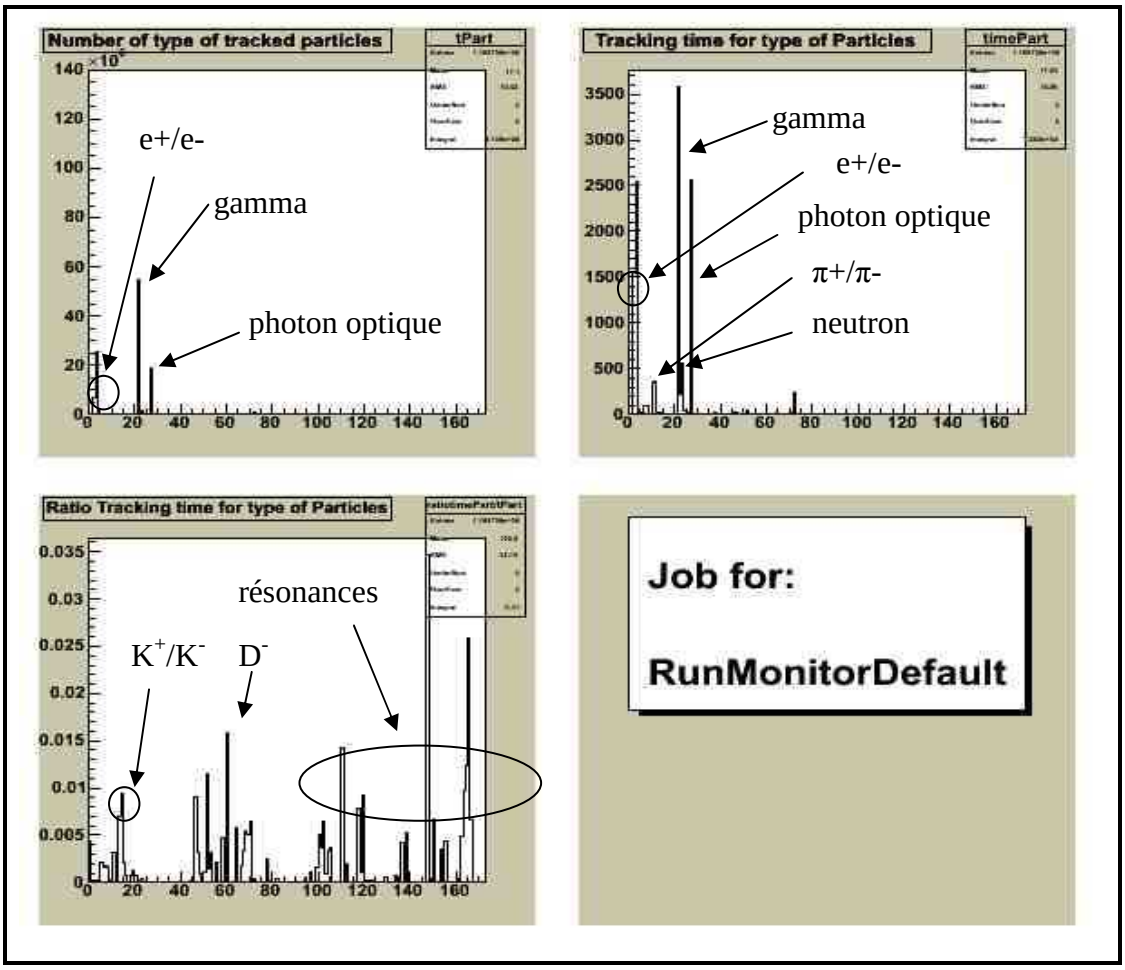
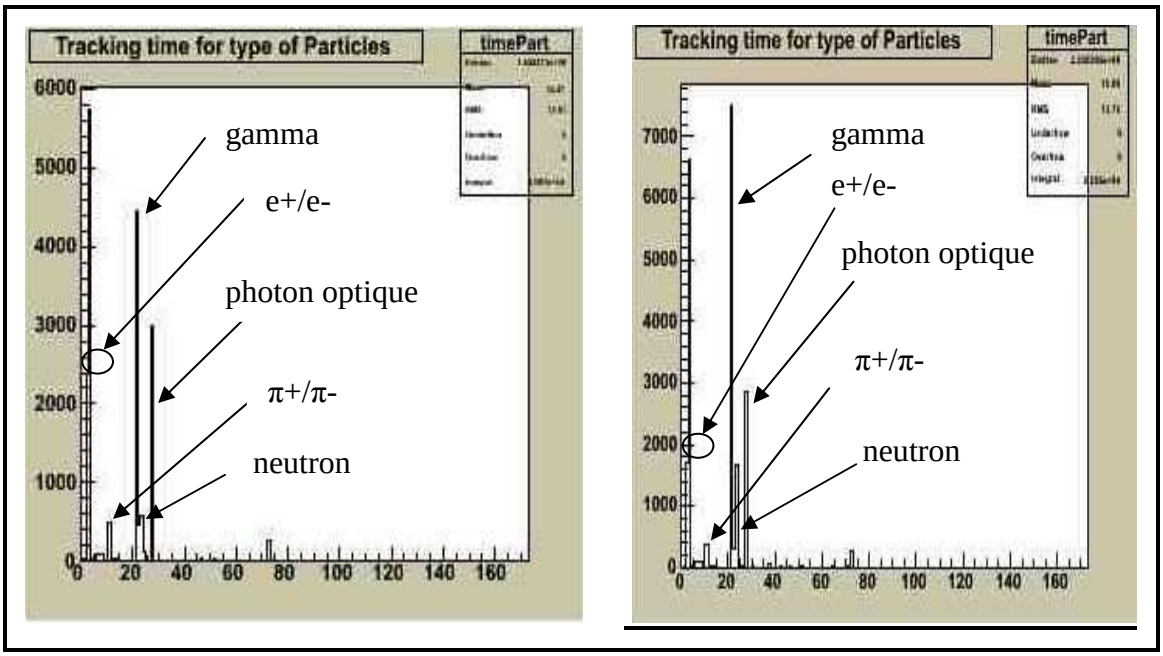


Figure 11 : Histogramme du travail par défaut

Chaque option de travail correspond aux histogrammes propres à elle. Cela nous aide à comparer le choix de coupure qu'on a fait en analysant les histogrammes obtenus. Comme nous avons vu ces quatre options de travail, nous allons comparer aussi leurs histogrammes. Nous préférons comparer ceux qui comportent une grande différence : « *Production cut* » et « *Tracking cut* ».

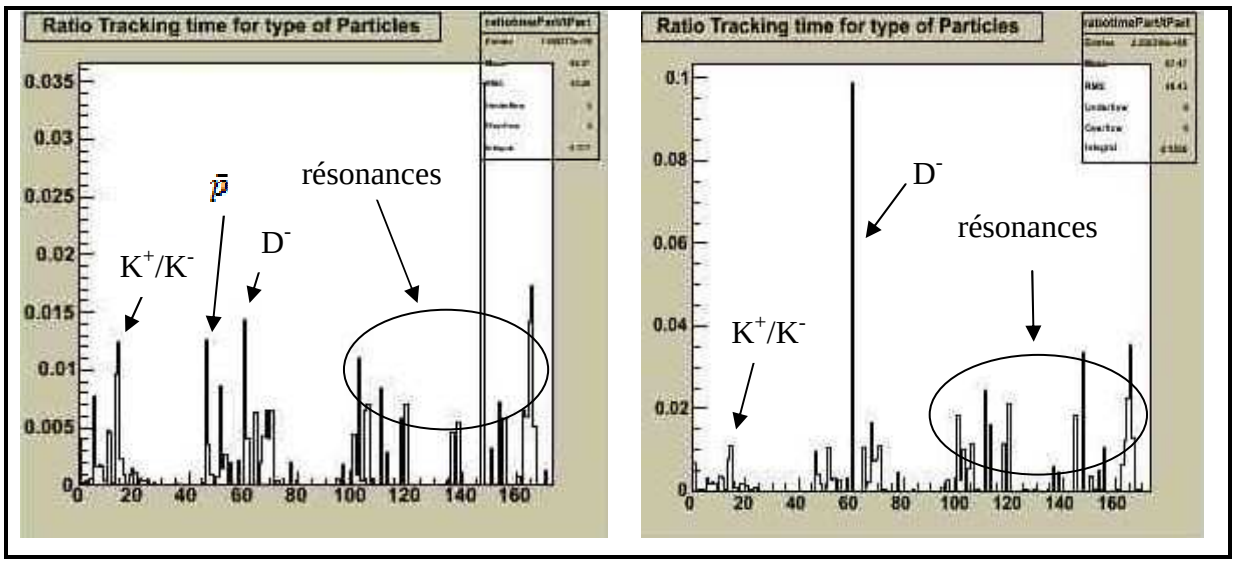


Production cut ***Tracking cut***

Figure 12* : *Histogrammes correspondant à « Production cut » et à « Tracking cut »

Dans ces histogrammes nous voyons la différence entre les valeurs de temps et en plus pour chaque type de particule. Ainsi, on connaît quel type de particule occupe plus de temps dans Geant4 et on imagine quelle est la coupure la plus convenable.

Il se peut que la grande durée vienne du nombre de particules très élevé et pas du traitement au niveau des algorithmes. Pour voir cela plus près, nous avons divisé l'histogramme illustrant le temps consommé avec celui qui illustre le nombre des particules tracées. Le rapport des histogrammes pour ces deux options de travail est montré dans la figure 13 ci-dessous.



Rapport « Production cut » ***« Rapport Tracking cut »***

Figure 13 : Rapport des histogrammes

A partir des deux histogrammes, nous pouvons voir pour quelle particule Geant4 prend beaucoup de temps. Cela nous aide à mieux régler la coupure en spécifiant la portée appropriée. C'est-à-dire que nous diminuons la portée des négatons si, par exemple, ceux-ci représentent une grande fraction de temps dans Geant4.

CONCLUSION

L'application de Gauss permet de simuler le comportement du détecteur LHCb. La génération des événements se fait très rapidement dans Gauss mais la simulation du détecteur y occupe beaucoup de temps. Ceci vient du fait que Geant4 passe une durée assez grande pour tracer la particule dans le détecteur.

Dans ce travail, nous avons pu constater que cette valeur de temps élevée est passée dans le traitement des algorithmes contrôlé par l'interface GiGa. Cette valeur devrait être diminuée à un niveau aussi bas qu'il est raisonnablement possible. Nous avons recouru aux processus physiques électromagnétiques afin de pouvoir mettre des coupures qui ont changé la valeur de temps consommé.

Le moniteur de base « *MonitorTrackAction* » prend des informations à partir de Geant4 au début et à la fin des traces. Il calcule le temps et donne dans le fichier de sortie le nombre de particules trouvées avec leur nom et le temps pour les tracer. Il donne aussi le nombre et le type de processus physiques qui ont créé les particules avec le temps passé pour chaque processus. Grâce à l'interface avec ROOT des histogrammes sont remplis et en suite exploitables.

L'introduction des quatre types de coupure nous a permis de dire qu'en augmentant la valeur de la portée pour le négaton, le positon et pour le photon gamma nous diminuons significativement la durée. Mais le nombre de trace dans Geant4 reste significatif. Nous pouvons affirmer aussi que la coupure de trace allant jusque 500 MeV pour toutes les particules se trouvant dans le système à muon ne diminue presque pas la valeur de temps consommé. Alors que cette action diminue légèrement le nombre de traces dans Geant4.

Ainsi, nous pouvons améliorer la performance de la simulation en termes de temps CPU consommé, en choisissant une coupure appropriée. Ce choix dépend essentiellement de l'objectif à rechercher. Si on ne voit pas des effets significatifs sur

la réponse du détecteur à la fin de la simulation, il est préférable d'utiliser la coupure comme « *production cut* ». Il suffit de choisir une option appropriée pour chaque utilisation envisagée.

Ce travail nous a permis aussi de vérifier que Geant4 passe beaucoup de temps dans les algorithmes car la durée totale dans Geant4 et celle dans GiGa sont presque les mêmes. Par la pensée, ceci est tout à fait logique, et nous avons pu le vérifier par l'expérience.

BIBLIOGRAPHIE

[1]Wikipedia Foundation (2008).
CP- Violation, http://en.wikipedia.org/wiki/CP_violation

[2]S. Eidelman, (2004) PDG (Particule Data Group), Particule Physics Booklet, National Laboratory Berkeley- Etats-Unis, 320p

[3]CERN Collaboration, (2008).
LHCb- A high Energy Physicy Studying CP violation at CERN's LHC http://lhcb-public.web.cern.ch/lhcb-public/html/introduction.ht

[4]LHCb Collaboration, (1998) LHCb Technical Proposal, 170p

[5] LUND University, (2008). Theoretical Physics http://www.thep.lu.se/ ~torbjorn/talks/realease81.pdf

[6] LUND University, (2008). Theoretical Physics, http://www.thep.lu.se/ ~torbjorn/Pythia-html

[7]LHCb Collaboration, (2003) LHCb Reoptimized Detector Disign and Performance, Technical Disign Report, 127p

[8]LHCb Collaboration,(2005) LHCb Computing, Technical Design Report, 104p

[9] CERN Collaboration (2007).THE GAUDI PROJECT, http://proj-gaudi.web.cern.ch/proj-gaudi/

[10] LHCb Collaboration (2007). LHCb Computing, http://lhcb-comp.web.cern.ch/LHCb-realease-area/DOC/lhcb/

[11] LHCb Collaboration (2007). The LHCb Event Model, http://cern.ch/lhcb-comp/Frameworks/EventModel/

[12]LHCb Collaboration (2007) . THE GAUSS PROJECT, http://lhcb-comp.web.cern.ch/lhcb-comp/simulation/Gauss.pdf

[13] LHCb Collaboration (2007) THE GAUSS PROJECT, http://lhcb-comp.web.cern.ch/LHCb-realease-area/DOC/gauss/

[14]LHCb Collaboration (2007) . THE GAUSS PROECT, http://cern.ch/LHCb-realease-area/DOC/gauss/generator/ evtgen.php

[15] CERN Collaboration (2008). HepMC- a C++ Event Record for Monte Carlo Generators, http://lcgapp.cern.ch/project/simu/HepMC

[16] CERN Collaboration (2008). HepMC- a C++ Event Record for Monte Carlo Generators - Summary, http://savannah.cern.ch/project/hepmc

[17] LHCb Collaboration (2007). LHCb Computing http://lhcb-comp.web.cern.ch/LHCb-realease-area/DOC/geant4

[18] CERN Collaboration (2007). Geant4, http://geant4.web.cern.ch/geant4

CONTRÔLE DE PERFORMANCE DE GAUSS, LE LOGICIEL DE SIMULATION DANS L'EXPERIENCE LHCb

Résumé : Cette étude se propose de contrôler la performance de Gauss, le logiciel de simulation dans l'expérience LHCb. Un moniteur de base a été mis en place pour évaluer cette performance en termes de temps CPU consommé.

Gauss est très lent et la cause majeure de cette occupation est le traitement des algorithmes qui emploient l'interface GiGa pour se servir de Geant4.Cela vient du fait que Geant4 prend beaucoup de temps pour tracer les particules dans le détecteur. Le temps de la simulation devrait, ainsi, être sous contrôle et si possible amélioré sans nuire aux résultats.

Nous avons créé un code appelé « MonitorTrackAction » qui prend des informations à partir de Geant4 au début et à la fin des traces. Il calcule le temps et donne dans un fichier de sortie le nombre de particules trouvées avec leur nom et le temps mis pour les tracer. Il donne aussi le nombre et le type des processus physiques qui ont créé les particules avec le temps passé pour tracer les particules produites et ceci pour chaque type de processus.

Les résultats trouvés ont montré que la performance de Gauss en termes de temps CPU consommé peut être contrôlée en choisissant une coupure appropriée. Ce choix dépend essentiellement de l'objectif recherché en faisant la simulation.

Mots clés : Violation CP - Expérience LHCb - Gauss - Simulation – Geant4 - Temps CPU consommé - Coupure

---

PERFORMANCE MONITORING OF GAUSS, THE SIMULATION SOFTWARE IN LHCb

Abstract: *The aim of this study is to control the performance of Gauss, the simulation software in the LHCb experiment. A basic monitor has been put in place to evaluate this performance in term of CPU time consumption.*

Running Gauss is very slow and the time spent is actually allocated in algorithm processing which uses GiGa interface for Geant4. It is due to the fact that Geant4 takes a lot of time to track particle through the detector. Therefore the simulation time needs to be kept under control and if possible improved without compromising the results.

We created a code called "MonitorTrackAction" which takes information from Geant4 at the beginning and the end of a track. It computes the time and gives in the output file: number of particles found with their names and the tracking time. It gives also the number of the processes, the type of process and time spent to track created particles for each process.

The results have shown that the performance of Gauss in term of CPU time consumption can be controlled by putting appropriate cut. The choice of cut depends basically on the simulation purpose.

Keywords: CP violation – LHCb experiment – Gauss – Simulation – Geant4 – CPU time consumption - Cut

---

Encadreur:
Pr RABOANARY Roland

Impétrant:
ANDRIANALA Fenompanirina
Tel: 033 14 804 15
E-mail: fenompanirina@yahoo.fr
Adresse: Lot II N 185 A ter Soavinandriana Antananarivo 101