CHAPITRE 5

Reconstruction de vertex et identification de jets de quarks b

A partir de l'analyse des données recueillies par les détecteurs étudiés précédemment, nous pouvons reconstruire les trajectoires des particules, déterminer le vertex primaire de l'interaction proton-proton du faisceau ainsi que les vertex secondaires, points de désintégration des particules qui ont volé.

La reconstruction du vertex primaire est utile entre autre pour mesurer précisément l'énergie transverse manquante. La reconstruction des vertex secondaires peut permettre d'identifier les jets[1] issus de la fragmentation de quarks b.
Ce chapitre a pour objet la présentation de différents algorithmes de reconstruction de traces, de vertex primaires et secondaires inclus dans le programme officiel de reconstruction et d'analyse de CMS : O.R.C.A. (Object-oriented Reconstruction for CMS Analysis), et en particulier de l'algorithme "Elastic Arms" (EA) consacré à la recherche de vertex secondaires.

5.1 Motivation de la recherche de la position des vertex

La recherche de la position du vertex primaire joue un rôle déterminant pour identifier l'événement sélectionné par le système de déclenchement parmi les 25 événements empilés par croisement de faisceau. Mais la reconstruction du vertex primaire permet aussi d'améliorer des performances et de réduire le temps de calcul de la reconstruction, surtout dans les événements où le calcul de l'énergie transverse manquante est essentielle et où peu ou pas de traces pointent vers le vertex primaire ($H \rightarrow \gamma\gamma$). Par exemple, la mesure de l'impulsion des particules provenant du vertex primaire est réajustée avec la position du vertex primaire.

[1] un jet est un ensemble collimé de particules hadroniques secondaires très énergétiques.

La recherche de la position des vertex secondaires offre pour principal intérêt d'accéder à une information sur les particules à faible durée de vie (entre quelques picosecondes et plusieurs centaines). Avant leur désintégration, la distance parcourue par ces particules est insuffisante pour pouvoir reconstruire leur trajectoire. Certaines de ces particules peuvent également être neutres. Cependant si leurs produits de décroissance sont précisément mesurés, il est possible de déduire les caratéristiques cinématiques et physiques de ces particules qui se sont désintégrées au point du vertex secondaire. Par exemple, une des manières possibles de mettre en évidence la violation CP est d'observer une différence entre les produits de désintégration d'un B^0 et ceux d'un $\bar{B}^0$ dans le canal $B^0 \rightarrow J/\psi K_s^0$ donnant accès la mesure de $sin(2\beta)$.

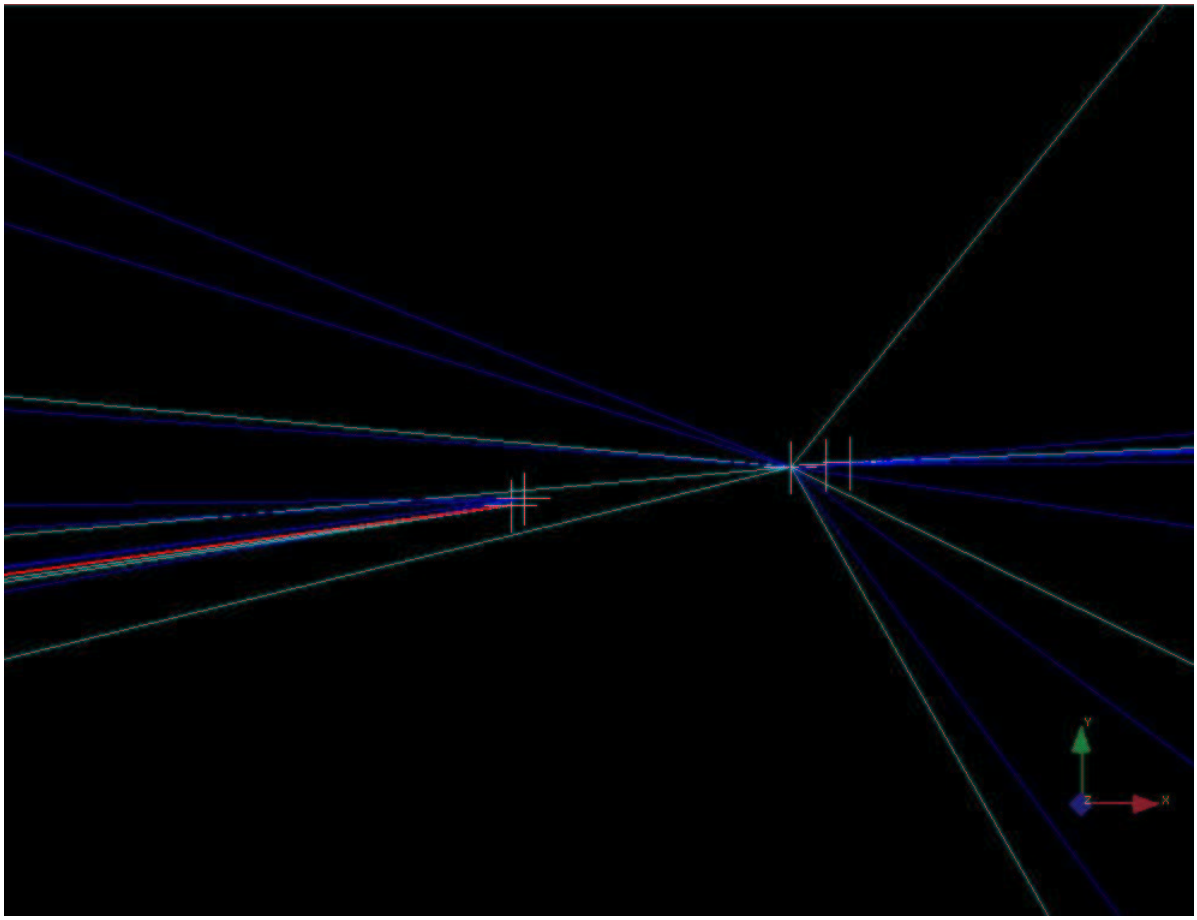

Figure 5.1 – Un événement $b\,\bar{b}$ composé de 21 traces (ligne de couleur), et de 5 vertex simulés dont deux vertex secondaires.

Afin d'aborder la violation CP dans les événements $b\bar{b}$ mais aussi reconnaître le quark top ou découvrir le boson de Higgs, il faut pouvoir identifier les jets issus de la fragmentation de quarks b. Presque toutes les signatures de jets de b ont au moins un vertex secondaire,

du fait du temps de vie relativement long des particules contenant un quark b ($\sim 1,5$ ps) qui équivaut à un parcours de 2-3 mm au LHC (avec un facteur de Lorentz $\gamma \simeq 4$) comme le montrent les vertex secondaires (les 2 croix à gauche du vertex primaire) de la figure 5.1.

Certaines particules ont un temps de vie plus long ($\tau(\Lambda) = 2,6\ 10^{-10}s$ ou $\tau(K_s^0) = 0,9\ 10^{-10}s$) et les vertex secondaires peuvent être créés à plusieurs dizaines de centimètres dans le trajectographe (long de 5,4m), ce qui complique considérablement la reconstruction des traces à cause de la faible multiplicité des points de reconstruction des traces.

Pour chaque type de vertex : primaires, secondaires proches (à moins de 4 cm) et secondaires éloignés (à plus de 4 cm), différents algorithmes de reconstruction de vertex sont disponibles (tableau 5.1). Le choix de l'algorithme de reconstruction est à la fois basé sur le type de particules étudiées et les performances des différents algorithmes en fonction des performances requises (meilleure résolution possible, meilleure efficacité possible, meilleure pureté possible...).

type de vertex	primaires	secondaires	
		proche	éloigné
particules	H, ...	B, τ, ...	Λ, K_s^0, V_0, ...
exemples de canaux de désintégration	$H \rightarrow ZZ^* \rightarrow 2l^+2l^-$	$\tau^- \longrightarrow \mu^-\bar{\nu_\mu}\nu_\tau$ $B_s^0 \rightarrow \mu^+\mu^-\phi \rightarrow \mu^+\mu^-K^+K^-$	$\Lambda \longrightarrow p\pi^-$ $K_s^0 \longrightarrow \pi^+\pi^-$
Algorithmes de reconstruction	Méthode des amas (partie 5.3.4.2)	Méthode de l'"Elastic Arms" (partie 5.3.5.2)	
	Méthode de "binning" (partie 5.3.4.1)	Méthode de PrimaryVertexFinder (partie 5.3.6.1)	
	Méthode des gaussiennes (partie 5.3.4.3)	Méthode de $D_0\phi$ (partie 5.3.6.2)	

Tableau 5.1 – Choix entre plusieurs algorithmes de reconstruction en fonction du type de vertex.

La reconstruction de vertex secondaires est donc primordiale pour la physique du B à la fois pour caractériser les événements contenant des quarks b mais aussi pour mesurer précisément le temps de vol.

5.2 Le logiciel de reconstruction et d'analyse de CMS : O.R.C.A.

Le logiciel O.R.C.A. (Object-oriented Reconstruction for CMS Analysis) [ORC] reconstruit des événements à partir de données simulées ou réelles et les analyse. Les différentes parties d'O.R.C.A. doivent être développées en prenant en compte :

- les performances comme la précision, l'efficacité et la rapidité d'exécution des programmes,
- la modularité permettant l'ajout de nouvelles méthodes, comme par exemple, pour la reconstruction des traces,
- la flexibilité permettant de redéfinir ou d'ajouter des classes,
- et la facilité de maintenance.

Pour ces raisons, le langage orienté objets de programmation utilisé est le langage C++ et la base de données Objectivity [OBJ] permet de stocker les événements simulés ainsi que certaines informations des événements reconstruits.

5.2.1 Principe de la programmation orientée objet

Trois générations de langage se sont succédées et coexistent encore aujourd'hui : les langages dits linéaires, modulaires puis objets.

Tous les premiers langages s'exécutent de façon linéaire. Chaque ligne du programme est lue puis exécutée jusqu'à la dernière. Il est possible de faire des sauts ou des boucles mais le principe reste le même. Cette approche peut difficilement s'appliquer à des programmes complexes. De plus, les développeurs ne peuvent pas réutiliser des outils de programmation déjà écrits. Par exemple le fortran et l'assembleur sont des langages linéaires.

Afin, de réutiliser le code et d'éviter les redondances, les langages dits modulaires voient le jour. Le principe est de regrouper un ensemble d'instructions dans des fonctions ou procédures. En effet, chaque tâche exécutée par un programme représente un nombre variable d'instructions. Ces instructions sont réunies (en bibliothèques) afin de pouvoir segmenter le code et favoriser la réutilisation de celui-ci. Le langage C fait partie de ces langages.

Enfin, la méthode orientée objet est apparue. Elle est en fait une évolution de l'approche modulaire. Elle lui apporte principalement trois aspects primordiaux :

- *L'encapsulation* : cette technique permet de réunir des variables et des fonctions au sein d'une même entité nommée classe. Les variables sont appelées les données membres, les fonctions sont appelées les méthodes. L'accès aux données et méthodes peut être aussi réglementé.
- *L'héritage* : cette technique permet de définir une hiérarchie de classe. Chaque classe fille hérite des méthodes et des données de ses "mères". En pratique, la classe de base est une classe générique, ainsi plus on descend dans la hiérarchie, plus on spécialise cette classe.
- *Le polymorphisme* : ce nom vient du grec et signifie "qui peut prendre plusieurs formes". Cette caractéristique offre la possibilité de définir plusieurs fonctions de même nom mais possédant des paramètres différents. La bonne fonction est choisie en fonction de ses paramètres lors de l'appel.

5.2.2 Un exemple : la classe RecVertex

5.2.2.1 La notion d'encapsulation

Le choix des caractéristiques de chaque objet est aussi pertinent que le choix des objets eux-mêmes. Par exemple, les données caractérisant un vertex ne sont pas les mêmes que celles qui caractérisent une trace (tableau 5.2).

événements physique	$H \to ZZ \to l^+l^-jj$ $K_s^0 \to K\pi$	
éléments reconstruits	trace trajectoire ...	vertex primaire vertex secondaire ...
caratéristiques principales	position erreur sur la position qualité de la reconstruction paramètre d'impact impulsion ...	position erreur sur la position qualité de la reconstruction l'ensemble des traces associées probabilité d'être un vertex primaire ...
classes C++ associées	RecTracks TrajectoryStateOnSurface FreeTrajectoryState ...	RecVertex LinearVertexFitter VertexFinder (partie 5.3.5.4) ...

Tableau 5.2 – Association entre les événements physiques, la reconstruction et le code O.R.C.A.

Pour illustrer ces propos, la classe RecVertex associée à chaque vertex reconstruit par O.R.C.A. va être détaillée. Pour chaque information physique utile définissant un vertex reconstruit (position, qualité de la reconstruction, etc...), on lui associe un membre privé de la classe RecVertex. Ces membres sont les suivants :

- un objet GlobalPoint (nommé thePos) correspondant à un vecteur à 3 dimensions (x, y, z) définissant la position du vertex.
- un objet GlobalError (nommé thePosError) correspondant à une matrice 3×3 définissant l'erreur sur la position du vertex.
- un objet RecTrackContainer (nommé theAssociatedTracks) correspondant à l'ensemble des traces associées au vertex.
- un nombre réel (nommé theChi2) correspondant au χ^2 de la reconstruction du vertex.
- un nombre réel (nommé theProb) correspondant à la probabilité du vertex d'être un vertex primaire.
- une valeur booléenne (nommée theValid) précisant si le vertex est valide ou non (un

vertex est invalide en cas de mauvaise reconstruction) mais en réalité tous les RecVertex sont valides.

De plus, un constructeur portant le même nom que la classe est une fonction membre appelée automatiquement lors de la création d'un objet et permet l'initialisation des variables.

Afin d'accéder aux valeurs stockées dans ces variables ou pour les modifier, nous avons défini les méthodes suivantes :

	RecVertex () Le constructeur par défaut d'un objet RecVertex
	RecVertex (const GlobalPoint &pos, const GlobalError &posError, const vector< const RecTrack *> &tracks, float chi2) Le constructeur initialisant les membres privés de l'objet RecVertex lors de la création.
GlobalPoint	position () const Cette méthode retourne la position du vertex via l'objet "GlobalPoint".
GlobalError	positionError () const Cette méthode retourne l'erreur sur la position du vertex via l'objet "GlobalError".
Vertex : :TrackPtrContainer	tracks () const Cette méthode retourne l'ensemble des traces associées au vertex via l'objet "RecTrackContainer"
Les méthodes position,	positionError et tracks sont héritées de la classe Vertex.
int	degreesOfFreedom () const Cette méthode calcule et retourne le nombre de degrés de liberté.
float	totalChiSquared () const Cette méthode retourne le χ^2 de la reconstruction du vertex.

float normalisedChiSquared () const
Cette méthode retourne le χ^2 normalisé par le nombre de degrés de liberté.

bool isValid () const
Cette méthode retourne le booléen de validité du vertex.

vector <const RecTrack *> originalTracks () const
Cette méthode retourne l'ensemble des traces à l'origine de la création du vertex via un vecteur de traces reconstruites.

5.2.2.2 La notion d'héritage

Dans le logiciel O.R.C.A., trois types de vertex ont été définis :
- les vertex issus de la reconstruction des événements (RecVertex, figure 5.2),
- les vertex simulés servant à la simulation et à la validation des algorithmes (TkSimVertex, figure 5.2)
- et enfin un vertex possédant peu d'informations, pour des utilisations simples ou pour les premières phases de test de programmes (TrivialVertex, figure 5.2).

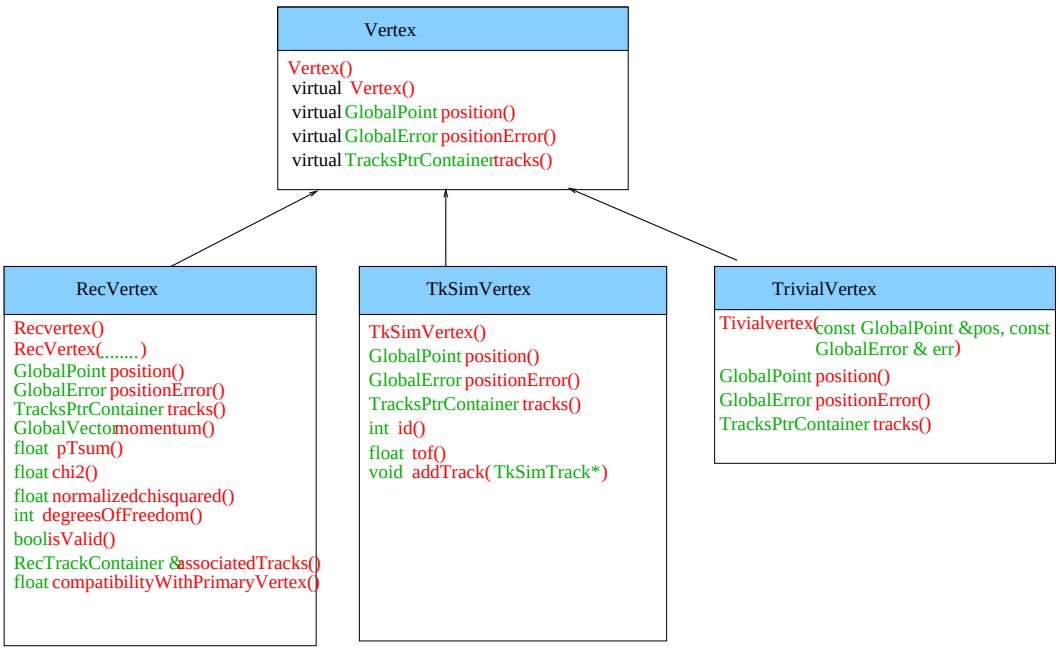

Figure 5.2 – Diagramme d'héritage de la classe Vertex.

Ces trois types de vertex possèdent certaines informations communes comme la position du vertex, l'erreur sur la position du vertex et la liste des traces qui lui sont associées et donc les mêmes méthodes pour y accéder. Il est alors intéressant de définir un type de vertex générique qui référence ces méthodes communes : il s'agit de la notion d'héritage.

Cette classe générique, la classe Vertex, est appelée classe mère et les classes RecVertex, TkSimVertex et TrivialVertex sont les classes filles qui héritent de la classe Vertex.

5.2.2.3 La notion de polymorphisme

Les méthodes de la classe mère peuvent avoir une implémentation mais si ce n'est pas le cas, ces méthodes sont dites virtuelles.

Si la méthode est virtuelle, il s'agit uniquement de définir un nom et une liste d'arguments pour la classe mère et l'action associée à cette méthode sera définie pour chaque classe fille.

Si la méthode n'est pas virtuelle, elle possède donc un ensemble d'actions pouvant être utilisé tel quel par les différentes classes filles, ou elle pourra être redéfinie en fonction des besoins dans chaque classe fille par polymorphisme. La figure 5.2 montre le diagramme d'héritage de la classe Vertex. Nous constatons que les trois méthodes filles possèdent les mêmes méthodes que la classe mère auxquelles s'ajoutent leurs propres méthodes.

5.2.3 Les différents modules d'O.R.C.A.

Ce logiciel se présente sous la forme d'un ensemble de modules et de sous-modules interagissant entre eux via un ensemble de classes. Les principaux modules de la version O.R.C.A_6_1_1 (figure 5.3) sont les suivants :

1. le module "C.A.R.F." (CMS Analysis Reconstruction Framework) décrit l'environnement de travail d'O.R.C.A. en incluant l'environnement de reconstruction, l'environnement d'analyse et la sauvegarde des données reconstruites lors de l'exécution d'un programme.
2. le module "GeneratorInterface" est dédié aux classes effectuant le lien entre les caractéristiques des particules données par le programme HEPEVENT et l'environnement C.A.R.F.
3. le module "Magnetic Field" décrit la cartographie du champ magnétique dans le détecteur.
4. le module "CommonDet" décrit les caractéristiques générales du détecteur comme sa géométrie et dispose de diverses classes utiles pour stocker des événements (simulés et reconstruits).

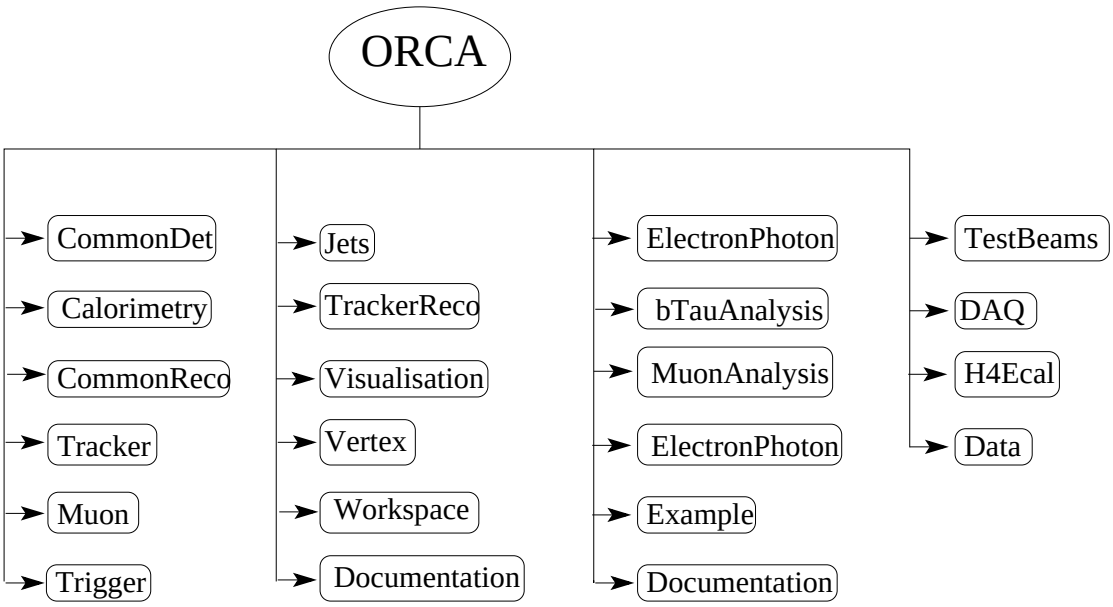

Figure 5.3 – La struture modulaire d'ORCA.

5. le module "Calorimetry" est dédié aux caractéristiques particulières des calorimètres hadronique et électromagnétique ainsi qu'à la reconstruction de traces et d'amas.
6. le module "Tracker" regroupe différents sous-modules dédiés aux caractéristiques particulières du trajectographe :
   (a) Les types de détecteurs :
   Un sous module décrit chaque type de détecteurs (détecteurs silicium à pixels et à micropistes). Les MSGC ne sont plus utilisées dans les dernières versions d'O.R.C.A..
   (b) Les générateurs :
   Ce sont des sous-modules permettant la génération de "hits"[(2)] à partir des fichiers de simulation, puis la génération de "digis"[(3)] à partir de ces "hits" et enfin la reconstruction des traces à partir de ces "digis".
   (c) Les interfaces :
   Des interfaces avec GEANT4 [GEA], CMSIM Magnetic Field [CMSIM] et CARF [CARF] ont aussi été implémentées sous forme de sous-modules.
7. le module "Muons" est dédié aux caractéristiques particulières des détecteurs de la chambre à muons.
8. le module "Trigger" décrit les conditions du déclenchement (de niveau 1) des calorimètres et des chambres à muons.

[2]Un "hit" correspond à la position reconstruite du passage d'une particule à travers un détecteur.
[3]Un "Digi" correspond au signal reçu par une piste à partir de données simulées.

9. le module "CommonReco" rassemble les classes génériques pour la reconstruction dans les chambres à muon et le trajectographe.
10. le module "TrackerReco" est dédié à la reconstruction des traces dans le trajectographe.
11. le module "Jets" est dédié à la recherche de jets à travers l'ensemble des détecteurs.
12. le module "Vertex" est dédié à la reconstruction des vertex.
13. le module "MuonReco" est dédié à la reconstruction de "hits" et de traces dans la chambre à muons.
14. le module "Visualisation" est dédié à la visualisation du détecteur et d'événements (simulés et reconstruits).
15. le module "ElectronPhoton" est dédié à la reconstruction d'électrons et de photons.
16. le module "bTauAnalysis" est dédié à la physique des B/τ.
17. le module "MuonAnalysis" est dédié à la physique des muons.
18. le module "HeavyIonAnalysis" est dédié à la physique des ions lourds.

Tous ces modules sont constitués de diverses classes C++ qui, une fois compilées, fournissent les librairies des classes utilisées lors de l'analyse des données.

5.3 Reconstruire les traces et les vertex au LHC

De façon simpliste, les informations extraites des données (réelles ou simulées) du détecteur CMS se regroupent ainsi :
- un ensemble de points obtenus dans les différents détecteurs de traces ;
- des gerbes développées dans les calorimètres ;
- un ensemble de points du passage de muons dans la chambre à muons.

A l'aide de ces différentes informations et à l'aide du logiciel O.R.C.A., on reconstruit les traces puis les vertex dont sont issues ces traces. L'analyse physique de l'événement reconstruit est alors possible.

5.3.1 Reconstruire au LHC

Lorsque le LHC sera à sa luminosité maximale, 25 interactions $p-p$ sont en moyenne attendues par croisement de faisceaux mais une seule de ces interactions est à l'origine du déclenchement et de la prise de données. Les autres interactions sont essentiellement constituées de collisions élastiques, diffractives, double diffractives et faiblement inélastiques ce qui complique considérablement la reconstruction des traces. En effet, même si l'on peut aisément supprimer, après reconstruction, les traces dont l'impulsion transverse est

inférieure à 1 GeV, rien ne peut nous aider à éliminer les "hits" correspondants avant la reconstruction des traces.

Les figures 5.4 et 5.5 illustrent la multiplicité des traces pour un événement $H \to \mu\mu\mu\mu$ présenté dans le plan transverse (perpendiculaire à l'axe du faisceau).

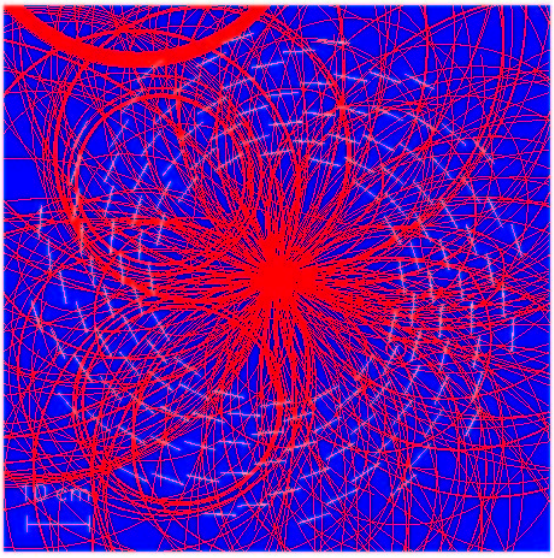

Figure 5.4 – avec les événements de pile-up superposés.

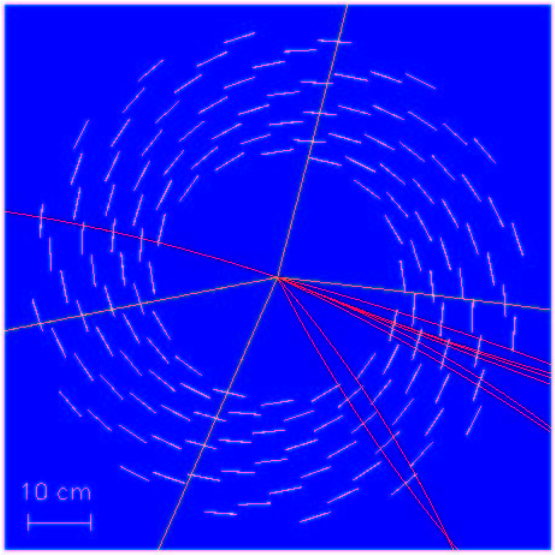

Figure 5.5 – sans les événements de pile-up superposés.

Figures 5.4 et 5.5 : Représentation dans le plan transverse (à l'axe du faisceau) de la multiplicité des traces lors d'une interaction produisant un $H \to \mu\mu\mu\mu$.

De plus, dans le cadre de la physique attendue au LHC, il est important de reconstruire de façon précise tous les vertex présents. Dans un premier temps, nous aurons à reconstruire le vertex primaire issu de l'interaction proton-proton puis les vertex secondaires plus éloignés de l'axe du faisceau. Ces deux types de vertex nécessitent des méthodes de reconstruction différentes. Les vertex primaires doivent être reconstruits de façon précise pour une bonne résolution sur le calcul de l'énergie transverse manquante. En effet sans une très bonne précision, la reconstruction des événements tels que $H \to \gamma\gamma$ où aucune trace ne pointe sur le vertex primaire (figure 5.6) est difficile.

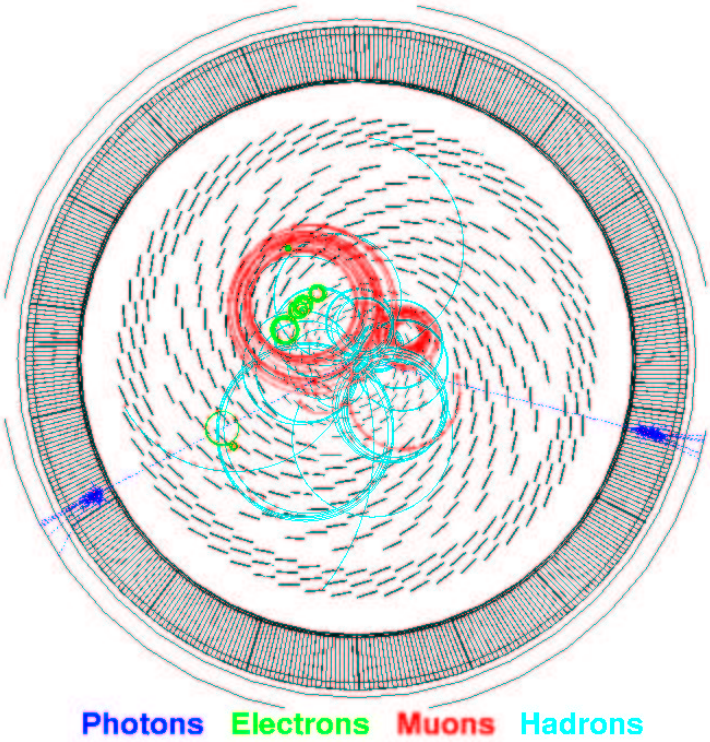

Figure 5.6 – Représentation dans le plan transverse du canal de désintégration du boson de Higgs en deux photons.

5.3.2 Reconstruire les traces

Le logiciel O.R.C.A. possède plusieurs méthodes de reconstruction de traces prenant en compte la diversité des événements et les besoins des utilisateurs.

Les différentes méthodes comporte deux tâches précises :

1. trouver la meilleure association de "hits" parmi les $4.10^3 - 5.10^4$ "hits" (par événement) répartis sur les différentes couches du trajectographe pour former une trajectoire dont voici quelques exemples de filtrage :
   - *Kalman Filter* : filtre classique pour la reconstruction des traces, détaillé ci-dessous (figure 5.7),
   - *GaussianSumFilter* : filtre où chaque paramètre des traces est décrit comme une somme de gaussiennes. Ce filtre est particulièrement bien optimisé pour des bruits non gaussien,
   - *DeterministicAnnealingFilter* : filtre itératif utilisant le *Kalman Filter*,
   - *MultiTrackFilter* : filtre particulièrement bien adapté aux jets de particules très denses.

2. reconstruire par lissage ("smoothing") la trace optimale correspondant aux "hits" précédemment associés.

En particulier, pour chaque "hit" des 3 couches de détecteurs silicium à pixels du trajectographe, on recherche tous les couples de "hits" compatibles (sur deux couches différentes). Ces couples constituent la base de la trajectoire.

Puis pour chaque couple de "hits", les paramètres de la trajectoire sont grossièrement estimés pour la recherche du "hit" de la couche suivante avec lequel il formera la trajectoire la plus compatible en impulsion et en position (figure 5.7).

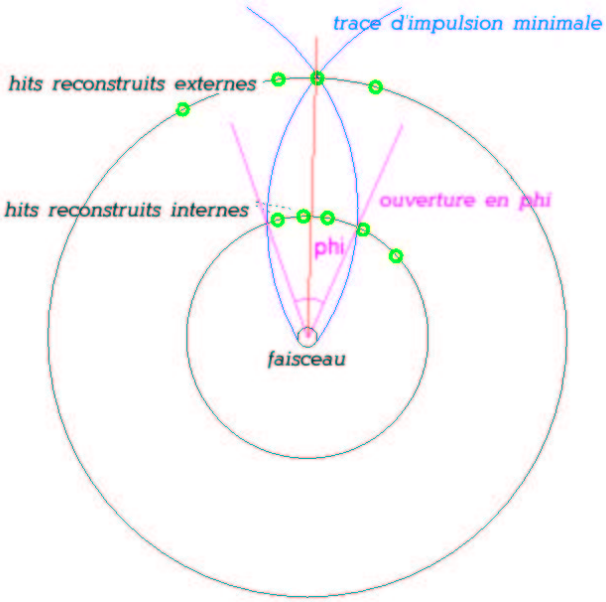

Figure 5.7 – Sélection des "hits" compatibles de la couche interne en fonction du "hit" de la couche externe et du faisceau.

On procède ainsi itérativement en propageant les paramètres de la trajectoire à la couche suivante tout en tenant compte du champ magnétique et de la perte d'énergie entre les couches de détecteurs. Chaque paramètre est réévalué en utilisant les informations relatives au nouvel "hit" trouvé et son estimation est alors de plus en plus précise au fur et à mesure que les couches sont analysées et les "hits" inclus à la trajectoire.

Ensuite, on procède dans le sens inverse des couches déjà analysées pour faire remonter l'information aux premières couches et ainsi améliorer l'estimation des paramètres sur chaque couche. La trace finale est l'ensemble des points moyens des deux trajectoires obtenues dans un sens et dans l'autre.

Enfin, les "hits" inutilisés sont récupérés pour essayer de compléter les traces partiellement reconstruites. De plus, toutes les traces d'impulsion inférieure à 1 GeV sont éliminées

(environ 70% des traces) comme le nombre important de "hits" non utilisés pour la reconstruction des traces sur les figures 5.8 et 5.9.

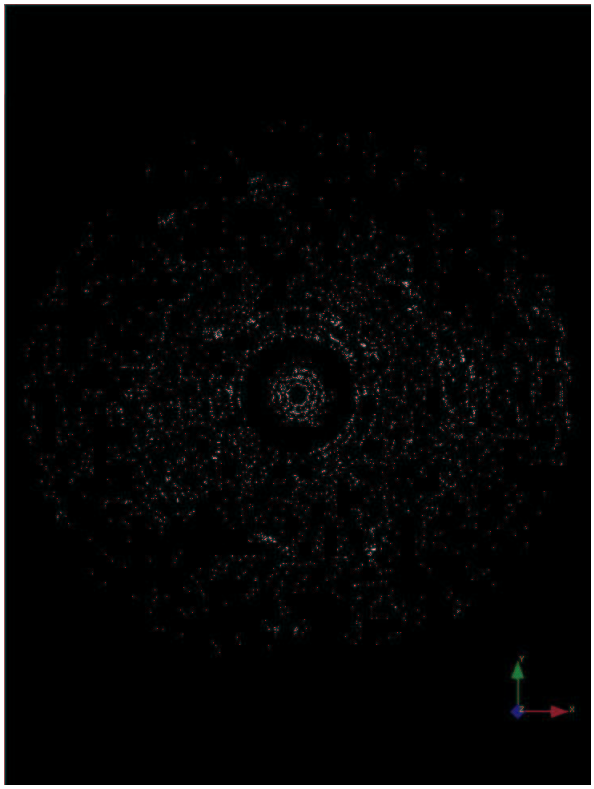

Figure 5.8 – Les 4209 "hits".

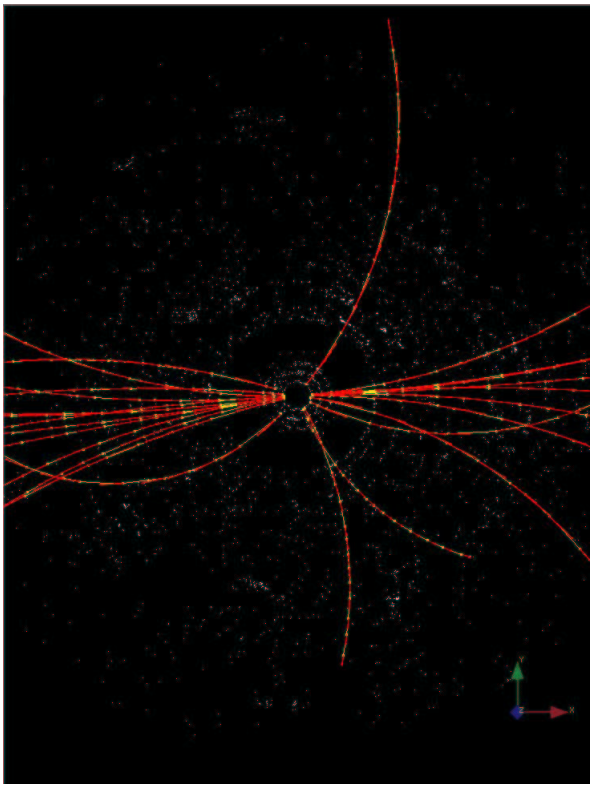

Figure 5.9 – Et les 25 traces reconstruites.

Figures 5.8 et 5.9 : Un événement $b\ \bar{b}$ de 50 GeV et avec une pseudo-rapidité $\eta < 1,4$ (sans empilement).

Après avoir obtenu l'ensemble des traces composant un événement, l'étape suivante est la reconstruction des vertex.

5.3.3 Reconstruire les vertex primaires

Pour reconstruire les vertex primaires, le plus simple est de sélectionner les traces en fonction de leur paramètre d'impact. Le paramètre d'impact correspond à la distance minimale entre la trace reconstruite et l'axe du faisceau (figure 5.10).

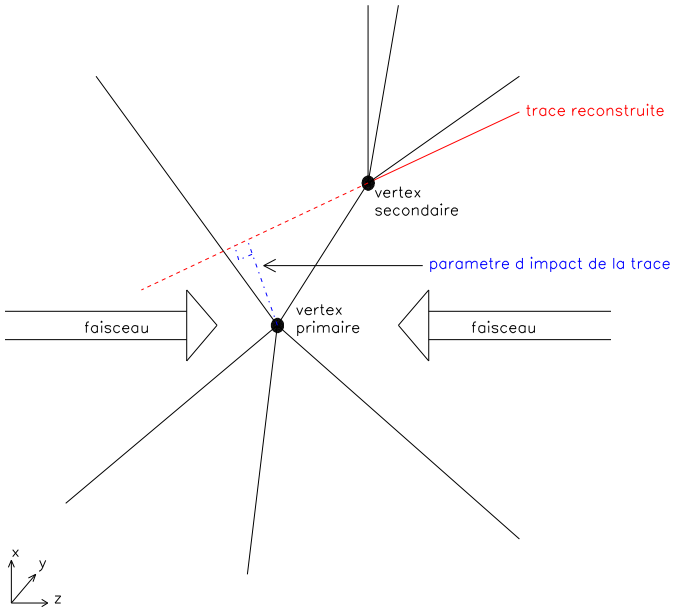

Figure 5.10 – Définition du paramètre d'impact et orientation du repère (x, y, z) par rapport au faisceau.

On ne conservera donc que les traces dont le paramètre d'impact dans le plan transverse[(4)] n'excède pas une valeur seuil de 5 σ [ALB01a] (où σ est l'erreur sur la position de la trace par rapport à l'axe du point d'impact). Toutefois on ne pourra jamais exclure toutes les traces provenant de vertex secondaires. De plus, on supposera que les coordonnées transverses (x, y) des vertex reconstruits sont nulles[5] (figure 5.10) et on se concentrera uniquement sur la reconstruction de la coordonnée z des vertex (z est la coordonnée qui repère la position le long de l'axe du faisceau). Trois méthodes de reconstruction de vertex primaires sont décrites ci-dessous [ALB01a].

5.3.4 Le type d'événements étudiés

L'algorithme EA a été utilisé pour l'analyse d'événements simulés disponibles dans la base de données O.R.C.A. Ces événements ont été générés par PYTHIA [PYT] pour différentes coupures en pseudorapidité η et en énergie E_t des partons à l'origine de l'événement :

[4]Le plan transverse est le plan (x, y), le plan perpendiculaire à l'axe du faisceau.
[5]On considère que la taille du faisceau est telle que $\sigma_x = 15\ \mu m$ et $\sigma_y = 15\ \mu m$.

- 2500 événements $q\bar{q}$ avec $E_t = 50$ GeV, avec $\eta < 1,4$ et sans empilement (q représente un quark léger) ;
- 150 événements $q\bar{q}$ avec $E_t = 50$ GeV, avec $\eta < 1,4$ et avec empilement (q représente un quark léger) ;
- 2500 événements $b\bar{b}$ avec $E_t = 50$ GeV, avec $\eta < 1,4$ et sans empilement ;
- 2500 événements $b\bar{b}$ avec $E_t = 100$ GeV, avec $\eta < 1,4$ et sans empilement ;
- 2500 événements $b\bar{b}$ avec $E_t = 100$ GeV, avec $1,4 < \eta < 2,4$ et sans empilement.

5.3.4.1 Reconstruction par la méthode de "binning"

La reconstruction de vertex par la méthode de "binning" repose sur un principe simple : la zone d'interaction située entre -25 cm et +25 cm est divisée en sections de 1 mm. L'association de toutes les traces dont la coordonnée z du paramètre d'impact est située dans un même intervalle de segmentation permet d'obtenir l'ensemble des traces issues d'un même vertex (figure 5.11).

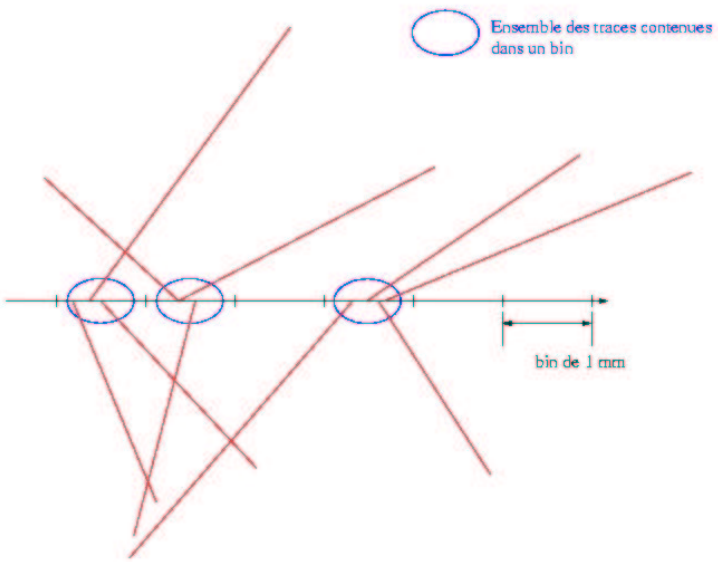

Figure 5.11 – Illustration de la sélection des traces par la méthode de "binning".

Puis une méthode barycentrique calcule, pour chaque ensemble de traces, la position du vertex et son erreur en utilisant la composante z des points d'impact des traces. Les poids utilisés pour ce calcul barycentrique sont définis comme l'inverse des erreurs sur la position en z du point d'impact des traces :

$$Z_{vertex} = \frac{1}{\Sigma_i 1/\sigma_i} \Sigma_i \frac{z_i}{\sigma_i} \tag{5.1}$$

Enfin on recherche si deux vertex proches ne constituent pas un seul et même vertex.

Pour évaluer les performances de la reconstruction des vertex primaires, on calcule les deux paramètres suivants :

- la résolution :
  On associe chaque vertex primaire reconstruit au vertex simulé le plus proche et on calcule leur différence en z ($\Delta z = z_{rec} - z_{sim}$). La résolution est définie comme l'écart-type de la distribution des Δz (figure 5.12).
- l'efficacité à 5 σ :

$$\text{efficacité} = \frac{\text{Nombre de vertex primaire reconstruit à moins de 5 } \sigma \text{ du vertex primaire simulé}}{\text{Nombre total de vertex primaire reconstruit}}$$

  où σ désigne la résolution.

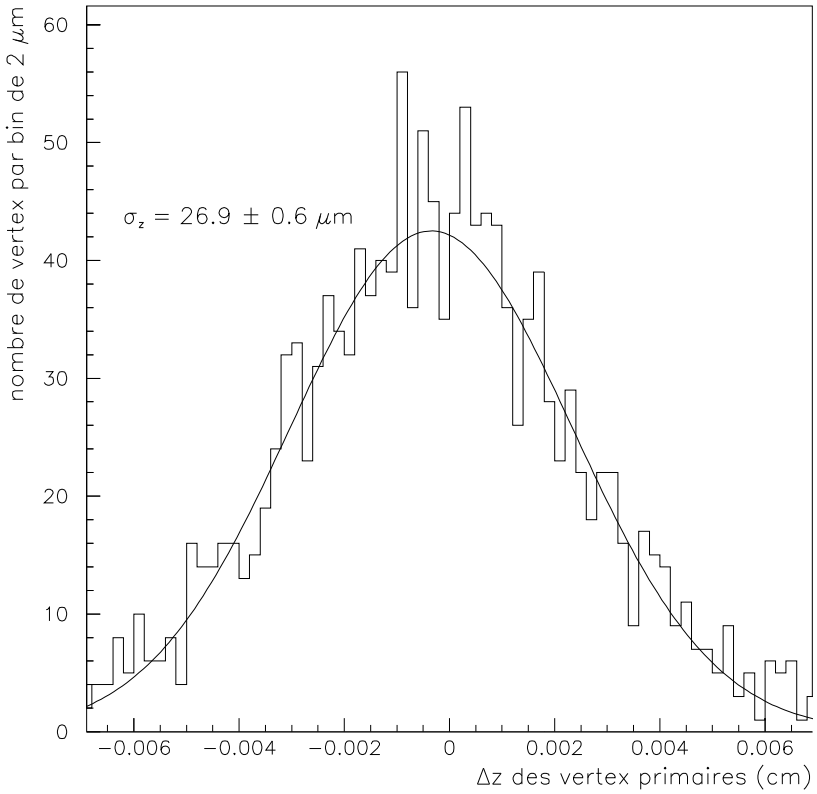

Figure 5.12 – Calcul de la résolution par la méthode de "binning" pour 2205 vertex primaires provenant d'événements $q\ \bar{q}$.

La résolution obtenue pour les événements $q\ \bar{q}$ et pour les événements $b\ \bar{b}$ n'est pas

comparable à cause de la différence de multiplicité des traces pour chaque type d'événement mais fournit tout de même un ordre de grandeur. Pour les événements $b\ \bar{b}$, la résolution obtenue pour les événements avec $E_t = 100$ GeV est meilleure que pour les événements avec $E_t = 50$ GeV pour la même raison.

Type d'événements	$q\ \bar{q}$ 50 GeV $\eta < 1,4$	$b\ \bar{b}$ 50 GeV $\eta < 1,4$	$b\ \bar{b}$ 100 GeV $\eta < 1,4$	$b\ \bar{b}$ 100 GeV $\eta > 1,4$
Nombre de vertex reconstruits	2205	2130	1184	402
Résolution en z (μm)	$29,9 \pm 0,6$	$75,8 \pm 2,1$	$61,9 \pm 2,2$	$116,4 \pm 8,6$
Efficacité à 5 σ (%)	91,8	99,9	99,8	92,8

Tableau 5.3 – Résolution obtenue pour différents types d'événements.

Sur un ensemble de $\sim$ 2500 événements $q\ \bar{q}$ et $b\ \bar{b}$ (paragraphe 5.3.4), les résolutions obtenues par la méthode de "binning" sont comprises entre 26 et 120 μm, les efficacités à 5 σ sont comprises entre 91.8% et 99.9% (tableau 5.3), ce qui est assez précis pour une méthode aussi simple et aussi rapide.

Mais on obtient parfois plus d'un vertex primaire reconstruit par événement [ALB01a]. L'explication de cet excès de vertex provient de la segmentation : les traces appartenant à un même vertex se répartissent sur deux intervalles consécutifs, ce qui entraîne la création de deux vertex reconstruits. Pour pallier ce problème directement dans la méthode de reconstruction, une méthode de création des amas a été conçue.

5.3.4.2 Reconstruction par la méthode des amas

Cette méthode ordonne les traces en fonction de la coordonnée z de leur paramètre d'impact. La distance entre cette trace et la trace suivante est testée à l'aide du calcul de la valeur du χ^2 : tant que ce χ^2 est inférieur à 5 [ALB01a], nous ajoutons la nouvelle trace testée à l'ensemble des traces qui serviront à la reconstruction du vertex primaire. La méthode utilisée pour reconstruire la position du vertex est la même que dans le cas de la méthode du binning.

Type d'événements	$q\,\bar{q}$ 50 GeV $\eta < 1,4$	$b\,\bar{b}$ 50 GeV $\eta < 1,4$	$b\,\bar{b}$ 100 GeV $\eta < 1,4$	$b\,\bar{b}$ 100 GeV $\eta > 1,4$
Nombre de vertex reconstruit	2455	2208	1244	407
Résolution en z (μm)	$26,9 \pm 0,6$	$47,4 \pm 3,3$	$53,7 \pm 3,9$	$111,0 \pm 14,7$
Efficacité à 5 σ (%)	96,6	99,4	99,7	99,5

Tableau 5.4 – Résolution obtenue pour différents types d'événements.

Les résolutions et les efficacités obtenues avec cette méthode (tableau 5.4) sont légèrement meilleures à celles obtenues par la méthode de "binning".

5.3.4.3 Reconstruction par la méthode des gaussiennes

Cette méthode modélise les posistions des traces par des gaussiennes dont la valeur moyenne est leur position z (celle de la coordonnée z de leur paramètre d'impact) et dont l'écart-type est l'erreur sur cette position.

Les maximums de la fonction "somme des gaussiennes" de toutes les traces (figure 5.13) donnent la position z des vertex associés. Dans le cas de deux vertex distants de moins de $10\mu m$, ceux-ci sont fusionnés en un seul vertex. Remarquons que toute trace à plus de 6 σ d'un vertex ne contribue plus à la fonction "somme des gaussiennes". De plus, la présence de maxima locaux de la fonction "somme des gaussiennes" (figure 5.13) peut conduire à des vertex primaires excédentaires.

Type d'événements	$q\,\bar{q}$ 50 GeV $\eta < 1,4$	$b\,\bar{b}$ 50 GeV $\eta < 1,4$	$b\,\bar{b}$ 100 GeV $\eta < 1,4$	$b\,\bar{b}$ 100 GeV $\eta > 1,4$
Nombre de vertex reconstruits	1953	1963	1108	373
Résolution en z (μm)	$23,1 \pm 0,5$	$40,2 \pm 1,2$	$30,3 \pm 1,6$	$63,4 \pm$ 9,4
Efficacité à 5 σ (%)	96.1	96.8	96.3	97.1

Tableau 5.5 – Résolution obtenue pour différents types d'événements.

Les résolutions obtenues avec cette méthode (tableau 5.5) sont bien meilleures que pour les deux autres méthodes mais en contrepartie l'efficacité est plus faible.

En comparant les performances de ces trois méthodes, la méthode des amas sera utilisée dans la suite de cette étude pour la reconstruction de vertex primaires.

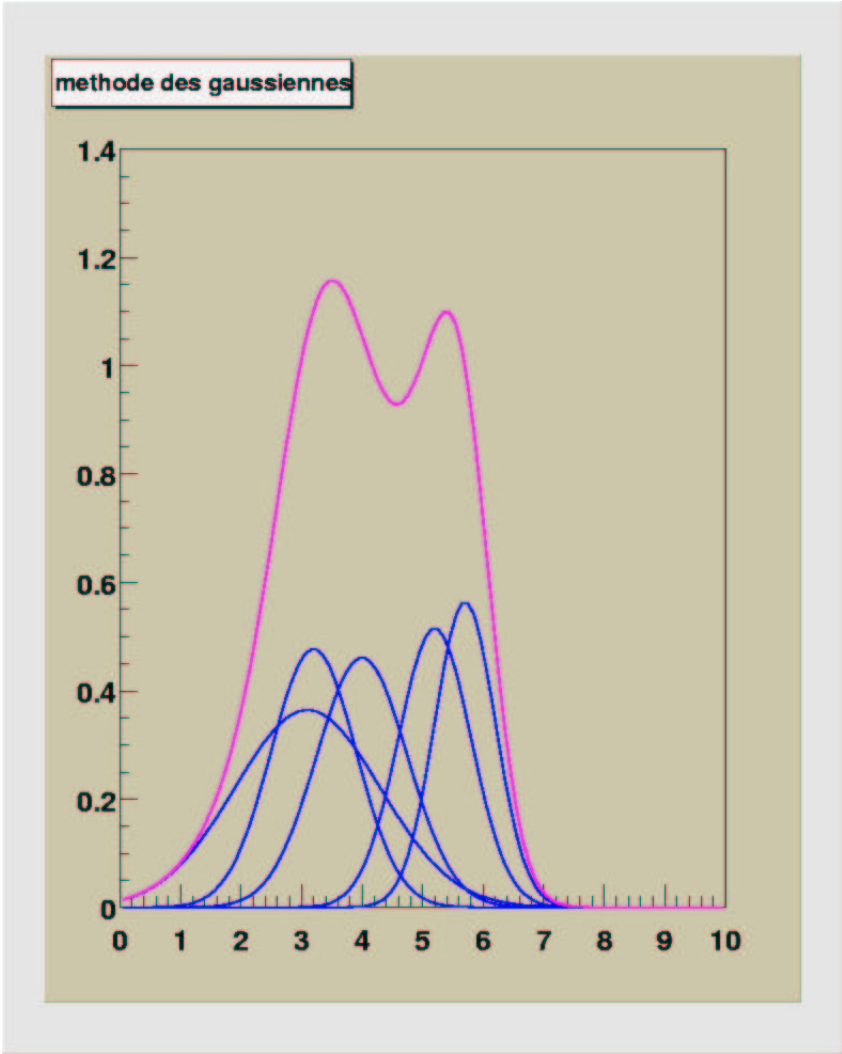

Figure 5.13 – Illustration de la reconstruction par la méthode des gaussiennes.

5.3.4.4 Reconstruction de vertex primaires et l'empilement d'événements

Une étude de reconstruction de vertex primaire par la méthode de "binning" (partie 5.3.4.1) a été faite pour 150 événements $q\ \bar{q}$ avec empilement d'événements. Dans notre cas, on définit l'événement "intéressant" comme l'événement qui a été sélectionné par le système de déclenchement et qui entraîne l'acquisition. Cet empilement d'événements sur l'événement "intéressant" (partie 5.3.1) apporte 15 fois plus de traces, mais la plupart de ces traces sont de faible impulsion. On peut alors distinguer facilement le vertex primaire associé à l'événement "intéressant" en choisissant le vertex primaire reconstruit ayant à

la fois la plus grande impulsion transverse [6] et le plus grand nombre de traces associées (figures 5.14 et 5.15).

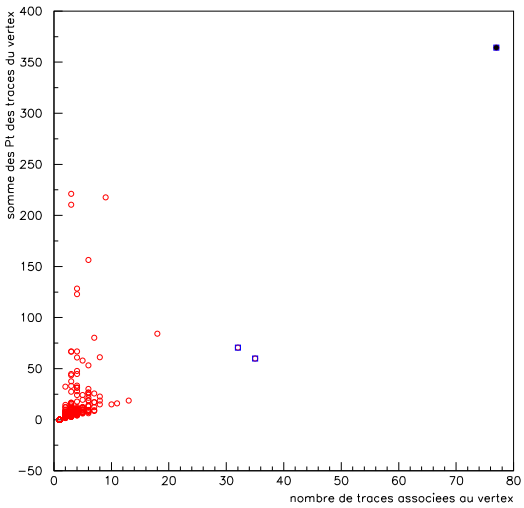

Figure 5.14 – Le nombre de traces associées à un vertex primaire en fonction de la somme des impulsions transverses des traces associées à ce vertex.

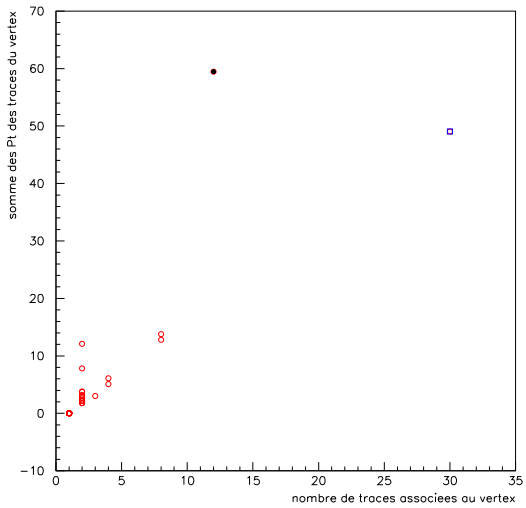

Figure 5.15 – Le nombre de traces associées à un vertex primaire en fonction de la somme des impulsions transverses des traces associées à ce vertex.

On se rend compte que ces deux conditions sont nécessaires car on reconstruit parfois des vertex primaires associés à un grand nombre de traces mais avec une impulsion transverse de quelques GeV au lieu des quelques dizaines de GeV attendus mais parfois le choix est plus complexe. Sur la figure 5.15, on observe deux vertex associés à un grand nombre de traces :

- un vertex associé à 30 traces et avec une impulsion transverse de $\sim$ 49,1 GeV, ce qui équivaut à 1,6 GeV/trace ;
- un vertex associé à 12 traces et avec une impulsion transverse de $\sim$ 59,5 GeV, ce qui équivaut à 5 GeV/trace.

Le vertex primaire de l'événement "intéressant" est le deuxième vertex car il est associé à des traces plus énergétiques.

Le fait de bien reconstruire la coordonnée z des vertex primaires est relativement simple. Mais reconstruire les trois coordonnées des vertex secondaires (associés à un nombre plus faible de traces) demande un algorithme de reconstruction plus complexe.

[6] L'impulsion transverse d'un vertex est la somme des impulsions transverses des traces qui lui sont associées.

5.3.5 Reconstruction de vertex secondaires par la méthode dite "Elastic Arms"

Le principe de cet algorithme appelé "Elastic Arms" (EA), est de reconstruire les vertex secondaires principalement dans le but de caractériser les jets issus de la fragmentation de quarks b. Cet algorithme se base sur le calcul d'un potentiel d'attraction des traces sur les vertex.

5.3.5.1 Généralités

L'enjeu de la reconstruction des vertex secondaires, déplacés par rapport à l'axe du faisceau, est à la fois d'associer les traces reconstruites d'un événement à un ensemble de $n+1$ vertex[7] et de positionner précisément ces vertex. Par exemple, pour un événement contenant une trentaine de traces et en considérant quatre vertex, le nombre de combinaisons possibles est de plus d'un milliard. Il est donc clair que nous n'avons pas le temps d'essayer toutes les combinaisons possibles et de comparer leur χ^2 (le χ^2 le plus faible donnant la meilleure combinaison possible).

Une autre solution consiste à associer chaque vertex à une paire de traces puis de calculer le χ^2 de chaque autre trace pour savoir avec quel vertex celle-ci doit être associée. Chaque décision d'association est alors prise localement (une trace à la fois) et on risque de trouver des solutions locales et non globales (figure 5.16).

Il y a fort longtemps, les alchimistes ont résolu ce problème en découvrant une façon intelligente d'éviter les minimums locaux. Leur métal devenait très cassant lorsqu'il refroidissait trop vite, mais un lent refroidissement (figure 5.17) laissait aux molécules le temps de s'installer niveau d'énergie global le plus bas (système pouvant être décrit par une distribution de Boltzman), ce qui a contribué à avoir une structure plus régulière et donc plus résistante. Cette technique est appelée "*Elastic Arms*" (EA) à cause de la relation d'association variable ("elastic") entre les vertex et les traces ("arms").

[7] où n est le nombre de vertex secondaires

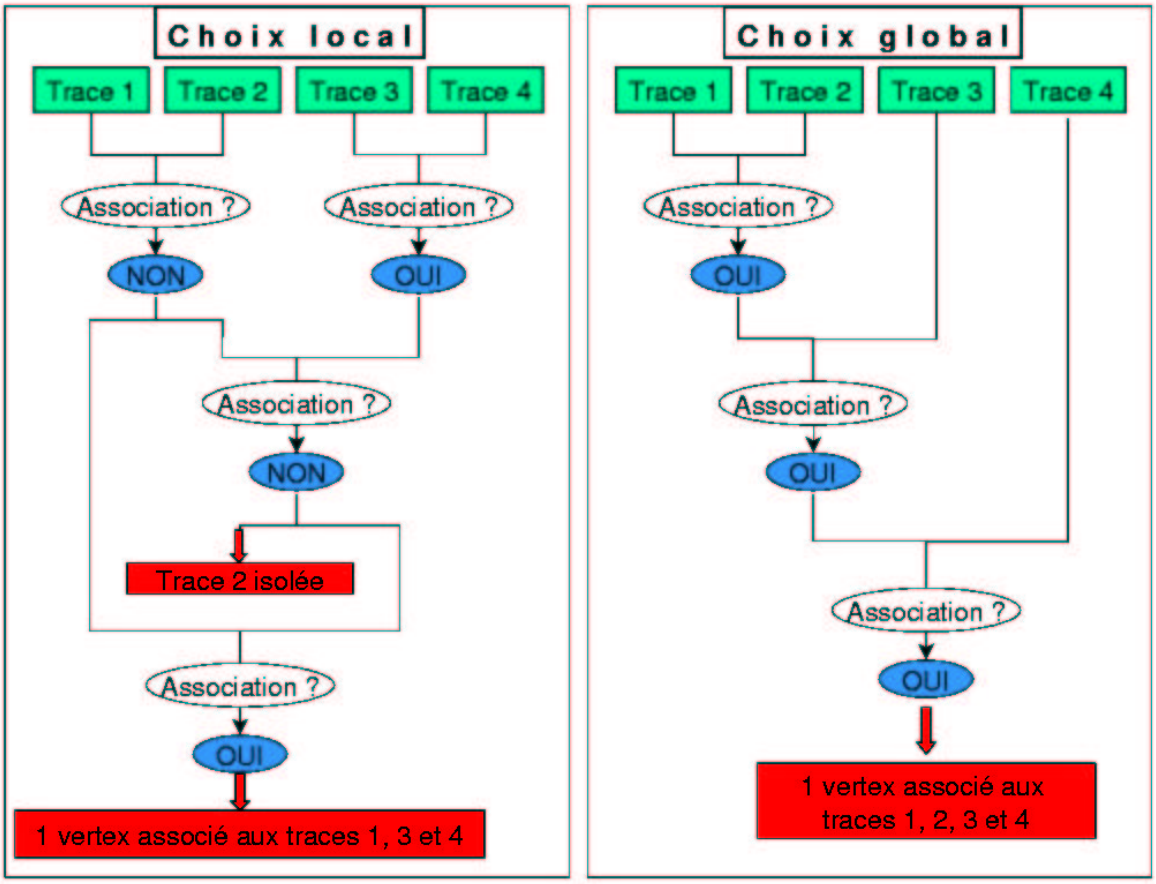

Figure 5.16 – A partir de 4 traces simulées provenant du même vertex primaire, on cherche à associer les traces reconstruites à un même vertex reconstruit. On remarque que le vertex reconstruit par la méthode locale (à gauche) n'est associé qu'á trois traces alors que le vertex reconstruit par la méthode globale (à droite) est bien associé aux quatre traces.

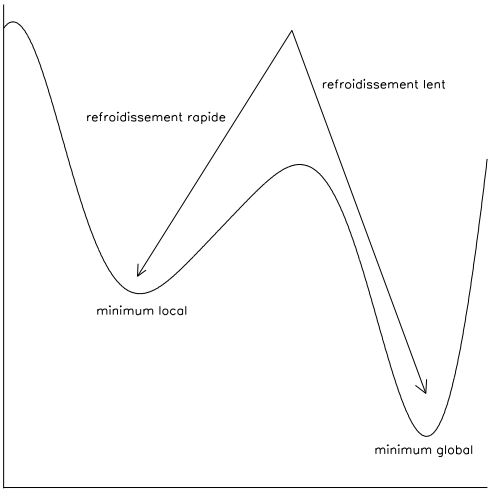

Figure 5.17 – Un refroidissement rapide conduit à un niveau d'énergie plus faible mais qui est minimum local alors qu'un refroidissement plus lent conduit au niveau d'énergie minimum qui est le minimum global.

Une façon d'appliquer cette idée d'EA à la recherche de vertex secondaires est de choisir une distribution de Boltzman en χ^2, contrairement à une simple coupure fixée en χ^2. En d'autres termes, de tester chaque solution possible à l'aide d'une série de solutions en requérant un χ^2 de reconstruction minimum et en abaissant petit à petit la température (celle de la distribution de Boltzman) jusqu'à un seuil. Ce χ^2 peut être considéré comme une énergie ; cette énergie est d'autant plus importante que les traces sont éloignées des vertex auxquels elles sont associées. Le but est donc de minimiser cette énergie. A l'aide d'itérations successives, ces multiples vertex reconstruits vont progressivement se déplacer vers leur véritable position (x, y, z) et lorsque la température du système s'est suffisamment abaissée, les paramètres devraient être proches des paramètres simulés.

L'algorithme EA a été développé au début des années 90 dans la cadre de l'expérience DELPHI par Ohlsson, Peterson et Yuille [OHL92] pour la reconstruction des traces en associant correctement les "hits" aux traces.

En s'appuyant sur ces idées, l'algorithme EA a pour objectif de trouver à la fois le nombre exact de vertex et de reconstruire précisément leur position à partir des traces reconstruites.

5.3.5.2 Description

Une démarche classique pour trouver plus facilement les vertex secondaires est de limiter la recherche en sélectionnant les traces qui passent loin du vertex primaire en établissant une coupure sur le paramètre d'impact. En réalité, l'idéal serait de reconstruire le vertex primaire avec les traces de faible paramètre d'impact (à moins de 5 σ, paragraphe 5.3.3) puis de reconstruire le vertex secondaire uniquement avec les traces incompatibles avec le vertex primaire. Dans notre cas, parmi toutes les traces reconstruites, seules les traces dont le paramètre d'impact dans le plan transverse (au faisceau) est supérieur à 3σ seront conservées (où σ est l'erreur sur la position par rapport à l'axe du point d'impact). Ce choix de coupure est justifié au paragraphe 5.3.5.3.1. Ces traces sélectionnées permettent de déterminer la position de l'ensemble des vertex candidats comme l'intersection de 2 traces sélectionnées. Chacun de ces vertex candidats est nécessaire comme point de départ à l'algorithme EA et est appelé "vertex seed". On conservera uniquement les "vertex seed" si leur χ^2 est inférieur à 20, sachant que le χ^2 est d'autant plus faible que la reconstruction est précise.

Ensuite, pour chaque itération en température T et pour chaque couple *(trace i, "vertex seed" j)*, l'agorithme itératif EA détermine le potentiel d'attraction $V_{i,j}$ (une distribution de Boltzman en χ^2) défini par

$$V_{i,j} = \frac{e^{-\chi^2_{i,j}/T}}{e^{-\Lambda/T} + \Sigma_j e^{-\chi^2_{i,j}/T}} \tag{5.2}$$

- où $\chi^2_{i,j} = P\ E^{-1}\ {}^TP$, avec E la matrice d'erreur en position de la trace i et le vecteur P la distance entre la trace i et le "vertex seed" j (${}^tP$ est le transposé du vecteur P) ;
- où $\Lambda = 4$ et permet le calcul du potentiel $V_{i,j}$ en tenant compte des traces isolées ;
- et où T est sans unité et $T = 200/k$ avec k l'indice itératif variant entre 1 et 50 par pas de 1. On remarque que la température T baisse fortement au départ (elle passe de 200 à 50 en 4 itérations) éliminant ainsi rapidement les "vertex seed" mal placés puis elle baisse plus progressivement pour laisser aux autres vertex trouver "naturellement" leur position.

En observant l'équation 5.2, on peut vérifier que plus les traces sont éloignées, plus le $\chi^2_{i,j}$ est important et moins le potentiel d'attraction $V_{i,j}$ est important. Mais ce potentiel varie assez peu en fonction de l'itération en température T.

L'objectif de ce potentiel d'attraction $V_{i,j}$ est de déterminer la valeur du déplacement de chaque "vertex seed". En effet, pour chaque température T et pour chaque trace i, les "vertex seed" j se déplacent de (Δx_j, Δy_j, Δz_j) définis par :

$$(\Delta x_j, \Delta y_j, \Delta z_j) = -\eta \times \Sigma_j V_{i,j} \times (\partial d_{i,j}/\partial x_j, \partial d_{i,j}/\partial y_j, \partial d_{i,j}/\partial z_j) \tag{5.3}$$

- où $\eta = 0,5 \times 10^{-4}/k^{0,8}$ détermine l'ordre de grandeur du déplacement ;
- et où les dérivées $(\partial d_{i,j}/\partial x_j, \partial d_{i,j}/\partial y_j, \partial d_{i,j}/\partial z_j) = (\partial \chi^2_{i,j}/\partial x_j, \partial \chi^2_{i,j}/\partial y_j, \partial \chi^2_{i,j}/\partial z_j)$ sont calculées numériquement par pas de 1 μm sur les axes x, y et z.

Toutes les traces i contribuent donc à déterminer la position de ces "vertex seed" j. De plus, on remarque que plus l'itération k augmente, plus la température T diminue ($T = 200/k$) et moins les "vertex seed" peuvent se déplacer. On dit que le système des "vertex seed" se refroidit peu à peu. La fonction de minimisation d'énergie est alors $E = \frac{1}{\beta}\Sigma_i log(e^{-\beta\Lambda+\sigma_j e-\beta\chi^2_{i,j}})$.

L'algorithme EA possède aussi une fonction de fusion des vertex très proches : si 2 "vertex seed" sont distants de moins de 100 μm, on ne garde qu'un seul "vertex seed". Ce processus de fusion est effectué à la fin de chaque itération en température.

L'algorithme EA arrête les itérations en température pour trois types de cas :
- Cas 1 : tous les "vertex seed" convergent en position (déplacement entre 2 itérations en température inférieur à 20 μm) après au moins 3 itérations en température,
- Cas 2 : tous les "vertex seed" divergent en position (déplacement entre 2 itérations en température supérieur à 500 μm) après au moins 3 itérations en température,
- dans tous les autres cas : arrêt à la 50eme itération.

A la fin des itérations en température, chaque trace est associée au "vertex seed" le plus proche. Notons que les "vertex seed" associés à moins de 2 traces sont éliminés. Puis lorsque l'on a identifié toutes les traces provenant d'un même vertex, on utilise la *méthode de linéarisation* (la classe *LinearVertexFitter*). Cette méthode itérative est basée sur l'approximation des traces par des droites (au voisinage du vertex), et le point où la trace est tangente à cette droite est appelé point de linéarisation. La position du vertex est

déterminé par l'intersection de ces droites (figure 5.18). Cette méthode très rapide fournit une estimation très précise de la position du vertex, avec une erreur de moins de 1 μm (si la distance entre le point de linéarisation et le vertex est de moins de 1 mm) [KAR97].

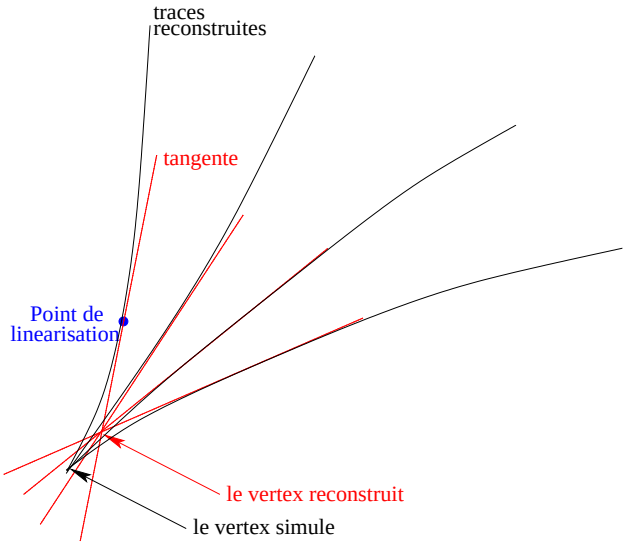

Figure 5.18 – Illustration de la méthode de linéarisation : les trajectoires sont approximées par des droites dont l'intersection fournit la position du vertex.

Cette méthode nous permet d'obtenir la position finale et le nombre final des vertex reconstruits par l'algorithme EA.

5.3.5.3 Choix des paramètres

Le but de l'algorithme EA est de faire converger la position des "vertex seed" vers la position des vertex simulés en un nombre minimum d'itérations en température. Les différents paramètres permettant d'améliorer les performances de l'algorithme EA pour un maximum d'événements sont :

- le paramètre η (équation 5.3) qui conditionne l'ordre de grandeur du déplacement en position des "vertex seed" ; il doit être suffisamment grand pour une convergence rapide tout en évitant la divergence en position des "vertex seed" ;
- le paramètre $\beta = \frac{1}{T}$ (équation 5.2) qui détermine la valeur du potentiel d'attraction entre un "vertex seed" et une trace ;
- le paramètre Λ (équation 5.2) qui doit être de l'ordre de grandeur du χ^2 pour une trace éloignée de l'ensemble des "vertex seed" et des autres traces ;
- le pas pour le calcul de la dérivée qui doit être assez petit pour un calcul précis ; dans notre cas, nous avons choisi un pas de 1 μm pour chaque axe x, y et z ;

– la distance limite de fusion entre deux vertex qui doit être harmonisée en fonction du déplacement moyen des "vertex seed" entre deux itérations en température.

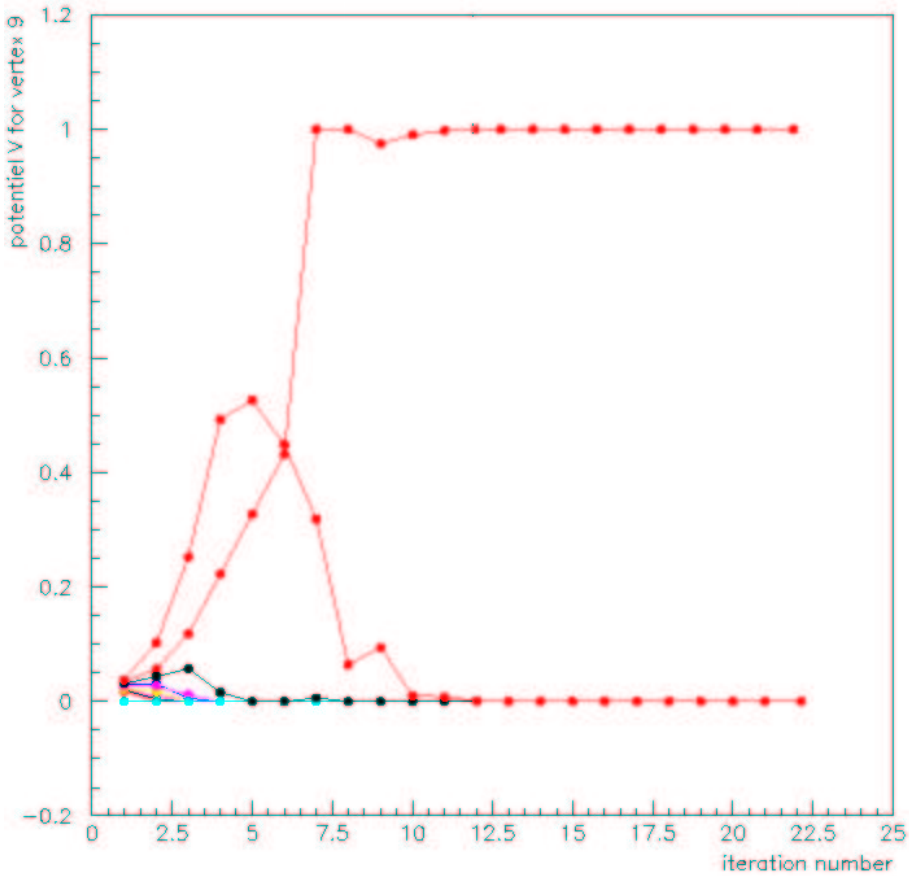

Figure 5.19 – Variation du potentiel d'attraction (équation 5.2) de chaque trace sur un "vertex seed" j en fonction des itérations en température.

Avant tout, le paramètre $\beta = \frac{1}{T}$ doit être relié à la distance entre le "vertex seed" j considéré et les traces. La figure 5.19 montre un vertex d'abord principalement attiré par 2 traces puis par une seule trace au fur et à mesure des déplacements de ce "vertex seed" à chaque nouvelle itération en température. L'attraction des autres traces reste négligeable pour ce "vertex seed" j.

Il faut de plus évaluer le paramètre η pour un déplacement de la position de chaque "vertex seed" qui ne soit ni trop grand, ni trop faible. Un déplacement trop important entraînerait une instabilité en position des "vertex seed" (figure 5.20). Ce phénomène de divergence est amplifié pour les "vertex seed" proches d'un grand nombre de traces qui attirent toutes ces "vertex seed". Et une valeur de η trop faible empêcherait les "vertex seed" de converger en une position unique (figure 5.21).

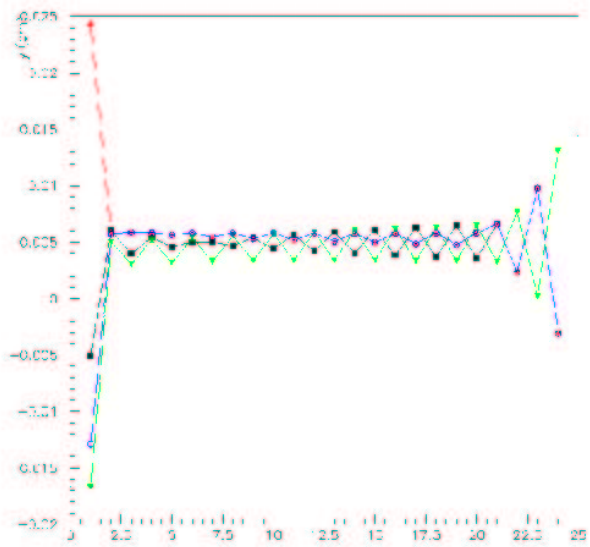

Figure 5.20 – Variation de la position y (en cm) des "vertex seed" (une couleur par "vertex seed") en fonction des itérations en température avec un paramètre η trop important.

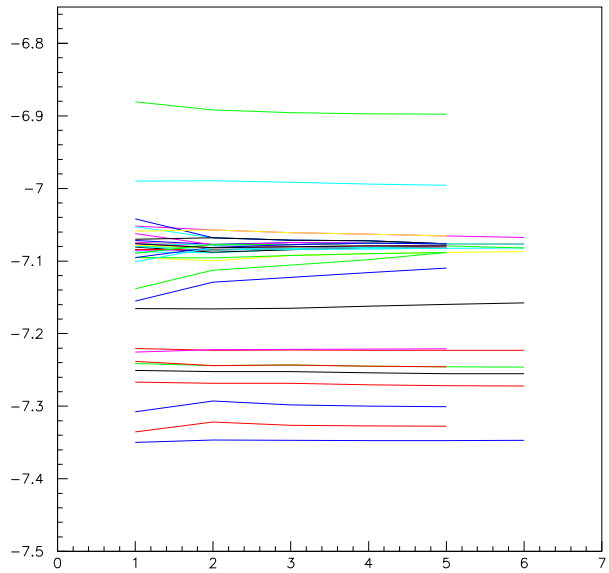

Figure 5.21 – Variation de la position z (en cm) des "vertex seed" en fonction des itérations en température avec un paramètre η trop faible.

Pour les traces (i) isolées, la valeur du $\chi^2_{i,j}$ avec les "vertex seed" j est faible et on a alors $\frac{e^{-\chi^2_{i,j}/T}}{\Sigma_j e^{-\chi^2_{i,j}/T}} \simeq \frac{1}{\text{Nombre de traces}}$. En revanche, grâce à l'introduction du terme $e^{-\Lambda/T}$ (avec $\Lambda = 4$), ces traces isolées n'attirent plus autant les "vertex seed". Néanmoins, ce terme ralentit la convergence en position de certains "vertex seed" éloignés (figure 5.22) et il apparaît nécessaire de redéterminer les paramètres η et β.

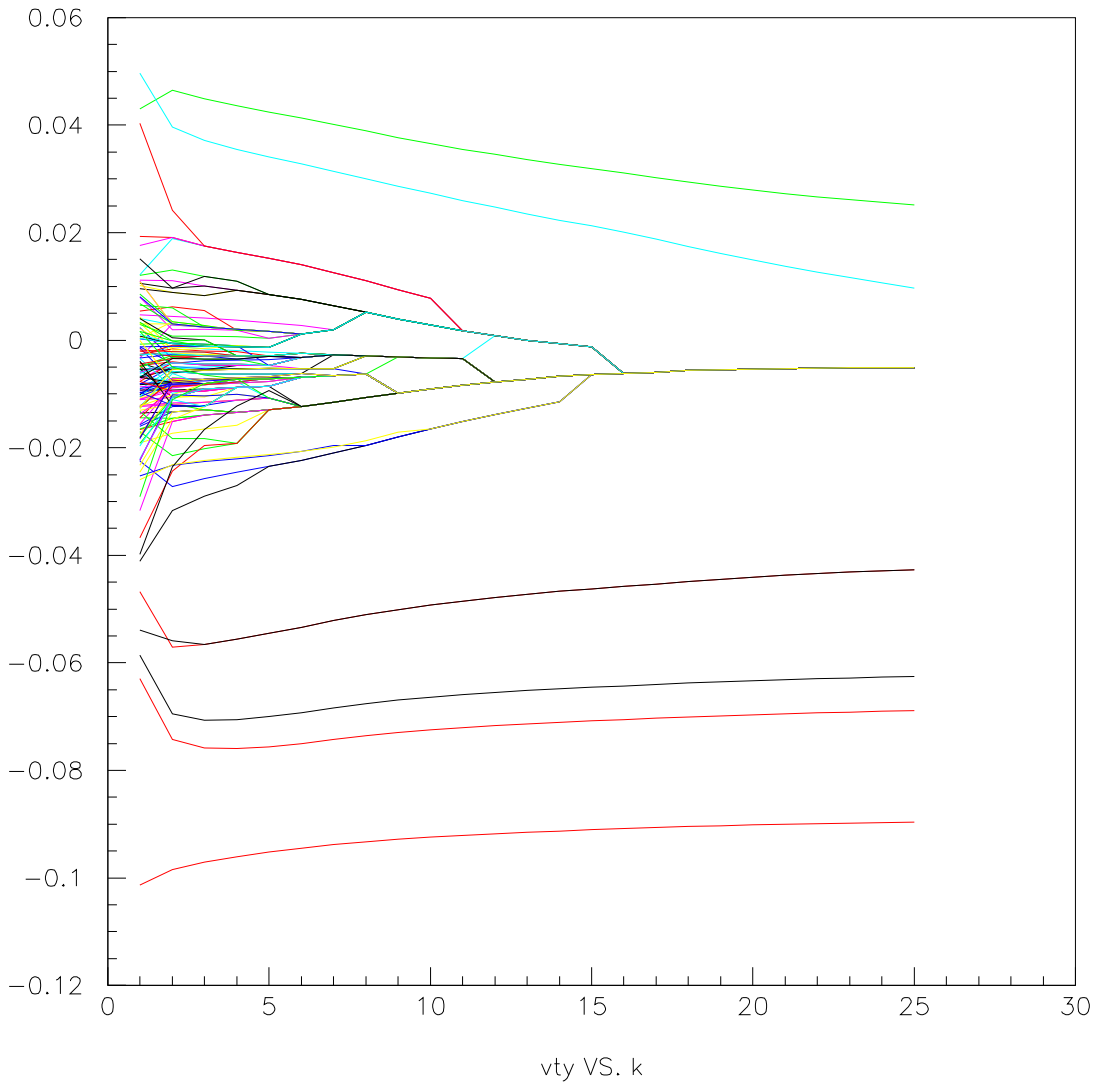

Figure 5.22 – Variation de la position y (en cm) des "vertex seed" en fonction des itérations en température après l'introduction du paramètre Λ.

De plus, pour certains événements, les "vertex seed" divergent spontanément en position (figure 5.23). Dans ce cas, soit on réévalue les paramètres η et β pour cet événement spécifique, soit on fixe une valeur maximale au nombre d'itérations en température (dans notre cas, nous choisissons un nombre d'itération maximale de 50).

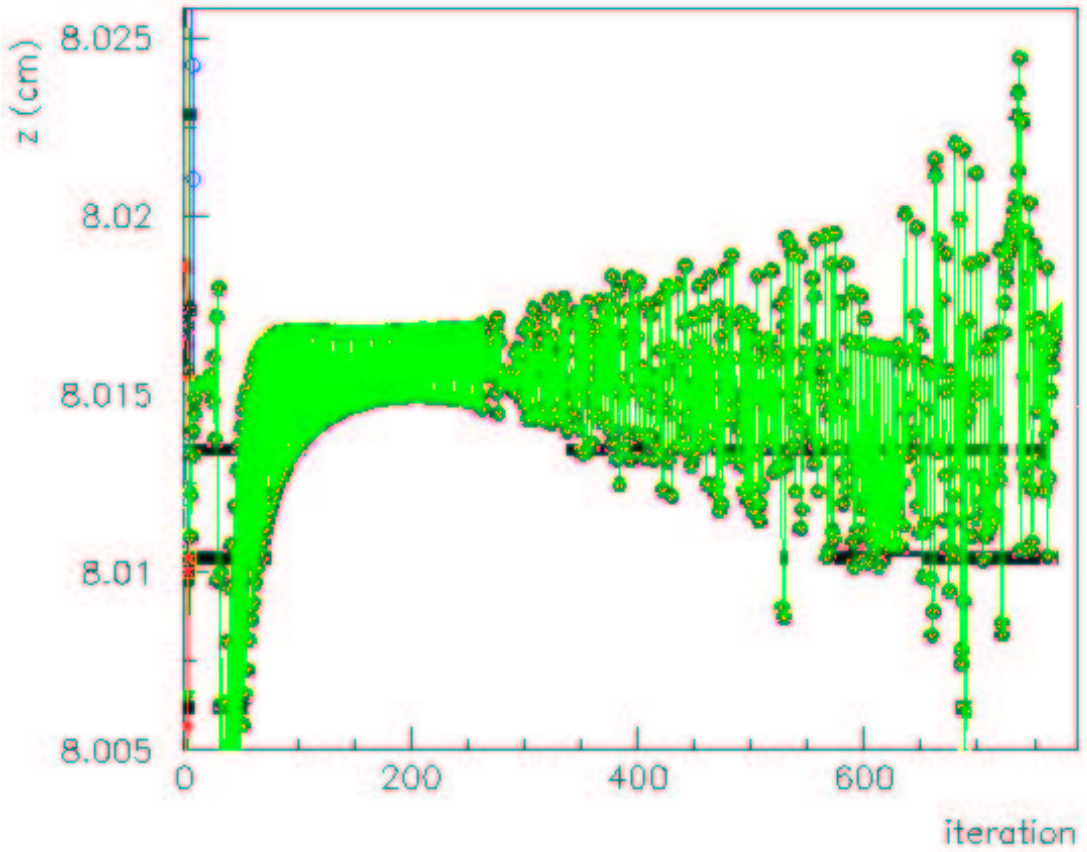

Figure 5.23 – Cas de divergence spontanée en position z (en cm) de "vertex seed" sans aucune constatation de convergence pendant 800 itérations en température.

Une fois tous ces paramètres évalués, les "vertex seed" peuvent alors se déplacer au fur et à mesure des itérations en température. Plus l'itération en température augmente, moins les "vertex seed" se déplacent pour converger vers une ou plusieurs positions finales (figure 5.24). De plus, on remarque le bon fonctionnement de la fusion des "vertex seed" proches

(moins de 100 μm).

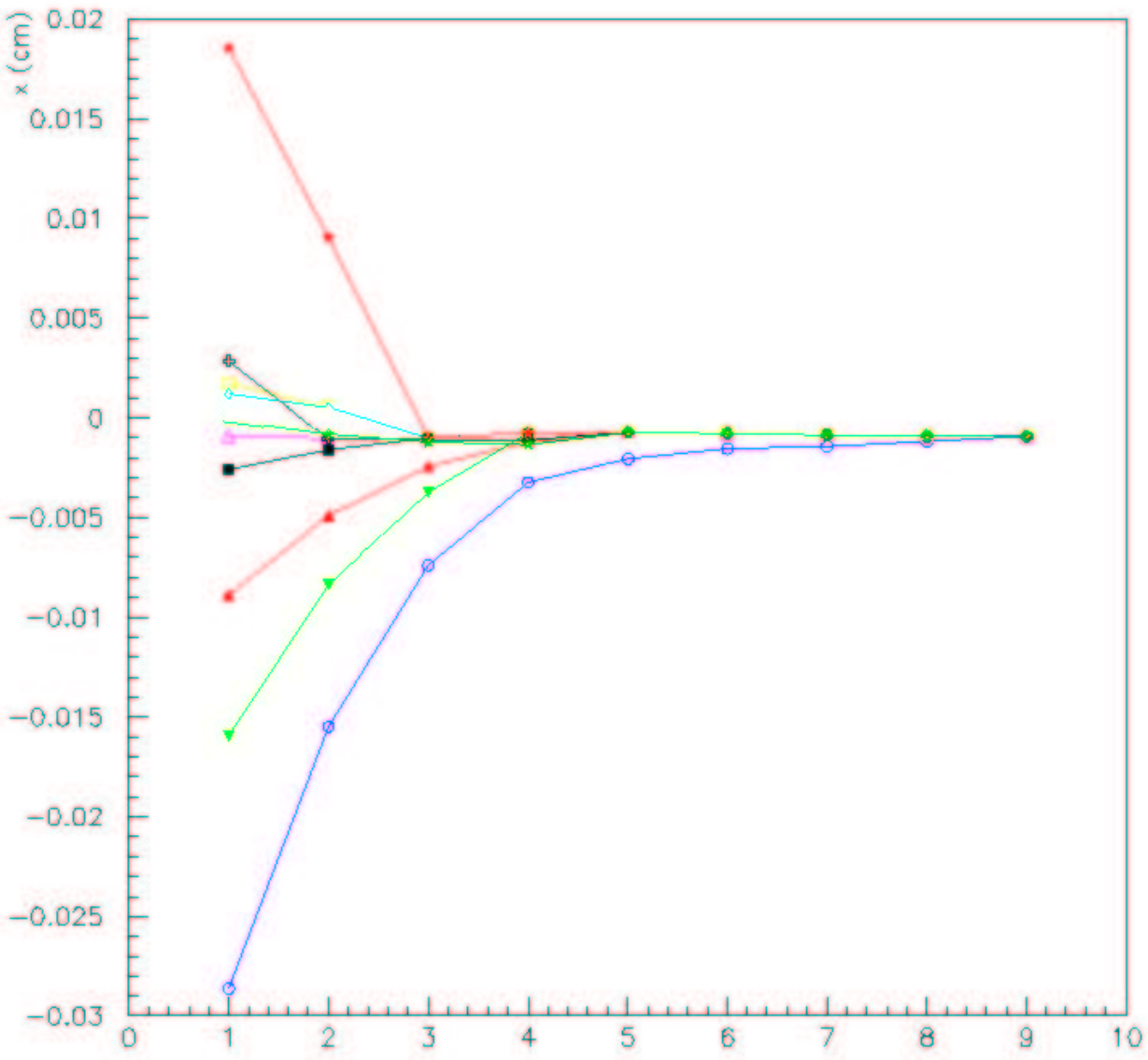

Figure 5.24 – Variation de la position x (en cm) de 10 "vertex seed" en fonction des itérations en température pour un événement contenant 5 traces.

Sur les figures 5.25 et 5.26, on visualise en 3 dimensions le pouvoir d'attraction des traces sur les "vertex seed" en direction du vertex primaire. Cette visualisation est rendue possible grâce au développement de programmes permettant la visualisation de la position de vertex simulés, de vertex reconstruits (avec accès à l'erreur sur la position) dans le cadre d'IGUANA (Interactive Graphical User ANAlysis) [IGU]. Ces programmes permettent aussi la visualisation du déplacement de "vertex seed" entre chaque itération en température de l'algorithme EA.

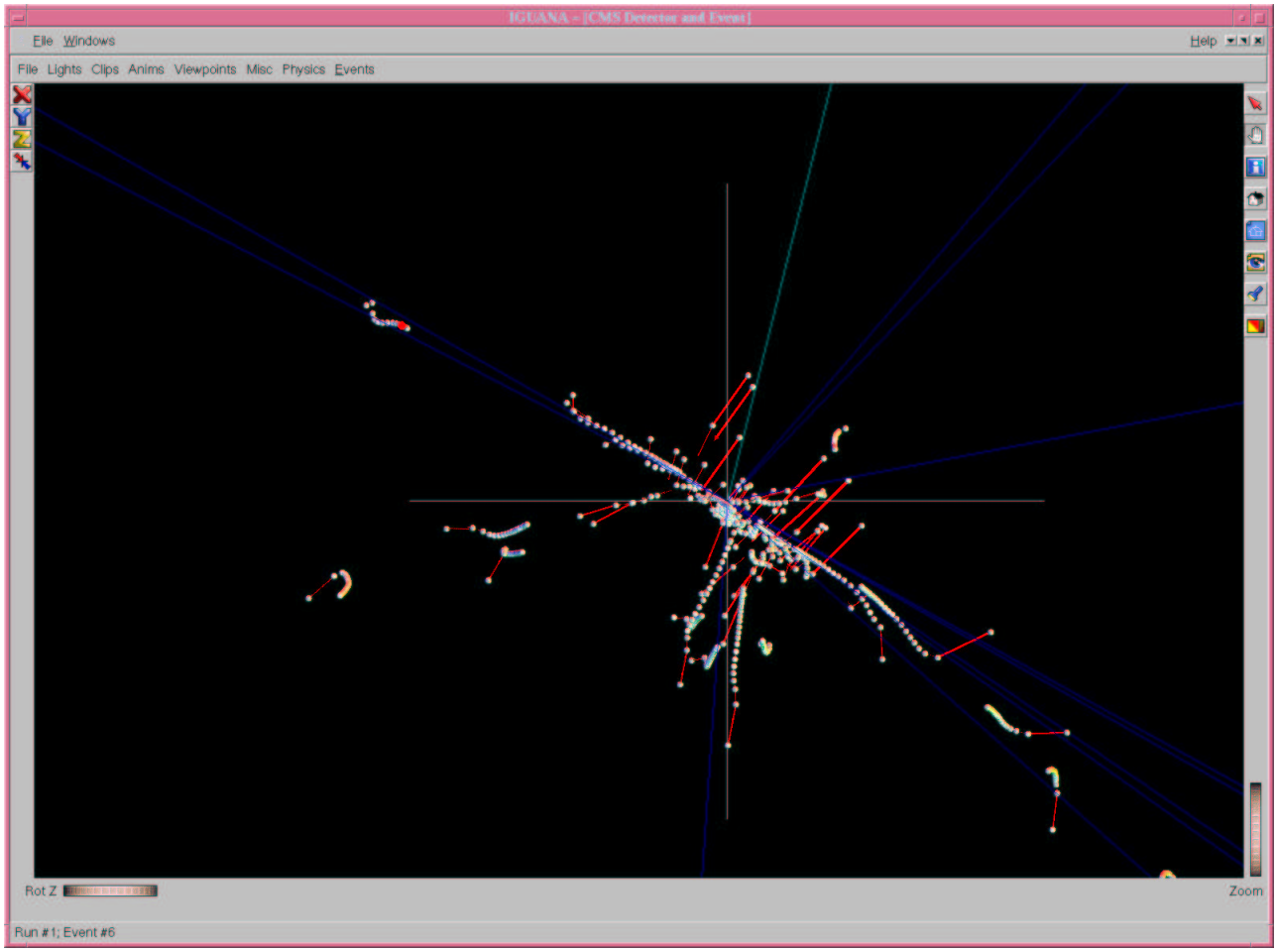

Figure 5.25 – Un événement $b\,\bar{b}$ contenant 48 traces simulées (ligne bleue) et 135 "vertex seed". Un "vertex seed" est representé par un ensemble de sphères (une sphère pour chaque itération) connectées entre elles par des lignes rouges.

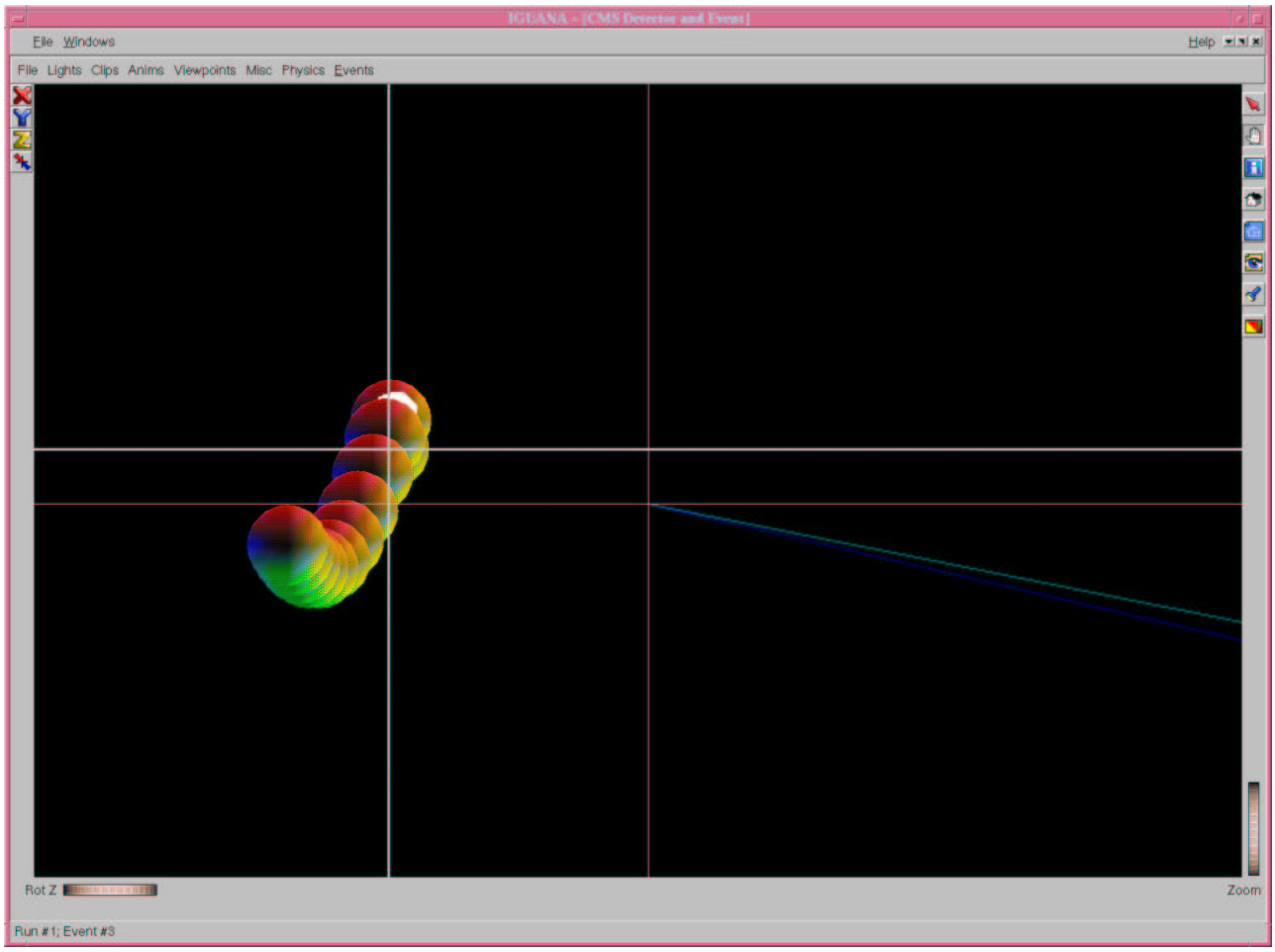

Figure 5.26 – Gros plan sur le déplacement d'un "vertex seed (representé par un ensemble de sphères, une sphère pour chaque itération, connectées entre elles par des lignes rouges) vers un vertex secondaire reconstruit (croix blanche épaisse) situé à 3.3 μm du vertex simulé (croix blanche fine) d'où partent deux traces (ligne bleue).

5.3.5.3.1 Effet de la coupure sur le paramètre d'impact

Le programme EA est consacré à la recherche de vertex secondaires en vue de l'identification de jets issus de la fragmentation de quarks b. Ainsi pour créer les "vertex seed", les vertex secondaires potentiels, on ne sélectionne que les traces avec un grand paramètre d'impact. Dans la table 5.6, le nombre de traces sélectionnées pour les événements $b\bar{b}$[8] et $q\bar{q}$[8] sont indiquées pour les différentes coupures sur la significance du paramètre d'impact transverse. La significance sur le paramètre d'impact transverse étant défini comme le rapport de la valeur du paramètre d'impact transverse sur son erreur. Cette coupure permet

[8]Evénements avec une coupure en énergie de 50 GeV, avec $\eta < 1,4$ et sans empilement.

d'éliminer un maximum de traces primaires reconstruites tout en gardant un maximum de traces secondaires reconstruites.

coupure sur la significance du paramètre d'impact transverse	2	3	4	5
nombre de traces sélectionnées pour les événements $b\bar{b}$	31,8%	24,7%	20,6%	17,2%
nombre de traces sélectionnées pour les événements $q\bar{q}$	13,0%	7,1%	5,1%	4,3%

Tableau 5.6 – Nombre de traces reconstruites ayant une significance du paramètre d'impact transverse plus importante que la coupure.

Les traces simulées des événements $q\bar{q}$ proviennent principalement des vertex primaires (90-95 %), et, les traces simulées des événements $b\bar{b}$ proviennent à la fois des vertex primaires ($\simeq$ 75 %) et secondaires ($\simeq$ 25 %). Le but de cette coupure sur le paramètre d'impact transverse est d'éliminer un maximum de traces des événements $q\bar{q}$ tout en gardant les traces des événements $b\bar{b}$ qui proviennent de vertex secondaires. La coupure sur le paramètre d'impact transverse à 3 σ semble être la coupure qui élimine le plus de traces $q\bar{q}$ sans trop éliminer de traces $b\bar{b}$.

5.3.5.4 Application à la reconstruction de vertex

Pour chaque événement, on détermine les traces reconstruites, les traces simulées, les vertex reconstruits et les vertex simulés. Les vertex primaires sont reconstruits par la méthode des amas (décrite chap 5.3.4.2) et dans le cas de plusieurs vertex primaires, on ne garde que le vertex associé à plus d'une trace et avec la plus grande impulsion transverse des traces associées au vertex. Les vertex secondaires sont reconstruits par l'algorithme EA (décrit chap 5.3.5.2). De plus, pour chaque trace reconstruite, on cherche la trace simulée associée puis on reconstruit le vertex simulé associé à chaque trace en ne gardant que les vertex simulés associés à un minimum de 2 traces.

5.3.6 Programmes alternatifs pour la reconstruction de vertex secondaires

Deux programmes alternatifs pour la reconstruction de vertex secondaires, la méthode de "PrimaryVertexFinder" et la méthode de $D_0\phi$, sont brièvement décrites ci-dessous. Les performances de reconstruction de ces deux programmes ont servi de comparaison à l'algorithme EA en terme d'efficacité, de taux de vertex fantômes (paragraphe 5.4.2.2) et de rapidité d'exécution (paragraphe 5.4.2.4). Les résolutions sur les vertex reconstruits ne sont pas connues pour ces programmes alternatifs.

5.3.6.1 La méthode de "PrimaryVertexFinder" (PVF)

Cette méthode a été conçue pour reconstruire le plus rapidement possible les vertex primaire et secondaires d'un jet issu de la fragmentation de quark b.

Cet algorithme récursif recherche le vertex associé à un ensemble de traces. D'abord on détermine la qualité de reconstruction du vertex reconstruit (par un calcul de χ^2) et si ce vertex est correctement reconstruit, les traces associées à ce vertex sont exclues du nouvel ensemble de traces (par un calcul de "compatibilité"). Une fois ces traces éliminées, on recherche un nouveau vertex avec un nouvel ensemble de traces. De nouveaux vertex sont recherchés tant qu'il reste des traces incompatibles avec les vertex progressivement trouvés [ORC].

L'optimisation de cet algorithme est simple car seulement deux paramètres peuvent être ajustés : la coupure sur la valeur du χ^2 de reconstruction du vertex et la coupure de "compatibilité" entre une trace et un vertex. Un vertex est considéré comme bien reconstruit si la valeur de son χ^2 de reconstruction est inférieur à 0,5 et une trace est dit "compatible" avec un vertex pour "probabilité de compatibilité" supérieure à 5%.

5.3.6.2 La méthode de $D_0\phi$

Cette méthode a déjà été utilisée par la collaboration CDF pour la découverte du quark top [ABE94]. Cet algorithme est basé sur l'existence d'une dépendance entre le paramètre d'impact transverse d'une trace (d_0) et son angle azimutal (ϕ) dans le plan transverse ($r - \phi$). En effet, si le paramètre d'impact transverse de la trace est du même signe que son moment angulaire, on a la relation suivante : $d_0 \sim lsin(\phi - \phi_0) \sim l(\phi - \phi_0)$ où l est la distance transverse entre le vertex primaire et le vertex secondaire considéré et ϕ_0 l'angle azimuthal associé (figure 5.27). Les traces provenant du vertex primaire sont définies telles que $(d_0 \simeq 0, \phi)$. Et les traces telles que $(d_0 = l(\phi - \phi_0), \phi)$ sont originaires d'un vertex secondaire. D'autre part, il existe autant de vertex secondaires que de couples (l, ϕ_0) (figure 5.27) [ORC].

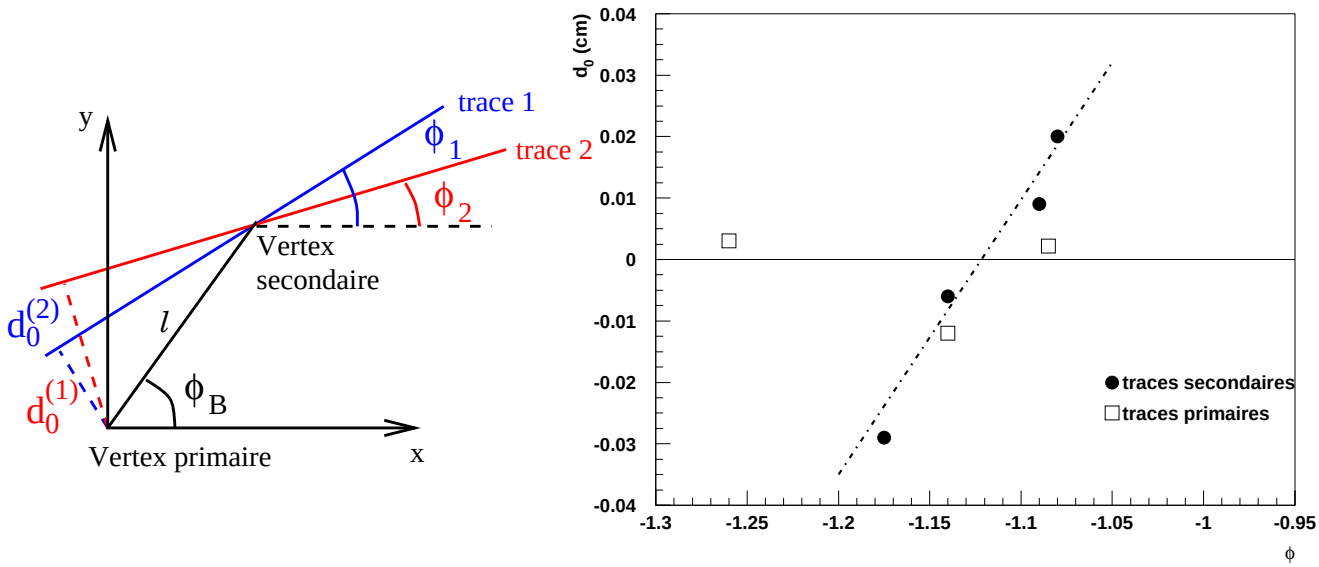

Figure 5.27 – Illustration de la méthode de $D_0\phi$.

Une fois l'ensemble des traces associées à un même vertex défini, on détermine la position du vertex par la *méthode de linéarisation* (la classe *LinearVertexFitter*).

5.4 Performance de l'algorithme EA et identification de jets b

5.4.1 Evaluation des performances de l'algorithme EA

Pour évaluer les performances de l'algorithme EA, diverses valeurs de paramètres comparatifs pour les événements $b\bar{b}$ et $q\bar{q}$ ont été résumées dans la table 5.7. L'étude a porté sur 2569 événements $b\bar{b}$ et $q\bar{q}$ de 50 GeV avec $\eta > 1,4$. Les événements $q\bar{q}$ sont analysés à titre comparatif. Seuls les événements dans lesquels au moins deux traces produisent un "vertex seed" de χ^2 supérieur à 20 peuvent être analysés par l'algorithme EA (ligne 6 du tableau 5.7). Le pourcentage important d'événements $q\bar{q}$ non analysés par l'EA est lié au fait que peu de traces $q\bar{q}$ passent la coupure sur le paramètre d'impact (ligne 9 du tableau 5.7). En effet, les événements $q\bar{q}$ produisent majoritairement des vertex primaires et très peu de vertex secondaires. De plus, pour les événements contenant un nombre important de traces, les vertex simulés n'ont pas pu être produits (ligne 7 du tableau 5.7) à cause d'une erreur encore non déterminée dans la génération des vertex simulés.

Une première façon d'évaluer les performances de l'algorithme EA est de savoir qu'il converge en une vingtaine d'itérations en température (ligne 8 du tableau 5.7) et que peu

1	type d'événement	$b\bar{b}$	$q\bar{q}$
2	Energie (GeV)	50 GeV	50 GeV
3	Limite en η	$\eta < 1,4$	$\eta < 1,4$
4	avec ou sans empilement ?	sans empilement	sans empilement
5	Nombre d'événements étudiés	2569	2569
6	Nombre d'événements étudiés par l'EA	2326 (90,5 %)	2542 (98,9%)
7	Nombre d'événements étudiés avec des vertex simulés	2007 (78,1 %)	2490 (96,9 %)
8	Nombre moyen d'itérations maximales	$19,7 \pm 6,4$	$29,1 \pm 15,4$
9	Nombre d'événements sans création de "vertex seed"	268 (10,4 %)	2167 (84,3 %)
10	Nombre d'événements ayant atteint 50 itérations	5 (0,2 %)	49 (1,9 %)
11	Nombre moyen de "vertex seed" à la première itération	$8,2 \pm 5,8$	$2,5 \pm 0,8$
12	Nombre moyen de "vertex seed" à la dernière itération	$5,8 \pm 4,2$	$2,2 \pm 0,6$
13	Nombre de vertex primaires	2569	2508
14	Nombre de vertex secondaires	4800	247
15	Nombre de traces par vertex primaire	$20,2 \pm 5,3$	$18,9 \pm 5,4$
16	Nombre de traces par vertex secondaire	$3,0 \pm 1,5$	$2,9 \pm 1,8$
17	Nombre moyen de vertex reconstruits par événement	$2,9 \pm 1,3$	$1,1 \pm 0,3$
18	Nombre moyen de vertex simulés par événement	$5,5 \pm 1,2$	$1,7 \pm 0,9$

Tableau 5.7 – Caractéristiques comparatives pour les événements $b\bar{b}$ et $q\bar{q}$ permettant d'évaluer les performances de l'algorithme EA.

d'événements nécessitent 50 itérations pour converger (ligne 10 du tableau 5.7). D'autre part, en négligeant le fort écart-type, l'évolution du nombre de "vertex seed" au cours des itérations (ligne 11 et 12 du tableau 5.7) montre une faible diminution du nombre de "vertex seed" bien que la condition de fusion de deux vertex sont d'être distants de moins de 100 μm. En outre, pour chaque itération en température , les "vertex seed" sont plus ou moins attirés par chaque trace reconstruite (formule 5.3) et l'ordre de grandeur du déplacement est d'environ 3,1 μm pour chaque trace.

Les vertex primaires sont associés à une vingtaine de traces alors que les vertex secondaires sont associés à en moyenne 3 traces. Le nombre de traces reconstruites associées à un vertex peut être utilisé pour différencier les vertex primaires et secondaires (tableau 5.8).

coupure sur le nombre de traces associées	moins de 5	moins de 7	plus de 15	plus de 20	plus de 25
vertex primaire de $b\bar{b}$	0,0%	0,2%	77,9%	44,6%	16,1%
vertex primaire de $q\bar{q}$	0,1%	0,6%	70,2%	36,5%	11,9%
vertex secondaire de $b\bar{b}$	61,2%	65,6%	0,0%	15,0%	7,1%
vertex secondaire de $q\bar{q}$	90,1%	96,4%	0,2%	0,1%	0,0%

Tableau 5.8 – Différentes coupures sur le nombre de traces reconstruites associées à un vertex reconstruit (primaire ou secondaire) sur une base de 2570 événements $b\bar{b}$ et $q\bar{q}$.

Enfin, dans le programme de test (paragraphe 5.3.5.4), les vertex primaires sont reconstruits par la méthode des amas. Mais, même si l'algorithme EA n'a pas pour but de reconstruire les vertex primaires, ce programme reconstruit 32 % des vertex primaires $b\bar{b}$ (avec une erreur inférieure à 5 μm) en dépit de la sélection sur les traces (significance sur le paramètre d'impact supérieure à 3). Il ne reconstruit que 15 % des vertex primaires $q\bar{q}$ car un faible nombre de traces $q\bar{q}$ ont une significance sur le paramètre d'impact supérieure à 3. Les vertex primaires reconstruits par l'algorithme EA sont associés à 13 ± 8 traces reconstruites au lieu de 20 ± 5 pour les vertex primaires reconstruit classiquement et la résolution sur les vertex primaires reconstruits par l'algorithme EA est deux fois moins bonne que celle sur les vertex reconstruits par la méthode des amas.

5.4.2 Les vextex : résolution, efficacité et vertex fantômes

Maintenant que l'algorithme EA a été évalué, il faut déterminer si ce programme est compétitif pour la reconstruction de vertex secondaires pour des événements $b\,\bar{b}$ par rapport aux méthodes PVF et $D_0\phi$ (paragraphes 5.3.6.1 et 5.3.6.1). A cette fin, nous étudierons quatre critères basés sur les performances de l'algorithme :

– la résolution, c'est-à-dire si le vertex reconstruit est proche du vertex simulé ;

- l'efficacité, c'est-à-dire si on reconstruit le bon nombre de vertex ;
- le taux de vertex fantômes, c'est-à-dire si on ne reconstruit pas trop de vertex ;
- et le temps CPU mis par l'algorithme EA pour reconstruire ces vertex secondaires.

5.4.2.1 La résolution

A chaque vertex reconstruit (x_r, y_r, z_r), on associe le vertex simulé le plus proche (x_s, y_s, z_s) et on détermine leur différence en position selon les axes x, y et z : ($\Delta_x = x_r - x_s$, $\Delta_y = y_r - y_s$, $\Delta_z = z_r - z_s$). Les résolutions en x, y et z sont définies comme égales aux écarts-types des distributions Δ_x, Δ_y et Δ_z que l'on estime par une gaussienne pour les vertex primaires (figures 5.28 et 5.29) et par deux gaussiennes (une large et une plus piquée) sommées pour les vertex secondaires (figures 5.30 et 5.31). La résolution est fournie par les paramètres de la gaussienne la plus piquée.

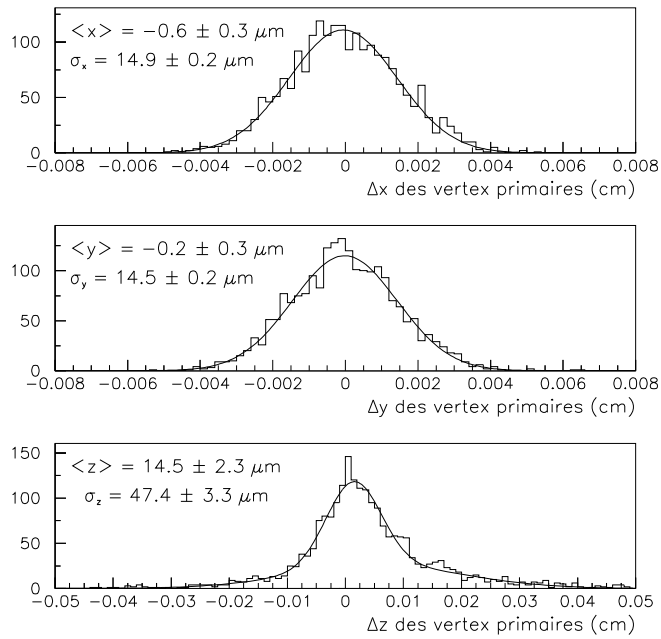

Figure 5.28 – Evénements $b\bar{b}$ à 50 GeV.

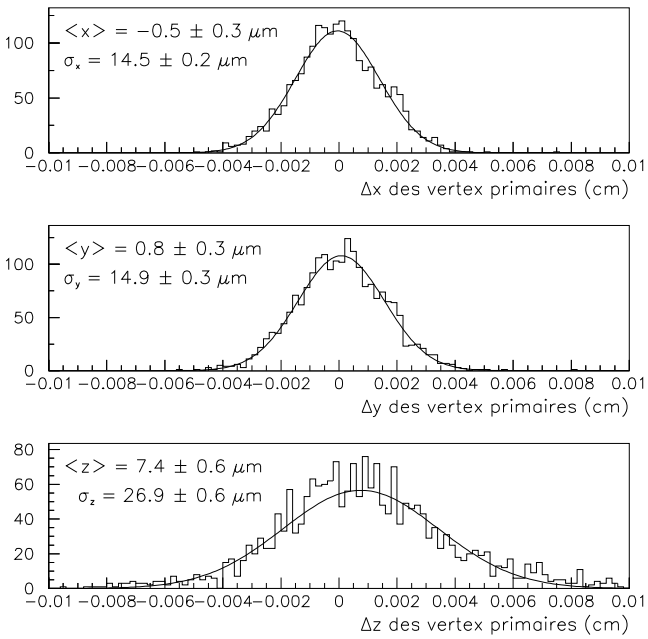

Figure 5.29 – Evénements $q\bar{q}$ à 50 GeV.

Figures 5.28 et 5.29 : Résolution en x, y et z sur les vertex primaires reconstruits par la méthode des amas (2569 evénements).

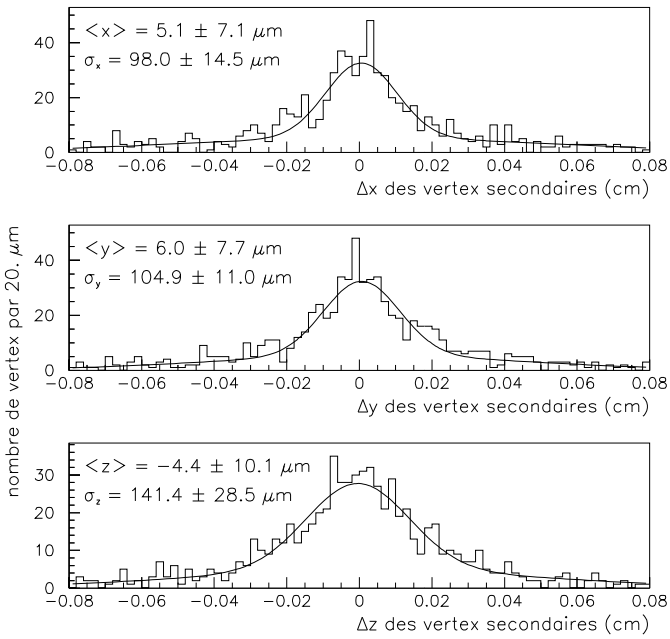

Figure 5.30 – Evénements $b\bar{b}$ à 50 GeV.

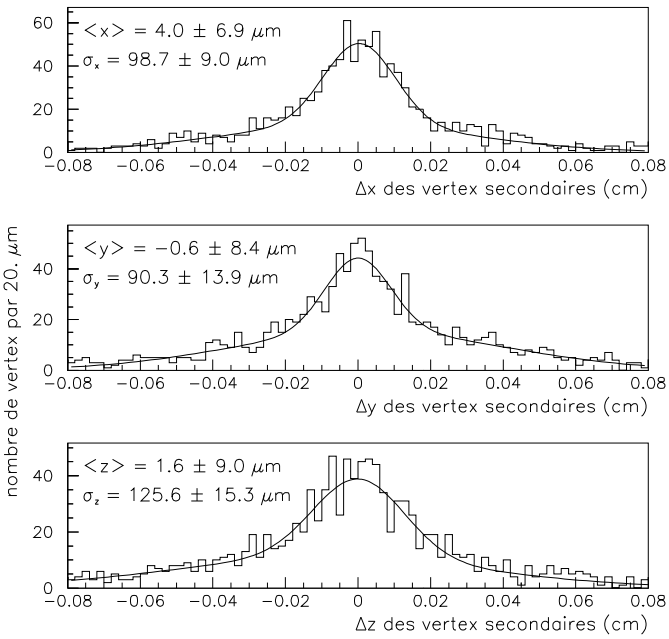

Figure 5.31 – Evénements $b\bar{b}$ à 100 GeV.

Figures 5.30 et 5.31 : Résolution en x, y et z sur les vertex secondaires reconstruits par l'algorithme EA (2569 evénements).

Les résolutions pour les événements $q\ \bar{q}$ et trois types d'événements $b\ \bar{b}$ (paragraphe

5.3.4) sont résumées ci-dessous dans le tableau 5.9.

Type d'événements	particules	$q\ \bar{q}$	$b\ b$	$b\ b$
	Energie	50 GeV	50 GeV	100 GeV
	limite en η	$\eta < 1,4$	$\eta < 1,4$	$\eta < 1,4$
Nombre de vertex primaires		2455	2208	1244
résolution sur les vertex primaires en μm	x	14,5 ± 0,2	14,9 ± 0,2	14,1 ± 0,3
	y	14,9 ± 0,3	14,5 ± 0,2	14,7 ± 0,4
	z	26,9 ± 0,6	47,4 ± 3,3	53,7 ± 3,9
Nombre de vertex secondaires		61	2497	3091
résolution sur les vertex secondaires en μm	x		98,0 ± 14,5	98,7 ± 9,0
	y		104,9 ± 11,0	90,3 ± 13,9
	z		141,4 ± 28,5	125,6 ± 15,3

Tableau 5.9 – Résolution sur les vertex primaires et secondaires pour 2500 événements.

Le faible nombre de vertex secondaires pour les événements $q\ \bar{q}$ ne permet pas de déterminer la résolution correspondante.

Une résolution en z entre 20 et 55 μm est obtenue pour les vertex primaires (reconstruits par la méthode des amas). Pour les vertex secondaires (reconstruits par l'algorithme EA), on obtient une bonne résolution de $\sim 100\ \mu m$ pour x,y et $\sim 130\ \mu m$ pour z. L'algorithme EA reconstruit des vertex secondaires avec une bonne résolution en comparaison avec la résolution en z sur les vertex primaires.

5.4.2.2 L'efficacité et le taux de vertex fantômes

On évalue ensuite l'efficacité et le taux de vertex fantômes. De façon générale, l'algorithme EA reconstruit un vertex primaire par événement $b\bar{b}$ et $q\bar{q}$ et 2 vertex secondaires par événement $b\bar{b}$ (lignes 13 et 14 du tableau 5.7), ce qui est généralement le nombre de vertex moyen pour les événements $b\ \bar{b}$ contenant deux jets. Même si cela reste bien en-dessous des plus de 5 vertex simulés (ligne 18 du tableau 5.7).

On associe à chaque vertex reconstruit le vertex simulé qui ont le plus grand nombre de traces en commun. On définit aussi la pureté d'un vertex reconstruit par

$$Pureté = \frac{Nombre\ de\ traces\ partagées\ par\ les\ vertex\ reconstruit\ et\ simulé}{Nombre\ total\ de\ traces\ associées\ au\ vertex\ reconstruit}$$

Dans notre cas, l'association d'un vertex reconstruit à un vertex simulé est réussie pour une pureté supérieure à 60%. Les vertex reconstruits non associés à un vertex simulé sont appelés "vertex fantômes". L'efficacité de reconstruction des vertex est définie par le nombre de vertex reconstruits associés à un vertex simulé, divisé par le nombre total de vertex simulés. Et le taux de vertex fantômes ou "fake rate" est défini par le nombre de vertex reconstruits fantômes divisé par le nombre total de vertex reconstruit. En résumé, l'efficacité ϵ et le taux de vertex fantôme F sont déterminés comme suit :

$$\epsilon = \frac{\text{Nombre de vertex reconstruits associés à un vertex simulé}}{\text{Nombre de vertex simulés}} \tag{5.4}$$

$$F = \frac{\text{Nombre de vertex reconstruits NON associés à un vertex simulé}}{\text{Nombre de vertex reconstruits}} \tag{5.5}$$

Remarquons que dans ce paragraphe, la définition des vertex primaires et secondaires change : un vertex est considéré comme primaire si sa position transverse est compatible avec celle du faisceau (on considère que la taille du faisceau est telle que $\sigma_x = 15\ \mu m$ et $\sigma_y = 15\ \mu m$) avec une probabilité plus grande que 5 %, tous les autres vertex sont considérés comme secondaires. Les différentes valeurs d'efficacité et du taux de vertex fantômes pour les vertex secondaires sont résumées dans le tableau 5.10 pour différents algorithmes :

- l'algorithme EA, décrit dans le paragraphe 5.3.5.2,
- le programme PVF, très brièvement décrit au paragraphe 5.3.6.1,
- et le programme $D_0\phi$, succintement décrit au paragraphe 5.3.6.2.

type d'événements		$q\bar{q}$, 50 GeV $\eta < 1,4$	$b\bar{b}$, 50 GeV $\eta < 1,4$	$b\bar{b}$, 100 GeV $\eta < 1,4$	$b\bar{b}$, 100 GeV $\eta > 1,4$
Nombre de vertex secondaires		80	2623	2855	838
Méthode EA	efficacité	38,7 ± 5,4 %	29,9 ± 0,9 %	27,3 ± 0,8 %	23,7 ± 1,5 %
	vertex fantômes	< 1%	< 1%	< 1%	< 1%
Méthode $D_0\phi$	efficacité	46,2 ± 5,6 %	27,7 ± 0,8 %	37,2 ± 0,9 %	46,2 ± 5,6 %
	vertex fantômes	42 ± 4 %	54 ± 1 %	76 ± 0,3 %	42 ± 4 %
Méthode PVF	efficacité	31,2 ± 5,2 %	24,9 ± 0,8 %	21,6 ± 0,7 %	31,2 ± 5,2 %
	vertex fantômes	30 ± 3 %	47 ± 1 %	49 ± 1 %	60 ± 2 %

Tableau 5.10 – L'efficacité et le taux de vertex fantômes pour les vertex secondaires reconstruits par trois méthodes.

Les sources possibles de l'inefficacité de reconstruction ($\simeq 70 - 75$ %) sont en grande partie due à un grand nombre de vertex reconstruits associés à moins de 2 traces. Mais cette inefficacité est aussi due à une pureté d'association des vertex reconstruits aux vertex simulés inférieure à 55%, ou parfois à une mauvaise reconstruction des vertex.

Les performances de l'algorithme EA sont assez compétitives par rapport aux principaux algorithmes de reconstruction de vertex secondaires d'O.R.C.A. Même si l'efficacité n'est pas aussi importante que pour la méthode $D_0\phi$, l'algorithme EA se distingue par un taux de vertex fantômes presque nul pour tous les types d'événements étudiés.

5.4.2.3 Identification de jets contenant des quarks b

Pour chaque événement, on recherche alors les vertex secondaires (reconstruits par la méthode EA) en ne sélectionnant qu'un seul vertex secondaire par jet ; en faisant l'hypothèse que chaque événement $b\ \bar{b}$ produit deux jets de direction opposée. On identifie ainsi 2877 jets (56,2%) pour 2569 événements $b\ \bar{b}$ (soit idéalement 5138 jets) et 232 jets (4,5%) pour 2569 événements $q\ \bar{q}$. Puis on établit différentes coupures sur les caractéristiques des vertex secondaires reconstruits à la fois en distance tranverse ($d_t = \sqrt{x_{vtx}^2 + y_{vtx}^2}$) et en χ^2_{ndf} ($\chi^2_{ndf} = \frac{\chi^2}{\text{Nombre de traces associées au vertex}}$). Puis on recalcule le nombre de jets identifiés pour chaque cas (tableau 5.11). ll, : Command not found. Illegal variable name.

coupure	aucune	$d_t > 10\mu m$	$\chi_{ndf} < 3$	$d_t > 10\mu m$ $\chi_{ndf} < 3$	$d_t > 10\mu m$ $\chi_{ndf} < 2,5$
jets identifiés dans les événements $b\ \bar{b}$ (%)	56,2	56,2	51,7	51,4	50,1
jets identifiés dans les événements $q\ \bar{q}$ (%)	4,5	4,5	3,1	3,1	2,9
coupure	$d_t > 10\mu m$ $\chi_{ndf} < 2$	$d_t > 10\mu m$ $\chi_{ndf} < 1,5$	$d_t > 10\mu m$ $\chi_{ndf} < 1$	$d_t > 10\mu m$ $\chi_{ndf} < 0,5$	$d_t > 10\mu m$ $\chi_{ndf} < 0,3$
jets identifiés dans les événements $b\ \bar{b}$ (%)	48,6	44,9	40,0	27,8	19,8
jets identifiés dans les événements $q\ \bar{q}$ (%)	2,6	2,2	1,9	1,4	1

Tableau 5.11 – Pourcentage de jets identifiés comme issus d'un quark b pour 2570 événements $b\ \bar{b}$ et 2570 événements $q\ \bar{q}$ en fonction de coupure en χ^2 et en distance transverse (d_t).

On peut ainsi identifier $\sim$ 50 % de jets comme réellement issus d'un quark b contre $\sim$ 3% de jets identifié b alors qu'issus de quarks légers.

5.4.2.4 Le temps CPU

La reconstruction de vertex primaires est extrêmement rapide, ce qui est confirmée par les résultats résumés dans le tableau 55.12. Par contre, l'algorithme EA est un processus itératif volontairement lent pour une meilleure résolution ; deux fois plus lent que le plus lent des programmes de reconstruction de vertex (méthode $D_0\phi$).

Type de vertex reconstruit	Type de programme	temps CPU	temps CPU par événement
primaire	méthode des amas	370 ms	0,37 ms
primaire	méthode par "binning"	480 ms	0,48 ms
primaire	méthode des gaussiennes	2 790 ms	2,79 ms
secondaire	EA	2h57	10 620 ms
secondaire	$D_0\phi$	1h31	5 460 ms
secondaire	PVF	39 min	2 340 ms

Tableau 5.12 – Le temps CPU pour différents programmes de reconstruction pour 1000 événements $b\ \bar{b}$.

5.5 Conclusion

Les trois méthodes de reconstruction de vertex primaires ont des résolutions et des efficacités comparables. De plus, l'empilement des événements ne semble pas être un problème pour reconstruire le vertex primaire.

Malgré toutes les difficultés pour reconstruire les vertex secondaires (l'environnement difficile du LHC, la reconstruction des traces, etc...), l'algorithme EA conduit à une excellente résolution sur les positions x, y et z et à un taux de vertex fantôme presque idéal.

Des améliorations doivent néanmoins être apportées pour améliorer l'efficacité et surtout le temps CPU de reconstruction :
- améliorer l'élimination des "vertex seed" mal placés ;
- augmenter l'ordre de grandeur des déplacements tout en évitant les divergences.

De plus, ce programme permet d'identifier 50 % des jets pour les événements $b\bar{b}$ alors que 3 % des événements $q\bar{q}$ sont identifiés b ; cela uniquement en analysant la position des vertex secondaires reconstruits par l'algorithme EA.