Traitement online de données pour la physique des particules

Frédéric Magniette

Traitement online de données pour la physique des particules

Mémoire pour l'obtention de l'habilitation à diriger les recherches

présenté par

Frédéric Magniette

Laboratoire Leprince-Ringuet, CNRS, Ecole Polytechnique

Soutenu le 22 Mai 2019 devant la commission d'examen :

M.	Didier	Lacour	Président
M.	Didier	Contardo	Rapporteur
M.	Volker	Beckmann	Rapporteur
M.	Cristinel	Diaconu	Rapporteur
M.	Imad	Laktineh	Examinateur

À la mémoire de Pierre Mergault (1922-1993)
Directeur de recherche au CNRS
Directeur du Laboratoire de Physique des Liquides Ioniques
à l'Université Pierre et Marie Curie

Table des matières

Chapitre 1

Introduction

Le traitement *online* des données est un sujet qui devient crucial pour la physique des particules. Il s'agit d'appliquer sur les données brutes des algorithmes de traitement à des vitesses proches de celle de l'acquisition.

Il a longtemps été impossible d'imaginer un tel traitement à cause des capacités insuffisantes des ordinateurs. Aujourd'hui, le paradigme des infrastructures de calcul change, passant de l'ère du traitement séquentiel essentiellement monolithique à un paradigme beaucoup plus distribué. On trouve des processeurs dans presque tous les matériels, les micro-processeurs sont des mini-fermes de calcul, les cartes graphiques fournissent également de grosses capacités de traitement. On commence même à trouver des processeurs dans les cartes réseaux et dans les barrettes de mémoire vive. Cette multiplication des processeurs tout au long de la chaine de traitement implique une capacité de traitement individuellement plus faible mais démultipliée par le nombre des composants disponibles.

Par ailleurs, les détecteurs de particules qui sont conçus aujourd'hui disposent d'un nombre de canaux de mesures supérieurs de plusieurs ordres de grandeurs à ce qui était la norme jusque là. L'apparition de détecteurs ultra-granulaires, permet une gestion plus fine des traces. Ils sont capables d'absorber et d'exploiter des luminosités inimaginables jusqu'ici. Mais l'ultra-granularité se paie en terme de nombre de canaux et de bande passante dédiée à la donnée brute. Sur certaines expériences, cette bande passante est telle qu'elle dépasse largement les possibilités techniques traditionnelles, obligeant les ingénieurs à développer des technologies spécifiques à très haute vitesse et des politiques de déclenchement très sélectives.

Or, la donnée brute peut souvent être résumée au terme d'un traitement qui, s'il est correctement automatisé, peut réduire drastiquement cette bande passante. De nombreux algorithmes permettent de transformer ces données brutes en données utilisables (nature de la particule et quantité de mouvement). Naturellement, c'est une solution plus risquée que le stockage complet car en cas de mauvaise programmation ou de défaillance, celle-ci est perdue. Toutefois, il permet d'envisager un nombre de canaux inaccessible autrement.

Un autre enjeu du traitement *online*[1] de données est le monitoring. Il est indispensable pour un équipement dont le fonctionnement engendre des coûts importants de disposer d'un outil d'analyse temps-réel qui permette de vérifier la qualité des données. En effet, si la prise de données se fait à l'aveugle, toute erreur de configuration passera inaperçue et engendrera une perte

1. Les mots en italiques sont décrits dans le lexique situé à la fin du document

de données et donc ene perte en terme de coût de fonctionnement. La problématique centrale du monitoring est de traiter en temps-réel des données qui, par nature, sont trop importantes pour cela, particulièrement lorsque les traitements de monitoring sont complexes. Il est alors nécessaire de faire appel à des mécanismes de distribution de la charge ou de *sub-sampling*. Le principe de ce dernier est de choisir certaines données, statistiquement significatives, à une bande passante acceptable pour l'infrastructure de calcul.

Le dernier enjeu du traitement *online* des données est la sureté de fonctionnement du détecteur lui-même. Certains phénomènes physiques sont de nature à dégrader une partie du détecteur et doivent être évités. Sur de grosses infrastructures, on pourra acheter ou construire un système de détection spécifique basé par exemple sur des automates industriels, mais sur de nombreuses expériences de taille petite ou moyenne, seule l'analyse en temps-réel des données permettra d'ajuster son fonctionnement pour éviter les dégradations.

Afin de mener à bien ce travail d'analyse en temps-réel, les développeurs de l'informatique temps-réel (*online*) doivent adapter leurs méthodes de travail et intégrer des algorithmes réservés jusque là au développeurs qui s'occuppent de l'analyse (offline) mais dans un approche axée sur l'optimisation et l'échantillonnage. D'une manière générale, on observe une convergence très nette entre les deux catégories de développeurs jusque là strictement confinés dans leur spécialité.

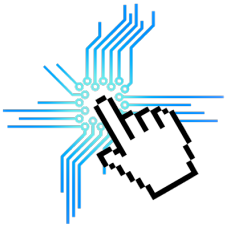
Dans ce mémoire, j'illustre cette évolution sur la base d'expériences réelles, en décrivant en détails les aspects techniques de ce traitement temps-réel. Je commencerai en décrivant le *framework* Pyrame que je développe au laboratoire Leprince-Ringuet depuis 2012 et dont la dernière version intègre des mécanismes permettant d'automatiser ces traitements *online*. Les mécanismes de *sub-sampling* et de distribution de données y sont implémentés de manière générique et mis à disposition des applications pour faciliter leur développement. Ces travaux ont permis de mettre en oeuvre des démarches de traitement *online* sur plusieurs expériences. En particulier, je décrirais les travaux que j'ai réalisé sur l'expérience *SiW-Ecal*, un calorimètre électromagnétique pour le futur International Linear Collider (ILC) et son détecteur, l'International Large Detector (ILD). Je décrirais également les travaux que j'ai effectué sur les détecteurs CMS-Ecal, Harpo et T2K-Wagasci.

Enfin, une grande partie de ces traitement *online* peuvent être accélérés par les techniques de machine learning, où la démarche de traitement est convertie en une approche statistique. J'ai été amené, au travers de trois expériences, à développer des méthodes originales dans le champ de la trajectographie. Ce problème, réputé difficile, trouve ici un traitement rapide et de bonne qualité, adapté au temps-réel comme à la reconstruction.

Ce mémoire est basé sur les travaux que j'ai eu la chance de réaliser au sein du laboratoire Leprince-Ringuet. j'y ai trouvé un contexte riche et stimulant, avec de nombreux projets variés et intéressants, stimulant ma passion pour la recherche scientifique. J'y ai aussi découvert un environnement humain particulièrement chaleureux, avec des collègues amicaux et prodigues de leur savoir, toujours prêts à m'encourager à emprunter les chemins de la créativité. Cet environnement stimulant m'a permis de travailler sur des réalisations valorisantes, des publications et des formations, dans un climat d'enthousiasme quasi-permanent. A travers la description de ces quelques réalisations, j'espère arriver à vous faire partager mon enthousiasme pour un domaine extraordinaire, scientifiquement, techniquement et humainement.

Chapitre 2

Pyrame



Pyrame est un *framework online* développé au laboratoire Leprince-Ringuet depuis 2012. Le projet a été initié pour la lecture et la configuration du calorimètre *SiW-Ecal* pour la collaboration Calice [8] décrit au chapitre 5. Pyrame est un *framework* souple et flexible mais néanmoins stable et performant. La version courante est la version 3.0.

L'opportunité de développer un nouveau *framework online* est née d'un besoin lié au développement de banc électroniques dans les expériences de physique des particules. Ce type de travail nécessite de pouvoir changer rapidement de configuration matérielle tout en conservant un système fonctionnel. Il est aussi indispensable que le système reste pérenne pendant tout le temps du développement qui s'étale fréquemment sur plusieurs années. Le paradigme ultime serait de pouvoir conserver le même système, de l'étape préliminaire du banc-test jusqu'à l'expérience proprement dite ou au moins des phases avancées de prototypage. Or dans le monde des logiciels *online*, peu de produits correspondent à ces conditions, se rangeant plus fréquemment dans deux catégories.

D'un côté des *frameworks* plutôt légers, faciles à programmer, flexibles et permettant des itérations de développement très rapide. L'exemple typique d'un tel *framework* est Labview. Son mode de programmation graphique et l'intégration de nombreux drivers en font un outil privilégié pour de nombreux électroniciens. L'inconvénient de ce *framework* est le manque de pérennité dans le temps des développements, essentiellement dû à la forme graphique de la programmation et au changement fréquent de versions incompatibles entre elles, doublé d'une stabilité en général assez faible.

De l'autre côté, on trouve des *frameworks* très structurés, fournissant de nombreux services, en général développés par des institutions comme le CERN. Un exemple parfait d'un tel *framework* est Xdaq, utilisé sur le détecteur CMS. Ce type de *framework* présente une stabilité exemplaire et une intégration profonde des outils. Malheureusement, sur une expérience en cours de développement, les outils nécessaires au bon fonctionnement d'un gros *framework* sont souvent trop lourds à déployer (nombreux serveurs, bases de données...).

C'est pourquoi il était nécessaire de disposer d'un outil qui réconcilie ces deux tendances, qui soit à la fois flexible et facile à utiliser mais en même temps, suffisamment stable pour fonctionner pendant des périodes de temps importantes. L'utilisation d'un langage de programmation standard (Python dans le cas de Pyrame) garanti de trouver des développeurs pour y faire les

développements nécessaires tout au long du projet, pérennisant ainsi les développements.

Les choix d'implémentations de Pyrame ont donc été dictés par les réalités du développement de prototype en laboratoire. Chaque fois que c'était possible, la flexibilité a été choisie, éventuellement au dépend des performances. Pyrame n'a pas été une démarche isolée, dans le même temps, un *framework* ressemblant, H4DAQ, a été développé au CERN.

Développé au départ dans le cadre du calorimètre électromagnétique Silicum-Tungstène du projet CALICE, il est vite devenu évident que le développement présentait des caractéristiques de généricité qui pouvait très facilement être transférées sur d'autre expériences. C'est ainsi que le projet original a été coupé en deux parties, l'une est devenue Pyrame, un *framework* générique, utilisé sur la plupart des expériences du LLR. L'autre partie est devenue Calicoes (CALICe OnlinE System) et contient la partie spécifique de ce projet. Le processus de séparation continue d'être actif et de nombreuses fonctionnalités, développées au départ dans le cadre d'une expérience spécifique finissent par être injectées dans Pyrame afin de pouvoir servir aux autres.

Pyrame a représenté un investissement modéré de la part du laboratoire. Typiquement du même ordre qu'un logiciel spécifique comme il s'en développe sur de nombreuses expériences. Il a été utilisé avec succès sur de nombreuses expériences, récapitulés par la table 2.1

Expérience	Détecteur	Localisation
Calice	*SiW-Ecal*	France et Japon
Calice	Banc Chip on Board	Corée
T2K	Wagasci	Japon
Harpo	Harpo	France
CMS	banc Skiroc 2 CMS	France
CMS	banc thermique	France
Pepites	Profileur de protons	France
CTA	Calibration de PM	France

Table 2.1 – Expériences utilisant Pyrame 3

J'ai initié le développement de Pyrame en 2012 et j'ai écrit l'essentiel de son coeur de système. En tant que coordinateur des développements online au LLR, j'ai également guidé mes collaborateurs de l'équipe online : Miguel Rubio-Roy, Floris Thiant et Lorenzo Bernardi pour le développements des systèmes et des matériels spécifiques à leurs expériences. Le projet Pyrame est également un projet très évolutif qui intègre en permanence de nouvelles fonctionnalités adaptées au besoins spécifiques des expériences du laboratoire.

Ce travail a donné lieu à deux publications dans la conférence "Computing for High Energy and nuclear Physics" en 2015 [36] un article décrivant globalement le fonctionnement de Pyrame et un autre article en 2016 [39] décrivant les mécanismes de distributions de données de d'analyse online.

2.1 Architecture globale

Pyrame est composé d'une collection de modules de commande qui implémentent chacun des fonctions différentes. Ces modules sont spécifiques à un matériel ou à un service purement logiciel.

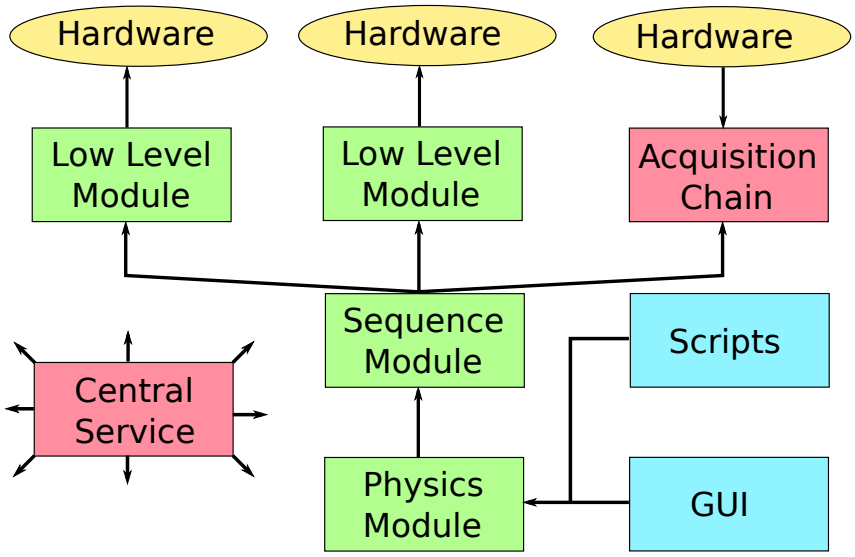


FIGURE 2.1 – Exemple d'architecture d'un système Pyrame.

La figure 2.1 [1] représente un système Pyrame typique. Les modules représentés en haut de la figure sont ceux qui sont au plus proche du matériel. Certains sont chargés de le configurer, d'autres d'en extraire des données, comme la chaine d'acquisition par exemple. Certains modules sont des séquenceurs qui permettent de coordonner le travail des modules de bas niveau. Au dessus, on retrouve les modules qui fournissent à l'utilisateur une abstraction logicielle de son détecteur. Celle-ci peut être pilotée à travers une interface graphique (GUI) ou un système de scripts qui permet d'obtenir des comportements complexes. Enfin, certains modules fournissent des services centraux qui seront utilisés par tous les autres modules. C'est le cas par exemple du module de configuration globale ou du module de partage de variables.

Une autre idée clé de Pyrame est qu'il offre la possibilité de répartir le système dans le réseau. En effet, les modules sont reliés entre eux par des connexions réseaux et rien ne s'oppose donc à ce que les modules soient sur des machines différentes, voire des sites différents. Cette approche donne une flexibilité totale, permettant de regrouper au sein d'un même système, des matériels qui se situent dans les lieux différents.

2.2 Module de commande

Le module de commande est le composant clé de Pyrame. C'est une base logicielle qui gère les aspects systèmes et protocolaires, laissant au développeur le soin de se concentrer sur la partie fonctionnelle. Celle-ci est exécutée dans une machine virtuelle qui permet de prendre en charge plusieurs langages. La figure 2.2 représente le fonctionnement interne du module de commande.

Un module de commande Pyrame est un processus indépendant qui écoute sur un port TCP qui lui est spécifique. Tant qu'aucune connexion n'est établie sur cette connexion, le module ne fait rien. Dès qu'une connexion est initiée, le module récupère la requête qui doit être formatée en XML comme suit :

```
<cmd name="test_module"><param>1</param></cmd>
```

1. Toutes les figures sont de l'auteur, sauf mention contraire.

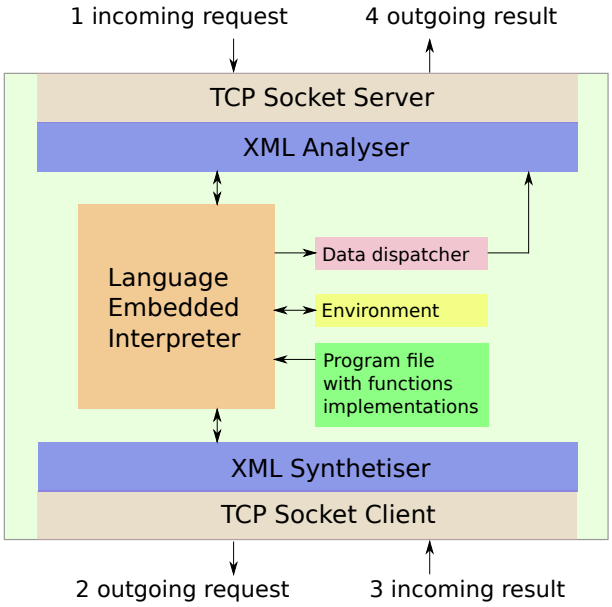


FIGURE 2.2 – Schéma de principe du module de commande. Les commandes sont décodées par les couches systèmes avant d'être exécutées dans la machine virtuelle. Celle-ci peut initier des connections vers d'autres modules pour un fonctionnement collaboratif.

Le module de commande extrait alors le nom de la commande ainsi que ses paramètres et va exécuter la fonction correspondante dans la machine virtuelle. La fonction doit lui retourner deux résultats. L'un numérique égal à 1 en cas de succès et 0 en cas d'échec. L'autre est une chaine de caractères pouvant être le résultat de la fonction ou un message d'erreur. Ces deux résultats sont renvoyés sur la même connexion TCP sous la forme xml suivante :

```
<res retcode=1>ok</res>
```

Dans le cas d'une erreur, on obtient un réponse du type :

```
<res retcode=0>unable to reset power supply <- unable to connect to
    GPIB converter</res>
```

Une fois la réponse obtenue, la connexion peut ensuite être fermée ou maintenue ouverte pour y envoyer d'autre commandes.

Le module de commande est configuré par un fichier de description XML qui indique le langage et le port à utiliser, ainsi que le fichier de code.

En fonction du langage sélectionné dans le fichier de configuration, le module de commande va lancer l'un ou l'autre de ses machines virtuelles. Plusieurs langages sont disponibles : C, C++, Python, Lua, Shell et R. Chacun d'entre eux présente des avantages et des inconvénients. Ainsi, le C doit être utilisé sur les modules demandant le plus de performance. C++ apporte également de la performance mais également le support de la librairie ROOT [5], très appréciée dans le domaine de la physique des particules. Python et Lua sont deux langages de scripts qui apportent une facilité de modification du code source par l'utilisateur. Shell est une facilité pour tout ceux qui utilisent un système Unix et veulent utiliser des scripts pré-existant à la version Pyrame de leur projet. Enfin R est un langage qui apporte une grande capacité d'analyse statistique des données et dont l'usage est également important dans la communauté. La table 2.2 récapitule les performances brutes du module de commande (*overhead*) pour les différends langages.

Langage	*Overhead* en micro-secondes
C	43
C++	47
Lua	55
Python	68
R	115
Shell	117

TABLE 2.2 – *Overhead* d'exécution de Pyrame 3 en fonction du langage fonctionnel utilisé. Il s'agit d'un appel à une fonction ne faisant rien permettant de mesurer uniquement le coût technique de l'opération.

Le module de commande fourni au code fonctionnel un mécanisme d'environnement qui lui permet de stocker ses données mais aussi de les synchroniser automatiquement avec le système de configuration globale du système. Cet environnement est une table interne du module de commande qui peut recevoir des variables typées et nommées associées à un nom d'objet. Ce mécanisme est relayé au niveau Python par une classe nommée pool qui implémente un pseudo-dictionnaire.

Enfin, le module de commande fourni également un mécanisme de diffusion des données en temps-réel appelé *dispatcher*. Ce mécanisme sera décrit en détail dans la section 2.7.

2.3 Le code fonctionnel

Au sein de la machine virtuelle du module de commande, le code fonctionnel est chargé de gérer un matériel ou un service au sein du réseau de module du système. Pour cela, il va s'appuyer sur un ensemble de fonctions que lui propose le module de commande et qui va lui permettre d'interagir avec les autres modules.

La table 2.3 représente l'ensemble des fonctions proposées par le module de commande au code fonctionnel. Elle sont liées à l'exécution de commandes sur d'autres modules, à l'utilisation de l'environnement, du *dispatcher*, ou encore à des exécutions retardées ou asynchrones. Ces fonctions sont disponibles dans tous les langages.

De nombreux modules sont disponibles dans Pyrame pour piloter la plupart des équipements standards que l'on trouve sur les bancs-test de notre laboratoire. La table 2.4 donne la liste précise de tous les modules disponibles.

Ces modules sont livrés avec une série de scripts shell qui permettent d'interagir avec eux en ligne de commandes. Ces scripts permettent d'accéder à toutes les fonctions des modules et de piloter très facilement un matériel. C'est le niveau le plus facile d'utilisation de Pyrame.

Si l'utilisation souhaitée est un peu plus complexe qu'un simple script, un connecteur (*binding*) existe en Python et en C/C++ qui permet d'automatiser les actions et de les associer à d'autres librairies (Root...) au sein d'un programme.

Nom	Description
submod_execcmd	exécute une commande sur un autre module
submod_setvar	crée une variable dans l'environnement
submod_getvar	récupère la valeur d'une variable de l'environnement
submod_delvar	supprimer une variable dans l'environnement
submod_gettype	récupère le type d'une variable de l'environnement
submod_newns	crée un nouveau domaine de nommage dans l'environnement
submod_checkns	vérifie qu'un domaine existe dans l'environnement
submod_delns	supprimer un domaine de l'environnement
submod_dumpns	donne la liste des variables d'un domaine
submod_listns	donne la liste des domaines de l'environnement
submod_setmyname	permet au module de fixer son propre nom
submod_get_port	permet de récupérer le numéro d'un port Pyrame
submod_exit	met fin à l'exécution du module
submod_getip	permet d'obtenir l'ip d'un autre module
submod_init_dispatcher	initialise le *dispatcher* de données
submod_flush_dispatcher	vide le *dispatcher* des anciennes données
submod_new_block	crée un nouveau bloc de données dans le *dispatcher*
submod_new_event	crée un nouvel évènement de données dans le *dispatcher*
submod_add_block	ajouter un nouveau bloc de données au *dispatcher*
submod_set_field_block	fixe la valeur d'un champs dans un bloc de données
submod_set_field_event	fixe la valeur d'un champs dans un évènement de données
submod_finalize_block	publie le bloc de données courant dans le *dispatcher*
submod_set_chans	modifie le schéma des données dans le *dispatcher*
submod_wakemeup	programme l'exécution d'une fonction dans le futur
submod_bgcontext	récupère le contexte d'exécution pour un traitement asynchrone
submod_sendres	envoie le résultat d'un traitement asynchrone

TABLE 2.3 – Commandes internes de Pyrame 3 permettant à un module d'exécuter des commandes sur d'autre module, de manipuler l'environnement, les ports réseau ainsi que le dispatcher de données.

2.4 Couches d'abstraction

Lorsque l'on écrit un code de haut niveau pour piloter une expérience, on aimerait pouvoir se référer à un système fonctionnel sans forcément spécifier le matériel que l'on utilise. Par exemple, on veut utiliser une alimentation électrique ou un générateur de pulsation mais le modèle exact de l'appareil effectivement utilisé est de peu d'importance.

Un tel mécanisme existe dans Pyrame et s'appelle les couches d'abstraction. Ce sont des modules Pyrame à qui l'on peut s'adresser pour effectuer des actions génériques, par exemple, allumer ou éteindre une alimentation. Grâce à ce mécanisme, on peut écrire un code complètement générique qui pourra être utilisé avec des appareils différents en changeant simplement la configuration de la couche d'abstraction.

Cette configuration se fait au travers de chaines de configuration. Ces chaines décrivent précisément le modèle de l'appareil que nous souhaitons utiliser mais aussi les modalités pour y accéder (adresse GPIB, adresse IP, convertisseur...)

Voici un exemple de chaine de configuration :

Type	Modèle
Bus	GPIB
Bus	Serial/USB
Bus	Raw Ethernet
Bus	TCP/IP
Oscilloscope	Lecroy ZI
Oscilloscope	Lecroy LT344
Gaussmètre	LakeShore 421
Gaussmètre	LakeShore 460
Amperemètre	Keithley 707b
Amperemètre	Keithley 6487
Amperemètre	Keithley 6517
Thermomètre	Omega ITHX
Sonde de pression	Pfeiffer TPG300
Axe motorisé	Thorlabs
Axe motorisé	Signatone WL350
Axe motorisé	Newport ESP301
Alimentation	Agilent e3631A
Alimentation	Agilent N6700B
Alimentation	CAEN SY527
Alimentation	Hameg 4030
Alimentation	Iseg HQ
Alimentation	Lambda gen8
Pompe à vide	Pfeiffer TC110
Robot sonde	Wavelink 350
Détecteur de particule	Omega Skiroc2
Détecteur de particule	Omega Spiroc2b/2d
Détecteur de particule	Omega Maroc3
Détecteur de particule	Omega Skiroc2CMS
Détecteur de particule	Omega Easyroc
Générateur de Pulsation	Agilent 33200
Générateur de Pulsation	Agilent 33500
Générateur de Pulsation	Agilent 81160
Générateur de Pulsation	Tektronix AFG 3000
Service	Chaine d'acquisition multi-media
Service	Module de configuration globale
Service	Module de représentation graphique des données
Service	Constructeur d'évènements
Service	scriptage des runs
Service	Base de données des runs
Service	Stockage des données
Service	partage de variables

TABLE 2.4 – liste des matériels et services disponibles dans Pyrame 3

```
mm_ki_6517(bus=gpib(bus=tcp(host=10.220.0.70),dst_addr=27))
```

Elle précise que l'appareil que l'on souhaite utiliser est une alimentation Keithley 6517 qui utilise le bus GPIB. Ce bus est interfacé avec un convertisseur disponible à l'adresse IP 10.220.0.70.

L'adresse GPIB de l'alimentation est le 27.

Nous pouvons ainsi écrire notre code en utilisant les fonctions du module "ps", la couche d'abstraction pour les alimentations électriques. Si nous désirons faire tourner ce même code sur une autre alimentation, seule la chaine de configuration est à modifier.

Des couches d'abstractions sont disponibles pour les alimentations électriques, les axes motorisés, les appareils de mesure (ampèremètres, thermomètres...) ainsi que les appareils électroniques (atténuateurs, générateurs de pulsations...).

2.5 Configuration globale

Lorsque l'on utilise un système d'acquisition de données en temps-réel, il est absolument indispensable de toujours savoir quelle est la configuration exacte de son matériel de mesure. C'est pourquoi, dans Pyrame, un mécanisme global a été implémenté afin de collecter toutes les informations de configuration et de les associer aux données.

Afin de mener à bien ces tâches, un module de configuration est disponible dans le *framework*. Son utilisation n'est pas obligatoire mais elle simplifie tellement la configuration du système qu'elle est maintenant systématique sur l'ensemble des projets utilisant Pyrame.

Le principe est de fournir un fichier XML qui décrit précisément le matériel utilisé ainsi que les paramètres qui seront transmis aux modules pour leur différentes phases de fonctionnement.

Le fichier de configuration est une collection d'objets structurée sous une forme hiérarchique. Chaque objet possède un type, un nom et des paramètres. Les types d'objets peuvent être de différente nature :

- Les domaines sont des objets indiquant sur quelle machine tournent les modules de commande. Ils prennent un paramètre indiquant l'adresse IP à laquelle sont accessibles ces modules.
- Les autres objets correspondent à des modules de commande spécifique. Leur type est égal au nom du module. Ils prennent des paramètres correspondant aux arguments de leurs fonctions.

Voici un exemple de fichier de configuration

```
<config name="test">
  <domain name="localhost">
    <param name="domain_ip">127.0.0.1</param>
    <ps name="alim_test">
      <param name="ps_conf_string">mm_ki_6517(bus=gpib(bus=tcp(host
         =10.220.0.70),dst_addr=27))</param>
      <param name="ps_cur_lim">5</param>
    </ps>
  </domain>
</config>
```

L'objet "config" dont le nom est "test" désigne la racine de la configuration et ne correspond à aucun matériel. Le domaine "localhost" indique que tous les modules tournent sur la machine locale car son adresse IP est 127.0.0.1 (adresse réseau locale). L'objet de type "ps" dont le nom

est "alim_test" est une alimentation électrique. Sa chaine de configuration donne les éléments permettant de s'y connecter. Le paramètre "ps_cur_lim" donne la limite en courant que l'on souhaite appliquer sur l'alimentation.

Sur la base de cette configuration et d'un fichier contenant toutes les valeurs par défaut de tous les paramètres, le module de configuration construit une représentation interne de l'arbre de configuration.

Le module de configuration peut ensuite être utilisé pour diffuser les paramètres à l'ensemble des composants du système. Pour cela, on utilise un système, dit de vagues de configuration. On choisit un nom de vague, par exemple init et on demande au module de configuration de l'appliquer. Le module parcourt son arbre de configuration et pour chaque objet, se connecte sur le module correspondant et exécute la fonction correspondant en trouvant de lui-même les valeurs des paramètres nécessaires.

Comme on l'a vu, le module de commande fournit un environnement dans lequel sont stockées toutes les variables des objets. A chaque fois que l'un d'entre eux reçoit un paramètre via le module de configuration, il le marque comme synchronisable. Par la suite, à chaque fois que le paramètre sera modifié, l'environnement se connectera de lui-même au module de configuration afin de le tenir à jour. Ainsi, à tout moment, le module de configuration contient l'entièreté des paramètres utilisés. Naturellement, il peut sauvegarder sa représentation interne dans un fichier XML qui pourra être réutilisé plus tard pour ramener le détecteur dans cette configuration.

Dans la plupart des applications utilisant Pyrame, cette fonctionnalité est couplée à la prise de données. A chaque fois que l'on lance une acquisition, une copie complète de la configuration est écrite sur le disque avec les données. Ainsi, on peut toujours retrouver dans quelles conditions expérimentales les données ont été prises. Si celles-ci contiennent des résultats que l'on souhaite reproduire, on peut, en un instant, reconfigurer le détecteur comme il était à ce moment là et refaire une prise de données.

2.6 Chaine d'acquisition

Tout système de traitement de données doit naturellement commencer par un système d'acquisition. Pyrame est donc doté d'une chaine d'acquisition multi-média à haut débit. C'est un composant à très faible latence, écrit entièrement en C pour des performances optimales et qui permet d'atteindre une bande passante de 4Gb/s. La figure 2.3 montre le fonctionnement schématique de la chaine d'acquisition.

Afin de permettre une acquisition hétérogène, très courante sur nos expériences, la chaine d'acquisition possède un système de greffon. Chacun d'entre eux implémente quelques fonctions de base, comme l'initialisation et l'acquisition d'un bloc de données. Ce greffon est interrogé régulièrement pour alimenter le système.

Une fois acquise sous la forme d'un bloc, la donnée est transférée à un autre greffon (nommé uncap) qui lui, dépend du format utilisé. Ce greffon a pour tâche de vérifier l'intégrité de la données, par exemple en surveillant le numéro de séquence des paquets. Ensuite, il doit discriminer le type du paquet, qui peut être de données ou de contrôle. En effet, certaines cartes électroniques mélangent leurs flux de données et de contrôle et il est nécessaire de les trier. Enfin, le greffon va devoir extraire la donnée du paquet et éventuellement, la trier dans différents flux de données.

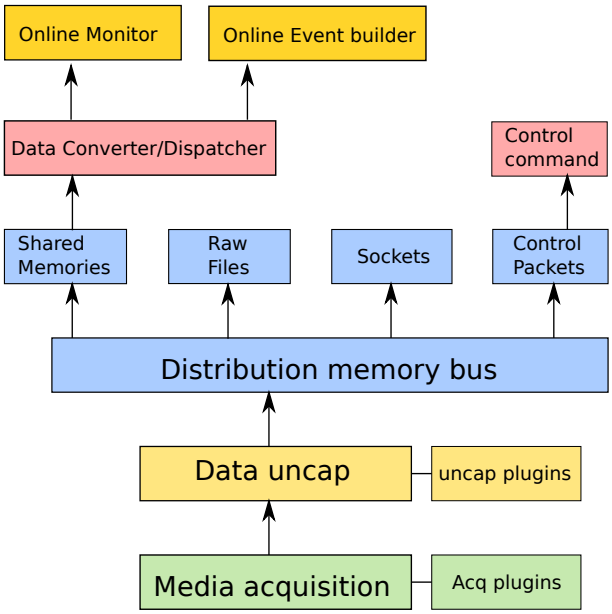


FIGURE 2.3 – Schéma de principe de la chaine d'acquisition. Un système de greffon permet d'acquérir et de traiter des données de plusieurs médias en même temps. Le bus de distribution permet de mettre les données à disposition selon plusieurs modalités. Des décodeurs temps-réel permettent d'implémenter facilement des moniteurs.

Une fois les données extraites et triées, elles sont insérées dans le bus de distribution. Celui-ci est un mécanisme logiciel qui permet de distribuer ces données suivant plusieurs modalités :

- Les paquets de contrôle sont placés dans un buffer circulaire. Les applications utilisatrices de la chaine d'acquisition peuvent les y retrouver au moyen d'une interface de recherche par identifiant.
- Les données sont stockées dans des fichiers. On utilisera un disque local pour ce stockage afin de limiter la latence d'accès au disque. Ces données pourront être recopiées sur un autre support (réseau, bande...) par un script appelé automatiquement après la période d'acquisition.
- Les données sont mises à disposition au travers d'un système de mémoires partagées, permettant un traitement *online*, local mais de haute performance.
- Les données sont également mises à disposition via des connexions TCP. Ce mécanisme permet un traitement *online* distant. Si l'application distante est trop lente ou si le réseau n'est pas assez rapide, l'utilisation de ce service peut mener à une surcharge de la chaine et à une perte de données.

Un mécanisme de conversion *online* est disponible directement dans la chaine d'acquisition. Encore une fois, un système de greffon a été choisi. Celui-ci reçoit les données par flux, les décode et les met à disposition à travers un *dispatcher* de données. Ce mécanisme est décrit avec précision dans la section suivante.

2.7 Dispatcher de données

Le coeur logiciel qui permet à Pyrame de faire du traitement de données *online* est une nouveauté de la version 3.

Le principal problème lorsque l'on veut traiter de la données *online* est d'accorder la vitesse de traitement avec la vitesse de collecte. En effet, l'acquisition de données consiste toujours à remplir des buffers et à les déverser dans des médias de stockage ou de traitement.

Comme nous l'avons vu précédemment, la chaine d'acquisition de Pyrame dispose de plusieurs modalités de mise à disposition des données. Le problème est que si un consommateur s'y connecte et ne consomme pas assez vite les données, il y a un risque de saturation des buffers. Ceux-ci vont se remplir et la chaine finira par perdre des données. C'est ce que nous appellons la contrainte temps-réel.

Cette contrainte temps-réel peut être facilement contournée par un mécanisme de *sub-sampling*, c'est-à-dire que seule une partie des données sont distribuées. Le problème d'un tel mécanisme est le réglage de son débit. Si on le règle trop bas, on gâche une potentialité d'analyse temps-réel et si on le règle trop haut, on retrouve le problème d'explosion de buffer.

C'est pourquoi le mécanisme de diffusion de données de Pyrame est adaptatif. Il va stocker les données en quantité limitée et les mettre à disposition des consommateurs pendant un temps règlable dépendant de la taille des buffers et du débit d'arrivée des données.

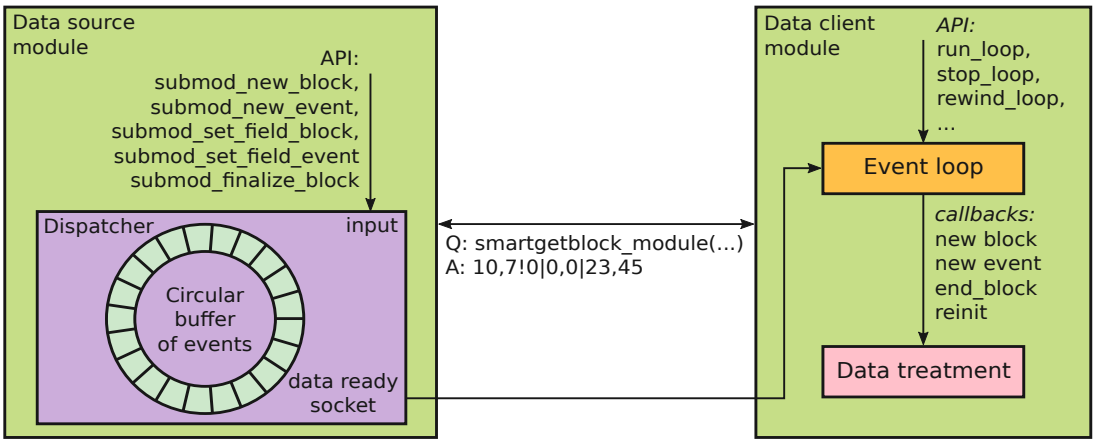


FIGURE 2.4 – Schéma de principe du *dispatcher* de données. Les données sont stockées au fur et à mesure de leur production dans un buffer circulaire. Le client récupère les données à la vitesse à laquelle il est capable des les traiter. S'il est trop lent, il n'obtiendra qu'une partie des données : c'est le sub-sampling.

La figure 2.4 montre le principe de fonctionnement du *dispatcher* de données. Le module qui les produit dispose d'une *API* pour créer des blocs contenant des évènements. Ces blocs et ces évènements contiennent des champs où se trouve l'information effective. Ces blocs sont ensuite munis d'un identifiant unique puis rangés dans un buffer circulaire. Ce processus de rangement s'effectue à la vitesse de production des données. Lorsqu'il n'y a plus de place disponible dans le buffer, le plus vieux des blocs est effacé et remplacé par le nouveau.

Du côté du client, un autre mécanisme, nommé boucle d'évènements (event loop) prend en charge l'arrivée des nouvelles données. A chaque fois qu'un nouveau bloc ou évènement est disponible, la

boucle appelle une fonction (callback) en lui fournissant les données correspondantes. La boucle attend ensuite la fin du traitement de tous les évènements du bloc avant d'en télécharger un nouveau. Ainsi, si le traitement est long, il est possible que la boucle ne trouve pas le bloc suivant immédiatement celui qu'elle vient de traiter car il a déjà été remplacé par un nouveau : c'est le sub-sampling.

Ce sub-sampling est adaptatif car un processus rapide aura le temps de télécharger tous les blocs alors qu'un processus plus lent utilisera le sub-sampling à la fréquence précise de ses capacités de traitement. Avec ce mécanisme, on est certain d'optimiser le débit du traitement.

La structuration des données en blocs et évènements est particulièrement adaptée à la physique des particules. Un évènement est défini par des champs de localisation spatiale, temporelle, ainsi que des mesures effectuées. La structuration en bloc garantit une certaine équité statistique dans les données qui sont traitées, on parle de sub-sampling équitable (fair sub-sampling). Un bloc est censé être une entité complètement indépendante de tout autre bloc.

L'autre avantage des blocs est qu'ils disposent de champs spécifiques globaux (que nous appelons les propriétés) et qui évitent la redondance d'information dans les évènements. Ainsi, si un bloc correspond à une étiquette temporelle donnée, on stockera cette information, non dans les évènements mais dans le bloc lui-même.

Aujourd'hui, aucun format de données ne fait consensus dans la communauté. Un travail est en cours avec les autres acteurs de la discipline pour faire émerger une proposition de format binaire performant et expressif. En attendant, nous avons choisi un simple format ASCII pour faire la démonstration de principe de fonctionnement du *dispatcher* de données.

Une autre particularité du *dispatcher* de données est qu'il publie son schéma de données. Ainsi, au sein de la boucle d'évènements, on peut extraire les informations des blocs et des évènements par leur nom, évitant une implémentation qui soit trop dépendante du format des données. Dans l'exemple suivant, on extrait un champ d'énergie simplement en donnant son nom "en" :

```
energy=atof(get_event_field(e,"en"));
```

Ce mécanisme de *dispatcher* de données est implémenté au sein du module de commande. Ainsi, tout module peut devenir une source de données. Afin de pouvoir s'y connecter sans configuration compliquée, chacune de ces sources s'enregistre auprès d'un annuaire centralisé sous un nom qui lui est propre. Ainsi, tout client peut facilement s'y connecter en connaissant ce nom et produire à son tour des données. Les producteurs de données peuvent ainsi former une chaine de traitement.

Le mécanisme de diffusion est basé sur les connexions TCP, ainsi rien n'empêche que la boucle d'évènement s'exécute sur une autre machine (ou même sur un cluster de machine comme dans le cas du constructeur d'évènements). A travers cette connexion TCP, la boucle d'évènement envoie des commandes Pyrame pour récupérer les données.

Ce mécanisme de diffusion de données est couplé avec la chaine d'acquisition au travers d'un module appelé converter. Il permet de faire de la conversion temps-réel des données brutes recueillies dans la chaine, transmises à travers les mémoires partagées. Le converter effectue la conversion dans le modèle bloc/évènements et les met à disposition des consommateurs via son *dispatcher* de données en supprimant ainsi les problèmes liés à la contrainte temps-réel. Ce module est totalement générique et fournit à ses plugins une abstraction pratique pour le décodage. Il représente la donnée comme un ruban infini sur lequel on peut lire et se déplacer en avant

ou en arrière. On peut également le lire jusqu'à tomber sur un motif attendu. Afin de préserver la mémoire, il fourni également une fonction de coupure, semblable à des ciseaux virtuels, permettant de couper le ruban lorsque les données qui précèdent peuvent être abandonnées.

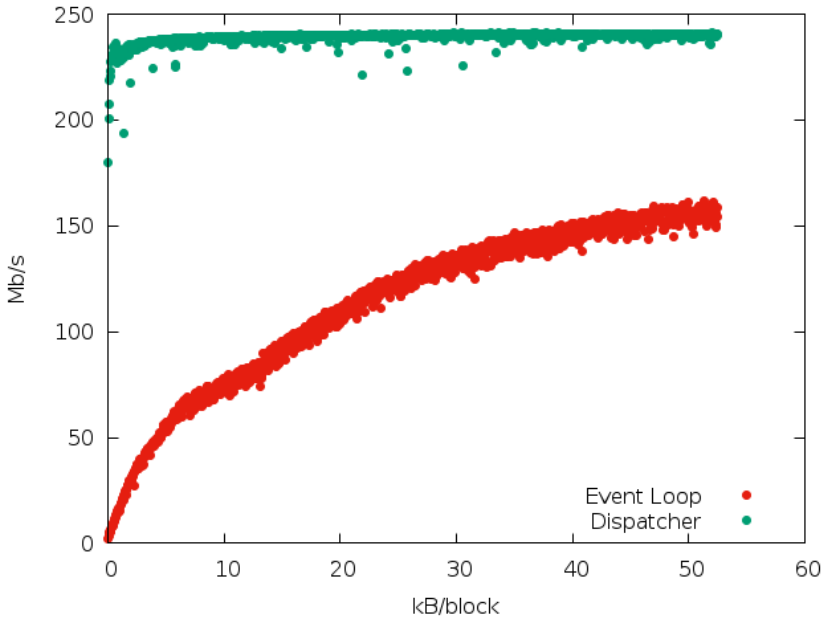


FIGURE 2.5 – Performances du *dispatcher* de données

La figure 2.5 montre les performances du système avec le format ASCII. Comme on le voit, les performances du *dispatcher* sont de l'ordre de 240 mega-bits par seconde indépendamment de la taille des blocs. Cela peut paraître faible comparé au 4Gb/s de la chaine mais cela correspond à un seul flux. Pour la partie boucle d'évènements, par contre, la performance dépend de la taille des blocs car l'*overhead* réseau est important pour les petits paquets et le parsing du format texte est linéaire. Lorsque la taille des paquets devient raisonnable (vers 20 kilo-octets), la performance atteint 100 mega-bits puis augmente légèrement jusqu'à atteindre un plateau aux alentour de 150 mega-bits par seconde. Nous travaillons aujourd'hui sur l'implémentation d'un format binaire ayant les même propriétés qui devrait considérablement augmenter les performances.

2.8 Logger

Un autre mécanisme permet d'alimenter Pyrame en données : le logger. Dans un système de détection, on s'appuie parfois sur des appareils comme des oscilloscopes numériques ou des appareils de mesures packagés, qui renvoient leurs données uniquement sur sollicitation. Les modules Pyrame spécifiques gérant ces modules implémentent les fonctions permettant d'interroger les appareils, mais ils n'intègrent pas d'ordonnanceur permettant de programmer l'interrogation régulière et l'intégration des résultats.

Le logger est un module complexe qui intègre un ordonnanceur permettant d'exécuter des commandes Pyrame selon des timings précis et complètement paramétrables. Le principe de configuration de ce module est basée sur le concept de variable. Une variable est un nom auquel correspond une périodicité, un module, une commande et des paramètres. Dès le moment où l'acquisition est lancée, le logger doit s'efforcer d'exécuter la commande sur le module avec la

périodicité voulue.

A chaque étape de lecture de données, l'ordonnanceur lance autant de processus légers (threads) que de commandes à exécuter. Cela permet d'éviter qu'une commande trop lente ne ralentisse les autres. Une fois, toutes les valeurs récupérées, le logger va les publier selon deux modalités. Tout d'abord, il utilise un *dispatcher* de données afin de mettre les valeurs à disposition des programmes d'analyse *online*. En même temps, il publie les données dans un fichier CSV, ce qui le rend compatible avec de nombreux logiciels dont ROOT, R et Gnuplot. Ce dernier fournit une fonctionnalité d'exécution de script qui permet en quelques lignes de code, d'obtenir une GUI minimaliste qui affiche un ou plusieurs plots et les met à jour en temps-réel comme on le voit sur la figure 2.6.

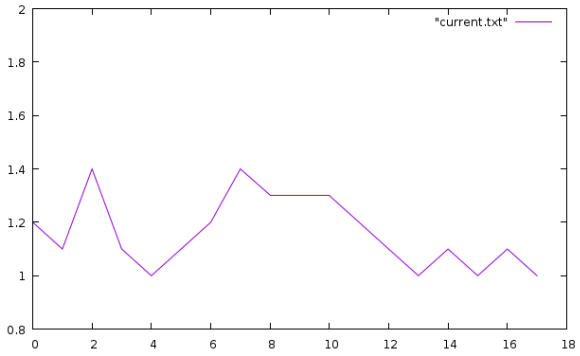


FIGURE 2.6 – Exemple de micro-GUI avec gnuplot

2.9 Stockage des données

On l'a vu, il existe différentes façons d'acquérir des données dans le système Pyrame. Ces différents modules produisent des fichiers qui contiennent les données brutes. Il est nécessaire pour le bon fonctionnement d'un système global d'unifier les temps et les espaces utilisés pour faire ces acquisitions.

Ainsi, Pyrame implémente un modèle de structuration de données basé sur les concepts de run et d'acquisition. Une acquisition est une durée durant laquelle on effectue des mesures. Un run est un ensemble d'acquisitions qui ont une finalité commune.

Dans une acquisition standard, la notion de run et d'acquisition se rejoignent et chaque run contient une unique acquisition. Mais dans le cas de run complexes, par exemple des runs de calibration, on effectuera de nombreuses acquisitions séparée par des reconfigurations du système.

Le module de configuration permet de synchroniser le début des acquisitions par son système de vagues. Ainsi, la chaine d'acquisition et le logger implémentent les fonctions start_acq et stop_acq qui permettent respectivement de lancer ou de stopper une acquisition. Une fois celle-ci stoppée, les données sont stockées sur le disque aux emplacements spécifiques à ces modules.

Un module spécialisé, nommé storage, permet de regrouper les données au même endroit, d'effectuer des traitements systématiques et de signaliser sur ces évènements. Pour cela, les producteurs de données s'enregistrent au niveau du module storage, pour lui indiquer leur emplacement de stockage. Ils utilisent en général des disques locaux pour des raisons de performance.

Lorsque l'acquisition démarre, le module storage va créer un emplacement dédié dans lequel il stockera les données à l'issu de l'acquisition. Il génèrera un fichier de log contenant l'heure de départ et peut publier ces informations sur un *elog* (Electronic Logbook). On peut également lui donner des paramètres utilisateurs qui seront stockés avec les données. Le module se met ensuite en attente de la fin de l'acquisition.

Lorsqu'un module a terminé son acquisition, il exécute la commande mark_as_finished sur le module storage. Celui-ci décrémente son compteur de producteurs. Lorsque ce compteur atteint 0, le module va effectuer toutes les actions liées à la fin d'acquisition :

- Finaliser le fichier de log en y stockant l'heure de fin ainsi que des statistiques accumulées pendant l'acquisition.
- générer une entrée dans le *elog* contenant les statistiques
- transférer les données vers un emplacement unique éventuellement disponible sur le réseau
- Appliquer un script de conversion et / ou de traitement sur l'ensemble des données

Ainsi à la fin de l'acquisition, les données sont regroupées en un endroit unique et éventuellement converties dans un format pour l'analyse.

2.10 Reconfiguration dirigée par les données

Une première approche de la gestion des données est celle que l'on peut appeler semi-*online*. Le concept, c'est de faire des acquisitions de données segmentées. A la fin de chacune de ces acquisitions, on effectue des traitements automatiques qui amenent à une reconfiguration adaptée. On utilise ce genre de traitement dans les phases préliminaires aux prises de données réelles, typiquement les phases de calibration ou d'ajustement dynamique des paramètres.

Nous avons dit que le module storage déclenche l'exécution d'un script à la fin de chaque acquisition. Celui-ci peut tout à fait mener des actions sur la base des données collectées. Ainsi il est possible d'analyser les données et d'en tirer des conclusions sur le fonctionnement du détecteur. Le script peut alors reconfigurer le détecteur en s'adressant aux modules concernés. Grâce au mécanisme de synchronisation de l'environnement, les nouveaux paramètres sont automatiquement intégrés dans le module de configuration et seront pris en compte lors de la prochaine acquisition. C'est ce que nous appelons la reconfiguration dirigée par les données (data driven reconfiguration).

2.11 Gestion de systèmes complexes

La structuration hiérarchique de Pyrame permet de gérer des systèmes d'une complexité arbitraire. En effet, on peut ajouter autant de modules de séquence que nécessaire pour gérer plusieurs niveaux et les emboiter les uns dans les autres. Toutefois, au delà de cette dépendance causale, il est parfois nécessaire de disposer de certains services centraux pour assurer un fonctionnement harmonieux de l'ensemble.

2.11.1 Module de variables

Le module de variables (varmod) est l'un de ces mécanismes. Son rôle est de permettre un partage d'information entre les différents modules du système. Pour cela, il implémente de nombreuses fonctions spécifiques qui sont accessibles par tous les modules. Comme tout module central, son utilisation est possible uniquement si tous les modules connaissent son emplacement dans le réseau. C'est pourquoi son utilisation est rendue plus simple si elle est combinée avec le module de configuration.

Varmod permet de partager les objets suivants :

- variables textes
- variables entières
- variables flottantes
- jetons
- sources de données

Afin d'éviter les conflits de noms, les variables sont regroupée au sein d'espaces de nommage. Aucune sécurité particulière n'est disponible dans ce module, qui sert d'annuaire public pour tous les modules et il est de la responsabilité du programmeur d'assurer qu'il n'y a pas de conflit de nommage.

Les variables textes sont des chaînes de caractères sans limitation de longueur. L'*API* du module permet de créer ou de supprimer de telles variables, mais également de leur appliquer une opération de concaténation.

Les variables entières ont également leurs opérations spécifiques, addition, soustraction, multiplication et division entière. L'intérêt de telles opérations réside dans l'accumulation de statistiques par des modules différents. Par exemple, les décodeurs *online* vont ajouter leurs statistiques à des variables entières qu'ils partagent entre eux (dans un domaine de nommage spécifique). L'interface graphique pourra interroger le module de variables pour obtenir cette statistique consolidée et l'afficher.

Les variables flottantes implémentent les mêmes opérations que les variables entières mais sur les nombres réels.

Les jetons sont des objets qui permettent de garantir une exécution en exclusion mutuelle entre plusieurs modules. Au moment de l'exécution de leur section critique, les modules demandent le jeton. S'il l'obtiennent, il peuvent exécuter leur section critique. Dans le cas contraire, ils doivent attendre et refaire une demande ultérieure. Notez que pour des raisons de synchronisme du protocole, le module de variables ne peut rester en attente jusqu'à ce que le jeton soit disponible. Il ne garantit donc aucune équité sur l'attribution des jetons.

Enfin, les sources de données sont les coordonnées des serveurs qui distribuent de la donnée en temps-réel ainsi qu'il est décrit dans la section 2.7.

2.11.2 Contrôle d'exécution par script

Sur les détecteurs, certaines tâches sont scriptées car elles durent longtemps. Elles font intervenir de nombreuses étapes et peuvent être assez compliquées. C'est pourquoi, fournir un environnement de script performant est un véritable enjeu pour un système d'acquisition de données. Au

sein de Pyrame, cette fonctionnalité est assurée par deux modules : le contrôleur d'exécution (run control) et le scripteur (run control script).

Le principe est de fournir un système qui abstrait au maximum le système afin que les scripts soient faciles à écrire et à maintenir. Ceci est assuré dans Pyrame par une surcouche Python qui permet d'accéder facilement au système. Les fonctions de cette surcouche sont détaillées dans le tableau 2.5.

Nom	Fonctionnalité
declare_param	déclaration d'un paramètre avec valeur par défaut
int_sleep	pause interruptible
rc_exec	exécute une commande sur n'importe que objet du système
load_config_file	charge un fichier de configuration
save_config_file	sauve la configuration courante dans un fichier
transition	exécute une vague de configuration
new_run	crée ou recrée un run
new_acq	crée ou recrée une acquisition
set_param_run	ajoute un paramètre au run
set_param_acq	ajoute un paramètre à l'acquisition

TABLE 2.5 – liste des fonctions de script du contrôle d'exécution

D'autre part, le contrôleur d'exécution doit pouvoir lancer le script mais aussi l'arrêter en cours de route. C'est pourquoi toutes les fonctions sont interruptibles. Un système de communication entre le script et le contrôleur permet une signalisation continue entre les deux processus permettant de relayer les ordres de l'utilisateur en temps-réel.

Enfin, une interface graphique doit pouvoir s'interfacer facilement avec le contrôleur mais aussi avec le script pour demander les bons paramètres ou afficher l'état d'avancement du script. A cette fin, le contrôleur implémente des commandes permettant de connaitre les scripts exécutables, les paramètres correspondants, de lancer ou de stopper un script et de connaitre son état d'avancement.

2.12 Interfaçage avec le monde extérieur

Un système tel que Pyrame peut fonctionner seul, pilotant l'ensemble des composants du système. Toutefois, il est fréquent qu'il soit nécessaire de l'interfacer avec d'autres outils. Afin de faciliter ces opérations, nous avons implémenté plusieurs connecteurs (*bindings*) dans Pyrame.

Un *binding* Pyrame est composé de quelques fonctions qui permettent d'abstraire la logique interne consistant à se connecter au module souhaité, à lui envoyer la commande en XML puis à analyser le résultat.

En standard, Pyrame fournit un *binding* pour le langage C, le C++, le Python et le shell. Cela permet de l'interfacer avec la plupart des autres systèmes d'acquisition et de contrôle de données (SCADA). En particulier, nous avons pu l'utiliser avec Tango, OPC Unified Architecture et XDAQ [24], particulièrement utilisés dans le domaine de la physique des hautes énergies.

La légèreté du protocole permet également d'exécuter Pyrame sur des plateformes embarquées. Le *framework* Pyrame a été écrit pour Linux, bien que la plupart de son code soit compatible POSIX. Il est donc possible de l'exécuter sur une plateforme ARM tournant sous Linux. Ainsi

tous les Raspberry-Pi ou équivalents permettent d'exécuter le module de commande, combinant la légèreté de la plateforme et l'intégration dans le système complexe et hiérarchique géré par Pyrame.

Pyrame a également été porté pour la plateforme Arduino, au moyen d'une librairie C++ spécifique. En effet, le module de commande, trop complexe, ne peut être porté directement sur Arduino. C'est pourquoi, j'ai écrit une version spécifique qui intègre un micro-parser XML et une couche réseau. Cette implémentation est très légère et compile pour tous les arduinos y compris les plus petits. Cela permet de générer directement des signaux électroniques à travers les ports numériques et analogiques du microcontrôleur.

Enfin, un *binding* pour javascript a été réalisé qui permet d'écrire des interfaces graphiques utilisables à travers le réseau, au moyen d'un simple navigateur. Ce *binding* permet d'exécuter des commandes sur tous les modules et d'en afficher les résultats au sein de l'interface. Nous verrons plusieurs exemples dans le chapitre 5.

2.13 Applications

Dans cette section, je vais évoquer les applications les plus basiques de Pyrame. Ce sont des bancs de test du laboratoire que nous utilisons pour qualifier les composants de nos détecteurs. Les expériences en tant que telle donneront lieu à des chapitres dédiés.

2.13.1 Banc de cartographie magnétique

Ce banc est un système qui permet d'établir des carte de champs magnétiques de façon automatique. Il a été développé en collaboration avec Miguel Rubio-Roy et Lorenzo Bernardi dans le cadre du projet Galop, qui étudie l'accélération des particules par laser-plasma.

L'expérience, représentée sur la figure 2.7 est composée d'un Gaussmètre dont on voit la sonde Hall (la tige noire horizontale). Celle-ci est fixée sur un dispositif mécanique composé de trois moteurs pilotables permettant de la déplacer sur les 3 axes. Le problème consiste à déplacer cette sonde dans l'espace, pour effectuer des points de mesure du champ, sans heurter l'aimant cible (en bleu) ni les autres équipements.

Comme nous l'avons déjà évoqué, Pyrame dispose de drivers pour les moteurs ainsi qu'une couche d'abstraction adaptée. Pour mener à bien ce projet, nous avons écrit un module nommé "paths", qui permet de discrétiser l'espace et d'y construire des chemins, basés sur une description précise des zones accessibles ou non.

Une interface Web permet de piloter l'ensemble. Celle-ci permet la saisie des éléments complexes de géométrie et de lancer l'acquisition, une fois défini le chemin dans l'espace. L'acquisition va alors remplir des fichiers textes contenant les valeurs du champs magnétique avec les coordonnées correspondantes.

2.13.2 Banc Laser

Le banc laser est un équipement permanent du laboratoire. Il permet d'envoyer un faisceau laser mobile sur des surfaces de détection, permettant ainsi de les caractériser. Ce travail a été effectué

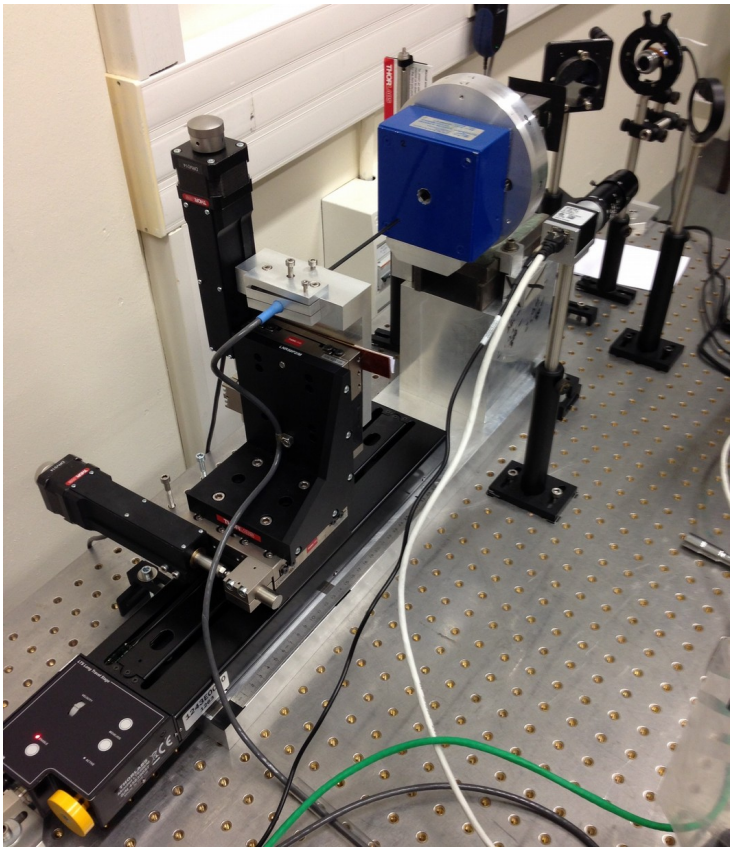


FIGURE 2.7 – Banc de cartographie magnétique. La sonde Hall est déplacée par les axes motorisées pour mesurer le champ au sein de l'aimant (en bleu)

en collaboration avec Miguel Rubio-Roy.

Comme on le voit sur la figure 2.8, la tête laser est montée sur une platine motorisée qui permet de la déplacer dans l'espace. Le dispositif peut être complété si besoin avec un second moteur pour l'axe longitudinal.

Le faisceau laser, dont la puissance et la longueur d'onde peuvent être modulées par un jeu de têtes interchangeables, est déplacé au dessus d'un système de détection. Sur la photo, c'est un scintillateur qui est testé.

Un photo-multiplicateur est accolé au scintillateur et délivre des signaux qui sont lus par un oscilloscope digital que Pyrame interroge pour acquérir les données via le protocole Lecroy VICP. L'oscilloscope, en mode histogramme, va accumuler des mesures pendant un temps d'intégration dépendant du photo-multiplicateur. Pyrame coordonne les déplacement des moteurs, les tensions des alimentations et la récupération de ces histogrammes.

Avec les données obtenues, on peut construire des graphiques de réponses du scintillateur en fonction de l'exposition au rayonnement. La partie de droite de la figure 2.8 montre un tel plot. Le but, sur ce cas précis, étant de déterminer les effets de bord du détecteur.

Le pilotage de ce banc se fait à travers une série de scripts car il est presque toujours nécessaire d'ajuster le comportement fin du système : amplitude du déplacement, mode de lecture, post-traitement... C'est pourquoi une interface graphique n'apporte pas assez de souplesse pour une

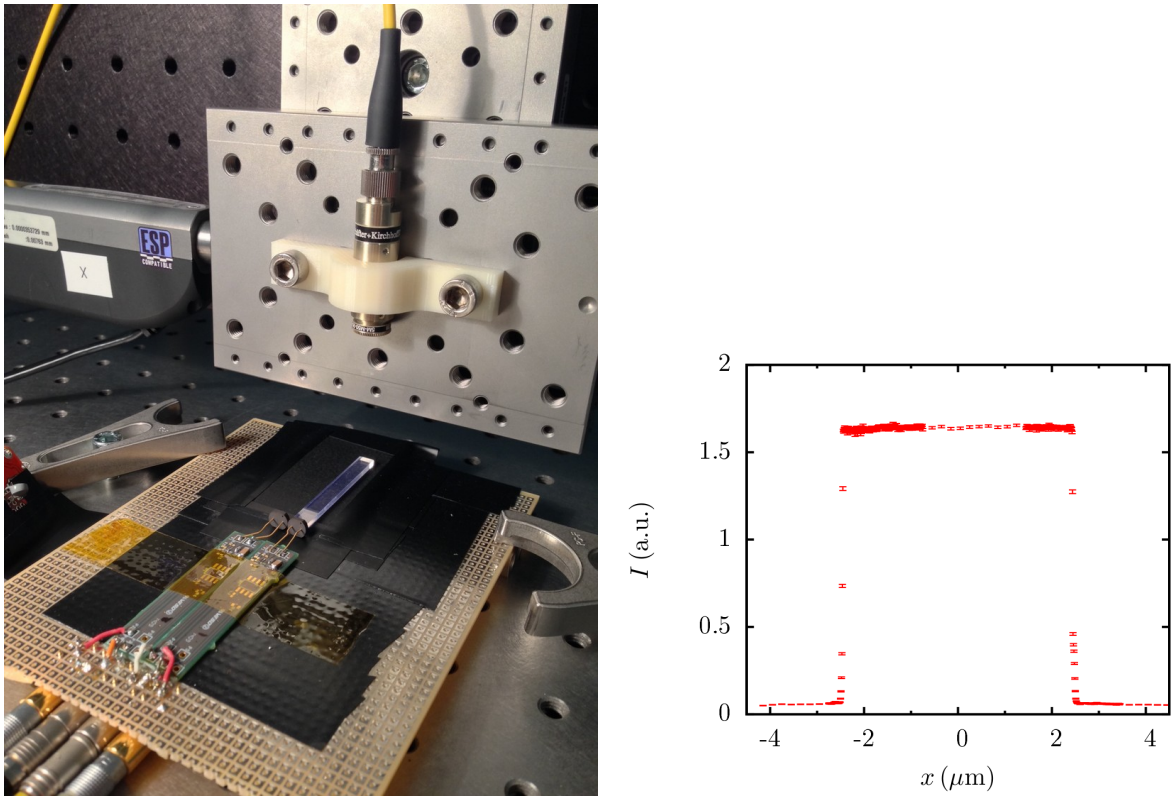


FIGURE 2.8 – Banc laser et caractérisation de photo-multiplicateur

telle application.

2.13.3 Banc Captinov de qualification de *wafers*

Le banc de qualification des *wafers* nommé “Captinov” est un équipement qui a été financé par le labex P2IO. Il est installé au LAL à Orsay. La partie gauche de la figure 2.9 montre une vue générale du banc.

C’est un système qui permet de faire des tests sous pointe de galettes de silicium (*wafers*). Il est basé sur un robot-sonde de type Wavelink 350. C’est une machine qui permet de définir précisément des positions de sondes sur les *wafers* puis qui effectue les mouvements des pointes pour fournir des mesures. Il fonctionne en mode semi-automatique, c’est-à-dire qu’il doit être piloté par un programme d’encadrement.

Dans le cas de Captinov, il convient de coordonner les déplacements du robot avec les ajustements des alimentations à haute tension. C’est naturellement Pyrame qui se charge de cette tâche.

Le LLR a utilisé cet équipement pour qualifier les *wafers* du projet *SiW-Ecal* (chapitre 5). Le test consiste à amener les sondes sur des points précis des *wafers*, les pixels et à qualifier leur courant de fuite. Pour cela, on injecte une tension sur le pixel et on mesure le courant ou la charge pour obtenir une courbe de courant en fonction du voltage “I de V” ou de charge en fonction du voltage “C de V”. Ces courbes sont très importantes pour qualifier les *wafers* et déterminer si leur qualité est suffisante pour les utiliser dans un calorimètre électromagnétique.

Comme dans le cas du banc laser, ce sont des scripts Python qui permettent d’obtenir les

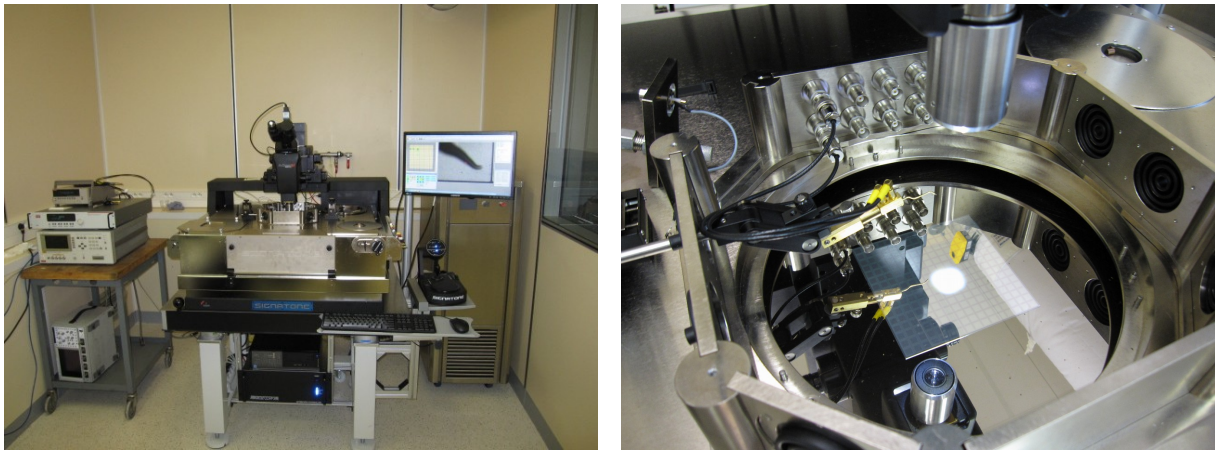


FIGURE 2.9 – Banc Captinov

mesures. En effet, la procédure n'est jamais tout à fait la même d'un cas à l'autre et il convient de l'ajuster à chaque cas spécifique.

2.13.4 Banc thermique pour CMS

Ce banc a été développé pour faire une étude thermique pour le futur calorimètre HGCal de CMS. Il a été développé par un stagiaire de DUT, Hugo Bonnemaison, co-encadré par le service mécanique et le pôle online. Le principe est d'instrumenter l'analogue d'une pièce mécanique avec de nombreuses sondes de température. Ensuite, ce système est soumis à des contraintes thermiques et la cinétique de température est mesurée. Cela permet de comparer aux simulations pour les calibrer.

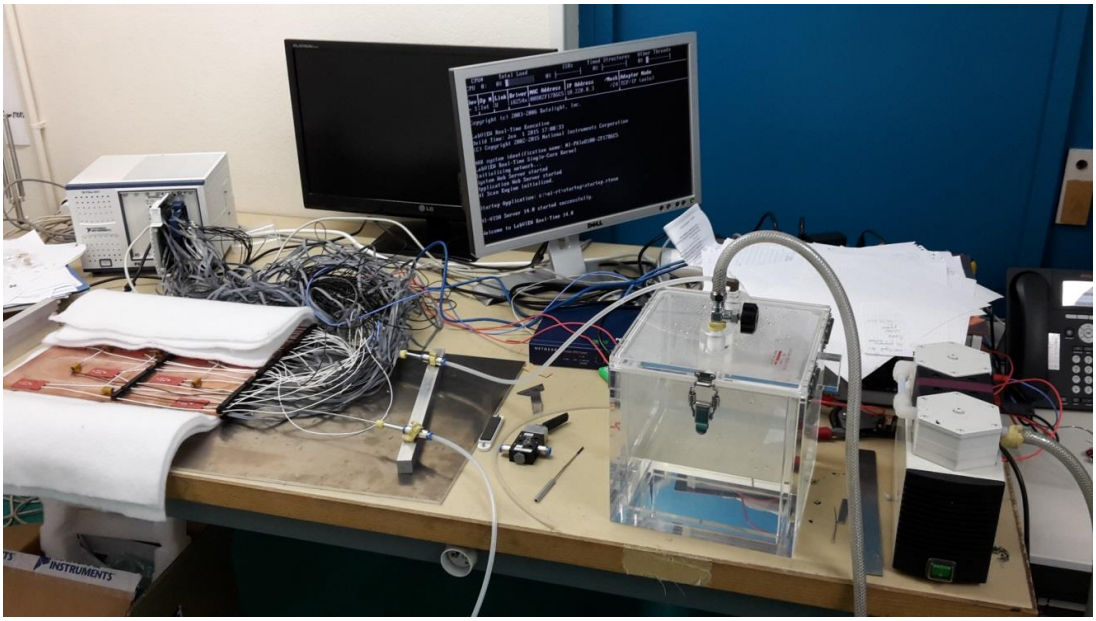


FIGURE 2.10 – Banc thermique CMS

La figure 2.10 présente une vue globale du système. A gauche, au premier plan, une pièce en

cuivre est instrumentée, par des résistances chauffantes et des tapis chauffants (pour simuler les composants électroniques qui dissipent de la chaleur), et sur l'autre face par des sondes PT100. On peut voir ces sondes sur la partie gauche de la figure 2.11.

Derrière la plaque, on voit un module NI Pxie-1071 qui permet de lire les sondes. C'est un système programmable en labview de National Instrument. Afin d'assurer la compatibilité avec Pyrame, nous avons implémenté dans ce contrôleur un programme qui le fait se comporter comme un module Pyrame. Nous avons ensuite étalonné les sondes dans l'eau glacée.

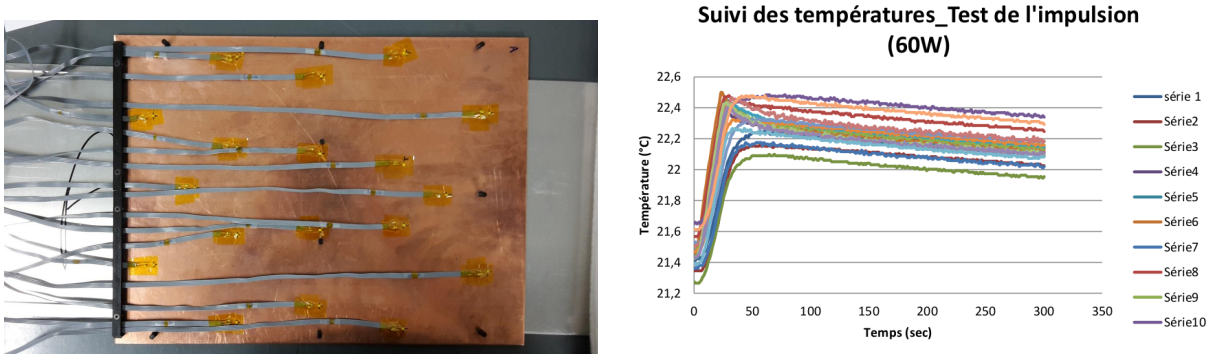


FIGURE 2.11 – Détail de la pièce instrumentée par les sondes et graphique de résultat d'une cinétique thermique

Un script Python permet de coordonner le pilotage des résistances et des tapis chauffants à travers une alimentation et la lecture des sondes. Le données sont collectées sur le controlleur des PT100 et assemblées dans des fichiers textes qui permettent leur exploitation à posteriori.

La partie droite de la figure 2.11 montre une cinétique correspondant à la température des points de tests après une impulsion de 60W répartis sur les différentes résistances chauffantes. On peut ainsi visualiser le régime de chauffe et le régime de retour à la normale.

2.14 Conclusion

Nous avons vu que Pyrame est un framework *online* qui permet de piloter une grande diversité de matériel mais aussi de gérer des acquisitions de données, y compris sous une forme complexe et scriptée. Ce framework générique est la base de nombreux projets au sein du laboratoire et permet d'envisager des temps de développements *online* très courts. Il permet également de mutualiser de nombreux codes entre les projets.

Du point de vue du traitement online des données, Pyrame implémente de nombreux mécanismes qui permettent d'automatiser de nombreuses fonctionnalités critiques pour le traitement temps-réel. Cette boite à outils permet d'implémenter rapidement et facilement de véritables chaînes de traitement online et de monitoring.

Nous verrons dans les chapitres suivants que son champs d'application ne se limite pas à des petits bancs-test mais qu'on peut combiner les modules jusqu'à atteindre des systèmes très complexes.

Chapitre 3

Harpo

Harpo (Hermetic ARgon Polarimeter) est une chambre à dérive (*TPC*) développée en collaboration par le LLR et le CEA de Saclay. C'est un télescope et un polarimètre à haute résolution pour les rayons gamma dans la gamme du MeV au Gev. Sa résolution angulaire est meilleure d'un ordre de grandeur par rapport à Fermi et EGRET.

Les motivations pour disposer d'un tel détecteur sont multiples. La polarisation linéaire de la lumière émise par les sources cosmiques est un outil d'analyse pertinent de la physique de ces sources. Une polarimétrie au dessus du MeV permettrait par exemple de discriminer les modèles hadroniques et leptoniques de l'émission gamma des blazars. En gamma, aucune mesure significative de polarisation n'est disponible à ce jour. D'autre part, entre la plage en énergie des rayons X durs et gammas de faible énergie où les télescopes Compton sont particulièrement sensibles, et la plage en énergie des rayons de haute énergie (de 1 à 300 GeV) où les télescopes sur satellite comprenant un tracker de paires électron/positron à convertisseurs en tungstène sont très efficaces, un fossé en sensibilité handicape la compréhension des spectres. En effet, la forte dégradation de la résolution angulaire des télescopes à création de paires à basse énergie empêche l'identification des sources dans les zones très peuplées du ciel, telles que le plan galactique.

Dans les détecteurs utilisant des plaques de convertisseur à Z élevé avec des couches fine de détection comme COS-B, EGRET ou Fermi-LAT, la résolution angulaire reste de l'ordre de quelques degrés dans cette plage d'énergie. Dans une *TPC* au contraire, la faible densité du gaz diminue la dispersion angulaire, permettant d'atteindre des précisions de l'ordre de 0.5°. Cette précision est suffisante pour mesurer l'angle de polarisation des photons incidents.

L'objectif ultime de la collaboration serait de faire un télescope polarimètre embarqué sur un satellite d'observation. Pour cela, un premier prototype "au sol" a été développé, permettant de valider les concepts et d'affiner les outils de simulation. Un second prototype permettant de faire des tests en vol ballon est la deuxième étape, aujourd'hui en attente de financement.

3.1 L'expérience Harpo

Harpo est donc le premier prototype d'un tel télescope. C'est une chambre à dérive de 30 cm^3 remplie d'un mélange gazeux d'argon et d'isobutane (95 :5) à 2.1 bar. Une cage fournit un champ

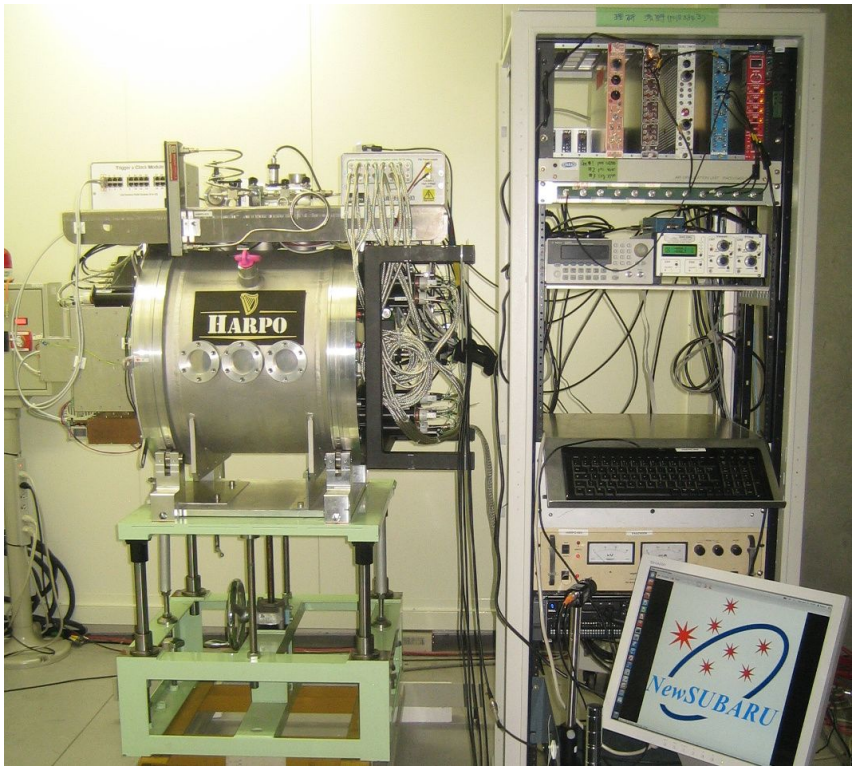


FIGURE 3.1 – Le prototype Harpo en *test faisceau* à NEWSUBARU (Credit Y. Geerebaert, LLR)

de dérive de 200V/cm. Lorsqu'un photon gamma interagit avec le gaz, l'interaction dominante est la création d'un paire électron / positron. Ceux-ci dérivent alors dans le champ électrique vers un plan de lecture avec une vitesse constante d'environ 3.3cm/μs. Le plan de lecture est équipé de deux multiplicateurs d'électrons (GEM) servant de pré-amplificateurs. Ensuite, une structure Micromegas [25] permet de multiplier les électrons dérivants et de collecter les charges. Cette collecte s'effectue sur deux ensembles de strips perpendiculaires d'un pas de 1mm. Les signaux sont digitalisés par des *ASIC* AFTER, développés spécifiquement au CEA ainsi que leur carte front-end associées (FEC). La figure 3.2 présente une vue éclatée de l'ensemble.

La figure 3.3 présente une trace typique de cette *TPC*. Les plans X et Y sont échantillonnés en temps, donnant de nombreux points décrivant la trajectoire des électrons et positrons. Les traces ne sont pas tout à fait droites car la diffusion multiple vient perturber la migration des charges.

La cinématique de l'interaction entre le photon incident et le gaz de la *TPC*, décrite dans [48] et surtout l'extrapolation de ses paramètres à partir des mesures projectives sont assez complexes. Dans Harpo, les trajectoires sont reconstruites sur les deux plans en utilisant un algorithme de recherche du plus proche voisin, basé sur un filtre de Kalman [27]. Cette méthode permet d'extraire les traces continues de l'environnement qui peut se révéler bruyant. Par contre, les traces sont tellement intriquées près du vertex qu'il est impossible de distinguer précisément sa position. Pour obtenir ce point, on utilise une transformée de Hough pour *fitter* les points avec des régressions linéaires, dont l'intersection donne le vertex [17]. Les traces 2D ainsi obtenues sont ensuite mixées afin d'obtenir le vecteur en 3 dimensions de l'impulsion du photon incident.

Toutefois, l'analyse est compliquée par la phénoménologie complexe de l'interaction photon-gaz.

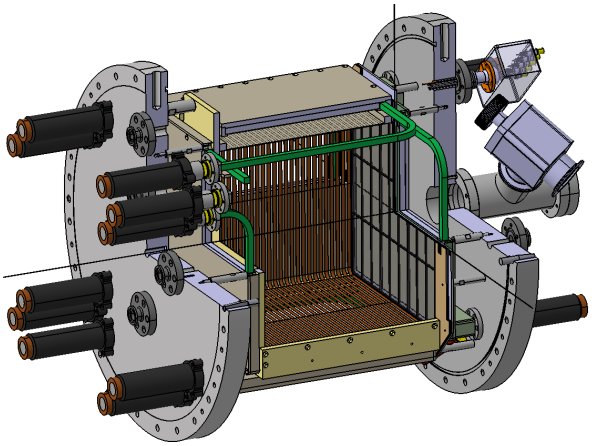


FIGURE 3.2 – Vue éclatée de la *TPC* (Credit M. Frotin, LLR)

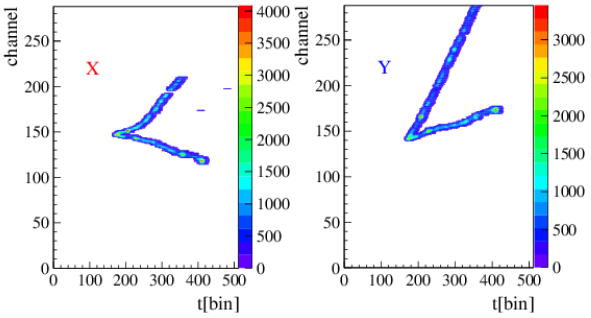


FIGURE 3.3 – Trace collectée (Credit P. Gros, LLR)

En effet, la création de paires, même si elle représente l'interaction dominante est mélangée avec de la diffusion Compton et de la production de triplets. La diffusion multiple va également perturber considérablement la reconstruction.

Comme on le voit sur la figure 3.4, la diffusion compton ne génère qu'une seule trace. Dans le cas d'un paire, on obtient 2 traces. Les triplets génèrent trois traces. On voit également sur la composante Y de la paire un exemple de diffusion qui rend plus difficile la régression linéaire.

La *TPC* a été mise à l'épreuve lors d'un *test faisceau* à NEWSUBARU en novembre 2014. Un faisceau de photons gammas y est produit par diffusion Compton (Laser Compton Scattering) d'un laser optique sur un faisceau d'électrons de haute énergie (0.6-1.5GeV). Un scan en énergie a été effectué démontrant l'excellente performance du polarimètre sur tout le spectre. Ces travaux ont donné lieu à un article [30] dans le journal "Astroparticle Physics".

3.2 Analyse longue durée du gaz

Nous l'avons vu, Harpo est un démonstrateur, sensé ouvrir la voie à des tests en ballons puis éventuellement à une exploitation sur un satellite. C'est pourquoi il est nécessaire de pouvoir opérer le détecteur pendant de longues périodes de temps. Or jusqu'au *test faisceau*, le détecteur ne pouvait être en opération que sous la surveillance d'un *shifter*.

En effet, le composant micromegas, fortement chargé, a tendance à faire des étincelles entre sa grille et son plan de lecture, risquant fort de l'endommager. C'est pourquoi il a été nécessaire de sécuriser son fonctionnement au moyen d'une analyse *online* des courants effectuée en temps-réel dans Pyrame. Ces travaux ont été menés en collaboration avec Philippe Gros alors post-doctorant au laboratoire et aujourd'hui chercheur à la Queen's University de Kingston (Canada) et sous la direction de Denis Bernard.

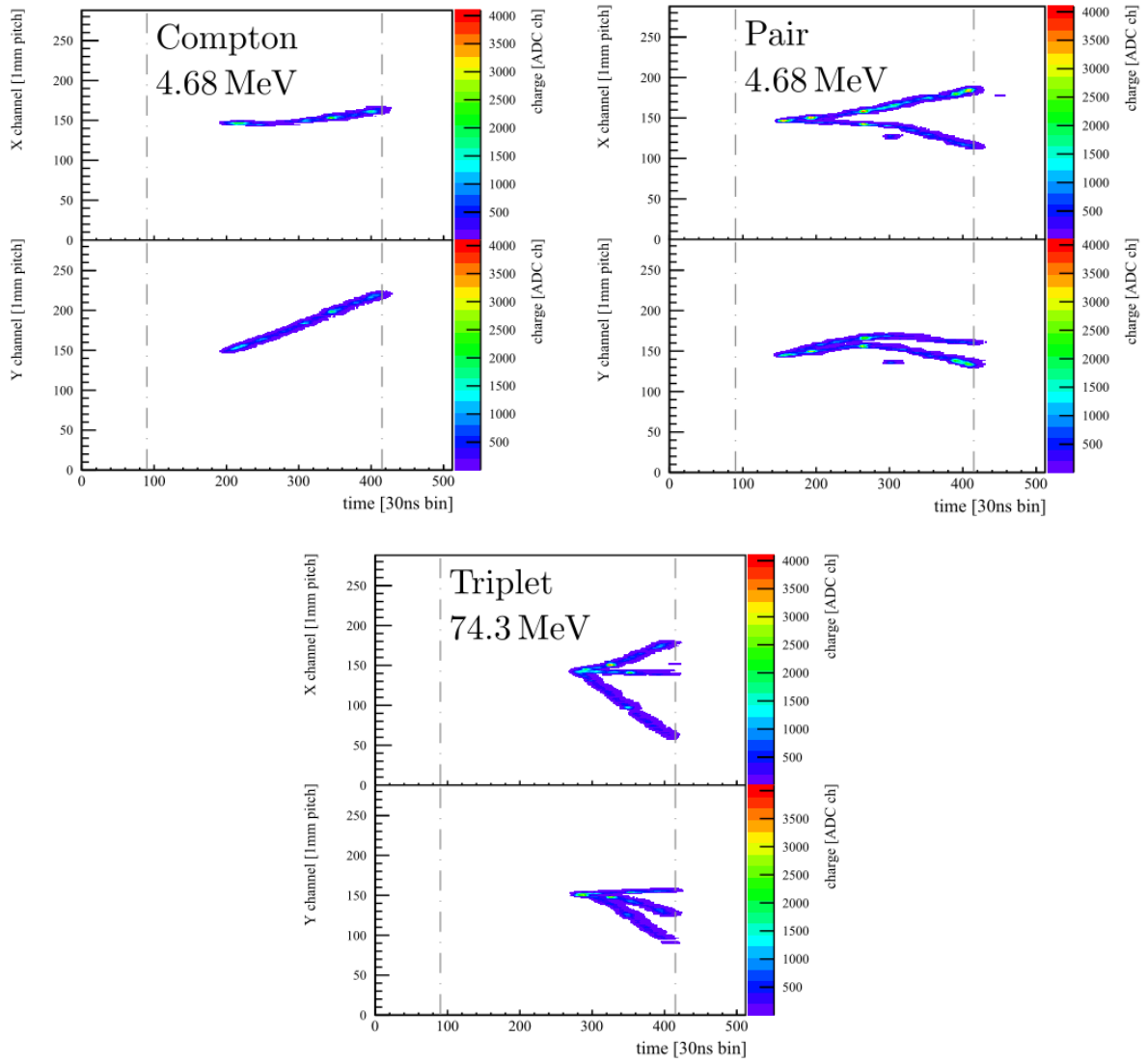


FIGURE 3.4 – Différentes traces d'interactions dans Harpo (Credit P. Gros, LLR)

3.2.1 Détecteur d'étincelles

Les étincelles qui apparaissent dans le micromegas sont mesurables à travers le courant de l'alimentation électrique qui le polarise. Ces étincelles sont normales et ne posent pas de problèmes si elles sont isolées. Mais parfois, un régime d'avalanche permanent apparait qui pourrait endommager définitivement le délicat matériau du micromegas. Si l'on souhaite utiliser le détecteur sans supervision, il faut être capable de faire une analyse fine de ces étincelles pour en discriminer les différents régimes.

L'analyse s'effectue en surveillant le courant entre la grille et les strips, qui est normalement extrêmement bas mais qui présente des pics quand des étincelles apparaissent. Ces pics provoquent un léger courant, suffisamment élevé pour que l'alimentation le détecte. Il est donc nécessaire d'échantillonner la tension relevée sur l'alimentation pour la monitorer.

Cette opération est assurée par le logger de Pyrame décrit dans la section 2.8. Il est configuré

pour interroger l'alimentation toutes les deux secondes au moyen d'une interface série. Il récupère ainsi les intensités et les tensions correspondant à l'alimentation de la grille des strips, sur les deux électrodes du micromegas. Le logger publie ensuite ces données selon deux modalités. D'une part, sous la forme de fichiers textes dans lesquels les données sont formatées en CSV, pouvant facilement être intégrés dans le moniteur ROOT qui publie ses graphiques sur un site web ou visualisé directement dans gnuplot. D'autre part, le logger publie ses données au moyen d'un *dispatcher* de données afin de les mettre à disposition du programme d'analyse.

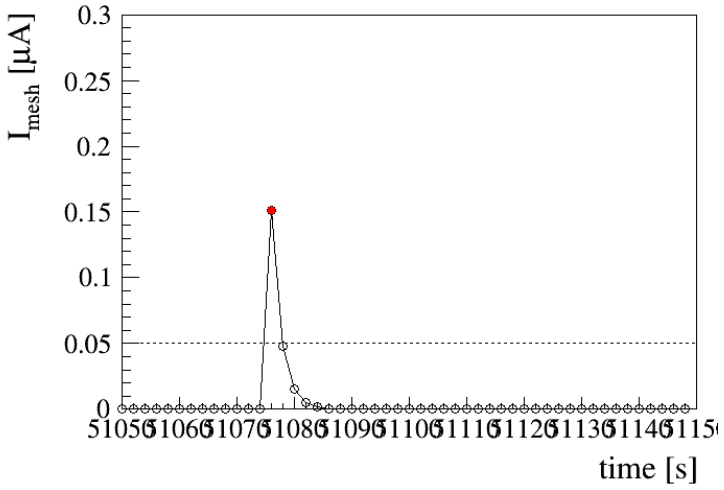


FIGURE 3.5 – Étincelle dans Harpo (Credit P. Gros, LLR)

Un module, développé spécifiquement en ROOT et nommé le "détecteur d'étincelle", se connecte sur ce *dispatcher* et analyse les données en temps réel selon plusieurs modalités.

La première modalité consiste à détecter des étincelles courtes mais rapprochées. Son fonctionnement est basé sur un niveau de référence pour la détection (threshold) qui est fixé à 10nA. L'état bas correspond à un courant plus petit que la référence et l'état haut au contraire. Dès que le courant dépasse la référence, nous considérons qu'une étincelle est en cours. La figure 3.5 présente une telle étincelle isolée. Dès lors, le détecteur d'étincelle va compter ces transitions au sein d'une fenêtre glissante de 100s. Si au sein de cette fenêtre, le nombre d'étincelles dépasse un seuil règlable, le détecteur est considéré comme en régime d'étincelles rapprochées et une alerte est émise vers le gestionnaire de sureté (décrit ci-après). Ce seuil a été fixé à 5 étincelles.

La deuxième modalité consiste à détecter des régimes où les étincelles sont tellement rapprochées qu'elles sont mesurée comme un état haut permanent. Pour le mesurer, le détecteur d'étincelle va simplement compter les états haut consécutifs. Si un seuil, actuellement fixé à 4, est dépassé, le détecteur est considéré en régime d'étincelle permanent.

La troisième modalité consiste à surveiller les écarts de voltage qui dépassent un certain seuil (voltage drop) et qui traduisent un dysfonctionnement, non liés aux étincelles mais qui dégradent le fonctionnement du détecteur.

La quatrième et dernière modalité est très technique. Il s'agit de s'assurer que l'alimentation donne des résultats cohérents. En effet, il arrive qu'elle se bloque dans un état d'erreur dans lequel elle ne nous transmet par les valeurs correctes de tension. Nous détectons ce mode en vérifiant si la valeur transmise a fluctué au cours de 1000 derniers échantillonnages. Si aucune fluctuation n'est constatée, l'alimentation est considérée en panne. Il faut alors la redémarrer manuellement.

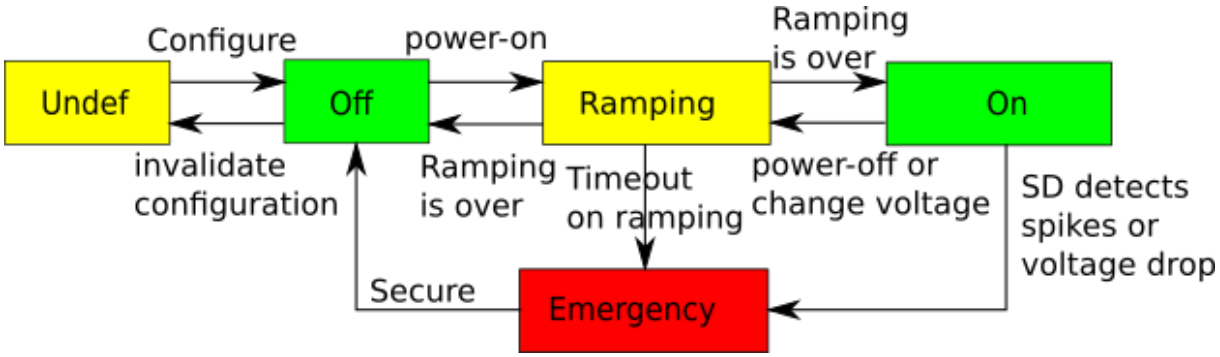


FIGURE 3.6 – Machine d'état du gestionnaire de sureté

Lorsque le détecteur d'étincelle détecte une situation anormale, il envoie un signal spécifique vers un autre module, le gestionnaire de sureté. Celui-ci est basé sur une machine d'état, décrite par la figure 3.6. Son rôle est de gérer la sureté de fonctionnement de l'ensemble en pilotant les alimentations.

Lorsque le système démarre, la machine d'état est dans l'état "undef" (où aucune configuration n'est chargée). Lorsque la transition "configure" est activée, le fichier de configuration xml est lu et tous les modules du système sont configurés. Si cette configuration se passe bien, on passe dans l'état "off" ou le système est prêt à allumer les alimentations. La transition "power-on" déclenche une rampe qui permet d'amener les alimentations à leur valeur de configuration mais avec une vitesse contrôlée afin de protéger les composants qui craignent les transitoires. L'état "ramping" est actif pendant toute la montée en tension. Si les tensions de référence sont atteintes, le système passe dans l'état "on". Si au contraire la tension n'est pas atteinte, une temporisation finit par détecter que le ramping est trop long et passe dans l'état "emergency" qui correspond à une erreur.

Une fois les alimentations à leur tension nominale, le détecteur d'étincelle se met en marche et commence à analyser les courants. En cas de mesure d'un dysfonctionnement, une alerte est envoyée et le système passe dans l'état "emergency". Dans ce cas, les alimentations sont ramenées à zéro afin de préserver le matériel. Un email est envoyé au *shifter* car une action manuelle de sa part est requise. Pour ramener le système à son fonctionnement normal, il devra appliquer la transition "secure" à la machine d'état, ce qui la fera revenir à l'état "off", prête pour un nouveau cycle d'alimentation.

Naturellement, nous aurions pu automatiser complètement le processus de ré-enclenchement mais par prudence, nous avons préféré imposer une action manuelle qui permet d'intégrer des vérification inaccessibles à la machine, comme l'historique ou des circonstances externes.

3.2.2 Principaux résultats

Grâce à ce système, le détecteur Harpo peut maintenant fonctionner 24h sur 24 en toute sécurité. Cela nous a permis de mener des études de stabilité à long terme. Ainsi, nous avons pu mettre en évidence un effet de pollution du gaz, dégradant le gain du détecteur avec le temps. Une étude de la composition chimique du gaz a montré la présence d'oxygène dans le mélange, provenant de l'environnement. L'équipe mécanique a mis au point un mécanisme de circulation du gaz dans lequel est intégré une cartouche Oxysorb, qui absorbe l'oxygène du mélange. La mise en oeuvre de la circulation a permis de voir une amélioration nette du gain. Par contre, le gaz

purifié favorise clairement l'apparition des étincelles dont la fréquence à augmenté sensiblement. En effet, grâce aux mesures ininterrompues, nous avons pu établir une corrélation linéaire entre le gain et l'apparition des étincelles. La figure 3.7 montre clairement cette corrélation.

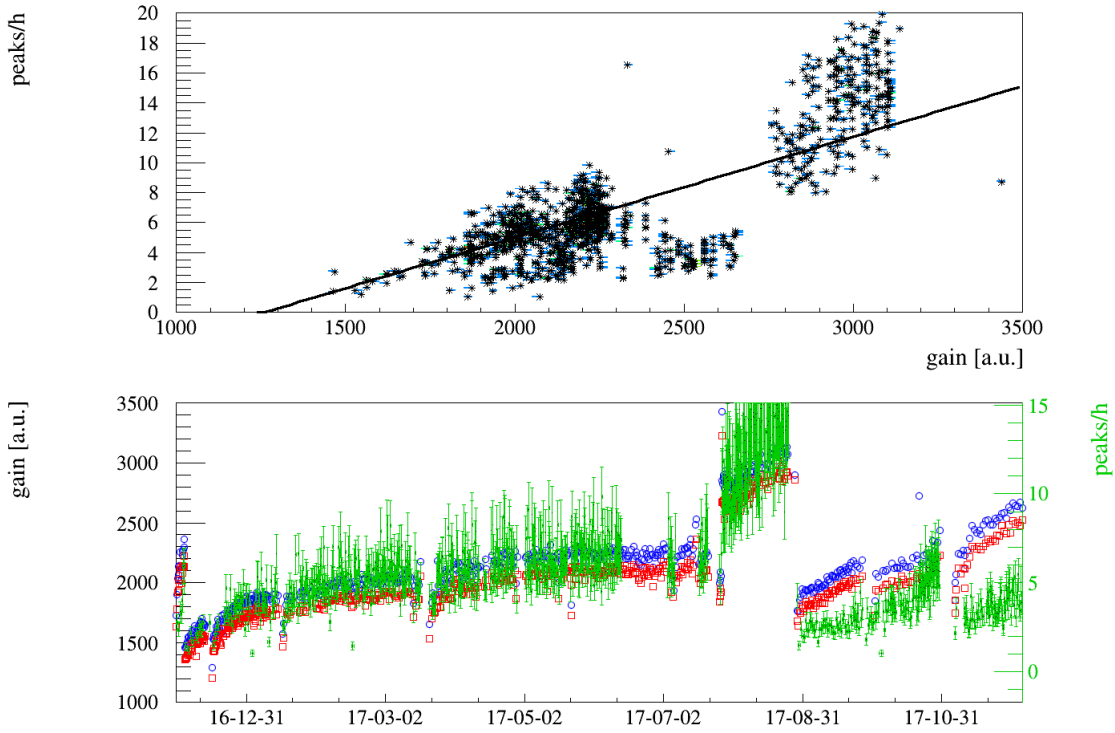


FIGURE 3.7 – Taux d'apparition d'étincelle en fonction du gain (Credit P. Gros, LLR)

La figure 3.8 montre l'évolution des paramètres du détecteur sur un an. On y voit l'évolution du taux de capture de électrons qui augmente linéairement avec le temps à cause de la pollution de l'oxygène. Fin juillet (au niveau de la barre noire verticale, nous avons mis en oeuvre le circulateur. On voit que le taux de capture chute exponentiellement pour revenir à son état antérieur. La cartouche Oxysorb s'est ensuite lentement saturée sur une centaine de jours (sa couleur en atteste) jusqu'à devenir totalement inopérante. Depuis, on voit le taux de capture remonter linéairement. On peut également observer sur la dernière ligne, que l'intensité de l'alimentation I_{mesh} a été baissé plusieurs fois. En effet, la purification a fait monter le taux d'étincelles jusqu'à un niveau inacceptable, nous obligeant à baisser la tension.

Une analyse de gaz, présentée sur la table 3.1 montre la composition du gaz initialement placé dans l'enceinte au début de la prise de données et une autre après la re-circulation. On voit que la cartouche a bien éliminé l'essentiel de l'oxygène. L'azote apparu est le résidu de l'air qui s'est infiltré dans le détecteur et le CO2 provient d'un dégazage, probablement dû au plastique des scintillateurs. On voit aussi que du quencher (l'isobutane) a disparu. Cela est du en partie aux opérations de flushing liées à la mise en oeuvre du circulateur et qui ont été effectuée à l'argon pur, en partie consommée par la cartouche Oxysorb.

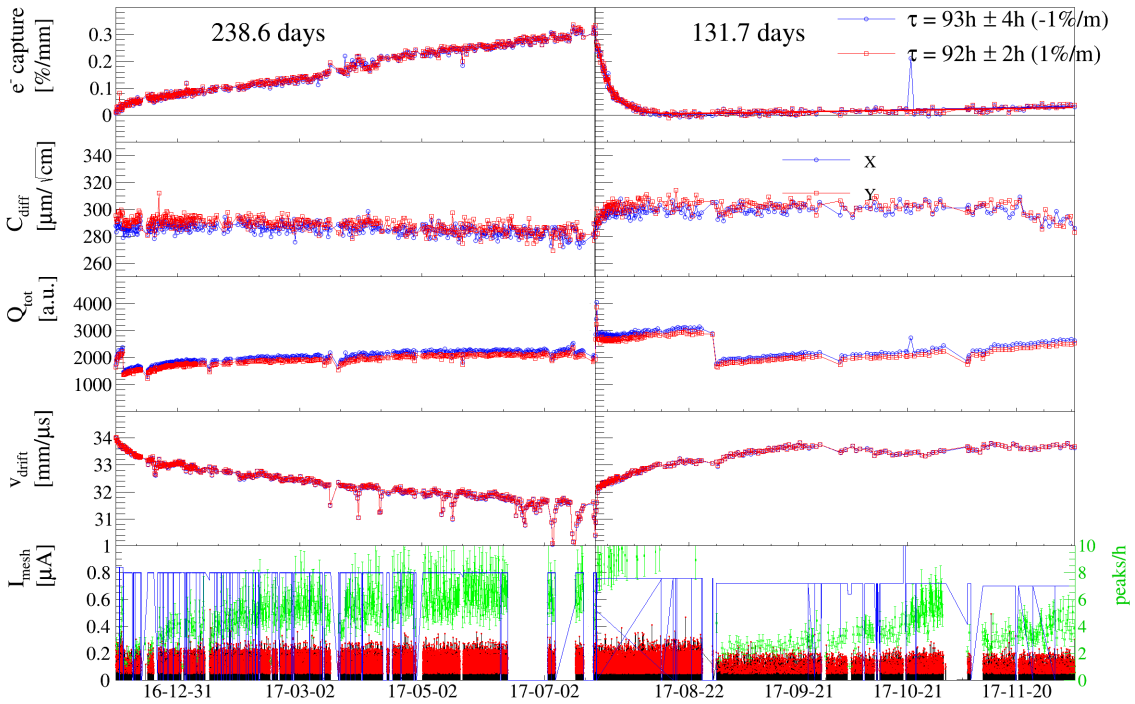


FIGURE 3.8 – Évolution des paramètres du détecteur en fonction du temps (Credit P. Gros, LLR)

	H2	H2O	CO	N2	O2	CO2	Isobutane	Ar
Gaz de référence	0,000	0,000	0,000	0,000	0,003	0,000	5,161	94,836
Prélèvement après re-circulation	0,002	0,003	0,008	0,119	0,003	0,009	4,707	95,150

TABLE 3.1 – Analyse de gaz au début de la phase scellée et après la phase d'absorption

3.2.3 Conclusion

L'ensemble des modules Pyrame nous ont permis d'implémenter rapidement une solution pour un problème assez spécifique mais qui se ramène basiquement à une acquisition et une analyse temps-réel de données. La fraction de développement spécifique est très faible et véritablement concentrée sur la problématique (la détection d'étincelle) et non pas sur les problématiques arides de distribution de données. Cela illustre bien l'intéret de posséder un framework générique tel que Pyrame.

Ces travaux intéressent la communauté des détecteurs gazeux et une publication est en cours de rédaction. Il est également envisageable de mettre à disposition les éléments spécifiques dans le framework pour être utilisables sur d'autres expériences.

Chapitre 4

Compact Muon Solenoid

CMS (Compact Muon Solenoid) est l'un des quatre détecteurs installés sur l'anneau LHC [26], le grand collisionneur de hadrons situé au CERN, l'accélérateur de particules le plus puissant au monde. C'est un détecteur polyvalent, permettant d'explorer aussi bien la physique du boson de Higgs que celle de la matière noire ou des particules super-symétriques. Il est décrit dans [11]. C'est le détecteur frère d'ATLAS décrit dans [7], qui bien que basé sur des concepts différents, permet d'effectuer des observations similaires.

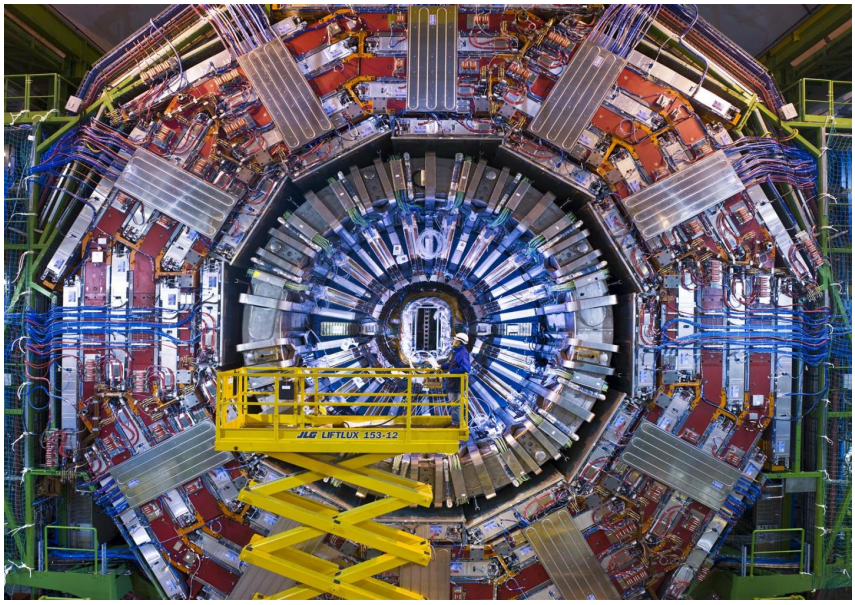


FIGURE 4.1 – CMS en cours de construction (Credit Ph. Maximilien, CERN)

CMS, tel que décrit dans son technical proposal [9], est construit autour d'un gigantesque aimant solénoïde, construit sous la forme d'une bobine cylindrique supraconductrice permettant d'atteindre un champ magnétique de 3.8 Teslas uniforme le long de l'axe du faisceau, confiné par une culasse d'acier. La taille de cette structure implique des dimensions imposantes au détecteur CMS : 28.5 mètres de long, 15 mètres de diamètre. Ces dimensions n'ont pas permis de construire tout le détecteur en surface. Il été descendu dans le puits par tranches entières, et l'assemblage a été finalisé dans la caverne.

Sa construction et son opération ont nécessité la constitution de l'une des plus grande collaboration scientifique qui ait jamais existé, rassemblant 4300 chercheurs, ingénieurs, techniciens et étudiants provenant de 179 universités ou instituts de 41 pays.

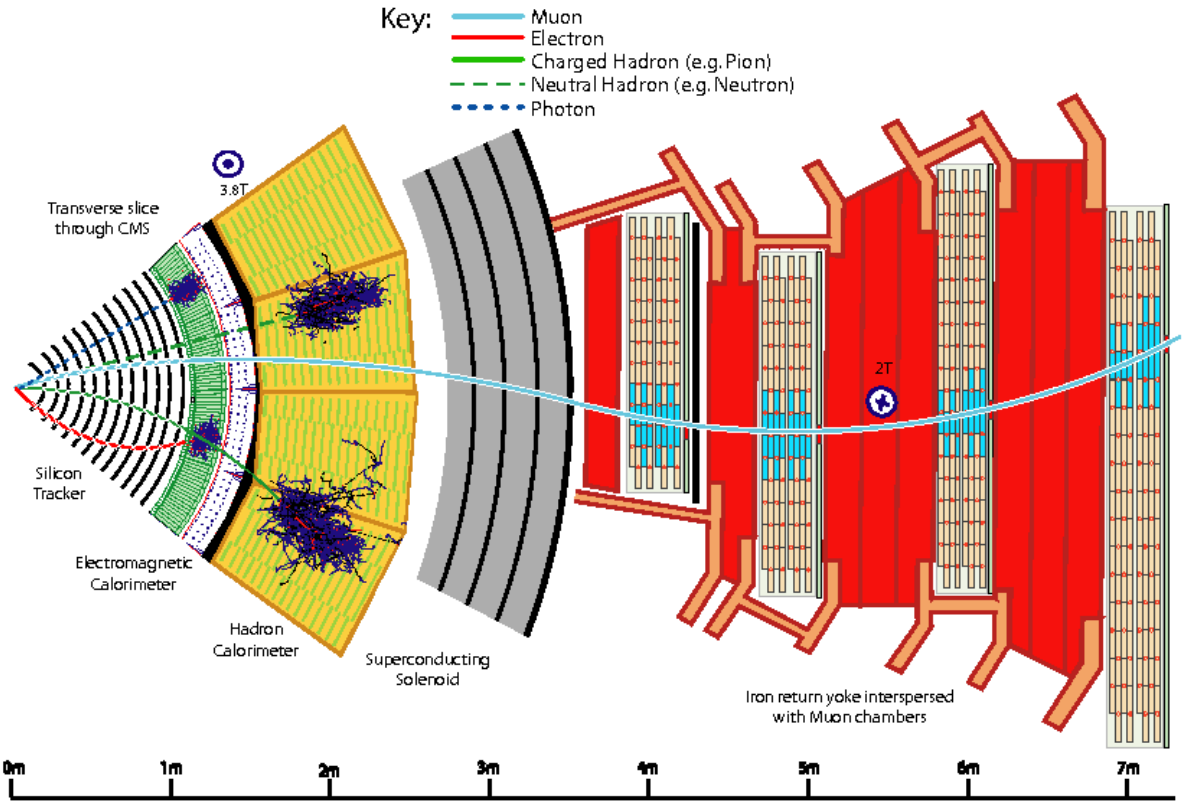


FIGURE 4.2 – Schéma de principe de CMS. Il est composé de couches de détection successives : tracker silicium, calorimètres électromagnétique puis hadronique, le tout plongé dans le champs magnétique. Des chambres à muons sont incluses dans la culasse de l'aimant. (Credit D. Barney, CERN)

Comme le montre la figure 4.2, il est composé de plusieurs couches de détection, spécialisées dans un type particulier d'interaction. Tout d'abord, un *trajectographe* en silicium repère les points d'impact ainsi que les trajectoires des particules chargées. Ensuite, un calorimètre électromagnétique (ECAL) mesure l'énergie des électrons et des photons. Puis un calorimètre hadronique (HCAL) mesure l'énergie des hadrons. Le tout est placé dans le champs magnétique de 3.8 Teslas induit par le solénoïde, qui courbe la trajectoire des particules afin de déterminer leur charge et leur moment. Enfin des chambres à muons ceinturent l'ensemble. Cette succession de couches de détections prend une forme de tonneau pour la partie centrale (barrel) complété par deux bouchons pour les extrémités (endcaps).

4.1 Calorimètre électromagnétique de l'expérience CMS

CMS est doté d'un calorimètre électromagnétique homogène décrit dans son technical design report [10]. Il est composé de 75848 cristaux scintillants, dont la lumière est convertie en signal électrique par des photo-détecteurs. Ces cristaux d'une dimension de $2.2 \times 2.2 cm^2$ sur 23 cm de

long sont composés de tungstate de plomb ($PbWO_4$) et représentent $28X_0$. Ils sont regroupés en deux demi cylindres et deux bouchons, formant un assemblage hermétique. Il est complété par un détecteur de pied de gerbe (pre-shower) au niveau de l'axe de collision, où la présence hadronique est la plus forte.

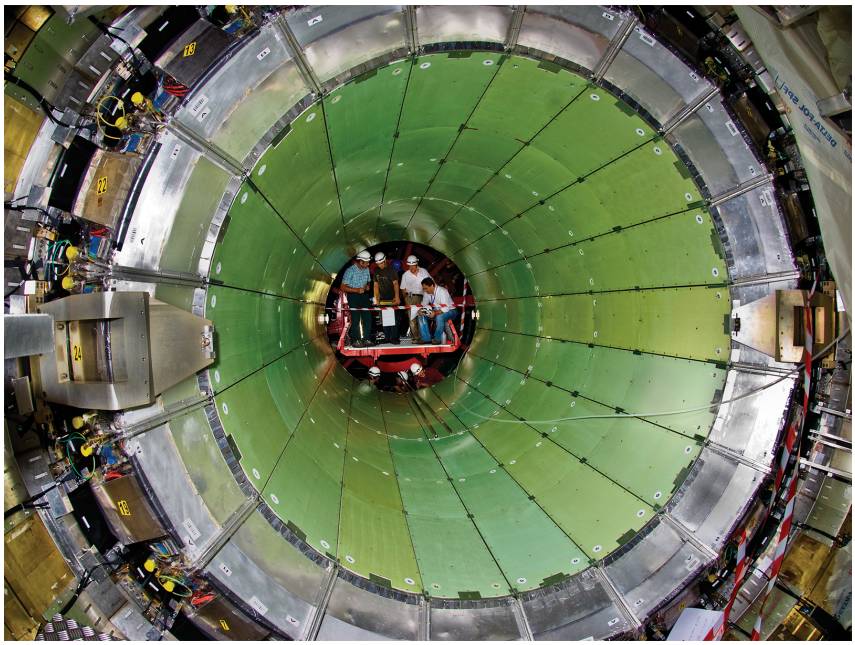


FIGURE 4.3 – Le calorimètre électromagnétique de CMS (Credit CERN)

Les photons générés par scintillation des cristaux vont être convertis par de photodiodes à avalanche (APD) dans le tonneau et par des photo-triodes à vides dans les bouchons, plus résistantes aux radiations et au champ magnétique. Les charges produites sont intégrées et amplifiées sur une gamme dynamique du GeV au TeV. Le signal est ensuite échantillonné par des convertisseurs analogique-digital sur 12 bits plus 2 bits de codage du gain. Des cartes appelées very front end (VFE) regroupent les informations provenant de cinq cristaux et les transmettent à une carte front end (FE). Les données y sont stockées avant d'être envoyées vers l'électronique hors détecteur à nouveau sous forme groupée (tour de *trigger*).

4.2 Acquisition de données et *Trigger* de niveau 1

Le LHC délivrant des collisions à une fréquence de 40 méga-hertzs, un stockage exhaustif des traces de collisions n'est pas envisageable. En effet, les débits autorisés par l'électronique puis l'informatique de lecture sont trop faibles pour envisager de transmettre l'intégralité de l'information générée par le détecteur qui représentent environ un péta-octet par seconde. Il est donc nécessaire de choisir finement les collisions qui présentent le plus d'intérêt en fonction de la physique que l'on souhaite observer. Pour cela, on va utiliser deux niveaux de *trigger*, tel que décrit dans [21]. Un premier niveau entièrement électronique qui va ramener la fréquence à 100 kilo-hertz puis un deuxième niveau informatique qui va à nouveau faire une coupure jusqu'à la fréquence de 1000 hertz, ce qui représente un débit binaire final de cent méga-octets par seconde.

Pour procéder au *trigger* de niveau un, on va appliquer des algorithmes rapides, implémentés directement dans l'électronique consitutée de FPGA, et qui permettent de sélectionner les

meilleurs enregistrements. Le temps pour effectuer ce choix est limité par la taille des buffers du front-end. Pour CMS, il est de l'ordre de 8.8 micro-secondes.

Des cartes spécifiques, les cartes de concentration de *trigger* (TCC) ont été développées au LLR, par Yannick Geerebaert et Thierry Romanteau, afin de répondre à cette problématique. Ils en existe deux versions dont l'électronique comme le firmware sont différents, une pour le tonneau et une pour les bouchons. La figure 4.4 montre ces cartes. Elles sont décrites dans [37].

Ces TCC génèrent des primitives de *trigger*, c'est-à-dire qu'elles effectuent, localement, une série de calculs, qui donneront une idée de l'intérêt de la collision, une fois agrégés . Naturellement, elles reçoivent les données du front-end à la vitesse des collisions (40MHz). Les primitives sont envoyées à un *trigger* régional puis un *trigger* global pour être enfin mélangées avec les primitives de *trigger* des chambres à muons pour produire la décision finale. Lorsque cette décision est positive, le *trigger* global envoie un signal (dit L1) au front-end qui transmet alors les données de la collision choisie vers une carte appelée Data concentrator card (DCC) qui s'occupe d'acheminer la données vers le *trigger* de niveau 2.

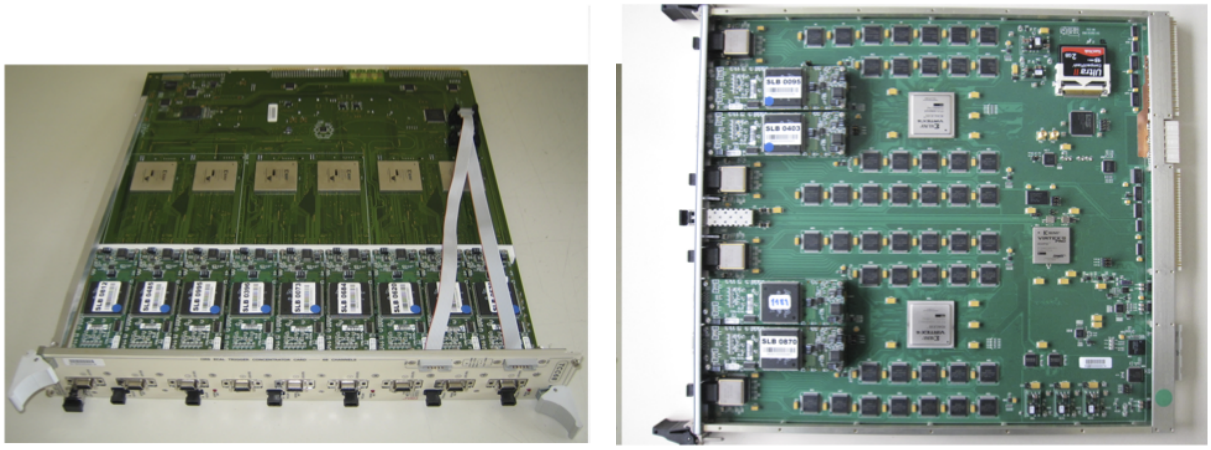


FIGURE 4.4 – Cartes de *trigger* TCC

Ces cartes TCC, 36 pour le tonneau et 72 pour les bouchons sont situés dans l'électronique hors-détecteur, c'est-à-dire dans un local sous-terrain situé à la même profondeur, mais séparé de la caverne du détecteur par des gros murs de bétons dans lesquels passent les fibres optiques. Cet environnement ne présente pas de problème radiatifs et permet un accès pendant les opérations, contrairement à la caverne.

Les TCC sont pilotées par un logiciel spécifique qui lui fournit les éléments de configuration et les commandes relayées du niveau central. Pour cela, des serveurs de calculs sont installés dans la salle hors-detecteur. Ils sont connectés aux cartes TCC au travers d'un bus VME présent sur les châssis accueillant le matériel. Ce bus permet de transmettre des ordres de lecture ou d'écriture vers les registres des cartes et ainsi de les piloter. Ce logiciel doit également recevoir des ordres du logiciel de gestion global du calorimètre. Pour cela, il utilise le *framework* XDAQ [24] qui lui envoie des ordres, collecte les statuts et des données de monitoring en temps-réel.

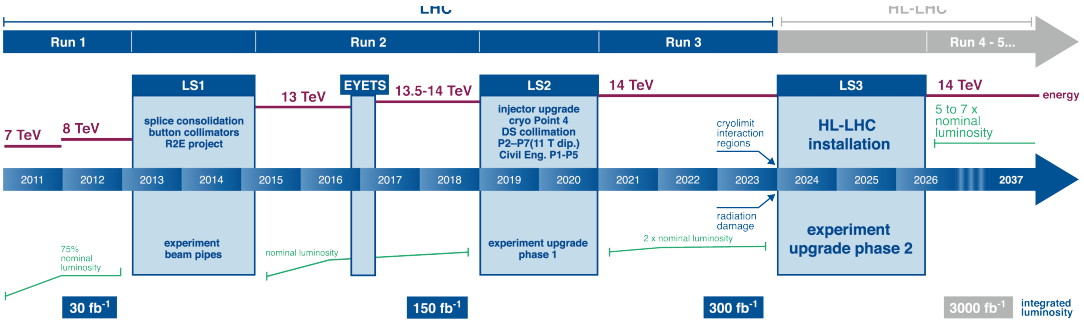


FIGURE 4.5 – Planning du LHC : Les phases de prises de données (runs) alternent avec des arret de mise à niveau technique du système (long stop). (Credit CERN)

4.3 Refonte complète du logiciel

Comme on le voit sur la figure 4.5 représentant le planning du LHC, celui-ci a été mis en service pour le premier run en 2011. Les données de ce run ont permis de valider la théorie du boson de Higgs [12] et de découvrir son énergie vers 125 GeV.

En 2013, pendant l'arrêt long numéro 1 (LS1), le détecteur a reçu de nombreuses modifications lui permettant de supporter la montée en énergie et en luminosité. En particulier, il a été décidé de procéder à différentes évolutions sur le système des TCC et de réparer les zones endommagées par les radiations.

Le logiciel, résultat de nombreux développements incrémentaux avait accumulé une dette technique certaine. L'architecture logicielle présentait une inutile complexité dans son modèle de classes C++, ne correspondant pas à son mode de fonctionnement, relativement séquentiel. Par ailleurs, de nombreuses redondances dans le code ainsi qu'une documentation dispersée, le rendaient ardu à maintenir.

Les conséquences de cette architecture se faisait sentir dans le fonctionnement courant du détecteur :

- L'interface graphique était très instable et crashait régulièrement.
- Certains comportements des drivers ne correspondait pas totalement à leur spécification.
- Le temps de configuration était lent par rapport aux autres composants du détecteur.
- De nombreux éléments de configuration étaient en dur dans le code au lieu d'être stockés dans la base de données des configuration

Pour ces raisons, une modification en profondeur du logiciel était nécessaire.

Par ailleurs, ce nouveau logiciel devrait s'adapter à une nouvelle configuration, tant matérielle que logicielle. Les cartes filles de communication des TCC devaient être remplacées par des modules optiques, imposées par les nouvelles conditions de luminosité et nécessitant une configuration spécifique. Les machines hébergeant le logiciel devaient également évoluer en terme de système d'exploitation (passage à Scientific Linux 6) et de logiciel d'interconnexion (XDAQ 12). Il a alors été décidé de procéder à une restructuration et à un ré-usinage (refactoring) complet du code. J'ai effectué ce travail en collaboration étroite avec Floris Thiant, aujourd'hui responsable de tous les développements CMS au LLR, Alexandre Zabi, le responsable du trigger et Pedro Parracho, le responsable de la suite logicielle du calorimètre.

La figure 4.6 présente les modalités de la restructuration de code. Les drivers représentent la

partie la plus proche du matériel. Il permet d'effectuer les opérations de base sur les cartes à travers le bus VME (lecture et écriture de registres). Les superviseurs sont des programmes de séquences qui reçoivent des ordres du système central (configuration, acquisition, reset...). Enfin, l'interface graphique permet de visualiser et de configurer les cartes TCC directement. C'est l'outil privilégié de l'expert pour comprendre le comportement de son système.

Comme on le voit sur la figure, les codes des interfaces graphiques et des superviseurs ont été fusionnés car les différences entre les deux cartes, si elles sont importantes au niveau électronique, ne le sont que peu d'un point de vue du logiciel. Ce qui a pu être mutualisé entre les deux drivers a été regroupé dans un driver TCC générique et seules les couches les plus basses ont été maintenues différenciées. Enfin, le driver VME a été enrichi d'un mécanisme générique d'accès aux registres par leur nom et non par leur adresse. Ainsi, de nombreuses redondances de code ont été éliminées.

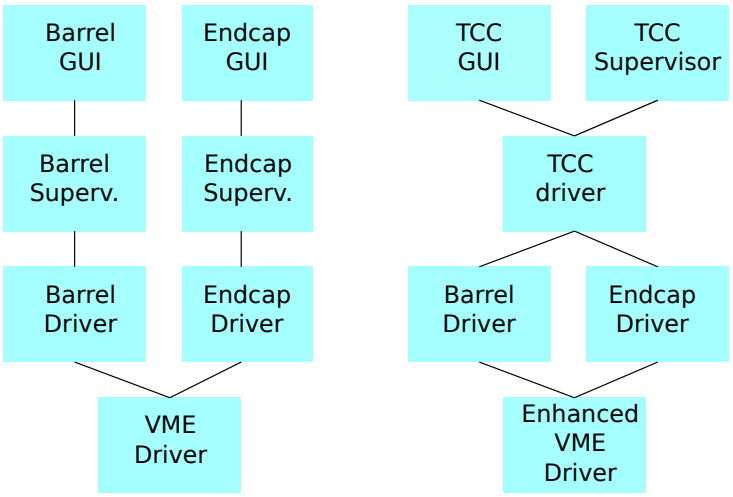


FIGURE 4.6 – Comparaison des anciennes et nouvelles architectures du logiciel des TCC

Le temps de configuration a été grandement amélioré, d'une part en incluant dans le firmware des cartes TCC les valeurs par défaut les plus couramment utilisées mais également en rendant parallèle le système de configuration. Comme chaque controleur est connecté à plusieurs châssis VME, cela permet d'optimiser le débit de données montant vers les cartes et de diminuer significativement le temps de configuration. Une tentative a également été faite pour tenter d'utiliser un mécanisme nommé multi-read permettant de diminuer la latence protocolaire du VME en aggrégeant des requètes multiples. Malheureusement, son utilisation engendrait un conflit d'accès avec la librairie HAL.

L'ensemble du système a bénéficié de cette ré-écriture du code en gagnant significativement en stabilité. Les plantages de superviseur et de GUI, courants auparavant, ont complètement disparus, faisant de ce trigger l'un des sous-système les plus stables de CMS.

4.4 Masquage automatique

La principale faiblesse d'un *trigger* de niveau un comme celui de CMS est l'apparition de taux de *trigger* trop élevé, générant des prises de données non conformes aux attentes et ainsi des temps morts dans la prise de données. Cet excès de taux de *trigger* viennent en général d'un ou de plusieurs canaux bruyants, c'est-à-dire qui annoncent des valeurs d'énergie incohérents.

Durant le run I, les canaux bruyants ont été masqués au fur et à mesure de leur découverte et sans retour en arrière, permettant au détecteur de fonctionner correctement. Toutefois, à la fin du run I, une analyse de performance [15] a clairement montré que ce masquage était la principale source d'inefficacité du détecteur. La figure 4.7 montre l'efficacité de déclenchement du *trigger* en 2010. Les masques (les zones noires) représentait seulement 0.44% du tonneau (269 cristaux) et 2.04% des bouchons (299 cristaux). La figure montre que ces pourcentages faibles de masquage peuvent induire d'importante pertes d'efficacité. Cette dégradation est surtout marquée lorsque les tours masquées sont contiguës. En effet, si une tour masquée est frappée par un électron, l'une au moins des tours voisine collectera un dépôt d'énergie périphérique. Cela permettra d'identifier la présence une particule, même si l'évaluation de son énergie n'est pas correcte. Dans le cas où plusieurs tours contiguës sont masquées, des particules peuvent être purement et simplement ignorée, perturbant profondément l'algorithme de déclenchement et conduisant à la dégradation d'efficacité que nous observons.

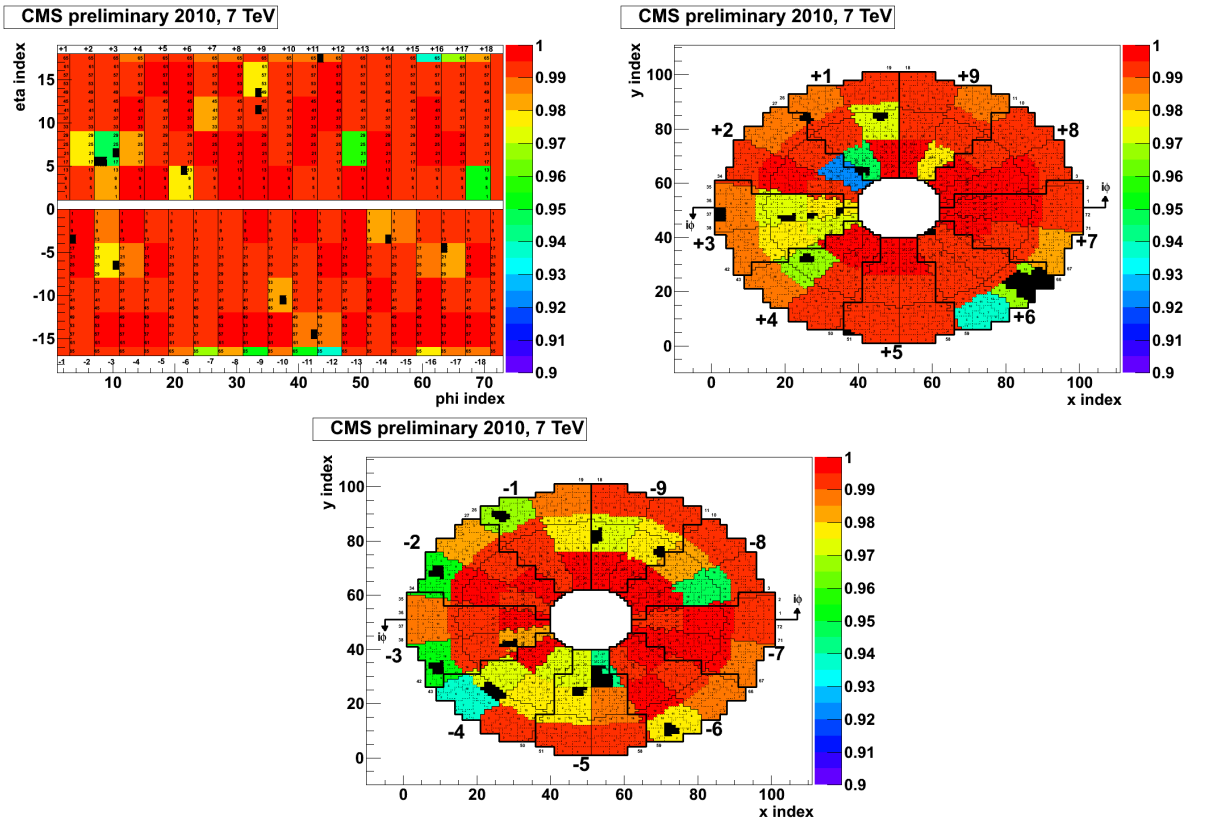


FIGURE 4.7 – Efficacité du *trigger* (couleur) et masquage (noir) pour le tonneau en haut à gauche, le bouchon plus en haut à droite et le bouchon moins en bas (Credit A. Zabi, LLR)

Un canal peut être bruyant pour différentes raisons. Soit l'électronique front-end peut être défaillante au niveau de son alimentation basse tension, de ses modules optiques (GOH/GOL), de ses modules de distribution d'horloge (Token ring) ou de ses modules de générations d'horloge (QPLL). Soit les liens optiques eux-même peuvent être endommagés ou même simplement sales.

Lors de l'arrêt long I, toutes les pannes de connections ont été réparées lorsque c'était possible : nettoyage des liaisons optiques, ajustement des tensions, remplacement des cartes défaillantes, ou simple démasquage de canaux ayant changé de comportement. Cette campagne de démasquage a conduit à une proportion de masquage de 0.2% dans le tonneau (126 cristaux) et 1.3% dans les bouchons (190 cristaux). Enfin, une nouvelle politique de masquage des canaux a été mise

en place. L'idée était de ne pas considérer un lien comme définitivement défectueux mais de le masquer sur détection d'un taux de *trigger* trop élevé puis de le démasquer à la prochaine configuration. Cela permet d'éviter de masquer définitivement un tour qui pourrait redevenir utilisable dans le futur.

Pour cela, deux mécanismes ont été mis en place au niveau du firmware des cartes TCC

- Le premier, appelé Link Swing Detector, permet de détecter les bagottements des liens optiques. Il compte les changements d'état du lien et décide de la masquer si ce compteur dépasse une limite paramétrable.
- Le second, appelé Cumulative Overflow Killing Engine, permet de masquer les liens sur lequel les niveaux d'énergie sont jugés trop haut. Un système de seuil permet de compter les niveaux jugés trop haut. Si leur nombre dépasse une limite paramétrable, le lien est masqué.

Le système a montré une excellente performance et a masqué avec succès tous les canaux bruyants tout en permettant leur utilisation ultérieure sans configuration particulière.

Le problème d'un tel système est qu'il permet un masquage totalement aveugle à la réalité globale du détecteur. En effet, les TCC n'ont qu'une vue très partielle qui correspond à une petite surface de détection. C'est pourquoi le superviseur doit surveiller en permanence que le masquage ne devient pas problématique pour l'efficacité du détecteur. Pour cela, le superviseur récupère en permanence les valeurs de masquage des canaux dans les registres des cartes TCC et en alimente le programme de surveillance global XMAS, ce qui permet à cette information d'être disponible dans l'ensemble du système et aux experts de réagir s'ils constatent un taux de masquage trop élevé.

Ce même système de récupération permet au superviseur de reporter tous ces masquages vers une interface graphique qui permet aux experts du *trigger* de les monitorer en temps-réel, mais aussi d'appliquer des actions correctives manuelles.

4.5 Identification d'erreurs graves

La détection et le masquage des canaux bruyants permet au détecteur CMS de fonctionner correctement en évitant les temps morts mais la partie corrective reste totalement manuelle. C'est pourquoi nous avons oeuvré pour essayer de fournir des outils de diagnostics en temps-réel afin de réunir des éléments qui pourrait mener, un jour, à la réparation automatique de ces dysfonctionnements.

Le principe est que les causes de bruit sont en nombres limitées et qu'il est parfois possible de décider leur cause la plus probable. Ainsi, si un token ring tombe en panne, huit tours de *trigger* seront masquées en même temps, avec une forme caractéristique. De même si l'alimentation des cartes ne fonctionne pas correctement, c'est un ensemble de tours de *trigger* qui vont être impactées selon une forme qui dépend de la géométrie du détecteur, qui est très variable en fonction de l'endroit où sont placés les canaux défectueux.

La figure 4.8 présente tous les motifs qui correspondent à une erreur sur un un groupe de cartes front-end (CCU). Ces motifs correspondent aussi à une perte d'efficacité très importante du détecteur car ce sont les configurations où des tours voisines sont masquées simultanément.

Un algorithme de détection de ces motifs a donc été implémenté dans le superviseur qui permet

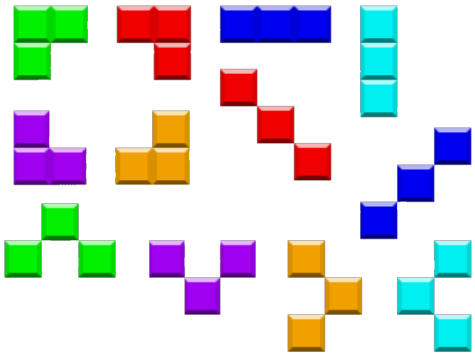


FIGURE 4.8 – Motifs de défaillance des tours de *trigger*

d'identifier ces cas problématiques. Pour cela, les informations de masquage sont récupérées au niveau du superviseur et à chacune des évolutions de ces informations, on va calculer le voisinage de chaque tour. Un simple comptage permet alors d'identifier les motifs de la figure 4.8.

Grâce à ce mécanisme, le superviseur peut identifier des causes de pannes et les transmettre à des programmes de correction.

4.6 SEU

Dans le système de contrôle de CMS, il existe un mécanisme qui permet de traiter les problèmes graves mais ne nécessitant pas une reconfiguration globale, source importante de temps mort. Ce mécanisme s'appelle le Single Event Upset (SEU) et il est décrit dans [38]. Son principe est simple, chaque fois qu'un superviseur détecte une erreur problématique mais non bloquante, il exécute une transition qui déclenche le SEU. Celui-ci est transmis au système de plus haut niveau qui va effectuer un pause dans l'acquisition. Un code particulier permet d'identifier le problème et de s'adresser au sous-système qui nécessite une action corrective. Sitôt que celui-ci a mené son action, l'acquisition est relancée, minimisant ainsi le temps de pause au strict nécessaire.

Dans notre cas, nous avons couplé le mécanisme de SEU au détecteur de motifs décrit précédemment. Cela permettra lorsque le superviseur de front-end le permettra d'effectuer des actions correctives automatiques sur les cartes de front-end, en particulier une reconfiguration des cartes incriminées.

4.7 Conclusion

Ce travail permet d'illustrer sur une véritable expérience que l'analyse des données en temps-réel permet d'améliorer substantiellement la performance des détecteurs de particules. Elle montre clairement aussi, qu'un système de détection n'est efficace que si le système de contrôle-commande peut effectuer des actions correctives en fonction de l'analyse. La partie détection et correction d'erreur a donné lieu à une publication [41] dans les actes de la conférence TWEPP 2018.

Chapitre 5

SiW-Ecal

Le projet SiW-Ecal est un projet visant à proposer un calorimètre électromagnétique(Ecal) pour le futur International Linear Collider (ILC) et son détecteur l'International Large Detector (ILD). Il s'agit d'un calorimètre à échantillonnage basé sur une alternance de détecteur en silicium(Si) et d'absorbeur en tungstène(W). Plusieurs prototypes ont été développés dans le cadre de la collaboration Calice (Calorimetry for ILC) et maintenant au sein de la collaboration naissante ILC/ILD. Le concept développé au LLR a été adapté pour d'autre détecteur comme CLIC et CEPC et a servi de source d'inspiration pour le HGCal de CMS.

5.1 ILC, ILD et Calice

L'International Linear Collider (ILC) est un projet d'accélérateur de particules. Il est décrit dans un document conséquent, le "Technical Design Report" (TDR) [33] auquel j'ai eu la chance de participer. Son principe est d'accélérer linéairement des électrons et des positrons et de les faire collisionner à une énergie au centre de masse comprise entre 250 et 500 GeV.

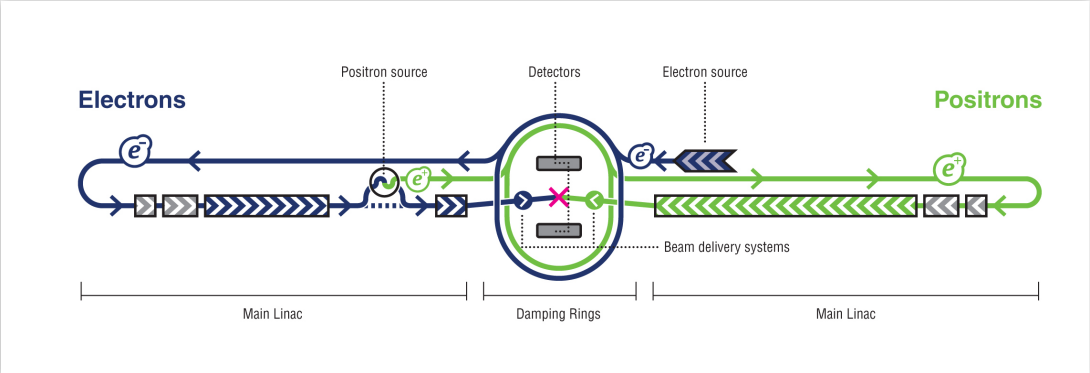


FIGURE 5.1 – Schéma de principe de l'ILC : les électrons et les positrons sont accélérés linéairement dans deux branches opposées de l'accélérateur pour venir se collisionner au centre. (Credit ILC collaboration)

La figure 5.1 montre une vue schématique de l'accélérateur. Des électrons et des positrons sont

produits, groupés et stockés dans des anneaux avant d'être collimatés et accélérés en direction opposée. L'accélérateur a une longueur globale de 20 à 31 kilomètres. Son implantation est prévue au nord du Japon, dans la région des montagnes Kitakami.

Atteindre une telle énergie en si peu de distance est rendu possible grâce à des cavités accélératrices très novatrices. Elles fonctionnent à très hautes fréquences (1.3 GHz), et sont constitués de niobium pur. Ces cavités supraconductrices présentent une capacité accélératrice record (25 ± 3.2 MV/m) mais elles imposent une température de fonctionnement très basse, de l'ordre de 9.2K. Elles ont été développées par la collaboration TESLA et décrites dans [18]. La figure 5.2 montre une telle cavité. L'avantage d'utiliser un linac, comparé aux accélérations circulaires du LHC, est l'absence de pertes radiatives (synchrotron) lors du passage dans les aimants. Une fois accélérés, électrons et positrons sont collisionnés au sein d'un détecteur multicouches.

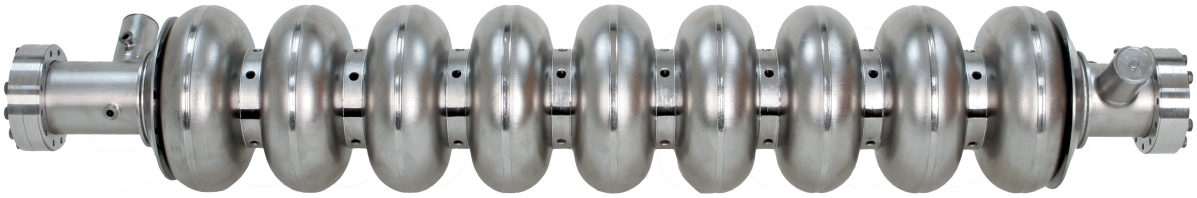


FIGURE 5.2 – Cavité accélératrice de l'ILC constitué de Niobium pur (Credit TESLA collaboration)

L'utilisation d'électrons et de positrons apporte de multiples avantages comparés aux hadrons. Contrairement aux protons, les électrons et positrons sont des particules élémentaires soumises uniquement à l'interaction électro-faible. Leur énergie au centre de masse ainsi que leur polarisation sont parfaitement connues et peuvent être ajustés. Leur collision produit peu de particules secondaires et en particulier aucun débris hadroniques, ce qui devrait faciliter grandement leur identification et l'évaluation de leurs paramètres (énergie, impulsion transverse...).

L'ILC a été conçu comme une usine à Higgs, c'est à dire un collisionneur optimisé pour les processus de production du boson de Higgs. Les processus principaux de productions sont décrits sur la figure 5.3. Il s'agit du Higgsstrahlung, où l'annihilation électron/positron produit directement un boson de Higgs et un boson Z : $e^+e^- \to ZH$. Le deuxième processus est la fusion de W : $e^+e^- \to \nu\nu H$ et le troisième est la fusion de Z : $e^+e^- \to e^+e^- H$.

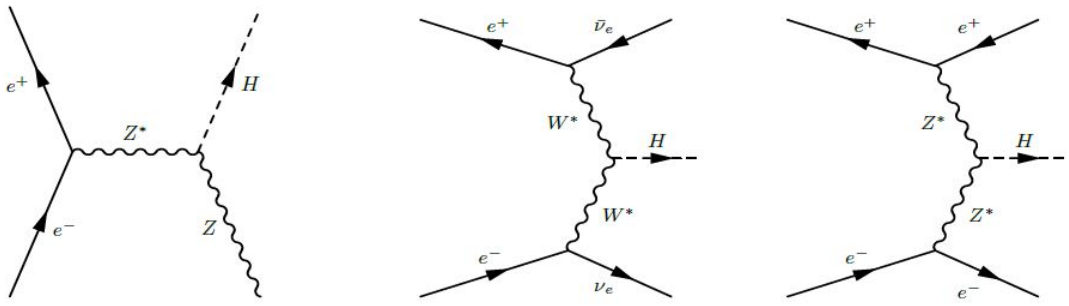


FIGURE 5.3 – Processus de production de boson de Higgs dans un collisionneur e^+e^-. De gauche à droite, Higgsstrahlung, fusion de W et fusion de Z.

La figure 5.4 montre les sections efficaces des différents processus à l'oeuvre dans un collisionneur électron/positron de type ILC en fonction de l'énergie dans le centre de masse. L'ILC a été prévu pour fonctionner dans un premier temps à 250 GeV, valeur qui optimise la production de Higgs

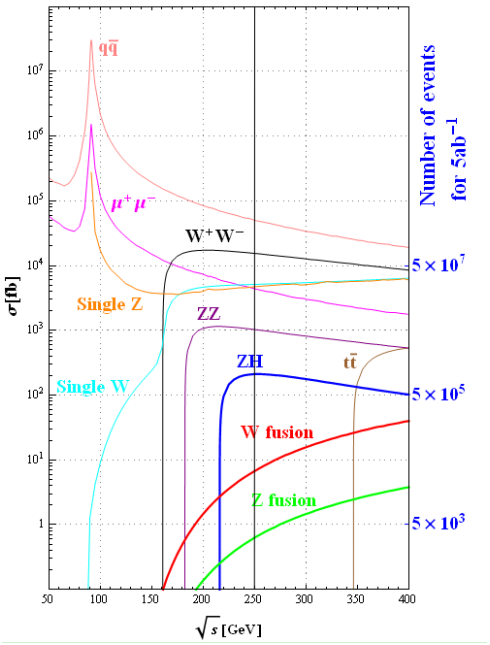


FIGURE 5.4 – Section efficace dans l'ILC. L'énergie cible est 250 GeV afin de maximiser la production de bosons de Higgs par Higgsstrahlung (courbe bleu foncé indiquée ZH)

par Higgsstrahlung (courbe bleu foncé indiquée ZH). On voit aussi qu'à cette énergie, le rapport de section efficace entre le Higgsstrahlung et le principal bruit de fond $e^+e^- \to q\bar{q}$ est de trois ordre de grandeur, ce qui est très peu comparé aux six ordres de grandeurs que l'on subit aujourd'hui sur le LHC. Dans un deuxième temps, l'ILC pourrait passer à 500 GeV, une énergie qui maximise la production de Higgs par la fusion de W (courbe rouge).

L'objectif principal de l'ILC est de faire des mesures de grandes précisions sur la masse du boson de Higgs. Pour obtenir ces précisions, le parti-pris sur l'ILC a été de favoriser la résolution en énergie sur les jets de désintégration des bosons (H,W,Z) à l'aide d'algorithmes, dits de "particule flow", combinant au mieux l'information des trajectographes et des calorimètres [42]. Ces algorithmes nécessitent une granularité très fine des sous-détecteurs et en particulier des calorimètres à pixels fins.

L'ILC est actuellement prévu pour héberger deux détecteurs interchangeables, l'International Linear Detector (ILD) et le Silicon Detector (SiD). Ils seront montés sur un système coulissant qui permettra d'amener l'un ou l'autre au centre de la collision.

L'ILD, présenté sur la figure 5.1 est un système de détection de particule qui combine un tracking de grande précision, réalisé avec une *TPC* et des calorimètres ultra-granulaires, compatibles avec les algorithmes de particle flow. Ainsi, la taille des pixels est de l'ordre du centimètre carré pour le calorimètre hadronique et du quart de centimètre carré pour le calorimètre électromagnétique. L'ILD a été décrit précisément dans le volume 4 du TDR ILC [13].

La construction d'un tel détecteur est surtout liée à la capacité de développer des calorimètres d'une finesse d'un ordre de grandeur plus petit que les existants. Afin de mener à bien cette mission, la collaboration CALICE a été montée. C'est un groupe de recherche et développement qui regroupe 340 physiciens et ingénieurs provenant de 53 instituts et 17 pays de 4 continents (Afrique, Amérique, Asie et Europe). Les groupes essayent différentes technologies afin d'évaluer

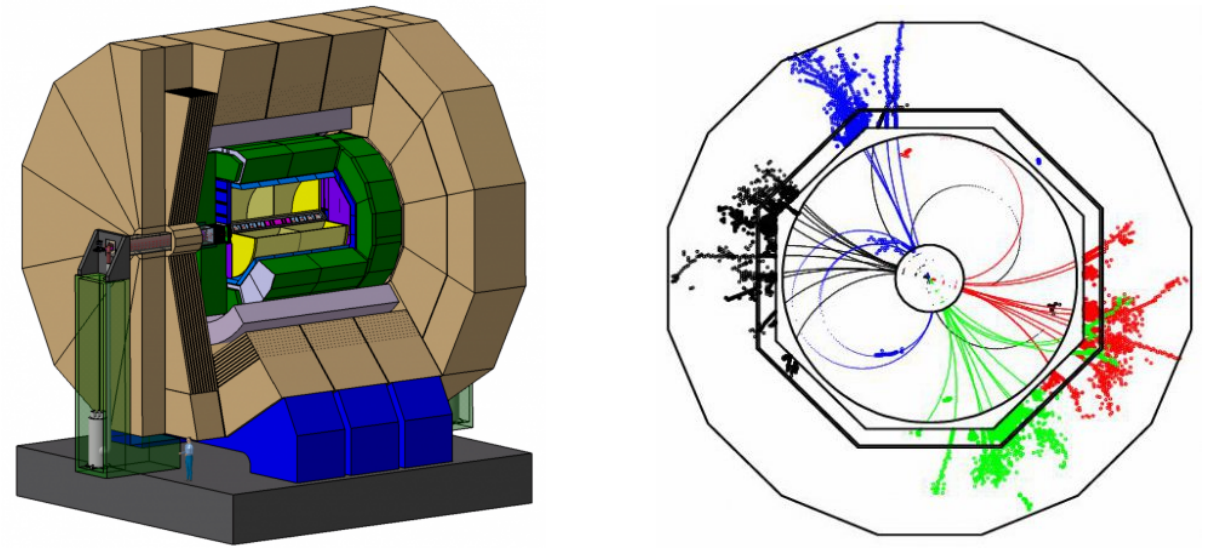


FIGURE 5.5 – Vue en coupe du détecteur ILD et Trace simulée d'une collision au sein de l'ILD (Credit ILD collaboration)

les meilleures pour la réalisation de l'ILD.

Pour la calorimétrie électromagnétique, trois projets sont actifs :

- SC-ECAL : calorimètre basé sur l'utilisation de scintillateurs et de photo-multiplicateurs en silicium (SiPM), décrit dans [45].
- MAPS-ECAL : calorimètre digital basé sur des matrices de pixels de silicium (MAPS), décrit dans [47].
- SiW-ECAL : calorimètre basé sur les matrices de diodes PIN comme détecteur et de tungstène comme matériau absorbeur.

Pour la calorimétrie hadronique, trois projets sont également actifs :

- AHCAL : calorimètre analogique basé sur des scintillateurs et des Photo-multiplicateurs au silicium (SiPM), décrit dans [23].
- DHCAL : calorimètre digital (1 seul seuil de déclenchement) basé sur des plaques de verre résistif, décrit dans [3]. Le projet est arreté.
- SDHCAL : calorimètre semi-digital (3 seuils) gazeux basé sur des plaques de verre résistif (GRPC), décrit dans [19].

Mon travail sur le calorimètre *SiW-Ecal* s'est intégré dans le projet CALICE, tout d'abord en temps que chef de projet sur le logiciel de *DAQ* (Calicoes) sous la direction de Rémi Cornat, puis en temps que coordinateur technique de l'ensemble des développements CALICE au LLR. Le projet évolue maintenant pour s'intégrer progressivement dans la collaboration ILD. Dans ce cadre, je suis le coordinateur technique de toutes les activités *DAQ* et prototypes au LLR, sous la direction de Vincent Boudry et Vladislav Balagura. Je suis également directeur technique de tous les tests en faisceau de ces détecteurs.

5.2 Le calorimètre électromagnétique *SiW-Ecal*

SiW-Ecal est un calorimètre à échantillonnage (sampling calorimeter) à très haute granularité. Il est constitué d'une alternance de couches de détection très fines et de couches d'absorbeur en tungstène. Au sein de cet absorbeur, les photons et les électrons incidents vont cascader, par une alternance de radiation de freinage (bremsstrahlung) : $e \to e + \gamma$ et de création de paires : $\gamma \to e^+ + e^-$, formant des gerbes électromagnétiques, comme illustré sur la figure 5.6.

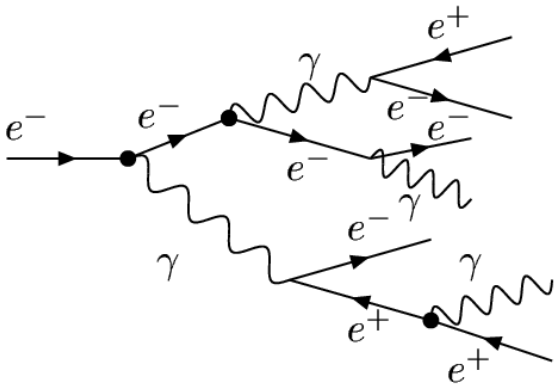


FIGURE 5.6 – Gerbe électromagnétique : les photons sont convertis en paires éléctron/positron qui produisent eux-même des photons par bremsstrahlung.

La surface sensible des couches de détection est constituée de silicium dopé structuré en pixel. Ces surfaces sont appelées *wafers* (galettes de silicium). Dans le silicium, aux énergies au dela du MeV, les électrons et positrons des gerbes vont essentiellement ioniser le milieu et créer des paires électrons/trous de basse énergie. Les *wafers* faisant quelques centaines de microns, le dépôt est suffisant pour qu'une électronique à bas-bruit lise la charge directement. Le détecteur inclut suffisamment de tungstène ($24X_0$) pour que les photons et les électrons soient tous arrêtés par l'absorbeur de façon à mesurer leur énergie totale.

Un prototype physique a été construit en 2004, démontrant la pertinence du concept et sa faisabilité technique, tel que décrit dans [20], avec 30 couches et des pixels de 10x10 mm^2 munis d'une électronique de lecture externe.

La particularité du SiW-Ecal est l'extrême finesse de ses pixels : 5x5 mm^2. Cette finesse permet d'utiliser des algorithmes de particle flow (PFA) pour reconstruire plus finement les traces, séparer des gerbes des différentes particules, et ainsi obtenir une précision accrue sur les mesures d'énergie mais également de rejeter plus de bruit de fond par une identification plus précise des particules en jeu dans la collision.

Cette granularité du détecteur induit un grand nombre de canaux : 4000 canaux par dm^3, soit 100 millions de canaux en tout. Ceci n'est possible que par l'utilisation d'une électronique de lecture intégrée dans le détecteur. Une autre particularité du détecteur est l'utilisation du power-pulsing. En effet, le faisceau de l'ILC est structuré en trains assez lents (5 ou 10 Hz), induisant des temps morts pour l'électronique de lecture. Par conséquent, il est inutile de maintenir le calorimètre actif tout le temps car cela implique une consommation électrique accrue et donc une dissipation thermique inutile. Aussi pendant les phases sans faisceau, l'alimentation de

l'électronique de lecture est coupé juste après la lecture des données.

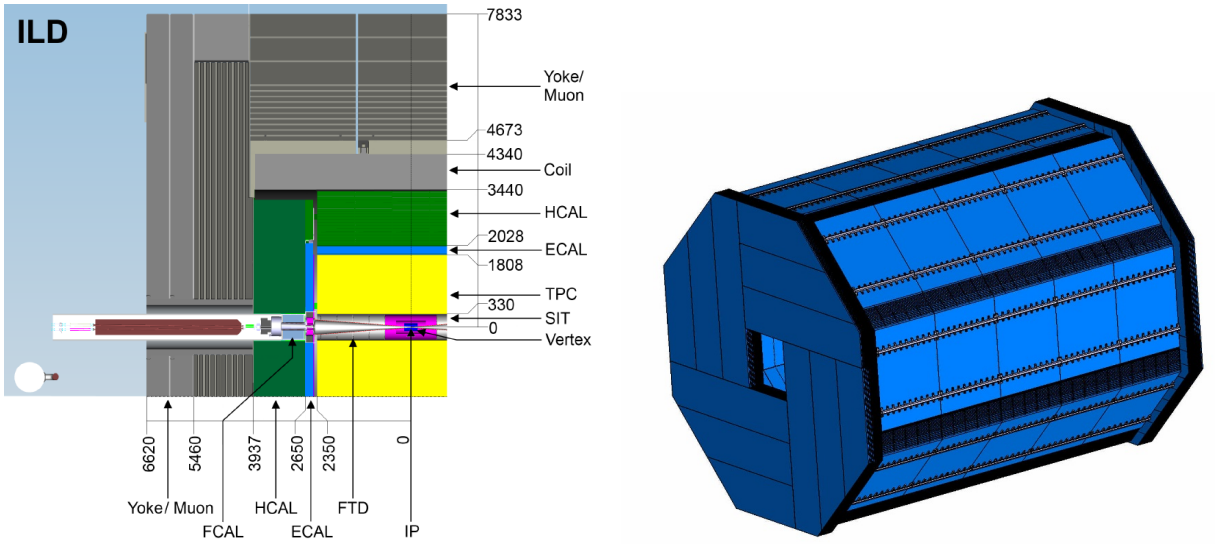


FIGURE 5.7 – Le calorimètre électromagnétique dans l'ILD. A gauche, coupe axio-radiale d'un quart de l'ILD. A droite, vue isométrique du ECal. (Credit M. Anduze, LLR)

SiW-Ecal est un candidat pour devenir le calorimètre électromagnétique de l'ILD. Il a été décrit en 2002 par H. Videau et J-C Brient à la conférence Calor [32]. En conséquence, il doit respecter les contraintes mécaniques correspondantes. Comme le montre la figure 5.7, le détecteur a une forme octogonale, de rayon interne 1.8 m sur 4.6 m de long. Comme on le voit sur la coupe axio-radiale, où il est représenté en bleu, il est très fin par rapport aux autres sous-détecteurs. Naturellement, il est placé entre la *TPC* et le calorimètre hadronique. Son design n'est pas encore totalement fixé mais l'épaisseur de devrait pas excéder une vingtaine de centimètre dans laquelle doivent venir se loger entre 20 et 30 couches avec leurs absorbeurs et leur système de refroidissement.

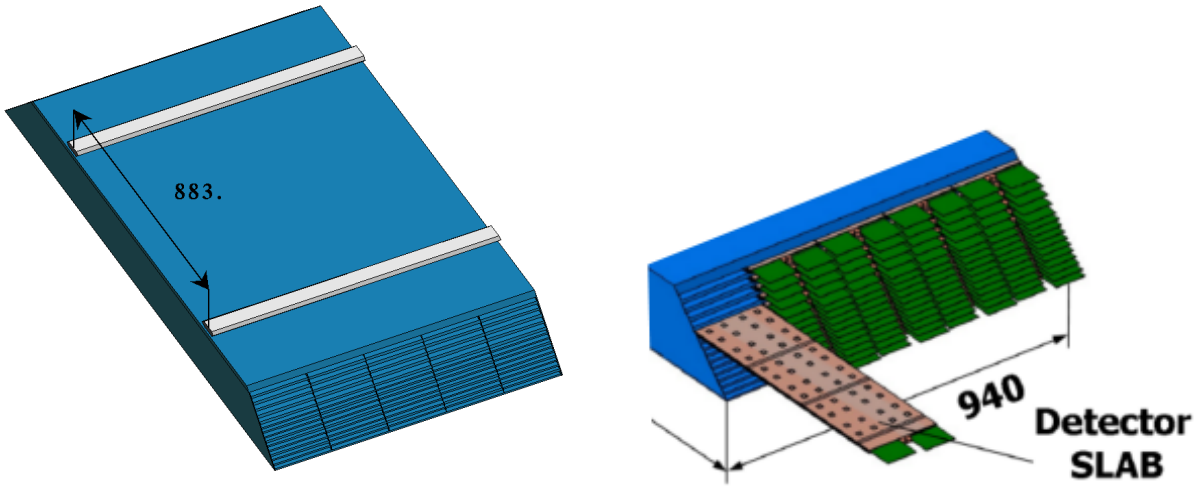


FIGURE 5.8 – Les modules du SiW-Ecal (Credit M. Anduze, LLR)

Le tonneau est constitué de modules trapézoidaux, visibles sur la figure 5.8. Chaque module est constitué d'un structure en carbone formant des alvéoles dans lesquelles viennent se placer les

cassettes de détection. Les bouchons utilisent des éléments similaires en forme de coin (2 ou 3 par quadrant en fonction des options de design).

5.2.1 *Slabs*

Les cassettes de détection munies de leur absorbeurs, sont appelées *slabs* et sont empilées dans les modules. Ce sont des structures en forme de H fabriqués en composite résine-fibre de carbone et qui forment un double berceau pour recevoir deux surfaces de détection. Ils sont décrits avec précision dans [1]. Leur dimension longitudinale est de l'ordre de 1.5 m (plus ou moins dépendant du modèle de détecteur et du positionnement du slab dans les modules). Ils ont une structure de sandwich qui intègre tous les éléments nécessaires à leur fonctionnement, l'absorbeur (plaque de tungstène), la surface de détection (*wafer*), l'électronique de lecture (*PCB* et *ASICs*) et une plaque de refroidissement passif (cuivre). La figure 5.9 montre cette structuration pour l'une des faces. La connexion électrique entre les couches est assurée par de la colle rendue conductrice par des micro-billes d'argent. La rigidité de l'ensemble est garantie par la rigidité de l'absorbeur et de la plaque de diffusion thermique.

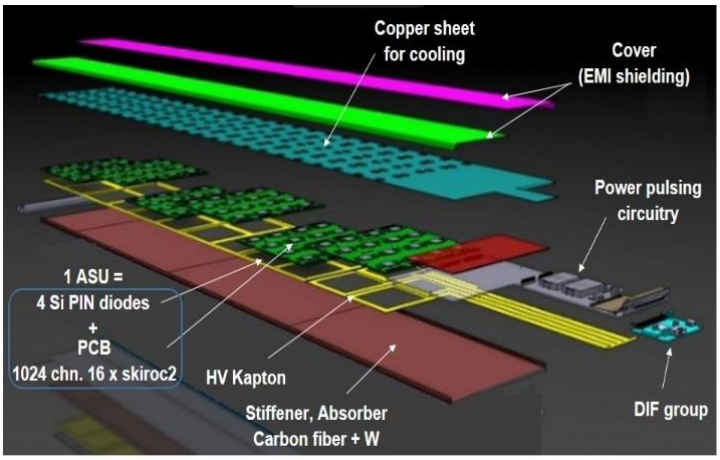


FIGURE 5.9 – Couches successives d'un demi-slab. Tous les éléments se retrouve à l'identique dans la partie inférieure du H. (Credit M. Frotin, LLR)

Naturellement, pour des raisons pratiques, il est impossible de produire des *PCBs* ou des surfaces de silicium de la longueur d'un slab. En conséquence, la partie centrale regroupant ces deux parties est segmentée en cartes carrées d'une vingtaine de centimètres de côté. Ces cartes, appelés des "Assembly single unit" (*ASUs*), sont constituées d'un *PCB* biface. Sur l'une des faces, on soude les composants de lecture et sur l'autre, on vient coller les *wafers*. Ce sont des galettes de silicium de haute résistivité, carrées de 9 cm de côté, produite par la société Hamamatsu.

La partie gauche de la figure 5.10 montre l'un de ces *wafers*. Elle se présente comme une matrice de pixels. Chacun de ses pixels est une diode PIN dont le schéma de principe est montré sur la partie centrale de la figure. C'est une jonction P/Intrisèque/N d'où son nom, où les parties P et N sont constituées de silicium fortement dopées, alors que la partie intrinsèque est constituée de silicium très faiblement dopé, formant une zone hautement résistive.

Pour son utilisation en détecteur de particule, la diode doit être polarisée en inverse par une haute tension. Celle-ci est choisie en fonction de la forme de la caractéristique du wafer (représentée sur la partie droite de la figure 5.10). Il faut choisir une tension sur le plateau qui soit un

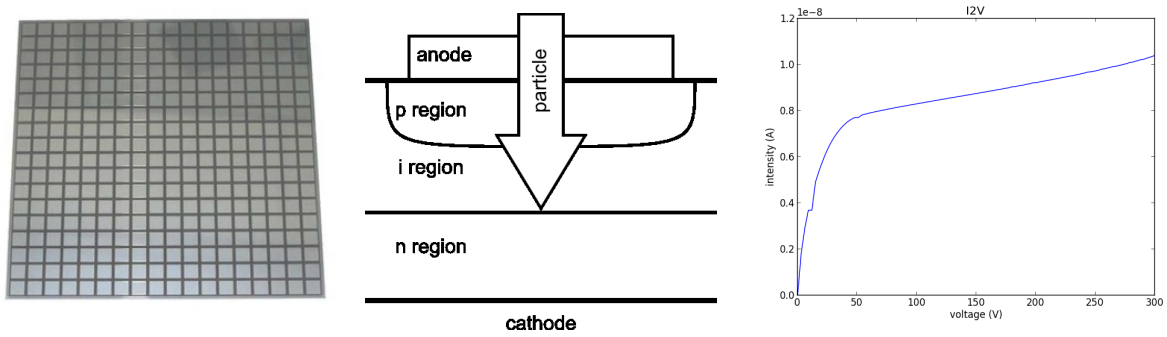


FIGURE 5.10 – *Wafer* 18x18 ($9cm^2$), schéma de principe de la diode PIN et courbe caractéristique d'un wafer silicium

compromis entre une déplétion complète et la génération d'un courant de fuite trop important. Sur un wafer de 300 μm comme celui dont on peut voir la caractéristique sur la figure, on choisira typiquement 100V, afin d'être au plus bas dans la partie plateau de la courbe. Quand une particule suffisamment énergétique traverse la région déplétée de la diode, elle crée des paires électron/trou. Grâce à la polarisation inverse, les charges vont migrer hors de la zone déplétée, créant un courant qui sera lu par l'électronique adaptée. Les *wafers* sont collés avec une colle conductrice au dos des *PCBs* sur des électrodes de la même taille que les pixels. A ces électrodes sont connectés les pattes de l'*ASIC* qui assure la lecture.

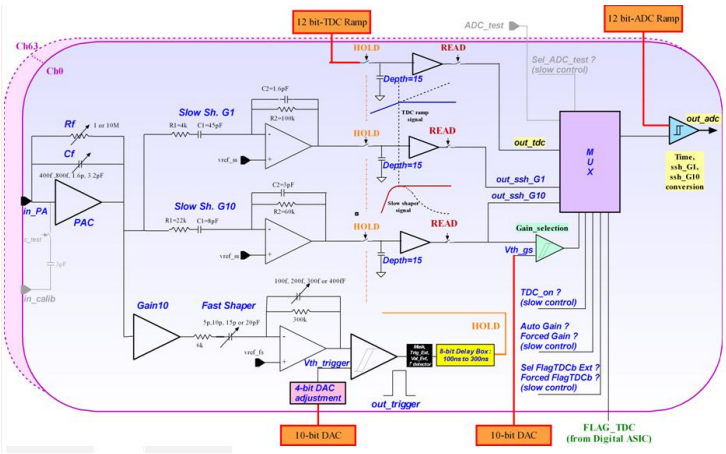


FIGURE 5.11 – Schéma de principe de l'ASIC Skiroc 2 (Credit S. Callier, OMEGA)

Cet *ASIC*, nommé Skiroc 2, est développé par l'unité OMEGA du CNRS. Le Skiroc 2 lit 64 voix en parallèle, comporte une partie analogique qui amplifie et intègre le courant sur chacune des électrodes et le compare à un seuil, comme on le voit sur la figure 5.11. Si le seuil est atteint sur l'une des voix, les valeurs de tous les canaux sont enregistrées dans des mémoires analogiques avec l'étiquette temporelle (énergie à bas gain, énergie à haut gain et éventuellement rampe de retard (TDC)). Ce système, appelé auto-trigger, élimine les besoins en *trigger* central externe et simplifie grandement l'architecture de l'électronique de lecture. Lorsque l'*ASIC* est plein (il possède 15 mémoires) ou lorsque l'acquisition est finie, les mémoires analogiques sont numérisées puis transmises à la *DAQ* (readout), avec le numéro d'ASIC et les étiquettes temporelles. On peut trouver une description complète de l'*ASIC* dans l'article de référence [6]. Les *ASUs* occupent toute la largeur des *slabs* soit 18cm. C'est pourquoi ils peuvent porter 4 *wafers* de 16x16 pixels et 16 *ASICs* Skiroc 2 pour leur lecture.

5.3 *DAQ*

Nous l'avons vu, les *ASICs* Skiroc 2 lisent les énergies et les stockent dans leur mémoire avant de les envoyer vers la *DAQ*. Le rôle de cette dernière est multiple. Elle doit configurer les circuits et leurs périphériques (alimentations haute tension), elle doit distribuer une horloge cohérente vers l'ensemble des circuits. Elle doit collecter les données lues et les agréger. Enfin, elle doit les stocker mais aussi les distribuer pour permettre un monitoring *online*.

Ces différentes fonctions sont assurées par six cartes électroniques et un ordinateur qui s'occupent chacun d'une fonction spécifique. La figure 5.12 présente une vue d'ensemble de l'interconnexion de ces différentes cartes.

L'*ASU* (caché sur la photo par le cache de protection) est la carte qui porte les *wafers* et les circuits Skiroc 2. Sur un slab pour l'ILD, ces cartes sont nombreuses et connectés entre elles. Au bout de cette chaîne d'*ASUs*, une carte adaptatrice nommée Sweat Mezzanine Board (*SMB*) apporte les services globaux à la chaîne : alimentation découplée, monitoring, connexion vers l'agrégation. Connecté à cette carte adaptatrice, on trouve la *DIF*, pour Detector Interface, une carte de collecte et de formatage des données. Elle est chargée du séquencement des opérations d'acquisition et de lecture. Ces *DIFs* (il y en a une par slab) sont connectées à des cartes d'agrégation, les *DCC* (Data Concentrator Cards) puis les *GDCC* (Gigabit DCC). Ces cartes agrègent les données et les transmettent au niveau supérieur. Elle servent aussi à diffuser les commandes et l'horloge qui proviennent de la CCC (control and command card) dont le rôle est de servir d'interface entre le calorimètre et le reste du détecteur. Enfin l'ordinateur reçoit les paquets de données des *GDCC* à travers le réseau Ethernet (cuivre ou fibre optique). Là, le logiciel prend les données en charge et les distribue selon plusieurs modalités, pour le stockage ou le monitoring temps-réel.

Cette DAQ est développée et améliorée au LLR depuis de nombreuses années et a donné lieu à deux publications dans les conférences TWEPP'13 [14] et TIPP'14 [29].

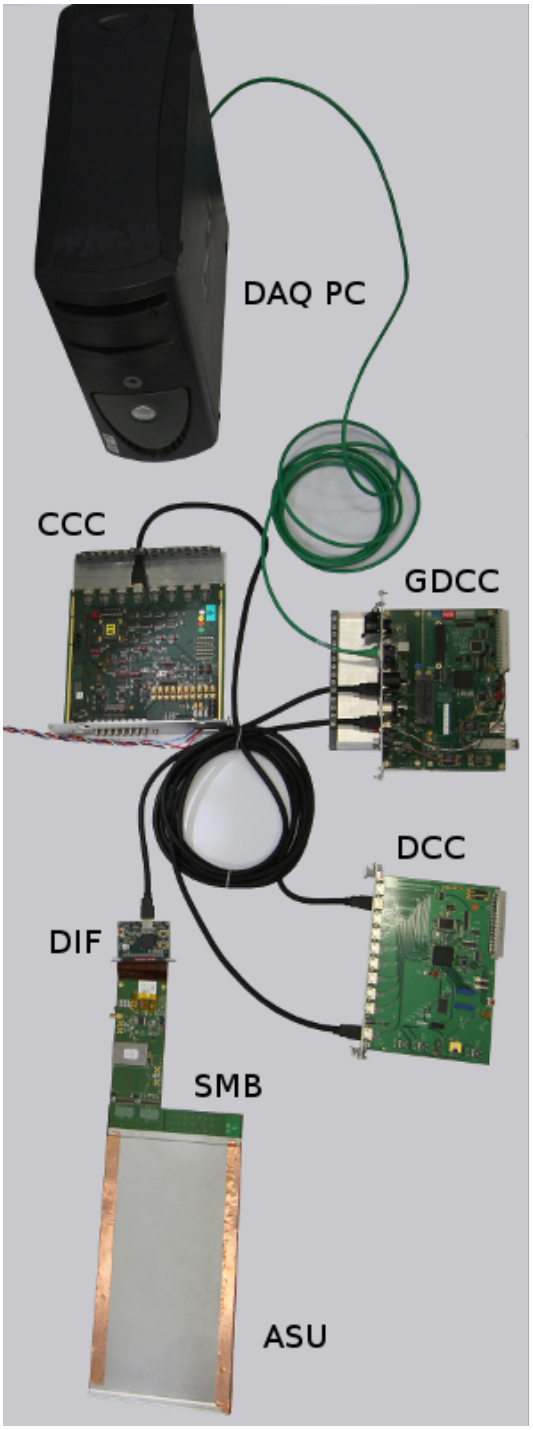


FIGURE 5.12 – Vue générale de la *DAQ*

5.3.1 *ASUs*

L'*ASU* est la carte exposée au flux des particules. Comme décrit précédemment, elle porte les *wafers* et les *ASICs* pour les lire. Comme on le voit sur la figure 5.13, la carte est double face. D'un côté, elle porte une matrice de pixels, destiné à venir en contact avec le *wafer*, collé avec la colle conductrice. De l'autre, sont soudés les composants électroniques. Les Skirocs 2, packagés dans des boitiers BGA, pour limiter leur épaisseur, mais aussi des capacités pour le filtrage des alimentations et le découplage. Sur cette même face, l'*ASU* a deux série de pattes de connexion afin d'être interconnectée à d'autres *ASUs* ou à la carte adaptatrice *SMB*.

Cette carte a été conçue, testée et développée au LLR, par Rémi Cornat, Stéphane Callier et Marc Louzir puis par Jérôme Nanni, Rémi Guillaumat et moi-même. Nous sommes aujourd'hui en train de tester la version 13, ce qui signifie que la carte, bien que d'un concept simple est délicate à réaliser. En effet, les *ASICs* Skiroc 2 ont une sensibilité extraordinaire (les courants mesurés sont de l'ordre de quelques dizaines de milliers d'électrons). Le prix a payer pour une telle précision est un environnement très peu perturbatif. En particulier les alimentations des *ASICs* doivent être parfaitement stables et peu parasitées par les signaux numériques de configuration, d'horloge ou de transfert de données.

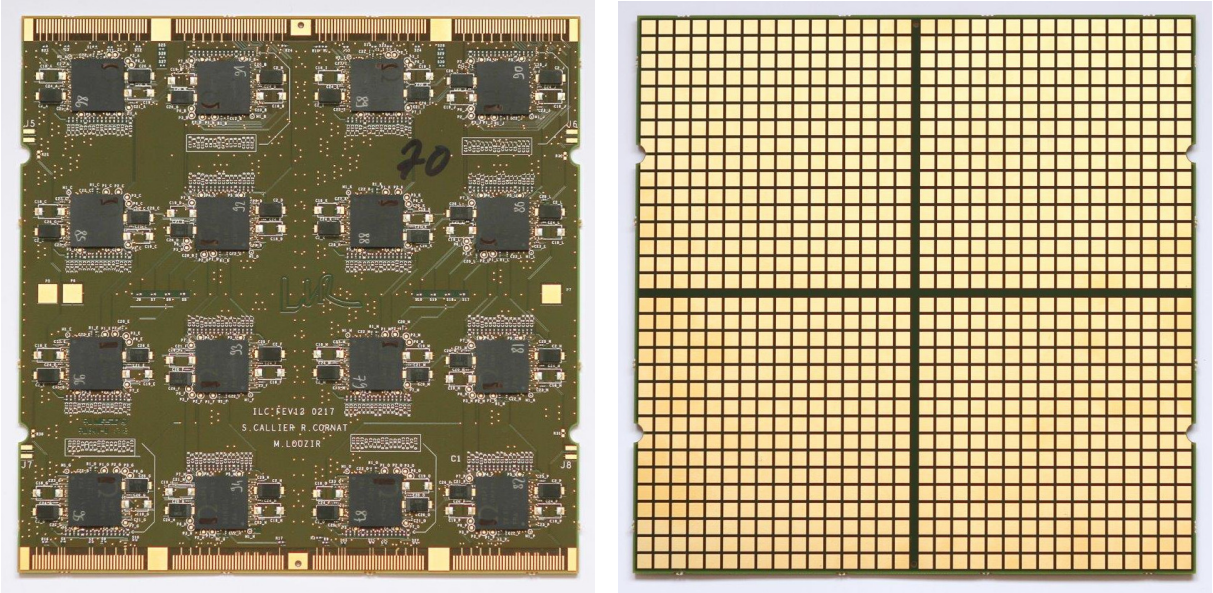


FIGURE 5.13 – PCB de l'*ASU* version 11, côté *ASIC* et coté détection

Fort heureusement, le timing de l'expérience aide considérablement. En effet, les phases d'acquisition sont bien séparées des phases de transfert de données ou des transitoires dues au power-pulsing. Toutefois, quelques signaux électroniques doivent bouger pendant l'acquisition. C'est le cas en particulier des horloges, mais aussi des signaux qui indiquent le début de l'acquisition (startacq), mémoire pleine (chipsat) ou d'interruption d'acquisition (busy). Ces signaux doivent donc être parfaitement isolées des plans d'alimentations des *ASICs*.

Cette isolation est obtenue en intercalant des plans de masse entre les plans de routage de ces différents signaux. Ainsi, l'*ASU* comporte plusieurs de ces plans de masse, couches de cuivre couvrant l'intégralité de la surface de la carte et noyées dans le *PCB*. De même, au niveau des interconnexions, les signaux perturbants sont séparés des lignes d'alimentation par des pattes à la masse.

Ces précautions, alliées à une conception respectant toutes les techniques métier de l'électronique bas-bruit nous permettent d'obtenir un rapport signal sur bruit exceptionnel, de l'ordre de 20, ainsi que nous le verrons plus tard dans la section des résultats 5.5.2.

5.3.2 *SMB*

La *SMB* est une carte adaptatrice qui vient se positionner au bout de la chaîne d'*ASUs*. Comme on le voit sur la figure 5.14, elle est connectée à gauche à la *DIF* et à droite au premier *ASU* de la chaîne. C'est une carte assez imposante dont nous reprenons le design actuellement en collaboration avec l'université de Kyushu pour s'adapter aux contraintes du futur détecteur.

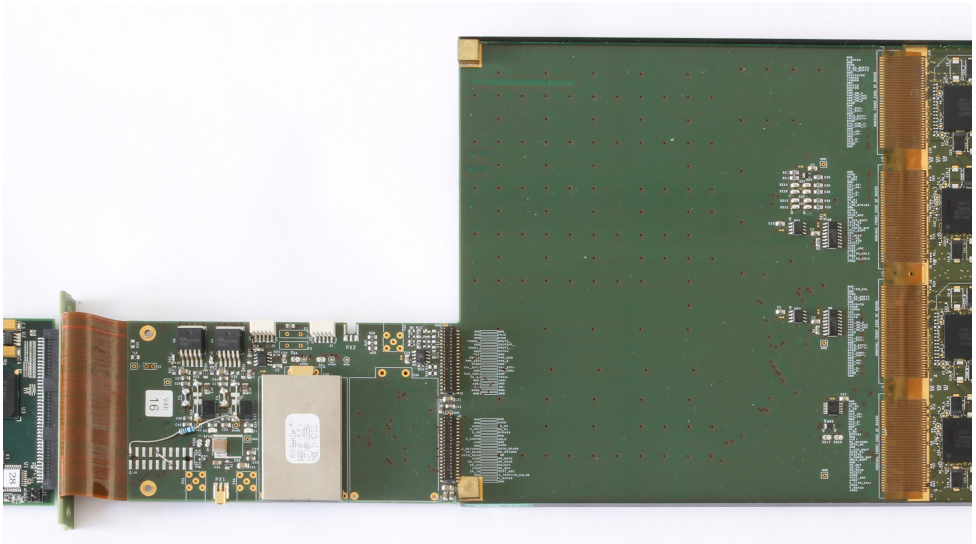


FIGURE 5.14 – Carte adaptatrice *SMB*

Elle assure trois rôles

- Distribuer des alimentations stabilisées : elle reçoit sur ses connecteurs des alimentations haute et basse tension qu'elle va répartir sur les différentes pistes pour alimenter les *wafers* et les *ASICs*. Elle dispose de deux régulateurs pour ajuster au mieux les tensions délivrées. On le voit sur la figure, elle porte également une imposante capacité dont le rôle est de fournir du courant lors du réveil des *ASICs* pour le power-pulsing. Dans la nouvelle version, cette capacité est répartie sur les *ASU* pour minimiser les transferts de courant sur les liaisons ainsi que l'encombrement sur la *SMB*.
- Distribuer les signaux : les signaux de contrôles, les horloges ainsi que les configurations proviennent de la *DIF*. La *SMB* a pour rôle de transmettre ces signaux à tous les *ASICs*, en les amplifiant si nécessaire au moyen de drivers.
- Informer la *DAQ* : la *SMB* porte un composant de la marque Dallas qui peut fournir un identifiant unique et une mesure de la température. Si l'information de température est peu pertinente en raison de son placement excentré, l'identifiant unique est une information intéressante pour marquer les données, adapter les configurations dynamiquement et faire du suivi à long terme de nos *slabs*.

5.3.3 *DIF*

La *DIF* (Detector InterFace) est une carte qui permet de configurer et de lire les *ASICs* pour remonter les données vers les concentrateurs. Son design est générique et permet d'utiliser différents type de *ASICs*, dont Skiroc 2 bien sûr. Elle a été développée par Rémi Cornat sur la base d'un design ancien. La figure 5.15 montre la carte qui est de la taille d'une carte de crédit. On peut voir sa connectivité :

- Un connecteur vers la *SMB* (en haut) qui sert à passer les horloges et les signaux pour les *ASICs*.
- Un connecteur HDMI (en bas, le 2e en partant de la gauche) pour la connexion vers les cartes d'agrégation.
- Un connecteur USB (à droite) pour un readout alternatif (que nous n'utilisons plus)
- Un connecteur JTAG pour la mise à jour du firmware (en bas le 3e en partant de la gauche)
- Un connecteur d'alimentation (en bas, le premier en partant de la gauche)
- Un connecteur de monitoring (en bas le dernier en partant de la gauche) qui permet de visualiser le fonctionnement de la carte avec des diodes luminescentes.

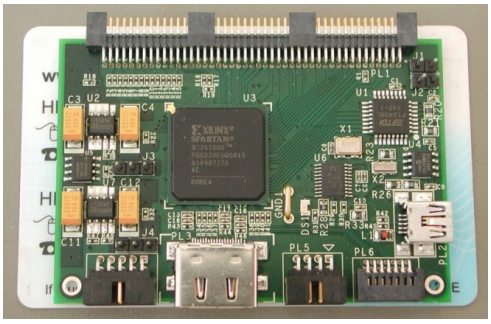


FIGURE 5.15 – Carte *DIF*

Le centre de la carte est occupé par un FPGA qui gère les protocoles vers l'agrégation et vers les *ASICs*.

Du côté *ASICs*, la *DIF* interprète les signaux envoyés par la carte de contrôle (CCC) et les convertit en signaux compréhensibles par les *ASICs*. Pour la configuration, elle contient une mémoire dans laquelle on peut charger les *bitstreams* de configuration des *ASICs*, puis sur sollicitation de la *DAQ*, elle peut envoyer ces *bitstreams* dans les *ASICs* et vérifier que le transfert s'est bien passé. Enfin, lors du readout, elle s'adresse au *ASICs* pour leur demander d'envoyer leur données.

Ces données sont formatée à l'intérieur du FPGA, c'est-à-dire que des balises sont insérées entre les blocs de données binaires. Ces balises sont de deux types :

- début de spill : un spill est un groupe d'acquisition correspondant à un groupe de collisions. La balise contient les informations correspondantes, en particulier le numéro unique identifiant le spill.
- fin de spill
- début d'*ASIC* : cette balise indique le début d'une plage de données correspondant à un spill entier sur un *ASIC*. Elle est accompagnée d'informations dont l'identifiant de l'*ASIC* correspondant.
- fin de *ASIC* : fin de la plage de données

Ces informations balisées sont envoyées telle quelle dans le lien HDMI vers les cartes d'agrégation en utilisant le protocole 8b/10b.

5.3.4 DCC

La *DCC* (Data Concentration Card) est une carte d'agrégation de données utilisant le protocole 8b/10b. Cette carte a été développée au LLR par Franck Gastaldi, initialement pour le SDHCal. Elle est conçue pour être connectée à 9 *DIFs* et une carte d'agrégation de niveau supérieur. Comme on le voit sur la figure 5.16, la carte possède une connectivité correspondante : neuf connecteurs HDMI pour les *DIFs* et un autre (le plus à droite) pour la connexion montante. Un connecteur (en haut) permet de mettre cette carte dans une châssis Schroff lui fournissant l'alimentation et une connectivité de fond de panier.

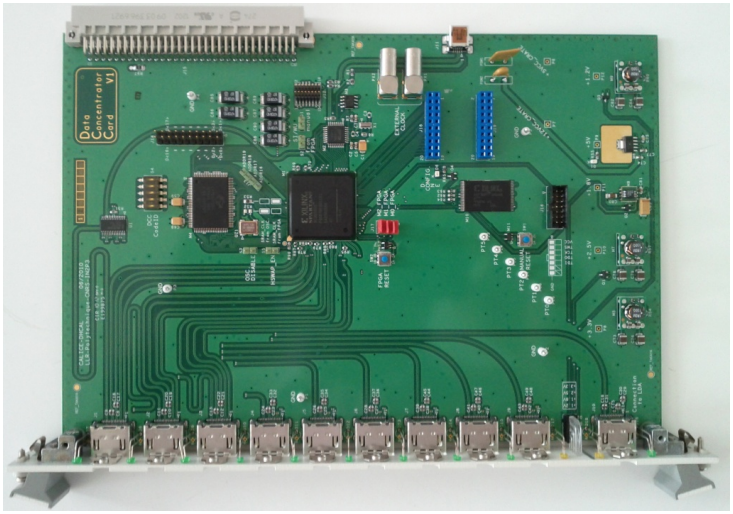


FIGURE 5.16 – Carte d'agrégation *DCC*

Un FPGA que l'on voit au centre de la carte implémente une sorte de multiplexeur qui va négocier avec les *DIFs* pour obtenir leur données et les mixer sur le lien montant. Elle va également distribuer les signaux provenant de la carte de contrôle vers les *DIFs*, sans les modifier. Le débit sur les liens entrants et le lien sortant étant le même, il permet de maximiser l'utilisation du lien montant. C'est une carte essentiellement passive qui ne nécessite pas de configuration particulière.

5.3.5 *GDCC*

La *GDCC* (Giga Data Concentrator Card) est une carte très similaire à la *DCC*, développée également par Franck Gastaldi au LLR. La différence majeur est que le lien montant est un lien Ethernet pouvant atteindre le Gigabit par seconde. C'est pourquoi, elle est capable d'agréger plusieurs *DCC*. Elle est représentée sur la figure 5.17. Le lien Ethernet peut utiliser le connecteur cuivre ou le connecteur fibre optique qui sont présent sur la carte en haut à droite.

Son rôle est d'être l'agrégateur de haut niveau de l'ILD et d'être connecté à un ordinateur

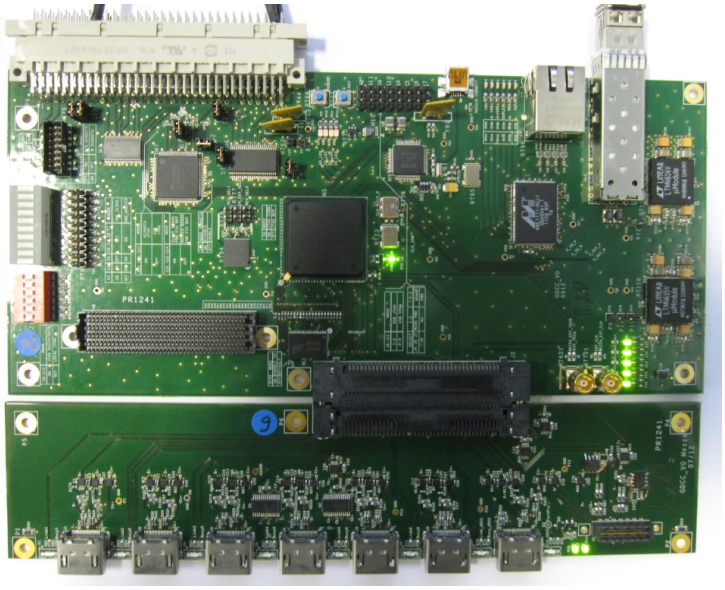


FIGURE 5.17 – Carte d'agrégation Giga-DCC

(éventuellement via un switch) qui lira les données pour le stockage et le traitement. Aujourd'hui elle utilise un format ethernet brut propriétaire qui présente des risques pour la fiabilité des transmissions à haut débit. Des travaux sont en cours pour migrer vers le protocole UDP, mieux supporté par les systèmes d'exploitation.

5.3.6 CCC

La CCC (control and command card) est une carte qui génère les signaux de contrôle afin d'ordonnancer les actions du détecteur. Comme le montre la figure 5.18, sa connectivité est constituée d'une rangée de connecteurs HDMI à droite destinés à se connecter aux *GDCC*. De l'autre côté, une série de connecteurs lemo permet de recevoir les signaux provenant de l'accélérateur ou d'un contrôleur central de l'ILD. En effet, dans ILC, les collisions ont une structure bien particulière. Elles sont regroupées au sein d'empilements, appelés spill, qui sont des séquences temporelles de collisions. Ce signal de spill est reçu par la CCC puis transformé en mot de contrôle. Ces mots sont diffusé par les liens HDMI à toutes les cartes *GDCC* qui les transmettent à leur tour aux *DIFs*.

A la réception de ce signal, les *DIFs* lancent une horloge lente correspondant à la fréquence des collisions pour déclencher l'acquisition. Les spills ont une fréquence typique de 5Hz et les collisions de 2.5MHz. Il est prévu de l'ordre de 3000 collisions par spill, pour une durée d'environ 1 ms.

5.4 Calicoes

L'ensemble du système ne peut fonctionner que s'il est piloté par un logiciel capable d'en gérer la complexité. C'est le rôle de Calicoes que je développe au LLR depuis 2012. Il est basé sur le *framework* Pyrame qui en est un spin-off. La plupart des fonctionnalités implémentées dans Pyrame ont été développée pour Calicoes à l'origine, ce qui en fait la réalisation la plus aboutie

FIGURE 5.18 – Carte d'horloge et de contrôle CCC

basée sur ce *framework*.

Afin d'assurer le contrôle-commande de la *DAQ*, Calicoes contient un ensemble de drivers permettant d'accéder à l'ensemble de ses cartes.

5.4.1 ASICs Omega

En premier lieu, un driver spécifique est fourni pour les différents *ASICs* de lecture supportés par la *DAQ* : Skiroc, Spiroc, Maroc et Easiroc. Ces modules permettent de générer les bistreams de configuration. Ces *bitstreams* sont formés par la concaténation de nombreux mots binaires de taille variable correspondant à des fonctionnalités particulières.

Par exemple, le bitstream du Skiroc 2 contient 1240 bits de configuration permettant d'ajuster de nombreux paramètres dont les principaux sont le mode et le seuil de *trigger*, le gain de DAC, la constante du fast-shaper, le délai de retenue (hold), les capacités de compensation et de feedback, le délai de *trigger*. Pour chaque canaux, on peut également régler individuellement l'allumage des pré-amplis, un ajustement de *DAQ*, le veto de *trigger* et un ajustement du niveau de *trigger*.

Ces paramètres seraient très fastidieux à configurer un à un dans le fichier de configuration. Chaque carte *ASU* possède 16 *ASICs* Skiroc 2, aussi le nombre de paramètres serait très grand. C'est pourquoi le mécanisme qui a été implémenté permet de simplifier cette configuration. Il est possible de spécifier un *bitstream* sous une forme hexadécimale ou binaire ou sous la forme d'un chemin pointant vers un fichier contenant le *bitstream*. Celui-ci sert de valeur par défaut et peut être surchargé par des configurations spécifiques, adressées par leur nom. Ainsi, on peut proposer un *bitstream* global pour l'ensemble du détecteur et n'écrire dans le fichier que les options spéciales, comme par exemple le seuil de *trigger*, très spécifique à la finalité physique du run.

5.4.2 Cartes de la *DAQ*

On peut configurer la *DIF* en lui envoyant des commandes à travers sa *GDCC* de rattachement. Elle accepte deux types de commandes :

- les fast-commands (FC) qui peuvent être envoyées par l'électronique, en particulier la CCC mais également par le logiciel. Elles servent à effectuer des actions à temps de latence controlé, pendant l'acquisition.
- Les blocks transfer commands (BTCMD) sont des commandes purement logicielles qui permettent de configurer les registres ou de déclencher des actions à destination des *ASICs*.

Pour la configuration des Skirocs, le module décrit précédemment génère le *bitstream*. Ensuite, le module de commande de la *DIF* l'envoie dans la mémoire de celle-ci et lance une BTCMD pour déclencher la configuration des *ASICs* et vérifier l'intégrité du *bitstream* transmis.

Les *DCC* et *GDCC* ne nécessitent pas de configuration particulière. Par contre, la *GDCC* doit être sollicitée pour lancer une procédure de synchronisation avec les *DIFs* qui lui sont connectées. Le logiciel peut effectuer un contrôle de cette synchronisation et ainsi vérifier l'état de la *DAQ* dans son intégralité avant de commencer l'acquisition.

5.4.3 Module d'acquisition

Un module spécifique s'occupe des aspects haut niveau de l'acquisition de données sur le système. Il est basé sur la chaine d'acquisition et sur le module storage. Il gère la destination des données, leur stockage et éventuellement l'exécution de scripts de conversion.

La chaîne d'acquisition est configurée de manière à construire un flux de données par *DIF*. Pour cela, une analyse temps-réel est effectuée sur les paquets de données afin de les affecter au bon canal. Cette tâche est effectuée par un plugin de décapsulation spécifique, capable de comprendre les entêtes des paquets de données et d'en vérifier l'intégrité et la nature, en particulier s'il s'agit de données ou de paquets de contrôle. Les données sont débarrassés de leur entêtes et injectées dans le bus mémoire afin d'être distribué selon les trois modalités : fichiers, connexion TCP et mémoires partagées.

Les paquets de données sont stockés dans un buffer circulaire où ils vont attendre une requête de lecture. S'ils ne sont pas lus, ils finiront par être supprimés pour libérer l'espace afin de stocker un nouveau message. Afin de retrouver les paquets de contrôle corrects, les *DIFs* et *GDCCs* implémentent un système d'identificateur de paquet. Lorsque le logiciel leur envoie une commande, celle-ci contient dans son entête, un champ spécial contenant un identificateur que le logiciel génère de manière aléatoire. La réponse porte le même identifiant, ce qui permettra de la retrouver dans la mémoire circulaire.

5.4.4 Module de contrôle principal

L'ensemble de la *DAQ* et du système nécessitent des procédures complexes devant être effectuée selon un timing bien précis. C'est pourquoi il est préférable que l'ensemble de ces opérations soient pilotées par le logiciel de contrôle. Il est indispensable que l'utilisateur final puisse interagir facilement avec le logiciel à travers une interface simple et intuitive. Pour simplifier ces échanges,

le module de contrôle principal implémente une abstraction qui représente l'état de l'ensemble des composants d'une manière schématique.

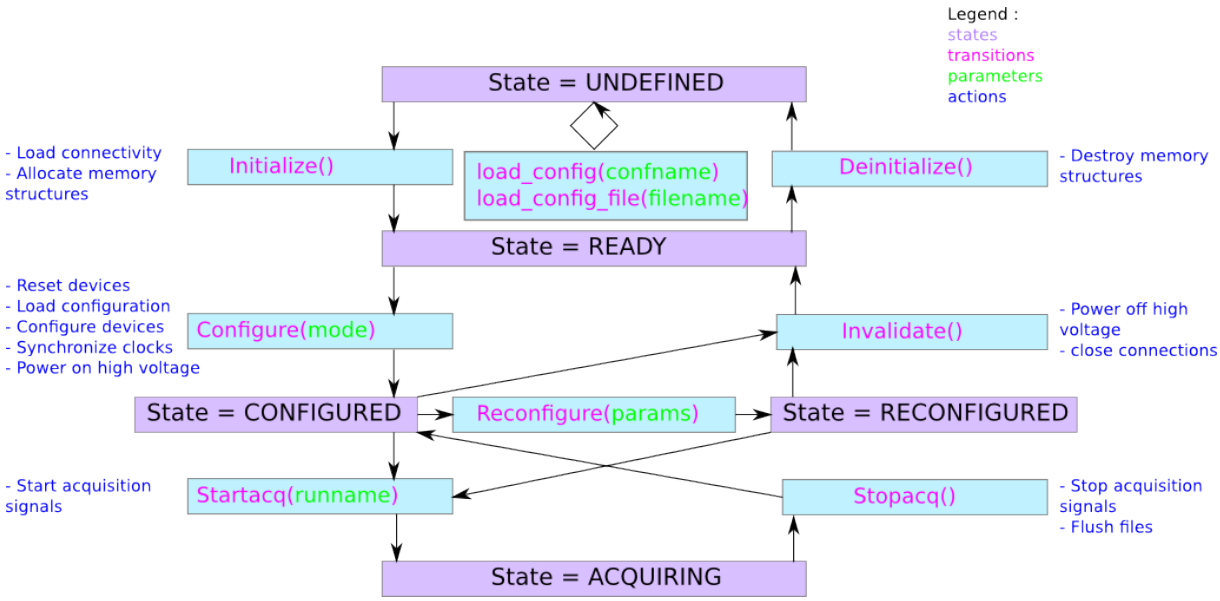


FIGURE 5.19 – Machine d'état du module de contrôle principal

La machine d'état, représentée sur la figure 5.19, ne comporte que cinq états mais ceux-ci permettent de connaitre l'état du détecteur. Au départ, la machine d'état démarre dans l'état "undefined", ce qui signifie que le logiciel n'a pas encore connaissance de la configuration. On doit alors charger un fichier de configuration et exécuter l'action "initialize" pour arriver dans l'état "ready". Il est important de noter que cette phase d'initialisation ne comporte aucune communication avec le matériel, c'est juste une initialisation logicielle.

La transition "configure" au contraire est celle où le logiciel va communiquer avec le matériel pour l'amener dans une configuration cohérente avec le fichier chargé. Cette transition mène, en cas de succès, dans l'état "configured".

A partir de cet état, la transition "startacq" permet de lancer un acquisition simple, qui correspond à l'état "acquiring". Cette transition prend en argument le nom de l'acquisition. La transition "stopacq" permet de quitter l'état d'acquisition pour rejoindre l'état "configured".

"invalidate" et "deinitialize" sont les transitions inverses de "configure" et "initialize".

Enfin, une transition "reconfigure" permet de reconfigurer, à partir de l'état "configured", n'importe quel composant du système. Grâce au mécanisme de synchronisation de l'environnement des modules, la modification est automatiquement reportée dans le module de configuration. La transition mène à un état "reconfigured" qui, dans les représentations graphiques, est confondue avec l'état "configured".

5.4.5 Interface homme-machine

L'interaction entre l'utilisateur et le logiciel peut se faire selon plusieurs modalités. Tout d'abord, une série de commandes peuvent être exécutées dans un terminal ou dans un script shell. Elles permettent d'effectuer manuellement des transitions de la machine d'état ou des reconfiguration.

D'autre part, une interface graphique est également proposée. Celle-ci, implémentée avec les technologies Web, est accessible au moyen d'un simple navigateur. Pour cela, elle utilise le binding javascript de Pyrame. Son design est adaptatif, ce qui permet de l'utiliser même sur un smartphone ou une tablette.

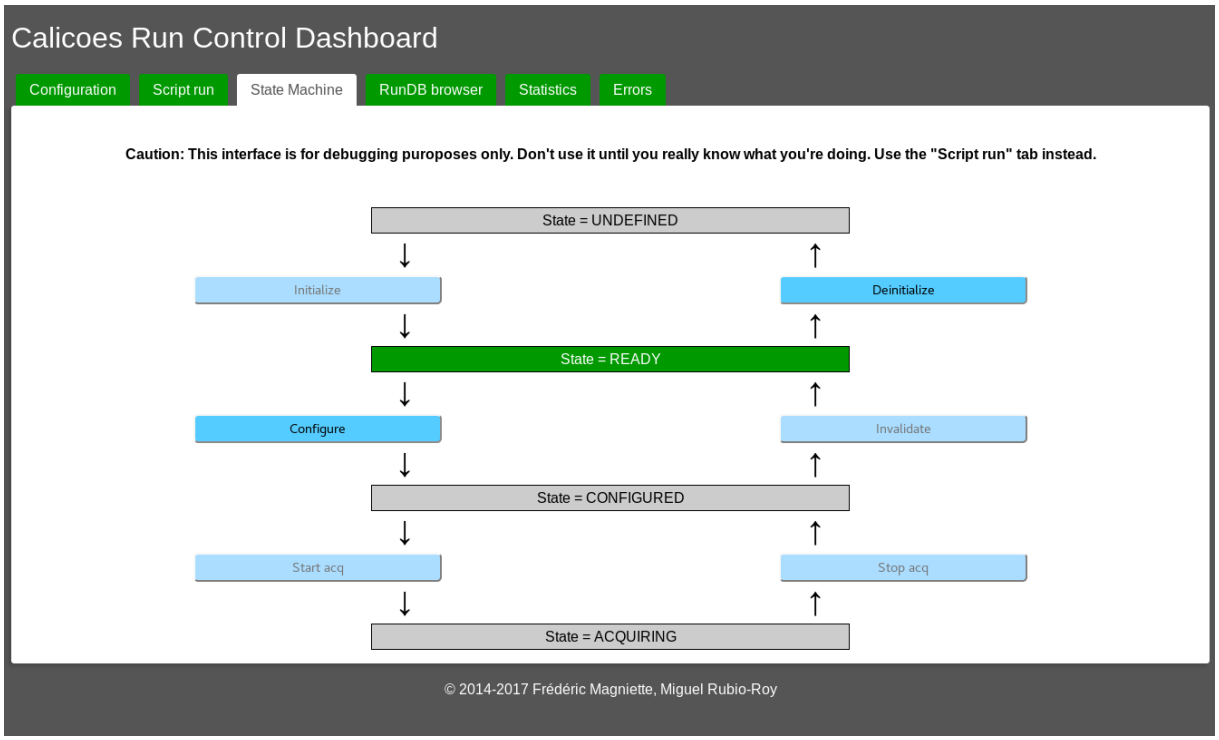


FIGURE 5.20 – Interface Web de Calicoes permettant de piloter dans la machine d'état du système

La figure 5.20 présente une vue de l'onglet principal de cette interface. On y voit la machine d'état précédemment décrite. L'état coloré en vert est l'état courant. Il suffit de cliquer sur les rectangles éclairés (bleu soutenu) pour déclencher les transitions.

Comme on le voit sur le bandeau supérieur, de nombreux autres onglets sont disponibles, dont je n'ai pas mis de représentation car leur forme est essentiellement textuelle. Ils permettent de choisir le fichier de configuration, d'exécuter un script. Les onglets suivants permettent d'accéder à un navigateur des prises de données (runs) passés avec un requêteur, une page de statistiques se mettant à jour en temps-réel dans l'état d'acquisition. Enfin le dernier onglet devient rouge si une erreur survient et présente un historique des messages correspondants.

5.4.6 Système de script

En plus des interfaces graphiques et textuelles, il est souvent indispensable de disposer d'un système de script qui permet d'interlacer des appels à Calicoes pour déclencher des acquisitions ou des reconfigurations, avec des programmes d'analyse ou des actions sur le matériel. Pour faciliter le développement de tels script, Calicoes s'appuye sur Python, une couche d'abstraction spécifique nommée Pycaldaq.

Pycaldaq est une librairie Python qui implémente des fonctions de haut-niveau permettant de contrôler facilement le détecteur. Il est basé sur le module run_control de Pyrame. Le principe

de Pycaldaq est simple, il suffit d'appeler ses fonctions dans un programme ou la console Python pour contrôler le détecteur.

Tout d'abord, Pycaldaq implémente les fonctions de pilotage de la machine d'état : initialize, configure, start_acq, stop_acq, invalidate et deinitialize. Aucune ne prend de paramètre sauf start_acq qui prend le nom de l'acquisition. Une fonction get_state permet de savoir à tout moment, dans quel état se trouve la machine d'état.

Pycaldaq fournit également la fonction reconfigure qui permet de reconfigurer n'importe quel composant du système. Pour cela, il suffit d'appeler la fonction reconfigure avec, en paramètre, le nom du composant tel qu'il apparait dans le fichier de configuration, le nom de la fonction que l'on souhaite exécuter et les paramètres de cette fonction.

Par exemple, supposons que l'on souhaite modifier le voltage de l'alimentation portant le nom "hv" dans le fichier de configuration, pour le passer à 20 volts. L'appel correct est

```
reconfigure("hv",set_voltage_ps,"20")
```

La machine d'état passera alors dans l'état "reconfigured".

Un cas particulier de contrôle du matériel est celui de la table XZ. Le prototype du *SiW-Ecal* peut être installé sur une table permettant de déplacer le détecteur suivant deux axes. Naturellement, les coordonnées de la table ne font pas partie de la configuration. On pourrait utiliser reconfigure pour lui envoyer des ordres de déplacement. Toutefois, afin de simplifier la tâche du *shifter*, Pycaldaq fournit un schéma de déplacement plus simple. la fonction table_goto prend en paramètre le nom d'un emplacement et exécute toutes les opérations nécessaires pour s'y rendre. Les emplacements peuvent avoir les noms "center", "zero" ou "gridX" avec X un entier de 1 à 81. Cette grille découpe l'espace en une matrice de 9 sur 9 points.

5.4.7 Acquisitions Pycaldaq

Au delà des actions unitaires déjà décrites, Pycaldaq offre également plusieurs procédures prédéfinies pour des actions plus complexes. Ces actions ont été regroupées à partir des développements les plus utiles et génériques, effectués par les utilisateurs.

Tout d'abord, trois acquisitions sont proposées :

- timed_acq est une acquisition dont la durée est spécifiée comme un paramètre exprimé en secondes. Son principal intérêt est d'être interruptible à tout moment via l'interface graphique.
- nb_hits_acq est une acquisition dans laquelle on garanti le nombre de hits produits. Ce comptage est obtenu via les décodeurs *online*s qui extraient, en temps-réel, les informations de la donnée brute. Ce type d'acquisition est intéressant pour garantir un effectif statistique suffisant pour faire une analyse.
- nb_spills_acq est une acquisition dont la durée est spécifiée mais en spills.

Un autre type de fonctionnalité complexe est la production de courbe en S. Cette courbe est une caractérisation du bruit électronique par rapport au seuil de *trigger* de l'*ASIC*. Le principe est de faire varier le seuil de *trigger* et pour chacune de ces valeurs, d'effectuer une acquisition de durée fixe. On représente alors le nombre de hits obtenus en fonction du seuil. La courbe présente un plateau qui correspond aux valeurs de seuil inférieur au bruit électronique, puis le plateau

tombe brutalement lorsque le seuil dépasse le bruit. Ensuite, la courbe reste nulle. La figure 5.21 montre un exemple de courbe en S effectuée sur huit canaux d'un Skiroc 2. Celle-ci est tronquée en deça, car le plateau de l'abscisse 180 ne présente pas d'intérêt pour la calibration. Ce qui est intéressant, c'est le point d'inflexion de la courbe qui indique précisément le seuil à partir duquel on peut commencer à voir du signal. On voit que ce point n'est pas tout à fait à la même valeur d'un canal à l'autre. Aussi lors de la calibration, il faudra trouver le meilleur compromis afin d'avoir le plus de canaux actifs tout en ne coupant pas trop de signal (en mettant le seuil trop haut) sur les meilleurs canaux. Les canaux les plus bruyants seront coupés. C'est pourquoi le calcul de cette courbe est un élément crucial de la calibration du détecteur.

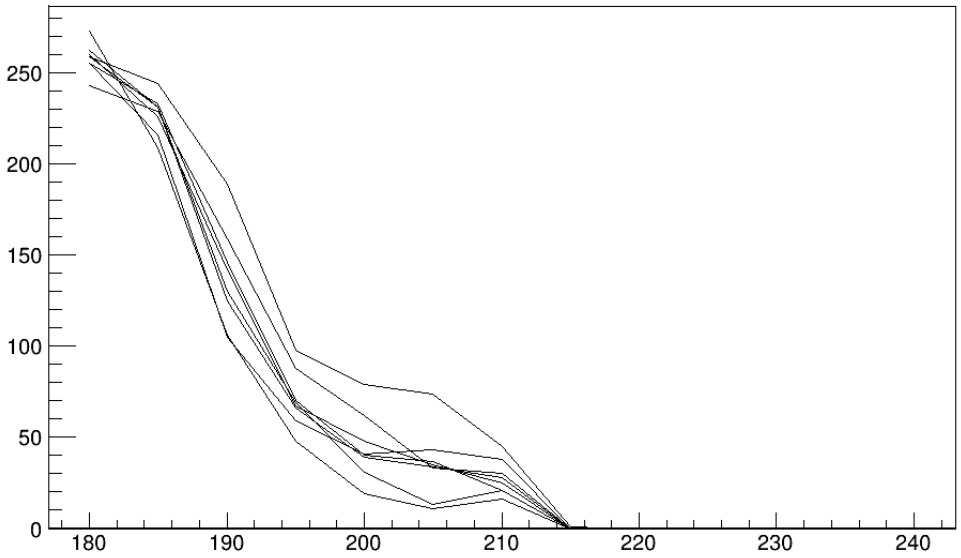


FIGURE 5.21 – Courbe en S de 8 canaux d'un *ASIC* Skiroc 2

Pycaldaq propose une fonction qui permet de produire une courbe en S en une seule ligne de code.

```
timed_scurves(acq_name,run,trig_max,trig_min,step,acq_time,roctype)
```

5.4.8 Analyse semi-*online*

Une autre utilisation intéressante de Pycaldaq est l'analyse semi-*online*. Le principe est d'alterner des commandes de configuration et d'acquisition d'une part, et des analyses offlines d'autre part. Un exemple typique de cette utilisation est le script find-noisy.

Ce script est utilisé pour masquer les canaux bruyants en fonction du seuil de *trigger* que l'on souhaite utiliser. Pour cela, hors faisceau, on fixe le seuil à une valeur très élevée, correspondant à une zone de la courbe en S où l'on attend aucun hit. On fait alors une acquisition. Tous les canaux qui déclenchent une lecture sont considérés comme bruyant et sont masqués (en coupant leur pré-amplificateur pour limiter le bruit électronique). Ensuite, on descend progressivement le seuil et on répète l'opération pas à pas jusqu'à obtenir le seuil souhaité. L'intérêt de procéder par étape est de raffiner progressivement en diminuant les niveaux de bruits.

Pour pouvoir faire cette opération, il est nécessaire de procéder à une analyse des données après chaque acquisition. Pour cela, nous utilisons deux outils. Tout d'abord, un script, nommé

raw2root permettant de transformer les données brutes en arbre root. Ensuite, un programme root procède à l'analyse pour découvrir les canaux bruyants. Il va alors générer un fichier texte, contenant des commandes reconfigure. Typiquement, on y retrouvera des commandes de masquage de type

```
reconfigure("skiroc_1_1_2_1_15","disable_preamp_chans_skiroc","42")
```

Le script Pycaldaq va alors récupérer ce fichier et appliquer la reconfiguration correspondante. Il passe ensuite à l'acquisition suivante et ainsi de suite jusqu'au seuil désiré.

5.4.9 Décodeur *online*

Le *framework* pyrame inclut un décodeur *online* pour l'*ASIC* Skiroc 2. Ce plugin est lancé automatiquement durant la phase d'initialisation de Calicoes et analyse en temps-réel toutes les données qui sont collectées sur les *slabs*. C'est un composant essentiel pour le monitoring temps-réel. Il alimente en particulier le système de statistiques qui, relayé par la GUI, informe de l'avancement de la prises de données courante.

Ce décodeur est un plugin qui s'adapte sur le module de conversion de Pyrame (converter). Il extrait les données du format brut et les publie sur un *dispatcher* de données, en les enrichissant de quelques reconstructions basiques (position x,y,z, reconstruction du temps). Son flux de sortie est constitué d'évènements de lecture regroupés par spill (un train de collision) marqués par un tag temporel (le bcid).

Un tag est calculé qui permet d'évaluer la qualité des données. En effet, le système présente des petits défauts qui dégradent la qualité de la mesure. En particulier, ils provoquent parfois des évènements pleins, c'est-à-dire des évènements où tout les canaux de l'*ASIC* déclenchent en même temps. Cela se produit lorsque plusieurs canaux déclenchent en même temps. Un autre phénomène de re-triggering apparait également parfois, provoquant des séquences de *trigger* très rapprochés (de 1 à 4 bcids), non justifiés par la physique. Enfin, il arrive que l'*ASIC* soit saturé (overflow) ou au contraire sous le seuil de lecture (underflow). Ces défauts sont détectés dans les données par un simple comptage et le tag est un champ de bits avec le codage suivant :

- bit 0 : re-triggering
- bit 1 : évènement plein
- bit 2 : underflow
- bit 3 : saturation

Enfin, des contrôles d'intégrité des données sont effectués dans le décodeur et des bits vont être levés dans un champ d'erreur en fonction de la nature de la corruption (perte de paquet, mauvaise taille de paquet, corruption de données...).

Chaque décodeur (il y en a un par *DIF*), va diffuser ses évènements dans son *dispatcher* de données, les mettant ainsi à la disposition des moniteurs *online*.

5.4.10 Chaine de traitement des données

Sur la base de ce décodage, une véritable chaine de traitement des données peut se mettre en place, formée de modules spécifiques qui combinent une ou plusieurs boucles d'évènement et un

dispatcher de données. La figure 5.22 montre une vue synthétique de cette chaîne dont nous allons détailler chaque composants dans les sections suivantes.

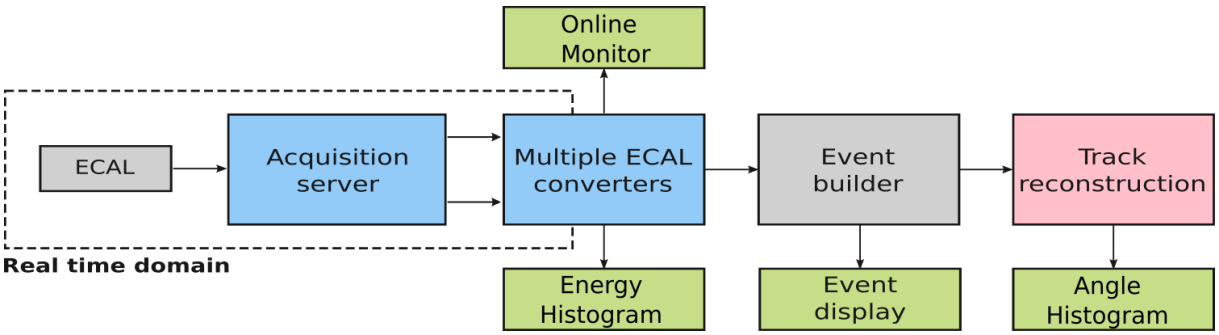


FIGURE 5.22 – Chaine de données Calicoes

Ces travaux ont donné lieu à une présentation à la conférence “Calorimetry for High Energy Frontier (CHEF) 2017” et a été publié dans JINST [35].

5.4.11 Moniteurs *online*

Comme il a été décrit précédemment, les décodeurs *online* diffusent leur informations au moyen d'un *dispatcher* de données. Ainsi, tout consommateur peut s'y connecter et traiter l'information en faisant une nouvelle analyse ou en affichant un graphique.

Les moniteurs *online* rentrent dans cette dernière catégorie, ils vont instancier une boucle d'évènements qui va leur permettre de remplir des graphiques au fur et à mesure de la production des données.

Un premier exemple basique est l'histogramme des dépots d'énergie. C'est un programme en ROOT qui instancie un histogramme simple. A chaque fois que la boucle d'évènement signale un nouvel évènement, il extrait l'énergie et l'insère dans l'histogramme. La figure 5.24 présente un tel histogramme. On reconnait la distribution de Landau, typique des dépots d'énergie dans un détecteur silicium.

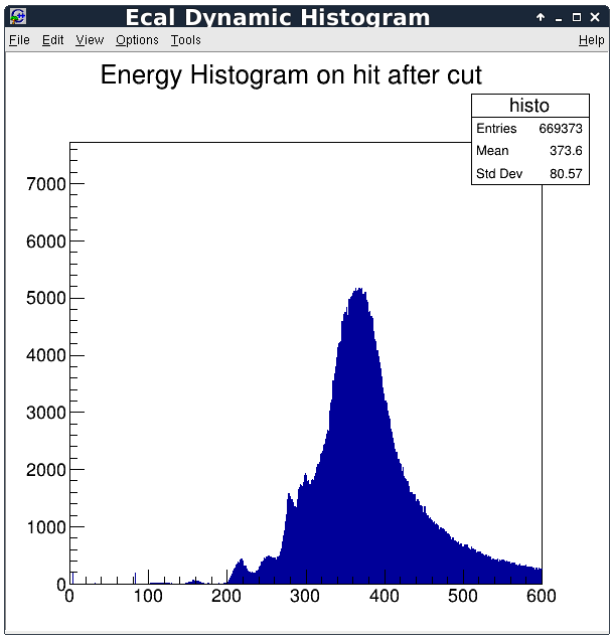


FIGURE 5.24 – Histogramme d'énergie d'un slab

Sur le même schéma, nous avons développé un moniteur *online* qui permet de voir en temps-réel, le compte des hits en histogramme, en carte ainsi qu'une carte des énergies mesurées en cumul. La figure 5.25, présente une vue de ce moniteur. On y voit le point du faisceau (beam-spot), aussi bien en hit qu'en énergie déposée.

Un autre version, dite experte, permet de monitorer les dysfonctionnements du détecteur, en affichant des histogrammes et des cartes de tous les types d'erreur ou de qualité (tag). Elle n'est pas représentée ici car elle est essentiellement similaire hormis le nombre de plots.

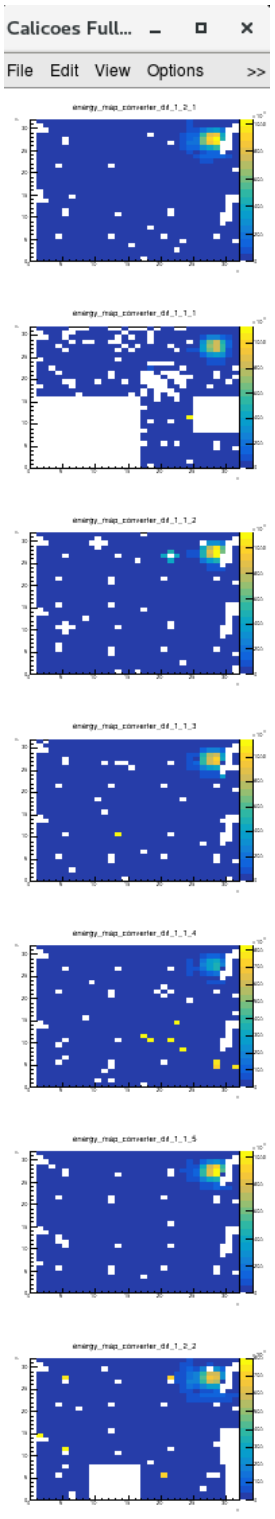


FIGURE 5.23 – Moniteur de flux

Enfin, le moniteur de flux permet de visualiser en une seule fenêtre, les cumuls des énergies sur tous les *slabs* en même temps. Cette version est très pratique pour surveiller le beam-spot et avoir une vue d'ensemble du système pendant une acquisition. Elle est représentée sur la figure 5.23. Chaque slab est représenté l'un au dessus de l'autre. On y voit le

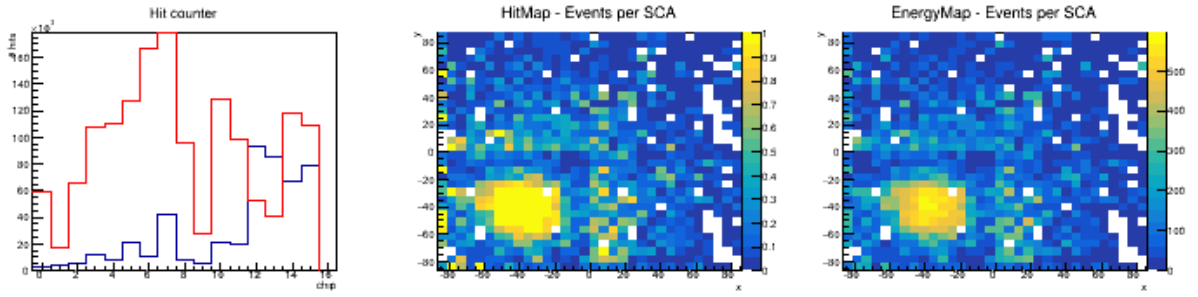


FIGURE 5.25 – Moniteur simple

beam-spot en haut à droite de chaque carte.

Le problème des moniteurs qui représentent des cumuls ou des moyennes est qu'il sont peu réactifs à des changements de configuration. Par exemple, si le faisceau s'arrête, on ne s'en rendra pas compte. Aussi la plupart des moniteurs intègrent-ils une fonction de reset régulier sur la base de quelques minutes. Ainsi, le plot reste toujours vivant et correspond aux dernières minutes de run.

5.4.12 Construction d'évènements

Comme on l'a vu précédemment, chaque décodeur fournit les données provenant de sa *DIF* ainsi que des informations de qualité et de reconstruction basique. Ces informations sont limitées géographiquement au slab correspondant à la *DIF*. Au niveau du temps, elles sont associées par groupes correspondant aux spills.

Afin de reconstruire ces trajectoires, une première opération nécessaire est de grouper les évènements en temps. Pyrame est doté, pour cela, d'un constructeur d'évènement, qui est formellement un multiplexeur temporel qui permet d'agréger de façon générique n'importe quel flux de données diffusé par des décodeurs *online*.

Le module se connecte sur chacun des décodeurs et collecte séquentiellement les données avant de les stocker dans une mémoire tampon. Il va ensuite rapprocher les évènements qui se sont produits dans une fenêtre de temps compatible avec une tolérance réglable. Il est également capable d'appliquer des coupures paramétrées sur les données. Ces nouveaux blocs de données sont ensuite publiée sur son propre *dispatcher* de données.

Dans Calicoes, les groupes ainsi formés sont constitués d'évènements des différentes *DIFs*, groupés en temps, sur la base du spill et du bcid avec une tolérance correspondant à la dérive relative des horloges du système (typiquement 1 bcid).

Comme le traitement de multiplexage est une opération très lourde, il est possible qu'un seul serveur soit insuffisant pour traiter toute les données en temps-réel. Cela ne pose aucun problème puisque le *dispatcher* de données a été conçu spécifiquement pour convertir cette lenteur en *subsampling*. Mais il peut être intéressant de densifier la reconstruction *online*. Pour ce cas, le constructeur d'évènement a été écrit pour pouvoir tourner en plusieurs instances sur un cluster.

Pour cela, une synchronisation est effectuée entre les différentes instances en utilisant varmod. Un système de vote permet de répartir le travail entre les serveurs. Par contre, chaque serveur publie indépendamment le résultat de ses calculs et il est donc nécessaire d'avoir un agrégateur

spécifique si on veut récupérer tous les évènements au même endroit.

5.4.13 Visualisation en 3 dimensions

Le constructeur d'évènements diffuse les résultats de son multiplexage via un *dispatcher* de données. Aussi, il est aisé de s'y connecter pour récupérer les blocs de données et les afficher.

Calicoes fournit en standard un afficheur temps-réel écrit en ROOT qui s'occupe de cette tâche. Pour cela, il utilise une boucle d'évènements. Les fonctions de callbacks récupèrent les données, en extraient les coordonnées et les affiche dans un graphique.

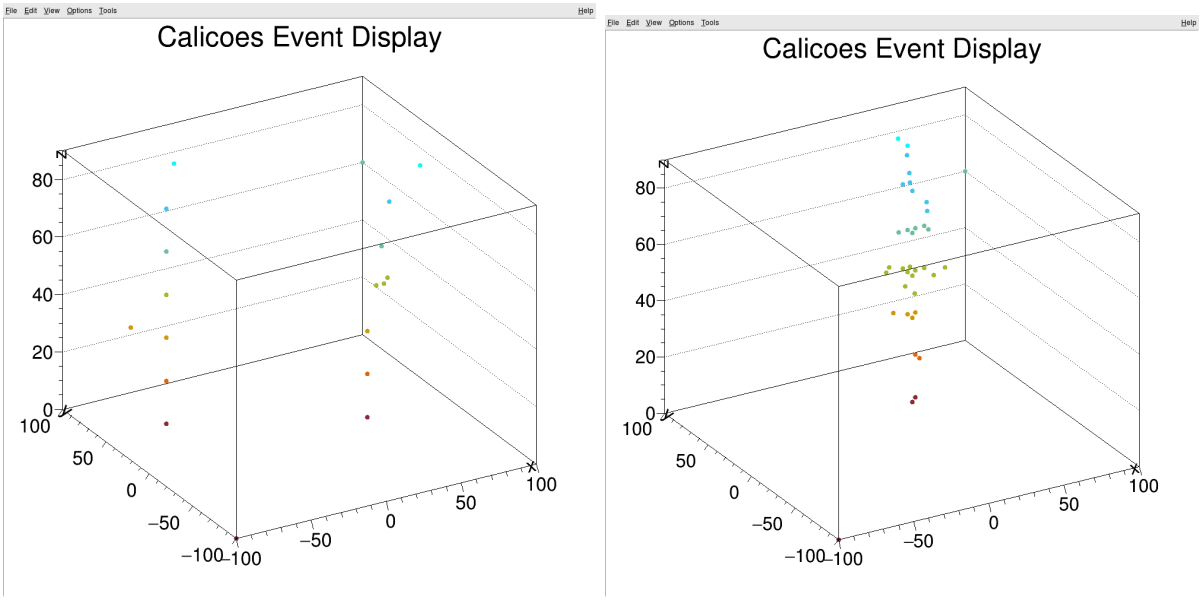


FIGURE 5.26 – Affichage en 3D de deux traces parallèles légèrement gerbées en mode *trajectographe* (c'est à dire sans ses absorbeurs en tungstène) à gauche, et une gerbe en mode calorimètre à droite

La figure 5.26 montre un exemple de cette représentation. Chaque couche porte une couleur différente. L'axe vertical correspond à l'axe du faisceau. Son échelle correspond à la distance en millimètre par rapport à la référence du détecteur (le coin inférieur gauche de son châssis). Les deux autres axes définissent un plan parallèle à chacune des surface de détection.

5.4.14 Reconstruction de traces simples

Lorsque le détecteur est en mode *trajectographe* (c'est à dire sans ses absorbeurs en tungstène), il peut être intéressant de reconstruire les trajectoires linéaires afin d'étudier la distribution des traces et de vérifier l'alignement latéral de notre détecteur.

Une fois encore, on utilisera ROOT pour effectuer les calculs nécessaires. Tout comme l'afficheur 3D, le programme se connecte sur le constructeur d'évènements au moyen d'une boucle d'évènements. A chacun des blocs, il construit une structure de type Polyline3D. Il utilise ensuite le VirtualFitter de ROOT pour effectuer une régression linéaire en 3 dimensions. Le résultat est une droite dans l'espace qu'il affiche en même temps que les points, comme le montre la figure 5.27.

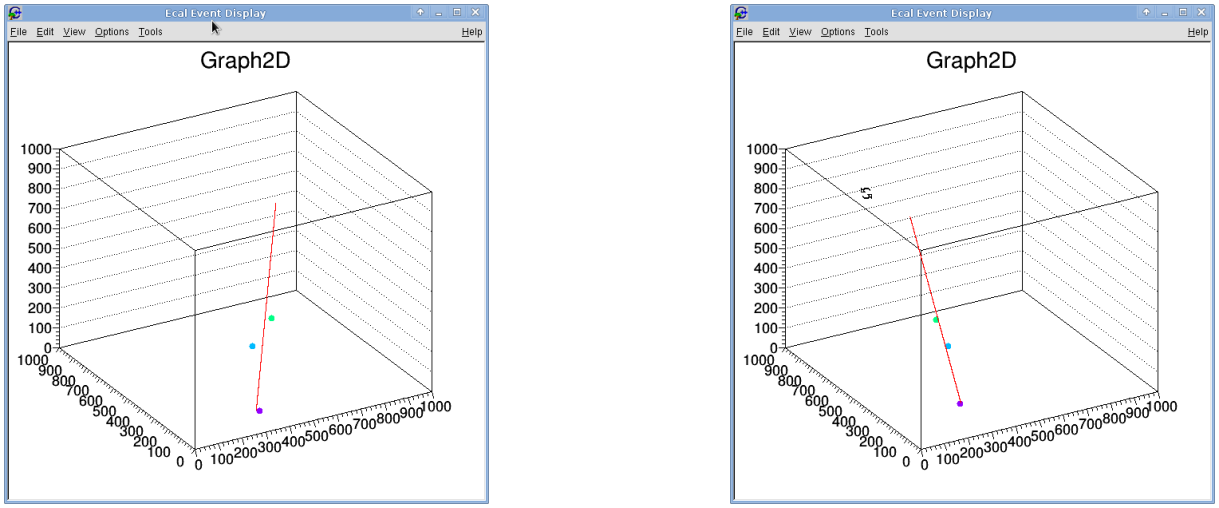


FIGURE 5.27 – Régression linéaire des hits d'un SiW-Ecal à trois couches avec Root

A partir de cette droite, on peut calculer facilement l'angle qu'elle fait avec l'axe longitudinal du détecteur. Il ne reste plus qu'à placer ces valeurs dans un histogramme comme sur la figure 5.28 et on obtient la répartition angulaire du faisceau. On voit ici que l'essentiel des trajectoires sont droites, le pic étant bien centré sur zéro. Cela signifie que notre détecteur est aligné de façon satisfaisante par rapport à l'axe du faisceau. En ce sens, c'est une aide appréciable pour vérifier que le montage est correct. On voit également que les traces peuvent avoir un angle jusqu'à 18 degrés de l'axe du faisceau, de manière symétrique mais avec une statistique totalement négligeable, cela permet de caractériser en moyenne le faisceau et éventuellement d'en tenir compte dans le *fit* du *MIP*.

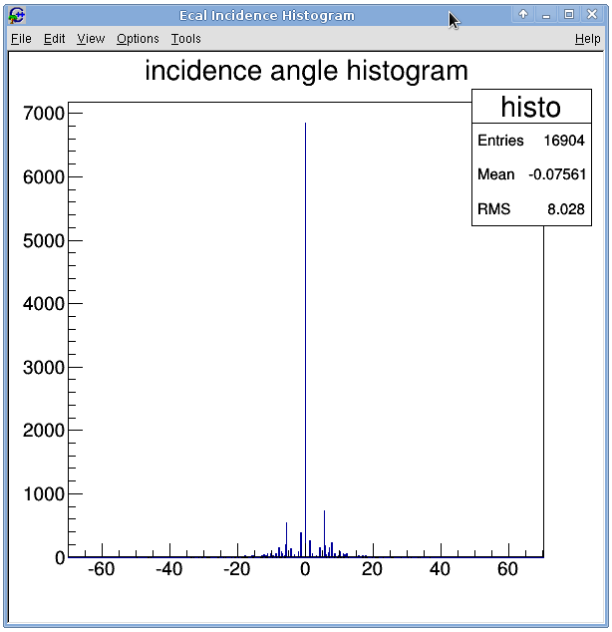


FIGURE 5.28 – Histogramme de directions des traces au DESY. Le détecteur est orthogonal au faisceau (pic en 0) et la dispersion est faible.

Il est à noter que l'algorithme utilisé ici ne fonctionne que si une seule trace est présente dans

le bloc de données. Dans le cas contraire, le *fit* donne une droite située entre les deux traces. Il n'existe pas d'algorithme trivial pour distinguer ce cas, car il arrive qu'une même particule active deux pixels voisins à la fois. Les méthodes de comptage sont donc inefficaces. Nous reviendrons abondamment sur ce sujet au chapitre 7 sur la trajectographie.

5.4.15 Analyse offline via le décodeur

On peut aussi utiliser ce décodeur pour faire une analyse offline en lui faisant rejouer un fichier précédemment enregistré. Cette démarche permet d'unifier les codes d'analyse entre l'analyse online et offline puisqu'ils utilisent tous deux la même interface d'accès aux données. Lors d'un stage de Licence au LLR, Diana Pereira-Catarino a implémenté une telle analyse. Il s'agit d'un module qui calcule les piedestaux du détecteur après chaque acquisition et qui les stocke dans une base de données. Cela permet de monitorer le comportement du détecteur sur le long terme et ainsi de pouvoir détecter des effets de déplacement de piedestaux qui traduisent en général un vieillissement du matériel. On peut ainsi utiliser ce code aussi bien pour l'analyse après le run, que pendant, permettant ainsi un monitoring des piedestaux durant l'acquisition. Toutefois, cette méthode souffre d'un défaut majeur. Comme les données sont obtenues sur une base séquentielle, on ne peux pas faire plusieurs passes. Lors d'une opération comme la soustraction de piedestal, il est nécessaire de faire une première passe pour calculer les moyennes avant de les soustraire aux données lors de la deuxième passe. C'est pourquoi son utilisation est limitée à des problématiques très simple et ne saurait remplacer un véritable framework d'analyse offline.

5.5 Le prototype à *slabs* courts

Le prototype technologique à slab courts est une réalisation destinée à démontrer l'effectivité du concept en relâchant certaines contraintes géométriques liées à l'ILD. Il permet de valider les aspects calorimétriques du détecteur (niveau de bruit, linéarité,...) sans engager les dépenses d'un slab ILD.

5.5.1 Description

Un slab court est composé d'un seul *ASU* et d'une *SMB*. L'*ASU* est à peu près similaire à ceux que l'on utilisera sur ILD. La seule différence réside dans l'épaisseur des composants qui sont un peu plus épais. La carte adaptatrice est, elle, très différente. De grande taille, elle permet d'ajouter de nombreux composants nécessaires pour les tests, qui disparaitront dans la version finale.

Le prototype à *slabs* courts est une réalisation qui a évolué depuis 2010, allant vers une intégration plus poussée et un design plus efficace. Ainsi, en 2014, l'ASU version 8 n'intégrait que 4 Skiroc 2 et un seul *wafer* pouvait être lu. Depuis l'ASU version 11, la carte intègre 16 *ASICs* Skiroc 2, soit 1024 canaux, correspondant au niveau d'intégration de l'ILD.

Comme on le voit sur la figure 5.29, les *slabs* sont rangés verticalement dans une structure de plastique et d'aluminium. La fenêtre à gauche, correspond à la zone de détection. A droite, on voit l'interconnexion des *SMB* vers les *DIFs* puis vers la baie électronique. A gauche, sur le dessus, on voit la structure qui permet d'insérer des plaques de tungstène (jusqu'à 24 X0) devant, derrière et entre les *slabs*. Sur l'image de droite, on voit la baie électronique qui contient

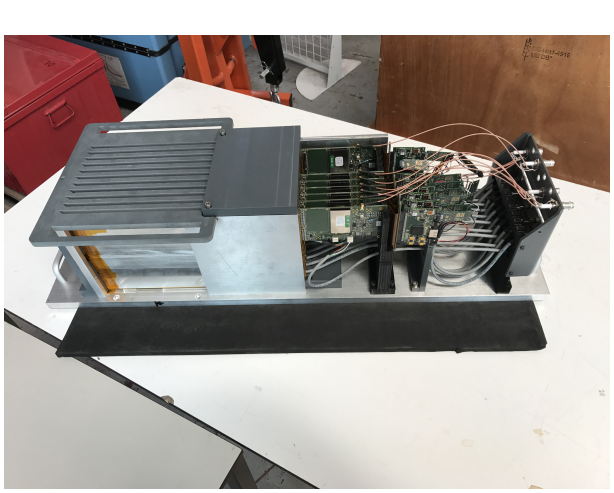


FIGURE 5.29 – Prototype à *slabs* courts. A gauche, la structure mécanique équipée de 7 slabs. Les fentes sur le dessus permettent d'insérer des absorbeurs en tungstène. A droite, la baie électronique pour la lecture et l'analyse online des données.

les alimentations basse et haute tension, le panneau d'interconnexion, la *DAQ* (*GDCC*, switchs Ethernet et PC), ainsi qu'un générateur de pulsation.

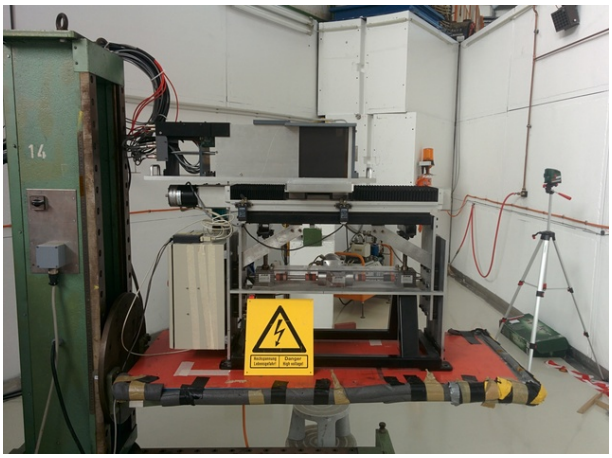


FIGURE 5.30 – Prototype *slabs* courts sur sa table de déplacement à DESY pour le *test faisceau* de 2017

Sur la figure 5.30, on voit le prototype en *test faisceau* au DESY. Il est monté sur une table XY pilotable à distance par Calicoes, elle même posée sur la table fournie par la zone de faisceau (en rouge et vert). La table permet de monter et descendre le détecteur mais aussi de le déplacer de gauche à droite, permettant ainsi un scan de toute la surface de détection.

5.5.2 *tests en faisceaux*

Le prototype à *slabs* courts a été testé à de nombreuses reprises en *test faisceau*. Depuis Mars 2017, je suis le directeur technique de ces tests. Dans cette section, je présente les résultats que nous avons obtenu au DESY en juin 2017. Le détecteur était alors composé de 7 *slabs* de version 12 et a été placé dans un flux d'électrons d'une énergie variant entre 2 et 5.8 GeV.

En amont du déplacement, de nombreux tests ont été effectués afin de caractériser finement les *slabs*. Une procédure de qualification permettant de tester les aspects électriques et physiques a été mise en place, aboutissant à la rédaction d'un passeport pour chacun des *slabs*. Lors de cette procédure, nous avons appliqué des coupures très conservatives, pour masquer toutes les voies pouvant générer du bruit. Nous avons abouti à un détecteur dont 8% des voies étaient masquées.

Le programme de physique comportait de nombreux points :

- Test d'uniformité du piedestal
- Calibration du *MIP* : scan en position en mode *trajectographe* à 3GeV
- Scan en énergie avec 3 configurations de tungstène
- Calibration à 45 degrés
- Tests en champ magnétique

Tout d'abord, nous avons mesuré le piedestal du piédestal des *slabs*. Celui-ci correspond au niveau de bruit détecté par les *ASICs*. Il nous est accessible car à chaque fois qu'un hit est détecté dans le Skiroc 2, toutes les cellules sont échantillonnées et non seulement celles qui ont détecté le hit. Ces valeurs permettent de calculer la valeur de référence (la moyenne) du bruit du détecteur. Ces piedestaux varient très légèrement d'une cellule à l'autre mais aussi dans la séquence d'acquisition.

La figure 5.31 présente la distribution spatiale de ces piédestaux sur l'ensemble d'un *slab*. Chaque petit carré correspond à un pixel du détecteur. On voit les disparités d'un *ASIC* à l'autre. En blanc, on voit également les canaux masqués. La couleur correspond à la valeur du piédestal en coups d'ADC. On voit que sa dispersion est très faible, de l'ordre de 11 coups d'ADC soit un sixième du *MIP*.

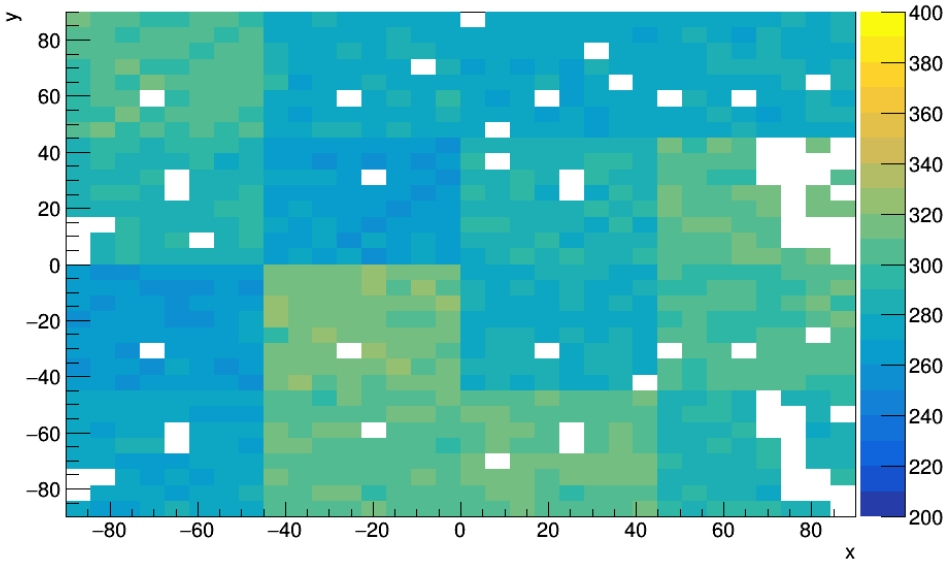


FIGURE 5.31 – Piédestal d'un slab entier. Les pixels blancs sont les canaux masqués. On voit clairement un effet ASIC (blocs 8x8)

Ensuite, nous avons réalisé la calibration du *MIP*, qui consiste à scanner, en position, l'ensemble du détecteur sous un flux d'électrons à 3GeV (le régime fournissant le plus de statistiques). Le détecteur est déplacé dans le faisceau grâce au système de script pilotant la table XY. 81 points ont ainsi été mesurés pendant une demi-heure chacun. Vu la dispersion du faisceau (environ

3cm), tous les pixels ont ainsi reçu suffisamment de particules pour calibrer correctement le *MIP*, c'est à dire le dépot d'énergie correspondant au passage d'une seule particule.

Pour effectuer cette calibration, on construit l'histogramme des dépots d'énergie. On extrait toutes les valeurs d'énergie qui sont associées à un hit, on leur soustrait le piédestal et on le stocke dans l'histogramme. Celui-ci est ensuite *fitté* au moyen d'une distribution de Landau convolué avec une Gaussienne. La distribution de Landau est une sorte de Gaussienne munie d'une longue queue, provoquée par la production de particules secondaires. Pour tenir compte du bruit de mesure, on la convolue avec une Gaussienne. Cette convolution est appelée Langauss. On obtient donc quatres paramètres de fit : la valeur du pic de la landau (MPV pour "Most Probable Value"), sa dispersion (Sigma), la moyenne et la dispersion de la Gaussienne convoluée. Le MPV est la valeur du *MIP*. Un tel fit est présenté à gauche sur la figure 5.32. Si plusieurs particules sont mesurées en même temps, une seconde (voire une troisième) distribution de Landau peut être nécessaire pour fitter la queue de distribution avec plus de précision.

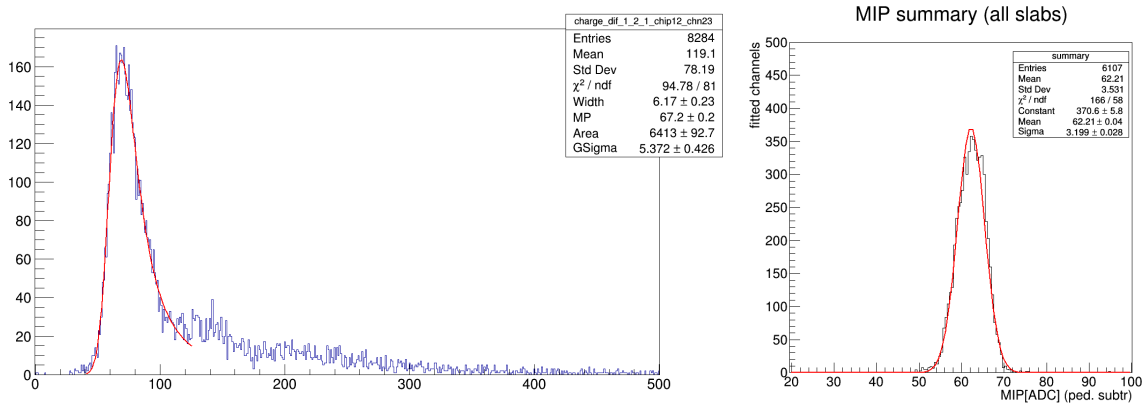


FIGURE 5.32 – Calibration du *MIP* par une distribution de Landau convoluée avec une Gaussienne (à gauche). A droite, l'histogramme de ces calibrations pour toutes les cellules de tous les slabs. (Credit A. Irles, LAL)

Une fois la valeur du *MIP* obtenue sur l'ensemble des cellules, on peut les représenter sur un histogramme, comme sur la droite de la figure 5.32. L'histogramme est clairement Gaussien et un fit permet d'obtenir la valeur moyenne du MIP pour l'ensemble du détecteur. Cette valeur est de 62.2 coups d'ADC avec une dispersion de 3.2 ADC (5.1%). Cela correspond à un rapport signal/bruit de 20.3 dans la branche de lecture à haut-gain, ce qui est excellent et démontre s'il en était besoin que les diodes PIN en silicium sont parfaitement adaptées pour ce genre de mesure.

Une autre façon de *fitter* le *MIP* est d'utiliser des traces, en sélectionnant celles où chaque couche non masquée a reçu au moins un dépôt. Après un filtrage basique pour enlever les artefacts, on peut *fitter* le *MIP* au moyen des trois distributions de Landau, comme représenté sur la figure 5.33. Cette technique permet d'éliminer toutes les imperfections de mesure en ne gardant que les meilleures, et d'obtenir ainsi une bien meilleure précision, bien sûr au prix d'une baisse drastique des effectifs statistiques.

Nous avons ensuite testé le détecteur en mode calorimètre avec trois configurations différentes de tungstène. En effet, notre prototype dispose de fentes sur le dessus qui permettent d'enfoncer des plaques d'épaisseurs différentes entre les couches. Les configurations comportaient un total de 12.32, 15.68 et 19.04 X0. La figure 5.34 montre les profils de gerbes correspondants, obtenus par nos mesures.

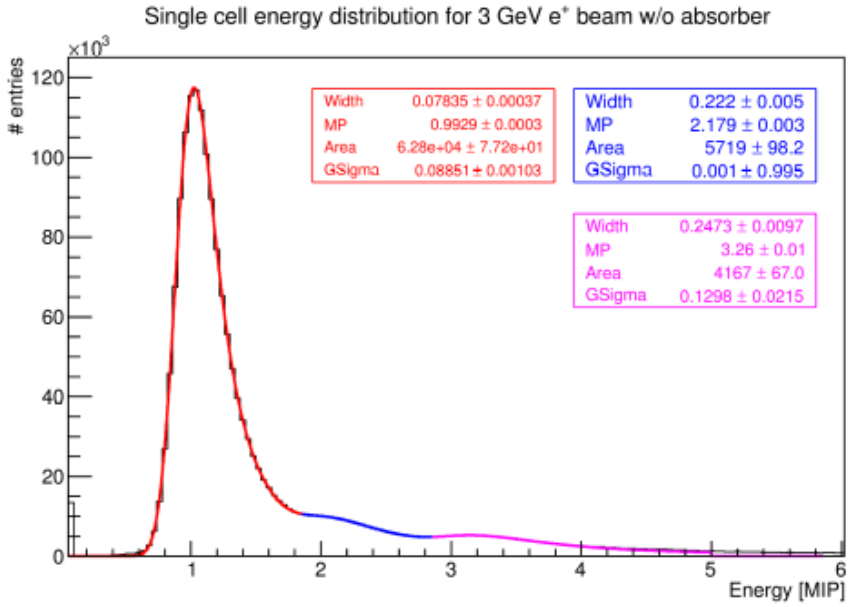


FIGURE 5.33 – Calibration du *MIP* avec des traces. L'utilisation de traces consolidées sur toutes les couches permet d'éliminer tous les artefacts de lecture et d'obtenir un fit de très bonne qualité. (Credit A. Irles, LAL)

Pour ces différents profils, nous avons fait un scan en énergie entre 1 et 5.8GeV, à position fixe. Les énergies centrales autour de 3 GeV donnent beaucoup de statistiques et une heure suffit pour accumuler suffisamment des hits. En revanche, sur les énergies les plus hautes et les plus basses, il faut tourner toute la nuit pour atteindre le même objectif. Le scan complet en énergie a pris trois jours complets.

Après ce scan, nous avons procédé à une calibration du *MIP* (en mode *trajectographe*) en tournant le détecteur de 45 degrés par rapport au faisceau. En effet, le nombre de paires électrons-trous créées dans le silicium est directement proportionnel à l'épaisseur du matériau traversé par les électrons incidents. Aussi à 45 degrés, la distance parcourue est augmentée de $\sqrt{2}$. Le dépôt d'énergie étant plus important, nous pouvons nous attendre à un décalage du *MIP* dans les même proportions.

Les contraintes mécaniques étaient trop fortes pour obtenir une placement adéquat. La table XY était elle-même posée sur une autre table à déplacement, assez étroite. Le tout étant lourd et fragile, nous n'avons pris aucun risque. Nous avons ensuite mesuré l'angle obtenu et trouvé 43.6 ° .

La figure 5.35 montre la trace du faisceau dans ces conditions. On voit que le beam-spot se déplace bien d'un slab à l'autre. Ce déplacement est cohérent avec l'angle et il est du à l'espacement entre les couches. Le *fit* du *MIP* qui se situe cette fois vers 86 coups d'ADC. On sait que la valeur du *MIP* à un angle α, que nous notons M_α, est reliée à la valeur du *MIP* à angle nul noté M_0 par la formule

$$M_\alpha = \frac{M_0}{\cos\alpha} \tag{5.1}$$

Donc, pour 43.6 ° , on attend une valeur de $62.2/\cos 43.6 = 85.9$ soit un très bon accord avec la mesure expérimentale.

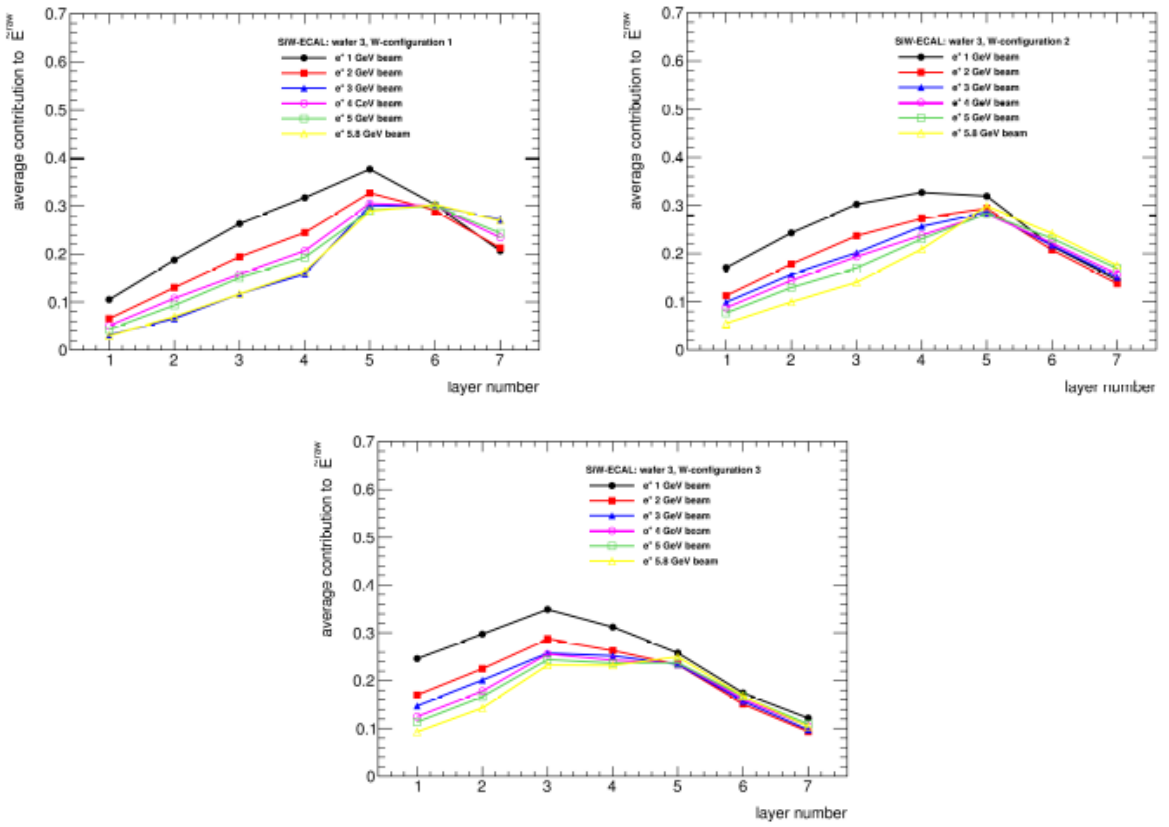


FIGURE 5.34 – Profils de gerbes électromagnétiques pour des configurations différentes de la répartition des absorbeurs en Tungstène aux 6 énergies testées (Credit A. Irles, LAL)

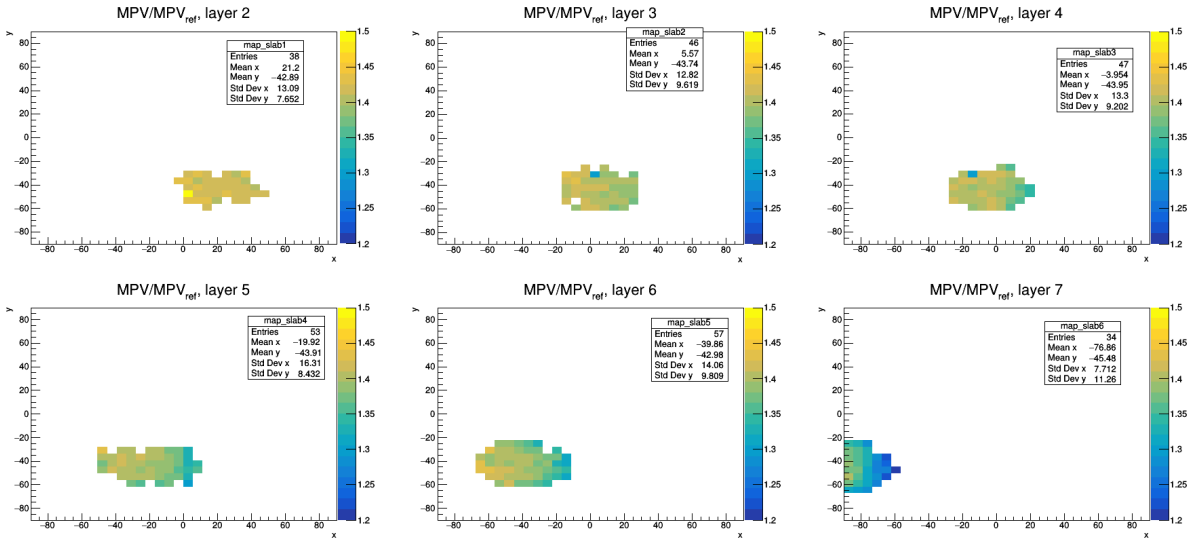


FIGURE 5.35 – Trace du faisceau sur les 6 slabs pour le détecteur orienté à 43.6 degrés par rapport au faisceau (le 7ième est hors faisceau). Le beam spot se décale vers la gauche dans le sens de léloignement des slabs par l'effet de l'angle. (Credit A. Irles, LAL)

Enfin, nous avons pris des mesures avec un de nos *slabs* placé dans un champ magnétique afin

de tester sa résistance mécanique. En effet, comme l'électronique est pulsée, les changements de tension induisent des forces d'arrachement sur le composants qui peuvent les endommager. Nous voulions également voir si le fonctionnement de l'électronique était altéré par le champ magnétique. La zone 24 au DESY présente la particularité de posséder deux sous-zones, l'une en faisceau classique et une autre, placée après la première, où un aimant supra-conducteur est disponible. Il fournit un champs magnétique de 1 tesla dans un très grand volume. Il a été développé pour la mise au point de la *TPC*, futur *trajectographe* de l'ILD.

Nous avons monté l'un de nos *slabs* dans la cavité de cet aimant, attaché au moyen d'un structure en plastique. L'électronique de mesure a été positionnée sur le côté, le plus loin possible de l'aimant pour éviter toute perturbation. Tout ce qui pouvait être déporté encore davantage a été ramené jusqu'à la salle de *shift* grâce à des câbles noyés dans les murs. Les alimentations électriques par exemple ont été déportées par des câbles de plus de vingt mètres, nous obligeant à hausser les voltages pour compenser les pertes en ligne. Le PC de *DAQ* a lui aussi été déporté via des câbles RJ45.

La figure 5.36 montre les trois mesures effectuées. Une première de référence sans champ magnétique, puis avec le champ à 0.5 T et la dernière avec le champ à 1 T. Le slab a montré une parfaite stabilité mécanique, et a fonctionné de manière totalement normale pendant tout le test. On voit deux contributions sur le graphique, l'une inchangée correspondant aux photons et l'autre, correspondant aux électrons, défléchis par l'aimant.

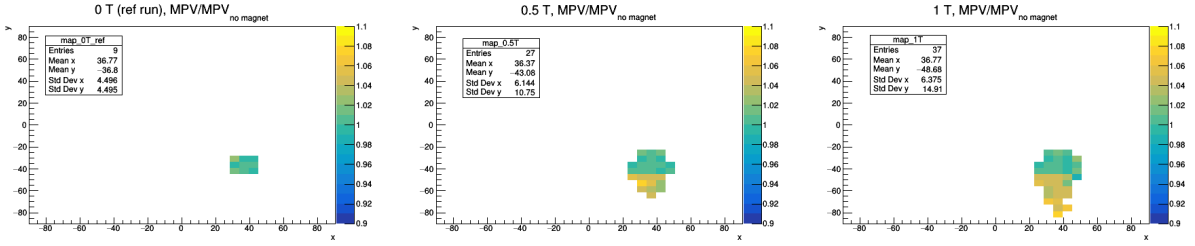


FIGURE 5.36 – Faisceau en champ magnétique. La trace comporte deux composantes, la première, inchangée par le champ, qui correspond aux photons et la deuxième, défléchie, qui correspond aux électrons. (Credit A. Irles,LAL)

L'ensemble de ces travaux ont été présentés par Adrian Irles à la conférence CHEF17. Une publication du groupe est en cours de finalisation.

5.6 Conception d'une nouvelle génération de slabs courts

Après ces tests en faisceau, nous avons initié une collaboration avec l'université de Kyushu dans le but d'apporter des modifications au design pour essayer diverses options en vue de la production du détecteur final.

On peut regrouper les objectifs de ce projet en trois catégories

- Limiter au maximum les causes de dégradation des données, en particulier les canaux masqués et les évènements auto-déclenchés.
- Préparer la production des futurs slabs longs en adaptant l'électronique au problématiques de pistes longues.
- Adapter le détecteur à la géométrie préconisée pour l'ILD.

Ces travaux ont été menés pendant la saison 2017/2018 et ont donnés lieu à 2 tests en faisceau au DESY et au CERN qui ont démontré le bon fonctionnement du nouveau prototype.

5.6.1 Limitation des mauvaises données

Les slabs courts du précédent prototype ont un rapport signal sur bruit de l'ordre de 20 dans la branche de lecture à haut-gain et de l'ordre de 12 dans la branche trigger, ce qui est suffisant pour un détecteur comme l'ILD. En revanche, une quantité non négligeable des données sont perdues à cause des canaux masqués. Ceux-ci sont détectés lors de la phase de calibration et sont ensuire masqués électroniquement.

La calibration consiste à diminuer progressivement le seuil de déclenchement des ASICs et à masquer les canaux dont l'activité est nettement plus élevée que celle des autres. C'est une opération délicate qui nécessite de trouver un bon compromis entre un taux de masquage raisonnable et un taux de déclenchement qui limite le bruit. Le taux typique de masquage des slabs du prototype à slab court est typiquement de 8%. Comme on le voit sur la figure 5.31, ce masquage présente de nombreuses systématiques. Ainsi, le canal 37 est toujours masqué. Des formes apparaissent également plusieurs fois dans le schéma de masquage, indiquant que les taux trop élevés peuvent être induits par le design lui-même.

Dans l'optique de réduire le nombre de canaux masqués, un re-design systématique de toutes les lignes de données a été entrepris. Certaines maladresses de design ont été corrigées, comme des angles trop obtus sur les pistes.

Un autre défaut dans l'acquisition des données est induite par les auto-déclenchements. L'*ASIC* Skiroc semble générer des triggers spontanés dans certaines conditions. L'un d'entre eux, appelé "plane event" est l'apparition d'un déclenchement sur l'ensemble des canaux lorsque le nombre de déclenchements simultanés dépasse un seuil. L'autre est un re-déclenchement qui apparait temporellement très proche d'un déclenchement antérieur. Ces deux phénomènes peuvent être facilement filtrés dans les données. Le problème vient de la taille limitée des mémoires de l'*ASIC*. Si les circonstances sont défavorables, les mémoires peuvent être remplies presque entièrement par ces artefacts, empêchant d'obtenir les évènements de physique.

Afin de tenter de règler ces problèmes, l'unite Omega a conçu une nouvelle version incrémentale de son *ASIC*, le Skiroc 2a, compatible pin à pin, avec des modifications de protection du pré-amplificateur. Le nouvel *ASU* a donc été conçu pour pouvoir accueillir cet *ASIC*.

D'autre part, nous avons entrepris de séparer les alimentations analogique du pré-amplificateur, la partie la plus sensible de l'*ASIC*, du reste des parties analogiques de l'*ASIC*. Pour cela, il a fallu modifier la *SMB* afin de l'équiper d'un régulateur intégré de tension supplémentaire. Des plans de masse ont été insérés dans le design de la *SMB* pour isoler le plus possible ces parties sensibles.

Enfin, nous avons apporté des modifications au firmware de la *DIF* pour permettre de couper l'horloge de transfert pendant l'acquisition pour limiter au maximum les signaux qui pourraient venir perturber le déclenchement.

5.6.2 Adaptation pour le futur slab long

Les slabs courts qui avaient composé le prototype jusque là n'ont pas intégré l'ensemble des contraintes découlant de la longueur du détecteur final. En particulier, le chemin des données au sein d'une chaine d'*ASUs* n'a pas été optimisé. Il oblige le dernier *ASIC* de cette chaine a envoyer les données sur une ligne de 2 mètres de long. Son buffer de sortie n'est pas adapté pour cette transmission.

Nous avons donc changé ce cheminement de données pour lui donner la forme d'un U, permettant d'avoir le dernier *ASIC* de la chaine au plus près de la carte front-end. De même, quelques modifications ont du être apportées au design de la carte *SMB* afin d'intégrer des composants d'adapation d'impédance des lignes.

Un autre problème récurrent du prototype à slabs courts est qu'il est impossible de lier automatiquement les données à un identifiant unique pour chaque slab. Cette possibilité avait été envisagée lors de la conception de la *SMB* version 4 en intégrant un composant Dallas2432, fournissant la température et un identifiant unique, toutefois la fonctionnalité n'avait jamais été implémentée dans la *DIF*. Un bloc implémentant le protocole OneWire a été écrit et intégré dans la *DIF* pour permettre d'intégrer cet identification dans le processus d'acquisition des données.

5.6.3 Adaptation géométrique

Le détecteur ILD impose des contraintes géométriques fortes sur le design des slabs longs. Plusieurs points restaient problématiques dans le design précédents.

Ainsi la *SMB* et la *DIF* pour l'ILD doivent pouvoir tenir dans un espace de 70 par 40 millimètres contre plusieurs dizaines de centimètres pour la *SMB* V4. Les grosses capacités disposées sur la *SMB* et servant à absorber les variations de courant liées au power-pulsing, ne pouvaient être conservées. Elles ont été remplacées par un tout nouveau modèle ultra-plat qui a pu être positionné directement sur les *ASUs*. Ainsi, chaque *ASU* comporte ses propres capacités, limitant la longueur du circuit correspondant et donc l'intensité électrique nécessaire au niveau des connecteurs.

5.6.4 Nouveau prototype

L'ensemble de ces adaptations ont été recensées par mes soins sous la forme d'un document synthétique listant 17 actions modificatives sur l'ancien prototype et qui a servi de fil conducteur pour la définition du nouveau prototype. Les travaux de définition et de prototypage ont été effectués sous ma coordination technique, en collaboration entre l'université de Kyushu en particulier par Taikan Suehara et son étudiant Yu Miura, et le LLR, en particulier Jérôme Nanni et Rémi Guillaumat. Ils ont donné lieu à un poster dans la conférence IEEE NSS-MIC à Sydney et donneront lieu à une publication prochainement.

Les travaux de refonte du design ont débuté par la définition d'un nouvel *ASU*. Celui-ci intègre toutes les modifications évoquées dans les paragraphes précédents. C'est un *PCB* plus complexe que le précédent de par son nombre de couches accru.

Pour la *SMB*, il a été choisi de faire un design intermédiaire avant le design final pour pouvoir tester la faisabilité des différentes options. La figure 5.37 montre la géométrie de la nouvelle

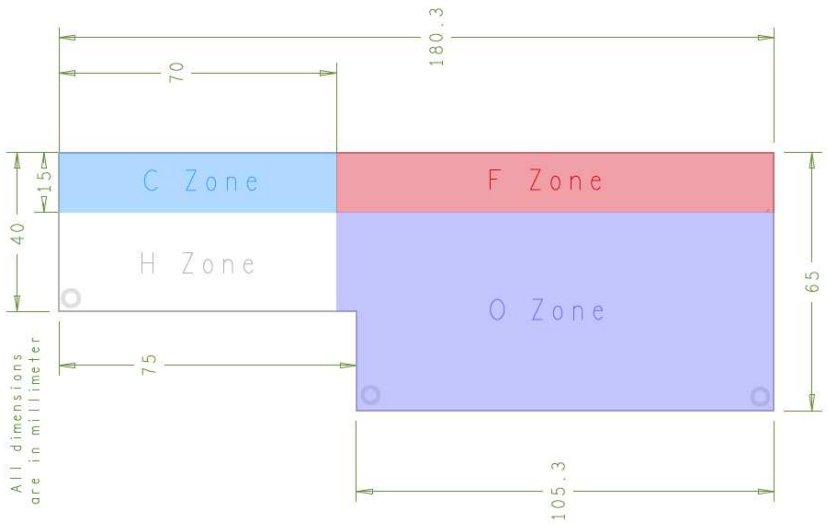


FIGURE 5.37 – Géométrie de la *SMB* version 5. Les zones F et O sont les zones de debuggage qui seront supprimées dans le design final, ne laissant que les zones C et H, compatibles avec la géométrie de l'ILD. (Credit R. Guillaumat, LLR)

carte *SMB*. Celle-ci comporte une partie (H+C) de 40x70 mm^2 comportant tous les composants essentiels de la carte. Une autre zone (0+F) comporte une empreinte pour des capacités, au cas où les nouvelles capacités ne fonctionneraient pas comme attendu. Sur cette zone, on trouve aussi la connectique de débuggage.

La carte a été ainsi conçue pour permettre d'utiliser la partie de debuggage pendant les phases de conception, mais elle peut être supprimée lors de la phase de production.

Les fonctionnalités de la *DIF* ne sont pas encore intégrée dans cette carte qui comporte un connecteur compatible avec les anciennes *DIFs*. Toutefois, tous les composants sont soudés sur une face de la carte, ainsi nous pourrons intégrer les composants de la *DIF* sur l'autre, lors du prochain design, permettant ainsi d'avoir un ensemble *SMB*/*DIF* totalement compatible avec la géométrie ILD.

Les nouvelles cartes ont été utilisées pour assembler 5 nouveaux slabs dont on voit un exemplaire sur la figure 5.38. Cet assemblage a été réalisé à l'université de Kyushu. Il a été choisi d'équiper ces slabs avec des wafers d'une épaisseur de 650 μm (4 unités) et 320 μm (1 unité), afin de vérifier d'éventuelles dépendances.

5.6.5 Test en faisceaux

Ces nouveaux slabs ont été testés lors de deux tests en faisceau. L'un en juillet 2018 au DESY avec des électrons de 3 GeV, l'autre au CERN en septembre 2018 avec de muons, des pions et des électrons dans une gamme d'énergie allant de 30 à 150 GeV.

Les analyses sont en cours mais certains éléments sont déjà avérés. D'une part, le nombre de canaux masqués a chuté drastiquement. Seul le canal 37 reste bruyant sur l'ensemble des *ASIC*s ce qui fait penser à une systématique dans le packaging BGA. Le slab en lui-même est nettement moins bruyant que l'ancien avec un rapport signal sur bruit de l'ordre de 40 dans la branche de lecture à haut-gain.

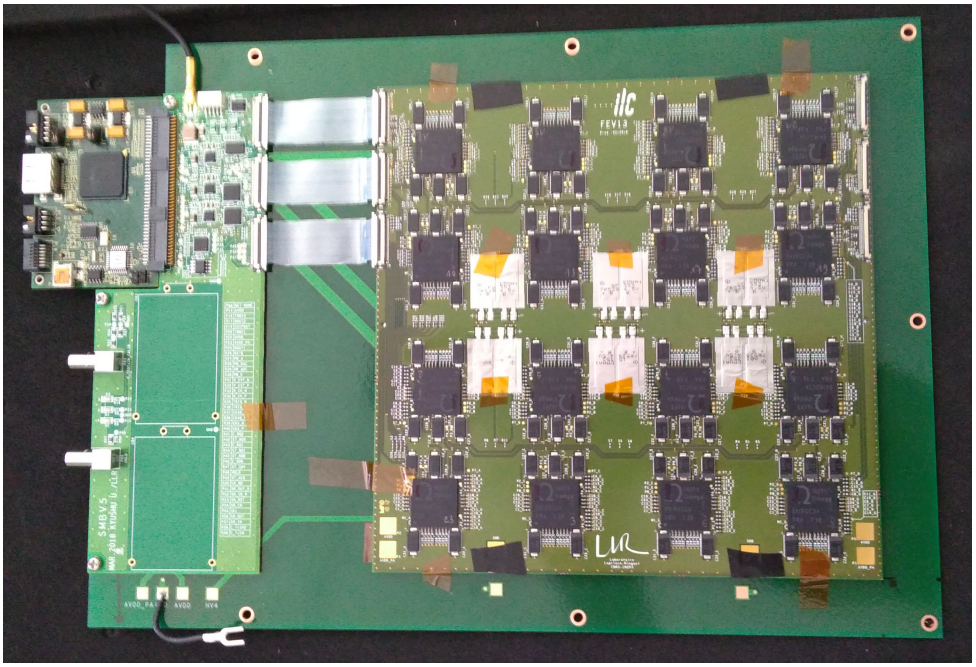


FIGURE 5.38 – Le nouveau slab équipé des nouvelles cartes ASU version 13 et SMB version 5.

5.7 Le prototype slab long

Les prototypes à slabs courts ont démontré que le concept du SiW-Ecal fonctionne parfaitement et que la sensibilité d'un tel détecteur est largement suffisante pour un projet tel que l'ILD. Toutefois, ils ne permettent pas de démontrer la faisabilité technique de slabs complets, c'est à dire composés d'*ASUs* enchainés sur une grande longueur.

Dans le projet actuel du SiW-Ecal pour l'ILD, le barrel est constitué de slabs de 8 *ASUs* de long et les endcaps sont constitués de slabs de taille variable pouvant aller jusqu'à 12 *ASUs*. Cette longueur est susceptible de faire apparaitre des difficultés pour la transmissions des signaux et l'alimentation (basse et haute tension). C'est pourquoi, nous avons conçu et développé un prototype slab long afin de découvrir et résoudre ces problèmes. Dans un deuxième temps, nous pourrons construire un véritable slab long répondant aux exigences géométriques du futur détecteur en nous appuyant sur cette expérience.

Les nouvelles capacités installées sur les *ASUs* fonctionnent parfaitement et nous pouvons donc envisager sereinement de placer toute l'électronique front-end dans les $40x70\ mm^2$ que nous impose le design de l'ILD.

Les nouveaux slabs corrigent donc de façon satisfaisante la plupart des défaut des version précédentes et ce nouveau design pourra servir de base à une future version désignée spécifiquement pour l'ILD lorsque le projet deviendra une collaboration.

5.7.1 Design

Bien qu'il soit basé sur une extension du prototype à slab court, le slab long présente de nombreuses difficultés électroniques. C'est pourquoi, il a fallu concevoir des stratégies pour adapter son design aux nouvelles conditions de fonctionnement. Ses dimensions nécessitaient également

une refonte complète de son support mécanique. Tous ces travaux ont été effectués sous ma coordination technique avec les équipes du LLR, en particulier Jérôme Nanni pour l'électronique et Evelyne Edy pour la mécanique.

Le prototype slab long est basé sur un châssis pouvant accueillir un assemblage d'*ASUs* chainés entre eux. Le châssis, représenté sur la figure 5.39 est une structure en PCV dans laquelle sont percées 12 alvéoles permettant de recevoir un *ASU*. Le PVC est fixé dans un cadre en aluminium qui assure sa rigidité et permet de le positionner dans le contexte des expériences. Afin de garantir l'obscurité nécessaire aux mesures, des capots noirs peuvent être fixés sur le cadre.

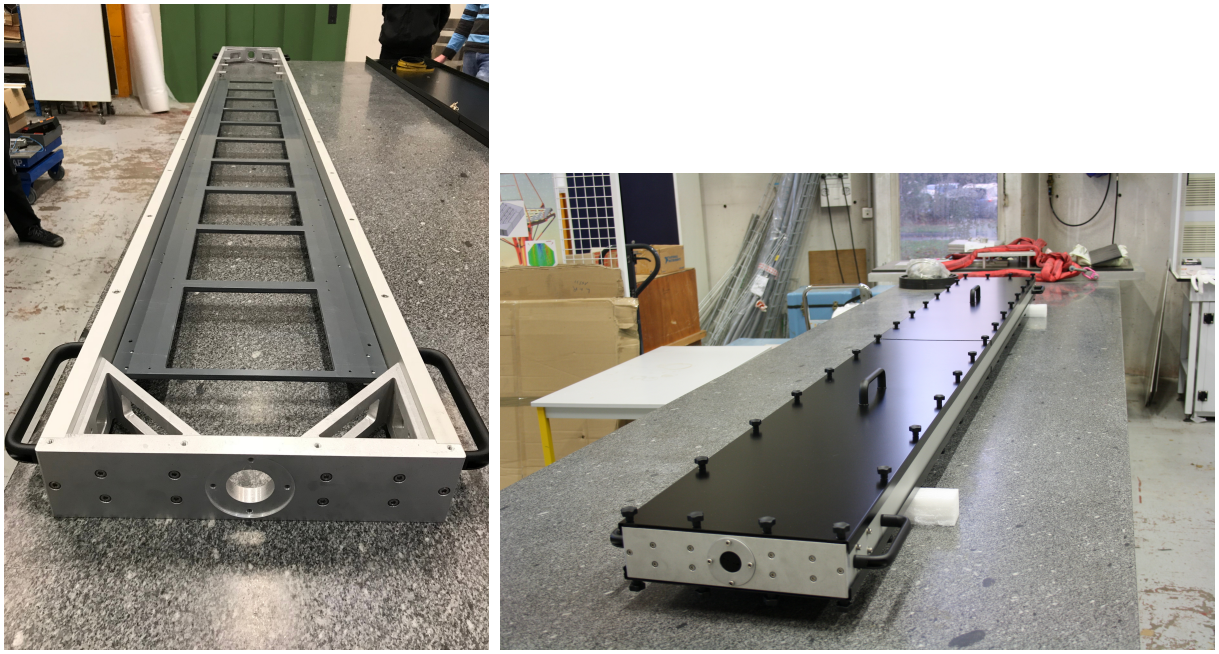


FIGURE 5.39 – Cadre en aluminium du slab long soutenant le chassis en PVC. Des capots noirs peuvent être fixés dessus et dessous pour masquer la lumière.

Sur ce châssis, nous avons constitué une chaine de 8 *ASUs* connectés en ligne. Au niveau électrique, nous avons fait le choix d'utiliser des connecteurs souples à nappe plutôt que la soudure de kapton que nous utilisons habituellement pour les *slabs* courts. Les nappes permettent plus de souplesse de manipulation et leurs performances électriques sont à peu près similaires.

Ce développement ayant été contemporain du développement des ASU version 13, décrite dans la section précédente, le slab long est équipé avec des ASU version 12 cablées avec la version 2 de l'*ASIC* Skiroc. Elles sont fixées sur le châssis au moyen de fixation souples réalisées à l'imprimante 3D en NinjaFlex. Ce système permet de supporter une flexion de l'ensemble de la structure sans endommager les wafers qui, eux, ne supportent aucune flexion tant ils sont fragiles.

Les *ASUs* sont toutes équipées d'un baby-wafer, c'est à dire un petit *wafer* de 4x4 pixels. En effet, il aurait été trop onéreux d'équiper tous les *ASUs* de *wafers* en taille standard pour un simple prototype électrique. Néanmoins, les quatre centimètres carrés suffisent largement à contenir un faisceau collimaté d'électrons.

La figure 5.40 montre le slab long équipé de toute son électronique, la *DIF*, la *SMB* et la chaine d'*ASUs*.

FIGURE 5.40 – Electronique du slab long. Les cartes sont chaînées entre elles par des connecteurs à nappe et reliées à l'électronique de lecture en bout de slab. (Credit J. Nanni, LLR)

Afin de pouvoir tester le slab long dans un faisceau de particules, nous avons également construit une structure porteuse, représentée sur la figure 5.41. Celle-ci, relativement simple, est un portant de hauteur variable, ajustable à gros grain par une système de picots et disposant d'un deuxième système à vis pour le règlage fin. Afin de pouvoir la positionner dans le faisceau, la structure est équipée de roues à blocage directionnel. Un système d'axe permet de présenter le slab long avec une orientation arbitraire repérée par un rapporteur. Les baby-wafers sont matérialisés sur les capots par de petites cibles blanches afin de pouvoir ajuster l'ensemble avec un laser. La DAQ nécessaire à l'acquisition des données est posée sur une table roulante solidarisée avec la structure.

5.7.2 Mise en service et Validation

Afin de mettre en évidence tout problème de fonctionnement lié à la longueur du prototype, nous l'avons assemblé étape par étape avec un protocole de validation complet. A chacune de ces étapes, le fonctionnement correct de l'électronique était testé extensivement et une validation physique était effectuée au moyen d'une source radio-active. Grâce à ce protocole, nous avons pu déterminer la source de trois disfonctionnements distincts.

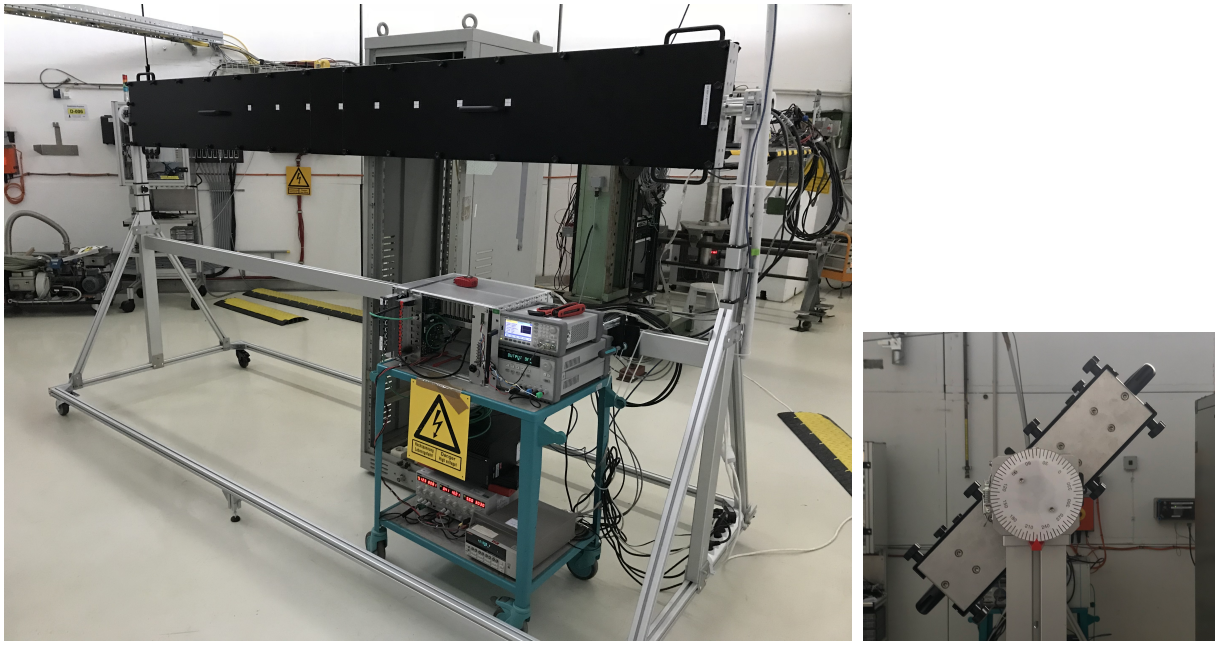


FIGURE 5.41 – Setup expérimental du slab long en faisceau. Le système est ajusté à la hauteur du faisceau puis l'ensemble du slab peut être tourné pour contrôller l'angle d'incidence des électrons.

Bruits d'alimentation

Outre les problèmes techniques mineurs qui relevaient directement du firmware, le premier problème sérieux auquel nous avons été confronté a été l'apparition de bruit dès le troisième *ASU*. Le rapport signal sur bruit que nous mesurions à partir d'une source de Cesium 137 s'est dégradé dans des proportions intolérables.

Le problème provient du cumul de bruit dans le circuit de haute tension. En effet, chaque wafer va induire un peu de bruit dans ce circuit mais les niveaux restent suffisamment bas pour ne pas perturber la lecture dans le cas d'un slab court. Au contraire, pour un slab long, les bruits des wafers vont se cumuler et atteindre des seuils qui induisent des perturbations dans le préamplificateur. Pour limiter ces bruits, il convient d'isoler au maximum les wafers les uns des autres. Pour cela, nous avons utilisé des filtres RC (passe-bas) que nous avons disposé tous les deux *ASUs*. Le bruit a complètement disparu et nous avons retrouvé le rapport signal sur bruit normal.

Cette constatation nous a amené à reconsidérer le design du futur détecteur. En effet, il était prévu d'alimenter tous les wafers avec un grand kapton de la longueur du slab. Au vu des niveaux de bruit induits, nous avons révisé notre design en optant pour des kaptons courts, de la taille d'un seul *ASU* et d'une haute tension acheminée par les connecteurs. Ce nouveau design devra être évalué.

Perturbations d'horloge

Le deuxième problème auquel nous avons été confronté est apparu au delà du cinquième *ASU*. Il devenait impossible de configurer le détecteur. Cette configuration consiste essentiellement à envoyer un train de bits sur une ligne dédiée à 2.5MHz qui passe d'*ASIC* en *ASIC*. Une analyse

du lien série a montré que le nombre de bit transmis était trop élevé, signe d'un problème d'horloge. La mesure à l'oscilloscope de cette horloge a en effet révélé de nombreuses oscillations parasites qui induisait de coups d'horloge supplémentaires.

Afin de trouver une solution à ce problème, une simulation de la ligne d'horloge a été effectuée. Celle-ci a mis en évidence les mêmes oscillations. Cela nous a permis de tester différentes valeurs de filtres RC permettant de couper ces fréquences indésirables. Nous avons finalement trouvé des valeurs qui amélioraient la simulation (avec une coupure à 10MHz). Cette amélioration a été implémentée sur les cartes, résolvant ponctuellement le problème. Dans le futur, cette ligne d'horloge devra être optimisée comme l'ont été les lignes de données afin d'éviter les rebonds.

5.7.3 Tests en faisceau

Après les tests extensifs en cosmiques et avec la source au Césium, le slab long a été testé en faisceau au DESY. Le programme a été essentiellement composé de calibrations de *MIP* avec différents angles d'incidence (0, 45 et 60 degrés). Le programme a été relativement long car le slab doit être déplacé manuellement pour pouvoir exposer chaque *ASU* à son tour au flux de particules. En effet, comme on le voit sur la figure 5.42, seul un *ASU* reçoit du faisceau et sa petite taille limite la statistique.

L'analyse de ces données a été effectuée en collaboration avec Vladislav Balagura et Oleksandr Korostyshevskyi, un étudiant de l'université de Kiev.

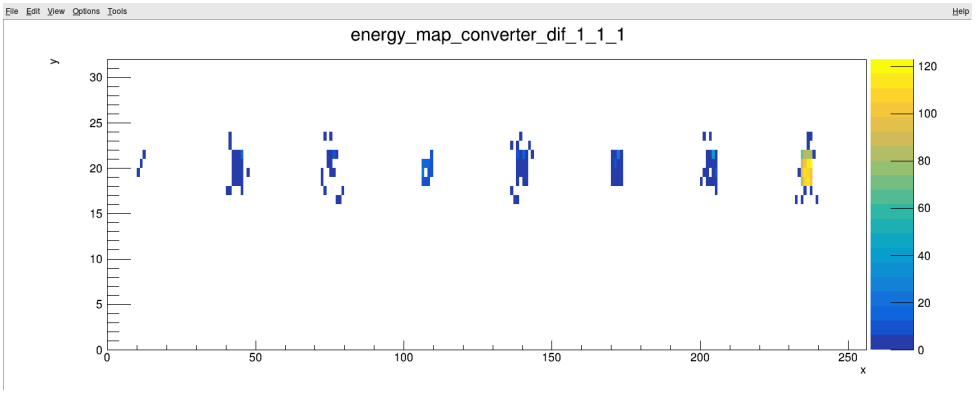


FIGURE 5.42 – Visualisation temps-réel du faisceau d'électrons intéragissant avec le 8^e wafer du slab long. Les contributions des autres ASUs proviennent du bruit et des cosmiques.

Pour chacun des *ASUs*, un histogramme de dépot d'énergie est construit puis fitté comme décrit dans la section 5.5.2. Sur le premier *ASU* de la chaine avec un angle nul, on retrouve une valeur proche de 60 pour le MPV de la première Langauss, comme on le voit sur la figure 5.43, ce qui est cohérent avec les mesures effectuées sur les slabs courts.

Lorsqu'on modifie l'angle d'incidence des électrons, la courbe change, notamment dans la partie à basse énergie, comme on le voit sur la partie gauche de la figure 5.44. Comme illustré sur la partie droite de la figure, une fraction des particules traverse deux cellules contigües, en y déposant une énergie proportionnelle à la longueur de silicium traversé. Ces petits dépots forment une composante de basse énergie qui vient s'ajouter aux *MIPs*.

Pour le modéliser, on part du principe que pour chaque bin représentant l'histogramme, une fraction fixe c (dépendant de l'angle α et du rapport largeur du pixel sur épaisseur) va être

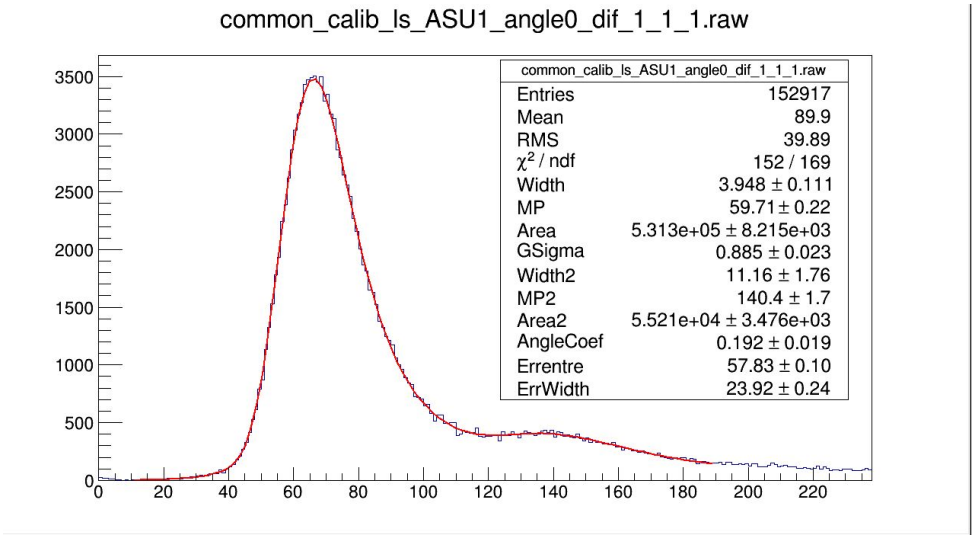


FIGURE 5.43 – Fit du dépot d'énergie au moyen de deux distributions de Landau convoluées avec une Gaussienne (Credit A. Korostyshevskiy, LLR)

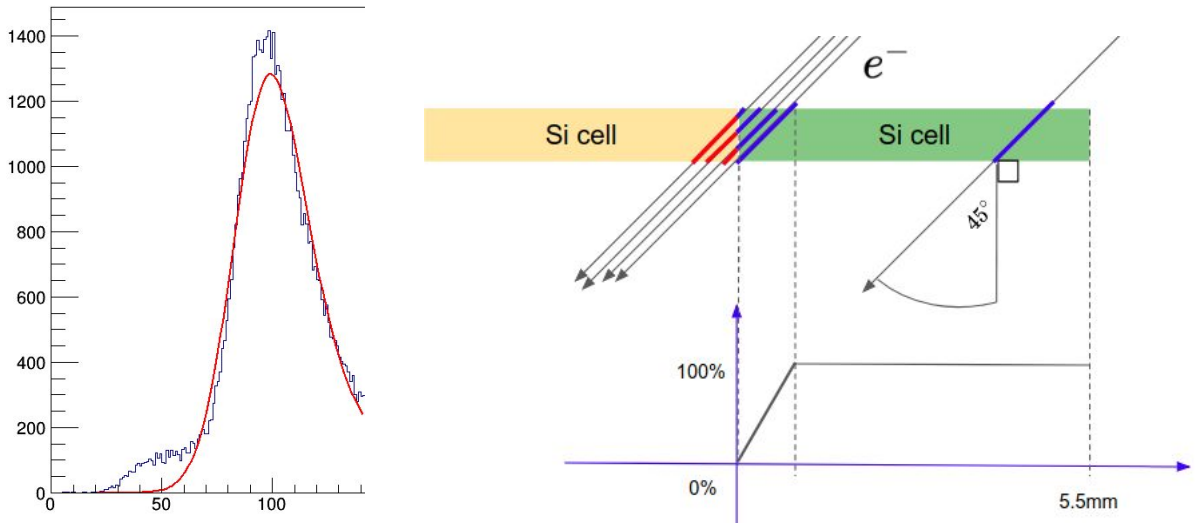


FIGURE 5.44 – Fit du dépot d'énergie pour un angle de 45 degrés. La partie basse énergie de l'histogramme révèle une composante additionnelle. L'explication à droite, les électrons déposent de l'énergie dans des cellules adjacentes. (Credit A. Korostyshevskiy, LLR)

réparti uniformément sur les bins précédents. En effet, la taille des pixels étant petites devant celle du faisceau, les dépots partiels vont être répartis uniformément en position et donc tous les découpages possibles sont équiprobables. Si on appelle $L(x)$ la fonction de dépot formée par la première Langauss, la fonction de dépot avec un angle aura la forme :

$$(1-c)\frac{L(x)}{\int_0^{+\infty} L(t)\,dt} + c\frac{\int_x^{+\infty} \frac{L(t)}{t}\,dt}{\int_0^{+\infty}\int_x^{+\infty} \frac{L(t)}{t}\,dt\,dx} \tag{5.2}$$

On part du principe que la deuxième Langauss, même si elle participe au plateau sera négligeable en raison de son amplitude très faible.

Afin de valider l'idée, nous avons effectué une simulation Geant4. Celle-ci génère bien un plateau mais continu jusqu'à zero, comme on peut le voir sur la figure 5.45. Or dans l'*ASIC* Skiroc 2, un seuil d'énergie règlable permet de sélectionner les évènements. C'est l'effet de ce seuil qui coupe le plateau.

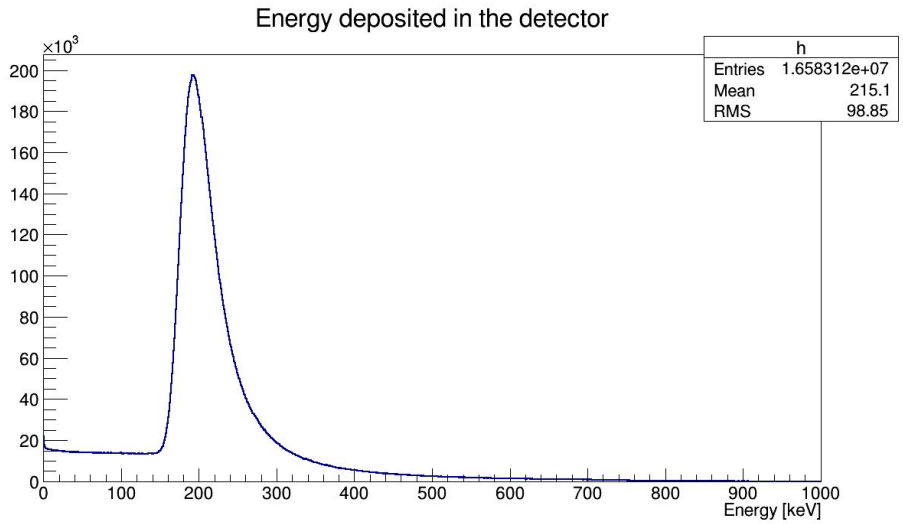


FIGURE 5.45 – Simulation du passage des électrons sur deux cellules adjacentes. Un plateau apparait à gauche de la distribution du *MIP*. (Credit A. Korostyshevskiy, LLR)

On modélise la coupure du trigger avec une fonction d'erreur (erf) que l'on multiplie à l'équation 5.2. Grâce à ce modèle, on peut donc fitter l'histogramme du dépot d'énergie et évaluer les paramètres suivants :

- Le MPV et la largeur de chacune des Langauss
- Le coefficient de dépot c
- La moyenne et l'écart type de la fonction de seuil

Un tel fit est représenté sur la figure 5.46. Il est à noter que la moyenne de la fonction d'erreur nous permet d'estimer la valeur de la coupure effectuée par l'*ASIC* en fraction de *MIP*. Cela permet d'établir une correspondance entre les valeurs en unités arbitraires utilisé à l'intérieur de l'*ASIC* et les valeurs réelles du paramètre en fonction du *MIP*. Ainsi, on peut ajuster les *ASIC*s sur une valeur comme celle qui sont estimées dans les simulations.

Au moyen de ce fit, nous pouvons estimer la valeur du *MIP* pour les différents *ASUs* qui composent le slab long. Ce fit est représenté sur la figure 5.47 Sur le plot de gauche, les valeurs sont simplement reportées pour les *ASUs* (en abscisse) et pour différentes valeurs angulaires (en couleur). On constate une forme commune pour les différents angle. Sur la droite de la figure, nous représentons les mêmes données multipliées par le cosinus de l'angle pour les normaliser. On voit apparaitre un inflexion indépendante de l'angle.

La raison de cette inflexion n'est pas encore connue mais nous supposons qu'il s'agit d'une référence en tension qui diminue sur la longueur du slab, une telle chute étant également observée sur la largeur du piedestal. Une étude spécifique est en cours pour déterminer la cause exacte.

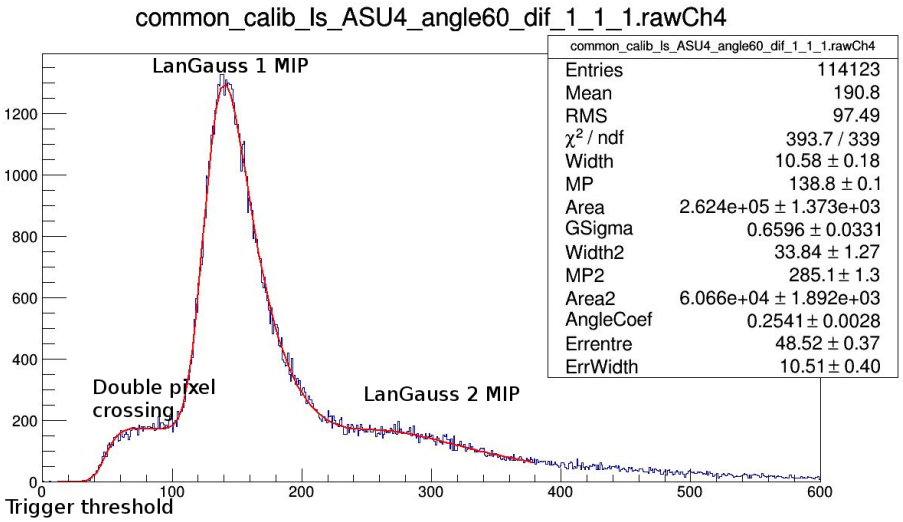


FIGURE 5.46 – Fit complet du dépot d'énergie avec un angle différent de zéro. Le fit est composé de quatre partie : les deux distributions de Landau correspondant à 1 et 2 *MIPs*, le plateau correspondant aux dépots partiels et la coupure provoquées par le seuil de déclenchement. (Credit A. Korostyshevskiy, LLR)

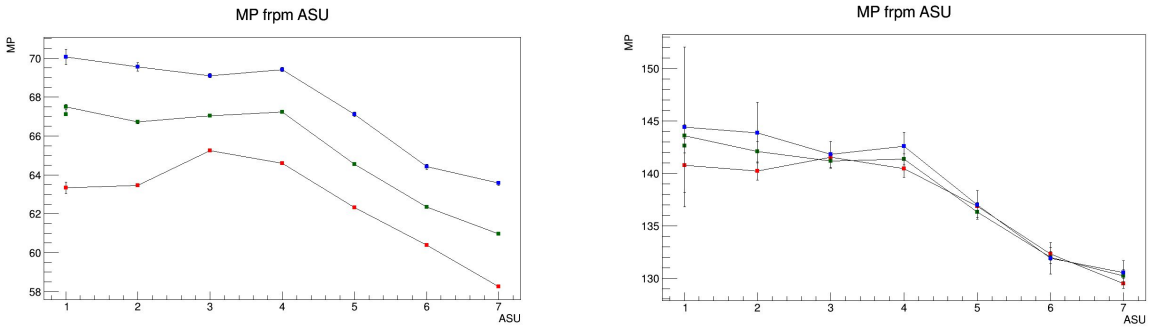


FIGURE 5.47 – Valeurs de *MIP* pour les différents *ASUs* et les différents angles (corrigés par cos(angle) pour le plot de droite). On voit clairement une inflexion sur la longueur du slab. (Credit A. Korostyshevskiy, LLR)

5.8 Intégration inter-détecteurs

Un détecteur comme l'ILD est constitué d'un ensemble conséquent de sous-détecteurs qui ont chacun leur spécificité. Afin d'obtenir un fonctionnement synchronisé de l'ensemble, il est nécessaire que les logiciels de gestion et les électroniques des sous-détecteurs puissent communiquer avec un système central permettant de les gérer dans les phases de configuration comme d'acquisition de données.

La ligne directrice pour une telle intégration est basée sur des travaux entrepris dans le cadre d'un projet européen H2020, AIDA 2020, qui comprend un work-package dédié à ces problématiques. Le groupe de travail a émis un document décrivant le niveau de base de coopération des sous-détecteurs. Ceux-ci devront pouvoir être pilotés par un logiciel nommé Eudaq pour les aspects logiciels et par une carte électronique nommée TLU.

Afin d'anticiper sur cette démarche globale, nous avons entrepris en 2015 un *test faisceau* com-

mun avec le SDHcal, un calorimètre hadronique développé par l'équipe d'Imad Laktineh, à l'IPNL de Lyon, decrit dans [19]. Pour cela, nous avons monté les deux détecteurs alignés dans le faisceau du SPS au CERN. Le niveau d'avancement du projet TLU ne permettant pas encore de s'en servir dans un *test faisceau*, les électroniciens des deux projets ont développés une carte spécifique nommée "Master CCC" qui fourni les signaux dont ont besoin les CCCs pour fonctionner.

D'autre part, nous avons tenté une synchronisation des deux logiciels de DAQ avec Eudaq. Celui-ci, alors en version 1, arrivait à synchroniser correctement les deux *DAQ*. Malheureusement, la collecte de données ne fonctionnait pas correctement et les fichiers de données qu'il produisait étaient corrompus.

C'est pourquoi nous avons du re-développer un système de coopération entre nos deux détecteurs en attendant que TLU et Eudaq soient améliorés par leurs concepteurs. Comme nous n'avions pas de point central de collecte des données, nous avons décidé de transmettre les données d'un système à l'autre dans les deux sens.

Du côté Calicoes, un programme similaire à un moniteur *online* recevait les données des décodeurs *online*. Ces données étaient ensuite formattées de façon à être lisible par la *DAQ* du SDHcal et envoyée vers sa machine de *DAQ* via une connexion TCP. Dans l'autre sens, l'ensemble des données du SDHcal étaient récupérée par la chaine d'acquisition de Pyrame, connecté directement au système d'export de données du SDHcal, basé sur DIM.

Les données ainsi échangées ont pu être analysées et des gerbes communes ont pu être mise en évidence comme le montre la figure 5.48.

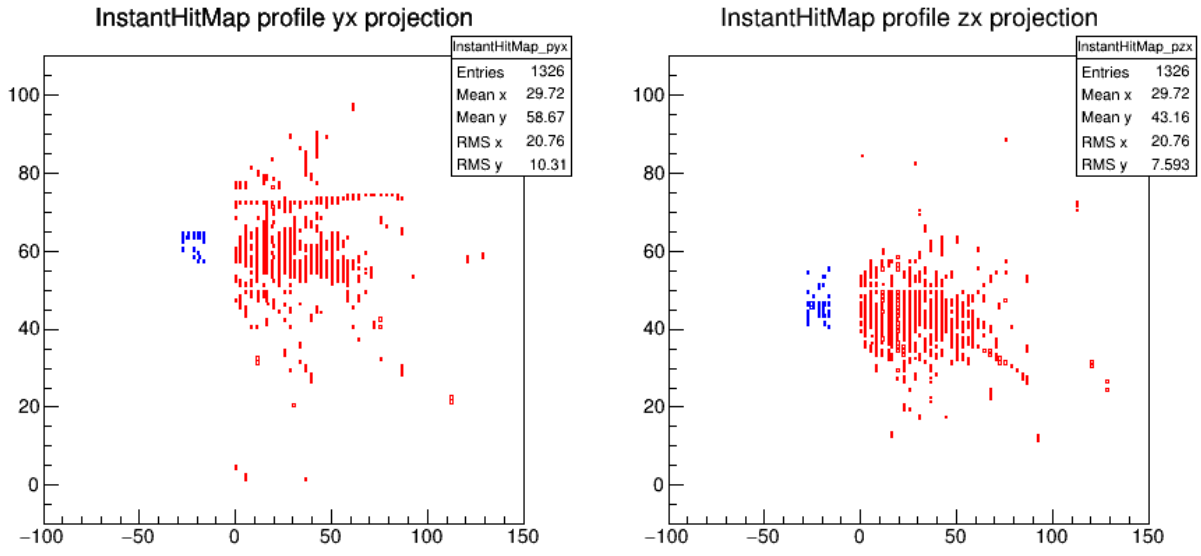


FIGURE 5.48 – Trace combinée d'un pion provoquant une gerbe dans le *SiW-Ecal* en bleu et le SDHcal en rouge (Credit L. Mirabito, IPNL)

En 2018, nous avons effectué un nouveau test combiné avec le SDHcal. Les données sont en cours d'analyse.

5.9 Conclusion

Le projet SiW-Ecal est un projet très actif qui requièrent de nombreux développement tant dans le domaine du contrôle-commande que dans celui du traitement *online* des données. Le projet Pyrame sert de base même si des composants spécifiques ont du être développés spécificiquement. Le système de *dispatcher* de données permet de nombreux développements dans le domaine de la reconstruction, de l'analyse *online* et du monitoring.

Chapitre 6

Wagasci - T2K

Wagasci (Water Grid And SCIntillator) est une expérience destinée à améliorer significativement la précision de la mesure de section efficace par courant chargé de l'interaction des neutrinos sur l'eau et sur le plastique scintillant (atomes de C et H). Cette mesure est aujourd'hui entâchée d'une grande imprécision qui est la principale limitation pour déterminer précisément les coefficients de la matrice d'oscillation des neutrinos.

6.1 T2K

L'expérience T2K (Tokai to Kamioka), au Japon est un complexe de recherche visant à mesurer cette oscillation. Un flux de neutrinos muoniques est produit au J-PARC à Tokai. Ce flux est mesuré par les détecteurs proches, Ingrid (sur l'axe) et ND280 (hors axe). Il est ensuite mesuré par le détecteur lointain Super-Kamiokande, situé à 295 km, comme indiqué sur la figure 6.1. La différence de taux d'apparition des neutrinos électroniques entre les deux détecteurs permet d'estimer la probabilité d'oscillation $\nu_\mu \rightarrow \nu_e$.

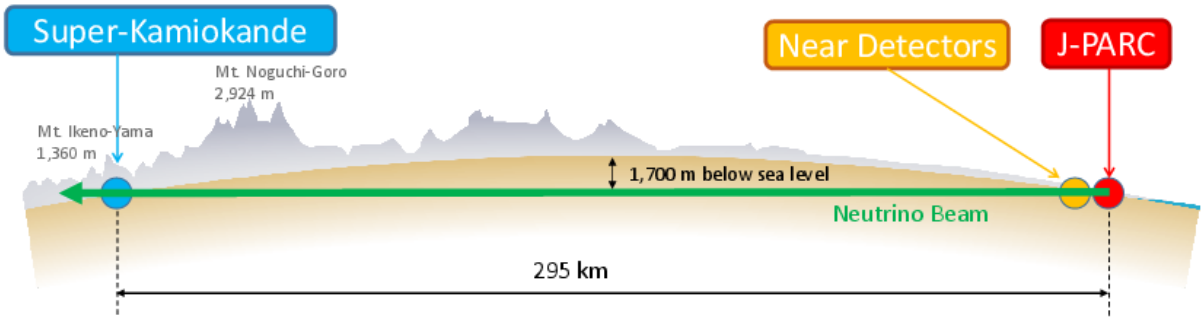


FIGURE 6.1 – T2K

Le J-PARC contient trois accélérateurs de protons, un accélérateur linéaire de 400 MeV (LINAC), un synchrotron rapide de 3 GeV (RCS) et un synchrotron principal à 30 GeV (MR). Pour produire les neutrinos, ces protons sont envoyés sur une cible en graphite. Les pions produits se désintègrent dans un tunnel de désintégration, produisant un flux de neutrinos. Le détecteur

lointain est décalé de la cible de 2.5 ° pour optimiser le spectre d'énergie des neutrinos, produisant un flux au premier maximum d'oscillation autour de 600 MeV.

6.2 Wagasci

La motivation pour l'expérience Wagasci vient des différences structurelles qui existent entre les détecteurs proche et lointain de T2K. En effet, le détecteur proche ND280 est formé de lattes de plastique scintillant lues par des photo-multiplicateurs (MPPC). Il mesure essentiellement les leptons chargés dont l'impulsion est dirigée vers l'avant du faisceau. Le détecteur lointain est une piscine cylindrique de 40 m de haut pour 40 m de diamètre, contenant 50 kt d'eau. La détection des neutrinos se fait par la détection de lumière Cherenkov provoquée par la désintégration du neutrino en lepton chargé dans l'eau. Les murs de la piscine sont recouverts de photo-multiplicateurs et le détecteur a donc une acceptance de 4π.

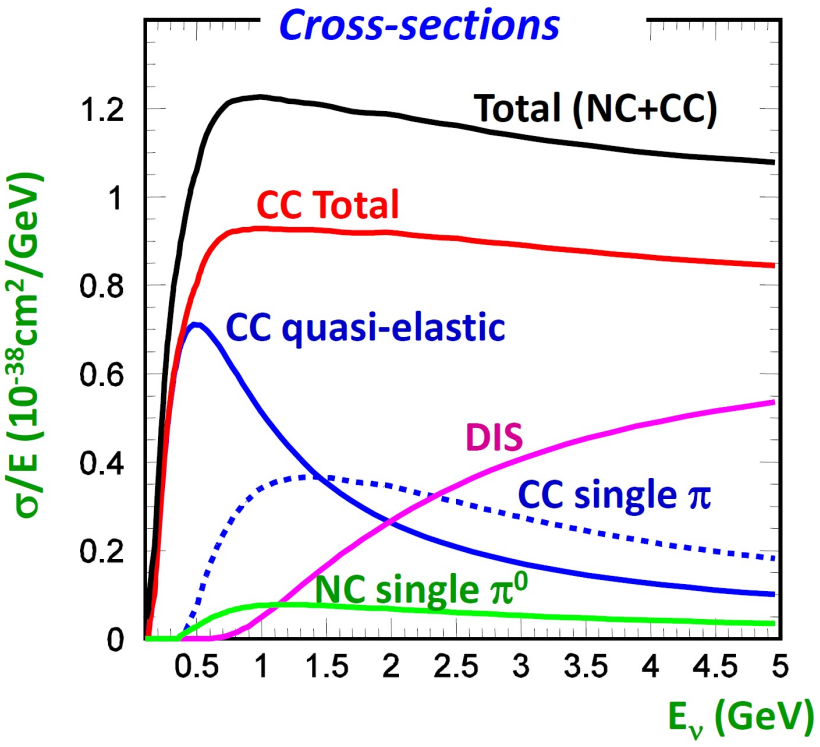


FIGURE 6.2 – Sections efficaces des interactions de neutrinos avec des noyaux. (Credit Y. Hayato, ICRR, University of Tokyo)

Comme le montre la figure 6.2, à l'énergie de T2K, quatre mécanismes d'interactions dominent :

- interaction CCQE : il s'agit du courant chargé quasi-élastique où un neutrino échange un boson W avec un neutron pour donner un proton et un lepton de même saveur que le neutrino incident : $\nu_l + n \rightarrow l + p$. C'est le signal que l'on cherche dans Wagasci car il est facile de mesurer l'impulsion du muon et ainsi de reconstruire la direction et l'énergie du neutrino.
- interaction CC1π : un pion est produit par résonance d'un baryon Δ : $\nu_l + N \rightarrow l + N' + \pi$. Cette interaction est notre bruit de fond principal.
- CCDIS/CCmultiπ : courant chargé profondément inélastique. L'interaction va produire

des hadrons ou plusieurs pions : $\nu_l + N \rightarrow l + N' + hadrons/multi - pions$.
- NCela : il s'agit d'un courant neutre élastique où le neutrino échange un boson Z avec un proton : $\nu_l + N \rightarrow \nu_l + N$.

Les sections efficaces de ces différentes interactions dépendent de la nature des noyaux des absorbeurs et n'est connue qu'avec une précision faible.

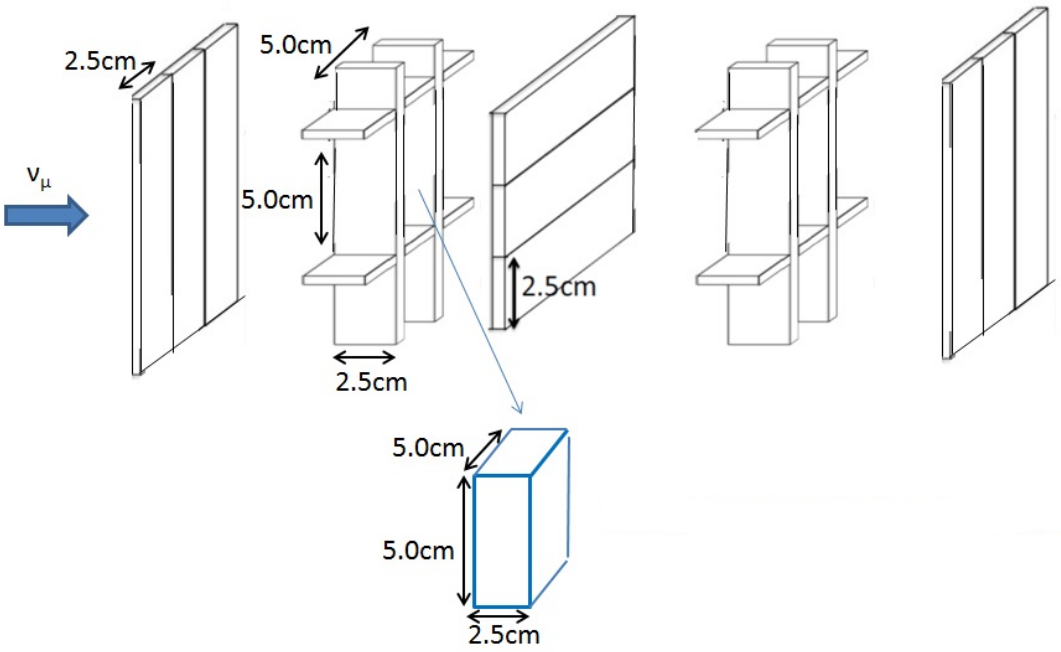


FIGURE 6.3 – Schéma d'organisation des scintillateurs centraux. (Credit Wagasci collaboration)

Ces différences structurelles introduisent des biais importants dans l'évaluation de la différence de flux entre les deux détecteurs, limitant fortement la précision. Le but de Wagasci est de faire une mesure d'un même flux de neutrinos, simultanément avec du plastique et de l'eau pour contraindre le modèle et gagner en précision sur les mesures d'oscillation.

La partie centrale du détecteur contient la cible matérielle des neutrinos. Elle est constituée de modules d'une taille de $100 \times 100 \times 50$ cm^3 placés les uns à la suite des autres. Il sont composés d'un lacis de barres de scintillateurs en forme de croisillons très fins. Ils sont représentés sur la figure 6.3. Ils vont permettre d'obtenir la trace des particules secondaires. Leur forme permet en effet de reconstruire des traces sur un large angle d'incidence d'où une acceptance élevée. La partie vide entre les croisillons peut recevoir les matériaux interagissant avec les neutrinos. Dans la configuration actuelle, celle-ci est remplie d'eau ou laissée vide.

Lorsqu'un neutrino interagit avec la matière, des particules secondaires sont produites, dont la trace est détectée dans les scintillateurs. Autour de cette partie centrale, comme on peut le voir sur la figure 6.4, une partie plus dense baptisée MRD (Muon Range Detector), en cours de construction, permettra d'identifier plus finement les particules et de mesurer l'impulsion des muons. Ces MRDs sont composés d'un sandwich de plaques de fer et de scintillateurs. Le bruit de fond principal est composé des interactions entre les neutrinos et les MRDs ou les murs du bâtiment du détecteur. Ils sont rejetés par le temps de vol (TOF) entre le détecteur central et les MRDs.

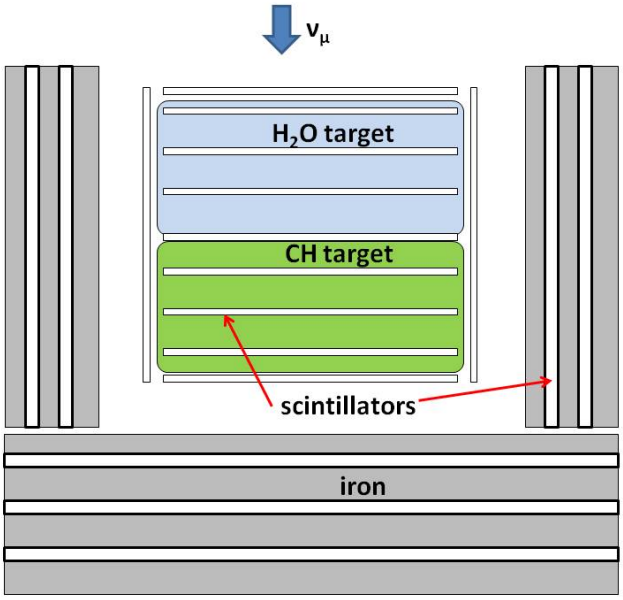


FIGURE 6.4 – Schéma de principe du détecteur. Les cibles sont placées dans le flux de neutrinos. A chaque fois qu'une intéraction a lieu, elle est détectée par le lacis de scintillateurs mais aussi par la partie métallique latérale. (Credit T. Koga, Wagasci collaboration)

Les scintillateurs sont lus par des fibres changeuses de longueurs d'onde (Wavelength shifting fibers) et des compteur de photons multi-pixels en semi-conducteurs (MPPC).

6.3 *DAQ*

La lecture des MPPCs est effectuée par une *DAQ* basée sur l'*ASIC* Spiroc de l'Omega et sur la *DAQ* générique du LLR (celle développée initialement pour le *SiW-Ecal*). Cette dernière a reçu quelques ajustements permettant d'intégrer le détecteur dans l'environnement du J-PARC. Ce travail a été réalisé conjointement avec Franck Gastaldi et Naruhiro Chikuma sous la direction de Thomas Mueller. Il a donné lieu à une publication [22].

Tout d'abord, un *ASU* spécifique a été développé. Il est représenté sur la figure 6.5 à gauche. Il joue le rôle d'intermédiaire pour transporter les horloges, les alimentations et les signaux. Ces *ASUs* sont chainés au sein de modules de détection (à droite sur la figure) formés de 20 *ASUs*, une carte adaptatrice en haut sur laquelle vient se fixer la *DIF* (en dessous de la carte adaptatrice). Les *ASUs* sont chainés entre eux par des nappes dont on voit les torons en haut du module.

La *DIF* a reçu quelques modifications de timing pour être compatible avec celui des flux de neutrinos du J-PARC. En tant que carte générique, elle intégrait déjà les spécificités du Spiroc, utilisé quelques années auparavant dans le cadre de tests préliminaires à la fabrication du Skiroc 2. Les formats de données provenant de la *DIF* sont très similaires à ceux du *SiW-Ecal*. En effet, les balises de la *DIF* ne changent pas et les formats des Skiroc 2 et des Spiroc sont essentiellement identique. Les seules différences viennent du nombre de voix (36 pour Spiroc, 64 pour Skiroc 2) et le double gain, présent uniquement sur Skiroc 2.

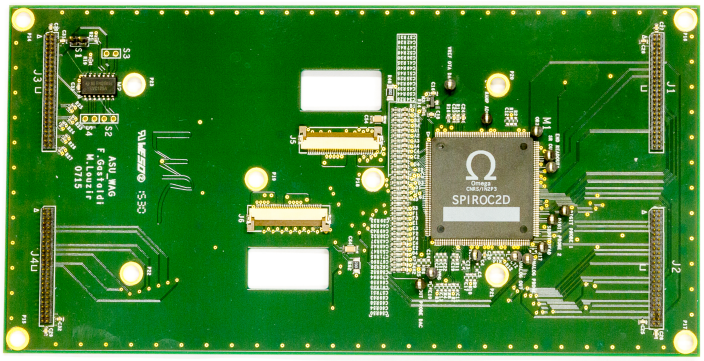


FIGURE 6.5 – *ASU* Wagasci, détail de la carte et montage dans le module de détection. (Credit F. Gastaldi, LLR - N. Chikuma, Wagasci collaboration)

Au niveau de la *GDCC*, une adaptation a été nécessaire afin d'intégrer dans les paquets de données, un numéro de spill fourni par l'accélérateur. Cette possibilité n'existait pas dans notre *DAQ* et nous avons implémenté une solution rapide basée uniquement sur la *GDCC*. A chaque fois qu'un nouveau numéro de spill est envoyé par l'accélérateur, celui-ci est converti en une commande spécifique envoyée à la *GDCC*. Celle-ci stocke le numéro dans un registre. Par la suite, tout paquet de données relayés par la *GDCC* recevra ce numéro dans un entête. Cette implémentation n'est pas totalement satisfaisante car elle oblige à mélanger des morceaux d'entête de la *GDCC* avec les données brutes au niveau de la chaine d'acquisition. Une nouvelle implémentation est en cours, qui sera commune avec *SiW-Ecal* (pour recevoir les numéros de la TLU).

La CCC, qui est une carte ancienne et impossible à fabriquer à cause de l'obsolescence des composants, a été remplacée par une *GDCC* qui joue son rôle. Cette dernière a également reçu un module TCP permettant de le configurer à distance.

Enfin une Z-Board reçoit les numéros de spill de l'accélérateur et les transmet aux *GDCC* par l'intermédiaire du réseau Ethernet.

Le logiciel Calicoes est également utilisé presque sans modification. Un driver spécifique a été écrit pour l'*ASIC* Spiroc, ainsi qu'un décodeur *online* de données. Ce dernier est basé sur le même principe que le décodeur Skiroc 2. Les différences majeures proviennent du vocabulaire détecteur qui est différent. Ainsi les plans de détections utilisés pour *SiW-Ecal* n'ont aucun sens dans Wagasci et ont été remplacés par la numérotation des plans de scintillateurs.

Un programme de monitoring spécifique a également été réalisé. Celui-ci permet de monitorer un module de détection en temps-réel. C'est-à-dire de surveiller en temps réel le taux de bruit de fond, le gain, le taux de remplissage et le timing des évènements. Un event-display a également

été intégré. La figure 6.6 montre une vue de ce moniteur.

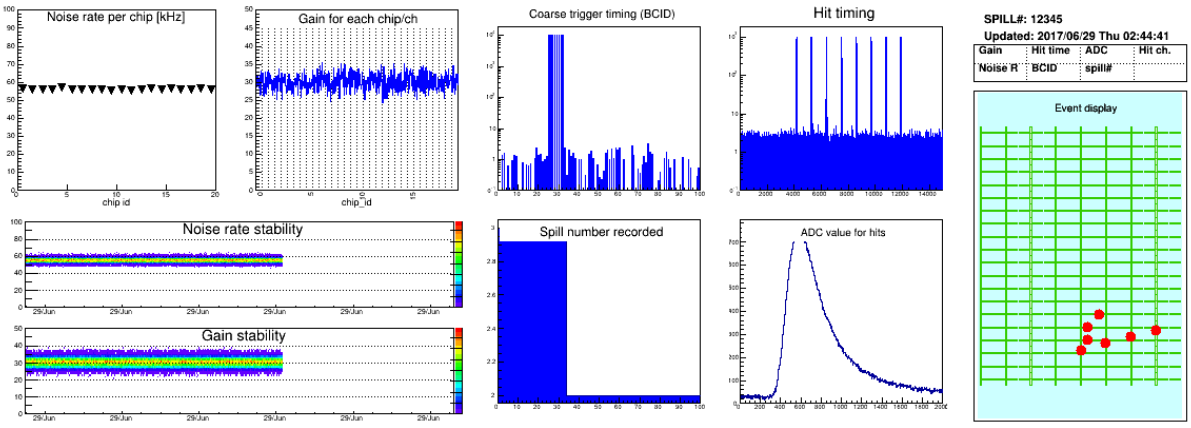


FIGURE 6.6 – Moniteur *online* Wagasci. Le système convertit les données en temps réel et permet d'afficher les différentes courbes de monitoring du système ainsi qu'un event display.

6.4 Premiers résultats

Durant l'été 2017, un premier module rempli d'eau a été installé au J-PARC au centre du détecteur Ingrid. Le 16 Octobre 2017 lors d'un premier *test faisceau*, nous avons obtenu les premières traces de collision de neutrinos, représentées sur les figure 6.7 et 6.8.

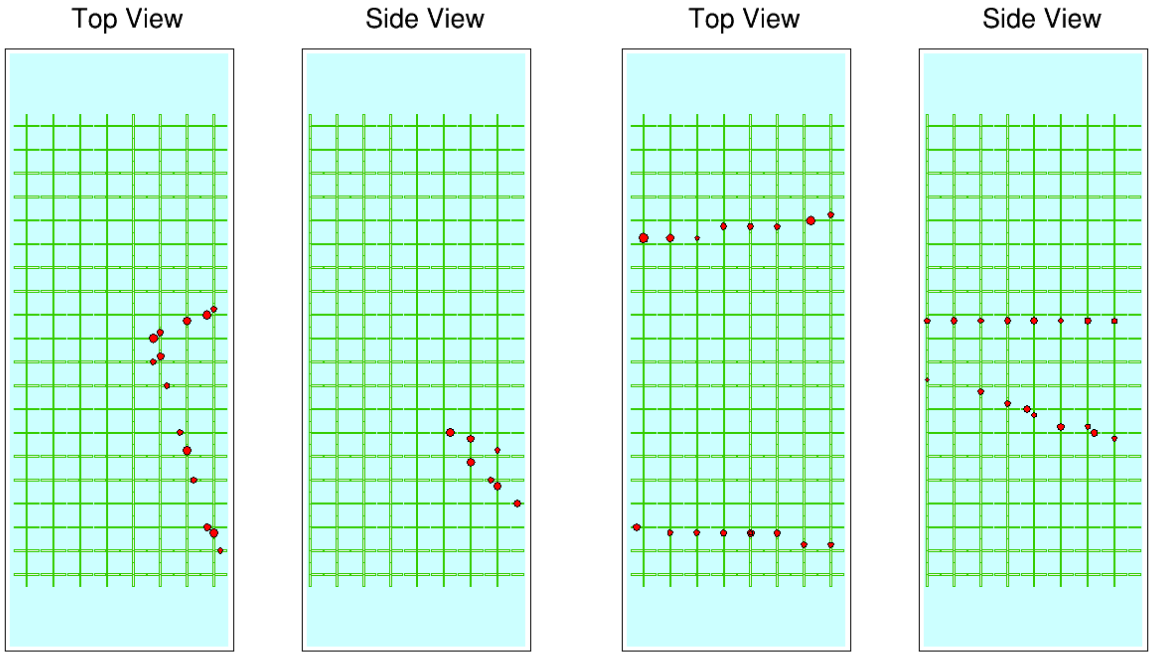


FIGURE 6.7 – Premiers évènements Wagasci. A gauche, une intéraction neutrino-noyau, et à droite, une intéraction hors-champ. (Credit Wagasci collaboration)

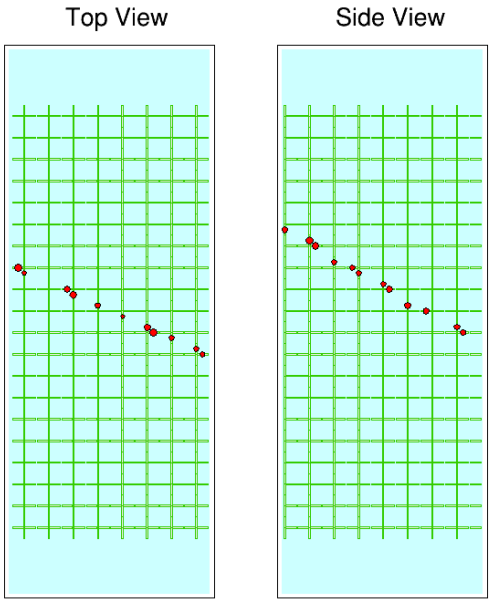


FIGURE 6.8 – Premiers évènements Wagasci. Ici un cosmique enregistré par le détecteur.(Credit Wagasci collaboration)

Sur la figure 6.7 à gauche, on voit une interaction neutrino-noyau typique soit un lepton et un hadron. A droite, on voit un bruit de fond typique, formé d'une collision hors-champ. Sur la figure 6.8, on voit un autre bruit de fond formé d'un muon cosmique.

6.5 Reconstruction

Le but de l'expérience étant une mesure différentielle des sections efficaces, il est primordial de reconstruire avec une très bonne qualité pour être certain de bien discriminer les différentes interactions. Cette reconstruction est prévue en trois étapes :

- Reconstruction indépendante des traces en deux dimensions
- Matching avec les traces des MRDs
- reconstruction en 3 dimensions

La reconstruction des traces en deux dimensions, héritée de T2K est basée sur un automate cellulaire. Cette méthode donne des résultats satisfaisants mais est relativement lente. Nous verrons dans le chapitre suivant qu'il est possible d'utiliser d'autres techniques avec des résultats tout aussi satisfaisants et une performance nettement accrue.

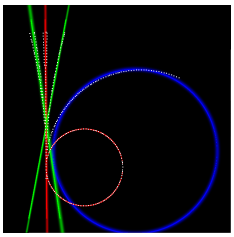
6.6 Conclusion

Pour conclure sur Wagasci, il est intéressant de constater que cette aventure, première "vraie" expérience pour Pyrame est l'illustration parfaite, bien que suivant un chemin tortueux, du

concept évoqué dans les motivations que nous avions pour démarrer le développement de Pyrame : avoir un système qui puisse nous suivre des premiers bancs de test jusqu'à l'expérience finale. Cela a été le cas pour cette *DAQ* générique, développée par la collaboration CALICE puis par le LLR, elle utilise Pyrame depuis qu'il existe et son utilisation nous a permis d'envisager des traitements de plus en plus complexes, jusqu'à fournir un système quasiment clé en main, que nous avons exploité pour Wagasci. C'est une démonstration opérationnelle de la validité du concept et la preuve que Pyrame a su devenir un *framework* générique, utilisable pour de nombreuses expériences.

Chapitre 7

Trajectographie *online*



Comme nous l'avons vu dans les chapitres précédents, les trajectoires des particules dans les détecteurs sont captées comme des nuages de points dont il faut extraire la forme, généralement linéaire. La reconstruction de ces trajectoires est facile si celles-ci sont clairement isolées dans l'espace et des algorithmes robustes, comme la transformée de Hough, permettent d'en calculer les paramètres. Toutefois, la tendance sur les détecteurs modernes est d'augmenter considérablement la luminosité, afin de maximiser la statistique obtenue et donc les chances d'observer des évènements à très petite section efficace. Cet augmentation de luminosité va provoquer de l'empilement de particules (pile-up), c'est à dire que les trajectoires simultanées vont devenir fréquentes, ce qui complique considérablement la reconstruction.

Dans ce cadre, j'ai été amené à explorer une classe d'algorithmes qui est relativement peu usitée dans le domaine de la physique des hautes énergies : les algorithmes EM. Ceux-ci permettent, entre autres, de séparer des composantes Gaussiennes au sein d'un signal. Nous verrons dans ce chapitre comment il est possible de développer des modèles dérivés pour faire de la trajectographie. Cette méthode a été utilisé avec succès sur trois jeux de données, celles d'Harpo (durant son *test faisceau* à NEWSUBARU), celles de Wagasci et celle du *SiW-Ecal* en mode *trajectographe*. Ce travail est original et a été publié dans le journal "Computer Physics Communications" [34].

7.1 Problématique

Dans le contexte d'un traitement *online* des données, la fréquence des collisions impose un rythme soutenu d'analyse et la performance de l'algorithme de reconstruction est un élément décisif dans la réalisation d'un tel système. D'autre part, il est crucial que le programme puisse réaliser ces analyses d'une manière entièrement autonome car aucune aide humaine ne peut être apportée à ces fréquences. Enfin, les environnements de mesure ne sont jamais parfaits. Un bruit permanent et à plusieurs composantes s'ajoute aux signaux physiques. La mesure est donc affectée d'une certaine imprécision, contrôlée certes, mais néanmoins limitante pour les algorithmes. Les données peuvent être en deux ou trois dimensions avec de plus en plus, des informations temporelles très précises qui constituent une quatrième dimension. En toute généralité, le problème de l'ajustement linéaire de trajectoire peut se formuler sous la forme suivante :

Soit un nuage de points dans l'espace, est-il possible d'ajuster (*fit*) un ensemble de droites en minimisant l'erreur et avec les contraintes suivantes :

- Traitement complètement automatique
- Sans hypothèses préliminaires (par exemple sur la forme du détecteur)
- Suffisamment rapide pour un traitement *online*
- En dimension arbitraire
- Avec une précision contrôlée
- Même en présence de bruit

7.1.1 Transformée de Hough

La transformée de Hough est une méthode, décrite dans [31] et qui permet de faire un tel ajustement de trajectoire. Son fonctionnement est simple et élégant.

Soit P un ensemble de n points P_i. Pour chaque couple de points (P_i, P_j), on va calculer l'angle du segment avec une droite de référence (l'un des axes du repère). Avec ces angles, on va construire un histogramme dans lequel les pics vont correspondre aux directions privilégiées par les couples de points. Il suffit ensuite de sélectionner ces pics, l'un après l'autre, avec une détection par seuil, récupérant ainsi les directions des droites les plus proches des données. La figure 7.1 montre un exemple de ce type de *fit*.

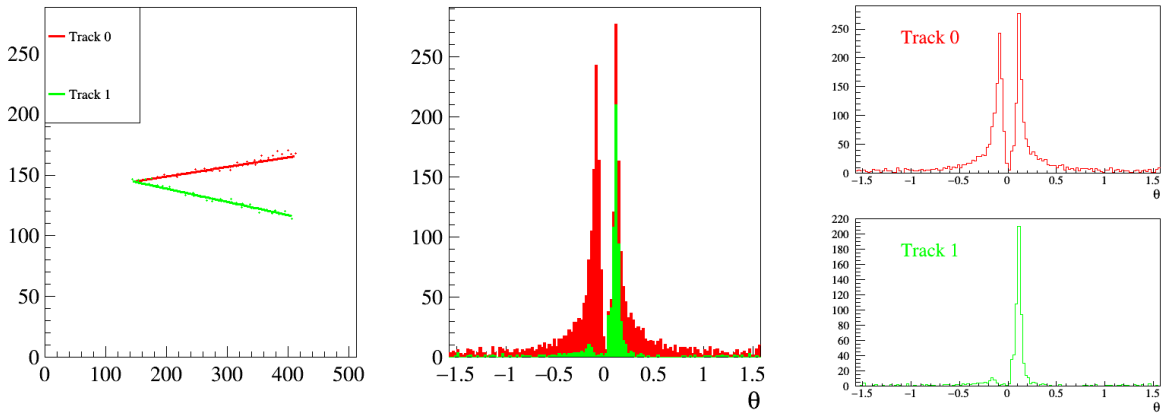


FIGURE 7.1 – *Fit* d'un nuage de points par une transformée de Hough. Au gauche, le nuage de points et leur fit, au centre l'histogramme des directions et à droite les différentes composantes. (Credit P. Gros, LLR)

Cette méthode, décrite ici en dimension deux est généralisable à n'importe quelle dimension et fournit facilement des *fits* de bonne qualité. Elle permet de faire un *fit* sur quasiment n'importe quel type de données (pas seulement géométriques). Elle est également extensible à d'autre formes que la droite (par exemple des fits circulaires).

Néanmoins, cet algorithme présente quelques désavantages dans notre cas :

- Il n'est pas facile de choisir le découpage de l'histogramme (le binning). S'il est choisi trop fin, la complexité de l'algorithme explose et s'il est choisi trop grossier, on perd en résolution. Dans la pratique, on est souvent obligé de faire tourner plusieurs fois l'algorithme, tout d'abord avec un gros grain puis en zoomant sur les zones comportant des pics.

- L'algorithme est en $O(n^2)$ en dimension 2 mais cette complexité va avoir tendance à devenir exponentielle dans les dimensions supérieures ou sur des formes non linéaires. Ceci, couplé aux exécutions multiples à grain différent peut rendre cet algorithme trop gourmand en ressources computationnelles pour être utilisable dans un contexte *online*.

Pour toutes ces raisons, il est intéressant de développer des approches alternatives à cette transformée, même si celle-ci a toujours toute sa place dans notre discipline dans de nombreux cas.

7.2 Rappels théoriques

7.2.1 Ajustement

Un ajustement (fit) est une procédure pour évaluer les paramètres d'un modèle à partir d'un jeu de données. Cette évaluation est effectuée au moyen d'un algorithme dédié pour chaque paramètre, appelé son estimateur. La qualité d'un *fit* peut être estimée en calculant une fonction d'erreur, généralement basée sur une distance entre les données et le modèle. Le meilleur ajustement (qu'en général nous appelons simplement l'ajustement) est celui qui minimise la fonction d'erreur. Si l'ajustement est pondéré, un poids que nous noterons τ_i est affecté à chacun des points, noté P_i. La fonction d'erreur doit également intégrer cette pondération. Si une distance est définie entre les points P_i et le modèle M de paramètre θ, la fonction d'erreur doit avoir une forme du type

$$\varepsilon = \frac{\sum_{i=1}^{n} \tau_i d(P_i, M(\theta))}{\sum_{i=1}^{n} \tau_i}. \tag{7.1}$$

7.2.2 Mélanges Gaussiens

Les modèles de mélange Gaussien sont des fonctions du type

$$G(x, \Theta) = \sum_{k=1}^{K} \pi_k g(x, \theta_k) \tag{7.2}$$

où la fonction g est la fonction Gaussienne dont les θ_k sont les paramètres. Les π_k sont des coefficients réels qui représentent la pondération du mélange. Leur somme doit valoir 1. La fonction G est donc un mélange pondéré de Gaussiennes. Ces modèles, classés dans les méthode d'apprentissage automatique, sont très utilisés dans de nombreux champs d'applications scientifiques nécessitant une analyse dirigée par les données et contenant des facteurs explicatifs cachés.

Leur succès tient essentiellement à un algorithme qui permet de retrouver les paramètres π_k et θ_k à partir de la fonction G et d'expliquer un phénomène complexe par une analyse de ces diverses composantes. Cet algorithme, introduit en 1977 par Dempster et al [16], s'appelle EM pour Expectation-Maximization. En effet, l'algorithme est itératif et repose sur ces deux opérations qui sont appelées alternativement jusqu'à obtenir la convergence d'un critère numérique.

Imaginons que nous disposons de n réels x_i dont nous souhaitons expliquer la distribution par un mélange de K Gaussiennes. Nous initialisons les K distributions au hasard puis nous appliquerons les deux étapes suivantes, illustrées par la figure 7.2 :

- La phase Expectation consiste à calculer pour chaque point, sa probabilité d'appartenance à chacune des Gaussiennes. On utilise pour cela la vraisemblance normalisée de la distribution. Pour le point x_i :

$$w_{i,k} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i - \bar{x}_k)^2}{2\sigma_k^2}} \tag{7.3}$$

  Ces coefficients sont ensuite normalisés
- La phase Maximisation consiste à calculer les paramètres des Gaussiennes sur la base des points pondérés par la phase précédente :

$$\bar{x}_k^* = \frac{\sum_{i=1}^n w_{i,k} x_i}{\sum_{i=1}^n w_{i,k}} \tag{7.4}$$

$$s_k^* = \sqrt{\frac{\sum_{i=1}^n w_{i,k}(x_i - \bar{x}_k^*)^2}{\sum_{i=1}^n w_{i,k}}} \tag{7.5}$$

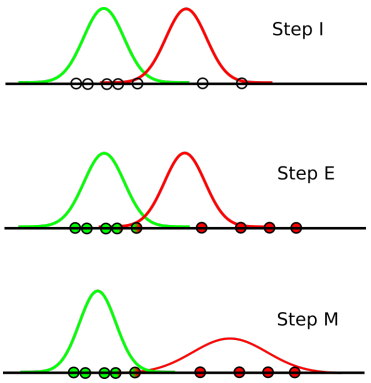


FIGURE 7.2 – Illustration des 3 phases de l'algorithme EM : Initialisation : les Gaussiennes sont tirées au hasard. Expectation : les poids d'affectation des points sont estimés, représentés par la couleur des petits cercles. Maximisation : les Gaussiennes sont ré-évaluée sur la base des nouveaux poids

Comme critère de convergence, on utilise la stabilité de la log-vraisemblance. A chaque étape, on calcule

$$ll = \log(\prod_{i=1}^{n} P(x_i)) = \sum_{i=1}^{n} \log(P(x_i)) \tag{7.6}$$

L'algorithme nous garantit que cette grandeur décroit par construction à chaque étape jusqu'à une valeur minimale. Lorsque la différence de cette log-vraisemblance passe sous un seuil entre deux itérations $I-1$ et I :

$$\mid ll_{I-1} - ll_I \mid < s \tag{7.7}$$

on peut considérer que l'algorithme a convergé. Malheureusement, la minimalité de la log-vraisemblance n'est garantie que localement. L'algorithme ne donne aucune garantie d'atteindre un minimum global s'il existe. Ainsi, il existe des cas où l'algorithme peut "tomber" dans un minimum local et ne jamais en sortir. L'algorithme ne donne pas non plus de borne sur le nombre

d'itérations nécessaires à la convergence. Ainsi, il existe des cas où celle-ci est tellement lente que l'algorithme peut être considéré comme divergent.

D'une manière générale, l'algorithme est extrêmmement sensible à l'initialisation. Celle-ci, consistant à tirer au hasard les distributions, peut mener à des exécutions radicalement différentes : convergence rapide vers un minimum global, convergence rapide vers un minimum local ou encore divergence. Cela peut obliger, dans le cas où l'espace des phases est très accidenté à utiliser un méta-algorithme, voir par exemple [2], qui va créer un ensemble de conditions initiales, faire tourner l'algorithme sur toutes ces configurations et choisir la meilleure en terme de log-vraisemblance.

Il existe un autre cas où un méta-algorithme est nécessaire, c'est celui où le nombre de composantes n'est pas connu. Dans ce cas, il va falloir essayer l'algorithme avec plusieurs tailles de mélange et faire un choix parmi les résultats. De nombreux travaux dans le domaine, dit de la sélection de modèle, fournissent des cadres théoriques précis mais aussi des méthodes empiriques pour choisir les meilleurs résultats en maximisant la log-vraisemblance tout en pénalisant les modèles par un fonction de leur nombre de degrés de liberté. En effet, le meilleur *fit* est celui qui propose une Gaussienne par point mais le résultat n'a aucun intéret d'un point de vue pragmatique. Par exemple, le critère BIC, présenté dans [40] pénalise les modèles par une fonction du nombre d'observations et du nombre de paramètres.

7.3 Modèle de mélanges de régression linéaire

Le cas qui nous intéresse fréquemment en physique des particules n'est pas celui des points répartis selon une Gaussienne mais plutôt celui d'un bruit Gaussien réparti autour d'une droite de trajectoire. Un modèle dérivé du cas Gaussien permet de traiter ce cas, il s'agit du modèle de mélange de régressions linéaires, que nous désignerons dans la suite par l'acronyme LRMM (Linear Regression Mixture Model). Ce modèle a été adapté dans [44] avec la forme générale de la k^{ieme} composante k de la forme

$$y = x\beta_k + \varepsilon_k \tag{7.8}$$

La distribution d'erreur ε_k suit une Gaussienne centrée $N(0, \sigma_k^2)$.

La vraisemblance n'est pas calculable analytiquement, aussi on utilise un algorithme EM pour estimer les paramètres du mélange. Durant la phase de maximisation, les paramètres Gaussiens sont estimés en même temps que les pentes des régressions.

Cette méthode a été implémentée dans le package "mixreg" décrit dans [43] et disponible pour le logiciel de statistiques R.

7.4 Jeux de données de l'étude

Pour ce chapitre, nous allons reprendre trois jeux de données qui ont déjà été rapidement décrits dans les chapitres précédents. Le premier est celui du test faisceau de Harpo à NEWSUBARU décrit dans le chapitre 3. Il génère des traces comme celle de la figure 7.3. Comme on le voit, elles sont formées d'une, deux ou trois droites plus ou moins tordues par la diffusion multiple dans le gaz. Les trajectoires à une droite sont en général des diffusions Compton $e^-\gamma \to e^-\gamma$. La production majoritaire aux énergie de Harpo sont constituée de deux droites et correspondent

à une production de paire électron/positron : $\gamma Z \rightarrow e^+e^-Z$. Enfin des traces à trois droites correspondent à une production de triplet $\gamma e^- \rightarrow e^+e^-e^-$).

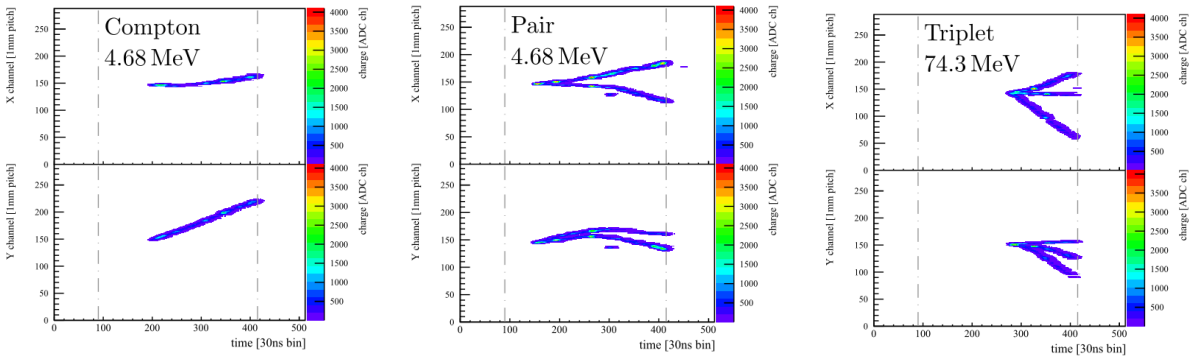


FIGURE 7.3 – Les trois formes de traces $x(t), y(t)$ generées par Harpo. A gauche, un diffusion Compton, au milieu une production de paire et à droite une production de triplet. Ces traces sont perturbées par la diffusion multiple et le bruit électronique. (Credit P. Gros, LLR)

Le détecteur Harpo est une chambre à dérive qui produit énormément de charges et donc énormément de points dans les graphes (typiquement plusieurs milliers par interaction). La plupart des algorithmes de fit linéaire ont une complexité qui dépend du nombre de points et sont incompatibles avec une telle abondance. C'est pourquoi nous utilisons un pré-processing afin de réduire leur nombre. Pour cela, nous appliquons une grille sur l'espace et nous réduisons le jeu de données à l'ensemble des barycentres dans chaque cellule de la grille. On peut voir graphiquement ce processus sur la figure 7.6 où les points noirs sont les points originaux et ceux en rouge ou vert sont les barycentres utilisés pour le fit.

Un autre pré-processing utile pour Harpo consiste à retirer les artefacts liés au trigger comme celui que l'on voit au milieu de la figure 7.3 sous la paire. Un algorithme calcule les composantes connexes de barycentres et élimine celles qui sont considérées comme trop petites, c'est à dire représentant une fraction des points trop petite.

Le second jeu de données est celui des tests en faisceau du SiW-Ecal décrit dans le chapitre 5, spécialement les mesures effectuées sans absorbeur. Le jeu de données est constitué de traces linéaires en trois dimensions qui peuvent être légèrement perturbée par un début de gerbe électromagnétique comme décrit dans la figure 5.6. Dans environ dix pour cent des cas, plusieurs traces sont présentes, justifiant l'utilisation d'un algorithme de mélange. L'intéret d'analyser les traces sans absorbeur est de calibrer le minimum d'ionisation du détecteur. Or cette mesure dépend de l'angle d'incidence de la particule, donc pour calibrer correctement le détecteur, il est nécessaire de reconstruire cet angle avec précision. Comme on le voit sur la figure 7.4, le nombre de point par trace est assez faible et correspond au nombre de couches utilisées. Avec cette version du prototype, nous disposons de 7 couches. Dans le détecteur final nous devrions avoir 26 couches.

Le dernier jeu de données est celui de l'experience Wagasci décrite dans le chapitre 6. Ces traces représentent l'interaction entre un neutrino et un noyau, ce qui donne un muon et une autre particule chargée (proton ou secondaire). Le but du fit est, d'un part, de discriminer les muons cosmiques qui laissent une unique trace et d'autre part, de déterminer le vertex d'interaction qui est le point d'intersection des droites fittées.

La figure 7.5 présente les traces typiques d'interaction de Wagasci.

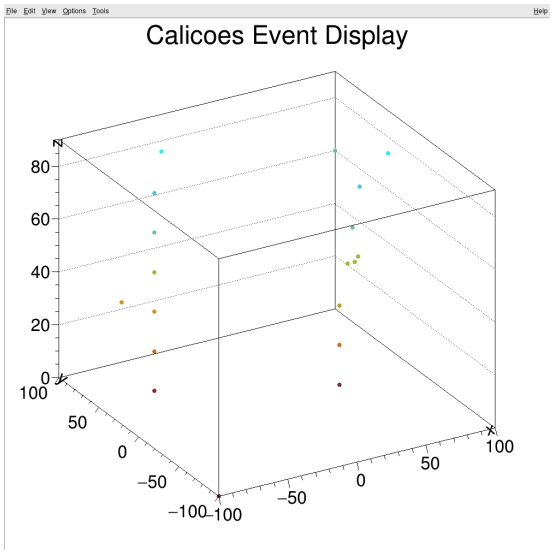


FIGURE 7.4 – Trace typique d'empilement dans le SiW-Ecal sans absorbeur : deux électrons atteignent le détecteur pendant la même période d'intégration. Leur traces sont presque linéaires.

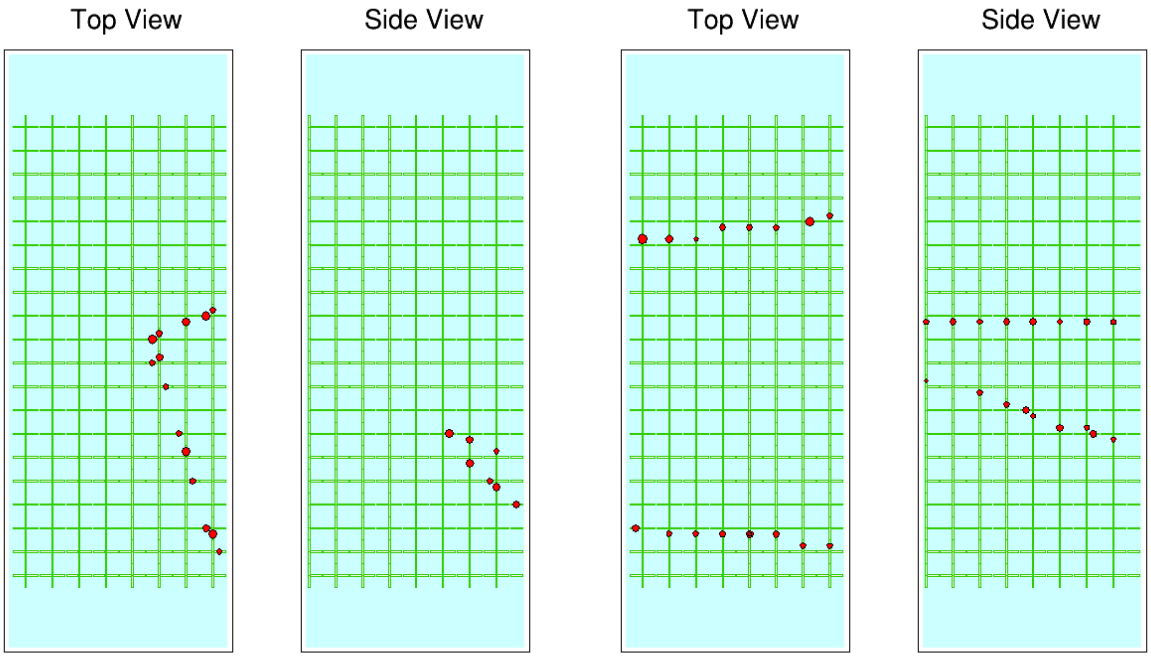


FIGURE 7.5 – évènements d'intéraction noyau-neutrino dans Wagasci, dans le champ (à gauche) et hors champ (à droite) (Credit Wagasci Collaboration)

7.5 Anisotropie du LRMM

Le LRMM donne de bons résultats sur la plupart de données Harpo pré-processées. La figure 7.6 montre deux exemples d'ajustement pour lequel l'accord entre les données et le modèle est tout

à fait correct. A droite, les points sont largement intriqués et malgré cela, l'estimation du vertex d'interaction est tout à fait plausible. Ces fits ont été effectués avec R et le module mixreg.

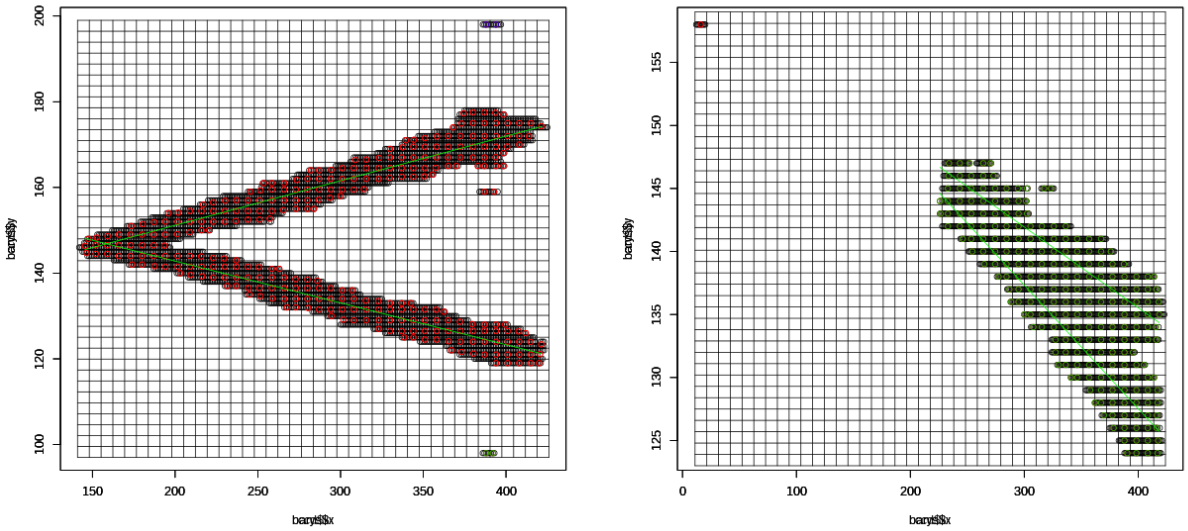


FIGURE 7.6 – Deux fits de trajectoires dans Harpo, réalisés avec le modèle LRMM

En revanche, il existe certains cas (rares mais non négligeables), où l'algorithme diverge complètement, comme on le voit sur la figure 7.7. Cela arrive lorsque l'une des trajectoires est verticale, typiquement pour une pente plus grande que 4. On peut donc supposer que le modèle n'est pas isotrope. Pour s'en convaincre, il suffit de tourner les données d'un angle de $\pi/8$ radians comme sur la partie droite de la figure. On constate que le *fit* redevient en accord avec la trajectoire.

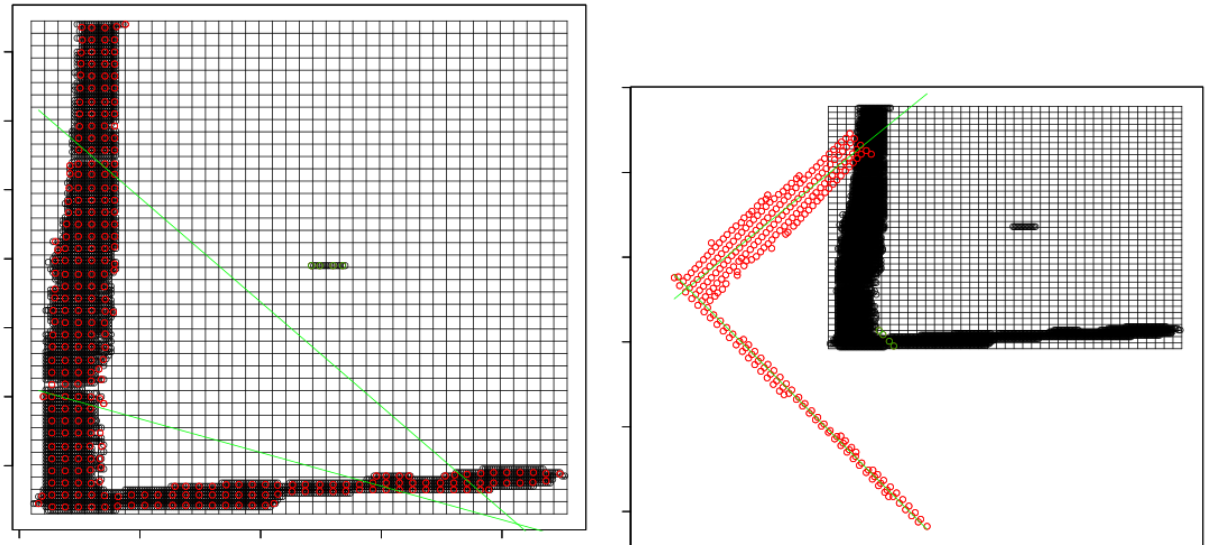


FIGURE 7.7 – Lorsque le fit contient une droite verticale, l'EM diverge. En tournant les points d'un angle de 45 degrés, on retrouve un *fit* correct

Cette anisotropie provient de l'estimateur de la variance de l'erreur Gaussienne

$$\sigma_k^2 = \frac{\sum_{i=1}^{n} \gamma_{ik}(y_i - x_i\beta_k)^2}{\sum_{i=1}^{n} \gamma_{ik}} \tag{7.9}$$

C'est-à-dire que la variance de la Gaussienne d'erreur ε est définie comme l'écart pondéré moyen entre la valeur estimée $x_i\beta_k$ et la valeur mesurée y_i. Cette différence est la distance entre le point mesuré et sa projection sur la droite selon l'axe des ordonnées. Sur la figure 7.8, on voit une représentation de la situation. La droite estimée $y = \beta x$ forme un angle $\alpha = arctan(\beta)$ avec l'axe des abscisses. Pour le LRMM, l'erreur, notée ε_T est le segment p-py, entre le point et sa projection sur la droite selon l'axe des y. Si l'on pivote les axes du repère dans le sens horaire, on voit que la distance ε_T va augmenter. Cela implique donc que la distance choisie dépend de la pente de la droite et cela explique l'anisotropie que nous avons constaté sur cet algorithme.

Pour éliminer cet anisotropie, il faut remplacer cette distance par une distance orthogonale, invariante par rotation par construction. Celle-ci est nommée ε_G sur la figure 7.8 et correspond au segment p-po.

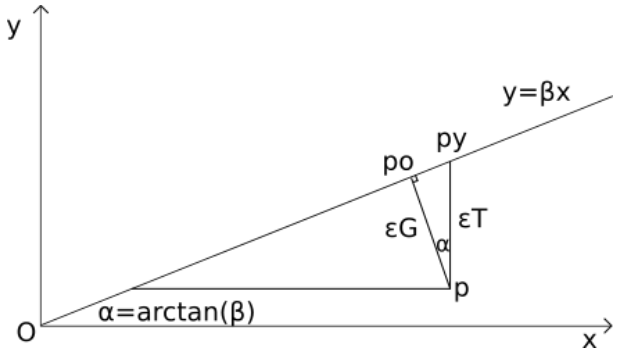


FIGURE 7.8 – Distance du point à la droite pour LRMM ou pour une distance orthogonale. L'utilisation de la distance orthogonale rend le modèle invariant par rotation.

L'angle entre les deux segments p-po et p-py n'est autre qu'α par construction. Il existe donc une relation trigonométrique entre les deux projection. ε_T, est l'hypoténuse du triangle formé par le point p, po et py, alors que ε_G est le côté adjacent, donc

$$\cos(\alpha) = \frac{\varepsilon_G}{\varepsilon_T} = \cos(arctan(\beta)) \tag{7.10}$$

or par construction du cosinus et de l'arc-tangente

$$\cos(\arctan(\beta)) = \frac{1}{\sqrt{1+\beta^2}} \tag{7.11}$$

donc

$$\varepsilon_G = \frac{\varepsilon_T}{\sqrt{1+\beta^2}} \tag{7.12}$$

Ainsi, on obtient un nouvel estimateur pour le paramètre ε_k qui garantit l'isotropie du modèle.

La fonction $1/\sqrt{1+\beta^2}$ étant continue, on peut appliquer le lemme porte-manteau [4] qui garanti les propriétés de consistance du nouvel estimateur.

Un autre avantage d'utiliser une distance orthogonale est que celle-ci possède une formulation vectorielle

$$d(P,(d)) = \frac{\overrightarrow{RP} \cdot \overrightarrow{D}}{|| \overrightarrow{D} ||} \tag{7.13}$$

où (d) est la droite représentée par un point P et un vecteur directeur $\overrightarrow{D}$. Nous pouvons donc généraliser l'algorithme à un nombre arbitraire de dimensions à condition de pouvoir calculer une régression linéaire pondérée sous forme vectorielle.

7.6 Régression linéaire pondérée en dimension quelconque

Dans cette section, nous décrivons un algorithme, appelé Φ_{line} qui permet d'effectuer une regression linéaire pondérée en dimension arbitraire. Il prend en entrée les n points à *fitter* notés P_i, τ_i leur poids respectifs et Q la dimension de l'espace.

Sous forme vectorielle, une droite est caractérisée par un point que nous appellerons la référence et que nous noterons R et un vecteur directeur que nous noterons $\overrightarrow{D}$. Le principe de l'algorithme est donc de trouver un point de référence correct puis de calculer la direction moyenne pondérée entre cette référence et tous les points du *fit*. Un choix évident pour le point de référence est le barycentre pondéré des points défini comme

$$R = \frac{\sum_{i=1}^{n} \tau_i P_i}{\sum_{i=1}^{n} \tau_i}. \tag{7.14}$$

Pour calculer $\overrightarrow{D}$, nous utilisons la même technique que celle utilisée pour la transformée de Hough. Nous construisons un histogramme des directions des couples de P_i et nous cherchons un pic pour trouver une direction privilégiée. Comme un seul pic est attendu et que l'erreur est supposée Gaussienne, le pic coincide avec la moyenne empirique des directions. Ceci conduit à une formulation analytique pour $\overrightarrow{D}$

$$\overrightarrow{D} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \tau_i \tau_j (P_i - P_j). \tag{7.15}$$

Comme la norme du vecteur directeur n'a pas d'importance, nous ne normalisons pas l'équation

pour simplifier la notation. L'ensemble est décrit par l'algorithme 1. Sa complexité est $O(n^2)$.

Variables: ε : array of orthogonal distance between the points and the fit line
$R = \frac{\sum_{i=1}^{n} \tau_i P_i}{\sum_{i=1}^{n} \tau_i}$;
$\overrightarrow{D} = \overrightarrow{0}$;
for $i \in 1..n$ **do**
  for $j \in i+1..n$ **do**
    $\overrightarrow{D} = \overrightarrow{D} + \tau_i \tau_j (P_i - P_j)$;
$\varepsilon_{i=1..n} = \| \overrightarrow{RP_i} - \frac{\overrightarrow{RP_i}.\overrightarrow{D}}{\|\overrightarrow{D}\|^2} \overrightarrow{D} \|$;
return $(R, \overrightarrow{D}), \varepsilon$;

Algorithm 1: Algorithme exact Φ_{line}

Le problème de cette complexité quadratique est qu'elle peut devenir vite limitante car l'algorithme Φ_{line} est une routine qui sera appelée très souvent lors d'un fit. Par conséquent, il est nécessaire de tenter de réduire cette complexité. C'est pourquoi je propose une heuristique nommée Refsplit. L'idée est de remplacer la partie quadratique, c'est à dire, le calcul de la moyenne des couples de points par la moyenne des couples reference-point. On prend la direction moyenne des vecteurs $\overrightarrow{RP_i}$. Pour trouver cette direction, on pourrait prendre naïvement la moyenne pondérée de ces vecteurs

$$\overrightarrow{D} = \sum_{i=1}^{n} \tau_i (P_i - R). \tag{7.16}$$

Mais cette somme est nulle par construction car R est le barycentre des P_i. Pour lever ce problème, nous utilisons le fait que la direction d'un vecteur est la même que la direction de son opposé. Ainsi, nous allons retourner certains vecteurs avant de calculer la moyenne. Pour cela, nous allons découper l'espace vectoriel en deux sous-espace complémentaires, séparés par un hyperplan. Tout les vecteurs d'une des deux parties resteront à l'identique tandis que les autres seront inversés. Par exemple, nous pouvont découper l'espace entre les vecteurs pour qui $x \geq 0$ et ceux pour qui $x < 0$. Nous inverserons ceux pour qui la composante est négative. Comme la somme des vecteurs initiaux est nulle, on sait que l'erreur induite par l'inversion d'un vecteur sera compensée par la somme vectorielle équivalente dans l'autre sous-espace. Nous pouvons définir notre vecteur directeur comme

$$\overrightarrow{D} = \sum_{i=1}^{n} \tau_i \delta(\overrightarrow{RP_i}) \overrightarrow{RP_i}. \tag{7.17}$$

La somme n'est plus nulle et le vecteur obtenu est une bonne approximation de celui obtenu par la méthode exacte à condition que l'hyperplan soit bien choisi. Pour cela, nous calculons un vecteur amplitude

$$\overrightarrow{A} = \sum_{i=1}^{n} | \overrightarrow{RP_i} |, \tag{7.18}$$

où $|\ .\ |$ est la valeur absolue, coordonnée par coordonnée. Nous choisirons l'hyperplan défini par la coordonnée qui a la plus grande valeur dans $\overrightarrow{A}$, et qui correspond à l'axe le plus proche du

vecteur directeur. Le fit basé sur l'heuristique Refsplit est décrit par l'algorithme 2. Sa complexité est $O(n)$.

Variables: a : preferred axis
ε : array of orthogonal distance between points and fit line
δ : reversion indicator

$R = \frac{\sum_{i=1}^{n} \tau_i P_i}{\sum_{i=1}^{n} \tau_i}$;
$\overrightarrow{A} = \sum_{i=1}^{n} \mid \overrightarrow{RP_i} \mid$;
$a = \max_{i=1}^{Q} A_i$;
for $i \in 1..n$ **do**
  if $\overrightarrow{RP_{ia}} < 0$ **then** $\delta_i = -1$;
  else $\delta_i = 1$;
$\overrightarrow{D} = \frac{\sum_{i=1}^{n} \tau_i \delta_i \overrightarrow{RP_i}}{\sum_{i=1}^{n} \tau_i}$;
$\varepsilon_{i=1..n} = \| \overrightarrow{RP_i} - \frac{\overrightarrow{RP_i}.\overrightarrow{D}}{\|\overrightarrow{D}\|^2} \overrightarrow{D} \|$;
return $(R, \overrightarrow{D}), \varepsilon$;

Algorithm 2: Algorithme Φ_{line} basé sur l'heuristique Refsplit

L'algorithme Φ_{line} retourne deux résultats : les paramètres de la droite fittée (R et $\overrightarrow{D}$) et un tableau de distances orthogonales entre les points et la droite. Cela permet que toutes les opérations liées à la géométrie soit confinées dans cet algorithme.

Une simulation en deux dimensions a démontré que l'erreur induite par l'heuristique Refsplit n'excède pas 7.10^{-6} pour le ratio D_x/D_y qui caractérise complètement la direction du vecteur. L'étude incluait un nombre variable de points et une exploration complète des angles (degré par degré) ainsi que des variations de la dispersion (1620000 cas). La moyenne des erreurs est de l'ordre de 10^{-11} avec un écart-type de l'ordre de 2.10^{-7}. Ces valeurs très basses (de l'ordre de la précision des calculs) montre que l'heuristique Refsplit ne dégrade nullement les performances de l'algorithme Φ_{line}.

7.7 Fit linéaire et isotrope en dimension arbitraire

Dans cette section, nous proposons une extension de l'algorithme EM, nommée GEM (pour generalized EM). C'est une amélioration du LRMM car cette version est complètement isotrope et permet de travailler en dimension arbitraire.

L'algorithme est basé sur quatre étapes. Tout d'abord, les poids τ_{ij} sont initialisés avec des valeurs aléatoires issues d'une loi uniforme et normalisées. Ensuite, L'algorithme GEM exécute en boucle les trois phases suivantes :

- Regression : calcul du fit pondéré par Φ_{line}
- Maximisation : calcul des paramètres de la Gaussienne d'erreur (basé sur les distances orthogonales fournies par Φ_{line}
- Expectation : calcul des poids des points (vraisemblance normalisée de la Gaussienne)

Les paramètres sont J, le nombre de composantes du mélange, λ, la vraissemblance Gaussienne normalisée et I_{ij}, la probabilité d'appartenance du point i à la composante j du mélange. La description de GEM est donnée à l'algorithme 3.

La convergence est décidée sur un critère de proximité entre les points et le fit. Pour chaque point P_i, nous définissons $prox(P_i)$, la droite la plus proche. Le critère de convergence est

$$S = \sum_{i=1}^{n} dist(P_i, prox(P_i)) \tag{7.19}$$

Dès que ce critère devient stable, l'algorithme est considéré comme convergé.

Variables: L : array of lines parameters
D : array of distances from point to line
τ : matrix of weights

for $j \in 1..J$ **do**
  for $i \in 1..n$ **do**
    τ_{ij} = normalized random;
repeat
  /* Regression step */
  for $j \in 1..J$ **do**
    $L_j, D_j = \Phi_{line}(p, \tau_j)$;
  /* Maximization step */
  for $j \in 1..J$ **do**
    $s_j^* = \sqrt{\frac{\sum_{i=1}^{n} \tau_{ij} D_{ij}^2}{\sum_{i=1}^{n} \tau_{ij}}}$;
  /* Expectation step */
  for $i \in 1..n$ **do**
    for $j \in 1..J$ **do**
      $l_{ij} = \lambda(d_{ij}, D_j, \tau_{ij})$;
      $\pi_j = \frac{\sum_{i=1}^{n} \tau_{ij}}{n}$;
      $\tau_{ij} = \frac{\pi_j l_{ij}}{\sum_{l=1}^{J} \pi_l l_{il}}$;
until *converged*;
return L, τ

Algorithm 3: GEM Algorithm

La figure 7.9 montre deux exemples d'utilisation de l'algorithme. Deux lignes de points affectées d'un bruit Gaussien sont ajustées par un mélange de deux régressions linéaires. A gauche, en 2 dimensions, Les droites rouges et vertes sont les droites résultats. Le dégradé de couleur de part et d'autre des droites est une représentation de la largeur de la Gaussienne autour de chacune des droites. A droite, la même chose en 3 dimensions. Comme attendu, l'algorithme permet de *fitter* les mélanges de régressions linéaires en dimension quelconque et sans anisotropie.

Deux limitations demeurent avec cet algorithme, en regard des exigences qui sont celles requises pour les trajectoires de particules. Le nombre de composantes doit etre déterminé par un algorithme extérieur et d'autre part, l'algorithme reste très sensible à son initialisation (ceci est hérité de l'EM).

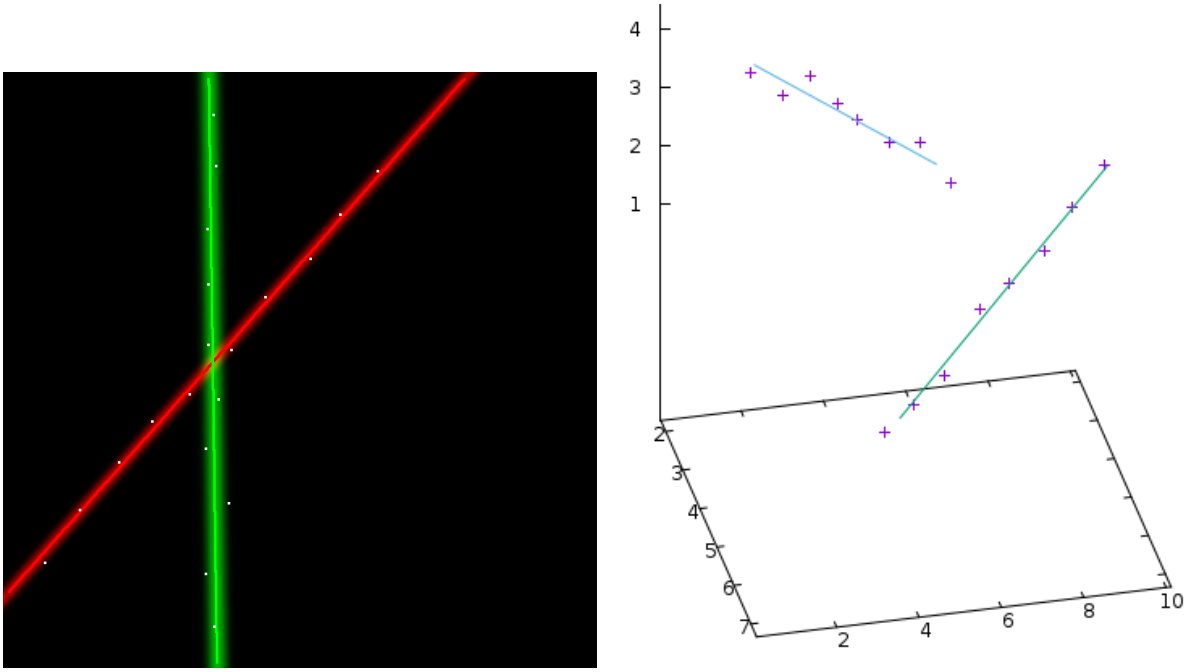


FIGURE 7.9 – Résultat de l'algorithme GEM en 2D et en 3D sur deux lignes de points bruitées

7.8 Fit linéaire adaptatif en dimension arbitraire

Dans cette section, nous proposons un algorithme qui permet de lever ces deux limitations. L'idée est de partir avec une seule droite et donc d'initialiser l'algorithme avec une simple régression linéaire non pondérée puis d'ajouter des droites au fur et à mesure si nécessaire. Les droites successives ne seront pas ajoutées au hasard mais seront centrées sur les points les plus mal *fittés*. Ainsi le problème de l'initialisation sera réglé en même temps que celui du nombre de composantes. Deux détails restent à régler : comment choisir la nouvelle droite et comment décider si une nouvelle droite est nécessaire ?

Pour choisir la nouvelle droite, nous commencerons par chercher le point, nommé wfp, qui est le plus mal ajusté sur une trajectoire, c'est-à-dire le point qui a la plus grande valeur de $dist(P_i, prox(P_i))$. On va ensuite construire un voisinage de ce point à distance fixée. Nous utilisons ensuite ce voisinage pour construire la nouvelle droite. Pour cela, nous utilisons GEM avec des poids adaptés. Les points choisis reçoivent un poids proche de $1 - 10^5$ et les autres un poids de 10^5 (nous évitons 0 et 1 qui provoquent de problèmes numériques). Il faut naturellement renormaliser les poids en conséquence. Une fois les poids ré-affectés, les droites sont recalculées avec les nouveaux poids.

Pour décider si une nouvelle droite est nécessaire, nous comparons l'écart type de la distribution d'erreur autour de chacune des droites avec un critère d'échelle. Celui-ci est défini comme une fraction de la diagonale de l'espace formé par les données. Ce critère d'échelle permet de sélectionner un fit plus ou moins précis. En conséquence, il va piloter la résolution angulaire du

fit, comme le montre le graphe de gauche de la figure 7.10. MFit est décrit par l'algorithme 4.

Parameters: c : convergence criterion
s : scale criterion
Variables : D : array of parameters of shape S
G : array of Gaussian distributions
τ : fit weights
J : number of fit lines
converged : Boolean indicator of convergence

$D_1 = \Phi_S(P, [1/n, 1/n, ..., 1/n])$;
$G_1 = error(D_1, P, \tau)$;
$J = 1$;
repeat
  $wfp, prox_{wfp} = worst\_fitted(P_i, D)$;
  $J = J + 1$;
  /* Weights for new line */
  for $P_i \in P$ **do**
    if $d(wfp, p) < prox_{wfp}$ **then**
      $\tau_{iJ} = 1 - 10^{-5}$
      for $j \neq J$ **do**
        $\tau_{ij} = 10^{-5}/(J-1)$
    else
      $\tau_{iJ} = 10^{-5}$
      for $j \neq J$ **do**
        $\tau_{ij} = \tau_{ij} - 10^{-5}/(J-1)$
  /* Apply GEM and determine convergence */
  $D, G, \tau = GEM(J, P, \Phi_S)$;
  $converged = True$;
  for $j \in 1..J$ **do**
    if $G_j.\sigma > s/D$ **then**
      $converged = False$
until *converged*;
return D, G, τ, J;

Algorithm 4: MFit Algorithm

L'algorithme MFit est naturellement résistant au bruit uniforme. Sur la partie droite de la figure 7.10, on peut voir une courbe représentant le taux d'erreur sur le nombre de composantes en fonction du bruit uniforme. Nous pouvons voir que le bruit ne pose aucun problème jusqu'à 15% et qu'il devient rapidement insurmontable au delà de 20%. En effet, plus ce pourcentage augmente, plus le système va avoir tendance à trouver des corrélations linéaires dans le bruit.

L'isotropie a également été testée exhaustivement. Pour cela, on prend un nuage de points *fitté* par deux droites. On fait alors pivoter les points sur 360 degrés, par pas de 1 degré. A chaque rotation, on applique l'algorithme MFit et on vérifie que le résultat est le même à la rotation près. Le résultat obtenu est toujours exactement celui attendu. Comme prévu par le modèle, l'algorithme est complètement isotrope.

L'algorithme MFit étant purement additif, il arrive que certaines droites soient ajoutées au

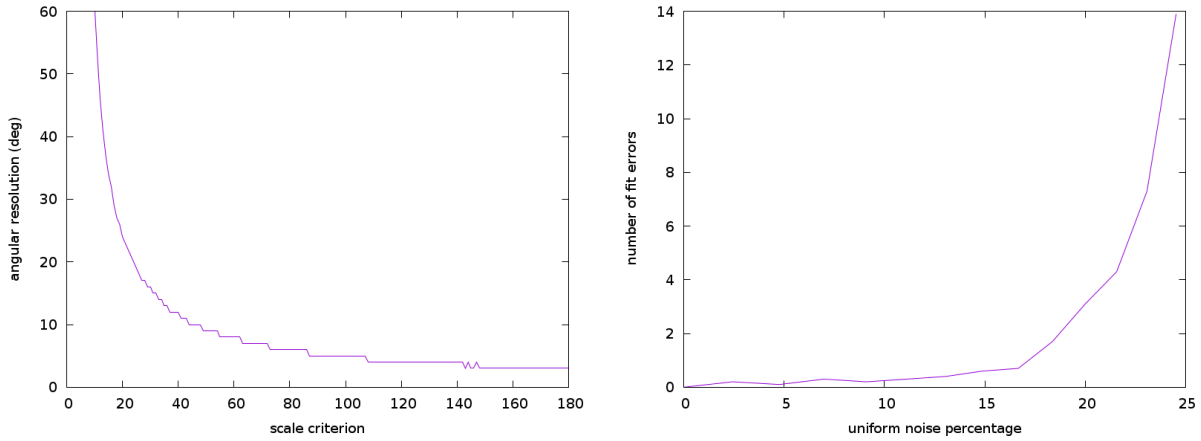


FIGURE 7.10 – Résolution angulaire en fonction du critère d'échelle (à gauche) et résistance de l'algorithme au bruit uniforme (à droite).

mélange dont elles représentent une composante à une étape donnée. Par la suite, pour des raisons numériques, elle ne disparaitront pas forcément et resteront dans le mélange. Il convient de supprimer de telles droites que nous appellerons dégénérées. Fort heureusement, elles sont très faciles à détecter, en effet, elle ne sont la droite la plus proche que de très peu de points. Un simple comptage de complexité linéaire comme représenté dans l'algorithme 5 permet de les supprimer.

```
1 for D_j ∈ D do
2     c = 0;
3     for P_i ∈ P do
4         if D_j = prox(P_i) then
5             c = c + 1
6     if c < s then
7         Remove(D_j)
```

Algorithm 5: Algorithme de suppression des droites dégénérées

Nous avons effectué une simulation afin de mesurer la performance globale de l'algorithme et l'intéret de la suppression des droites dégénérées. Des mélanges de droites non-colinéaires de différentes tailles ont été générées pour un espace à deux dimensions (2000 mélange par taille). Pour chacun d'entre eux, nous avons effectué un fit avec l'algorithme MFit. Sur la figure 7.11, nous pouvons voir les performances obtenue comme un poucentage d'erreur sur la taille du fit obtenu. Nous constatons que sans post-processing (courbe violette), la performance est très rapidement dégradée par les droites dégénérées. Avec le post-processing qui les supprime (courbe verte), la performance est bien meilleure. Une étude qualitative a montré que certaines droites pouvaient être dupliquées. Ces dernières peuvent être supprimée par un post-processing spécifique dont la complexité est $O(J^2)$, où J est le nombre de composante du mélange. La courbe bleue montre l'efficacité de ce nouveau post-processing. Le gain est constant et n'améliore donc pas sensiblement la performance. C'est pourquoi dans les applications, nous utilisons toujours la suppression des droites dégénérées mais pas celle des droites dupliquées.

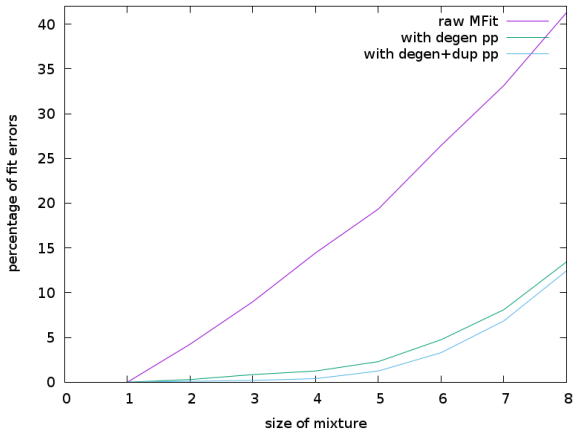


FIGURE 7.11 – Performance de l'algorithme MFit avec differents post-processings.

7.9 Applications de MFit

Tous les algorithmes décrits ici ont été implémentés dans une librairie spécifique appelée "libgem". Elle est open-source et peut être téléchargée sur github à l'addresse *https ://github.com/fredllr/libgem*.

Elle a été écrite en C pour obtenir la meilleure performance possible. Des connecteurs pour d'autres langages seront disponibles prochainement. Des programmes en ligne de commande permettent de l'utiliser sur des fichiers csv. Leur utilisation est documentée sur le site github. La librairie peut également être interfacée avec d'autres frameworks d'analyse (en particulier Root). L'API est simple : un dataset doit être rempli avec les coordonnées des points, puis les différents algorithmes peuvent leur être appliqués. La librairie fournit des outils de visualisation en 2D et en 3D comme on peut le voir sur les figures de fit de ce chapitre (hormis les fits LRMM qui sont produits avec R et mixreg).

7.9.1 Données Harpo

Nous pouvons maintenant ré-analyser les données d'harpo avec l'algorithme MFit. Nous prenons toujours les données réduites par la grille barycentrique pour limiter le nombre de points. Nous testons tout d'abord le cas le plus fréquent, où l'interaction entre le photon et le gaz produit une paire peu déviée par la diffusion multiple (l'intéraction majoritaire aux énergies considérées).

La figure 7.12 présente ce cas avec deux angles d'incidences. On voit à droite que la trace verticale n'est plus un problème. Les droites sont *fittés* correctement et la largeur de distribution (le halo autour des droites) est bien proportionnelle en largeur à la dispersion des données. C'est particulièrement évident à droite ou la trace verticale est beaucoup plus dispersée que la trace horizontale. On voit que l'intersection des droites donne une bonne approximation du vertex (le point d'interaction du photon incident avec un noyau), qui est essentiel à la reconstruction de la trace en 3D.

La figure 7.13 montre deux autres cas. A gauche, il s'agit d'une trace unique, probablement produite par une diffusion Compton. A droite, on voit le cas où la diffusion multiple a tellement perturbé la trajectoire de l'électron qu'il a complètement changé de direction. L'algorithme a ajouté une droite.

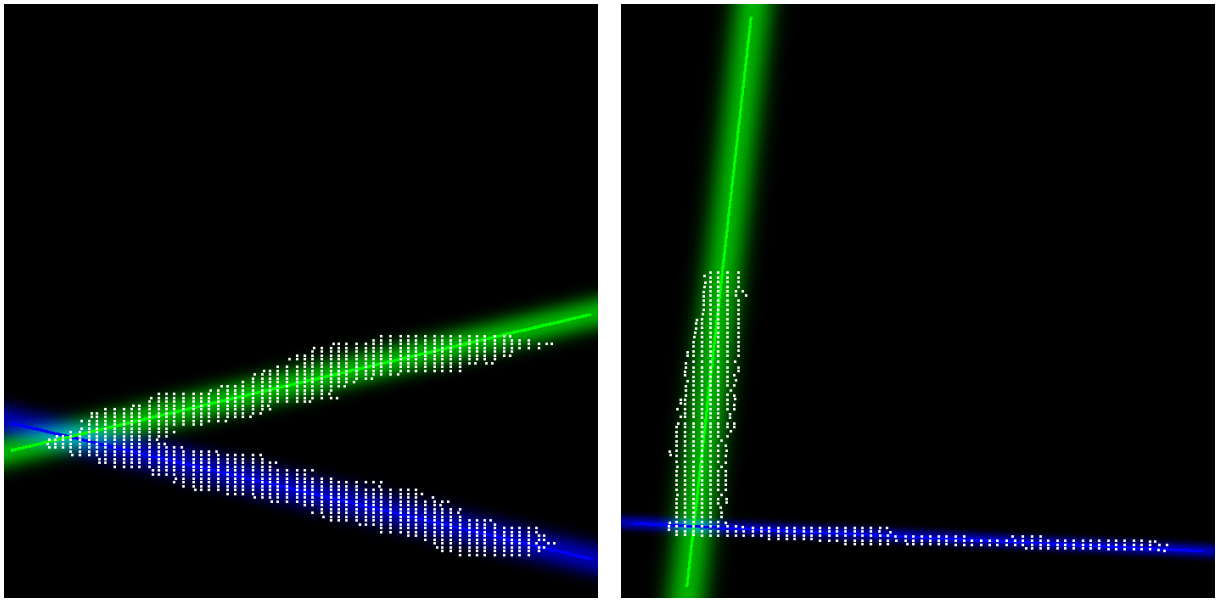


FIGURE 7.12 – Fit d'une trace typique de Harpo en position ordinaire ou verticale

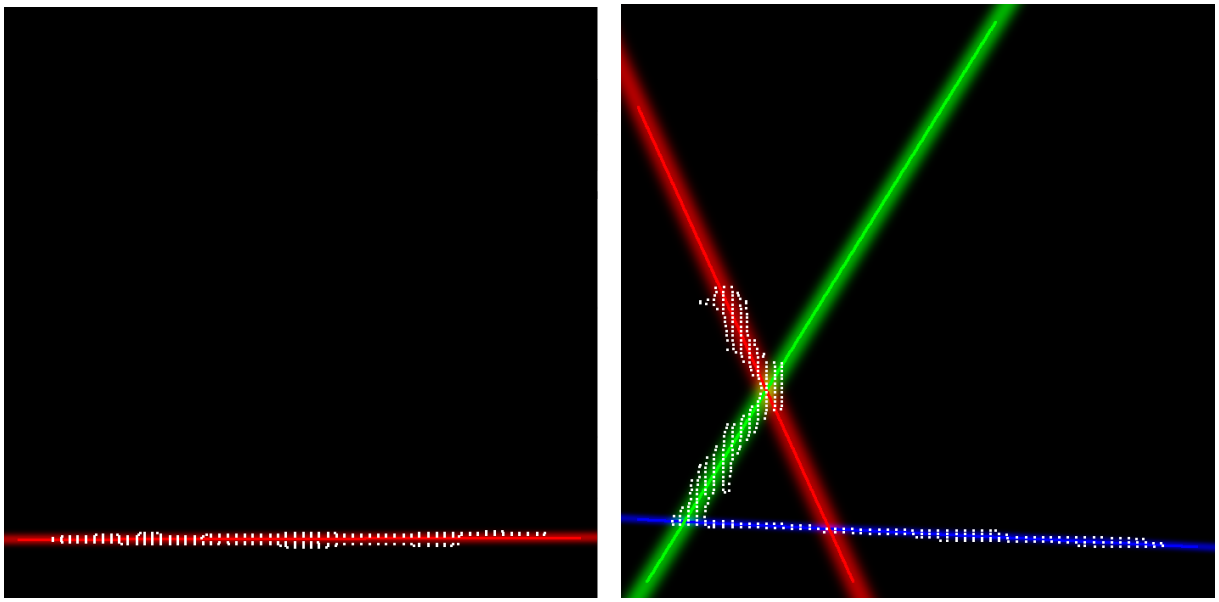


FIGURE 7.13 – *Fit* d'une diffusion Compton et d'une création de paire avec diffusion multiple

Comme expliqué dans le chapitre 3.1, les traces recuillies dans la *TPC* vont toujours par deux, l'une pour l'axe des x en fonction du temps, et l'autre pour l'axe y en fonction du temps. Pour déterminer la nature de l'interaction (Compton ou paire), on procède au fit des deux mesures et on compte les droites obtenues :

- Si on obtient 1 pour x et pour y, il s'agit d'un Compton.
- Si on obtient 1 pour x et de 2 pour y ou l'inverse, il s'agit d'une paire projetée sur un axe.
- Dans le cas ou le nombre de droites est 2 pour x et pour y, on a une paire avec peu de diffusion.

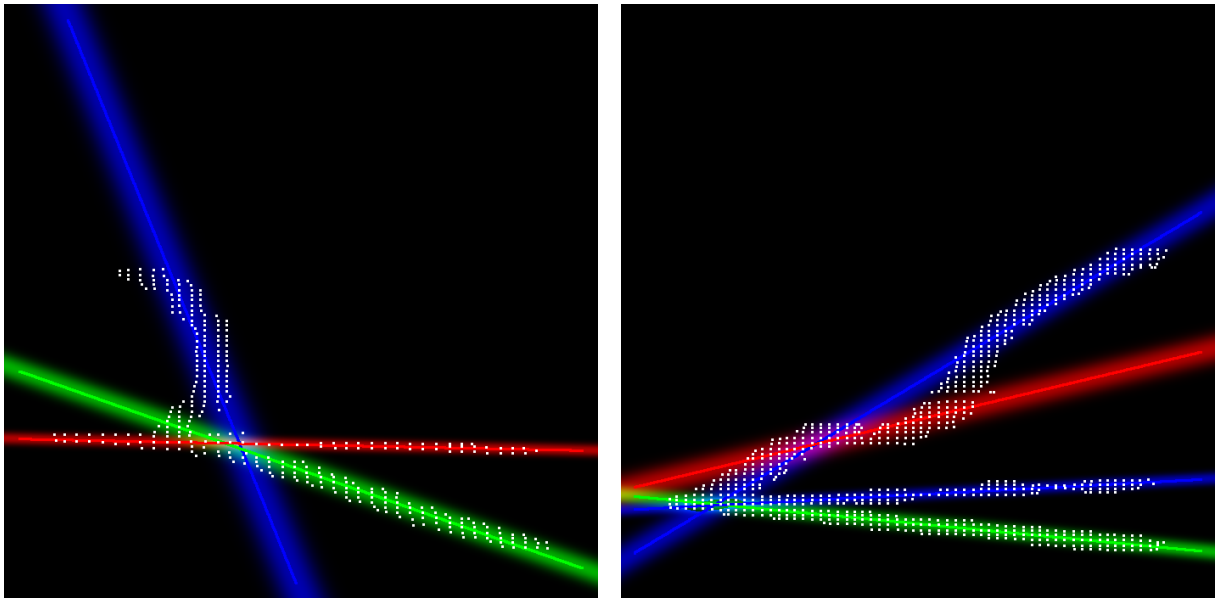


FIGURE 7.14 – *Fit* de triplets avec plus ou moins de diffusion multiple

La figure 7.14 montre des traces de triplets avec plus ou moins de diffusion multiple. A gauche, la diffusion a presque retourné la trace la plus haute. On voit que le *fit* proposé est presque similaire à celui de la trace précédente si ce n'est que dans le cas du triplet, les trois points de contact entre droites sont quasiment confondu. C'est ce qui permet de discriminer ce cas de la paire avec diffusion importante. Enfin à droite, le triplet a lui-même été perturbé par la diffusion, ce qui aboutit à un *fit* à quatre droites. Le fait d'avoir trois points de contacts groupés permet toutefois de conclure à un triplet.

On a donc vu que l'application de MFit sur les données de Harpo permettent d'améliorer le classement des mesures. Il permet de calculer le vertex d'interaction pour les paires qui ne sont pas trop perturbées par la diffusion ou pour les triplets même perturbés. En revanche, pour les paires avec diffusion, il est très délicat de trouver le vertex sans faire de supposition additionnelle. Toutefois, comme ce cas est assez rare, on peut couper ces données ou les traiter avec un autre algorithme.

7.9.2 Données Wagasci

Nous l'avons vu, Wagasci est un détecteur de neutrino basé sur une structure de croisillons de plastique scintillant. La reconstruction se fait en trois étapes dont la première est une régression linéaire multiple. Naturellement, c'est un candidat idéal pour l'utilisation de MFit. Ce travail a été réalisé avec un étudiant de l'école Polytechnique, Philipp Windischhofer que j'ai co-encadré avec Thomas Mueller.

Pour utiliser libgem dans le contexte de Wagasci, nous avons développé un petit *framework* en ROOT permettant de transformer les données des simulations en dataset libgem. Comme nous pouvons le voir sur la figure 7.15, les données correspondent en forme aux scintillateurs. Pour fabriquer les points du *fit*, on prend le barycentre de chacun d'entre eux. L'erreur ainsi induite est compensée par la longueur des traces qui est suffisante pour obtenir des fits de bonne qualité.

L'algorithme donne des résultats satisfaisant sur les données comme on peut le voir qualitative-

ment sur les figures 7.15 et 7.16. Par contre, la dispersion importante des points due à la largeur des scintillateurs a tendance à favoriser des mélanges avec plus de droites que nécessaire (droites dégénérées). C'est pourquoi, nous avons du augmenter le seuil du post-traitement afin de les supprimer.

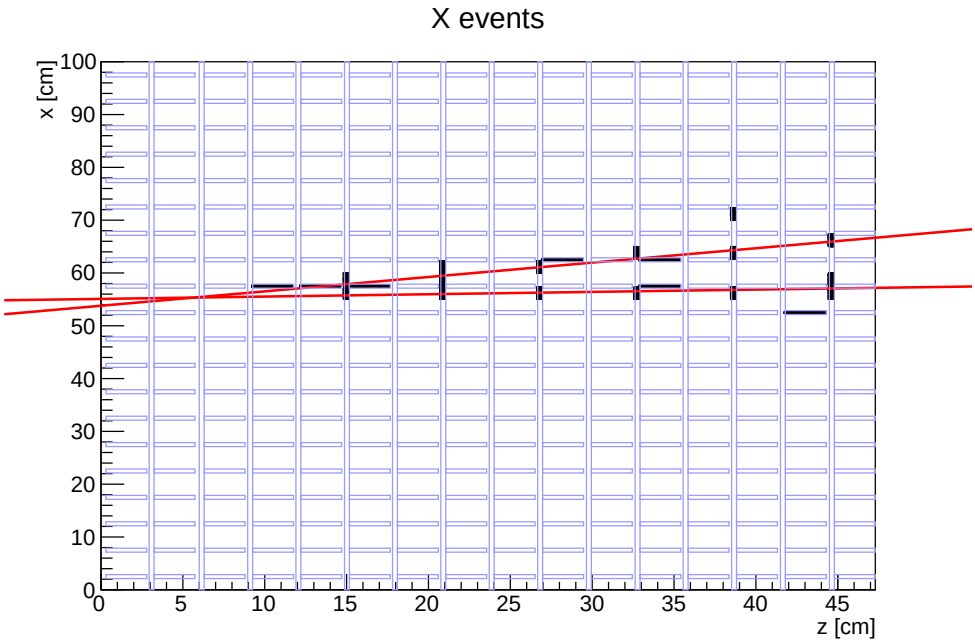


FIGURE 7.15 – Traces en x *fittées* avec libgem. (Credit P. Windischhofer)

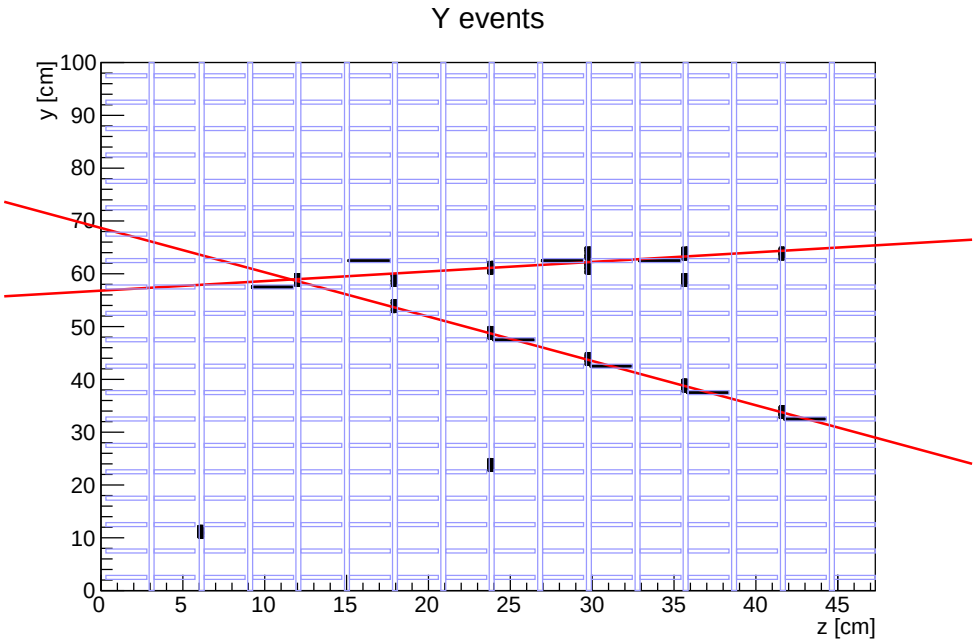


FIGURE 7.16 – Traces en y *fittées* avec libgem. (Credit P. Windischhofer)

Jusque là, la reconstruction de ces traces étaient effectuées avec un algorithme basé sur un automate cellulaire. Afin de pouvoir comparer les deux méthode, nous avons utilisé un jeu

de données de simulation qui a été analysé avec la méthode des automates cellulaires et avec l'algorithme MFit. La première constatation concerne le temps d'exécution. Plus de 90 minutes pour l'automate cellulaire, contre moins de 10 minutes pour MFit, soit près d'un facteur 10. En effet, la librairie, programmée en C est très efficace et les données étant assez propres, l'algorithme évite les cas difficiles avec un grand nombre d'objets.

D'autre part, les méthodes ont été comparées quantitativement par rapport au nombre de traces et par rapport à la distance au vertex vrai (on le connait car il s'agit de simulations). La figure 7.17 présente cette comparaison. A gauche, on voit le nombre de traces obtenue avec les deux méthodes. On voit que MFit trouve plus de double traces que l'automate, ce qui signifie qu'il discrimine mieux les traces doubles intriquées malgré un paramètre d'échelle modéré (30). A droite sur la figure, on voit la distance au vertex vrai pour les deux méthodes. On constate que les deux méthodes donnent des résultats similaires avec des variations statistiques assez importantes. De nouvelles études de performance plus complètes sont d'ailleurs en cours pour préciser davantage ces différences.

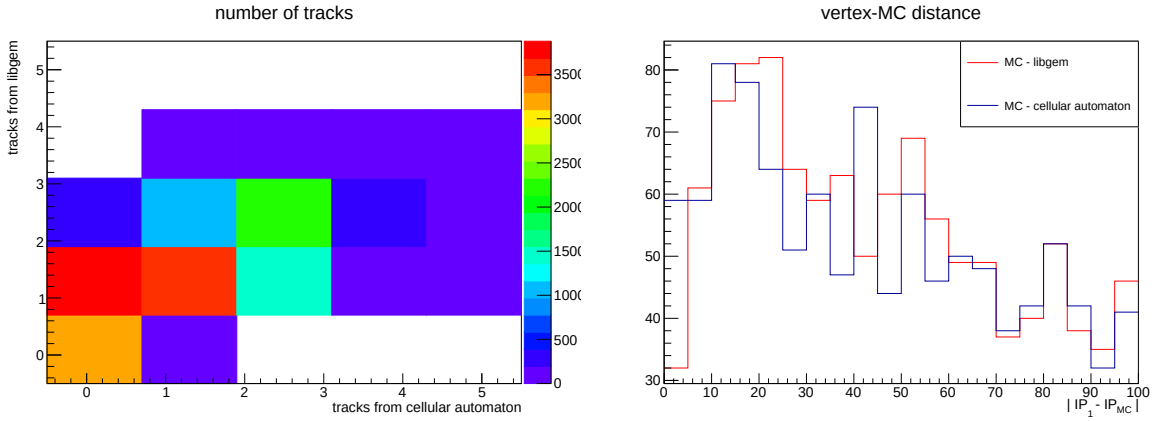


FIGURE 7.17 – Comparaison des deux méthodes de *fit* : automate cellulaire et MFit. MFit est légèrement meilleur pour discriminer les traces doubles fortement intriquées. La performance sur l'évaluation du vertex est équivalent. Mfit est environ 10 fois plus rapide que l'automate. (Credit P. Windischhofer)

Pour résumer, l'algorithme MFit est parfaitement adapté à la problématique de reconstruction des traces de Wagasci. Par rapport à la méthode des automates cellulaire, les résultats sont similaires en terme de précision, avec un pouvoir de discrimination légèrement meilleurs sur les traces doubles intriquées et avec un gain de performance de l'ordre de 10.

7.9.3 *SiW-Ecal* en mode *trajectographe*

Nous l'avons vu dans un chapitre précédent, *SiW-Ecal* peut fonctionner comme un *trajectographe* s'il n'est pas garni de ses absorbeurs en tungstène. Sa haute granularité permet d'obtenir des traces fines, permettant de calculer des trajectoires assez précises. Naturellement, l'algorithme MFit est parfaitement adapté pour fitter ces trajectoires.

Comme l'afficheur 3D décrit à la section 5.4.13, ce système est complètement *online*. La chaine d'acquisition lance les décodeurs *online* qui sont la source de données pour l'event-builder. Lui-même fournit les données agrégées à tous les programmes qui souhaitent les analyser. Le programme de trajectographie va donc se connecter sur l'event-builder et récupérer les coordonnées

des nuages de points.

Ce programme se base sur la libgem et sur l'event-loop de Pyrame pour construire les datasets et les *fitter* avec MFit. Il utilise la partie graphique de libgem, basée sur gnuplot, pour afficher ses résultats.

La plupart des traces que l'on obtient sur un faisceau d'électrons comme celui de DESY sont des traces simples correspondant à une particule unique. En mode *trajectographe*, ces traces sont essentiellement droites comme on le voit à gauche sur la figure 7.18. Dans une proportion non négligeable des cas, la particule gerbe un peu. comme on peut le voir à droite sur la figure. On peut constater que l'algorithme MFit tolère bien ce bruitage.

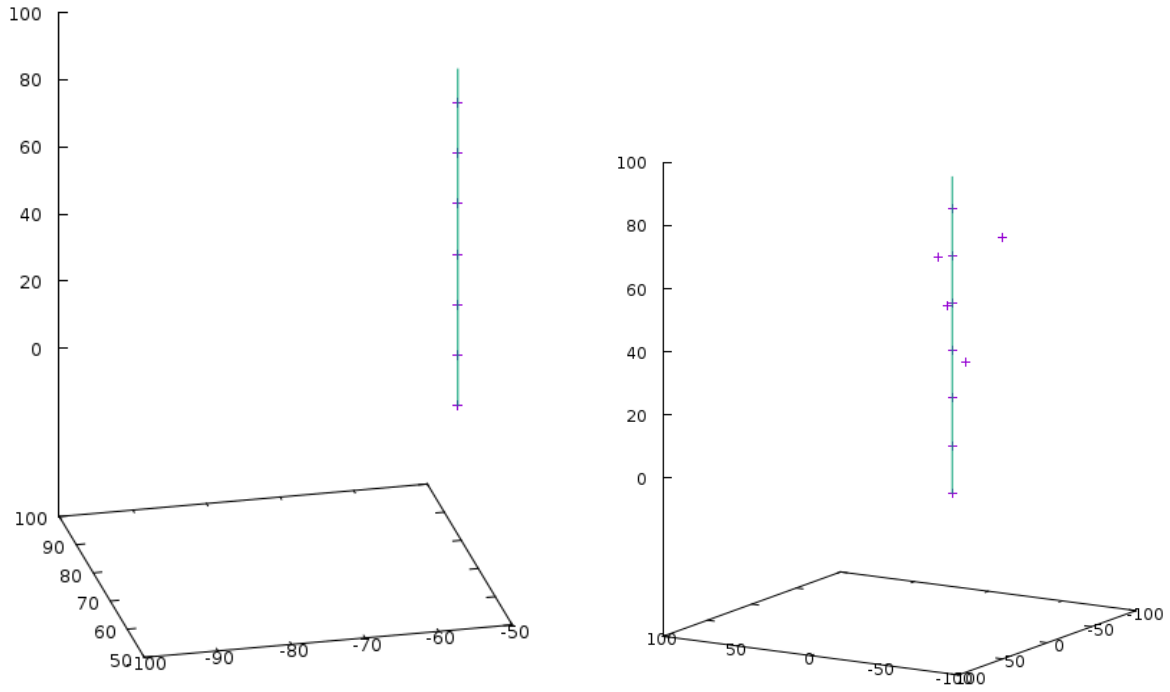


FIGURE 7.18 – Trace *SiW-Ecal* unique *fittée* avec MFit : le faisceau rentre dans le détecteur par la couche de coordonnée z=0 (celle du bas). La trace est strictement linéaire à gauche (interaction minimale avec la matière), un peu gerbée à droite à partir de la troisième couche.

Dans une certaine fraction des cas (de l'ordre de 10% au DESY), du pile-up apparait. Comme on le voit sur la figure 7.19, l'algorithme permet de répondre correctement à ces cas, même lorsque les traces sont très proches et même en présence d'un peu de bruit de lecture.

En revanche, dans les cas où des traces voisines commencent à former une gerbe, il est très délicat de les distinguer. Ainsi, à gauche sur la figure 7.20, il est difficile de déterminer s'il s'agit de trois traces ou de deux traces gerbées. A droite, on voit un évènement problématique où l'électronique génère ce que l'on appelle un évènement plein (plane event), c'est-à-dire un emballement du système de *trigger* qui provoque un hit généralisé dans tous les canaux. Dans ce cas, quatre *ASICs* ont fait un évènement de ce type, ce qui est rare. On voit que dans ce cas, l'algorithme tente un *fit* difficile qui est très couteux en temps de calcul. En effet, celui-ci dépend du nombre de points du *fit* mais aussi du nombre d'objet. Ainsi, ce type d'évènement provoque un ralentissement considérable dans le flux et doit être filtré en amont. C'est pourquoi les décodeurs *online* proposent cette option.

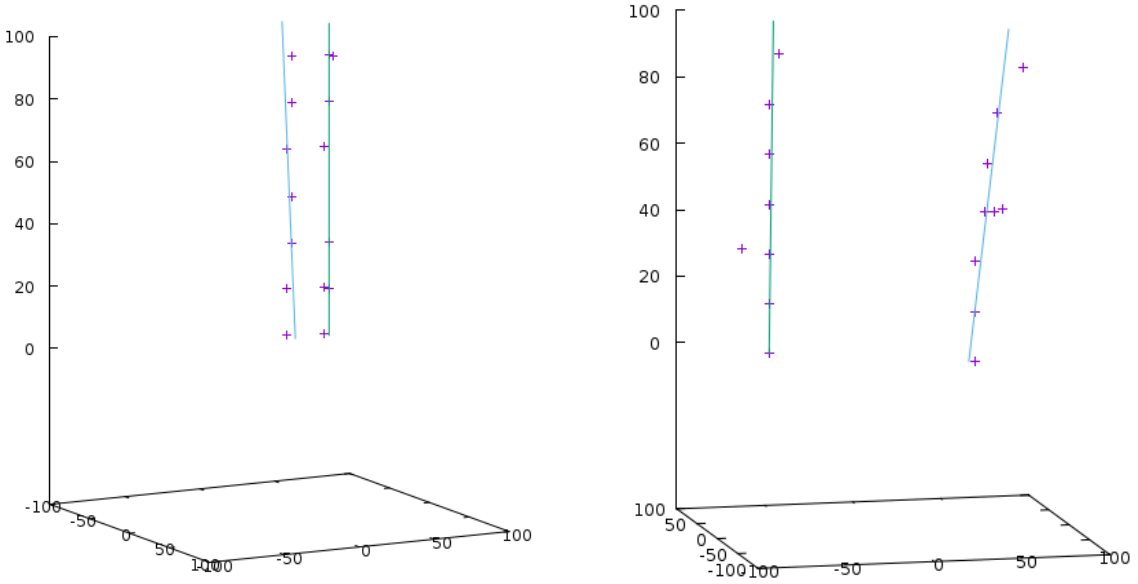


FIGURE 7.19 – Doubles traces *SiW-Ecal fittées* avec MFit

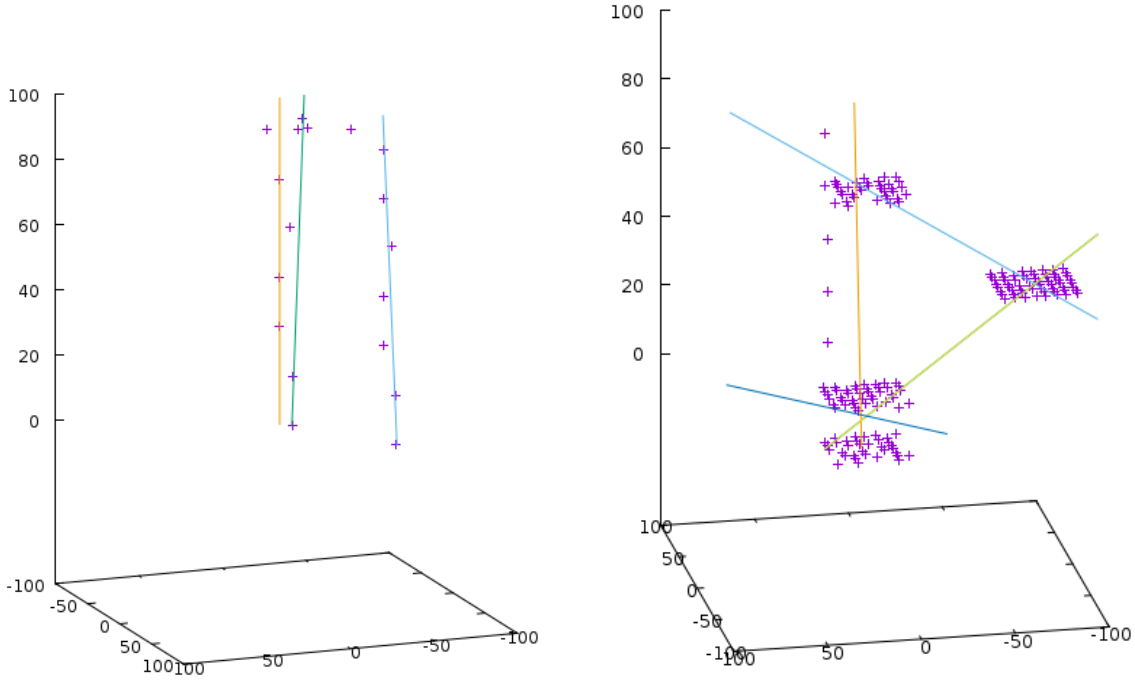


FIGURE 7.20 – Traces SiW-ECal triples ou gerbées à gauche, 4 evènements pleins à droite

7.10 Extension à d'autres formes

Comme nous l'avons déjà mentionné, rien dans l'algorithme GEM n'utilise le fait que les traces soient linéaires. Toute la partie géométrique est dans l'algorithme Φ_{line}. Nous pouvons en déduire une propriété importante de l'algorithme

Propriété 1 *Pour toute forme S, si un algorithme Φ_S existe, fournissant un fit pondéré pour la forme S ainsi qu'une distance orthogonale, GEM peut fitter un mélange de la forme S.*

De fait, GEM est une extension de l'algorithme EM pour toute forme dont on peut calculer un fit pondéré, c'est pourquoi il s'appelle "Generalized EM". En fait, c'est un opérateur algorithmique qui permet de construire automatiquement un algorithme de fit multiple à partir d'un algorithme de fit pondéré.

Pour démontrer cette aptitude, nous en proposons une version effectuant des fits circulaires. En effet, le cercle est la figure que l'on trouve le plus fréquemment dans nos détecteurs après les droites. Ces figures correspondent aux trajectoires de particules chargées dans un champ magnétique uniforme. C'est d'ailleurs un outil puissant pour identifier les particules qui constituent une trace car le sens de rotation indiquera le signe de la charge et le rayon du cercle indiquera l'impulsion de la particule. Il est donc très intéressant d'avoir à notre disposition un algorithme d'ajustement des trajectoires circulaires.

Le problème consistant à fitter un cercle est plus compliqué que celui du fit linéaire car il n'existe pas de forme analytique. Il est nécessaire d'utiliser un algorithme itératif pour explorer l'espace décrit par la fonction d'erreur pondérée et y trouver un minimum. La stratégie habituelle pour une telle recherche est d'utiliser un algorithme de recherche linéaire sur la fonction d'erreur pondérée définie par

$$\varepsilon(C) = \sum_{i=1}^{n} \frac{\tau_i(\bar{d} - d(C, P_i))}{\sum_{j=1}^{n} \tau_j}, \tag{7.20}$$

où C le centre estimé du cercle, les P_i sont les points du fit et les τ_i sont leurs poids respectifs. La distance moyenne entre les P_i et C est notée $\bar{d}$. Comme on le voit sur la gauche de la figure 7.21, cette fonction a une forme régulière pour un cercle complet avec un seul minimum et une pente raide, ce qui rend facile la recherche de l'optimum. Si le cercle est incomplet (à droite sur la figure), la fonction reste régulière et possède toujours un minimum à condition de disposer d'au moins trois points pour le fit.

La méthode traditionnelle pour trouver le minimum d'une telle fonction est d'utiliser un algorithme à direction de descente, basé sur le gradient de la fonction d'erreur, comme par exemple celui de Newton-Raphson [28]. Les valeurs des coordonnées de C sont évaluées itérativement en utilisant une dérivée partielle de la fonction d'erreur. L'inconvénient de cet algorithme est que le calcul des matrices de dérivées partielles est très gourmand en temps de calcul. C'est pourquoi nous proposons une heuristique. Au lieu de calculer la direction de descente avec une dérivée, nous utilisons la direction des points vers le centre précédent.

Au départ, le centre estimé est initialisé au barycentre pondéré des points. Puis à chaque étape, un vecteur d'erreur est calculé pour chaque point, dirigé vers le centre et proportionnel à l'erreur pondérée. Ensuite, la direction de descente et l'amplitude sont calculées en même temps, par la somme vectorielles des vecteurs d'erreurs individuels. Cela donne un nouveau centre estimé pour la prochaine itération. On stoppe l'algorithme quand la position du centre estimé devient stable. La figure 7.22 montre les positions successives du centre estimé du cercle sur la fonction d'erreur. Les étapes successives sont décrites dans l'algorithme 6.

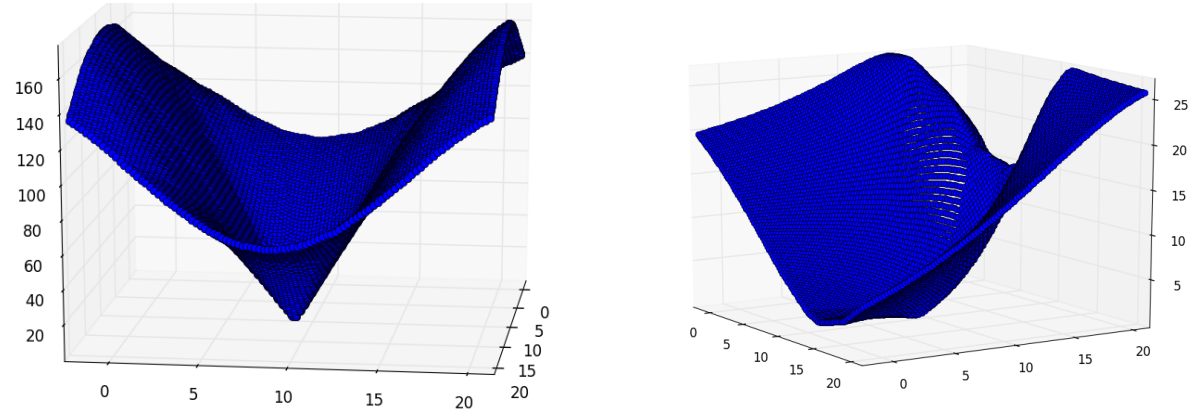


FIGURE 7.21 – Fonction d'erreur pondérée pour un cercle formé de 20 points - Fonction d'erreur pondérée pour un quart de cercle formé de sept points.

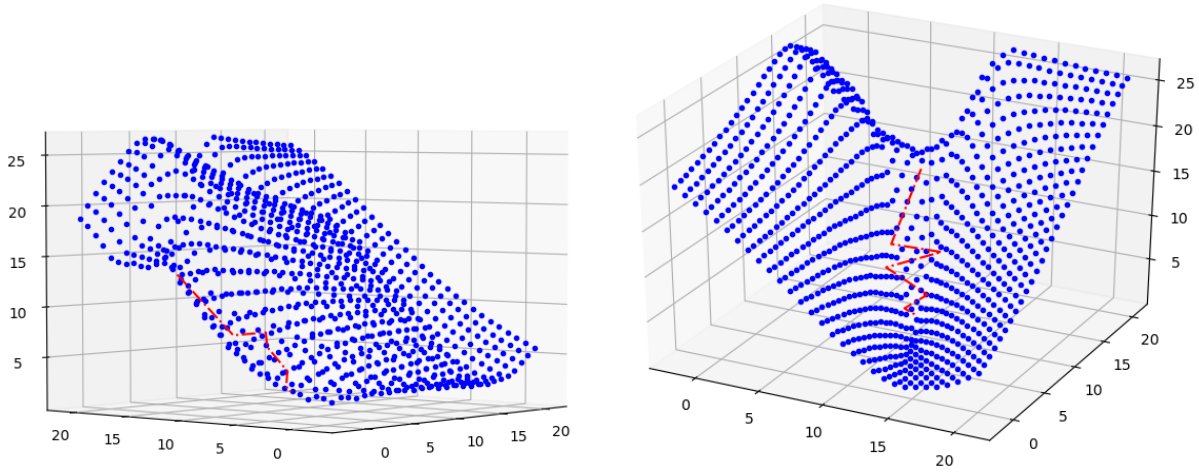


FIGURE 7.22 – Estimations successives du centre estimé sur une fonction d'erreur correspondant à sept points disposés en quart de cercle. Deux angles de vue sont proposés.

Variables : C : center of the circle
r : radius of the circle
$\overrightarrow{M_i}$: move vectors
ε : array of distance between points and circle

$C = weighted\_barycenter(P, \tau)$;
repeat
  $\bar{d} = \frac{\sum_{i=1}^{n} ||\overrightarrow{P_iC}||}{n}$;
  for $i \in 1..n$ **do**
    $\overrightarrow{M_i} = \frac{\overrightarrow{P_iC}}{||\overrightarrow{P_iC}||} \sum_{i=1}^{n} \frac{\tau_i(\bar{d} - ||\overrightarrow{P_iC}||)}{\sum_{i=1}^{n} \tau_i}$;
  $C = C + \sum_{i=1}^{n} \overrightarrow{M_i}$;
until *center stable*;
$r = \frac{\sum_{i=1}^{n} \tau_i ||\overrightarrow{CP_i}||}{\sum_{i=1}^{n} \tau_i}$;
for $i \in 1..n$ **do**
  $\varepsilon_i = d(P_i, C) - r$;
return $(C, r), \varepsilon$;

Algorithm 6: Φ_{circle} algorithm

Naturellement, cet algorithme est compatible avec MFit, ce qui permet de faire un fit multiple et adaptatif de traces circulaires. La figure 7.23 montre deux exemples d'un tel fit.

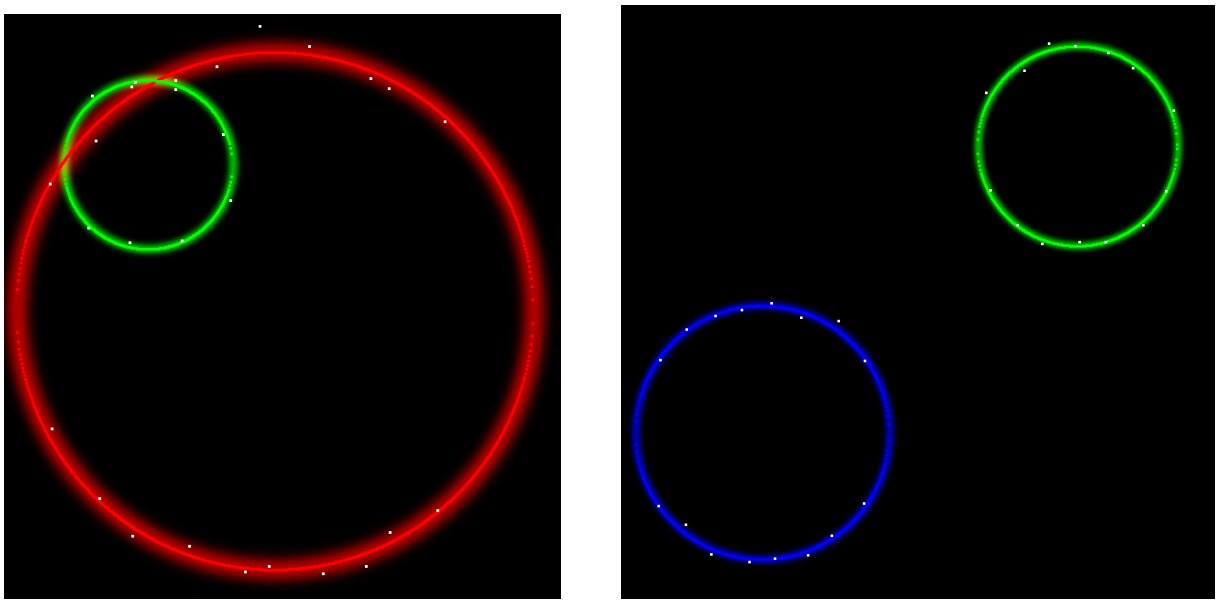


FIGURE 7.23 – Fit circulaire et adaptif de deux cercles séparés ou intriqués avec les algorithmes MFit et Φ_{circle}.

7.11 *Fits* combinés

Une fois qu'on dispose de deux type de *fit*, il est tentant de vouloir les combiner. Dans GEM/MFit, rien ne distingue Φ_{line} ou Φ_{circle} et on peut imaginer d'appliquer indifféremment l'un ou

l'autre et de comparer les fonctions d'erreur comme si les objets étaient de même nature. La seule difficulté est d'essayer alternativement les deux formes pour trouver le bon mélange de droites et de cercles. Pour cela, je propose un algorithme, baptisé UFit et qui effectue cette tâche en construisant un arbre de combinaisons et en comparant les solutions entre elles. C'est un algorithme assez naïf dont le but est de démontrer la possibilité de mixer les différentes formes entre-elles plutôt que de proposer une implémentation efficace.

Dans cet algorithme, on ne considère plus des droites ou des cercles mais simplement des objets génériques dont on peut calculer le *fit* pondéré et la distance orthogonale. Pour ne pas s'interdire plus tard, d'ajouter d'autres formes, l'algorithme est générique et considère un nombre quelconque de types d'objets.

Un système de scoring permet de comparer les différents essais réalisés par l'algorithme. Ce score est obtenu en sommant pour tous les points, la distance de ce point à son objet le plus proche. La formule du score est donc $Score = \sum_{i=1}^{n} dist(P_i, prox(P_i))$. C'est en général un bon indice qu'un *fit* est correct.

L'algorithme va essayer des mélanges successifs en commençant par une droite, puis un cercle... A chaque essai, il va calculer le score et comparer avec le meilleur score obtenu jusque là. Si le score est le meilleur obtenu jusque là et que tous ses objets satisfont le critère d'échelle, le mélange est déclaré solution et l'algorithme s'arrête. Dans les autre cas, on continue l'exploration, c'est-à-dire qu'on va ajouter un objet de chaque type après tous les cas déjà explorés. On va donc explorer [L] (1 droite), [C] (1 cercle), puis [LL] (2 droites), [LC], [CL], [CC] puis [LLL] [LLC] [LCL] [LCC]... Il est nécessaire d'explorer aussi bien [LC] que [CL] car l'algorithme n'est pas forcément commutatif (le point le plus mal fitté est très différent si l'on a commencé par une droite ou par un cercle).

L'algorithme UFit donne de très bons résultats lorsque le nombre d'objets à *fitter* est faible, typiquement jusqu'à 8. Il permet de trouver la meilleure solution en explorant au plus 2^k où k est la taille du mélange solution. Au delà d'une dizaine de composantes dans le mélange, le nombre de cas à explorer devient trop grand et un autre algorithme, incluant des coupures doit être envisagé.

Afin de tester cet algorithme dans des conditions proches de nos problématiques, nous avons écrit un code de simulation en Géant4 qui permet de le tester sur un cas concret. Des électrons de 1 GeV sont envoyés sur un petit cube de tungstène de 50 microns dans un champ magnétique de 0.3 Teslas. Un début de gerbe électromagnétique commence à se former dans le tungstène, puis les particules qui la composent ressortent du métal avec des énergies diverses, en ligne quasi-droites ou en trajectoire courbée en fonction de leur énergie. Grâce à cette simulation, nous avons à notre disposition de nombreux exemples qui permettent de tester la pertinence de l'algorithme en fonction des cas.

Pour illustrer les forces et les faiblesses de l'algorithme UFit, je présente ici de nombreux cas, essentiellement classés par le nombre de traces de particules qui les compose. On commencera par des traces uniques pour aller jusqu'aux limitations de l'algorithme. Tout d'abord, on constate que Ufit discrimine correctement les traces simples comme le montre la figure 7.24. On voit que l'algorithme Φ_{circle} fonctionne parfaitement même lorsque le cercle est incomplet. Pour ces cas très simples, le temps de calcul est réduit à 2 appels à l'algorithme GEM : l'un pour un *fit* linéaire, l'autre pour un *fit* circulaire. Dès que l'objet correspondant a été déclaré "à l'échelle", le *fit* s'arrête. Le temps de calcul est de l'ordre de 5 millisecondes tout compris.

Lorsque deux traces de natures différentes se mélangent, l'algorithme arrive à discriminer cor-

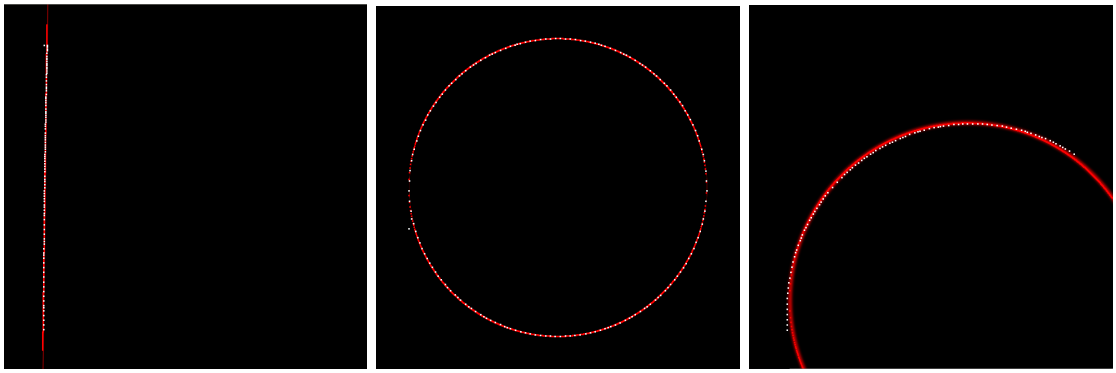


FIGURE 7.24 – Ufit sur une trace

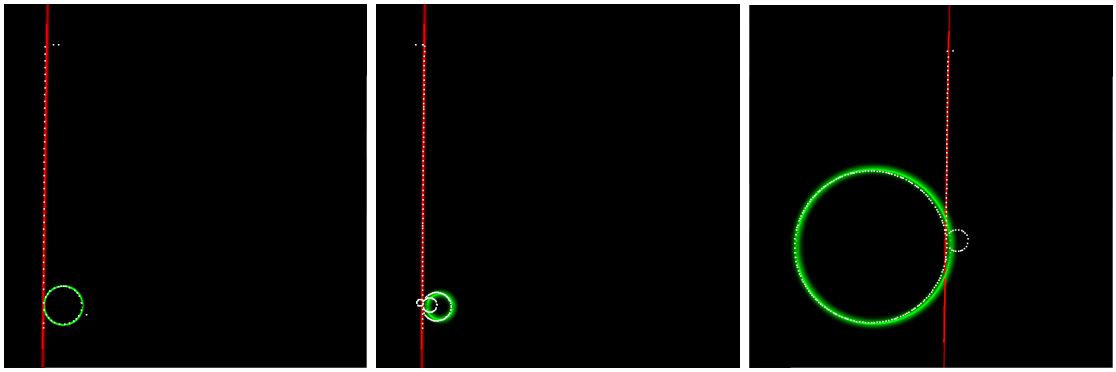


FIGURE 7.25 – Ufit sur deux traces différentes

rectement les objets. Par contre, il va ignorer les objets qui sont à une échelle nettement plus petite que les objets principaux. C'est logique car les objets sont déclarés à l'échelle sur la base de la taille globale du dessin. Par conséquent, les objets qui sont typiquement de la taille d'une fluctuation d'un plus gros objet seront ignorés. Pour les cas à deux traces, l'algorithme GEM est appelé 6 fois au maximum (C,L,CC,CL,LC,LL). Les résultats sont visibles sur la figure 7.25.

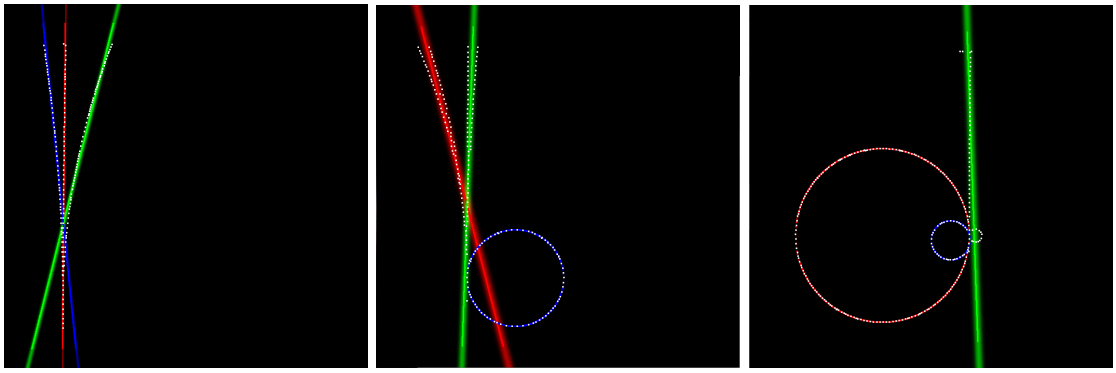


FIGURE 7.26 – Ufit sur trois traces différentes

L'algorithme donne sa pleine puissance sur les cas entre 3 et 5 traces. Dans ce cas, le nombre d'appel à GEM reste encore raisonnable (14 appels pour 3 traces, 30 pour 4 traces et 62 pour 5 traces) et le pouvoir de discrimination de l'algorithme est très bon. Comme on le voit sur la figure 7.26, les trajectoires principales sont ajustées correctement (toujours en ignorant les plus petites structures). On voit sur le plot central que les traces droites ont commencé à gerber

au deux tiers de l'image. Mais comme la largeur de cette gerbe est encore de la largeur de la distribution, la dissociation n'est pas effectuée. Si les traces étaient plus longues, elles finiraient par s'éloigner suffisamment l'une de l'autre pour sortir de la zone d'erreur autour de la droite. On voit ce cas sur le plot de gauche.

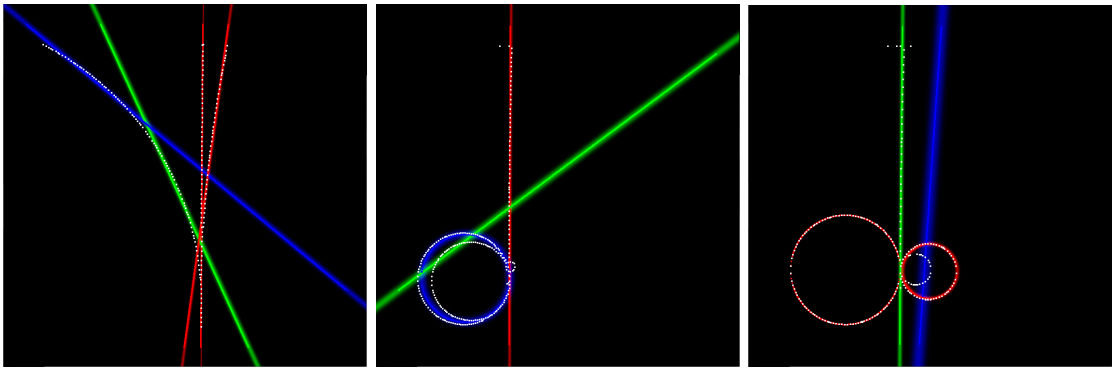


FIGURE 7.27 – Cas où Ufit privilégie à tort les droites sur les cercles

Dans certains cas, UFit va échouer à voir un cercle car ses points sont déjà en partie pris en charge par un autre objet mais trop de points restent seuls pour qu'il ne rajoute pas un objet. Dans ce cas, c'est souvent une droite qui est choisie. Comme on le voit sur la figure 7.27, le grand cercle à gauche est *fitté* par deux droites car son rayon est grand et que ses points les plus bas sont déjà *fittés* par les droite centrale. Au centre, le cercle inscrit dans l'autre cercle est mal *fitté* pour la même raison. Une partie de ses points sont communs avec le cercle externe. Ceux qui restent sont (mal) *fittés* par une droite. Le cas de droite est sensiblement équivalent.

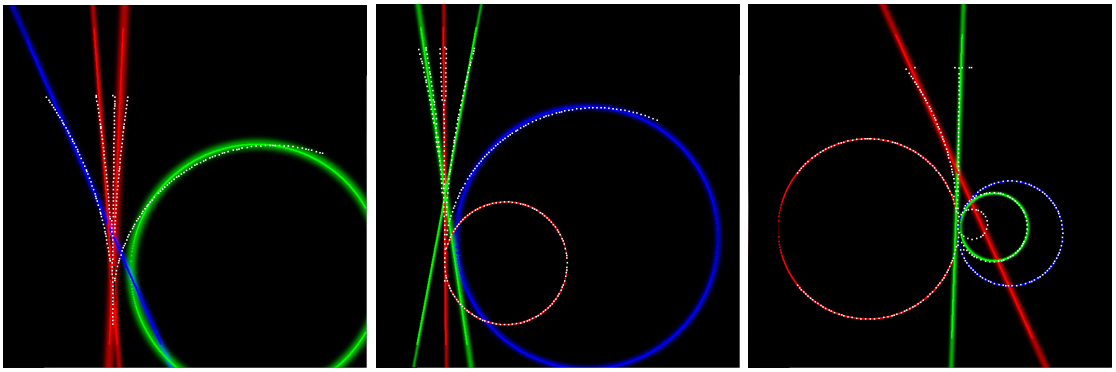


FIGURE 7.28 – Ufit sur 4 ou 5 traces différentes

L'algorithme arrive à discriminer 5 trajectoires en même temps comme le montre la figure 7.28. On voit que les traces gerbées mais non encore complètement séparées sont considérées comme des droites simples. On voit aussi que les 2 grands cercles vert et bleu à gauche et au centre, ont un diamètre légèrement plus petit que la trajectoire de la particule. Une fois de plus, cela s'explique par le fait que les points les plus proches du bloc de tungstène sont pris en charge par les traces droites.

Enfin, il existe certains cas plus complexes, comme celui représenté par la figure 7.29. On voit dans ce cas, que l'algorithme sélectionne correctement certaines trajectoires mais que les autres sont trop mélangées pour être détectées correctement. En outre, il faut parfois plusieurs secondes pour *fitter* une telle image car le nombre d'objets et la profondeur de l'arbre d'exploration crée une explosion combinatoire. Dans une utilisation *online*, il est nécessaire de limiter la profondeur

de l'arbre de recherche afin d'éviter que ces cas pathologiques ne viennent écrouler la performance globale de la reconstruction.

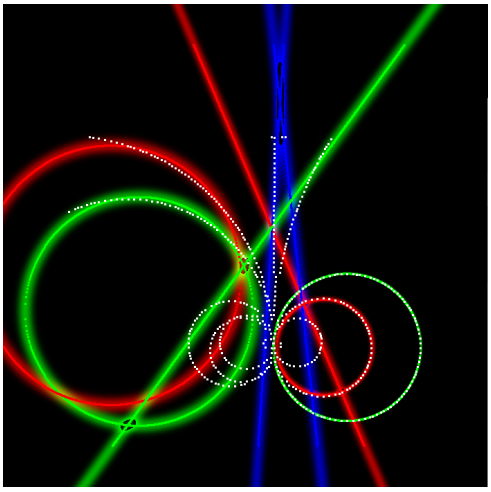


FIGURE 7.29 – Cas trop complexe pour l'algorithme

Comme on l'a vu, l'algorithme UFit arrive à *fitter* de nombreux cas, n'échouant que dans des cas spéciaux dont la probabilité d'occurrence est faible. Il peut donc être utilisé pour une reconstruction *online*, notamment pour éliminer les loopers, ces particules de faibles énergie qui tournoient dans le champ magnétique en générant de nombreuses traces qui perturbent l'analyse.

7.12 *Fit* projectifs

On peut également utiliser les fits de base (droites et cercles) dans une forme combinée par une projection. C'est typiquement le cas pour une spirale dont le sens de propagation ne coïncide pas avec l'axe longitudinal du détecteur. En effet, une spirale est une suite de points dirigés par une droite porteuse, distribués circulairement autour de cette droite avec une vitesse angulaire à peu près constante. La figure 7.30 représente le nuage de points qui nous servira d'exemple dans cette section.

L'idée du *fit* spiral est de trouver une droite que nous appellerons porteuse pour nos points. C'est la droite qui est à peu près équidistante de tous les points. Cette droite doit être *fittée* avec précision car toutes les autres opérations vont en dépendre. Pour obtenir cette droite, on va simplement utiliser l'algorithme GEM en 3 dimensions avec une seule droite à *fitter*. Cela revient à une simple régression linéaire dans l'espace.

On peut voir sur la figure 7.31 le résultat de la régression suivant deux angles. On constate que si les points sont représenté selon l'axe de la droite, ils forment un cercle en projection sur l'écran.

Nous allons utiliser cette technique pour obtenir les points en 2 dimensions afin de *fitter* le cercle correspondant. Pour cela, on va construire une base de projection, orthonormée et dont la droite porte un vecteur avec l'algorithme de Gram-Schmidt. On peut voir cette base sur la figure 7.31, ce sont les étoiles bleues.

Une fois cette base obtenue, on projette tous les points de la spirale sur le plan formé des deux vecteurs perpendiculaires à la droite. Dans ce plan, les points forment un cercle, comme nous l'avons constaté visuellement. On utilise alors l'algorithme GEM pour obtenir les paramètres

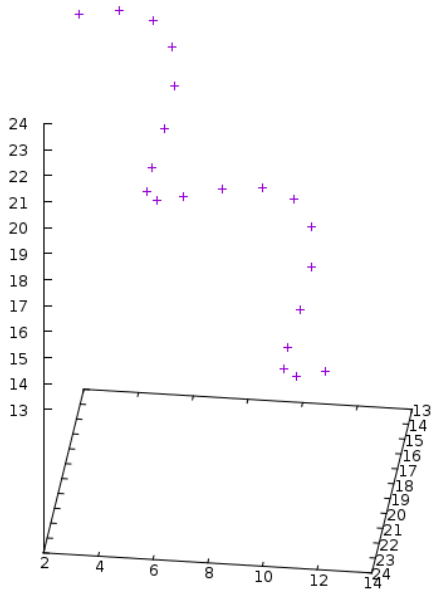


FIGURE 7.30 – Trace spiralée à *fitter*

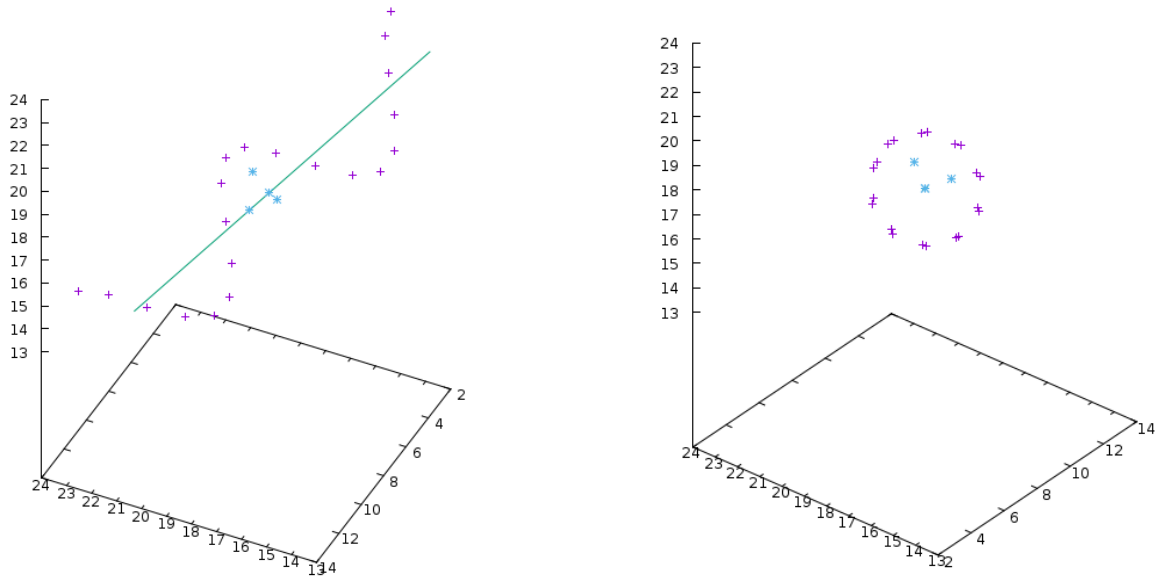


FIGURE 7.31 – A gauche, La droite porteuse et sa base de Gram-Schmidt, calculées à partir des points de la spirale (en rose). A droite, la même figure vue dans l'axe longitudinal de la spirale.

du *fit* circulaire des points. Comme on le voit sur la figure 7.32, on obtient les coordonnées du centre du cercle dans le repère de Gram-Schmidt et le rayon.

L'étape suivante est le calcul de la vitesse angulaire et de l'offset angulaire de la spirale. Pour les calculer, on commence par mesurer l'angle que les points projetés font avec l'axe des x du repère projectif. On mesure alors une vitesse basée sur cet angle et la coordonnée longitudinale de chaque point. On obtient une valeur de la vitesse pour chaque point et on en prend la moyenne. Il n'y a plus qu'à calculer l'angle au niveau du zéro du repère et l'on obtient l'offset angulaire.

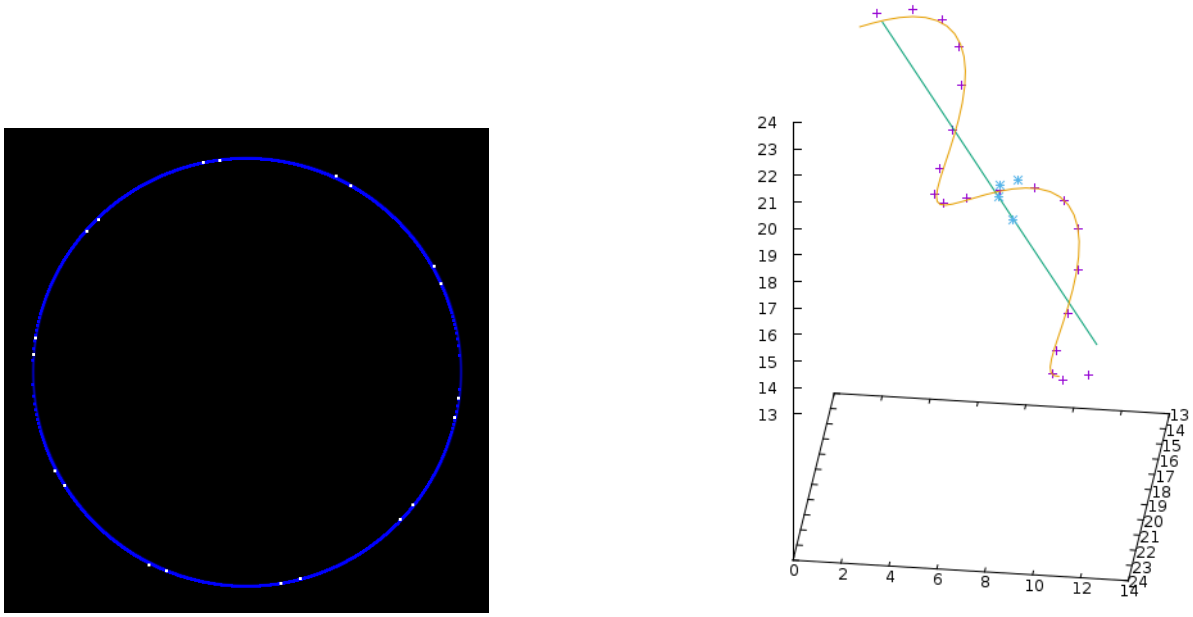


FIGURE 7.32 – A gauche, la projection des points dans la base de Gram-Schmidt et leur *fit* circulaire. A droite, le résultat final : la spirale *fittée* (en jaune) après calcul de la vitesse et de l'offset angulaire.

Une fois que tous les éléments sont réunis, on peut calculer tous les points de la spirale et l'afficher avec les points comme sur la figure 7.32 (en jaune).

Toutes les étapes décrites ici sont regroupées dans un algorithme appelé SpFit et qui est implémenté dans la libgem. Un travail est en cours pour pouvoir faire des fits multiples de spirales.

Chapitre 8

Conclusion et Perspectives

J'espère avoir montré aux travers de ces quelques pages, à quel point la problématique du traitement *online* couvre de nombreux champs applicatifs et nécessite des techniques précises et coordonnées. Ainsi, la gestion de la configuration ou le pilotage des équipements matériels sont en réalité indissociables du traitement qui est effectué en temps-réel sur les données correspondantes car leur paramètres font partie intégrante d'une telle analyse. La chaîne de programmes *online* toute entière doit être implémentée dans une logique de synergie afin d'aboutir à un traitement de bonne qualité. Les nombreux problèmes qui se posent, comme la contrainte temps-réel, doivent recevoir une réponse algorithmique adaptée.

Pyrame peut naturellement aider à atteindre un tel objectif. Les nombreuses réalisations qu'il a rendu possible démontrent bien l'intérêt d'avoir un tel *framework* généraliste, unifié et souple. Outre les réalisations simples qui sont rendues extrêmement faciles, les expériences plus complexes ont montré la pertinence de la démarche "du banc-test au détecteur final". Aujourd'hui la seule limitation de Pyrame est la vitesse de son acquisition qui devra être répartie sur plusieurs machines pour soutenir le rythme des collisions sur une expérience à haute fréquence. Cette répartition pose d'autres problèmes algorithmiques intéressants qui seront développés dans le futur de Pyrame.

La version 3 de Pyrame représente une certaine forme d'aboutissement en terme d'architecture logicielle. Celle-ci a intégré de nombreuses améliorations structurelles par rapport à la version 2. Naturellement, un tel développement se doit de toujours rester dynamique pour intégrer les nouveaux besoins amenés par les nouvelles expériences. Ainsi il est probable que de nombreux modules seront ajoutés au *framework* mais sans remettre en cause ses concepts fondamentaux.

Pyrame a également su séduire plusieurs expériences et se diffuse aujourd'hui au delà des frontières du laboratoire. C'est un véritable défi de faire vivre le logiciel dans un contexte de développement éclaté. Des besoins d'animation de communauté commencent à voir le jour, pour permettre de partager les connaissances et les développements pour aboutir à un logiciel de meilleure qualité, poussant sa logique de généricité pour répondre aux besoins de tous. Au travers des développements spécifiques, une politique d'intégration est nécessaire pour assimiler les spécificités des projets dans le *framework* sans en menacer l'intégrité globale.

Le développement de Calicoes, lui aussi, va bénéficier de son utilisation multiple. Par exemple, les nombreux programmes de monitoring implémentés pour *SiW-Ecal* devraient pouvoir être mis à disposition des autres projets au travers d'une librairie de briques de base qui serait facilement ré-utilisable pour construitre de nouveau moniteurs. Son interface graphique pourrait

également devenir un système partagé par toutes les expériences, une fois qu'il aura été rendu plus générique.

D'autre part, Calicoes est également pressenti pour devenir le logiciel d'acquisition pour d'autres électroniques d'acquisition que la *DAQ* générique du LLR. Par exemple, pour la prochaine version de CMS, de nouvelles cartes sont développées et leur test nécessite une daq flexible et facile à mettre en oeuvre : Calicoes est le candidat sélectionné. En conséquence, il devra intégrer de nouveaux drivers, comme par exemple les châssis μTCA, que le CERN a choisi pour devenir le nouveau standard de bus sur ses expériences. Les spécificités de notre *DAQ* actuelle devront être migrés complètement vers ces modules dédiés.

Du côté de Harpo, le bloc de surveillance des alimentations pourrait aussi être mutualisé. Toutes les expériences qui utilisent des hautes tensions aiment à surveiller les courants, pour des raisons différentes. Par exemple, sur *SiW-Ecal*, nous voulons monitorer le courant de fuite de nos *wafers* qui est un indicateur du niveau de bruit. Ce module devra donc être intégré à Pyrame comme un module générique, pret à intégrer les besoins des autres expériences.

Harpo lui-même, est succeptible de connaitre des développements ultérieurs. La nouvelle version St3g, conçue pour voler en ballon, si elle est financée, devra intégrer de nombreux aspects de monitoring et de pilotage supplémentaire des équipements. Il sera sans doute nécessaire de pousser la logique de coordination des configurations et des acquisitions de données pour que le controleur embarqué puisse prendre des décisions pertinentes même s'il se retrouve déconnecté du contrôleur au sol.

Le *SiW-Ecal*, lui aussi, devrait amener son lot d'évolution. Les différents prototypes imposent des fonctionnements tous légèrement différents. La version compacte de la *DAQ* impose de développer des nouvelles cartes qu'il faudra intégrer. Si la construction de l'ILD est décidée, il faudra alors optimiser la version actuelle pour lui permettre d'atteindre des performances en accord avec les besoins d'une telle expérience. Des défis de parallélisation apparaitront alors et avec eux des problématique de sureté de fonctionnement.

Du côté de la trajectographie, j'espère que les études que j'ai réalisées vous aurons convaincu du bien fondé d'une approche exploratoire en matière algorithmique. De nombreux algorithmes intéressants existent au sein de la communauté informatique qui ne demandent qu'à être adaptés à nos problématiques de physique des particules et qui constituent une amélioration sensible de la boite à outils qui est à notre disposition. Cette approche trans-disciplinaire, même si elle est parfois ardue, est une manne d'outils innovants et pertinents.

L'algorithme MFit lui-même peut être amélioré de plusieurs façon. En particulier, j'envisage d'en faire une version qui manipule, non plus des droites, mais des segments de façon à pouvoir étudier les points d'interconnexion. Cette approche nous donnera, j'en suis convaincu, un outil pertinent pour la reconstruction des gerbes électromagnétiques et hadroniques, par une caractérisation complète de la gerbe, particule par particule.

D'autres formes m'intéressent particulièrement. Par exemple les ellipses qui sont très souvent utilisées pour qualifier les jets hadroniques. Leur proximité géométiques avec les cercles donne des pistes pour les apréhender. Les possibilités d'autres extensions geométriques sont quasiment sans borne.

Une autre approche que je souhaite aborder est la segmentation du problème. Il est fréquent qu'une collision soit un espace très grand qui induit un très grand nombre de points à traiter alors que de nombreuses composantes non connexes représentent des objets très différents. Le

découpage de l'espace en angles solides, puis un traitement parallèle pourrait fournir des primitives à un algorithme d'agrégation qui les assemblerait en une vision cohérente. Cette approche algorithmique (scatter/gather) est connue pour donner des résultats intéressant et pourrait sans doute apporter des éléments de réponses sur nos problématiques de pile-up.

Du côté de Wagasci, nous continuerons de construire le détecteur qui est aujourd'hui dans une forme préliminaire. Ainsi, les espaces de *fit* vont augmenter et se complexifier. L'algorithme de reconstruction tel qu'il est envisagé aujourd'hui cherche des vertex et des trajectoires dans le module central puis cherche à les ajuster avec celles des MRDs. Je pense qu'avec MFit, nous pouvons envisager de traiter tous les points ensemble. Grâce aux convertisseurs *online* de Pyrame et à la performance de MFit, nous pouvons également envisager de passer d'une reconstruction offline, comme c'est le cas aujourd'hui, à une reconstruction *online* permettant un monitoring avancé en temps-réel.

Enfin, les algorithmes adaptatifs sont aujourd'hui en pleine révolution. Les volumes de données qui sont les notres aujourd'hui imposent des algorithmes toujours plus performants. Or, il est possible, par les techniques de machine learning, d'entrainer un système de façon à lui faire approximer un problème complexe avec des temps d'exécution très faibles. Les réseaux de neurones artificiels sont certainement l'une des voies les plus prometteuses de ce domaine et je compte les intégrer dans Pyrame sous forme d'outils d'aide à l'analyse *online* par exemple pour effectuer des coupures impossible à utiliser aujourd'hui à cause de la complexité de l'algorithme. La version "apprise" par le réseau de neurone remplacera l'algorithme original, nous offrant la performance nécessaire pour un traitement *online*. Ainsi, je travaille actuellement pour faire une version neuronale de l'algorithme MFit, fondamentalement itératif, qui pourra être remplacé par sa version approximée à temps de calcul constant.

Cette démarche sera particulièrement féconde pour les triggers. Ainsi dans le cadre du futur calorimètre HGCal pour le LHC à haute luminosité, nous développons actuellement une plateforme permettant d'entrainer et de sélectionner des architectures neuronales adaptées aux problématiques de clusterization et de sélection des évènements. Dans un deuxième temps, ces réseaux seront intégrés dans l'électronique de trigger de niveau 1 au travers d'une librairie permettant d'implémenter des neurones artificiels dans des composants reconfigurables (FPGA optimisés pour le deep-learning).

Cette convergence de l'électronique et de l'informatique, couplé aux changements induits par l'omniprésence des processeurs dans tous les composants (RAM, GPU, disques, réseaux...) va modifier radicalement notre façon d'intégrer les algorithmes dans les détecteurs de particules. En particulier, le traitement online va énormément bénéficier de cet apport de puissance de calcul réparti. Cela arrive d'ailleurs fort opportunément au moment où la luminosité et le nombre de canauxd de nos détecteurs explosent, rendant beaucoup plus complexes les post-traitements et même le stockage. Naturellement, cette répartition des calculs posera de nombreux défis techniques qui nécessiteront des outils algorithmiques de dernière génération, des langages nouveaux et des techniques de vérification réparties. Tout cela justifie pleinement que des efforts soient consentis par la communauté pour adapter nos techniques dès à présent et rester en pointe pour construire les détecteurs du futur.

Bibliographie

[1] M. Anduze, A. Bonnemaison, V. Boudry, and M. Frotin. Utilisation de capteurs à réseaux de Bragg pour la mesure in-situ de déformée d'une structure alvéolaire élaborée en matériaux composite. In *7ème Colloque Interdisciplinaire en Instrumentation*, Saint-Nazaire, France, January 2016. URL `https://hal.archives-ouvertes.fr/hal-01280213`.

[2] C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics and Data Analysis*, 41(1) :561–575, 2003.

[3] B Bilki, J Butler, G Mavromanolakis, E May, E Norbeck, J Repond, D Underwood, L Xia, and Q Zhang. Hadron showers in a digital hadron calorimeter. *Journal of Instrumentation*, 4(10) :P10008, 2009.

[4] Patrick Billingsley. Convergence of probability measures, 2e ed. *Wiley series in probability and statistics*, page 16, 1999.

[5] R. Brun and F. Rademakers. ROOT - An Object Oriented Data Analysis Framework. *Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. and Meth. in Phys. Res.*, A 389 :81–86, 1997. URL `http://root.cern.ch/`.

[6] S. Callier, F. Dulucq, C. de La Taille, G. Martin-Chassard, and N. Seguin-Moreau. Skiroc2, front end chip designed to readout the electromagnetic calorimeter at the ilc. *Journal of Instrumentation Volume 6 (JINST 6)*, C12040, 2011.

[7] ATLAS collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3 :S08003, 2008. doi : 10.1088/1748-0221/3/08/S08003.

[8] Calice collaboration. Calice web site. 2004. URL `https://twiki.cern.ch/twiki/bin/view/CALICE/CaliceCollaboration`.

[9] CMS collaboration. CMS, the Compact Muon Solenoid : Technical proposal. 1994.

[10] CMS collaboration. *The CMS electromagnetic calorimeter project : Technical Design Report*. Technical Design Report CMS. CERN, 1997. URL `http://cds.cern.ch/record/349375`.

[11] CMS collaboration. The CMS Experiment at the CERN LHC. *JINST*, 3 :S08004, 2008. doi : 10.1088/1748-0221/3/08/S08004.

[12] CMS Collaboration. Observation of a new boson with mass near 125 gev in pp collisions at $\sqrt{s} = 7$ and 8 tev. *Journal of High Energy Physics*, 2013(6) :81, Jun 2013. doi : 10.1007/JHEP06(2013)081.

[13] ILC collaboration. The international linear collider technical design report - volume 4 : Detectors. *arxiv :1306.6329.*, 2013.

[14] R. Cornat, F. Gastaldi, and F. Magniette. Acquisition and control command system for power pulsed detectors. *Journal of Instrumentation Volume 9 (JINST 9)*, 2013.

[15] N. Daci. CMS level-1 electron/photon trigger performance. *XXIst International Europhysics Conference on High Energy Physics*, 2011.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1 :1–38, 1977.

[17] R.O. Duda and P.E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM 15*, 1972.

[18] B. Aune et al. Superconducting TESLA cavities. *Physical Review Special Topics - Accelerators and Beams, Volume 3*, 2000.

[19] G. Baulieu et al. Construction and commissioning of a technological prototype of a high-granularity semi-digital hadronic calorimeter. *Journal of Instrumentation*, 10(10) :P10039, 2015.

[20] J.Repond et al. Design and electronics commissioning of the physics prototype of a si-w electromagnetic calorimeter for the international linear collider. *JINST 3 P08001, arXiv :0805.4833.*, 2008.

[21] M. Jeitler et al. The level-1 global trigger for the cms experiment at lhc. *JINST*, 2(01) : P01006, 2007.

[22] N. Chikuma et al. Development of electronics and data acquisition system for the J-PARC T59 (WAGASCI) experiment. *PoS*, EPS-HEP2017 :780, 2017. doi : 10.22323/1.314.0780.

[23] V. Andreev et al. A high granularity scintillator hadronic-calorimeter with SiPM readout for a linear collider detector. *Nucl. Instrum. Meth.*, A540 :368–380, 2005. doi : 10.1016/j.nima.2004.12.002.

[24] V. Brigljevic et al. Using xdaq in application scenarios of the cms experiment. *CHEP-2003-MOGT008*, 2003.

[25] Y. Giomataris et al. MICROMEGAS : A high granularity position sensitive gaseous detector for high particle flux environments. *Nucl. Instr. and Meth. A*, 376 :29, 1996.

[26] Lyndon Evans and Philip Bryant. LHC Machine. *JINST*, 3 :S08001, 2008. doi : 10.1088/1748-0221/3/08/S08001.

[27] R. Frühwirth. Application of kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, 262(2) :444 – 450, 1987. ISSN 0168-9002. doi : https://doi.org/10.1016/0168-9002(87)90887-4. URL `http://www.sciencedirect.com/science/article/pii/0168900287908874`.

[28] A. Galántai. The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1) :25 – 44, 2000. ISSN 0377-0427. doi : https://doi.org/10.1016/S0377-0427(00)00435-0. URL `http://www.sciencedirect.com/science/`

article/pii/S0377042700004350. Numerical Analysis 2000. Vol. IV : Optimization and Nonlinear Equations.

[29] F. Gastaldi, R. Cornat, F. Magniette, and V. Boudry. A scalable gigabit data acquisition system for calorimeters for linear collider. *Technology and Instrumentation in Particle Physics*, 2014.

[30] P. Gros, S. Amano, D. Attié, P. Baron, D. Baudin, D. Bernard, P. Bruel, D. Calvet, P. Colas, S. Daté, A. Delbart, M. Frotin, Y. Geerebaert, B. Giebels, D. Götz, S. Hashimoto, D. Horan, T. Kotaka, M. Louzir, F. Magniette, Y. Minamiyama, S. Miyamoto, H. Ohkuma, P. Poilleux, I. Semeniouk, P. Sizun, A. Takemoto, M. Yamaguchi, R. Yonamine, and S. Wang. Performance measurement of HARPO : a time projection chamber as a gamma-ray telescope and polarimeter. 2017. URL https://arxiv.org/abs/1706.06483.

[31] A. S. Hassanein, S. Mohammad, M. Sameer, and M. E. Ragab. A Survey on Hough Transform, Theory, Techniques and Applications. *CoRR*, abs/1502.02160, 2015. URL http://arxiv.org/abs/1502.02160.

[32] H.Videau and J.C. Brient. A Si-W calorimeter for linear collider physics. *Proceedings of the 10th International Conference on Calorimetry in High energy Physics (CALOR 2002)*, pages 309–320, 2002.

[33] ILC collaboration. The International Linear Collider Technical Design Report. Volume 1 (Executive Summary), Volume 2 (Physics), Volume 3 (Accelerator), Volume 4 (Detectors), 2013. URL https://www.linearcollider.org/ILC/Publications/Technical-Design-Report.

[34] F. Magniette. Statistical algorithms for particle trajectography. *Computer Physics Communications*, 232 :59 – 70, 2018. ISSN 0010-4655. doi : https://doi.org/10.1016/j.cpc.2018.05.023.

[35] F. Magniette and A. Irles. Pyrame 3, an online framework for Calice SiW-Ecal. *Journal of Instrumentation*, 13(03) :C03009, 2018. URL http://stacks.iop.org/1748-0221/13/i=03/a=C03009.

[36] F. Magniette, M. Rubio-Roy, and F. Thiant. Pyrame, a rapid-prototyping framework for online systems. *Journal of Physics, Conference Series*, Vol 664, 2015.

[37] Pascal Paganini. CMS electromagnetic trigger commissioning and first operation experiences. *J. Phys. Conf. Ser.*, 160 :012062, 2009. doi : 10.1088/1742-6596/160/1/012062.

[38] Pedro Parracho. SEU handling in ecal. *CMS-doc-5930-v9*, 2013.

[39] M. Rubio-Roy, F. Thiant, and F. Magniette. Flexible online monitoring for high-energy physics with Pyrame. *Journal of Physics, Conference Series*, Vol 898, 2016.

[40] G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2) :461–464, 1978.

[41] Floris Thiant. New development in the cms ecal level-1 trigger system to meet the challenges of lhc run 2. *Topical Workshop on Electronics for Particle Physics 2018*, 2018.

[42] M.A. Thomson. Particle flow calorimetry and the PandoraPFA algorithm. *NIM A611, arXiv :0907.3577.*, page 25, 2009.

[43] T. R. Turner. Mixreg. 1998. URL https://cran.r-project.org/web/packages/mixreg.

[44] T. R. Turner. Estimating the rate of spread of a viral infection of potato plants via mixtures of regressions. *Applied statistics*, Vol 49, Part 3 :371–384, 2000.

[45] Satoru Uozumi. Performance of the scintillator-strip electromagnetic calorimeter prototype for the linear collider experiment. 293 :012070, 05 2011.

[46] J.C. Valognes, J.P. Bardet, and P. Mergault. Contribution à l'étude des effets d'électrodes. *Spectrochimica Acta*, Vol. 42B, Num. 3 :445–458, 1987.

[47] N Watson, J Ballin, J Crooks, P Dauncey, A-M Magnan, Y Mikami, O Miller, M Noy, M Stanitzki, K Stefanov, R Turchetta, M Tyndel, V Rajovic, E Villani, and J Wilson. A maps-based readout of an electromagnetic calorimeter for the ilc. *Journal of Physics : Conference Series*, 110(9) :092035, 2008. URL http://stacks.iop.org/1742-6596/110/i=9/a=092035.

[48] B. Wojtsekhowski, D. Tedeschi, and B. Vlahovic. A pair polarimeter for linearly polarized high-energy photons. *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, 515(3) :605 – 613, 2003. ISSN 0168-9002. doi : https://doi.org/10.1016/j.nima.2003.07.009. URL http://www.sciencedirect.com/science/article/pii/S0168900203023143.

Lexique

API : acronyme de Application Programming Interface, désigne la façon d'invoquer une fonction ou une méthode (nom, paramètres, protocole...)

ASIC : acronyme de Application Specific integrated circuit. Composant électronique dédié, par extension, composant responsable de la lecture des surfaces de détection des détecteurs.

ASU : carte de détection des expériences Wagasci et SiW-Ecal. L'ASU porte les ASICs de lecture et une connection vers la carte adaptatrice SMB.

Binding : ensemble de fonctions permettant d'acceder aux fonctionalités d'un système. Un binding est spécifique à un langage de programmation (langage source) et à un système cible.

Bitstream : suite de bits utilisé pour la configuration de composants électroniques.

Calorimètre : dispositif expérimental permettant de mesurer l'énergie des particules incidentes. Les particules sont arretées par le calorimètre et le dépot d'énergie correspondant permet d'extrapoler l'énergie incidente. On trouve des calorimètres électromagnétiques, pour les particules légères (photons, électrons) et des calorimètres hadroniques pour les particules plus lourdes.

CERN : Organisation européenne pour la physique nucléaire, infrastructure de recherche situé près de Genève regroupant de nombreuses installations d'accélération et de détection de particules.

DAQ : acronyme de Data Acquisition System, système (matériel et/ou logiciel) d'acquisition de données

DESY : acronyme de Deutshes Elektronen-Synchrotron, infrastructure de recherche en physique des particules, située à Hambourg en Allemagne.

DIF : acronyme de Detector InterFace, carte de lecture de la DAQ générique du LLR

Dispatcher (de données) : mécanisme de Pyrame permettant de mettre à disposition des données en temps-réel.

DCC : acronyme de Data Concentrator Card, carte d'aggrégation de données de premier niveau de la DAQ générique du LLR

Elog : acronyme de Electronic Logbook. Livre électronique partagé, accessible par le web et permettant de mettre en commun des déroulés d'opérations et des informations pour une expérience.

Fit : en français ajustement, technique qui consiste à estimer les paramètres d'un modèle de forme analytique à partir d'un ensemble de mesures. Dans ce mémoire, il est surtout question de fit de trajectoire pour lesquels les mesures sont des points géographiques et le modèle est un type de trajectoire (linéaire, circulaire...).

Framework : ensemble cohérent de composants logiciels, qui servent à créer les fondations d'un logiciel.

GDCC : acronyme de Gigabit Data concentrator card, carte d'aggrégation de données de deuxième niveau de la DAQ générique du LLR

ILC : acronyme d'International Linear Collider. Futur collisionneur linéaire décrit au chapitre 5.

ILD : acronyme d'International Large Detector. L'un des futurs détecteurs de l'ILC.

MIP : particule au minimum d'ionisation. Dépot d'énergie correspondant à une particule traversant le milieu de détection.

Online : qualifie les opérations et les traitements qui sont effectuées pendant la collecte des données par le détecteur. Le terme de temp-réel, bien que moins général, peut également être utilisé. En opposition à offline, qui qualifie les traitements apportés à posteriori aux données collectées.

Overhead : surcoût de performance, en général imputable aux opérations incompressibles (transmission de données, décodage...).

PCB : acronyme de Printed Circuit Board, synonyme de circuit imprimé ou carte électronique.

Shift : Période d'utilisation d'un détecteur. Le shifter est la personne responsable de l'instrument pendant le shift.

SiW-Ecal : calorimètre électromagnétique développé au LLR pour le futur détecteur ILD du futur collisionneur linéaire ILC.

Slab : structure de détection du *SiW-Ecal* formé d'une couche de détection et d'une couche d'absorbeur

SMB : acronyme de Sweet Mezzanine Board, carte adapatatrice de la DAQ générique du LLR

Sub-sampling : en français sous-échantillonnage, technique de traitement *online* consistant à analyser en temps-réel une fraction des données collectées

Test faisceau : lors d'une phase de développement d'un détecteur de particule, on le place régulièrement dans un faisceau de particules issues d'un accélérateur. Ces particules doivent être parfaitement connue en énergie et en direction afin de pouvoir caractériser le nouveau détecteur.

TPC : Chambre à dérive, en anglais Time Projection Chamber. C'est un dispositif expérimental constitué d'une chambre rempli d'un gaz. Lors du passage d'une particule, celle-ci ionise le gaz et un champ électrique fait migrer ces charges pour les mesurer. Ces détecteurs permettent une précision spatiale inégalée (de l'ordre de 150 μm. Par contre, la migration des charges est un processus lent qui limite ses performances temporelles.

Trajectographe : en anglais Tracker, est un dispositif expérimental qui permet de mesurer la trajectoire d'une particule en la pertubant au minimum. Des algorithmes spécifiques de trajectographie sont nécessaires pour inférer la trajectoire à partir des mesures effectuées par le trajectographe.

Trigger : système de déclenchement d'un détecteur de particule. C'est un système qui permet de déclencher l'enregistrement des données lorsqu'un évènement intéressant se produit. On peut trouver des triggers internes intégrés dans l'électronique de lecture ou des triggers externes matériels ou logiciels.

Wafer : galette de silicium utilisée comme surface de détection

Remerciements

Je souhaite tout d'abord remercier les membres de mon jury qui ont accepté de m'accompagner dans ce rite de passage qu'est l'habilitation à diriger les recherches. Merci pour vos rapports bienveillants et pour la riche discussion scientifique que vous m'avez offerte.

Je souhaite également remercier tous ceux qui m'ont permis de rejoindre le LLR, Jean-Claude, Marc, David et Rémi. Ils m'ont accordé leur confiance et m'ont offert un cadre qui m'a permis d'exprimer pleinement ma créativité.

Je voudrais remercier tous les physiciens avec qui j'ai eu le plaisir de travailler, Jean-Claude, Henri, Vincent, Vladislav, Denis, Philippe, Marc, Christophe, Thomas, Olivier, Yves, Alexandre, Jean-Baptiste, Florian, Bruno et Artur. Ils m'ont offert leur savoir avec une grande générosité et m'ont permis de collaborer dans leurs projets passionnants. Merci de m'avoir transmis votre enthousiasme.

Je voudrais également remercier tous mes collègues des services techniques et administratifs. J'apprécie vos grandes qualités professionnelles mais également votre gentillesse au quotidien. C'est un grand honneur de travailler parmi vous.

Merci à mes collègues du service électronique Franck, Jérôme, Rémi, Marc, Yannick, Thierry, Laura et Florence. Merci de m'avoir initié aux charmes du VHDL et de l'impédence de ligne. C'est un plaisir chaque jour renouvelé de travailler en synergie avec vous

Merci à mes collègues du service informatique, Emilia, Pascale, Andrea, Michel, Gilles, Arnaud, Igor, Julien, Arnaud et Michael. C'est un grand plaisir de travailler chaque jour avec vous, vous etes une équipe incroyablement talentueuse et j'espère que notre collaboration durera encore de nombreuses années.

Merci à mes collègues du service mécanique Julien, Mickael, Thomas, Antoine, Alain, Samy, Mathieu, Evelyne, Pascal et Mickael. Merci de m'aider à résoudre tous les problèmes avec compétence et efficacité.

Un remerciement particulier pour les membres du pôle online, Floris et Lorenzo mais également Miguel, parti vers d'autres sciences. Merci de votre contribution si précieuse et de m'accorder votre confiance pour la coordination du pôle.

Merci à mes collègues des services généraux et administratifs, Sylvaine, Alimata, Maïté, Stéphanie, Mélanie, Sandrine, Brigitte et Hamid. Merci de votre aide précieuse en toute circonstance.

Merci également à tous les autres, ceux qui m'ont offert leur collaboration ponctuelle et que je ne peux tous citer ici.

Je souhaite également remercier ma famille pour son soutien et en particulier mon épouse Marie-

Laure, dont la vision scientifique, très différente de la mienne mais très riche et inter-disciplinaire, nourrit fréquemment mes réflexions. Je voudrais également remercier mes deux enfants, Paul et Adrien, pour m'apporter chaque jour leur joie de vivre communicative, soutien si précieux dans un monde trop sérieux.

Je voudrais finir en rendant hommage à mon cher grand-père, Pierre Mergault (1922-1993), qui m'a montré la voie il y a fort longtemps. C'était un scientifique brillant, directeur de recherche au CNRS et directeur du laboratoire de physique des liquides ioniques de l'université Pierre et Marie Curie de Paris. Véritable pionnier, il avait compris à quel point l'informatique allait devenir un outil incontournable pour la physique moderne. Il a su l'intégrer sur ses expériences dès les années 70, sous la forme d'ordinateurs individuels dotés de convertisseurs analogique-digitaux, faisant preuve d'audace et d'avant-gardisme. Ses travaux sur les effets d'anode [46] ont démontré l'intérêt de la synergie, apportant l'explication d'un phénomène jusqu'alors incompris mais également des algorithmes innovants notamment en calcul matriciel. C'est lui qui m'a initié précocement à la physique et à l'algorithmique, et qui m'a donné le gout de la recherche scientifique qui me motive depuis tant d'années. Je lui en suis infiniment reconnaissant et je suis fier de suivre ses traces par delà le temps.